BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dalam berbagai bentuk dan media dapat tersebar dengan cepat. Namun, karena informasi dalam bentuk data digital, misalnya citra digital sangat mudah untuk dimodifikasi. Penyebaran data melalui internet juga memberikan kesempatan kepada pihak yang tidak berhak untuk membuat salinan tanpa izin dari pemilik sah, bahkan menyebarkannya untuk kepentingan komersial. Hal ini dapat menimbulkan persoalan hak cipta bagi citra digital yang tersebar. Salah satu cara yang dapat digunakan untuk melindungi hak cipta pada citra digital adalah dengan menyisipkan tanda hak cipta dengan metode *watermarking*.

Watermarking merupakan teknik penyisipan (*embedding*) informasi ke dalam data digital seperti citra, audio, dan video secara rahasia. Informasi yang disisipkan kemudian harus dapat diperoleh kembali meskipun data digital telah diproses, disalin, atau didistribusikan. Informasi yang akan disisipkan ke dalam data digital dinamakan tanda air digital (*digital watermark*), sedangkan data digital yang disisipi dinamakan data orisinal (*host data*). Untuk data digital yang telah disisipi *watermark* dinamakan data bertanda air (*watermarked data*) (Basaruddin, 2009). Pada tugas akhir ini, penyisipan *watermark* dilakukan pada data digital berupa citra gambar diam.

Dalam ilmu kriptografi, *watermark* termasuk dalam salah satu pengembangan steganografi. Dari steganografi ini kemudian lahirlah pemikiran-pemikiran baru, dimana salah satunya adalah *watermark.* Pada *watermark*, pesan yang disisipkan merupakan logo atau tanda khas dari pembuat media, sementara media yang disisipkan pesan tersebut adalah hasil karya pembuat media tersebut.

Berdasarkan metode pemrosesnya, *watermarking* dapat digolongkan menjadi dua bagian yaitu pada domain spasial dan domain transformasi. *Watermarking* yang bekerja pada domain spasial langsung mengubah nilai piksel pada citra asli. Metode tersebut memiliki kompleksitas komputasi yang rendah namun tidak tahan terhadap serangan dan *watermark* dapat mudah dihapuskan dari citra dengan mengganti semua bit pada *least significant bit* (LSB) pada citra. Sebaliknya teknik *watermarking* dalam domain transformasi seperti *Discrete Fourier Transform* (DFT), *Discrete*

Cosine Transform (DCT), *Discrete Wavelet Transform* (DWT), dan *Singular Value Decomposition* (SVD) memiliki lebih banyak keuntungan dan kinerja yang lebih baik daripada teknik yang bekerja pada domain spasial (Cahyana, 2007), salah satunya lebih tahan terhadap serangan. Pada tugas akhir ini, penyisipan *watermark* menggunakan metode *Singular Value Decomposition* (SVD).

Menurut Liu (2002), tiga karakteristik yang seharusnya dipenuhi oleh suatu teknik *watermarking* adalah *imperceptibility*, *robustness* dan *trustworthiness* (Yusuf, 2009). *Imperceptibility* menunjukkan antara citra asli dan citra ter-*watermark* secara persepsi tidak dapat dibedakan oleh mata manusia, *watermark* tidak mengalami interferensi dengan medianya. *Robustness* merupakan tingkat kekuatan *watermark* yang disisipkan terhadap serangan dan pemrosesan sinyal yang biasanya dilakukan pada suatu data digital seperti penambahan *noise*, proses *filter*, *scaling*, perputaran, pemotongan dan *lossy compression*. Sedangkan *trustworthiness* menjamin tidak akan dapat dibangkitkan *watermark* yang sama dengan *watermark* yang asli dan menyediakan bukti terpercaya untuk melindungi hak kepemilikan.

Teknik *watermark* dengan menggunakan SVD umumnya penyisipan dilakukan pada nilai-nilai singular berdasarkan pertimbangan bahwa nilai singular tidak akan mengalami perubahan signifikan jika terjadi sedikit gangguan pada citra (Basaruddin, 2009). SVD dikenal sebagai teknik yang sangat kuat, berkenaan dengan penyelesaian masalah persamaan atau matriks, baik singular maupun secara numerik mendekati singular. Sebuah keunggulan pada SVD adalah kemampuan untuk digunakan pada semua matriks real berukuran (m,n). Keunggulan lain dari SVD adalah ukuran matriks dari transformasi metode SVD tidak tetap dan dapat berupa matriks persegi atau lingkaran. Kemudian *singular value* mengandung informasi properti persamaan linear citra (Chang, 2005). SVD tergolong metode terbaru dalam *watermarking* dibanding pada penelitian-penelitian sebelumnya seperti DFT,DWT dan DCT.

Pada skripsi ini akan diteliti bagaimana penerapan proses *watermarking* pada citra digital menggunakan teknik *singular value decomposition* (SVD). Kelebihan dari skripsi ini adalah citra yang digunakan adalah citra berwarna dengan ukuran bebas. Penelitian sebelumnya yang ditunjukkan oleh Liu yaitu menggunakan citra

grayscale. Pada penelitian lainnya ditunjukkan juga bahwa citra yang digunakan dalam penelitian berukuran statis.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang permasalahan yang ada, maka rumusan masalah dari penelitian ini adalah:

1. Bagaimana proses *watermarking* pada citra digital menggunakan teknik SVD.
2. Bagaimana mengukur kualitas citra ter-*watermark* dengan citra asli (PSNR).
3. Bagaimana mengukur tingkat kemiripan citra *watermark* hasil ekstraksi dengan citra *watermark* asli (nilai korelasi) untuk otentifikasi.
4. Bagaimana ketahanan *watermark* terhadap serangan citra (*Blur effect, noise, pixelate, sharpening*).

1.3 Tujuan

Tujuan yang ingin dicapai dalam penulisan skripsi ini adalah :

1. Menghasilkan sebuah sistem yang digunakan untuk melindungi hak cipta dari suatu citra digital dengan cara menyisipkan *watermark* pada citra digital dan mengesktrak kembali *watermark* yang telah disisipkan menggunakan teknik *watermark singular value decomposition* (SVD).
2. mengetahui kualitas citra ter-*watermark* dibandingkan dengan citra asli (PSNR).
3. mengetahui apakah citra tersebut terdapat tanda air (*watermark*) atau tidak.
4. mendapatkan perbandingan ketahanan *watermark* terhadap serangan-serangan efek citra (*Blur effect, noise, pixelate, sharpening*).

1.4 Batasan Masalah

Batasan masalah bertujuan agar penelitian yang dilakukan tetap mengacu pada topik penelitian. Batasan masalah dalam penelitian ini adalah:

1. Input berupa file citra digital berwarna (format *.bmp).
2. File *watermark* berupa citra berwarna (format *.bmp) dan dimensinya <= dimensi citra asal.
3. Proses penyisipan *watermark* dan ekstraksi menggunakan teknik dekomposisi nilai singular (SVD).
4. Output berupa file citra digital yang telah ter-*watermark* dan file *watermark* hasil ekstraksi.
5. Proses ekstraksi menggunakan detektor *non-blind*, yaitu membutuhkan citra asli/*host* pada saat melakukan ekstraksi pada citra ter-*watermark*.

1.5 Manfaat

Manfaat yang didapatkan dari penyusunan Tugas Akhir ini adalah membantu melindungi hak cipta citra digital yang tersebar di internet agar tidak disalahgunakan oleh pihak tertentu untuk disalin ataupun kepentingan komersial.

1.6 Metodologi Penelitian

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur
   Mempelajari teori – teori yang berhubungan dengan konsep *watermarking* citra dan metode dari berbagai referensi.
2. Pendefinisian dan Analisis Masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan Implementasi Sistem
   Membuat rancangan model perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat piranti lunak *watermarking* citra digital menggunakan metode SVD.
4. Uji Coba dan Analisa Hasil Implementasi
   Menguji perangkat lunak dengan berbagai macam citra. Kemudian menganalisa hasil dari implementasi tersebut

sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori – teori yang berhubungan dengan citra, *watermarking* citra, metode SVD.
3. BAB III METODOLOGI PERANCANGAN
   Pada bab ini akan dijelaskan mengenai metode – metode yang digunakan dalam *watermarking* dengan SVD.
4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa model sistem perangkat lunak aplikasi *watermarking* menggunakan metode SVD.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

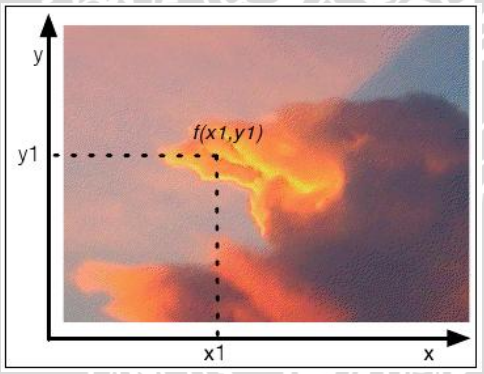

UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas beberapa konsep dasar yang berhubungan dengan algoritma watermarking digital menggunakan *singular value decomposition* (SVD), seperti definisi, *framework*, karakteristik dan klasifikasi teknik *watermarking* digital. Kemudian dilanjutkan dengan pembahasan singkat mengenai SVD.

2.1 Citra Digital

Menurut Balza Ahmad dan Kartika Firdausy (2005), citra dapat dikelompokkan menjadi citra tampak dan citra tidak tampak. Contoh dari citra tampak adalah foto, lukisan, dan sebagainya. Sedangkan citra tidak tampak misalnya citra digital. Citra digital dapat didefinisikan sebagai fungsi dua variabel f(x,y), dimana x dan y adalah koordinat spasial dan nilai f(x,y) adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan pada gambar 2.1.



Gambar 2.1 Citra Digital
Sumber: Suhendra, 2007

Citra digital MxN secara lengkap dapat ditulis dalam bentuk matriks yang tersusun seperti pada Gambar 2.2. Sisi sebelah kanan dari persamaan pada Gambar 2.2 merupakan representasi sebuah citra digital. Setiap elemen dari *array* matriks tersebut dinamakan piksel. Satu piksel dengan piksel lainnya yang bertetangga mempunyai jarak yang sama (Trianggono, 2007).

$$f(x,y)=\begin{bmatrix} f(0,0) & f(1,0) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.2 Representasi citra digital
Sumber: Trianggono, 2007

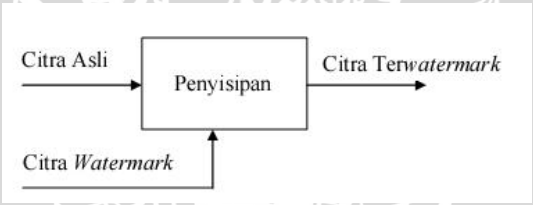
2.2 *Watermarking*

Menurut Supangkat (2000), *watermark* merupakan sebuah informasi yang disisipkan pada media lain dengan tujuan melindungi media yang disisipi oleh informasi tersebut dari pembajakan, penyalahgunaan hak cipta, dsb. *Watermarking* sendiri adalah suatu cara untuk penyembunyian atau penanaman data tertentu ke dalam suatu data digital lainnya, tetapi tidak diketahui kehadirannya oleh indera manusia dan mampu menghadapi proses-proses pengolahan sinyal digital sampai pada tahap tertentu (Yusuf, 2009).

Watermarking berkembang seiring dengan perkembangan jaman dengan munculnya *watermarking* pada media digital atau disebut dengan digital *watermarking*. Salah satu prinsip dalam digital *watermarking* adalah informasi yang disisipkan pada media digital tidak boleh mempengaruhi kualitas media digital tersebut. Jadi pada citra digital, mata manusia tidak dapat membedakan apakah citra tersebut disisipi *watermark* atau tidak. Oleh karena itu pada digital *watermarking* terdapat persyaratan bahwa digital *watermark* yang disisipkan dalam citra digital haruslah *imperceptible* atau tidak terdeteksi oleh sistem penglihatan manusia (*Human Visual System*) atau sistem pendengaran manusia (*Human Auditory System*). Digital *watermarking* sendiri adalah sebuah kode identifikasi yang secara permanen disisipkan ke dalam data digital dengan membawa informasi yang berhubungan dengan perlindungan hak cipta dan otentikasi data.

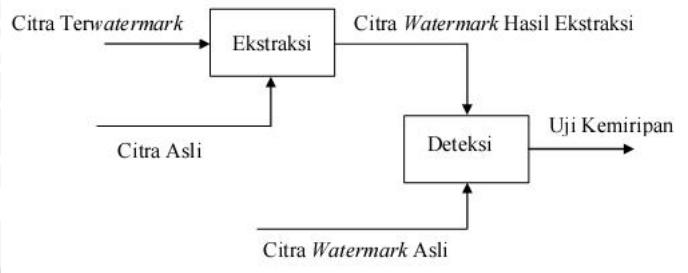
Menurut Bender (1996), ada beberapa tujuan yang ingin dicapai dari penggunaan *watermarking* sebagai suatu teknik penyembunyian data pada suatu data digital lain (Yusuf, 2009) yaitu:

1. *Tamper-proofing*
   Watermarking digunakan sebagai indikator yang menunjukkan apakah data digital yang asli telah mengalami perubahan dari yang aslinya(mengecek integritas data).
2. *Feature location*
   Watermarking sebagai alat identifikasi isi dari data digital pada lokasi-lokasi tertentu, misalnya penamaan suatu objek tertentu dari beberapa objek yang ada pada suatu citra digital.
3. *Annotation / caption*
   Watermark berisi keterangan tentang data digital itu sendiri. Selain itu *watermark* juga dapat digunakan untuk mengirimkan pesan rahasia.
4. *copyright-labelling*
   Watermarking digunakan sebagai metode untuk menyembunyikan label hak cipta pada data digital atau sebuah bukti otentik kepemilikan atas dokumen digital tersebut.



Gambar 2.3 Penyisipan *watermark*

Proses *watermarking* perlu didukung dengan proses ekstraksi *watermark* dari citra ber*watermark*. Proses ekstraksi ini bertujuan untuk mendapatkan kembali citra *watermark* yang disisipkan dalam citra digital tersebut. Umumnya proses ekstraksi melibatkan proses pembandingan citra digital asal dengan citra ber*watermark* untuk mendapatkan *watermark* yang disisipkan.



Gambar 2.4 Ekstraksi *watermark*

Menurut Lee (1999), teknik *watermarking* pada citra digital dapat diklasifikasikan dalam dua kategori, yaitu teknik ranah spasial dan teknik ranah frekuensi (Yusuf, 2009). Pada ranah spasial, penyisipan *watermark* dilakukan secara langsung ke dalam piksel citra sedangkan pada ranah frekuensi, penyisipan *watermark* dilakukan ke dalam koefisien transformasi. Biasanya teknik *watermarking* yang kuat (susah dipecahkan oleh berbagai serangan) memiliki kualitas gambar ber*watermark* yang kurang memuaskan, demikian juga sebaliknya, teknik *watermarking* yang menghasilkan kualitas gambar yang memuaskan biasanya kurang kuat menghadapi serangan. Secara garis besar teknik *watermarking* dibedakan menjadi dua (Alfatwa, 2009) yaitu:

1. *Non-blind watermarking*
   Merupakan teknik *watermarking* yang membutuhkan citra asli dan citra ber*watermark* untuk mengekstrak *watermark*.
2. *Blind watermarking*
   Merupakan teknik *watermarking* yang tidak membutuhkan citra asli atau *watermark* yang disisipkan untuk melakukan ekstraksi.

2.3 *Singular Value Decomposition* (SVD)

Dekomposisi nilai singular atau yang lebih dikenal sebagai SVD (Singular Value Decomposition) adalah salah satu teknik dekomposisi yang cukup terkenal. SVD berkaitan erat dengan nilai singular dari sebuah matriks yang merupakan salah satu karakteristik matriks (Wijna, 2007).

Matriks A dengan nilai eigen (nilai karakterisitik) dari matriks A^T A yaitu λ_i untuk setiap $1 \leq i \leq n$ dengan n yaitu jumlah

nilai eigen, maka nilai singular matriks A yaitu $\sigma_i = \sqrt[2]{\lambda_i}$ dan v_i merupakan vektor eigen matriks $A^T A$ yang bersesuaian dengan λ_i.

Secara umum algoritma dekomposisi nilai singular adalah sebagai berikut (Wijna, 2007) :

input : matriks A

output : matriks ortogonal U,V dan matriks singular S sehingga $A = USV^T$.

1. Dibentuk matriks $A^T A$ dengan nilai eigen λ_i untuk setiap $1 \le i \le n$ maka nilai singular matriks A yaitu $\sigma_i = \sqrt[2]{\lambda_i}$ .....(2.1)
2. Dibentuk matriks diagonal S = $\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n \end{pmatrix}$ .....(2.2)
3. Dicari himpunan vektor eigen dari matriks $A^T A$. Misalkan $\{v_1, v_2, ..., v_n\}$ merupakan vektor-vektor eigen matriks $A^T A$ dengan v_i merupakan vektor eigen yang bersesuaian dengan λ_i.
4. Dibentuk matriks ortogonal V = $[v_1, v_2, ..., v_n]$ .....(2.3)
5. Dibentuk himpunan vektor $\{u_1, u_2, ..., u_n\}$ dengan $u_i = \frac{1}{\sigma_i} A v_i$ .....(2.4)
   untuk setiap $1 \le i \le n$.
6. Dibentuk matriks ortogonal U = $[u_1, u_2, ..., u_n]$ .....(2.5)
7. Bentuk dekomposisi SVD $A = USV^T$ .....(2.6)

Dekomposisi tersebut disebut sebagai dekomposisi nilai singular. Nilai $\sigma_1 ... \sigma_n$ dari S disebut sebagai nilai-nilai singular dari A, kolom-kolom dari U merupakan vektor-vektor singular kiri dari A dan kolom-kolom dari V disebut sebagai vektor-vektor singular kanan dari A. Jika A adalah sebuah citra maka pengubahan sedikit pada nilai-nilai singular tidak mempengaruhi kualitas citra dan nilai-nilai singular tidak berubah banyak setelah citra diserang (Yusuf, 2009).

Misalkan suatu citra disajikan sebagai matriks A dan *watermark* yang akan disisipkan disajikan sebagai matriks W, maka penyisipan *watermark* W ke dalam citra A dilakukan dengan terlebih dahulu mendekomposisi citra A menjadi matriks U, S, dan V untuk mendapatkan nilai singular dari citra A. nilai singular S kemudian ditambahkan dengan hasil kali *watermark* W dengan nilai alfa (Cahyana, 2007).

$S_t = S + \alpha * W$ .....(2.7)

α adalah faktor intensitas yang menentukan kekuatan *watermark* yang akan disisipkan. Kemudian melakukan dekomposisi pada S_t untuk memperoleh nilai singular baru pada S_t.

$S_t = U_w S_w V_w^T$ .....(2.8)

Sebagai langkah terakhir, S_w yang diperoleh kemudian digunakan untuk membentuk citra yang telah di-*watermark* bersama dengan matriks U dan V dari citra asal.

$A_w = U S_w V^T$ .....(2.9)

Langkah-langkah dalam proses ekstraksi adalah sebagai berikut :

$A^*_w = U^* S^*_w (V^*)^T$ .....(2.10)

$S_t = U_w S_w V_w^T$

$$W = \frac{S_t - S}{nilai\ alfa} \quad \text{.....(2.11)}$$

2.4 Parameter kinerja *watermarking*

Kinerja *watermarking* diukur dari kualitas citra yang dihasilkan, nilai *Normalized Crosscorelation* (NC) *watermark* hasil ekstraksi dengan *watermark* asli, dan lama waktu penyisipan *watermark*.

2.4.1 *Peak Signal to Noise Ratio* (PSNR)

Peak signal to noise ratio (PSNR) digunakan untuk menentukan kualitas citra. Nilai PSNR diperoleh dengan membandingkan citra asli dan citra rekonstruksi. Untuk menentukan nilai PSNR digunakan rumus :

$$PSNR = 20 * \log_{10}\left({}^{225}\!\!\diagup\!\!{}_{\sqrt{MSE}} \right) \quad \text{.....(2.12.1)}$$

Dengan,

$$MSE = \left(1/mn\right) \sum_{y=1}^{m} \sum_{x=1}^{n} \left[I(x,y) - I'(x,y)\right]^2 \quad \text{.....(2.12.2)}$$

m dan n adalah jumlah baris dan kolom citra, x dan y adalah koordinat pixel citra, I dan I' adalah citra asli dan citra rekonstruksi. Ukuran kualitas citra disajikan dalam Tabel 2.1(Anwar, 2008).

Tabel 2.1 Nilai PSNR

PSNR(dB)	Kualitas citra
60	Istimewa (*excellent*)
50	Bagus (*good*)
40	Layak (*reasonable*)
30	Cukup (*poor picture*)
20	Tidak dapat dipakai (*unusable*)

2.4.2 *Normalized Crosscorelation* (NC)

Tingkat kemiripan antara citra *watermark* asli dengan citra *watermark* hasil ekstraksi secara kuantitatif diukur menggunakan *Normalized Crosscorelation* (NC) (Setiadikarunia, 2008).

NC didefinisikan sebagai berikut :

$$NC = \frac{\sum_i \sum_j w_{ij} w'_{ij}}{\sum_i \sum_j \left[w_{ij}\right]^2} \quad \text{.....(2.13)}$$

Dengan w_{ij} dan w'_{ij} adalah nilai bit pixel (i,j) pada citra *watermark* asli dan citra *watermark* hasil ekstraksi.

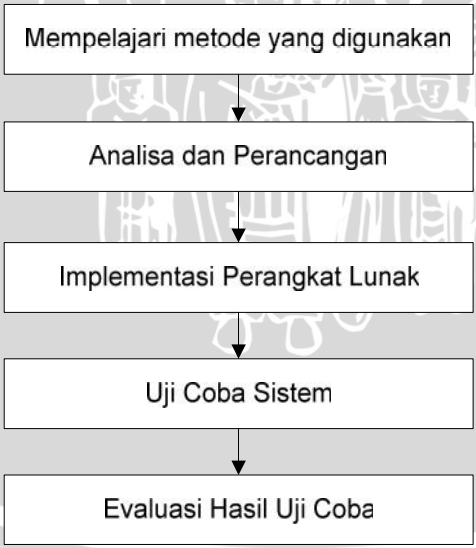

UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metodologi dan perancangan perangkat lunak untuk teknik kompresi citra menggunakan DNS berbasis blok. Penelitian dilakukan dengan tahapan – tahapan sebagai berikut:

1. Mempelajari literatur yang terkait masalah *watermarking* citra dengan teknik SVD.
2. Melakukan analisa dan perancangan perangkat lunak untuk *watermarking* dengan teknik SVD.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Melakukan uji coba perangkat lunak dengan melakukan *watermarking* citra dan ekstraksi citra yang telah ter-*watermark* terhadap beberapa citra.
5. Mengevaluasi tingkat keberhasilan perangkat lunak dalam melakukan *watermarking* citra dan ekstraksi citra yang telah ter-*watermark*.

Tahapan – tahapan penelitian yang dilakukan dapat digambarkan pada Gambar 3.1.



Gambar 3.1 Tahapan – Tahapan Penelitian

3.1 Analisa Sistem

Pada subbab ini akan dibahas mengenai deskripsi sistem penerapan *image watermarking* dengan metode SVD.

3.1.1 Deskripsi Sistem

Sistem yang akan dibangun adalah sistem untuk melakukan *watermark* terhadap citra. Sistem ini terdiri dari dua proses utama yaitu *embedding* (penyisipan) dan *extracting* (ekstraksi).

Proses penyisipan dimulai dengan menerima masukan berupa dua citra, yaitu citra asli yang akan diberi *watermark* dan citra logo asli sebagai *watermark* untuk disisipkan pada citra asli. Ukuran dimensi citra *watermark* atau logo harus lebih kecil daripada citra asli yang akan disisipi *watermark*. Selanjutnya matriks R,G dan B pada citra asli masing-masing didekomposisi menggunakan fungsi SVD menjadi tiga matriks S,U dan V, kemudian matriks S dimodifikasi dengan menyisipkan matriks dari citra *watermark* yang dikalikan dengan suatu nilai konstanta sebagai nilai intensitas kekuatan *watermark* dengan rentang 0,1 sampai 1 dengan jarak 0,1. Proses dilanjutkan dengan mengkomposisi matriks S hasil modifikasi lalu digabungkan dengan matriks U dan V dari citra asal. Citra hasil penyisipan *watermark* disimpan ke dalam file *image* baru.

Proses ekstraksi *watermark* dilakukan untuk membuktikan bahwa suatu citra disisipi oleh *watermark* atau tidak dan apabila tersisipi oleh *watermark,* maka citra asli dipisahkan dengan citra *watermark.* Teknik pemberian *watermark* berbasis SVD ini menggunakan transformasi spasial *non-blind,* yaitu pada proses ekstraksi membutuhkan komponen dari citra asli atau citra asalnya. Proses ekstraksi dimulai dengan memasukkan file citra asli atau citra asal dan file citra yang telah disisipkan *watermark.* Selanjutnya matriks R,G dan B dari citra asli dan citra ter-*watermark* dikomposisi menjadi tiga buah matriks. Proses ekstraksi membutuhkan matriks S dari citra asli untuk mendapatkan matriks U_w dan V_w . Setelah diperoleh matriks U_w dan V_w , kemudian dikalikan dengan matriks S^*_w yang berasal dari citra yang ter-*watermark* untuk mendapatkan matriks S*. Matriks S* yang telah didapat, kemudian dikurangi dengan matriks S pada citra asli dan hasilnya dibagi dengan nilai intensitasnya. Citra hasil ekstraksi disimpan ke dalam file *image* yang baru. Pada proses autentifikasi dilakukan korelasi antara *watermark* yang diperoleh, kemudian apabila nilai hasil korelasi

melebihi batas ambang, maka *watermark* yang disisipkan milik sah seseorang.

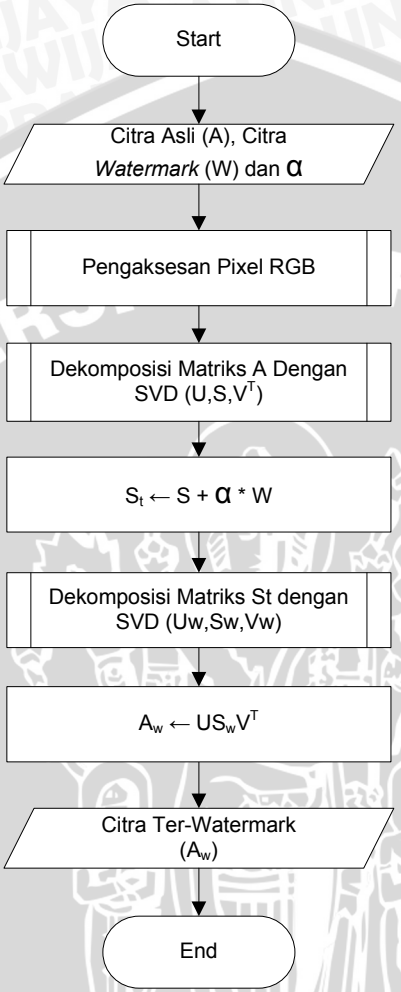
3.2 Perancangan Sistem

Pada subbab ini akan dibahas mengenai perancangan sistem penerapan *image watermarking* dengan metode SVD. Secara umum perancangan sistem dibagi ke dalam dua proses utama yaitu perancangan proses penyisipan *watermark* dan perancangan proses ekstraksi *watermark*. Pada proses penyisipan *watermark,* citra masukan akan melewati serangkaian proses hingga dihasilkan keluaran berupa file citra yang telah tersisipi oleh *watermark*. Sedangkan proses ekstraksi akan mengolah masukan berupa file citra yang telah tersisipi *watermark* sehingga dihasilkan keluaran berupa citra *watermark* hasil ekstraksi.

3.2.1 Perancangan proses penyisipan *watermark*

Proses penyisipan terdiri dari beberapa subproses, diantaranya proses pengaksesan nilai piksel RGB dari citra yang diinputkan, dekomposisi matriks citra asli dengan SVD, penggabungan matriks singular dengan citra *watermark*, dekomposisi matriks St, pembentukan citra ter-*watermark*. *Flowchart* proses penyisipan ditunjukkan pada gambar 3.2. Berikut ini langkah – langkah yang dilakukan pada proses penyisipan *watermark* :

1. *User* memasukkan citra asli, citra logo *watermark* dan nilai α dengan interval 0.1 sampai 1 (kelipatan 0.1).
2. Proses pengaksesan nilai piksel RGB dari citra yang diinputkan.
3. Dekomposisi matriks citra asli dengan SVD.
4. Penggabungan matriks singular dengan citra *watermark*.
5. dekomposisi matriks St.
6. Pembentukan citra ter-*watermark*
7. Keluaran yang dihasilkan berupa citra ter-*watermark.*

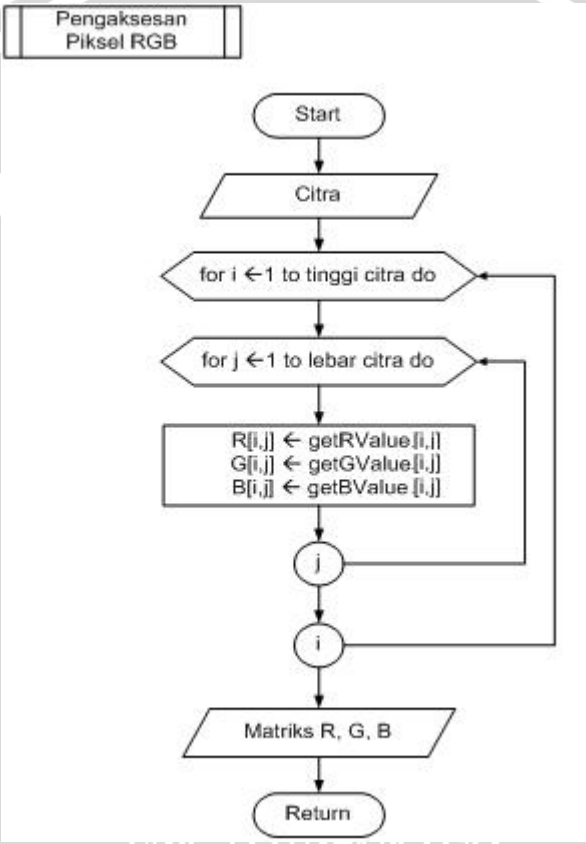


Gambar 3.2 *Flowchart* Proses Penyisipan *watermark*

3.2.1.1 Pengaksesan nilai piksel RGB

Pada proses ini, akan diambil nilai piksel R,G,B dari citra masukan. Pengambilan nilai ini dilakukan dengan melakukan proses *scanline* yaitu proses membaca *file* citra dengan indeks baris dan kolom sehingga diketahui nilai piksel indeks tersebut. Karena citra masukan yang digunakan adalah citra warna sehingga pembacaan indeks tiap piksel dilakukan sebanyak tiga kali untuk nilai piksel R,

piksel G dan piksel B. Nilai – nilai piksel ini disimpan dalam matriks R, matriks G dan matriks B dimana ketiga matriks ini adalah keluaran dari proses ini. *Flowchart* proses pengaksesan nilai piksel RGB ditunjukkan pada gambar 3.3.

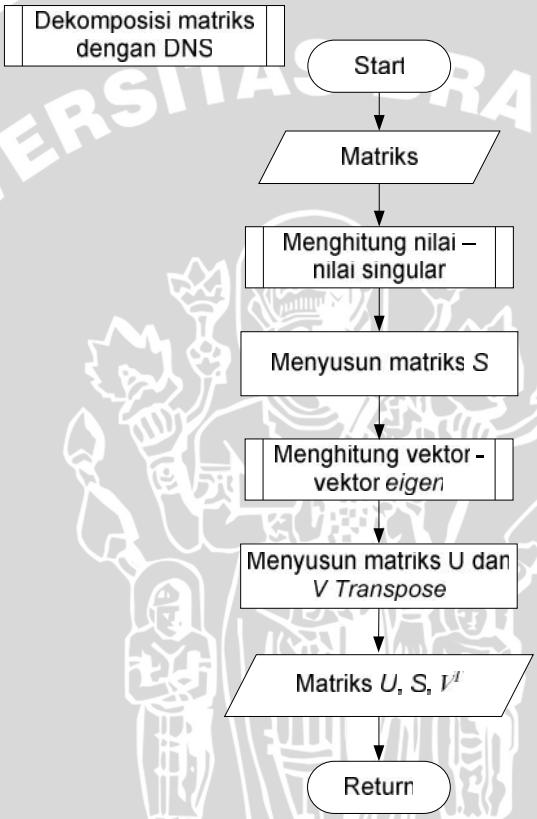


Gambar 3.3 *Flowchart* Proses pengaksesan nilai piksel RGB

3.2.1.2 Dekomposisi matriks citra asli dengan SVD

Proses dekomposisi matriks citra asli dengan menggunakan *singular value decomposition* dimulai dengan proses menginputkan matriks dari citra asli, kemudian dilakukan proses mencari nilai – nilai singular dari matriks tersebut. Nilai – nilai singular ini digunakan sebagai elemen untuk menyusun matriks diagonal *S*. Kemudian proses berikutnya adalah mencari vektor – vektor *eigen*.

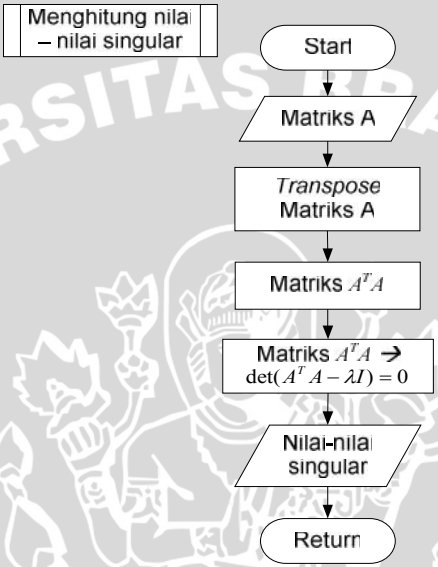
Vektor – vektor *eigen* ini merupakan elemen kolom dari matriks U dan V^T. Untuk mendapatkan kembali matriks awal, dapat diperoleh dengan mengalikan matriks – matriks dekomposisi U, S, dan V^T. *Flowchart* proses dekomposisi matriks dengan DNS ditunjukkan pada gambar 3.4.



Gambar 3.4 *Flowchart* Dekomposisi Matriks dengan SVD

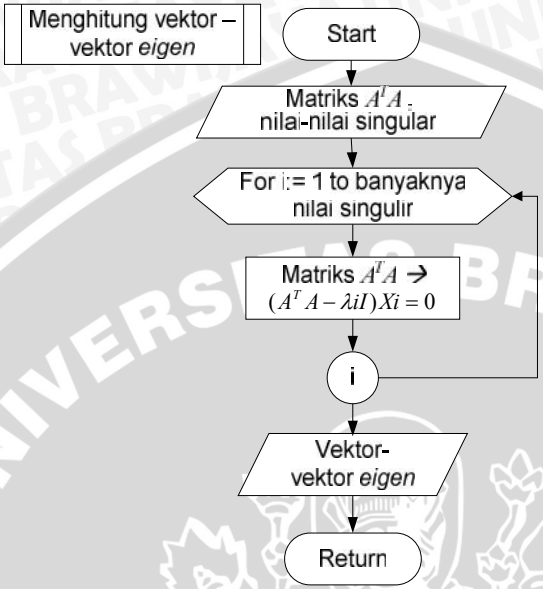
Flowchart proses untuk menghitung nilai – nilai singular ini ditunjukkan pada gambar 3.5. Proses ini menghitung nilai - nilai singular dari suatu matriks yang diinputkan. Langkah pertama mencari *transpose* dari matriks tersebut kemudin mengalikan hasil *transpose* dengan matriks awal sehingga dihasilkan matriks kovarian. Kemudian mencari persamaan karakteristik dari matriks kovarian

tersebut. Akar – akar dari persamaan karakteristik tersebut merupakan nilai – nilai *eigen* dari matriks yang diinputkan. Akar kuadrat dari nilai – nilai *eigen* ini merupakan nilai – nilai singular yang dicari.



Gambar 3.5 *Flowchart* Menghitung nilai – nilai singular

Proses menghitung vektor *eigen* ini ditunjukkan pada gambar 3.6. Proses ini mencari vektor *eigen* dari setiap nilai *eigen* yang diperoleh dari proses sebelumnya. Nilai dari vektor *eigen* ini merupakan elemen vektor kolom untuk menyusun matriks V^T dan matriks U pada proses dekomposisi nilai singular.



Gambar 3.6 *Flowchart* Menghitung vektor – vektor *eigen*

3.2.1.3 Dekomposisi matriks S_t dengan SVD

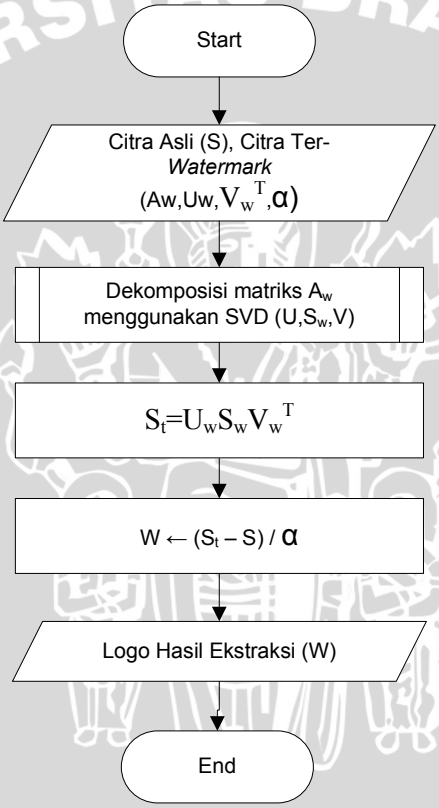
Proses dekomposisi matriks S_t dengan menggunakan SVD berfungsi untuk mendapatkan nilai S_w.Proses dekomposisi matriks S_t dimulai dengan proses mencari nilai – nilai singular dari matriks S_t tersebut. Nilai – nilai singular ini digunakan sebagai elemen untuk menyusun matriks diagonal S_w. Kemudian proses berikutnya adalah mencari vektor – vektor *eigen*. Vektor – vektor *eigen* ini merupakan elemen kolom dari matriks U_w dan $V_w{}^T$. *Flowchart* proses dekomposisi matriks S_t dengan SVD sama seperti pada gambar 3.4. S_w yang didapat digunakan untuk membentuk citra yang telah di-*watermark* bersama dengan matriks U dan V dari citra asal.

3.2.2 Perancangan proses ekstraksi *watermark*

Proses penyisipan terdiri dari beberapa subproses, diantaranya proses pengaksesan nilai piksel RGB dari citra yang diinputkan, dekomposisi matriks citra asli dengan SVD, penggabungan matriks singular dengan citra *watermark*, dekomposisi matriks St, pembentukan citra ter-*watermark*. *Flowchart* proses penyisipan

ditunjukkan pada gambar 3.2. Berikut ini langkah – langkah yang dilakukan pada proses penyisipan *watermark* :

1. *User* memasukkan citra ter-*watermark* dan citra asli.
2. Dekomposisi matriks A_w , SVD(A_w) = [U,S_w,V].
3. Mengalikan matriks U_w , S_w dan V_w^T , S_t=$U_w S_w V_w^T$.
4. Ekstrak logo *watermark* dengan mengurangi S_t dan S lalu dibagi dengan nilai alfa, W = (S_t – S) / α.
5. Keluaran yang dihasilkan berupa citra logo *watermark.*



Gambar 3.7 *Flowchart* Proses ekstraksi *watermark*

3.2.2.1 Dekomposisi matriks A_w dengan SVD

Proses dekomposisi matriks A_w dengan menggunakan SVD berfungsi untuk mendapatkan nilai singular S_w.Proses dekomposisi matriks A_w dimulai dengan proses mencari nilai – nilai singular dari

matriks A_w tersebut. Nilai – nilai singular ini digunakan sebagai elemen untuk menyusun matriks diagonal S_w. Kemudian proses berikutnya adalah mencari vektor – vektor *eigen*. Vektor – vektor *eigen* ini merupakan elemen kolom dari matriks *U* dan *V*. *Flowchart* proses dekomposisi matriks A_w dengan SVD sama seperti pada gambar 3.4. Nilai singular S_w yang didapat digunakan untuk memperoleh matriks S_t bersama dengan matriks U_w dan V_w^T.

3.3 Perancangan Uji Coba

Pada subbab ini akan dilakukan perancangan uji coba dari perangkat lunak *watermarking* citra dengan teknik SVD. Pengujian yang dilakukan pada perangkat lunak ini adalah pengujian nilai korelasi yaitu dengan membandingkan citra logo *watermark* asli dengan citra logo *watermark* hasil ekstraksi, pengujian nilai PSNR yaitu dengan menghitung tingkat kesalahan pada citra asli dibandingkan dengan citra setelah disisipkan logo *watermark dan* pengujian nilai MSE yaitu dengan menghitung rata – rata wilayah kesalahan (*error*).

3.3.1 Pengujian Kebenaran Sistem

Untuk menguji kebenaran perangkat lunak pengujian yang dilakukan adalah mencoba menyisipkan *watermark* ke dalam citra *host* atau citra asli dan mencoba mendapatkan kembali *watermark* yang telah disisipkan (ekstraksi *watermark*). Rancangan tabel pengujian kebenaran system ditunjukkan pada tabel 3.1.

Tabel 3.1 Tabel Uji Kebenaran

No	Citra *Host*	Citra *Watermark*	α	Hasil Penyisipan	Hasil Ekstraksi

3.3.2 Pengujian Kinerja Sistem

Skenario pengujian kinerja system antara lain :

1. Variasi citra asal (*host*) yang digunakan.
2. Variasi nilai α (0.1, 0.5, 1).
3. Menghitung MSE (*Mean Square Error*) dari tiap-tiap skenario pengujian.

Penghitungan MSE digunakan untuk mengetahui seberapa besar *noise* yang terjadi akibat proses *watermark* dimana nilai MSE menentukan seberapa dekat hasil *watermarking* terhadap HVS (*Human Visual System*). Rancangan pengujian kinerja system ditunjukkan pada tabel 3.2.

Tabel 3.2 Tabel Uji Kinerja Sistem

No	Citra *Host*	Citra *Watermark*	α	PSNR Citra Ter-*watermark*	MSE Logo Hasil Ekstraksi	Nilai Korelasi

3.3.3 Pengujian Ketahanan Citra Ber-*watermark*

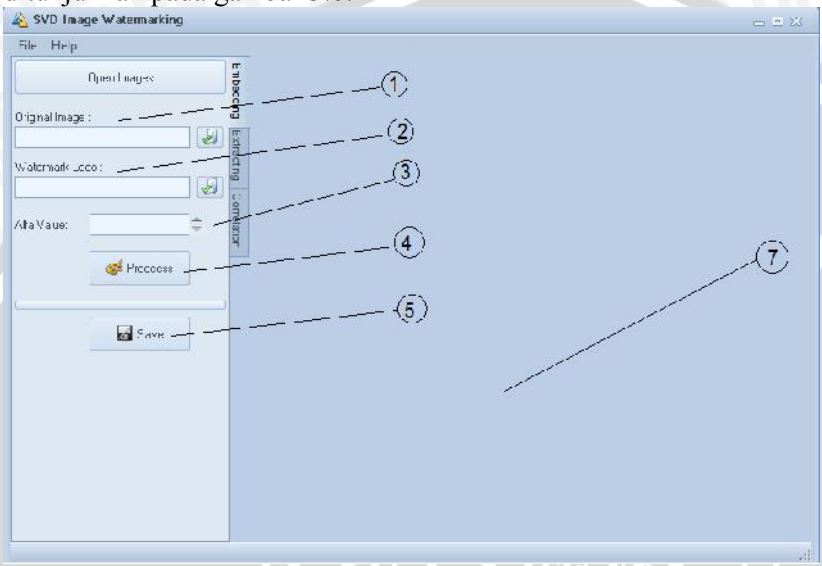
Pengujian ketahanan citra ber-*watermark* ini dilakukan untuk menguji apakah *watermark* yang disisipkan dapat tetap bertahan setelah mengalami beberapa perubahan. Jenis-jenis serangan yang digunakan yaitu efek *Blur, noise, pixelate, sharpening*.

Tabel 3.3 Tabel Uji Ketahanan Citra Ber-*watermark*

No	Citra Ter-*watermark*	α	Jenis Serangan	Nilai Korelasi	MSE

3.4 Perancangan Antar Muka

Perancangan antar muka dari aplikasi kompresi citra ditunjukkan pada gambar 3.8.



Gambar 3.8 Rancangan Form *Image watermarking*

Keterangan gambar 3.8:

1. Tombol untuk me*load* citra asli.
2. Tombol untuk me*load* citra *watermark*.
3. Spin edit untuk menentukan nilai alfa.
4. Tombol proses untuk proses penyisipan *watermark*.
5. Tombol simpan untuk menyimpan citra hasil.
6. Page control untuk memilih halaman proses penyisipan,ekstraksi dan korelas.
7. Bagian form untuk menampilkan form citra.

3.5 Perhitungan Manual

Pada perhitungan manual digunakan gambar *test.bmp* dengan dimensi 2x2. Berikut langkah – langkah untuk memperoleh matriks dekomposisi dari gambar *test.bmp* dengan gambar *logo.bmp* sebagai logo *watermark* dengan piksel biner atau tidak berwarna.

Berikut ini adalah matriks yang merepresentasikan gambar *test.bmp* dan *logo.bmp*.

Gambar 3.9 test.bmp

2	-2
1	2

Gambar 3.9 a) Nilai piksel red *test.bmp*

25	55
7	3

Gambar 3.9 b) Nilai piksel green *test.bmp*

10	8
15	7

Gambar 3.9 c) Nilai piksel blue *test.bmp*

0	1
1	1

Gambar 3.9 d) Nilai piksel *logo.bmp*

Dari ketiga tabel yang berisi nilai piksel R, G, dan B akan dikelompokkan menjadi matriks R, G, B. Setiap matriks akan didekomposisikan kedalam Dekomposisi Nilai Singulir (DNS).

Matriks red direpresentasikan sebagai matriks A = $\begin{bmatrix} 2 & -2 \\ 1 & 2 \end{bmatrix}$, citra *watermark* dengan matriks W = $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ dan nilai alfa = 1.

Langkah-langkah dalam proses penyisipan adalah sebagai berikut :

1. $A^TA = \begin{bmatrix} 2 & 1 \\ -2 & 2 \end{bmatrix}\begin{bmatrix} 2 & -2 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 5 & -2 \\ -2 & 8 \end{bmatrix}$
2. Menghitung nilai eigen untuk mendapatkan nilai singularnya dengan determinan matriks A^TA

   $A^TA - \lambda = \begin{bmatrix} 5-\lambda & -2 \\ -2 & 8-\lambda \end{bmatrix}$

   $|A^TA - \lambda| = 0$

   $(5-\lambda)(8-\lambda) - 4 = 0$

   $40 - 5\lambda - 8\lambda + \lambda^2 - 4 = 0$

   $\lambda^2 - 13\lambda + 36 = 0$

   $(\lambda - 9)(\lambda - 4) = 0$

   $\lambda_1 = 9$ dan $\lambda_2 = 4 \rightarrow$ nilai eigen

   Nilai singular $\rightarrow \sigma_1 = \sqrt{9} = 3$ dan $\sigma_2 = \sqrt{4} = 2$

   Maka matriks S = $\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$
3. Nilai-nilai eigen $\lambda_1 = 9, \lambda_2 = 4$ masing-masing bersesuaian dengan vektor eigen $v_1 = \begin{bmatrix} 1/\sqrt{5} \\ -2/\sqrt{5} \end{bmatrix}$ dan $v_2 = \begin{bmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix}$. Himpunan vektor-vektor eigen tersebut ortonormal sehingga dapat dibentuk matriks unitary $V = [v_1 \quad v_2] = \begin{bmatrix} 1/\sqrt{5} & 2/\sqrt{5} \\ -2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix}$

4. Kemudian matriks U dibentuk dari $u_i = \frac{1}{\sigma_i} A v_i$ sehingga diperoleh :

$$u_1 = \frac{1}{\sigma_1} A v_1$$

$$= \frac{1}{3}\begin{bmatrix} 2 & -2 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 1/\sqrt{5} \\ -2/\sqrt{5} \end{bmatrix}$$

$$= \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$$

$$u_2 = \frac{1}{\sigma_2} A v_2$$

$$= \frac{1}{2}\begin{bmatrix} 2 & -2 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 2/\sqrt{5} \\ 1/\sqrt{5} \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$$

Maka matriks $U = [u_1 \quad u_2] = \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}$

5. $A = USV^T = \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix}\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1/\sqrt{5} & -2/\sqrt{5} \\ 2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix}$

6. $S_t = S + nilaialfa * W$

$$= \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} + 1\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

7. Melakukan dekomposisi pada $S_t = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$ untuk memperoleh nilai singular dari S_t.

8. $S_t^T S_t = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$

9. Menghitung eigen untuk mendapatkan nilai singularnya dengan determinan dari matriks $S_t^T S_t$

$$S_t^T S_t - \lambda = \begin{bmatrix} 10-\lambda & 6 \\ 6 & 10-\lambda \end{bmatrix}$$

$$| S_t^T S_t - \lambda | = 0$$

$$(10-\lambda)(10-\lambda) - 36 = 0$$

$$100 - 10\lambda - 10\lambda + \lambda^2 - 36 = 0$$

$$\lambda^2 - 20\lambda + 64 = 0$$

$$(\lambda - 16)(\lambda - 4) = 0$$

$\lambda_1 = 16$ dan $\lambda_2 = 4 \rightarrow$ nilai eigen

Nilai singular $\rightarrow \sigma_1 = \sqrt{16} = 4$ dan $\sigma_2 = \sqrt{4} = 2$

Maka matriks $S_w = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$

10. Nilai-nilai eigen $\lambda_1 = 16, \lambda_2 = 4$ masing-masing bersesuaian dengan vektor eigen $v_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ dan $v_2 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$. Himpunan vektor-vektor eigen tersebut ortonormal sehingga dapat dibentuk matriks unitary

$$V_w = [v_1 \quad v_2] = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

11. Kemudian matriks U_w dibentuk dari $u_i = \frac{1}{\sigma_i} S_t v_i$ sehingga diperoleh :

$$U_1 = \frac{1}{\sigma_1} S_t V_1$$

$$= \frac{1}{4} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$U_2 = \frac{1}{\sigma_2} S_t V_2$$

$$= \frac{1}{2} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

$$= \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

Maka matriks $U_w = [u_1 \quad u_2] = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$

12. $S_t = U_w S_w V_w^T = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$

13. S_w yang diperoleh digunakan untuk membentuk citra yang telah di-*watermark* bersama dengan matriks U dan V dari citra asal.

14. $A_w = US_wV^T$

$$A_w = \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ -1/\sqrt{5} & 2/\sqrt{5} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1/\sqrt{5} & -2/\sqrt{5} \\ 2/\sqrt{5} & 1/\sqrt{5} \end{bmatrix}$$

$$A_w = \begin{bmatrix} 12/5 & 14/5 \\ 4/5 & 4/5 \end{bmatrix} \rightarrow \text{citra matriks red ter-}watermark$$

15. Selanjutnya lakukan pada matriks green dan blue kemudian matriks red, green dan blue digabungkan untuk membentuk image ter-*watermark*.

Pada proses ekstraksi, langkah-langkahnya adalah sebagai berikut :

1. Dekomposisi matriks citra ter-*watermark* A_w sehingga diperoleh nilai singular S_w. Nilai singular S_w yang diperoleh digunakan untuk memperoleh matriks S_t bersama dengan matriks U_w dan $V_w^T \rightarrow S_t = U_w S_w V_w^T$
2. Selanjutnya citra *watermark* dapat diperoleh dengan $W = \frac{S_t - S}{nilai\,alfa}$
3. $W = \frac{S_t - S}{nilai\,alfa}$

$$= \frac{\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} - \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}}{1}$$

$$= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Dari perhitungan manual tersebut diperoleh matriks citra pixel Red hasil penyisipan $A_w = \begin{bmatrix} 12/5 & 14/5 \\ 4/5 & 4/5 \end{bmatrix}$ seperti pada gambar 3.10 yang kemudian digabungkan dengan nilai pixel green dan blue sehingga terbentuk citra ter-*watermark*. Sedangkan pada

perhitungan manual proses ekstraksi diperoleh kembali citra logo *watermark* hasil ekstraksi yang mirip dengan citra logo *watermark* asli, yaitu $W = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$. Kemudian nilai-nilai pixel tersebut disimpan pada sebuah file citra digital dimensi dua diam atau biasa disebut *image*.

12/5	14/5
4/5	4/5

Gambar 3.10 Nilai piksel Red ter-*watermark*


UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan implementasi

Implementasi sistem ini berupa aplikasi *watermark* citra digital dengan menggunakan metode *singular value decomposition* untuk memberi tanda hak kepemilikan pada suatu citra digital. Adapun yang akan dijelaskan dalam subbab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan perangkat lunak *watermarking* pada citra digital ini adalah:

1. Prosesor AMD Turion(tm) II X2 M520
2. Memori 1 Gb
3. Harddisk 320 Gb
4. Layar 14''
5. Keyboard
6. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan program *watermarking* pada citra digital dan uji coba adalah:

1. Sistem operasi Windows 7 Ultimate
2. Borland Delphi 7.0
3. Adobe Photoshop Cs 2.

4.2 Implementasi Program

4.2.1 Implementasi Penyisipan *Watermark*

Langkah pertama yang dilakukan untuk proses penyisipan adalah memasukkan citra asli/*host* yang akan disisipi *watermark*, memasukkan citra logo *watermark*, menentukan nilai alfa, melakukan pengaksesan terhadap nilai R,G,B dari citra inputan. Berikut ini beberapa prosedur dan fungsi yang digunakan saat proses penyisipan :

4.2.1.1 Pengaksesan nilai piksel RGB

Proses ini adalah melakukan pengaksesan terhadap nilai R,G,B dari citra inputan. Untuk mengakses nilai piksel R,G,B digunakan prosedur *GetPixel* RGB yang akan menghasilkan 3 matriks 2D yaitu matriks R, G, B. Prosedur *GetPixel* RGB ditunjukkan pada Gambar 4.1.

```
procedure GetPixelRGB(Img: TBitmap;var sR,sG,sB:
TReal2DArray);
var
  h,w,i,j:integer;
begin
  h:= Img.Height;
  w:= Img.Width;
  SetLength(sR,h,w);
  SetLength(sG,h,w);
  SetLength(sB,h,w);
  for j:= 0 to h-1 do
  begin
    p:= Img.ScanLine[j];
  for i:= 0 to w-1 do
  begin
    sR[j,i]:=p[i].r;
    sG[j,i]:=p[i].g;
    sB[j,i]:=p[i].b;
  end;end;
```

Gambar 4.1 *Source Code procedure GetPixel* RGB

4.2.1.2 Dekomposisi citra dengan SVD

Setelah proses mengambil nilai piksel R,G,B dari citra inputan, langkah selanjutnya adalah mendekomposisikan citra dengan SVD. Matriks RGB pada masing-masing citra didekomposisikan dan digabungkan sehingga terbentuk matriks ter-*watermark*. Prosedur blok ditunjukkan pada Gambar 4.2.

```
    alfa:=StrToFloat(sEdit3.Text);
    GetPixelRGB(ImgAsli,mR,mG,mB);
    GetPixelRGB(Imglogo,mRw,mGw,mBw);
    //cek seimbang panjang,lebar matriks
    if h_a > w_a then z := h_a else z := w_a;
    mR := Matriks_seimbang(mR,h_a,w_a,z);
    mG := Matriks_seimbang(mG,h_a,w_a,z);
    mB := Matriks_seimbang(mB,h_a,w_a,z);
    mRw := Matriks_seimbang(mRw,h_w,w_w,z);
    mGw := Matriks_seimbang(mGw,h_w,w_w,z);
    mBw := Matriks_seimbang(mBw,h_w,w_w,z);

    //dekomposisi matriks image asli U S Vt
    RMatrixSVD(mr,z,z,2,2,2,Sr,mUr,mVtr);
    RMatrixSVD(mg,z,z,2,2,2,Sg,mUg,mVtg);
    RMatrixSVD(mb,z,z,2,2,2,Sb,mUb,mVtb);
    mSr := matrikssingular(Sr,z);
    mSg := MatriksSingular(Sg,z);
    mSb := MatriksSingular(Sb,z);

    //St = S + alfa * W
    Str :=
PenjumlahanMatriks(mSr,PerkalianMatrix2(mRw,alfa,z,z),z
,z);
    Stg :=
PenjumlahanMatriks(mSg,PerkalianMatrix2(mGw,alfa,z,z),z
,z);
    Stb :=
PenjumlahanMatriks(mSb,PerkalianMatrix2(mBw,alfa,z,z),z
,z);

    //dekomposisi matriks St = Uw,Sw,Vtw
    RMatrixSVD(Str,z,z,2,2,2,Swr,mUwr,mVtwr);
    RMatrixSVD(Stg,z,z,2,2,2,Swg,mUwg,mVtwg);
    RMatrixSVD(Stb,z,z,2,2,2,Swb,mUwb,mVtwb);
    mSwr := matrikssingular(Swr,z);
    mSwg := MatriksSingular(Swg,z);
    mSwb := MatriksSingular(Swb,z);
```

```
//bentuk matriks citra terwatermark Aw = U Sw Vt
    mAwr :=
PerkalianMatrix(PerkalianMatrix(mUr,mSwr,z,z,z),mVtr,z,
z,z);
    mAwg :=
PerkalianMatrix(PerkalianMatrix(mUg,mSwg,z,z,z),mVtg,z,
z,z);
    mAwb :=
PerkalianMatrix(PerkalianMatrix(mUb,mSwb,z,z,z),mVtb,z,
z,z);

    for i := 0 to z-1 do
    begin
      for j := 0 to z-1 do
      begin
        mAwr[i,j] := ToByte(mAwr[i,j]);
        mAwg[i,j] := ToByte(mAwg[i,j]);
        mAwb[i,j] := ToByte(mAwb[i,j]);
      end;
    end;

    if Formhasilterwatermark = nil then

Application.CreateForm(TFormhasilterwatermark,Formhasil
terwatermark);

Formhasilterwatermark.Image1.Picture.LoadFromFile(sEdit
1.Text);
    Formhasilterwatermark.Image1.Height := h_a;
    Formhasilterwatermark.Image1.Width := w_a;
    Formhasilterwatermark.ClientHeight := h_a;
    Formhasilterwatermark.ClientWidth := w_a;

    for i := 0 to h_a-1 do
    begin
      for j := 0 to w_a-1 do
      begin

Formhasilterwatermark.Image1.Picture.Bitmap.Canvas.Pixe
ls[j,i] :=
RGB(tobyte(round(mAwr[i,j])),tobyte(round(mAwg[i,j])),t
obyte(round(mAwb[i,j])));
      end;
    end;
```

Gambar 4.2 *Source Code* Penyisipan *watermark*

4.2.2 Implementasi Ekstraksi *watermark*

Proses ekstraksi, terdiri dari proses pembacaan file citra ter-*watermark*, citra *host*/asli dan citra logo watermark sebagai pembanding dengan logo hasil ekstraksi. Proses dimulai dari mendekomposisikan matriks citra ter-*watermark,* kemudian didapatkan nilai Uw Sw Vw yang dioperasikan dengan membalikkan proses seperti pada proses penyisipan. Berikut *procedure* yang digunakan untuk proses ekstraksi ditunjukkan pada Gambar 4.3.

```
GetPixelRGB(ImgAsli,mR,mG,mB);
GetPixelRGB(Imglogo,mRw,mGw,mBw);
GetPixelRGB(ImgHasil,mRhw,mGhw,mBhw);

//cek seimbang panjang,lebar matriks
if h_a >= w_a then z := h_a else z := w_a;
mR := Matriks_seimbang(mR,h_a,w_a,z);
mG := Matriks_seimbang(mG,h_a,w_a,z);
mB := Matriks_seimbang(mB,h_a,w_a,z);
mRw := Matriks_seimbang(mRw,h_w,w_w,z);
mGw := Matriks_seimbang(mGw,h_w,w_w,z);
mBw := Matriks_seimbang(mBw,h_w,w_w,z);
mRhw := Matriks_seimbang(mRhw,h_a,w_a,z);
mGhw := Matriks_seimbang(mGhw,h_a,w_a,z);
mBhw := Matriks_seimbang(mBhw,h_a,w_a,z);
mUwr := Matriks_seimbang(mUwr,sz_h,sz_w,z);
mUwg := Matriks_seimbang(mUwg,sz_h,sz_w,z);
mUwb := Matriks_seimbang(mUwb,sz_h,sz_w,z);
mVtwr := Matriks_seimbang(mVtwr,sz_h,sz_w,z);
mVtwg := Matriks_seimbang(mVtwg,sz_h,sz_w,z);
mVtwb := Matriks_seimbang(mVtwb,sz_h,sz_w,z);

//dekomposisi matriks image asli U S Vt
RMatrixSVD(mr,z,z,2,2,2,Sr,mUr,mVtr);
RMatrixSVD(mg,z,z,2,2,2,Sg,mUg,mVtg);
RMatrixSVD(mb,z,z,2,2,2,Sb,mUb,mVtb);
mSr := matrikssingular(Sr,z);
mSg := MatriksSingular(Sg,z);
mSb := MatriksSingular(Sb,z);

//EKSTRAK DEKOMPOSISI NILAI Aw
RMatrixSVD(mRhw,z,z,2,2,2,s_wr,u_,v_);
RMatrixSVD(mGhw,z,z,2,2,2,s_wg,u_,v_);
RMatrixSVD(mBhw,z,z,2,2,2,s_wb,u_,v_);
sw_wr := MatriksSingular(s_wr,z);
sw_wg := MatriksSingular(s_wg,z);
sw_wb := MatriksSingular(s_wb,z);
```

```
//St = UwSw'Vtw
    stwr := PerkalianMatrix(mUwr,sw_wr,z,z,z);
    stwg := PerkalianMatrix(mUwg,sw_wg,z,z,z);
    stwb := PerkalianMatrix(mUwb,sw_wb,z,z,z);
    stwr := PerkalianMatrix(stwr,mVtwr,z,z,z);
    stwg := PerkalianMatrix(stwg,mVtwg,z,z,z);
    stwb := PerkalianMatrix(stwb,mVtwb,z,z,z);

    //W=St-S/alfa
    wr :=
PembagianMatrix(PenguranganMatriks(stwr,mSr,z,z),alfa,z
,z);
    wg :=
PembagianMatrix(PenguranganMatriks(stwg,mSg,z,z),alfa,z
,z);
    wb :=
PembagianMatrix(PenguranganMatriks(stwb,mSb,z,z),alfa,z
,z);

    //tampil hasil

Formekstraksiwatermark.Image1.Picture.Bitmap.PixelForma
t:=pf24bit;

    Formekstraksiwatermark.ClientHeight := h_w;
    Formekstraksiwatermark.ClientWidth := w_w;
    for i := 0 to h_w-1 do
    begin
      for j := 0 to w_w-1 do
      begin

       Formekstraksiwatermark.Image1.Picture.Bitmap.Canv
       as.Pixels[j,i] :=
       RGB(tobyte(round(wr[i,j])),tobyte(round(wg[i,j]))
       ,tobyte(round(wb[i,j])));
      end;
    end;
```

Gambar 4.3 *Source Code* Ekstraksi *watermark*

4.2.3 Implementasi *Normalized Crosscorelation* (NC)

Proses *normalized Crosscorelation* digunakan untuk melakukan autentikasi logo *watermark* hasil ekstraksi dengan logo *watermark* asli. Proses ini membutuhkan inputan berupa citra logo *watermark* asli dan citra logo *watermark* hasil ekstraksi Fungsi untuk menghitung nilai korelasi ditunjukkan pada Gambar 4.4.

```
function NC(Img1,Img2:TBitmap):double;
var i,j,m,n:integer;
        NC,NCr,NCg,NCb,Nr,Ng,Nb:real;
    aR,aG,aB,bR,bG,bB:TReal2DArray;
begin
      m:=Img1.width;
      n:=Img1.height;
  GetPixelRGB(Img1,aR,aG,aB);
  GetPixelRGB(Img2,bR,bG,bB);
  SetLength(aR,m,n);
  SetLength(aG,m,n);
  SetLength(aB,m,n);
  SetLength(bR,m,n);
  SetLength(bG,m,n);
  SetLength(bB,m,n);
  begin
    NC:=0;
    NCr:=0;NCg:=0;NCb:=0;
    Nr:=0;Ng:=0;Nb:=0;
    for i:=0 to m-1 do
      for j:=0 to n-1 do
      begin
        NCr:=NCr+(aR[i,j]*bR[i,j]);
        NCg:=NCg+(aG[i,j]*bG[i,j]);
        NCb:=NCb+(aB[i,j]*bB[i,j]);
        Nr :=Nr+sqr(aR[i,j]);
        Ng :=Ng+sqr(aG[i,j]);
        Nb :=Nb+sqr(aB[i,j]);
      end;
      end;
  NC:=(NCr+NCg+NCb)/(Nr+Ng+Nb);
  Result:=NC;
end;
```

Gambar 4.4 *Source Code function NC*

4.2.4 Implementasi MSE

Proses MSE digunakan untuk menghitung tingkat kesalahan antara citra asli dengan citra terkompresi. Oleh karena itu, proses ini membutuhkan inputan berupa dimensi citra, citra asli dan citra terkompresi. Prosedur untuk menghitung MSE ditunjukkan pada Gambar 4.5.

```
function MSE(Img1,Img2:TBitmap):double;
var i,j,m,n:integer;
    MSE,MSEr,MSEg,MSEb:real;
    aR,aG,aB,bR,bG,bB:TReal2DArray;
begin
  m:=Img1.width;
  n:=Img1.height;
  GetPixelRGB(Img1,aR,aG,aB);
  GetPixelRGB(Img2,bR,bG,bB);
  SetLength(aR,m,n);
  SetLength(aG,m,n);
  SetLength(aB,m,n);
  SetLength(bR,m,n);
  SetLength(bG,m,n);
  SetLength(bB,m,n);
  begin
    MSE:=0;
    MSEr:=0;
    MSEg:=0;
    MSEb:=0;
    for i:=0 to m-1 do
      for j:=0 to n-1 do
      begin
        MSEr:=MSEr+sqr(aR[i,j]-bR[i,j]);
        MSEg:=MSEg+sqr(aG[i,j]-bG[i,j]);
        MSEb:=MSEb+sqr(aB[i,j]-bB[i,j]);
      end;
  end;
      MSE:=(MSEr+MSEg+MSEb)/3;
      MSE:=MSE/(m*n);
      Result:=MSE;
end;
```

Gambar 4.5 *Source Code procedure* MSE

4.2.5 Implementasi PSNR (*Peak Signal to Noise Ratio*)

PSNR digunakan untuk menghitung perbedaan intensitas piksel antara citra terkompresi dengan citra asli. Semakin besar nilai PSNR maka semakin baik citra terkompresi. Fungsi PSNR ditunjukkan pada Gambar 4.6.

```
function
PSNR(GambarAsal,GambarTransformasi:TBitmap):double;
var MSEret:double;
begin
  MSEret:=MSE(GambarAsal,GambarTransformasi);
  PSNR:=20*Log10(255/sqrt(MSEret));
end;
```

Gambar 4.6 *Source Code function* PSNR

4.3 Implementasi Antarmuka (*interface*)

Berdasarkan rancangan antarmuka pada subbab 3.4 dihasilkan antarmuka proses *watermarking* pada citra. Gambar 4.7 adalah tampilan antarmuka utama penyisipan *watermark*. Pada proses penyisipan *watermark,* user perlu menginputkan citra utama / *host* dan citra logo yang akan disisipkan seperti yang ditampilkan pada gambar 4.7.

Gambar 4.7 Antarmuka Penyisipan *Watermark*

Pada proses ekstraksi user perlu menginputkan citra yang telah disisipi *watermark,* citra asli dan citra logo *watermark* untuk proses perhitungan nilai korelasi apakah citra logo yang telah diekstraksi cocok atau tidak dengan citra logo asli. Tampilan untuk proses ekstraksi ditunjukkan pada gambar 4.8.

Gambar 4.8 Antarmuka Ekstraksi *Watermark*

4.4 Implementasi Uji Coba

Pada subbab ini akan dibahas mengenai pengujian yang telah dilakukan pada sistem dan evaluasi dari hasil yang dikeluarkan oleh sistem.

4.4.1 Skenario Pengujian

Pengujian terhadap perangkat lunak dibagi menjadi tiga, yaitu pengujian kebenaran perangkat lunak, pengujian kinerja perangkat lunak, dan pengujian ketahanan *watermark* terhadap serangan-serangan. Dalam pengujian ini, semua berkas citra BMP mempunyai kedalaman 24 bit dan resolusi sebesar 72 dpi. Citra uji yang akan digunakan dapat dilihat pada lampiran 1.

Tabel 4.1 Daftar Berkas Citra

No.	Nama Berkas	Resolusi (Piksel)
1	Fabregas.bmp	275 x 275
2	Flower.bmp	400 x 300
3	Garden.bmp	275 x 275
4	Baboon.bmp	150 x 150

Pada penelitian ini logo *watermark* yang akan disisipkan pada citra asli/*host* adalah Logo.bmp berukuran 180 x 180 piksel.

4.4.2 Hasil Pengujian

Berdasarkan perancangan pengujian yang telah dijelaskan pada bab 3.3, maka hasil pengujian beserta analisisnya akan dijelaskan menjadi beberapa subbab, yaitu hasil pengujian kebenaran sistem, hasil pengujian kinerja sistem, serta hasil pengujian ketahanan *watermark* terhadap serangan-serangan.

4.4.2.1 Hasil Uji Kebenaran Sistem

Pengujian penyisipan *watermark* dinyatakan berhasil apabila proses penyisipan tidak mengalami pesan kegagalan eksekusi dari sistem karena ukuran citra *host* yang lebih kecil daripada ukuran citra logo yang akan disisipkan. Proses ekstraksi dinyatakan berhasil apabila nilai korelasi antara logo hasil ekstraksi dengan logo asli tidak kurang dari 0.5. Pada tabel 4.2 merupakan hasil pengujian kebenaran sistem.

Tabel 4.2 Hasil pengujian kebenaran sistem

No	Citra *Host*	Citra *Watermark*	α	Hasil Penyisipan	Hasil Ekstraksi
1	Fabregas.bmp	Logo.bmp	0.1	Berhasil	Berhasil
2	Fabregas.bmp	Logo.bmp	0.5	Berhasil	Berhasil
3	Fabregas.bmp	Logo.bmp	1	Berhasil	Berhasil
4	Flower.bmp	Logo.bmp	0.1	Berhasil	Berhasil
5	Flower.bmp	Logo.bmp	0.5	Berhasil	Berhasil
6	Flower.bmp	Logo.bmp	1	Berhasil	Berhasil

7	Garden.bmp	Logo.bmp	0.1	Berhasil	Berhasil
8	Garden.bmp	Logo.bmp	0.5	Berhasil	Berhasil
9	Garden.bmp	Logo.bmp	1	Berhasil	Berhasil
10	Baboon.bmp	Logo.bmp	0.1	Gagal	-
11	Baboon.bmp	Logo.bmp	0.5	Gagal	-
12	Baboon.bmp	Logo.bmp	1	Gagal	-

Hasil uji menunjukkan bahwa sistem *image watermarking* yang telah dibuat pada skripsi ini telah memenuhi kebutuhan sistem yang telah dijelaskan pada bab 3. Hal ini dibuktikan dengan keberhasilan sistem dalam melakukan proses penyisipan *watermark* dan proses pendeteksian *watermark* / ekstraksi tanpa mengalami pesan kegagalan dan *watermark* dapat dideteksi sesuai dengan *watermark* yang disisipkan pada ketiga citra *host*. Pada percobaan menggunakan citra baboon.bmp program mengalami proses kegagalan dikarenakan ukuran citra *host* yang lebih kecil daripada citra logo *watermark* yang akan disisipkan.

4.4.2.2 Hasil Uji Dan Analisis Kinerja Sistem

Pengujian kinerja sistem dilakukan dengan mengukur nilai PSNR dari perbandingan citra *host* dan citra yang telah disisipi *watermark* serta nilai MSE dari perbandingan citra logo asli dengan logo hasil ekstraksi.

Tabel 4.3 Hasil pengujian kinerja sistem

No	Citra *Host*	Citra *Watermark*	α	PSNR Citra Ter-*watermark*	MSE Logo Hasil Ekstraksi	Nilai Korelasi
1	Fabregas.bmp	Logo.bmp	0.1	40.70	0.22	0.98
2	Fabregas.bmp	Logo.bmp	0.5	22.23	6.54	0.85
3	Fabregas.bmp	Logo.bmp	1	15.43	20.67	0.79
4	Flower.bmp	Logo.bmp	0.1	43.66	0.03	0.99
5	Flower.bmp	Logo.bmp	0.5	24.46	1.73	0.93
6	Flower.bmp	Logo.bmp	1	17.29	19.33	0.74
7	Garden.bmp	Logo.bmp	0.1	40.05	0.01	0.99
8	Garden.bmp	Logo.bmp	0.5	20.16	10.26	0.81
9	Garden.bmp	Logo.bmp	1	14.10	37.08	0.64

Hasil pengujian kinerja sistem seperti pada tabel 4.3 menunjukkan bahwa rata-rata nilai PSNR citra ter-*watermark* dibanding dengan citra *host* pada nilai alfa 0.1 adalah diatas 40 db atau dengan kata lain citra tersebut layak berdasarkan penggolongan nilai PSNR seperti dijelaskan pada bab 2.4.1. Semakin tinggi nilai alfa yang digunakan maka semakin rendah nilai PSNR nya dan semakin tinggi nilai MSE logo hasil ekstraksi. Nilai PSNR yang rendah bila nilai alfa semakin tinggi disebabkan karena semakin kentalnya kekuatan logo *watermark* yang disisipkan ke dalam citra *host*, sehingga citra ter-*watermark* menjadi tampak semakin buram bila nilai alfa ditinggikan.

Pada tabel 4.3 ditunjukkan bahwa semakin tinggi nilai α maka semakin rendah nilai korelasi dari logo hasil ekstraksi. Rendahnya nilai korelasi ini dapat ditunjukkan pula secara visual dengan banyaknya *noise* pada pixel-pixel logo hasil ekstraksi atau adanya perubahan warna.

4.4.2.3 Hasil Uji Dan Analisis Ketahanan Citra Ber-*Watermark*

Pengujian ketahanan sistem terhadap serangan dinyatakan berhasil apabila citra yang telah disisipi *watermark* dan yang telah diberi serangan berupa beberapa efek *Blur, noise, pixelate, sharpening* tersebut dapat terdeteksi setelah dilakukan proses ekstraksi. Nilai alfa yang digunakan pada pengujian ini adalah 0.1 dan 1. Berikut ini adalah tabel hasil pengujian ketahanan sistem pada citra fabregas.bmp ditunjukkan pada tabel 4.4

Tabel 4.4 Hasil pengujian ketahanan sistem citra fabregas.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Fabregas.bmp	0.1	*Motion Blur* (5)	Berhasil	0.89	15.39
2	Fabregas.bmp	0.1	*Motion Blur* (20)	Berhasil	0.65	54.74
3	Fabregas.bmp	0.1	*Motion Blur* (50)	Gagal	-	-
4	Fabregas.bmp	0.1	*Noise* (10)	Berhasil	0.95	9.17
5	Fabregas.bmp	0.1	*Noise* (20)	Berhasil	0.92	32.19
6	Fabregas.bmp	0.1	*Noise* (50)	Berhasil	0.87	123.30

7	Fabregas.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.85	25.23
8	Fabregas.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.59	74.92
9	Fabregas.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Fabregas.bmp	0.1	*Sharpening*	Berhasil	0.98	5.63
11	Fabregas.bmp	1	*Motion Blur* (5)	Berhasil	0.77	24.06
12	Fabregas.bmp	1	*Motion Blur* (20)	Berhasil	0.74	32.75
13	Fabregas.bmp	1	*Motion Blur* (50)	Berhasil	0.69	46.08
14	Fabregas.bmp	1	*Noise* (10)	Berhasil	0.78	22.42
15	Fabregas.bmp	1	*Noise* (20)	Berhasil	0.77	25.74
16	Fabregas.bmp	1	*Noise* (50)	Berhasil	0.73	42.06
17	Fabregas.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.77	25.35
18	Fabregas.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.73	35.54
19	Fabregas.bmp	1	*Pixelate Mozaic* (30)	Berhasil	0.68	49.80
20	Fabregas.bmp	1	*Sharpening*	Berhasil	0.79	20.65

Tabel 4.5 Hasil pengujian ketahanan sistem citra Flower.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Flower.bmp	0.1	*Motion Blur* (5)	Berhasil	0.93	8.03
2	Flower.bmp	0.1	*Motion Blur* (20)	Berhasil	0.73	38.70
3	Flower.bmp	0.1	*Motion Blur* (50)	Gagal	-	-
4	Flower.bmp	0.1	*Noise* (10)	Berhasil	0.95	16.15

5	Flower.bmp	0.1	*Noise* (20)	Berhasil	0.93	58.75
6	Flower.bmp	0.1	*Noise* (50)	Berhasil	0.88	162.03
7	Flower.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.89	16
8	Flower.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.63	63.89
9	Flower.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Flower.bmp	0.1	*Sharpening*	Berhasil	0.99	4.23
11	Flower.bmp	1	*Motion Blur* (5)	Berhasil	0.73	21.22
12	Flower.bmp	1	*Motion Blur* (20)	Berhasil	0.69	28.89
13	Flower.bmp	1	*Motion Blur* (50)	Berhasil	0.61	44.15
14	Flower.bmp	1	*Noise* (10)	Berhasil	0.73	22.49
15	Flower.bmp	1	*Noise* (20)	Berhasil	0.71	29.06
16	Flower.bmp	1	*Noise* (50)	Berhasil	0.63	59.23
17	Flower.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.73	22.31
18	Flower.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.68	32.22
19	Flower.bmp	1	*Pixelate Mozaic* (30)	Berhasil	0.61	47.01
20	Flower.bmp	1	*Sharpening*	Berhasil	0.74	19.76

Tabel 4.6 Hasil pengujian ketahanan sistem citra Garden.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Garden.bmp	0.1	*Motion Blur* (5)	Berhasil	0.87	17.92
2	Garden.bmp	0.1	*Motion Blur* (20)	Berhasil	0.71	49.75
3	Garden.bmp	0.1	*Motion Blur* (50)	Berhasil	0.55	79.93

4	Garden.bmp	0.1	*Noise* (10)	Berhasil	0.98	4.11
5	Garden.bmp	0.1	*Noise* (20)	Berhasil	0.96	12.86
6	Garden.bmp	0.1	*Noise* (50)	Berhasil	0.90	82.55
7	Garden.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.80	34.06
8	Garden.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.57	79.60
9	Garden.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Garden.bmp	0.1	*Sharpening*	Berhasil	0.99	6.33
11	Garden.bmp	1	*Motion Blur* (5)	Berhasil	0.62	42.05
12	Garden.bmp	1	*Motion Blur* (20)	Berhasil	0.58	50.02
13	Garden.bmp	1	*Motion Blur* (50)	Berhasil	0.54	62.57
14	Garden.bmp	1	*Noise* (10)	Berhasil	0.63	39.08
15	Garden.bmp	1	*Noise* (20)	Berhasil	0.62	43.07
16	Garden.bmp	1	*Noise* (50)	Berhasil	0.57	62.59
17	Garden.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.60	47.02
18	Garden.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.53	61.97
19	Garden.bmp	1	*Pixelate Mozaic* (30)	Gagal	-	-
20	Garden.bmp	1	*Sharpening*	Berhasil	0.66	37.69

Hasil pengujian ketahanan sistem seperti pada tabel 4.4, tabel 4.5 dan tabel 4.6 menunjukkan bahwa sistem *watermark* menggunakan SVD ini telah berhasil mendeteksi adanya *watermark* meskipun telah diberi serangan-serangan berupa *blur, noise, pixelate,* dan *sharpening* , meskipun pada beberapa efek serangan dengan kadar yang tinggi tersebut tidak dapat mendeteksi adanya *watermark*.

Pada tabel 4.5 dan 4.6 baris nomor 19 dengan citra uji flower.bmp dan garden.bmp yang diberi efek serangan berupa

pixelate mozaic (30) dan nilai α sebesar 1 menghasilkan hasil yang berbeda saat dilakukan ekstraksi padahal dengan nilai α dan efek serangan yang sama. Pada citra uji flower.bmp sistem dapat mendeteksi adanya *watermark*, sedangkan pada citra uji garden.bmp sistem tidak mendeteksi adanya *watermark*. Hal ini membuktikan bahwa nilai piksel suatu citra juga sangat mempengaruhi perhitungan dengan metode *singular value decomposition.*

Semakin kecil nilai MSE maka semakin kecil kadar error dari perbandingan citra logo asli dengan logo hasil ekstraksi. Semakin mendekati angka 1 pada nilai korelasi citra logo asli dengan logo hasil ekstraksi, maka semakin mirip perbandingan nilai piksel pada kedua citra tersebut. Pada tabel 4.4, tabel 4.5 dan tabel 4.6 terlihat bahwa apabila nilai α semakin besar maka nilai MSE semakin besar pula, namun secara kasat mata / *visual,* logo hasil ekstraksi lebih mudah dideteksi apabila nilai alfa semakin tinggi atau dengan kata lain tingkat kekuatan *watermark* lebih tangguh apabila nilai alfa semakin besar.

Pengujian penyisipan *watermark* dengan nilai alfa kurang dari 0.5 menghasilkan nilai PSNR yang cukup memuaskan. Dari beberapa pengujian pada bab 4.3.2.2 menunjukkan bahwa nilai alfa yang layak digunakan adalah tidak lebih dari 0.5 karena citra yang telah disisipi *watermark* dengan nilai alfa lebih dari 0.5 menghasilkan nilai PSNR yang rendah, atau citra kurang layak untuk digunakan karena citra ter-*watermark* tersebut berbeda cukup signifikan dibanding citra *host* atau citra aslinya.

Pada pengujian ketahanan sistem menghasilkan hasil yang sangat baik, yaitu *watermark* yang telah disisipkan kedalam citra dapat dikembalikan atau diekstrak kembali dengan baik meskipun citra yang disisipi *watermark* telah dimodifikasi. Pada proses penyisipan *watermark,* nilai alfa yang semakin rendah maka semakin baik tingkat *invisibility watermark* atau tingkat ketersembunyian logo *watermark* pada citra, namun tingkat *robustness watermark* atau tingkat kekuatan *watermark* terhadap serangan semakin menurun. Begitu juga sebaliknya, apabila nilai alfa yang semakin tinggi maka semakin buruk tingkat *invisibility watermark* atau tingkat ketersembunyian logo *watermark* pada citra, namun tingkat *robustness watermark* atau tingkat kekuatan *watermark* terhadap serangan semakin tinggi.


UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Pemasalahan penyisipan *watermark* pada citra berwarna dapat diaplikasikan dengan menggunakan metode *singular value decomposition* (SVD) dengan menggabungkan matriks citra logo dengan matriks singular dari citra *host.*.

2. Besarnya PSNR (*Peak Signal Noise to Ratio*), dan MSE (*Mean Square Error*) bergantung pada besarnya nilai alfa yang digunakan saat proses penyisipan *watermark*. Pada citra fabregas.bmp dengan nilai alfa 0.1 dihasilkan nilai PSNR dan MSE masing-masing 40.70 dan 0.22, saat nilai alfa 0.5 menghasilkan nilai PSNR dan MSE masing-masing 22.23 dan 6.54, saat nilai alfa 1 menghasilkan nilai PSNR dan MSE masing-masing 15.43 dan 20.67. Dari analisa hasil tersebut dapat disimpulkan semakin tinggi nilai alfa yang digunakan maka semakin kecil nilai PSNR dan semakin besar nilai MSE.

3. Nilai korelasi pada citra logo hasil ekstraksi dipengaruhi oleh kekuatan serangan modifikasi citra ter-*watermark*. Pada percobaan citra fabregas.bmp dengan nilai alfa 0.1 dan diberi serangan *motion blur* (5) menghasilkan nilai korelasi 0.89, saat kekuatan serangan dinaikkan menjadi 20 / *motion blur* (20) menghasilkan nilai korelasi 0.65. Dari analisa hasil tersebut dapat disimpulkan bahwa semakin kuat efek serangan modifikasi citra ter-*watermark,* maka semakin menurun nilai korelasi dari citra logo hasil ekstrasi begitu pula sebaliknya.

4. Permasalahan ketahanan citra ter-*watermark* terhadap serangan-serangan modifikasi *image* seperti *blur, pixelate, noise, sharpening* dapat dideteksi dengan metode *singular value decomposition* ini. Pada proses penyisipan *watermark* pada citra fabregas.bmp dengan nilai alfa 0.1 kemudian diberi efek serangan berupa

motion blur (50) setelah diberi *watermark* kemudian dilakukan proses ekstraksi, hasilnya *watermark* tidak terdeteksi. Sedangkan dengan citra yang sama dan efek serangan yang sama namun dengan nilai alfa 1, hasilnya logo *watermark* dapat terdeteksi. Dari hasil tersebut dapat disimpulkan bahwa nilai alfa yang semakin rendah maka semakin baik tingkat *invisibility watermark* atau tingkat ketersembunyian logo *watermark* pada citra, namun tingkat *robustness watermark* atau tingkat kekuatan *watermark* terhadap serangan semakin menurun. Begitu juga sebaliknya, apabila nilai alfa yang semakin tinggi maka semakin buruk tingkat *invisibility watermark* atau tingkat ketersembunyian logo *watermark* pada citra, namun tingkat *robustness watermark* atau tingkat kekuatan *watermark* terhadap serangan semakin tinggi.

5.2 Saran

Metode lain dengan membagi matriks citra menjadi beberapa blok dapat digunakan untuk mempersingkat waktu proses penyisipan dan ekstrasi dalam ukuran citra yang lebih besar.

DAFTAR PUSTAKA

Ahmad, Balza dan Firdausy, Kartika. 2005. *Teknik Pengolahan Citra Digital Menggunakan Delphi*. Ardi Publishing. Yogyakarta.

Basaruddin, T., 2006, *Dekomposisi Nilai Singulir dan Penerapannya*, Jurnal Ilmu Komputer dan Teknologi Informasi, Vol. 6 No 2

Chang, C, Lin, C and Hu, Y "An SVD Oriented Watermark Embedding Scheme With High Qualities For The Restored Images", *International Journal of Innovative Computing, Information and Control*, Volume 3, Number 3, June 2007

Garcia, Dr.Edel. 2006. Singular Value Decomposition (SVD) A Fast Track Tutorial.

Gonzalez, R.C. dan R. E. Woods. 2005. *Digital Image Processing*, 2nd ed., Prentice-Hall Inc., New Jersey.

Gonzalez, R. C. dan R. E. Woods. 2002. *Digital Image Processing*. Prentice Hall, New Jersey.

Liu, Jie, Niu Xiamu and Kong, Wenhai. 2006. Image Watermarking based on Singular Value Decomposition. Shenyang Normal University

Rajab, Lama, "Video Watermarking Algorithms Using the SVD Transform", *European Journal of Scientific Research* ISSN 1450-216X Vol.30 No.3 (2009), pp.389-401

Umran, Munzir dan Abidin, Taufik. 2009. *Pengelompokkan Dokumen Menggunakan K-Means dan Singular Value Decomposition : Studi Kasus Menggunakan Data Blog*. Jurusan Matematika FMIPA, Universitas Syiah Kuala-Banda Aceh.


UNIVERSITAS BRAWIJAYA

LAMPIRAN

Hasil Pengujian

Citra Asli

Fabregas.bmp

Citra Asli

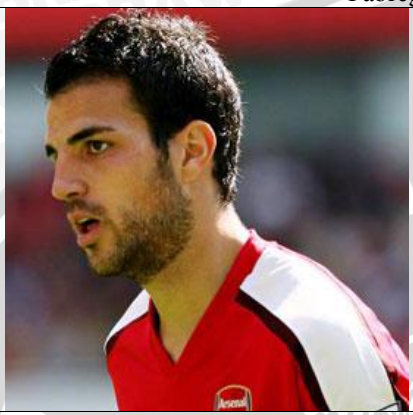
Flower.bmp

Citra Asli

Garden.bmp

Citra Asli

Baboon.bmp

Fabregas.bmp	
	
α = 0.1	α = 1

Flower.bmp	
	
α = 0.1	α = 1

Garden.bmp	
	
α = 0.1	α = 1

Data hasil uji fabregas.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Fabregas.bmp	0.1	*Motion Blur* (5)	Berhasil	0.89	15.39
2	Fabregas.bmp	0.1	*Motion Blur* (20)	Berhasil	0.65	54.74
3	Fabregas.bmp	0.1	*Motion Blur* (50)	Gagal	-	-
4	Fabregas.bmp	0.1	*Noise* (10)	Berhasil	0.95	9.17
5	Fabregas.bmp	0.1	*Noise* (20)	Berhasil	0.92	32.19
6	Fabregas.bmp	0.1	*Noise* (50)	Berhasil	0.87	123.30
7	Fabregas.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.85	25.23
8	Fabregas.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.59	74.92
9	Fabregas.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Fabregas.bmp	0.1	*Sharpening*	Berhasil	0.98	5.63
11	Fabregas.bmp	1	*Motion Blur* (5)	Berhasil	0.77	24.06
12	Fabregas.bmp	1	*Motion Blur* (20)	Berhasil	0.74	32.75
13	Fabregas.bmp	1	*Motion Blur* (50)	Berhasil	0.69	46.08
14	Fabregas.bmp	1	*Noise* (10)	Berhasil	0.78	22.42
15	Fabregas.bmp	1	*Noise* (20)	Berhasil	0.77	25.74
16	Fabregas.bmp	1	*Noise* (50)	Berhasil	0.73	42.06
17	Fabregas.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.77	25.35
18	Fabregas.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.73	35.54
19	Fabregas.bmp	1	*Pixelate Mozaic* (30)	Berhasil	0.68	49.80
20	Fabregas.bmp	1	*Sharpening*	Berhasil	0.79	20.65

Data hasil uji flower.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Flower.bmp	0.1	*Motion Blur* (5)	Berhasil	0.93	8.03
2	Flower.bmp	0.1	*Motion Blur* (20)	Berhasil	0.73	38.70
3	Flower.bmp	0.1	*Motion Blur* (50)	Gagal	-	-
4	Flower.bmp	0.1	*Noise* (10)	Berhasil	0.95	16.15
5	Flower.bmp	0.1	*Noise* (20)	Berhasil	0.93	58.75
6	Flower.bmp	0.1	*Noise* (50)	Berhasil	0.88	162.03
7	Flower.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.89	16
8	Flower.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.63	63.89
9	Flower.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Flower.bmp	0.1	*Sharpening*	Berhasil	0.99	4.23
11	Flower.bmp	1	*Motion Blur* (5)	Berhasil	0.73	21.22
12	Flower.bmp	1	*Motion Blur* (20)	Berhasil	0.69	28.89
13	Flower.bmp	1	*Motion Blur* (50)	Berhasil	0.61	44.15
14	Flower.bmp	1	*Noise* (10)	Berhasil	0.73	22.49
15	Flower.bmp	1	*Noise* (20)	Berhasil	0.71	29.06
16	Flower.bmp	1	*Noise* (50)	Berhasil	0.63	59.23
17	Flower.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.73	22.31
18	Flower.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.68	32.22
19	Flower.bmp	1	*Pixelate Mozaic*(30)	Berhasil	0.61	47.01
20	Flower.bmp	1	*Sharpening*	Berhasil	0.74	19.76

Data hasil uji garden.bmp

No	Citra Ter-*watermark*	α	Jenis Serangan	Hasil Ekstraksi	Nilai Korelasi	MSE
1	Garden.bmp	0.1	*Motion Blur* (5)	Berhasil	0.87	17.92
2	Garden.bmp	0.1	*Motion Blur* (20)	Berhasil	0.71	49.75
3	Garden.bmp	0.1	*Motion Blur* (50)	Berhasil	0.55	79.93
4	Garden.bmp	0.1	*Noise* (10)	Berhasil	0.98	4.11
5	Garden.bmp	0.1	*Noise* (20)	Berhasil	0.96	12.86
6	Garden.bmp	0.1	*Noise* (50)	Berhasil	0.90	82.55
7	Garden.bmp	0.1	*Pixelate Mozaic* (5)	Berhasil	0.80	34.06
8	Garden.bmp	0.1	*Pixelate Mozaic* (15)	Berhasil	0.57	79.60
9	Garden.bmp	0.1	*Pixelate Mozaic* (30)	Gagal	-	-
10	Garden.bmp	0.1	*Sharpening*	Berhasil	0.99	6.33
11	Garden.bmp	1	*Motion Blur* (5)	Berhasil	0.62	42.05
12	Garden.bmp	1	*Motion Blur* (20)	Berhasil	0.58	50.02
13	Garden.bmp	1	*Motion Blur* (50)	Berhasil	0.54	62.57
14	Garden.bmp	1	*Noise* (10)	Berhasil	0.63	39.08
15	Garden.bmp	1	*Noise* (20)	Berhasil	0.62	43.07
16	Garden.bmp	1	*Noise* (50)	Berhasil	0.57	62.59
17	Garden.bmp	1	*Pixelate Mozaic* (5)	Berhasil	0.60	47.02
18	Garden.bmp	1	*Pixelate Mozaic* (15)	Berhasil	0.53	61.97
19	Garden.bmp	1	*Pixelate Mozaic* (30)	Gagal	-	-
20	Garden.bmp	1	*Sharpening*	Berhasil	0.66	37.69