PERANCANGAN DAN PEMBUATAN APLIKASI PENANGANAN DISTORSI PADA STEGANOGRAFI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun Oleh:

RIZAL ARIF ZULFIKAR

NIM. 0710630086 - 63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2011

LEMBAR PENGESAHAN

PERANCANGAN DAN PEMBUATAN APLIKASI PENANGANAN DISTORSI PADA STEGANOGRAFI

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

RIZAL ARIF ZULFIKAR

NIM. 0710630086-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 23 Desember 2010
DOSEN PENGUJI

Ir. Muhammad Aswin., MT.	**R. Arief Setyawan, ST., MT.**
NIP. 19640626 199002 1 001	**NIP. 19750819 199903 1 001**

M. Aziz Muslim, ST., MT., Ph.D.

NIP. 19741203 200012 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS

NIP. 19580728 198701 1 001

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN APLIKASI PENANGANAN DISTORSI PADA STEGANOGRAFI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun oleh:

RIZAL ARIF ZULFIKAR
NIM. 0710630086 - 63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adharul Muttaqin, ST., MT.	**Mochammad Rifan, ST., MT.**
NIP. 19760121 200501 1 001	**NIP. 19710301 200012 1 001**

KATA PENGANTAR

Puji dan Syukur penulis panjatkan ke hadirat Allah SWT, yang tela*h* melimpahkan segala petunjuk dan rahmat-Nya sehingga penulis dapat menyelesaikan kewajiban menyusun tugas akhir ini. Penyusunan tugas akhir ini merupakan salah satu syarat untuk menyelesaikan program pendidikan Sarjana Strata-1 (S1) Jurusan Teknik *Elektro Universitas Brawijaya Malang*. Adapun judul dari laporan tugas akhir ini yaitu **"PERANCANGAN DAN PEMBUATAN APLIKASI PENANGANAN DISTORSI PADA STEGANOGRAFI".**

Penulis menyadari bahwa dalam penyusunan laporan tugas akhir ini masih terdapat banyak kekurangan, baik dalam materi maupun teknisnya. Hal ini disebabkan pengetahuan dan kemampuan penulis yang terbatas. Karenanya, dengan rendah hati penulis menerima kritik dan saran dari pembaca yang sifatnya membangun dan memperbaiki sehingga dapat mengarah kepada penyusunan yang lebih baik.

Penulis juga menyadari bahwa penulisan laporan tugas akhir ini tidak dapat diselesaikan dengan baik tanpa adanya bantuan dan kerja sama dari berbagai pihak. Pada kesempatan ini penulis mengucapkan banyak terima kasih kepada Bapak, Ibu serta keluarga yang selalu mendo'akan dan senantiasa memberikan dorongan, mengingatkan dan mendukung baik moril maupun materiil dalam berbagai aktifitas selama ini, serta kasih sayang dan pengertian yang telah mereka berikan kepada penulis, sehingga dapat memberi makna didalam hidup penulis, khususnya penulis dapat menyelesaikan skripsi ini. Serta tidak lupa penulis mengucapkan terima kasih kepada yang terhormat :

1. Bapak Adharul Muttaqin, ST., MT. Dan Bapak Mochammad Rif'an, ST., MT. selaku dosen pembimbing yang telah memberikan bimbingan dan pengarahan-pengarahan dalam proses penyusunan tugas akhir ini, sehingga dapat menambah ilmu bagi penulis.
2. Bapak Dr. Ir. Sholeh Hadi Pramono, MS. selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya Malang.
3. Bapak M. Aziz Muslim, ST., MT., Ph.D. selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya Malang.

4. Seluruh staff pengajar dan karyawan Jurusan Teknik Elektro Universitas Brawijaya Malang.
5. Oktavia Nur Rakhman yang selalu mendo'akan dan senantiasa memberikan dorongan, mengingatkan dan mendukung dalam penyelesaian laporan ini.
6. Untuk teman-teman seperjuanganku Pandu Pradito, Rahmatullah Adi, Rachmania Nur D, Titis Hayuning, Ita Dwi P, Suci Imani P, serta seluruh warga besar CORE terima kasih atas bantuan dan kerjasamanya selama ini.
7. Serta semua pihak yang tidak bisa penulis sebutkan satu persatu, yang telah membantu terselesaikannya laporan tugas akhir ini.

Akhir kata, semoga Allah SWT melimpahkan semua rahmat dan karunia kepada semua pihak yang telah membantu terselesaikannya Tugas Akhir ini. Dan semoga Tugas Akhir ini dapat bermanfaat bagi para pembacanya.

Malang, 24 November 2011

Penulis

ABSTRAK

Rizal Arif Z, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Oktober 2011, *Perancangan dan Pembuatan Aplikasi Penanganan Distorsi Pada Steganografi,* Dosen Pembimbing : Adharul Muttaqin, ST., MT. dan Mochammad Rif'an, ST., MT.

Steganografi adalah teknik penyisipan informasi pada suatu media. Salah satu metode penyisipan tersebut adalah Less Significant Bit (LSB) Insertion. Pada metode ini dilakukan penyisipan di setiap LSB dari biner tiap pixel pada media berupa citra 2D. Metode ini sangat rentan terhadap gangguan atau distorsi seperti rotasi, horizontal flip, dan vertical flip. Sekecil apapun distorsi yang diberikan pada citra 2D akan menyebabkan perubahan susunan posisi pixel. Terjadinya perubahan tersebut pada citra pembawa pesan akan mengakibatkan pesan yang telah disisipkan tidak dapat dibaca kembali. Karena itulah dibutuhkan sistem steganografi yang dapat mengatasi masalah distorsi tersebut.

Salah satu cara agar steganografi lebih tahan terhadap distorsi adalah dengan menggunakan metode penandaan yang dilakukan saat proses penyisipan. Pada tahap proses ekstraksi pesan, penanda digunakan untuk mengembalikan posisi pixel seperti pada saat proses penyisipan. Pengujian dilakukan pada 9 macam citra dengan 45 variasi distorsi dan 6 variasi ukuran pesan. Hasilnya adalah 93,33% dari citra pengujian dapat diambil kembali pesan sisipannya.

Kata kunci: *Steganografi, LSB* Insertion*, Distorsi, Rotasi,* Flip*,* Nearest Interpolation.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penyisipan informasi terhadap suatu data sering dijumpai dalam kehidupan sehari-hari. Penyisipan dapat dilakukan pada berbagai media data, seperti file dokumen, gambar, suara, bahkan video. Penyisipan dilakukan dengan berbagai alasan misalkan untuk mengetahui keaslian data, penyampaian informasi rahasia dan sebagainya.

Teknik penyisipan informasi ini sering dikenal dengan nama steganografi dan umumnya menggunakan metode *Less Significant Bit Insertion* (LSBI). Namun seiring berkembangnya waktu, semakin banyak masyarakat yang mengetahui metode penyisipan gambar dan apa saja kelemahanya. Sehingga data yang disisipkan mudah dirusak atau dimanipulasi oleh pihak ketiga.

Dalam penyisipan informasi terhadap citra 2D (gambar) terdapat salah satu kelemahan yaitu terhadap distorsi. Distorsi yang dimaksud adalah rotasi dan pembalikan gambar. Hanya dengan memberikan sedikit distorsi, maka data yang disisipkan tidak akan dapat terbaca karena terjadi perubahan posisi pixel. Sehingga perlu adanya metode penanganan distorsi terhadap citra steganografi.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang, maka ditemukan beberapa rumusan masalah sebagai berikut :

1. Bagaimana cara mengatasi masalah distorsi dan mengembalikan citra ke posisi sebenarnya.
2. Bagaimana menerjemahkan kode steganografi untuk mendapatkan informasi yang ada pada citra yang sudah diperbaiki tersebut.

.

1.3 Batasan Masalah

Beberapa batasan masalah yang ada pada skripsi ini adalah :

1. Distorsi yang dipakai dalam hal ini adalah *Horizontal flip*, *Vertical flip*, dan Rotasi kelipatan 1 derajat.
2. Latar belakang atau *background* citra saat mengalami rotasi adalah warna hitam.
3. Tipe citra yang dipakai adalah citra bitmap atau *lossless image*.

1.4 Tujuan

Tujuan penyusunan skripsi ini adalah :

1. Untuk merancang dan membuat sebuah aplikasi yang dapat menulis dan membaca kode yang disisipkan pada citra (steganografi).
2. Menangani salah satu masalah steganografi yakni distorsi dalam hal ini *Horizontal flip*, *Vertical flip*, dan rotasi

1.5 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat.

BAB II Dasar Teori

Menjelaskan kajian pustaka dan dasar teori yang digunakan pada skripsi ini.

BAB III Metodologi

Menjelaskan metode yang digunakan dalam pengerjaan skripsi berikut langkah – langkah yang akan diambil.

BAB IV Perancangan dan Implementasi

Menjelaskan langkah langkah perancangan dan hasil implementasi aplikasi penanganan distorsi pada steganografi berikut penjelasan algoritma yang digunakan dan penjelasan dari setiap langkah – langkahnya.

BAB V Pengujian dan Analisis

Membahas pengujian dari sistem yang telah dibuat berikut analisa dari hasil pengujian sehingga dapat diketahui kelebihan dan kekurangan sistem.

BAB VII Kesimpulan dan Saran

Berisi Kesimpulan hasil penulisan dan Saran untuk pengembangan sistem selanjutnya.

BAB II
Dasar Teori

Bab ini menjelaskan tentang kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi dan melandasi pelaksanan penelitian sistem pendukung keputusan penerima beasiswa.

2.1 Pengolahan Citra Digital

Pengolahan citra digital merupakan proses yang bertujuan untuk memanipulasi dan menganalisis citra dengan bantuan komputer. Pengolahan citra digital dapat digunakan untuk Mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis dengan ilmu pengetahuan pole (*pattern recognition*) yang umumnya bertujuan mengenali suatu objek dengan cara mengekstrak informasi penting yang terdapat pada suatu citra.

Bila pengenalan pola dihubungkan dengan pengolahan citra, diharapkan akan terbentuk suatu sistem yang dapat memproses citra masukan sehingga citra tersebut dapat dikenali polanya. Proses ini disebut pengenalan citra atau *image recognition*. Proses pengenalan citra ini sering diterapkan dalam kehidupan sehari-hari.

Sebuah citra diartikan sebagai sebuah fungsi kontinyu intensitas cahaya dalam bidang dua dimensi f(x,y) dengan koordinat “x” dan “y”. Setiap titik yang pada bidang ini “x” dan “y” dinyatakan dengan “f” yang merepresentasikan intensitas cahaya atau informasi warna citra.

Dalam citra digital, citra yang akan diproses diubah dalam bentuk diskrit dengan jarak yang sama tiap titik – titik terkecilnya yang disebut dengan *picture element* atau *pixel*. Citra ini dapat berupa citra bitmap yang mempunyai fungsi persamaan – persamaan matematis dari bentuk dasar yang membentuk citra tersebut .

Pada hakekatnya citra yang dilihat mata kita sendiri terdiri dari berkas-berkas cahaya yang dipantulkan oleh benda-benda sekitar kita. Jadi fungsi intensitas f(x,y)merupakan fungsi sumber cahaya i(x, y) yang menerangi

objek serta jumlah cahaya yangdipantulkan, r(x, y) oleh objek, dengan demikian f(x, y) dapat dinyatakan denganpersamaan :

f(x, y) =i(x, y)r(x,y)

dengan : 0 < i(x, y) < ~ (dominasi sumbercahaya)

0 < r(x, y) < 1 (koefisien pantul cahaya).

Semakin terang sumber cahaya, maka nilai iluminasi sumber semakin besar dan semakin terang warna suatu objek maka koefisien pantul objek tersebut semakin besar. Citra digital merupakan array dua dimensi dengan nilai f(x,y) nya telah dikonversi ke dalam bentuk diskrit baik pada koordinat citra maupun kecerahannya.

Citra digital adalah sebuah citra f(x,y) yang telah didiskretasi ke dalam koordinat special dan tingkat keabuan. Citra digital dinyatakan sebagai sebuah matrik NxN yang terdiri atas baris dan kolom untuk menyatakan sebuah titik pada citra dan elemen nilai matrik yang berupa nilai diskrit menyatakan tingkat keabuan pada titik tersebut.Citra digital tiap elemen dikenal sebagai *picture elemen t*atau pixel.

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

Gambar 2.1 Representasi citra digital dalam bentuk matrik

2.2. Citra BMP

Bitmap adalah representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. Dikembangkan oleh Microsoft dan nilai setiap titik diawali oleh satu bit data untuk gambar hitam putih, atau lebih bagi gambar berwarna.

Ukuran sebenarnya untuk n-bit (2^n warna) bitmap dalam byte dapat dihitung:

$$\textbf{ukuran file BMP} \approx 54 + 4 \cdot 2^n + \frac{lebar \cdot tinggi \cdot n}{8}$$

dimana tinggi dan lebar dalam pixel.

Kerapatan titik-titik tersebut dinamakan resolusi, yang menunjukkan seberapa tajam gambar ini ditampilkan, ditunjukkan dengan jumlah baris dan kolom, contohnya 1024x768.

Untuk menampilkan citra *bitmap* pada monitor atau mencetaknya pada printer, komputer menterjemahkan *bitmap* ini menjadi pixel (pada layar) atau titik tinta (pada printer). Beberapa format file bitmap yang populer adalah BMP, PCX dan TIFF.

2.3. Steganografi

Steganografi adalah seni dan ilmu menulis pesan tersembunyi atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Sebaliknya, kriptografi menyamarkan arti dari suatu pesan, tapi tidak menyembunyikan bahwa ada suatu pesan. Kata "steganografi" berasal dari bahasa Yunani *steganos*, yang artinya "tersembunyi atau terselubung", dan *graphein*, "menulis".

Kini, istilah steganografi termasuk penyembunyian data digital dalam berkas-berkas (*file*) komputer. Contohnya, si pengirim mulai dengan berkas gambar biasa, lalu mengatur warna setiap pixel ke-100 untuk menyesuaikan suatu huruf dalam alphabet (perubahannya begitu halus sehingga tidak ada seorangpun yang menyadarinya jika ia tidak benar-benar memperhatikannya).

Pada umumnya, pesan steganografi muncul dengan rupa lain seperti gambar, artikel, daftar belanjaan, atau pesan-pesan lainnya. Pesan yang tertulis ini merupakan tulisan yang menyelubungi atau menutupi. Contohnya, suatu pesan bisa disembunyikan dengan menggunakan tinta yang tidak terlihat di antara garis-garis yang kelihatan.

Teknik steganografi meliputi banyak sekali metode komunikasi untuk menyembunyikan pesan rahasia (teks atau gambar) di dalam berkas-berkas lain yang mengandung teks, *image*, bahkan audio tanpa menunjukkan ciri-ciri perubahan yang nyata atau terlihat dalam kualitas dan struktur dari berkas semula. Metode ini termasuk tinta yang tidak tampak, *microdots*,

pengaturan kata, tanda tangan digital, jalur tersembunyi dan komunikasi spektrum lebar.

Tujuan dari steganografi adalah merahasiakan atau menyembunyikan keberadaan dari sebuah pesan tersembunyi atau sebuah informasi. Dalam prakteknya, kebanyakan pesan disembunyikan dengan membuat perubahan tipis terhadap data digital lain yang isinya tidak akan menarik perhatian dari penyerang potensial, sebagai contoh sebuah gambar yang terlihat tidak berbahaya. Perubahan ini bergantung pada kunci (sama pada kriptografi) dan pesan untuk disembunyikan. Orang yang menerima gambar kemudian dapat menyimpulkan informasi terselubung dengan cara mengganti kunci yang benar ke dalam algoritma yang digunakan.

Pada metode steganografi cara ini sangat berguna jika digunakan pada cara steganografi komputer karena banyak format berkas digital yang dapat dijadikan media untuk menyembunyikan pesan. Format yang biasa digunakan di antaranya:

- Format *image* : bitmap (bmp), gif, pcx, jpeg, dll.
- Format audio : wav, voc, mp3, dll.
- Format lain : teks file, html, pdf, dll.

Kelebihan steganografi jika dibandingkan dengan kriptografi adalah pesan-pesannya tidak menarik perhatian orang lain. Pesan-pesan berkode dalam kriptografi yang tidak disembunyikan, walaupun tidak dapat dipecahkan, akan menimbulkan kecurigaan. Seringkali, steganografi dan kriptografi digunakan secara bersamaan untuk menjamin keamanan pesan rahasianya.

Sebuah pesan steganografi (*plaintext*), biasanya pertama-tama dienkripsikan dengan beberapa arti tradisional, yang menghasilkan *ciphertext*. Kemudian, *covertext* dimodifikasi dalam beberapa cara sehingga berisi *ciphertext*, yang menghasilkan *stegotext*. Contohnya, ukuran huruf, ukuran spasi, jenis huruf, atau karakteristik *covertext* lainnya dapat dimanipulasi untuk membawa pesan tersembunyi; hanya penerima (yang harus mengetahui teknik yang digunakan) dapat membuka pesan dan mendekripsikannya.

2.3.1 Metode Steganografi

Kebanyakan algoritma steganografi menggunakan sebuah kombinasi dari bidang jenis teknik untuk melakukan sebuah tugas dalam penyelubungan pesan rahasia dalam sebuah selubung berkas. Sebuah program steganografi dibutuhkan untuk melakukan hal-hal berikut (baik implisit melalui suatu perkiraan maupun eksplisit melalui sebuah perhitungan), menemukan kelebihan bits dalam selubung file yang dapat digunakan untuk menyelubungi pesan rahasia didalamnya, memilih beberapa diantaranya untuk digunakan dalam menyelubungi data dan penyelubungan data dalam bits dipilih sebelumnya.

2.3.1.1 *Least Significant Bit Insertion* (LSBI)

Metoda yang digunakan untuk menyembunyikan pesan pada media digital tersebut berbeda-beda. Contohnya, pada berkas image pesan dapat disembunyikan dengan menggunakan cara menyisipkannya pada bit rendah atau bit yang paling kanan (LSB) pada data pixel yang menyusun file tersebut. Pada berkas bitmap 24 bit, setiap pixel (titik) pada gambar tersebut terdiri dari susunan tiga warna merah, hijau dan biru (RGB) yang masing-masing disusun oleh bilangan 8 bit (byte) dari 0 sampai 255 atau dengan format biner 00000000 sampai 11111111. Dengan demikian, pada setiap pixel berkas bitmap 24 bit kita dapat menyisipkan 3 bit data.

Kekurangan dari LSB Invertion: Dapat diambil kesimpulan dari contoh 8 bit pixel, menggunakan LSB Insertion dapat secara drastis mengubah unsur pokok warna dari pixel. Ini dapat menunjukkan perbedaan yang nyata dari *cover image* menjadi *stego image*, sehingga tanda tersebut menunjukkan keadaan dari steganografi. Variasi warna kurang jelas dengan 24 bit image, bagaimanapun file tersebut sangatlah besar. Antara 8 bit dan 24 bit *image* mudah diserang dalam pemrosesan image,

seperti *cropping* (pemotongan)
dan *compression*(pemampatan).

Keuntungan dari LSB Insertion : Keuntungan yang paling besar dari algoritma LSB ini adalah cepat dan mudah. Dan juga algoritma tersebut memiliki *software* steganografi yang mendukung dengan bekerja di antara unsur pokok warna LSB melalui manipulasi *pallete* (lukisan).

2.3.1.2 *Algorithms and Transformation*

Algoritma *compression* adalah metode steganografi dengan menyembunyikan data dalam fungsi matematika. Dua fungsi tersebut adalah *Discrete Cosine Transformation* (DCT) dan*Wavelet Transformation*. Fungsi DCT dan Wavelet yaitu mentransformasi data dari satu tempat (domain) ke tempat (domain) yang lain. Fungsi DCT yaitu mentransformasi data dari tempat spatial (*spatial domain*) ke tempat frekuensi (*frequency domain*).

2.3.1.3 *Redundant Pattern Encoding*

Redundant Pattern Encoding adalah menggambar pesan kecil pada kebanyakan gambar. Keuntungan dari metode ini adalah dapat bertahan dari *cropping* (pemotongan). Kerugiannya yaitu tidak dapat menggambar pesan yang lebih besar.

2.3.1.4 *Spread Spectrum method*

Spread Spectrum steganografi terpencar-pencar sebagai pesan yang diacak (*encrypted*) melalui gambar (tidak seperti dalam LSB). Untuk membaca suatu pesan, penerima memerlukan algoritma yaitu *crypto-key* dan *stego-key*. Metode ini juga masih mudah diserang yaitu penghancuran atau pengrusakan dari kompresi dan proses *image* (gambar).

2.3.2 Kriteria Steganografi

2.3.2.1 *Impercepbility.*

Keberadaan pesan tidak dapat dipersepsi oleh indrawi. Jika pesan disisipkan ke dalam sebuah citra, citra yang telah disisipi pesan harus tidak dapat dibedakan dengan citra asli oleh mata. Begitu pula dengan suara, telinga haruslah mendapati perbedaan antara suara asli dan suarayang telah disisipi pesan.

2.2.2.2 *Fidelity.*

Mutu media penampung tidak berubah banyak akibat penyisipan. Perubahan yang terjadi harus tidak dapat dipersepsi oleh indrawi.

2.2.2.3 *Recovery.*

Pesan yang disembunyikan harus dapat diungkap kembali. Tujuan steganografi adalah menyembunyikan informasi, maka sewaktu-waktu informasi yang disembunyikan ini harus dapat diambil kembali untuk dapat digunakan lebih lanjut sesuai keperluan.

2.4. Rotasi

Rotasi merupakan suatu transformasi geometri memindahkan nilai-nilai pixel dari posisi awal menuju ke posisi akhir yang ditentukan melalui nilai variable rotasi sebesar θ^o (teta derajat) terhadap sudut 0(derajat) atau garis horizontal pada citra

Proses rotasi dapat dilakukan dengan rumus sebagai berikut.

$$x_2 = \cos(\theta) \times (x_1 - x_0) - \sin(\theta) \times (y_1 - y_0) + x_0$$

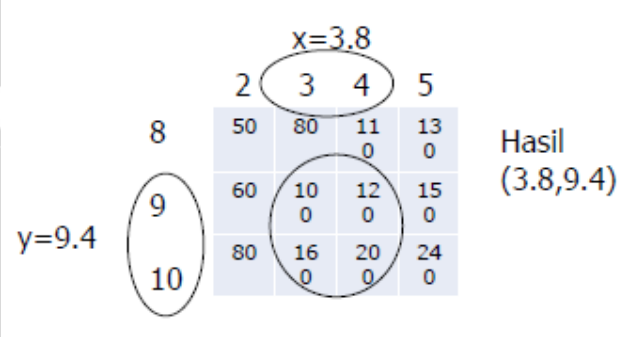
$$y_2 = \sin(\theta) \times (x_1 - x_0) + \cos(\theta) \times (y_1 - y_0) + y_0$$

Di mana (x_0, y_0) adalah koordinat titik pusat dari citra input dan θ adalah sumbu putar. Sumbu putar pada umumnya memiliki arah putar searah jarum jam dengan garis horizontal. Seperti halnya operasi translas, hasil perhitungan posisi hasil rotasi dapat memberikan nilai di luar batas output (apabila ukuran citra output sama dengan citra input). Untuk kasus seperti itu, ada beberapa implementasi yang membiarkan nilai pixel tersebut tanpa

dipetakan ulang dan ada yang memetakan ke citra output sehingga menyebabkan ukuran citra membesar.

2.4.1 *Nearest Interpolation*

Nilai piksel keluaran ditetapkan dari nilai piksel dari suatu titik yang ditentukan letak posisinya yang terdekat. Adapun piksel-piksel yang lain tidak diperhitungkan. Pada Citra 2D, di pilih 4 titik asal yang saling berbatasan satu sama lain.



Gambar 2.2 Gambar contoh *Nearest interpolation*

Misal diperoleh koord (3.8, 9.4), maka titik terdekatnya yang mungkin adalah (3,9), (3,10), (4,9), (4,10) dan yang dipakai adalah (4,9). Sehingga nilai Go(x',y') sama dengan Gi(4,9) yaitu 120.

2.5. *Horizontal flip*

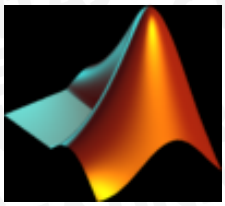
Metode dalam *horizontal flip* adalah dengan menyusun ulang urutan kolom dalam sebuah citra. Kolom pertama diletakkan pada kolom terakhir lalu kolom kedua pada kolom kedua terakhir, seterusnya hingga kolom terakhir pada citra asli menempati kolom pertama pada citra hasil. *Horizontal flip* tidak mengakibatkan perubahan ukuran citra, dan tidak mengakibatkan hilangnya data pixel pada citra asli.

2.6. *Vertical flip*

Metode dalam *vertical flip* adalah dengan menyusun ulang urutan baris dalam sebuah citra. Baris pertama diletakkan pada baris terakhir lalu baris kedua pada baris kedua terakhir, seterusnya hingga baris terakhir pada citra asli menempati baris pertama pada citra hasil. *Vertical flip* tidak mengakibatkan perubahan ukuran citra, dan tidak mengakibatkan hilangnya

data pixel pada citra asli.

2.7. MATLAB



MatLab adalah sebuah lingkungan komputasi numerikal dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks, MATLAB memungkinkan manipulasi matriks, pem-plot-an fungsi dan data, implementasi algoritma, pembuatan antarmuka pengguna, dan peng-antarmuka-an dengan program dalam bahasa lainnya. Meskipun hanya bernuansa numerik, sebuah kotak kakas (*toolbox*) yang menggunakan mesin simbolik MuPAD, memungkinkan akses terhadap kemampuan aljabar komputer. Sebuah paket tambahan, Simulink, menambahkan simulasi grafis multiranah dan Desain Berdasar-Model untuk sistem terlekat dan dinamik.

Pada tahun 2004, MathWorks mengklaim bahwa MATLAB telah dimanfaatkan oleh lebih dari satu juta pengguna di dunia pendidikan dan industri.

2.5.1 Sejarah

MATLAB (yang berarti "*matrix laboratory*") diciptakan pada akhir tahun 1970-an oleh Cleve Moler, yang kemudian menjadi Ketua Departemen Ilmu Komputer di Universitas New Mexico. Ia merancangnya untuk memberikan akses bagi mahasiswa dalam memakai LINPACK dan EISPACK tanpa harus mempelajari Fortran. Karyanya itu segera menyebar ke universitas-universitas lain dan memperoleh sambutan hangat di kalangan komunitas matematika terapan. Jack Little, seorang insinyur, dipertemukan dengan karyanya tersebut selama kunjungan Moler ke Universitas Stanford pada tahun 1983. Menyadari potensi komersialnya, ia bergabung dengan Moler dan Steve Bangert. Mereka menulis ulang MATLAB dalam bahasa pemrograman C, kemudian mendirikan The MathWorks pada tahun 1984 untuk melanjutkan pengembangannya. Pustaka yang ditulis ulang tadi kini dikenal dengan nama JACKPAC. Pada tahun 2000,

MATLAB ditulis ulang dengan pemakaian sekumpulan pustaka baru untuk manipulasi matriks, LAPACK

MATLAB pertama kali diadopsi oleh insinyur rancangan kontrol (yang juga spesialisasi Little), tapi lalu menyebar secara cepat ke berbagai bidang lain. Kini juga digunakan di bidang pendidikan, khususnya dalam pengajaran aljabar linear dan analisis numerik, serta populer di kalangan ilmuwan yang menekuni bidang pengolahan citra.

BAB III
METODOLOGI

Bab ini menjelaskan mengenai langkah-langkah yang dilakukan untuk membuat aplikasi penanganan distorsi pada steganografi. Metode penelitian yang digunakan dalam penyusunan skripsi ini adalah

3.1 Studi Literatur

Studi literatur berguna untuk memperoleh data dan menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. Mempelajari *image processing*.
2. Mempelajari Steganografi.
3. Mempelajari algoritma – algoritma *Rotation Transform*.
4. Mempelajari bahasa pemrograman *MatLab*.

3.2 Diagram Sistem

Citra yang akan menjadi input dari aplikasi adalah citra digital dari berbagai variasi warna dan ukuran. Contoh dari citra yang dimaksud dapat dilihat pada Gambar 3.1.

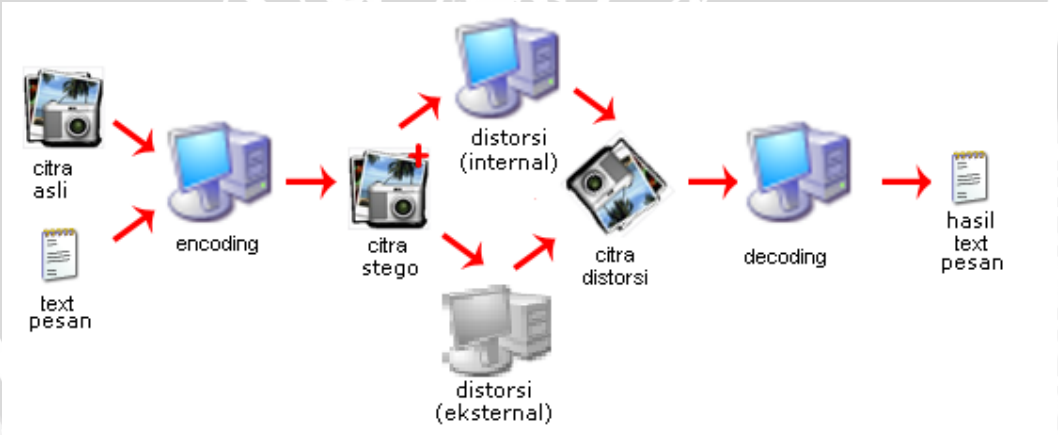
Gambar 3.1 contoh citra input

Komputer akan memproses masukan citra dengan metode dalam steganografi yaitu LSBI (*Less Significant Bit Insertion*) sehingga didapat hasil keluaran citra dengan informasi sisipan. Hasil output memiliki perubahan yang tidak dapat dipersepsi oleh sistem visual manusia.

Citra input yang telah disisipi informasi akan diberikan gangguan berupa distorsi sehingga posisi pixel semula akan berubah. Pada Gambar 3.2 adalah beberapa tampilan citra yang telah mendapat distorsi

Gambar 3.2 contoh citra input steganografi

Komputer akan memproses masukan yang telah disisipkan informasi dan diberi distorsi dengan menggunakan metode dalam pemrosesan digital sehingga didapatkan informasi sisipan yang diinginkan. Hasil dari pengolahan ini akan disimpan di dalam komputer itu sendiri dalam bentuk teks informasi sisipan. Diagram sistem tersebut dapat dilihat pada Gambar 3.3.

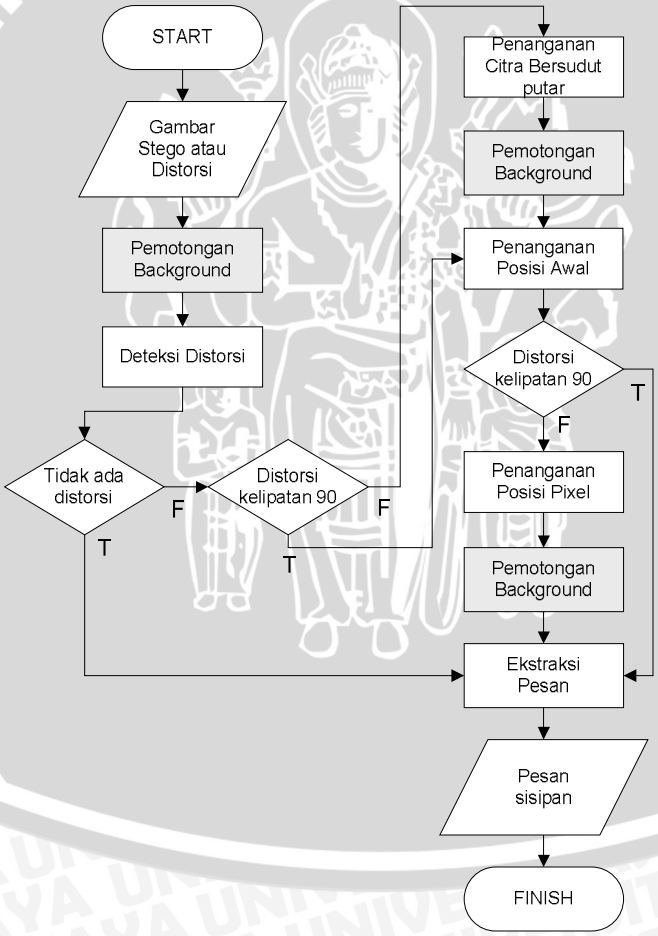


Gambar 3.3 diagram sistem

3.3 Cara Kerja Sistem

Berikut adalah flowchart dari cara kerja sistem yang akan dibuat :

Gambar3.4 *flowchart* cara penyisipan informasi



Gambar 3.5 *flowchart* cara ekstraksi informasi

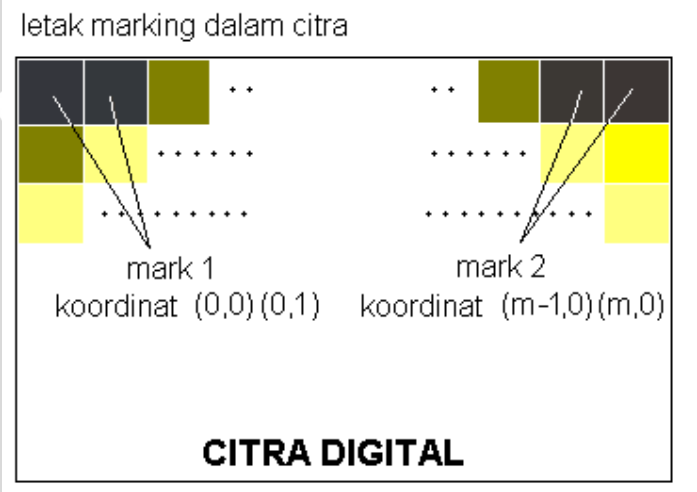
3.3.1 *Marking* (penanda)

Proses awal adalah dengan proses marking image atau penandaan pada gambar. Marking berfungsi untuk menandai citra agar

diketahui kemiringan yang terjadi jika citra di rotasi. Sistem marking adalah memberikan dua buah penanda yang diletakkan sejajar. Jika citra dirotasi, maka letak penanda berubah dan tidak sejajar lagi. Dari posisi kedua penanda yang tidak sejajar tersebut dilakukan perhitungan sehingga didapat besar sudut rotasi. Marking diletakkan pada pojok atas kiri dan kanan dengan nilai sebagai berikut,

Mark 1 :	R = 0011 0100 (52)	Mark 2 : R = 0011 1100 (60)
	G = 0011 1000 (56)	G = 0011 1000 (56)
	B = 0011 1100 (60)	B = 0011 0100 (52)

Misal ukuran data adalah M x N pixel maka *mark*1 berada pada koordinat (0,0) dan (0,1), *mark*2 berada pada koordinat (m-1,0) dan (m,0).



Gambar 3.6 Gambar letak *mark* 1 dan *mark* 2

3.3.2 Penulisan Data Steganografi

Penulisan data dilakukan dengan metode LSBI (*Least Significant Bit Insertion*). Pada metode ini data yang akan disisipkan diletakkan pada bit paling kanan dari bit warna RGB. Data yang disisipkan terlebih dahulu diubah ke dalam kode ASCII sehingga data berbentuk biner. Berikut adalah logika dalam penyisipan data.

Jika bit data = LSB citra maka	maka	LSB citra tetap
Jika bit data < LSB citra maka	maka	LSB citra dikurangi 1
Jika bit data > LSB citra maka	maka	LSB citra ditambah 1

3.3.3 Penulisan Bit Posisi

Bit posisi adalah bit yang digunakan untuk mengembalikan posisi bit yang meleset karena kesalahan saat melakukan interpolasi terdekat. Bit posisi diletakkan pada bit kedua terakhir dengan ketentuan.

Pada Baris: Perulangan tiap 3 Baris
- Baris 1 : nilai R bit ke-7 = 1
- Baris 2 : nilai G bit ke-7 = 1
- Baris 3 : nilai B bit ke-7 = 1

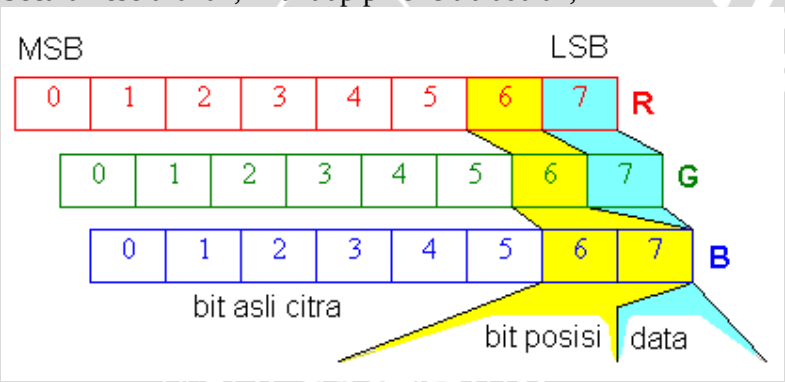
Pada Kolom: Perulangan tiap 3 Kolom
- Kolom 1 : nilai B bit ke-8 = 1

Kolom 2 : nilai B bit ke-7 = 1
Kolom 3 : nilai G bit ke-7 = 1
Serta terdapat pengecualian pada baris ke-3 dan kolom ke-3
Nilai B bit ke-8 =1
Sehingga nilai bit ke-7 dan ke-8 tiap baris dan kolom sebagai berikut:

	Kolom 1				Kolom 2				Kolom 3			
	R7	G7	B7	B8	R7	G7	B7	B8	R7	G7	B7	B8
Baris1	1	0	0	1	1	0	1	0	1	1	0	0
Baris2	0	1	0	1	0	1	1	0	0	1	0	0
Baris3	0	0	1	1	0	0	1	0	0	1	1	1

Tabel 3.1 Tabel penulisan Bit Posisi

Secara keseluruhan, nilai tiap pixel citra adalah,



Gambar 3.7 Skema penulisan sisipan tiap pixel

3.3.4 Rotasi

Proses rotasi dapat dilakukan dengan rumus sebagai berikut.

$$x_2 = \cos(\theta) \times (x_1 - x_0) - \sin(\theta) \times (y_1 - y_0) + x_0$$

$$y_2 = \sin(\theta) \times (x_1 - x_0) + \cos(\theta) \times (y_1 - y_0) + y_0$$

Rotasi dapat dilakukan didalam maupun diluar sistem. Rotasi diluar sistem dapat menggunakan program seperti image paint, adobe, corel, dll.

3.3.5 Penanganan Rotasi (Citra Bersudut putar)

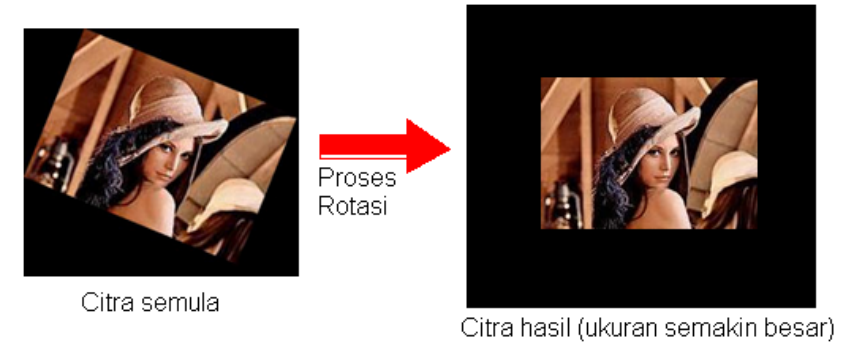
Langkah pertama dalam menangani rotasi adalah dengan mencari koordinat mark 1 dan mark 2. setelah koordinat kedua mark diketahui maka dicari besar sudut dengan rumus.

Mark 1 (x_a, y_a) dan **Mark 2** (x_b, y_b)

$$\theta = \arctan\left(|x_b - x_a| / |y_b - y_a| \right)$$

θ dilakukan pendekatan nilai kelipatan 1 (bilangan bulat).

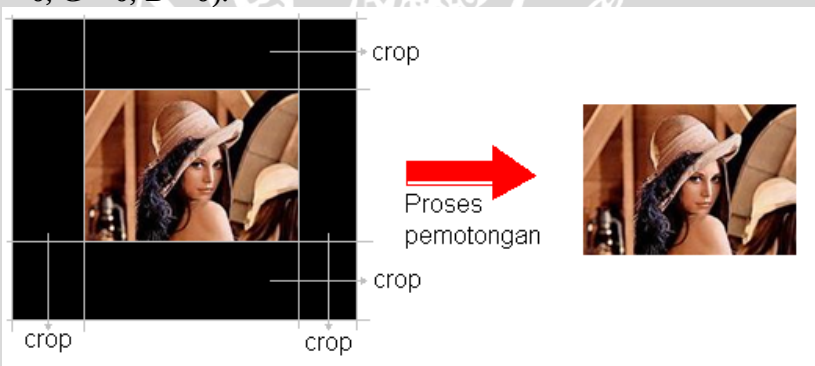
Setelah nilai sudut didapat maka gambar diputar sesuai sudut θ. Hasil keluaran dari citra yang diputar akan mengalami perubahan ukuran menjadi semakin besar. Hal ini dikarenakan untuk mengantisipasi hilangnya data citra yang terpotong saat rotasi dilakukan.



Gambar 3.8 Contoh hasil rotasi

3.3.6 Pemotongan

Citra yang telah diputar kemudian dipotong agar didapat ukuran yang mendekati gambar asli. Pemotongan dilakukan dengan cara menghilangkan kolom atau baris yang hanya berisi warna hitam (nilai R = 0, G = 0, B = 0).



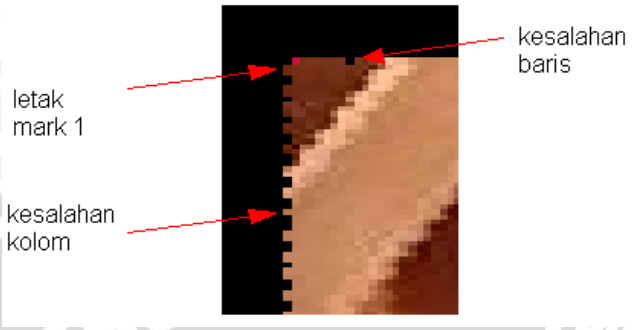
Gambar 3.9 Gambar Pemotongan

3.3.7 Penanganan Rotasi (Posisi awal citra)

Cari mark 1 di sekitar ujung-ujung citra. Putar gambar sehingga letak mark 1 berada di pojok kiri atas atau disekitar koordinat (0,0). Cari juga mark 2 disekitar pojok kiri bawah citra. Jika ada, maka lakukan transpose matrik pada citra.

3.3.8 Penanganan Kesalahan Posisi

Kesalahan posisi terjadi karena melakukan rotasi dengan menggunakan interpolasi terdekat. Kelebihan interpolasi terdekat adalah tidak mengubah nilai pixel yang diputar. Namun kelemahannya adalah saat menentukan posisi, terkadang posisi terdekat yang dipakai meleset dari posisi sebenarnya.



Gambar 3.10 Contoh hasil *nearest interpolation*

Kesalahan posisi mengalami penggeseran sebesar 1 pixel. Saat penyisipan data steganogafi, data ke-7 dan ke-8 setiap byte telah disisipkan bit posisi untuk menangani masalah kesalahan posisi. Setiap 3 pixel dalam baris maupun kolom memiliki bit ke-7 dan ke-8 yang berbeda. Bit ini akan dicocokkan dengan *key position* (KP) yang nilainya berulang. Pixel pertama yang di periksa adalah pixel mark 1. jika nilai tidak sesuai pixel akan dipindah ke segala arah dengan jarak 1 pixel. Setelah itu dilakukan pemotongan (*crop*) sekali lagi sehingga didapat ukuran asli citra.

3.3.9 Ekstraksi Kode

Setelah citra kembali ke posisi asal dapat dilakukan pengambilan informasi yang telah disisipkan. Proses pembacaan informasi dilakukan dengan mengambil nilai LSB dari tiap pixel. Kemudian hasil pembacaan yang berupa bahasa ASCII dirubah kedalam bentuk alfabet dan disimpan ke dalam komputer.

Contoh:

Pixel 1	Pixel 2	Pixel 3
1001 010**1**	1011 111**1**	1111 001**1**
0001 001**1**	0000 001**0**	0101 010**1**
1011	1100	0011

Dari data disamping maka informasi:

Akan didapat

110 101 111

Gambar 3.11 gambaran ekstraksi

3.4 Pengujian Sistem

Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang dirancang telah sesuai dengan yang diinginkan dan meminimalisir error yang terjadi pada sistem. Pengujian yang dilakukan meliputi variasi gambar yang akan disisipi informasi, dan variasi distorsi yang mungkin terjadi pada citra yang telah disisipi informasi.

3.5 Kesimpulan dan Saran

Pada tahap ini, diambil dari hasil pengujian dan analisa terhadap aplikasi yang telah dibuat. Tahap Selanjutnya adalah membuat saran untuk perbaikan terhadap penelitian selanjutnya sehingga dapat menyempurnakan kekurangan-kekurangan yang ada dan mengembangkan hasil yang diperoleh dari skripsi ini.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan perancangan dan implementasi perangkat lunak dari aplikasi penanganan distorsi pada steganografi. Implementasi menggunakan bahasa pemrograman Matlab versi 7.7 tahun 2008.

4.1 Blok Diagram Sistem

Blok diagram sistem menggambarkan garis besar kinerja sistem dalam penanganan distorsi pada steganografi.

Gambar 4.1 Diagram Blok Sistem

Berikut adalah penjelasan tiap blok diagram pada Gambar 4.1,

1. Citra Asli

   Citra asli adalah citra awal sebagai masukan yang akan disisipi oleh pesan. Ekstensi file citra dapat berupa bmp, gif, jpg, atau tif yang akan diubah ke dalam matriks citra RGB 8 bit. Posisi pada citra asli akan dipakai sebagai acuan posisi awal jika citra mendapat distorsi posisi.

2. *Text* Pesan

   Text pesan adalah teks yang akan disisipkan pada citra asli. Masukan teks diketikkan dalam sistem. Panjang teks yang akan disisipkan tergantung pada besar ukuran pixel citra asli.

3. *Encoding*

   Encoding adalah sistem yang melakukan penyisipan pesan kedalam citra asli. Masukan sistem berupa citra asli dan pesan text. Keluaran sistem berupa citra stego berekstensi bmp.

4. Citra Stego

   Citra stego adalah citra yang didalamnya terdapat pesan sisipan. Ekstensi file citra stego berupa file bmp.

5. Distorsi (*internal*)

   Distorsi ini dilakukan didalam sistem. Masukan system berupa citra stego, dan keluarannya berupa citra distorsi. Jenis distorsi yang tersedia adalah rotasi berbagai sudut, *horizontal flip* dan *vertical flip*. Pada rotasi, *background* citra secara otomatis akan terisi warna hitam murni (R=0,G=0,B=0).

6. Distorsi (*external*)

   Distorsi ini dilakukan diluar sistem, misal photoshop, corel, gimp atau program pengolahan citra lainnya.

7. Citra Distorsi

   Citra Distorsi adalah citra stego yang telah mengalami distorsi. Citra distorsi dalam sistem ini hanya memperbolehkan jenis distorsi berupa rotasi berbagai sudut dengan *background* warna hitam murni (R=0,G=0,B=0), *horizontal flip* dan *vertical flip*. Ekstensi citra distorsi berupa file bmp.

8. *Decoding*

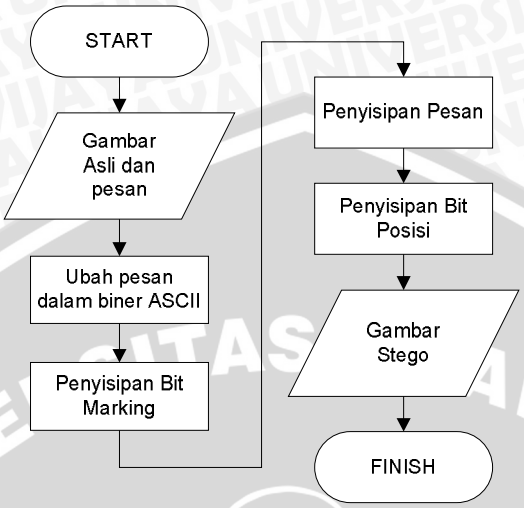
   Decoding adalah sistem yang melakukan ekstraksi pesan terhadap citra distorsi. Pixel citra distorsi yang memiliki pesan tersisip akan dikembalikan keposisi semula sebelum mendapat distorsi. Pixel yang telah kembali kemudian diekstrak sesuai urutan dan disusun kembali menjadi sebuah pesan. Masukan sistem berupa file bmp dan keluaran berupa teks.

9. Hasil *Text* Pesan

   Hasil *Text* pesan adalah pesan yang didapat setelah proses *decoding*.

4.2 Perancangan dan Implementasi Sistem *Encoding*

Sistem *encoding* berfungsi untuk menyisipkan pesan berupa teks kedalam sebuah citra. Diagram sistem penyisipan pesan terdapat pada Gambar 4.2



Gambar 4.2 Diagram Sistem Encoding

4.2.1 Masukan Sistem *Encoding*

Masukan sistem *encoding* berupa Gambar Asli dengan ekstensi bmp, jpg, gif, png atau tif dan teks pesan. Gambar akan diubah dalam bentuk matriks citra dengan format RGB dengan ukuran 8 bit. Gambar 4.3 adalah *coding* sistem membuka file citra dalam bahasa pemrograman matlab.

```
[namafile,direktori]=uigetfile({'*.bmp';'*.jpg';'*.
gif';'*.tif'; '*.png'}, 'Buka Gambar');
gbrA = imread(namafile);
```

Gambar 4.3 *Listing* program mengambil file citra

Dalam program tersebut dipakai *library* program untuk pengambilan masukan yakni fungsi `uigetfile`. Fungsi tersebut akan memasukkan data yang diambil dari media penyimpanan ke dalam variabel **namafile**. Kemudian dari variabel **namafile** akan dibaca dan diubah dalam bentuk matriks dengan fungsi `imread`. Citra yang telah dirubah dalam matriks dimasukkan dalam variabel **gbrA**.

Dalam menampilkan citra masukan pada *user interface* digunakan listing program pada Gambar 4.4.

```
set(proyek.figsisip,'CurrentAxes',proyek.gbrAaxes);
set(imshow(gbrA));
set(proyek.gbrAaxes,'Userdata',gbrA);
```

Gambar 4.4 *Listing* program menampilkan file citra

Pada baris pertama adalah menentukan bahwa *axes* atau panel tempat citra diletakkan adalah **gbrAaxes** pada *figure* **figsisip**. Baris kedua adalah menentukan variabel yang akan ditampilkan menggunakan fungsi `imshow` yaitu **gbrA**. Baris terakhir adalah menetapkan bahwa variabel **gbrA** yang akan ditampilkan akan diletakkan pada **gbrAaxes**.

Ukuran teks pesan tergantung pada besar ukuran pixel gambar asli. Perhitungan nilai maksimun karakter pesan dibagi 8 karena 1 karakter huruf akan diubah kedalam biner ASCII 8 bit. Frame yang digunakan adalah frame R dan G sehingga besar ruang yang dapat disisipi pesan menjadi 2 kali banyak pixel. Setiap bit pesan akan dimasukkan kedalam setiap 2 byte pixel gambar asli.

Max kemudian dikurangi oleh angka 4 karena pesan sebelum disisipkan akan dimasukkan 4 buah karakter ‘Q’. Empat buah karakter ‘Q’ digunakan untuk mengetahui akhir dari pesan yang disisipkan.

Sehingga perhitungan dalam menentukan panjang teks pesan yang dapat disisipkan adalah sebagai berikut,

$$Max = \frac{jbrs \times jklm \times 2_{frame}}{8_{bit} \times 2_{pix/\ pesan}} - 4_{karakterQ}$$

$$Max = \frac{jbrs \times jklm}{8} - 4$$

Keterangan:

Max	:	Besar maksimum karakter pesan yang dapat disisipkan
Jbrs	:	Banyak baris pixel Gambar Asli
Jklm	:	Banyak kolom pixel Gambar Asli

Panjang maksimun teks pesan ditampilkan dalam sistem. Hal ini berfungsi agar pengguna dapat mengetahui maksimum pesan saat citra masukan dipilih. Implementasi rumus dalam program dapat dilihat pada Gambar 4.5.

```
jbrs = size(gbrA,1);
jklm = size(gbrA,2);
max=((jbrs*jklm)/8)-4;
```

Gambar 4.5 *Listing* program panjang maksimum pesan

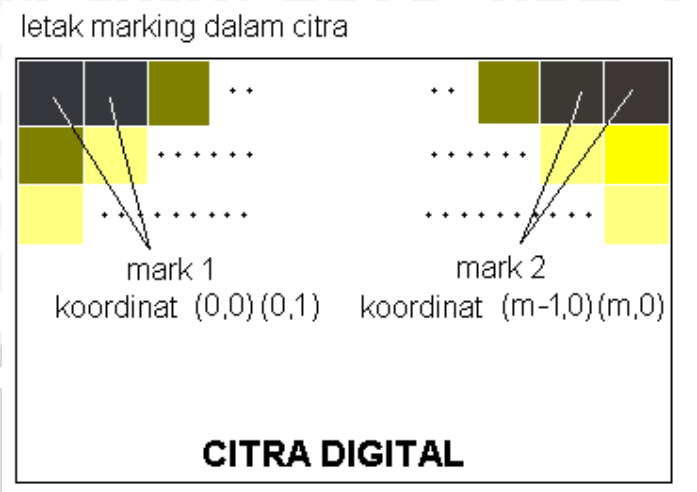
Fungsi `size` adalah fungsi untuk mendapatkan ukuran variabel baik banyak baris maupun kolom. Dalam pengambilan banyak baris ditandai dengan nilai 1 pada variabel kedua dalam fungsi `size`. Dalam pengambilan banyak kolom ditandai dengan nilai 2 pada variabel kedua dalam fungsi `size`. Sehingga baris pertama pada *listing* diatas adalah mengambil banyak baris pada variabel gbrA dimasukkan dalam variabel **jbrs** (jumlah baris) dan baris kedua adalah mengambil banyak kolom ke dalam variabel **jklm** (jumlah kolom). Baris terakhir adalah rumus maksimum karakter.

4.2.2 Penyisipan *Marking*

Penyisipan pixel *marking* digunakan untuk mengetahui besar sudut yang terjadi saat citra mendapatkan rotasi. *Marking* juga digunakan dalam penanganan citra distorsi sehingga citra dapat kembali pada citra kotak (tanpa sudut kemiringan). *Marking* diletakkan pada pojok atas kiri dan kanan dengan rumus marking sebagai berikut,

*Mark*1 :	R = 0011 0100 (52)	*Mark*2 :	R = 0011 1100 (60)
	G = 0011 1000 (56)		G = 0011 1000 (56)
	B = 0011 1100 (60)		B = 0011 0100 (52)

Misal ukuran citra adalah M x N pixel maka *mark*1 berada pada koordinat (0,0) dan (0,1) serta *mark*2 berada pada koordinat (m-1,0) dan (m,0).



Gambar 4.6 Gambar Letak *mark*1 dan *mark*2

Penyisipan dilakukan dengan warna yang jarang digunakan untuk mengurangi kemungkinan terjadinya penyamaan warna di ujung citra. Penyamaan warna dapat mengakibatkan kesalahan pendeteksian pixel marking sehingga gambar tidak dapat kembali ke bentuk kotak.

Warna juga dipilih agak gelap dengan perubahan antar frame yang relatif kecil agar tidak terlalu mencolok saat dilihat. Hal ini dilakukan agar citra yang telah disisipi pesan rahasia tidak terlalu terlihat telah mengalami perubahan yang mencolok pada ujung atas citra tersebut.

Implementasi sistem penyisipan bit making dalam bahasa perogaman dapat dilihat pada Gambar 4.7.

```
jklm=size(gbr,2);
gbr(1,1:2,1)=52;gbr(1,1:2,2)=56;gbr(1,1:2,3)=60;
gbr(1,jklm-1:jklm,1)=60;gbr(1,jklm-1:jklm,2)=56;
```

Gambar 4.7 *Listing* program penyisipan *marking*

Pada baris pertama Gambar 4.10 didapat banyak kolom citra masukan pada variabel **jklm**. Jika dalam suatu variabel citra 2D dituliskan `gbr(a,b,c)`, maka nilai a menyatakan posisi baris, b posisi kolom, dan c adalah posisi frame (1=R, 2=G, 3=B). Sehingga pada baris kedua *listing* diatas, nilai pixel pada baris dan kolom pertama diubah nilainya sesuai nilai *mark* 1. Sedangkan pada baris terakhir merupakan pengubahan nilai pixel sesuai dengan nilai *mark* 2.

4.2.3 Penyisipan Pesan

Pesan yang disisipkan sebelumnya ditambah dengan 4 karakter ‘Q’. Sehingga hasil pesan yang akan disisipkan memiliki tambahan ‘QQQQ’. Penambahan karakter dilakukan agar dapat diketahui akhir dari sebuah pesan sisipan. Setelah diketahui sistem akan berhenti membaca dan ditampilkan karakter sebelum 4 karakter ‘Q’. Hal ini dapat mempercepat kinerja sistem karena sistem tidak perlu membaca keseluruhan pixel jika panjang pesan kurang dari 2 kali total pixel yang ada.Dipilih 4 karakter ‘Q’ berurutan karena sangat jarang terdapat teks dengan 4 karakter ‘Q’ berurutan. Implementasi dalam program tersebut terdapat pada Gambar 4.8.

```
pesan = [pesan 'QQQQ'];
```

Gambar 4.8 *Listing* program penambahan karakter pesan

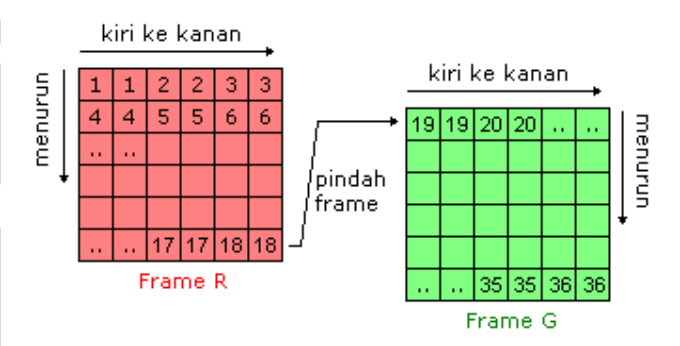
Metode penyisipan pesan yang pertama adalah dengan membuat nol bit ke-7 dan ke-8 frame R dan G dari tiap pixel. Pembuatan nol dilakukan dengan memberikan logika AND pada nilai citra dengan nilai 252 atau dalam biner adalah 11111100. *Listing* programnya dapat dilihat pada Gambar 4.9

```
gbr(i,j,:)=bitand(uint8(gbr(i,j,:)),252);
```

Gambar 4.9 *Listing* program membuat nilai nol bit

Pada Gambar 4.9 nilai variabel **i** dan **j** adalah nilai posisi dan merupakan nilai perulangan saat dilakukan proses pada tiap posisi pixel. Fungsi `bitand` adalah fungsi yang melakukan logika AND pada 2 variabel yang dimasukkan ke dalam fungsi. Misal pada `bitand(2,3)`, maka nilai `2` (biner 10) diberi logika AND dengan nilai `3` (biner 11) sehingga nilai keluaran adalah `2` (biner 10). Karena fungsi `bitand` hanya dapat membandingkan dua variabel dengan ukuran bit yang sama, maka digunakan fungsi `uint8`. Fungsi `uint8` membuat nilai desimal diubah dalam biner 8 bit. Misal `uint8(15)` yang seharusnya binernya 1111 akan ditulis 00001111.

Pesan dimasukkan secara berurutan dari pixel pojok kiri atas (0,0) ke arah kanan pada frame R. Jika penyisipan mencapai kolom maksimal maka penyisipan dilakukan dengan mengubah baris kebawah dan mengulang kolom dari kiri ke kanan. Jika keseluruhan frame R telah tersisipi, maka penyisipan diulang kembali pada frame G. Penyisipan 1 bit pesan akan dimasukkan dalam 2 pixel citra.



Gambar 4.10 Gambar urutan penyisipan

Penyisipan pesan dilakukan dengan membaca urutan nilai biner pesan. Jika nilai pesan sama dengan 1 (satu), maka pixel dari frame R atau G akan ditambah dengan nilai 1(satu), selain itu maka pixel tidak mengalami perubahan. *listing* programnya terdapat pada gambar 4.11.

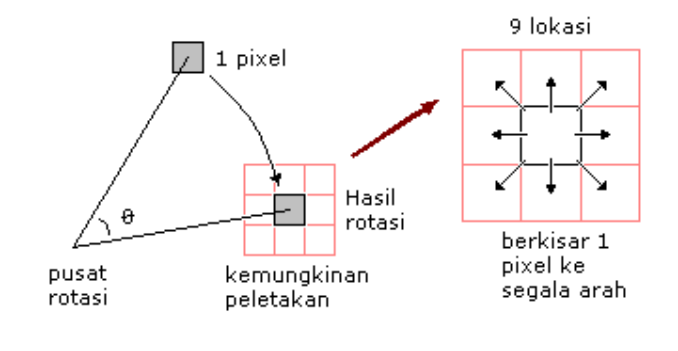
```
if bitget(uint8(gbr(i,j,fr)),1) < str2num(pesan(n))
        gbr(i,j,fr)=gbr(i,j,fr)+1;
```

Gambar 4.11 *Listing* program penyisipan pesan

Fungsi `bitget` adalah fungsi untuk mengambil nilai bit pada sebuah urutan biner. Misal `bitget(a,b)` maka ambil bit ke `b` (perhitungan dimulai dari LSB) dari variabel `a` (variabel `a` diubah dalam biner). Fungsi `str2num` adalah mengubah tipe variabel dari string menjadi number sehingga dapat dibandingkan. Secara keseluruhan *listing* program dapat dibaca sebagai berikut, jika bit LSB citra memiliki nilai lebih kecil dari bit pesan, maka nilai pixel ditambah dengan nilai satu.

4.2.4 Penyisipan Bit Posisi

Penyisipan bit posisi digunakan untuk penanganan citra yang mengalami kesalahan posisi pixel setelah dilakukan rotasi balik. Kesalahan yang mungkin terjadi saat interpolasi terdekat dilakukan adalah 1 pixel ke segala arah, seperti pada Gambar 4.12.



Gambar 4.12 Gambar kemungkinan peletakan pixel

Sehingga besar matriks minimal yang harus dibuat dalam penanganan kesalahan posisi adalah 3 x 3. Tiap matriks memiliki nilai yang berbeda sehingga pixel dapat bergerak 1 pixel kembali ke posisi semula.

Bit posisi disisipkan pada pixel yang telah memiliki pesan tersisip didalamnya ditambah dengan satu baris pixel tanpa pesan tersisip. Penyisipan satu baris setelah pesan selesai disisipkan bertujuan agar pixel tersebut tidak mengganggu pixel yang telah tersisipi pesan saat penanganan posisi pixel. Bit posisi membutuhkan minimal 4 bit dalam satu pixel, karena matriks minimal yang harus dibuat sebesar 3 x 3 atau 9 kemungkinan. Bit yang dipakai dalam penyisipan bit posisi adalah bit ke 7 frame R, G, B dan bit ke 8 frame B, dengan ketentuan,

Pada Baris: Perulangan tiap 3 Baris

Baris 1 : nilai R bit ke-7 = 1

Baris 2 : nilai G bit ke-7 = 1

Baris 3 : nilai B bit ke-7 = 1

Pada Kolom: Perulangan tiap 3 Kolom

Kolom 1 : nilai B bit ke-8 = 1

Kolom 2 : nilai B bit ke-7 = 1

Kolom 3 : nilai G bit ke-7 = 1

Serta terdapat pengecualian pada baris ke-3 dan kolom ke-3

Nilai B bit ke-8 =1

Sehingga nilai bit ke-7 dan ke-8 tiap baris dan kolom akan tampak seperti pada Tabel 4.1.

	Kolom 1				Kolom 2				Kolom 3			
	R7	G7	B7	B8	R7	G7	B7	B8	R7	G7	B7	B8
Baris1	1	0	0	1	1	0	1	0	1	1	0	0
Baris2	0	1	0	1	0	1	1	0	0	1	0	0
Baris3	0	0	1	1	0	0	1	0	0	1	1	1

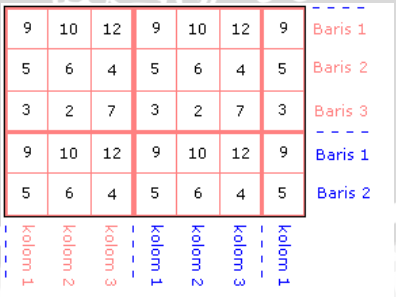
Tabel 4.1 Tabel Bit Posisi

Atau dalam desimal dapat dilihat pada Tabel 4.2.

	Kolom 1	Kolom 2	Kolom 3
Baris1	9	10	12
Baris2	5	6	4
Baris3	3	2	7

Tabel 4.2 Tabel Posisi

Penyisipan dilakukan dengan cara membuat bit yang bersangkutan yakni bit ke 7 frame R, G, B dan bit ke 8 frame B menjadi bernilai nol (nilai telah diubah menjadi nol pada penyisipan bit pesan). Kemudian nilai pixel akan dirubah berdasar ketentuan bit posisi. Setelah kolom atau baris ke-3, maka ketentuan penyisipan bit posisi akan diulang kembali dari kolom atau baris pertama.



Gambar 4.13 Gambar Penyisipan Bit Posisi

Listing program dalam penyisipan bit posisi dapat dilihat pada Gambar 4.14.

```
if mod(j,3)==1  %per 3 kolom
   gbr2(i,j,3)=bitor(uint8(gbr2(i,j,3)),1);
elseif mod(j,3)==2
   gbr2(i,j,3)=bitor(uint8(gbr2(i,j,3)),2);
elseif mod(j,3)==0
   gbr2(i,j,2)=bitor(uint8(gbr2(i,j,2)),2);
end

if mod(i,3)==1  %per 3 baris
   gbr2(i,j,1)=bitor(uint8(gbr2(i,j,1)),2);
elseif mod(i,3)==2
   gbr2(i,j,2)=bitor(uint8(gbr2(i,j,2)),2);
elseif mod(i,3)==0
```

Gambar 4.14 *Listing* program penyisipan bit posisi

Variabel **i** adalah nilai baris, dan **j** adalah nilai kolom. Fungsi `mod` adalah fungsi yang memberikan keluaran berupa sisa hasil bagi dari dua variabel. Misal `mod(10,4)` adalah `10` dibagi `4` memiliki sisa 2 jadi keluarannya adalah `2`, atau `mod (12,4)` adalah `12` dibagi `4` memiliki sisa 0 maka keluarannya `0`. Fungsi `bitor` adalah fungsi yang memberikan logika OR pada kedua masukannya. Dari seleksi kondisi pertama dilakukan pada tiap 3 kolom pada citra dan terus berulang. Hasil dari `mod(j,3)` jika bernilai 1 maka baris pertama, jika 2 maka baris kedua, jika 0 maka baris ketiga. Sama halnya pada seleksi kondisi kedua namun pada seleksi ini dilakukan pada tiap baris. Dipilih fungsi `bitor` karena nilai awal bit ke-7 dan ke-8 citra bernilai nol sehingga lebih mudah diubah jika diberi logika OR. Hasil keluaran disesuaikan dengan nilai pada tabel 4.1.

4.2.5 Keluaran Sistem *Encoding*

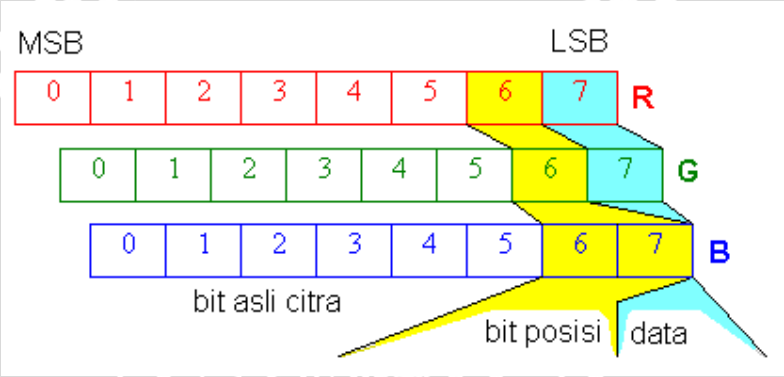
Keluaran sistem berupa citra stego berekstensi file bmp. Hasil dari penyisipan akan disimpan ke dalam media penyimpanan dalam

komputer. Besar ukuran citra stego akan sama persis seperti ukuran citra asli.

Besar perubahan yang terjadi setelah penyisipan pesan tergantung pada besar kecilnya ukuran citra asli dan banyaknya pesan yang disisipkan pada citra asli. Semakin besar ukuran citra, maka semakin kecil persentase perubahan yang terlihat. Semakin banyak pesan yang disisipkan maka semakin besar persentase perubahan yang terlihat.

4.2.6 Diagram Penyisipan Pixel

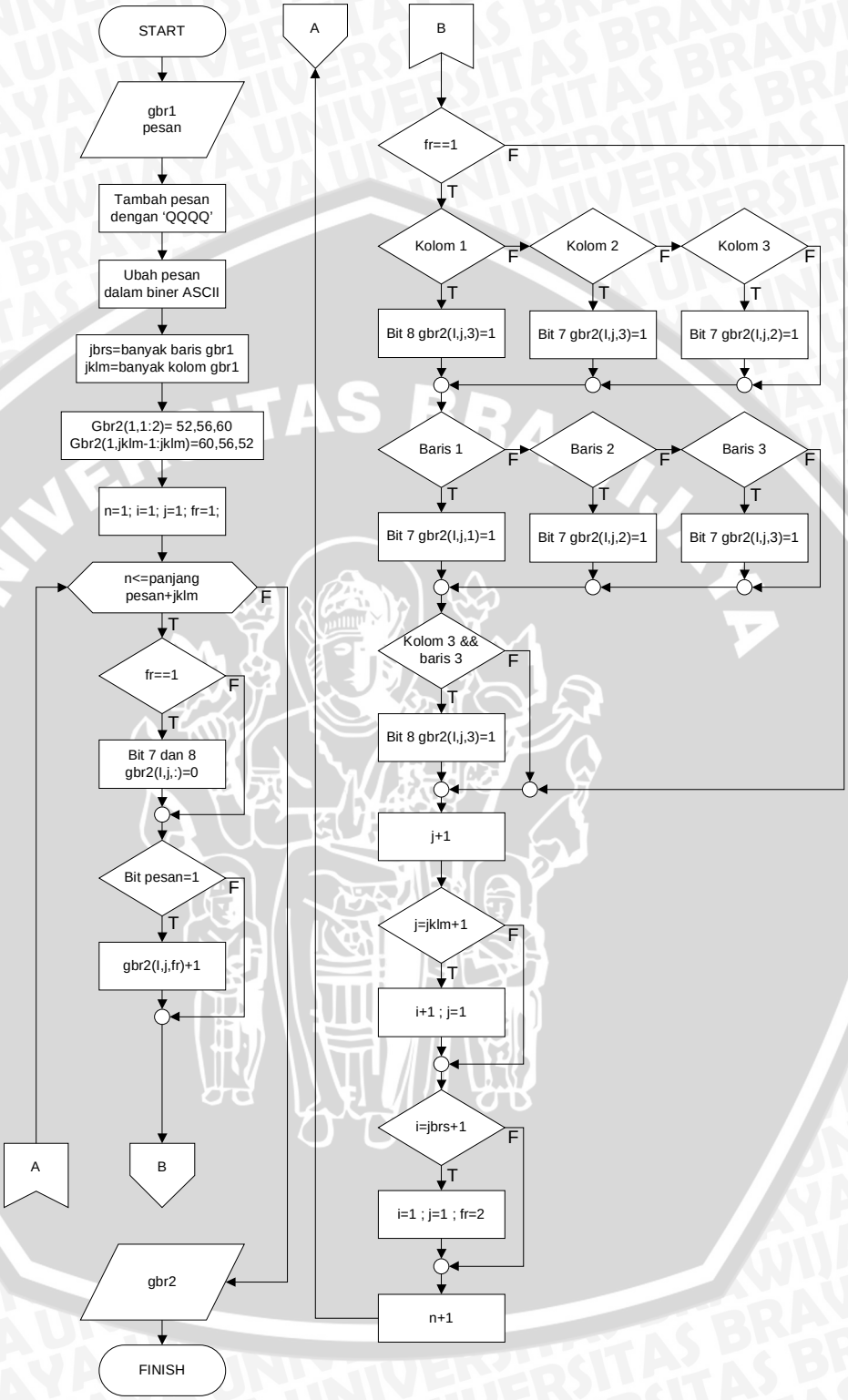
Secara garis besar penyisipan pada tiap pixel dapat dilihat sebagai berikut,



Gambar 4.15 Gambar diagram penyisipan pixel

Bit pertama hingga bit ke-6 merupakan data asli citra. Bit ke-7 frame RGB dan bit ke-8 frame B merupakan letak dimana bit posisi disisipkan. Bit ke 8 frame R dan G merupakan data pesan yang disisipkan. Sedangkan untuk *marking* berada pada 2 pixel citra dan disisipkan pada bit pertama hingga ke-enam.

Diagram alir pada *listing* program sistem penyisipan pesan dapat dilihat pada Ganmbar 4.16. *Listing* program pada penyisipan dimasukkan ke dalam satu fungsi yakni fungsi `stego`.



Gambar 4.16 Gambar Diagram Alir *Listing* Program

4.2.7 *User Interface* Sistem *Encoding*

Fungsi yang terdapat dalam *Interface encoding* meliputi:

1. Judul Program, digunakan agar dapat mengetahui nama program yang berjalan.
2. Panel gambar asli, berfungsi untuk meletakkan gambar asli yang diambil dari komputer.
3. Panel gambar stego, berfungsi untuk meletakkan gambar stego hasil dari penyisipan pesan pada gambar asli
4. Panel teks pesan, berfungsi untuk menampilkan hasil ketikan pesan yang akan disisipkan.
5. Text ukuran max, berfungsi untuk menampilkan maksimum karakter yang dapat disisipkan.
6. Tombol open, berfungsi untuk membuka file citra berekstensi bmp, jpg, gif atau tif dan diletakkan pada panel Gambar Asli.
7. Tombol sisip, berfungsi untuk menyisipkan teks pesan pada gambar asli.
8. Tombol save, berfungsi untuk menyimpan citra stego kedalam komputer dengan ekstensi bmp.
9. Tombol keluar, berfungsi untuk keluar dari program sistem encoding

Gambar 4.17 adalah gambar bentuk rancangan *interface encoding*.

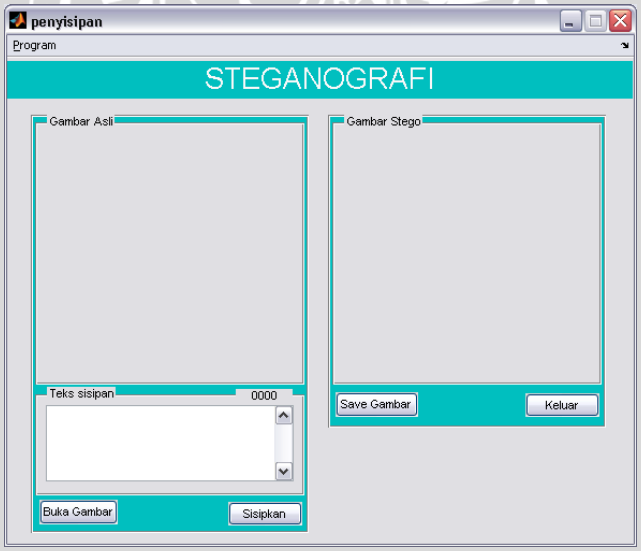
Gambar 4.17 Rancangan *Interface Encoding*

Beberapa hal yang perlu diperhatikan dalam pembuatan *user interface* Sistem *Encoding* menggunakan bahasa pemrograman matlab dapat dilihat dalam tabel 4.3.

no	Style	Name/String	Tag	visible
1	text	STEGANOGRAFI	text1	on
2	axes	-	gbrasliaxes	off
3	axes	-	gbrstegoaxes	off
4	text	0000	teks2	on
5	edit	-	txtsisip	on
6	pushbutton	Save Gambar	btnsavestego	on
7	pushbutton	Keluar	btnoutsisip	on
8	pushbutton	Buka Gambar	btnopenasli	on
9	pushbutton	Sisipkan	btnsisip	on

Tabel 4.3 Tabel Implementasi Sistem *Encoding*

Sehingga dari tabel 4.3 dideroleh hasil implementasi dari perancangan sistem *User Interface Encoding* seperti pada Gambar 4.18



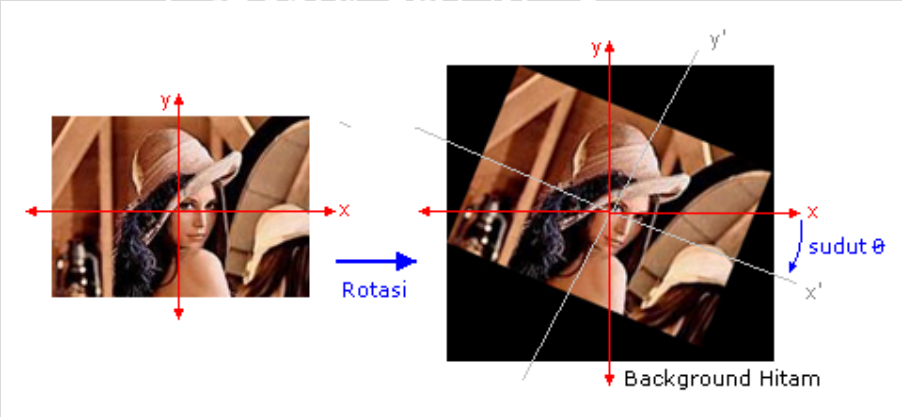
Gambar 4.18 Implementasi *Interface Encoding*

4.3 Perancangan dan Implementasi Sistem Distorsi

Sistem distorsi berfungsi untuk memberikan distorsi pada citra stego. Sistem distorsi dibuat untuk mempermudah pemberian distorsi setelah citra disisipi pesan. Bentuk distorsi yang tersedia adalah rotasi segala sudut, *horizontal flip*, dan *vertical flip*. Sistem ini adalah sistem distorsi internal, karena letak program ini terdapat dalam satu *folder* dan saling terkait dengan dua program lainnya dalam skripsi ini.

4.3.1 Rotasi

Rotasi adalah salah satu jenis distorsi yang megakibatkan posisi sebuah citra berubah dalam jarak dan sudut tertentu dari sebuah titik acuan.



Gambar 4.19 Gambar rotasi secara umum

Dari Gambar 4.19, titik acuan rotasi berada pada pusat sumbu koordinat. Ukuran gambar setelah rotasi mengalami perubahan menjadi lebih besar. Untuk ukuran citra M x N, nilai maksimum perbesaran ukuran citra hasil adalah $\sqrt{M^2 \times N^2}$, Sehingga bentuk citra menjadi persegi.

Karena terjadinya perbesaran, maka terdapat pixel yang kosong. Pixel kosong ini secara otomatis akan diberi nilai 0 tiap frame RGB yakni warna hitam pekat. Warna hitam ini sangat penting dalam

pemrosesan decoding pesan, sehingga tidak diperkenankan diganti dengan warna lain.

Penentuan lokasi pixel atau koordinat baru saat dilakukan rotasi adalah sebagai berikut,

$$x_2 = \cos(\theta) \times (x_1 - x_0) - \sin(\theta) \times (y_1 - y_0) + x_0$$

$$y_2 = \sin(\theta) \times (x_1 - x_0) + \cos(\theta) \times (y_1 - y_0) + y_0$$

Keterangan:

x_2	: Posisi x baru	y_2	: Posisi y baru
x_1	: Posisi x asal	y_1	: Posisi y asal
x_0	: Posisi x acuan	y_0	: Posisi y acuan
θ	: Besar sudut putar		

Rotasi yang dipakai dalam sistem distorsi merupakan rotasi umum dalam pengolahan citra. Rotasi yang digunakan adalah rotasi yang memakai metode interpolasi terdekat (*Nearest Interpolation*) sebagai penempatan pixel, sehingga memiliki kemungkinan error posisi. Rotasi ini juga memiliki error yang membuat beberapa bit pixel menjadi hilang atau bahkan 1 pixel berada pada 2 pixel pada citra hasil, hal ini dikarenakan rotasi merupakan *lossy distortion*. Gambar 4.20 adalah *listing* program fungsi rotasi yang tersedia pada matlab.

```
gbr2=imrotate(gbr1,sudut);
```

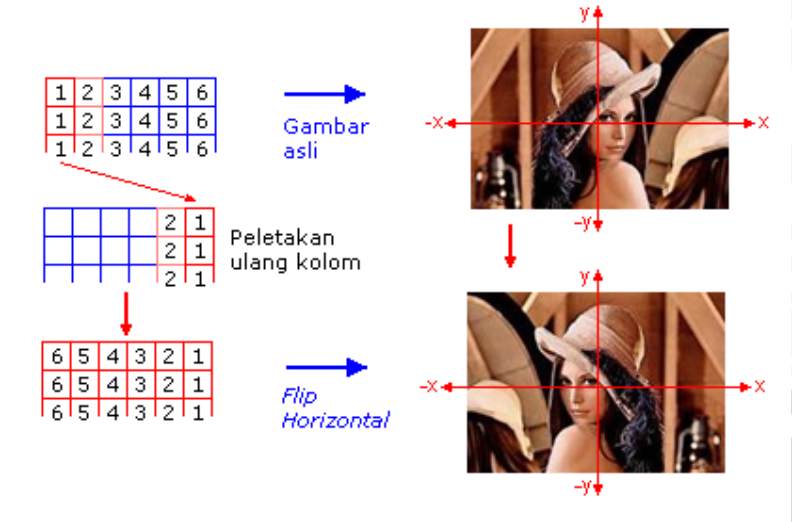
Gambar 4.20 *Listing* program rotasi

Fungsi `imrotate` akan memutar citra `gbr1` sebesar variabel sudut. Hasil keluaran akan diletakkan pada variabel `gbr2`. Nilai variabel sudut dapat berupa angka bulat atau desimal. Namun dalam sistem dibatasi bahwa sudut yang diperbolehkan adalah bilangan bulat. Ukuran antara gambar gbr1 dan gbr2 berbeda dengan adanya rotasi. Besar perubahan ukuran pada citra tergantung pada besar kecilnya nilai sudut yang dimasukkan.

4.3.2 *Horizontal Flip*

Horizontal Flip adalah salah satu bentuk distorsi yang mengubah

posisi tiap kolom dari sebuah citra.



Gambar 4.21 Gambar *horizontal Flip* secara umum

Metode dalam *horizontal flip* adalah dengan menyusun ulang urutan kolom dalam sebuah citra. Kolom pertama diletakkan pada kolom terakhir lalu kolom kedua pada kolom kedua terakhir, seterusnya hingga kolom terakhir pada citra asli menempati kolom pertama pada citra hasil. *Horizontal flip* tidak mengakibatkan perubahan ukuran citra, dan tidak mengakibatkan hilangnya data pixel pada citra asli. Gambar 4.22 adalah *listing* program dari sistem *horizontal flip*.

```
jklm = size(gbr1,2);
j=1;k=jklm;
while j <= jklm
    gbr2(:,j,:)=gbr1(:,k,:);
    j=j+1;k=k-1;
```

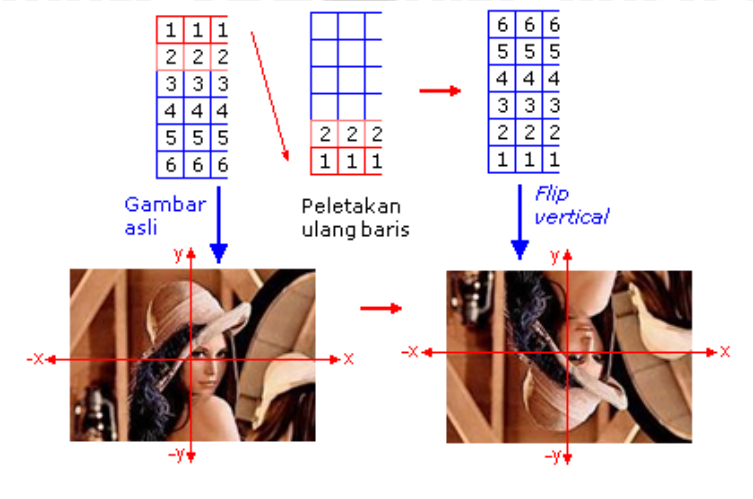
Gambar 4.22 *Listing* program *Horizontal Flip*

Variabel **jklm** berisi jumlah kolom pada **gbr1**. variabel **j** adalah posisi kolom **gbr2** dimulai dari awal kolom (`j=1`). Variabel **k** adalah posisi kolom **gbr1** dimulai dari akhir kolom (`k=jklm`). Perulangan dilakukan pada tiap kolom dari kolom pertama hingga akhir. Pemindahan nilai terjadi pada baris 4, nilai kolom ke-**k** pada **gbr1** dipindah pada kolom ke-**j** pada **gbr2**. kemudian nilai **j** bertambah

diikuti berkurangnya nilai **k**.

4.3.3 *Flip Vertical*

Vertical Flip adalah salah satu bentuk distorsi yang mengubah posisi tiap baris dari sebuah citra.



Gambar 4.23 Gambar *Vertical Flip* secara umum

Metode dalam *vertical flip* adalah dengan menyusun ulang urutan baris dalam sebuah citra. Baris pertama diletakkan pada baris terakhir lalu baris kedua pada baris kedua terakhir, seterusnya hingga baris terakhir pada citra asli menempati baris pertama pada citra hasil. *Vertical flip* tidak mengakibatkan perubahan ukuran citra, dan tidak mengakibatkan hilangnya data pixel pada citra asli. Gambar 4.24 adalah *listing* program dari sistem *vertical flip*.

```
jbrs = size(gbr1,1);
i=1;b=jbrs;
while i <= jbrs
    gbr2(i,:,:)=gbr1(b,:,:);
```

Gambar 4.24 *Listing* program *Vertical Flip*

Variabel **jbrs** berisi jumlah baris pada **gbr1**. variabel **i** adalah posisi baris **gbr2** dimulai dari awal baris (`i=1`). Variabel **b** adalah posisi baris **gbr1** dimulai dari akhir baris (`b=jbrs`). Perulangan dilakukan pada tiap baris. Pemindahan nilai terjadi pada baris ke-4, nilai baris ke-**b** pada **gbr1** dipindah pada baris ke-**i** pada **gbr2**.

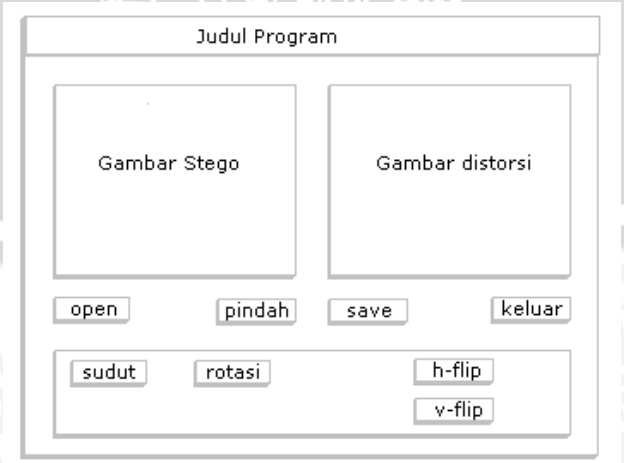
kemudian nilai **i** bertambah diikuti berkurangnya nilai **b**.

4.3.4 *User Interface* Sistem Distorsi

Fungsi yang terdapat dalam *Interface* distorsi meliputi:

1. Judul Program, digunakan agar dapat mengetahui nama program.
2. Panel gambar stego, berfungsi untuk meletakkan gambar stego yang diambil dari komputer.
3. Panel gambar distorsi, berfungsi untuk meletakkan gambar distorsi hasil dari rotasi, *horizontal flip*, atau *vertical flip*.
4. Tombol open, berfungsi untuk membuka file citra stego dan diletakkan pada panel gambar stego.
5. Tombol pindah, berfungsi untuk memindahkan citra pada panel distorsi kedalam panel stego agar dapat dilakukan distorsi kembali.
6. Tombol save, berfungsi untuk menyimpan gambar distorsi dengan ekstensi bmp kedalam komputer.
7. Tombol keluar, berfungsi untuk keluar dari program.
8. Panel sudut, berfungsi untuk mengetikkan besar sudut rotasi.
9. Tombol rotasi, berfungsi untuk melakukan rotasi pada gambar stego sebesar sudut dan diletakkan pada panel gambar distorsi.
10. Tombol h-flip, berfungsi untuk melakukan *horizontal flip* pada gambar stego dan diletakkan pada panel gambar distorsi.
11. Tombol v-flip, berfungsi untuk melakukan *vertical flip* pada gambar stego dan diletakkan pada panel gambar distorsi.

Gambar 4.25 adalah gambar bentuk rancangan *interface* distorsi:



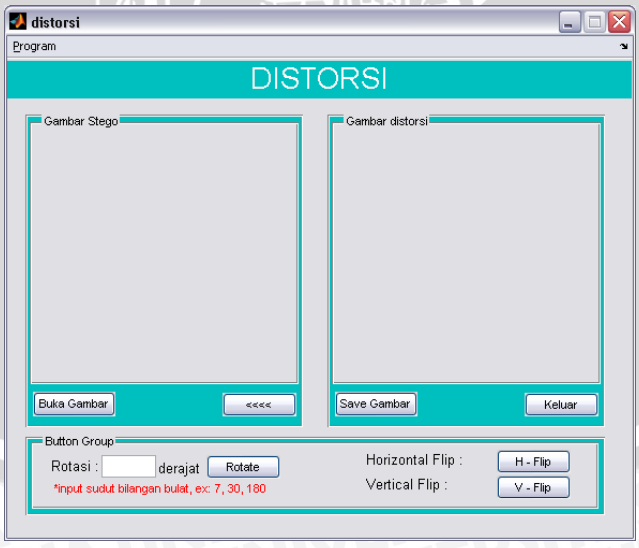
Gambar 4.25 Rancangan *Interface* Distorsi

Beberapa hal yang perlu diperhatikan dalam pembuatan *user interface* Sistem Decoding menggunakan bahasa pemrograman matlab dapat dilihat dalam tabel 4.4.

no	Style	Name/String	Tag	visible
1	text	STEGANOGRAFI	text1	on
2	axes	-	gbrstegoaxes2	off
3	axes	-	gbrdistoaxes	off
4	pushbutton	Buka Gambar	btnopenstego	on
5	pushbutton	<<<<	btnpindah	on
6	pushbutton	Save Gambar	btnsavestego	on
7	pushbutton	Keluar	btnoutdisto	on
8	edit	-	txtsudut	on
9	pushbutton	Rotasi	btnrotasi	on
10	pushbutton	H-Flip	btnhflip	on
11	pushbutton	V-flip	btnflip2	on

Tabel 4.4 Tabel Implementasi Sistem Distorsi

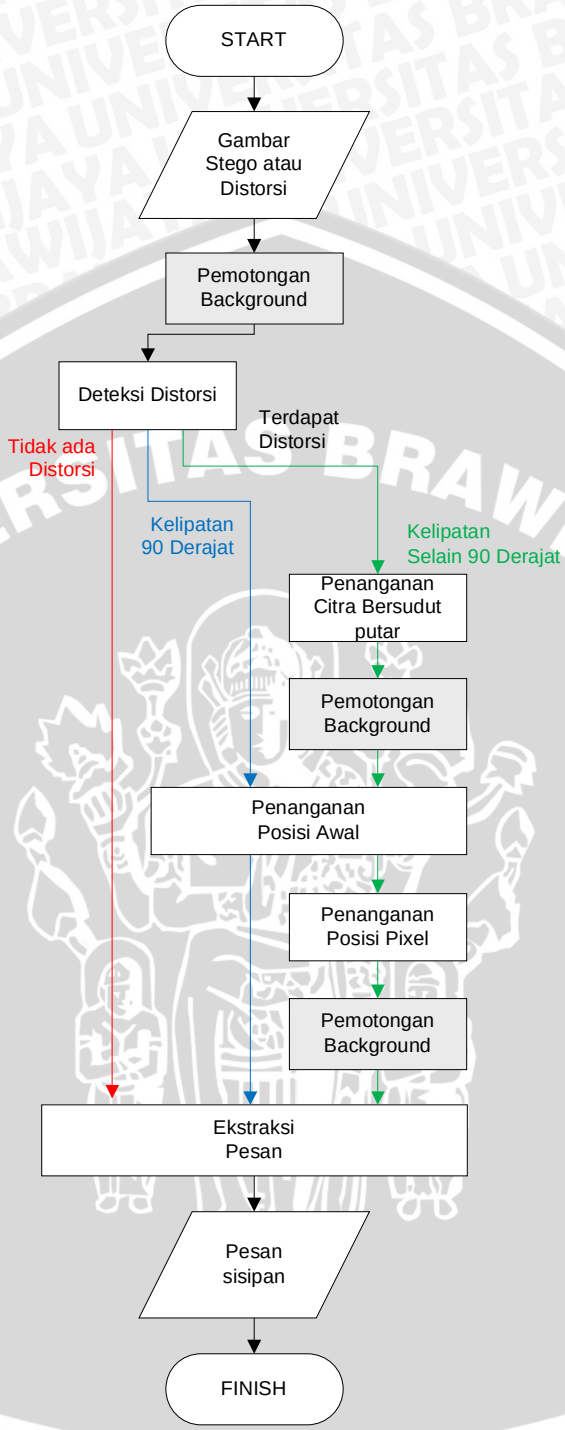
Hasil Implementasi dari perancangan *Interface* Distorsi pada Gambar 4.25 dan Tabel 4.4 terdapat pada Gambar 4.26.



Gambar 4.26 Implementasi *Interface* Decoding

4.4 Perancangan dan Implementasi Sistem *Decoding*

Sistem *decoding* berfungsi untuk mengekstraksi teks pesan dari sebuah citra stego. Gambar 4.27 adalah diagram sistem ekstraksi pesan:



Gambar 4.27 Diagram Sistem Decoding

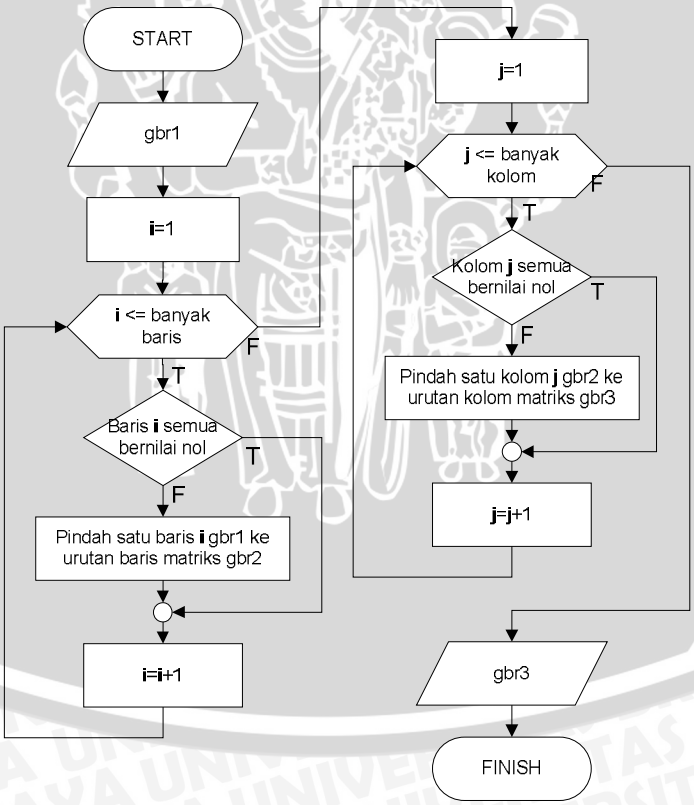
4.4.1 Masukan Sistem *Decoding*

Masukan sistem *decoding* berupa Gambar Distorsi ataupun Gambar Stego dengan ekstensi bmp. Gambar akan diubah dalam bentuk matriks citra dengan format RGB dengan ukuran 8 bit.

Ukuran citra tidak dibatasi, namun dalam pemrosesan, ukuran akan sangat mempengaruhi kecepatan sistem. Semakin besar ukuran citra maka semakin lama proses sistem dalam menampilkan keluaran.

4.4.2 Pemotongan *Background*

Pemotongan (*cropping*) backgrond citra dilakukan untuk membuang baris dan kolom paling luar agar pixel *marking* dapat terletak diposisi baris atau kolom paling luar dari citra distorsi. Pemotongan dilakukan dengan cara membuang satu atau lebih barisan dan/atau kolom yang memiliki nilai pixel 0 (nol) diseluruh anggota sebuah baris dan/atau kolom tersebut. Pada Gambar 4.28 adalah diagram alir dari pemotongan *background*,



Gambar 4.28 Diagram Alir Pemotongan *Background*

Pemotongan menggunakan tiga buah gambar yang akan diproses. Gambar gbr1 adalah masukan sistem dimana dalam gambar tersebut masih memiliki kemungkinan terdapat background berwarna hitam pekat (R=0, G=0, B=0) di keempat sisi citra. Gambar gbr2 adalah hasil pemotongan baris dari gbr1, sehingga bagian atas dan bawah gbr2 sudah tidak terdapat background. Gambar gbr3 adalah hasil dari pemotongan kolom dari gbr2, sehingga dalam gbr3 tidak terdapat background di bagian atas, bawah, kanan, dan kiri.

Gambar 4.29 Gambar Pemotongan *Background* Citra

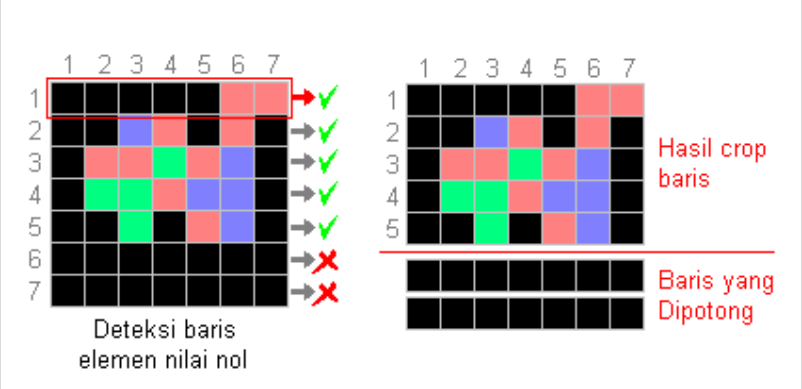
Pemotongan pertama dilakukan dalam baris, hal ini akan menghasilkan citra dengan pemotongan background bagian atas dan bawah. Pemotongan kedua pada kolom, sehingga hasil keluarannya adalah citra dengan pemotongan background bagian samping kanan dan kiri.

Pemotongan baris dan kolom dilakukan dengan cara mendeteksi baris dan/atau kolom yang seluruh elemen penyusunnya bernilai sama, yaitu pixel dengan nilai RGB=0. Jika paling sedikit terdapat satu saja elemen baris dan/atau kolom bernilai tidak nol, maka baris dan/atau kolom tersebut tidak terdeteksi. Gambar 4.30 adalah *listing* program dari pemotongan baris dalam bahasa pemrograman Matlab versi 7.7,

```
i=1;cbrs=1;
while i <= jbrs
    if gbr(i,:,:)==0
        cbrs=cbrs;
    else
        gbrvcrop(cbrs,:,:)=gbr(i,:,:);
        cbrs=cbrs+1;
```

Gambar 4.30 *Listing* program pemotongan *background*

Perulangan dilakukan pada setiap baris mulai dari baris pertama hingga baris paling akhir. Seleksi kondisi menyatakan jika dalam satu baris setiap elemen penyusunnya bernilai 0 (nol) termasuk pada nilai RGB, maka tidak dilakukan apapun, namun jika terdapat satu nilai saja bukan 0 (nol) dalam baris tersebut maka akan dilakukan pemindahan seluruh baris dari variabel **gbr** kedalam variabel **gbrvcrop**. Setelah pemindahan terjadi, maka nilai baris pada **gbrvcrop** akan bertambah.



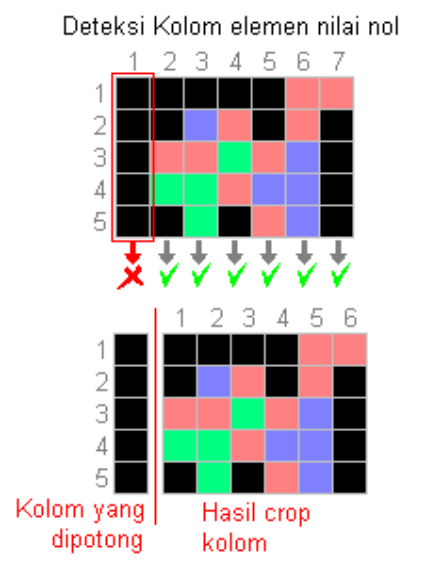
Gambar 4.31 Gambar Pemotongan Baris Matriks citra

Kemudian dilakukan pemindahan baris dan/atau kolom pada matriks gambar awal ke matriks baru. Baris dan/atau kolom yang terdeteksi tidak akan dipindahkan ke matriks baru, sehingga pada matriks baru tidak terdapat baris dan/atau kolom yang memiliki keseluruhan elemen sama bernilai nol. Gambar 4.32 adalah *listing* program dari pemotongan baris dalam bahasa pemrograman Matlab versi 7.7,

```
j=1;cklm=1;
while j <= jklm
    if gbrvcrop(:,j,:)==0
        cklm=cklm;
    else
        gbrcrop(:,cklm,:)=gbrvcrop(:,j,:);
        cklm=cklm+1;
```

Gambar 4.32 *Listing* program penyisipan bit posisi

Perulangan dilakukan pada setiap kolom mulai dari kolom pertama hingga kolom paling akhir. Seleksi kondisi menyatakan jika dalam satu kolom setiap elemen penyusunnya bernilai 0 (nol) termasuk pada nilai RGB, maka tidak dilakukan apapun, namun jika terdapat satu nilai saja bukan 0 (nol) dalam kolom tersebut maka akan dilakukan pemindahan seluruh kolom dari variabel **gbrvcrop** kedalam variabel **gbrcrop**. Setelah pemindahan terjadi, maka nilai kolom pada **gbrcrop** akan bertambah.



Gambar 4.33 Gambar Pemotongan Kolom Matriks citra

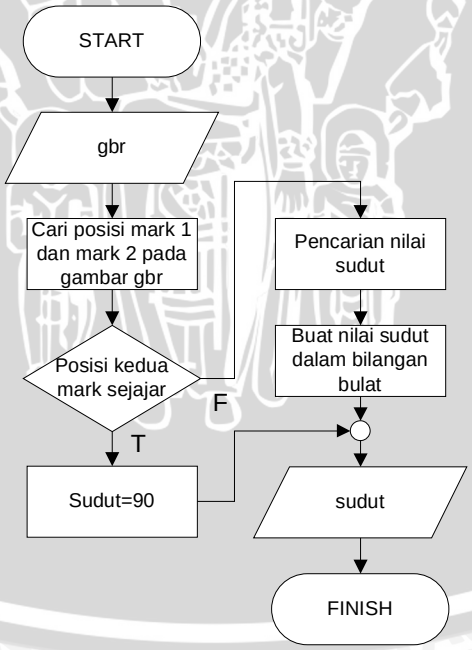
Pemotongan akan mengakibatkan perubahan ukuran citra dan kemungkinan terjadi perubahan susunan posisi dari tiap pixel. Pada gambar 4.33 dapat dilihat jika pemotongan dimulai dari awal

pemindahan pixel, maka kolom kedua pada gambar awal akan berada pada posisi pertama dari gambar hasil.

Pemotongan mungkin terjadi pada bagian tengah citra, namun tidak dapat terjadi pada pixel yang telah tersisipi pesan. Sehingga jika pada citra asli terdapat baris dan/atau kolom yang seluruh elemennya bernilai nol kemudian gambar asli disisipkan oleh pesan, maka hasilnya pixel bernilai nol pasti akan berubah nilai menjadi tidak sama dengan nol. Hal ini dikarenakan adanya penyisipan bit posisi yang selalu menyisipkan sebuah nilai pada citra.

4.4.3 Deteksi Distorsi

Deteksi distorsi digunakan untuk mengetahui apakah citra masukan sistem decoding telah mengalami distorsi atau tidak. Fungsi deteksi distorsi adalah untuk memotong proses yang tidak perlu agar hasil keluaran dapat segera diperoleh. Deteksi distorsi menggunakan pixel marking yang telah disisipkan pada citra masukan. Gambar 4.34 adalah diagram alir deteksi distorsi.



Gambar 4.34 Diagram Alir Deteksi Distorsi

Masukan dari deteksi distorsi adalah sebuah citra yang telah disisipi oleh sebuah pesan. Pada awalnya posisi pixel marking yaitu

*mark*1 dan *mark*2 dicari pada setiap pixel paling luar atau tepi sebuah citra. Kemudian dilihat apakah kedua *mark* memiliki posisi yang sejajar (nilai kolom atau baris sama). Apabila posisi kedua *mark*sejajar maka hasil keluaran akan bernilai 90 derajat. Apabila posisi kedua *mark*tidak sejajar, maka akan dilakukan pencarian nilai sudut.

Pencarian nilai sudut menggunakan rumus jarak x dan y pada sistem koordinat kartesian. Kemudian dilakukan *arcestangen* pada kedua titik. Rumus yang dipakai adalah sebagai berikut,

$$sudut = \arctan(\frac{|x_2 - x_1|}{|y_2 - y_1|})$$

Keterangan:

x_2 : Posisi x *mark* 2 y_2 : Posisi y *mark* 2

x_1 : Posisi x *mark* 1 y_1 : Posisi y *mark* 1

sudut : besar nilai sudut yang terjadi

Perhitungan rumus diatas akan menghasilkan nilai antara 0 dan 90 yang menunjukkan besar kemiringan citra setelah diberi distorsi. Gambar 4.35 adalah *listing* program dari pencarian sudut tersebut.

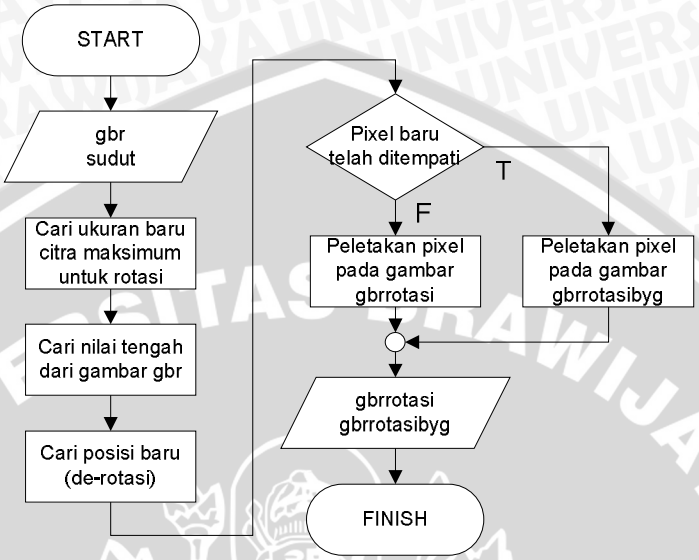
```
if xpos1==xpos2 || ypos1==ypos2
    sudut=90;
else
    arcteta=(xpos1-xpos2)/(ypos1-ypos2);
```

Gambar 4.35 *Listing* program deteksi distorsi

Variabel **xpos1** dan **ypos1** adalah nilai koordinat dari *mark1*. Variabel **xpos2** dan **ypos2** adalah nilai koordinat dari *mark2*. Seleksi kondisi menyatakan jika nilai salah satu koordinat *mark1* dan *mark2* sama atau sejajar, maka nilai sudut sebesar 90 derajat. Jika tidak terdapat nilai sama pada posisi *mark1* dan *mark2*, maka dilakukan perhitungan seperti pada rumus sudut. Fungsi `atand()` adalah melakukan inverse tangen pada masukan. Setelah didapatkan nilai sudut tersebut kemudian dilakukan pembulatan pada nilai sudut.

4.4.4 Penanganan Citra Bersudut Putar

Penangan citra bersudut putar digunakan untuk membuat citra masukan sistem *decoding* menjadi berbentuk kotak atau tidak memiliki sudut kemiringan. Gambar 4.36 adalah diagram alirnya.



Gambar 4.36 Diagram Alir Penanganan Citra Bersudut putar

Penangan citra bersudut putar dapat disebut juga sebagai rotasi balik dari citra yang telah diberi rotasi. Masukan dari sistem ini adalah citra masukan sistem *decoding* dan juga sudut hasil keluaran dari sistem deteksi distorsi. Rotasi balik dimulai dengan pencarian ukuran maksimum citra saat dirotasi. Ukuran baru maksimum dalam rotasi dapat dicari menggunakan rumus berikut,

$$\max PX = \sqrt{pjg^2 \times lbr^2}$$

Keterangan:

maxPX : Nilai ukuran maksimum pixel setelah rotasi

pjg : Nilai ukuran banyak kolom

lbr : Nilai ukuran banyak baris

Panjang maksimum tersebut didapat dari rumus diagonal sisi persegi panjang. Rumus tersebut sangat efektif terhadap citra dengan ukuran yang relatif besar. Sedangkan pada citra dengan ukuran relatif kecil, rumus tersebut akan mengalami kesalahan. Hal ini dikarenakan letak pixel pada citra adalah bilangan bulat. Sehingga hasil perhitungan

akan diarahkan pada bilangan bulat. Pembulatan akan menggeser letak nyata sebuah pixel. Penggeseran pada citra yang relatif besar tidak akan terlalu tampak, namun pada citra dengan ukuran yang relatif kecil, pergeseran 1 pixel akan tampak sangat jelas. Hal ini dalam rotasi dapat mengakibatkan nilai hasil dari rotasi melampaui nilai maksimum pada rumus diatas.

Untuk mengantisipasi masalah tersebut, maka nilai maksimum pixel setelah rotasi menggunakan rumus berikut,

$$\max PX = 2 \times \max PJG$$

Keterangan:

maxPX : Nilai ukuran maksimum pixel setelah rotasi

maxPJG : Nilai ukuran maksimum dari perbandingan antara banyak kolom dan banyak baris

Dengan rumus diatas maka ukuran citra akan bertambah dua kali lipat dan berbentuk persegi. Kelemahan rumus diatas adalah ukuran pixel pada citra yang relatif besar akan menjadi sangat besar. Namun, setelah rotasi selesai citra akan langsung dipotong dengan sistem pemotongan *background*. Sehingga citra dapat kembali kedalam ukuran semula. Gambar 4.37 adalah listing program dalam mencari ukuran baru citra.

```
jbrs = size(gbr,1);
jklm = size(gbr,2);
if jklm>jbrs
    bigsize=jklm;
else
    bigsize=jbrs;
```

Gambar 4.37 *Listing* program mencari ukuran baru

Variabel **jbrs** dan **jklm** adalah jumlah baris dan jumlah kolom pada citra **gbr**. Pada seleksi kondisi menyatakan jika **jklm** lebih besar dari **jbrs** maka nilai terbesar adalah **jklm** jika tidak maka nilai terbesar adalah **jbrs**. Baris terakhir adalah perhitungan kali 2 dari nilai terbesar.

Setelah ukuran citra didapat, maka dilakukan pencarian nilai tengah pada citra masukan. Pencarian nilai tengah dilakukan dengan cara membagi dua banyak kolom dan banyak baris. Kemudian dilakukan pembulatan pada keduanya.

$$x_0 = \text{(banyak kolom)} / 2$$

$$y_0 = \text{(banyak baris)} / 2$$

Keterangan:

x_0 : Nilai tengah kolom pada citra masukan

y_0 : Nilai tengah baris pada citra masukan

Pembulatan dari hasil diatas dilakukan dengan menambahkan nilai desimal hingga kedua nilai bernilai bulat. Pencarian nilai tengah digunakan dalam perhitungan pencarian posisi saat rotasi. Gambar 4.38 adalah listing program dalam mencari nilai tengah citra.

```
m0=(jklm/2)+(1-mod((jklm/2),1));
n0=(jbrs/2)+(1-mod((jbrs/2),1));
```

Gambar 4.38 *Listing* program mencari nilai tengah

Variabel m0 adalah nilai tengah kolom. pada m0 dilakukan pembagian dengan nilai 2 pada variabel jklm (jumlah kolom). Variabel n0 adalah nilai tengah baris. Pada n0 dilakukan pembagian dengan nilai 2 pada variabel jbrs (jumlah baris). Kedua variabel dibulatkan dengan ditambah fungsi mod yaitu hasil sisa pada variabel dengan pembagi nilai 1. Hal ini dilakukan karena nilai posisi pixel selalu bernilai bulat.

Pencarian posisi baru atau rotasi balik menggunakan rumus sebagai berikut,

$$x_2 = \text{hmar} + \cos(\theta) \times (x_1 - x_0) - \sin(\theta) \times (y_1 - y_0) + x_0$$

$$y_2 = \text{vmar} + \sin(\theta) \times (x_1 - x_0) + \cos(\theta) \times (y_1 - y_0) + y_0$$

Keterangan:

x_2	: Posisi x baru	y_2	: Posisi y baru
x_1	: Posisi x asal	y_1	: Posisi y asal
x_0	: Posisi x tengah	y_0	: Posisi y tengah
θ	: Besar sudut putar		

vmar : Setengah dari perbadaan antara nilai ukuran maksimum dan banyak baris citra masukan

hmar : Setengah dari perbadaan antara nilai ukuran maksimum dan banyak kolom citra masukan

Rumus diatas sama dengan rumus rotasi secara umum, namun ditambah dengan variabel **hmar** (*horizontal margin*) dan **vmar** (*vertical margin*). Kedua variabel tambahan berfungsi untuk membuat nilai posisi ***x*** baru dan ***y*** baru bernilai positif. Selain itu variabel tersebut juga berfungsi untuk meletakkan citra masukan hasil rotasi berada pada bagian tengah dari citra hasil. Gambar 4.39 adalah *listing* program dalam mencari posisi baru pixel.

```
vmargin=((newsize-jbrs)/2)+(1-mod((newsize-jbrs)/2,1));
hmargin=((newsize-jklm)/2)+(1-mod((newsize-jklm)/2,1));

m2=vmargin+cosd(sudut)*(m-m0)-sind(sudut)*(n-n0)+m0;
```

Gambar 4.39 *Listing* program mencari posisi baru pixel

Nilai **x** dan **y** pada rumus berturut-turut digantikan oleh variabel **m** dan **n**. Pada baris ke-3 dan ke-4 pada Gambar 4.39 adalah rumus rotasi dalam mencari posisi baru. Variabel **vmargin** (*vertical margin*) adalah batas atas dan bawah dari citra awal dan citra setelah didapat nilai baru. Variabel **hmargin** (*horizontal margin*) adalah batas kiri dan kanan dari citra awal dan citra setelah didapat nilai baru.

Peletakan pixel dilakukan dengan interpolasi terdekat (*nearest interpolation*). Sehingga peletakan memiliki 4 kemungkinan nilai setelah didapat nilai posisi baru. Dari keempat kemungkinan akan diambil kemungkinan terbesar (posisi terdekat). Jika pada kemungkinan terbesar ternyata telah ditempati oleh pixel sebelumnya, maka peletakan akan dilakukan pada posisi dengan kemungkinan terbesar kedua. Jika ternyata posisi tersebut telah terisi maka peletakan dilakukan pada kemungkinan ke-tiga dan seterusnya hingga kemungkinan ke-empat.

Apabila keempat kemungkinan peletakan ternyata telah terisi oleh pixel sebelumnya, maka peletakan pixel akan dilakukan pada citra hasil kedua dan dengan posisi kemungkinan terbesar. Citra kedua dapat disebut sebagai citra bayangan. Oleh karena itu pada sistem penangan citra bersudut putar memiliki 2 buah keluaran citra. Walaupun sudut kemiringan telah hilang, namun citra masih belum berada pada posisi asli. Terutama apabila citra asli diberi rotasi lebih dari 90 derajat atau diberi distorsi berupa *horizontal flip*, atau *vertical flip*. Gambar 4.40 adalah *listing* program dari interpolasi terdekat.

```
if gbrrotasi(n3,m3,:)==0
   gbrrotasi(n3,m3,:)=gbr(n,m,:);
   count=5;
else
   if count==1
      if (m3i+n3i==2 && m2i<n2i)||(m3i+n3i==3 &&
                                        (1-m2i)<n2i)
         n3=n3+1;
      elseif (m3i+n3i==2 && m2i>=n2i)||(m3i+n3i==4
                                    && m2i>=(1-n2i))
         m3=m3+1;
      elseif (m3i+n3i==3 && (1-m2i)>=n2i)||
                             (m3i+n3i==5 && m2i<n2i)
         m3=m3-1;
      elseif (m3i+n3i==4 && m2i<(1-n2i))||
                            (m3i+n3i==5 && m2i>=n2i)
         n3=n3-1;
      end
   elseif count==2
      if m3i+n3i==2
         if m2i<n2i
            m3=m3+1; n3=n3-1;
         else
            m3=m3-1; n3=n3+1;
         end
      elseif {…}
      end
```

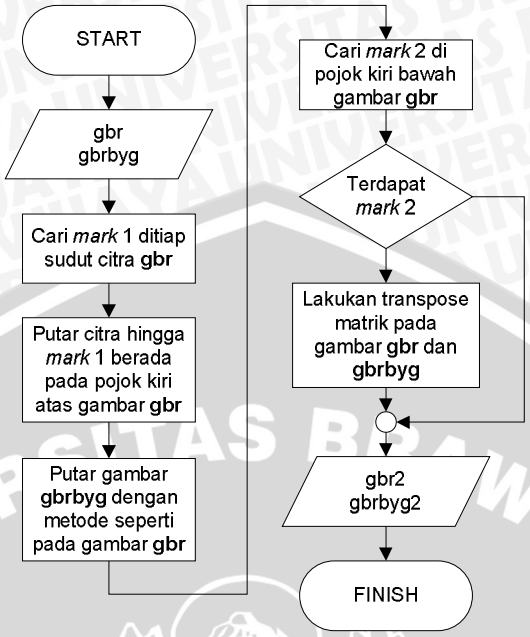
Gambar 4.40 *Listing* program interpolasi terdekat

Variabel citra yang dipakai pada listing program interpolasi terdekat adalah 3 buah citra yaitu **gbr**, **gbrrotasi**, dan **gbrrotasibyg**. Citra **gbr** adalah citra masukan sistem penangan posisi miring. Citra **gbrrotasi** adalah citra utama hasil rotasi balik. Citra **gbrrotasibyg** adalah citra bayangan dari hasil rotasi balik. Dari seleksi kondisi pertama menyatakan bahwa jika nilai posisi baru pada citra gbrrotasi bernilai nol, maka nilai pada gbr akan dipindah pada posisi baru tersebut. Jika tidak bernilai nol maka dilakukan proses lanjut. Proses lanjut tersebut adalah seleksi kondisi yang mencari nilai terdekat dengan menggunakan variabel **count** (penghitung). Namun jika nilai terdekat tersebut ternyta juga telah berisi nilai, maka variabel **count** akan bertambah. Proses berulang hingga tiga kali perulangan. Jika hingga hitungan ke tiga ternyata tidak didapat posisi kosong, maka dilakukan pemindahan nilai pada citra **gbrrotasibyg** (baris 37). *Listing* program yang berupa `{...}` adalah penyingkatan program karena merupakan perulangan kondisi.

Fungsi utama citra bayangan adalah untuk menangani hilangnya pixel citra saat terjadi rotasi. Rotasi sendiri pada umumnya adalah jenis lossy distorsi dimana saat proses dilakukan, pixel yang tidak mendapatkan posisi akan hilang. Sedangkan pada steganografi, setiap pixel citra sangatlah penting karena memiliki pesan didalamnya. Pesan yang tersisipi memiliki sifat continuitas, sehingga apabila terpotong atau hilang maka pesan tidak akan terbaca.

4.4.5 Penanganan Posisi Awal

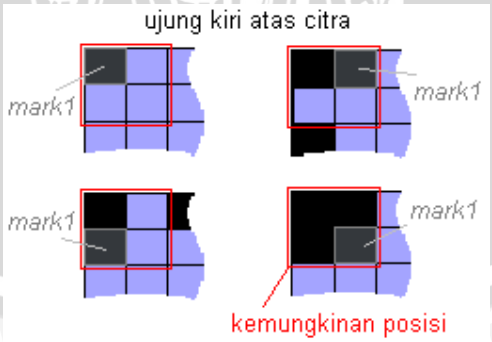
Penanganan posisi awal digunakan untuk mengembalikan posisi citra pada posisi asli. Masukan dari sistem ini adalah citra yang sudah tidak memiliki sudut kemiringan. Diagram alir dari penanganan posisi awal dapat dilihat pada Gambar 4.41.



Gambar 4.41 Diagram Alir Penanganan Posisi Awal

Masukan dalam sistem penanganan posisi awal adalah dua citra hasil dari penanganan citra miring dan telah melalui proses pemotongan *backgound*. Dalam sistem penanganan posisi awal, hanya dilakukan rotasi kelipatan 90 derajat dan juga transpose matriks.

Langkah pertama yang dilakukan adalah mencari letak *mark* 1 pada citra masukan. Kemungkinan keberadaan *mark* 1 adalah pada 4 posisi pixel di setiap pojok citra. Keadaan serupa pada gambar 4.42 terjadi pada ketiga sudut lainnya dari citra masukan.



Gambar 4.42 Kemungkinan Posisi *mark*1

Pada Gambar 4.43 adalah *listing* program dari pencarian posisi mark pada ujung citra.

```
find=0;
ay=1;
while ay<=2 && find==0
    ax=1;
    while ax<=2 && find==0
        gbrcrop2(ay,ax,:)=bitand(uint8(gbrcrop2
                                        (ay,ax,:)),252);
        if gbrcrop2(ay,ax,1)==52 && gbrcrop2(ay,ax,2)==
                              56 && gbrcrop2(ay,ax,3)==60
            pos1=1;find=1;
```

Gambar 4.43 *Listing* program mencari posisi mark1

Listing pada Gambar 4.43 adalah *listing* dalam pencarian nilai *mark1* hanya pada ujung kiri atas. Variabel **ay** adalah nilai baris dan **ax** adalah nilai kolom. Perulangan dilakukan pada saat nilai **ax** dan **ay** dari nilai satu hingga dua. Pixel yang bersangkutan terlebih dahulu dibuat memiliki nilai nol pada bit ke-7 dan ke-8 menggunakan fungsi `bitand()`. Kemudian nilai pixel disamakan dengan nilai *mark1*. jika ditemukan maka nilai variabel **pos1** (posisi mark1) bernilai satu.

Karena pengembalian posisi citra berdasarkan sudut putar, maka digunakan logika sederhana dalam sistem yaitu,

1. Jika posisi *mark* 1 berada pada pojok kiri atas citra, maka sudut putar bernilai nol atau posisi telah sesuai (pos1=1).
2. Jika posisi *mark* 1 berada pada pojok kanan atas citra, maka sudut putar bernilai 90 derajat (pos1=2).
3. Jika posisi *mark* 1 berada pada pojok kanan bawah citra, maka sudut putar bernilai 180 derajat (pos1=3).
4. Jika posisi *mark* 1 berada pada pojok kiri bawah citra, maka sudut putar bernilai 270 derajat (pos1=4).

Setelah diketahui posisi *mark* 1, maka citra akan diputar sesuai sudut putar sehingga didapat posisi *mark* 1 berada pada pojok kiri atas pada citra hasil.

Rotasi yang pada sudut kelipatan 90 derajat berbeda dengan rotasi dengan sudut bebas selain kelipatan 90 derajat. Hal ini dikarenakan,

terdapat metode cepat dalam penggunaannya agar rotasi dapat berlangsung dengan cepat. Pada rotasi balik sebelumnya nilai berkisar antara 0 sampai 90, sehingga tidak terdapat kelipatan 90 didalamnya.

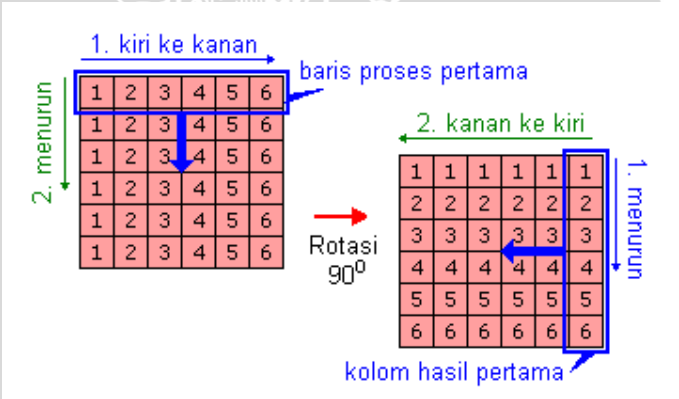
Pada rotasi kelipatan 90 derajat, terdapat empat kemungkinan dan metode tersendiri. Berikut adalah metode dalam penanganannya,

1. Sudut 0^o, Pada sudut ini citra yang dihasilkan akan tetap sesuai dengan citra masukan karena tidak ada perbedaan sudut.

```
gbrrotasi=gbr;
```

Gambar 4.44 *Listing* program rotasi nol derajat

2. Sudut 90^o, Pada sudut ini nilai baris pertama citra masukan dari awal elemen menuju akhir elemen akan dijadikan nilai kolom terakhir citra keluaran dari awal elemen menuju akhir elemen.



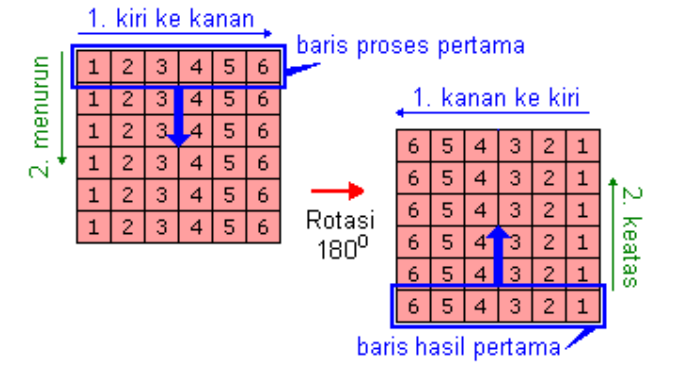
Gambar 4.45 Gambar Metode Rotasi Sudut 90^o

```
i=1;j2=1;
while i<=jbrs
    j=1;i2=jklm;
    while j<=jklm
        gbrrotasi(i2,j2,:)=gbr(i,j,:);
        j=j+1;i2=i2-1;
```

Gambar 4.46 *Listing* program rotasi 90 derajat

3. Sudut 180^o, Pada sudut ini nilai baris pertama citra masukan dari awal elemen menuju akhir elemen akan dijadikan nilai

baris terakhir citra keluaran dari akhir elemen menuju awal elemen.

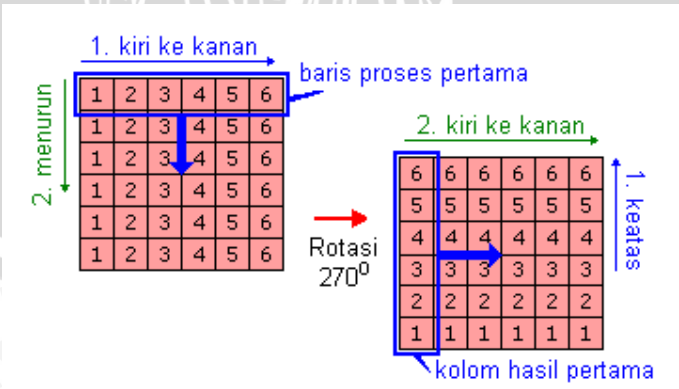


Gambar 4.47 Gambar Metode Rotasi Sudut 180^o

```
i=1;i2=jbrs;
while i<=jbrs
    j=1;j2=jklm;
    while j<=jklm
        gbrrotasi(i2,j2,:)=gbr(i,j,:);
        j=j+1;j2=j2-1;
    end
```

Gambar 4.48 *Listing* program rotasi 180 derajat

4. Sudut 270^o, Pada sudut ini nilai baris pertama citra masukan dari awal elemen menuju akhir elemen akan dijadikan nilai kolom terakhir citra keluaran dari akhir elemen menuju awal elemen.



Gambar 4.49 Gambar Metode Rotasi Sudut 270^o

```
i=1;j2=jbrs;
while i<=jbrs
    j=1;i2=1;
    while j<=jklm
        gbrrotasi(i2,j2,:)=gbr(i,j,:);
        j=j+1;i2=i2+1;
    end
```

Gambar 4.50 *Listing* program rotasi 270 derajat

Digunakan metode kelipatan 90 karena proses dapat dilakukan dengan cepat. Perbedaan kecepatan terjadi karena pada metode tersebut melakukan pemindahan tiap kolom atau baris, sedangkan pada metode umum dilakukan setiap pixel dan terdapat proses interpolasi terdekat. Pada metode ini juga tidak terjadi hilangnya pixel sedangkan pada metode umum terdapat pixel yang hilang.

Setelah *mark*1 berada pada posisi yang diinginkan yakni pojok kiri atas, kemudian dilakukan pencarian nilai *mark* 2. Tujuan dicari nilai *mark* 2 adalah untuk mengetahui apakah citra masukan telah mengalami distorsi *horizontal flip*, atau *vertical flip*.

Apabila citra telah mengalami distorsi *horizontal flip*, atau *vertical flip*, maka posisi *mark* 2 akan berada pada posisi pojok kiri bawah pada citra hasil rotasi kelipatan 90^o.

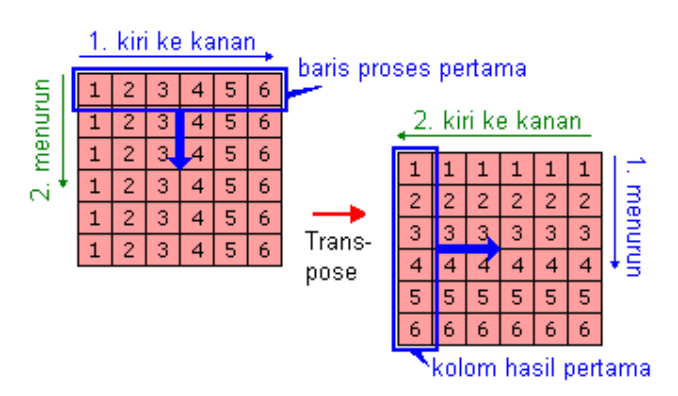
```
ey=jbrs;
while ey>=jbrs-1 && find==1
   ex=1;
   while ex<=2 && find==1
      gbra(ey,ex,:)=bitand(uint8(gbraw(ey,ex,:)),252);
      if gbra(ey,ex,1)==60 && gbra(ey,ex,2)==56 &&
                                    gbra(ey,ex,3)==52
         gbrawal=tranpos(gbraw);
         gbrawalbyg=tranpos(gbrawbyg);
         find=2;
      else
         gbrawal=gbraw;
         gbrawalbyg=gbrawbyg;
```

Gambar 4.51 *Listing* program pencarian nilai mark2

Listing pada Gambar 4.51 adalah *listing* dalam pencarian nilai *mark2* hanya pada ujung kiri bawah. Variabel **ey** adalah nilai baris dan **ex** adalah nilai kolom. Perulangan dilakukan pada saat nilai **ex** dan **ey** dari nilai satu hingga dua dan dan pada **ey** dari nilai maksimum (`jbrs`) hingga 1 pixel sebelum maksimum (`jbrs-1`). Pixel yang bersangkutan terlebih dahulu dibuat memiliki nilai nol pada bit ke-7 dan ke-8 menggunakan fungsi `bitand()`. Kemudian nilai pixel disamakan dengan nilai *mark2*. jika ditemukan maka pada gambar masukan atau variabel **gbrawal** dan **gbrawalbyg** akan dilakukan transpose. Jika tidak terdapat nilai mark2 maka gambar akan tetap seperti semula.

Jika citra tidak mengalami distorsi tersebut maka citra keluaran sistem penanganan posisi awal adalah citra hasil keluaran rotasi kelipatan 90^o. Jika citra mengalami distorsi tersebut maka citra masukan akan dilakukan transpose matrik terlebih dahulu, kemudian menjadi keluaran sistem penanganan posisi awal.

Transpose matriks adalah metode dimana urutan nilai pada awal baris citra akan menjadi urutan nilai pada awal kolom citra hasil. Seperti pada gambar 4.52 berikut,



Gambar 4.52 Metode Transpose matriks citra

```
i=1;j2=jbrs;
while i<=jbrs
    j=1;i2=1;
    while j<=jklm
        gbrrotasi(i2,j2,:)=gbr(i,j,:);
        j=j+1;i2=i2+1;
    end
```

Gambar 4.53 *Listing* program transpose matriks

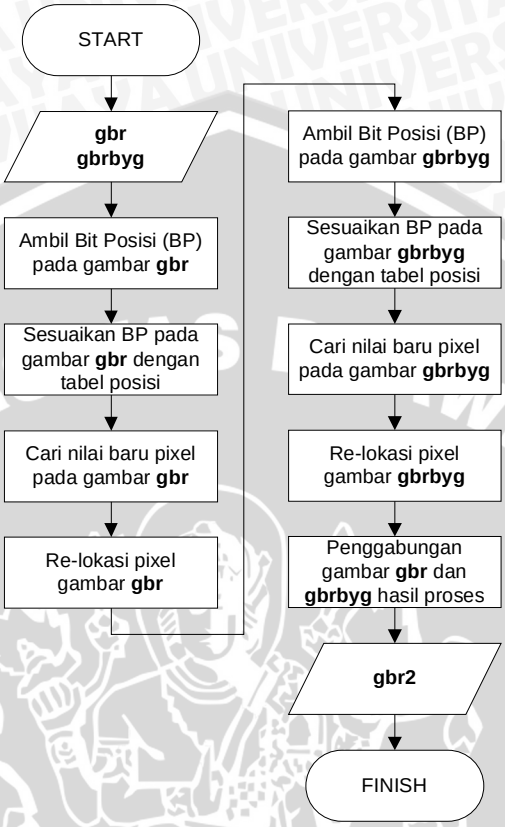
Secara garis besar pemindahan menggunakan metode transpose memiliki konsep yang sama seperti rotasi pada kelipatan 90^o. Perbedaan terdapat dalam urutan pixel yang diletakkan. Pada Gambar 4.53 variabel **i** dan **j** adalah posisi pixel pada citra awal dimulai dari angka satu. Variabel **i2** dan **j2** adalah posisi pada citra hasil dimulai dari akhir kolom dan baris. Setelah pemindahan terjadi (baris 5), maka nilai **j** bertambah diikuti berkurangnya variabel **j2**. hal yang sama terjadi pada nilai variabel **i**.

Saat dilakukan proses rotasi kelipatan 90^o dan transpose matrik, pada gambar bayang juga dilakukan hal yang serupa seperti pada citra masukan. Gambar bayang akan mengikuti gambar masukan saat gambar masukan mengalami perubahan posisi. Sehingga hasil keluaran sistem penanganan posisi awal adalah dua buah citra, yakni gambar keluaran dan gambar bayang. Keduanya secara kasat mata (terutama pada citra dengan ukuran relatif besar) kembali pada posisi sebenarnya dan memiliki nilai posisi pixel yang sangat mendekati dengan posisi semula saat penyisipan pesan dilakukan.

4.4.6 Penanganan Kesalahan Posisi Pixel

Penanganan kesalahan posisi pixel berfungsi untuk mengembalikan citra setelah mengalami rotasi balik. Karena pada rotasi balik terjadi interpolasi terdekat sehingga terdapat kemungkinan-kemungkinan peletakan pixel. Dalam sistem penanganan kesalahan posisi, kemungkinan tersebut akan dipastikan nilainya sehingga posisi pixel dapat kembali pada posisi yang benar-benar tepat seperti saat

penyisipan pesan dilakukan. Gambar 4.54 adalah diagram alir dari penanganan posisi pixel.



Gambar 4.54 Diagram Alir Penanganan Kesalahan Posisi Pixel

Masukan pada sistem Penanganan kesalahan posisi pixel adalah citra keluaran dari sistem penanganan posisi awal. Posisi setiap pixel telah mendekati posisi sebenarnya. Karena masukan sistem adalah keluaran sistem penanganan posisi awal maka masukan sistem adalah dua buah citra, yakni gambar utama dan gambar bayang. Keduanya memiliki ukuran yang relatif sama.

Untuk mengetahui posisi pixel sebenarnya, dilakukan pengambilan nilai bit posisi yang telah disisipkan bertepatan dengan penyisispan pesan. Jumlah bit posisi yang telah disisipkan berjumlah empat buah dengan letak pada bit ke-7 frame R, bit ke-7 frame G, bit ke-7 dan ke-8 frame B.

Keempat bit posisi menyusun sebuah nilai yang berbeda tiap tiga baris dan tiga kolom. Urutan tersebut adalah bit ke-7 frame R sebagai

MSB, kemudian diikuti bit ke-7 frame G, bit ke-7 frame B dan bit ke-8 frame B sebagai LSB. Perhitungan untuk mendapatkan satu nilai seperti pada tabel posisi adalah,

$$\mathbf{BP = (R_7 * 8) + (G_7 * 4) + (B_7 * 2) + (B_8 * 1)}$$

Keterangan:

BP : Bit posisi pada tabel posisi

R_7 : Nilai bit ke-7 pada frame R

G_7 : Nilai bit ke-7 pada frame G

B_7 : Nilai bit ke-7 pada frame B

B_8 : Nilai bit ke-8 pada frame B

Dengan demikian nilai BP akan bernilai desimal dengan nilai antara 0 sampai 15. Gambar 4.55 adalah *listing* program dari implementasi rumus perhitungan nilai BP.

```
bpa=bitget(uint8(gbrawal(i4,j4,1)),2);
bpb=bitget(uint8(gbrawal(i4,j4,2)),2);
bpc=bitget(uint8(gbrawal(i4,j4,3)),2);
bpd=bitget(uint8(gbrawal(i4,j4,3)),1);
```

Gambar 4.55 *Listing* program mencari nilai BP

Fungsi `uint8` membuat nilai desimal diubah dalam biner 8 bit. Fungsi `bitget` adalah fungsi untuk mengambil nilai bit pada sebuah urutan biner. Variabel **bpa** berisi nilai bit pada bit ke-7 frame R. Variabel **bpb** berisi nilai bit pada bit ke-7 frame G. Variabel **bpc** berisi nilai bit pada bit ke-7 frame B. Variabel **bpd** berisi nilai bit pada bit ke-8 frame B. Variabel **bptot** merupakan implementasi dari rumus perhitungan nilai BP.

Nilai tersebut akan dilakukan penyamaan terhadap tabel posisi. Sehingga dapat diketahui apakah posisi tersebut sesuai dengan posisi yang seharusnya. Jika tidak sesuai akan dicari posisi disekitar pixel dengan jarak satu pixel ke segala arah hingga didapat nilai yang sesuai.

Nilai pada tabel 4.2 juga dapat ditulis sebagai,

1. Nilai 9, maka pixel harus berada pada posisi kolom dengan rumus 3n-2 dan baris dengan rumus 3n-2

2. Nilai 10, maka pixel harus berada pada posisi kolom dengan rumus 3n-2 dan baris dengan rumus 3n-1
3. Nilai 12, maka pixel harus berada pada posisi kolom dengan rumus 3n-2 dan baris dengan rumus 3n
4. Nilai 5, maka pixel harus berada pada posisi kolom dengan rumus 3n-1 dan baris dengan rumus 3n-2
5. Nilai 6, maka pixel harus berada pada posisi kolom dengan rumus 3n-1 dan baris dengan rumus 3n-1
6. Nilai 4, maka pixel harus berada pada posisi kolom dengan rumus 3n-1 dan baris dengan rumus 3n
7. Nilai 3, maka pixel harus berada pada posisi kolom dengan rumus 3ndan baris dengan rumus 3n-2
8. Nilai 2, maka pixel harus berada pada posisi kolom dengan rumus 3n dan baris dengan rumus 3n-2
9. Nilai 7, maka pixel harus berada pada posisi kolom dengan rumus 3n dan baris dengan rumus 3n

Gambar 4.56 adalah *listing* program penyesuaian nilai BP dengan nilai pada tabel posisi.

```
if bptot==9 || bptot==5 || bptot==3
    bpklm=1;
elseif bptot==10 || bptot==6 || bptot==2
    bpklm=2;
elseif bptot==12 || bptot==4 || bptot==7
    bpklm=3;
else
    bpklm=0;
end

if bptot==9 || bptot==10 || bptot==12
    bpbrs=1;
elseif bptot==5 || bptot==6 || bptot==4
```

Gambar 4.56 *Listing* program mencari nilai BP

Variabel **bpklm** (bit posisi kolom) adalah letak posisi pixel pada kolom yang seharusnya. Variabel **bpbrs** (bit posisi baris) adalah letak

posisi pixel pada baris yang seharusnya. Kedua nilai variabel disesuaikan pada tabel posisi.

Setelah diketahui posisi asli pixel maka dilakukan perbandingan dengan nilai posisi pixel saat itu. Nilai perbandingan kolom dapat dilihat pada tabel 4.5.

A	B	C	D	E	F
no	countklm	bpklm	A-B	Mod(D,3)	Perpindahan Pixel
1	3 (1)	1	2	2	Kolom tetap (tidak bergeser)
2	3 (1)	2	1	1	Kolom -1 (Geser kekiri)
3	3 (1)	3	0	0	Kolom +1 (Geser kekanan)
4	4 (2)	1	3	0	Kolom +1 (Geser kekanan)
5	4 (2)	2	2	2	Kolom tetap (tidak bergeser)
6	4 (2)	3	1	1	Kolom -1 (Geser kekiri)
7	5 (3)	1	4	1	Kolom -1 (Geser kekiri)
8	5 (3)	2	3	0	Kolom +1 (Geser kekanan)
9	5 (3)	3	2	2	Kolom tetap (tidak bergeser)

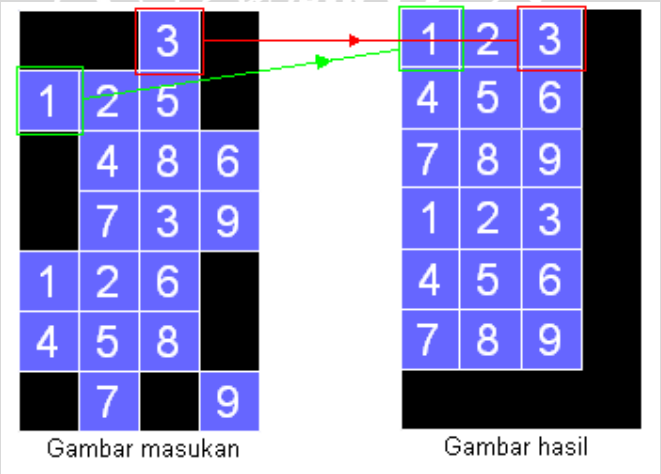
Tabel 4.5 Tabel perbandingan nilai kolom

Pada Tabel 4.5 nilai dalam tanda kurung pada kolom **countklm** adalah nilai **countklm** dikurangi 2. dari tabel 4.5 didapat jika nilai **countklm-2** sama dengan nilai **bpklm**, maka nilai kolom akan tetap. Namun jika tidak sama, dilakukan pengurangan **countklm** dan **bpklm**. Nilai **countklm** ditambah dengan nilai 2 agar saat dilakukan pengurangan tidak menghasilkan nilai negatif. Hasil pengurangan dibagi 3 dan hasil pembagian akan menentukan arah pergeseran posisi kolom. jika bernilai 1 maka posisi kolom akan bergeser ke kiri dan jika bernilai 0 maka nilai kolom akan bergeser kekanan. Hal yang sama juga terjadi pada nilai baris. Implementasi pada program dapat dilihat pada Gambar 4.57.

```
if mod(countklm-bpklm,3)==0
    j5=j4-1;
elseif mod(countklm-bpklm,3)==1
    j5=j4+1;
elseif mod(countklm-bpklm,3)==2
    j5=j4;
end
```

Gambar 4.57 *Listing* program relokasi pixel

Listing program pada gambar 4.57 seleksi kondisi pertama adalah relokasi nilai kolom pixel yang disesuaikan dengan tabel 4.5. Untuk relokasi nilai baris memiliki logika yang sama dengan relokasi nilai kolom. Seleksi kondisi kedua adalah penempatan nilai pixel dari variabel **gbrawal** (hasil sistem posisi awal) ke variabel **gbrbp** (gambar penanganan bit posisi). Penempatan nilai pixel dilakukan selama nilai **bptot** tidak sama dengan nol. Nilai **bptot** dapat bernilai nol jika yang diproses adalah pixel *background* (hitam RGB=0). Hal yang sama untuk relokasi pixel juga dilakukan pada gambar bayang.



Gambar 4.58 Hasil Proses Kesalahan Posisi

Kemudian karena citra memiliki *background*, dilakukan pemotongan menggunakan sistem pemotongan *background*. Pada gambar bayang dilakukan hal yang sama yang terjadi pada gambar

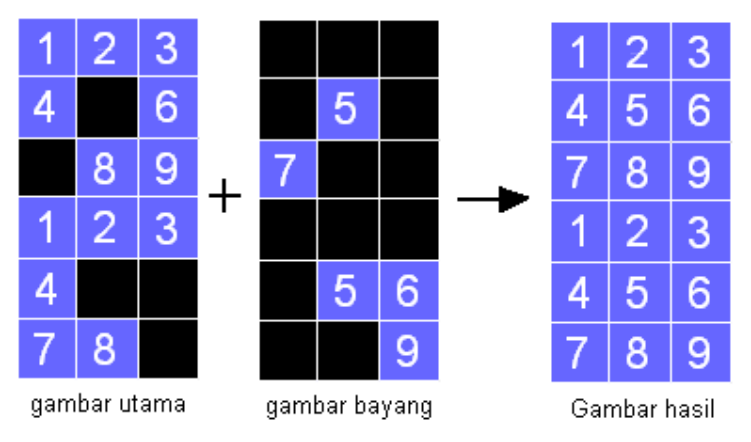
utama. Sehingga tiap pixel dari gambar bayang juga berada pada posisi yang seharusnya.

Setelah kedua gambar kembali pada posisi sebenarnya, maka dilakukan penggabungan citra. Penggabungan dilakukan agar pixel yang dipindahkan pada gambar bayang dapat kembali pada citra utama. Metode penggabungan cukup sederhana, yauitu dengan mendeteksi jika pada citra utama terdapat pixel dengan nilai nol tiap framenya, maka akan diambil nilai pixel gambar bayang pada posisi tersebut. Sehingga lubang pada gambar utama terisi dengan nilai pixel pada gambar bayang. Hal ini menjadikan keluaran sistem penanganan kesalahan posisi pixel adalah satu buah gambar utama. Gambar 4.59 adalah *listing* program dari implementasi rumus perhitungan nilai BP.

```
i6=1;
while i6 <= jbrs
    j6=1;
    while j6<= jklm
        if gbr1(i6,j6,:)==0
            gbr3(i6,j6,:)=gbr2(i6,j6,:);
        else
            gbr3(i6,j6,:)=gbr1(i6,j6,:);
        end
```

Gambar 4.59 *Listing* program penggabungan citra

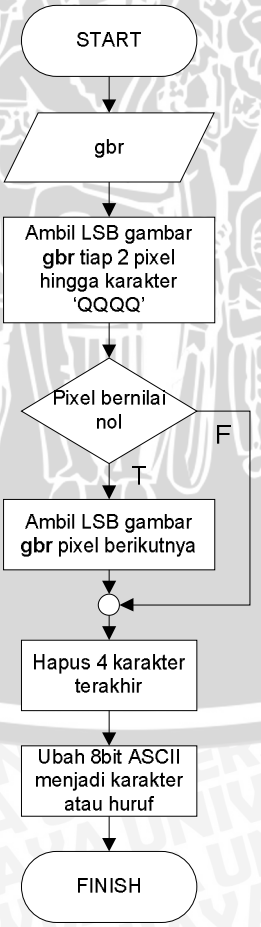
Pada Gambar 4.59 terdapat 3 buah variabel citra. Variabel gbr1 adalah gambar utama, gbr2 adalah gambar bayang, dan gbr3 adalah hasil penggabungan dari gambar utama dan gambar bayang. Pada seleksi kondisi menyatakan bahwa jika pada variabel gbr1 bernilai nol, maka pada variabel gbr3 diisi dengan nilai pixel pada variabel gbr2. Selain itu maka gbr3 diisi dengan nilai pixel pada variabel gbr1.



Gambar 4.60 Hasil Proses Penggabungan Citra

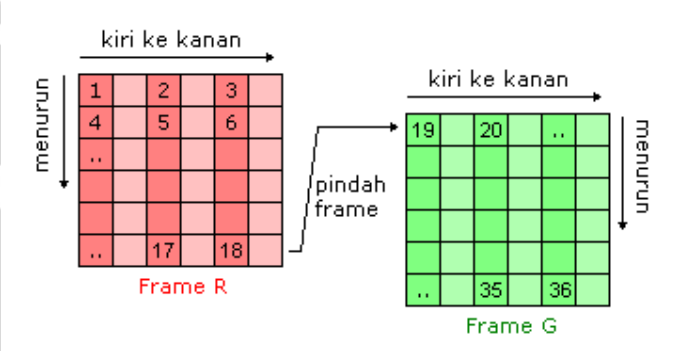
4.4.7 Ekstraksi Pesan

Ekstraksi pesan adalah pembacaan pesan yang telah disisipkan pada citra dan ditampilkan pada sistem. Gambar 4.61 adalah diagram alir sistem ekstraksi pesan.



Gambar 4.61 Diagram Alir Ekstraksi pesan

Masukan dari ekstraksi pesan adalah citra yang telah kembali pada posisi sebelum mengalami distorsi. Dalam ekstraksi pesan dilakukan dengan cara mengambil nilai LSB tiap 2 pixel secara berurutan. Karena setiap bit pesan disisipkan dalam 2 buah pixel. Urutan pengambilan dapat dilihat pada gambar 4.62.



Gambar 4.62 Gambar Urutan Pengambilan Data

Apabila nilai pixel yang akan diambil nilai LSB-nya bernilai nol (R=0, G=0, B=0), maka pengambilan nilai pixel akan dilakukan pada posisi pixel setelah pixel tersebut (nilai kolom +1). Hal ini dilakukan karena saat rotasi pada sudut tertentu terjadi hilangnya beberapa pixel. Sehingga dilakukan antisipasi agar pesan tetap dapat terbaca. *Listing* program pengambilan data dapat dilihat pada Gambar 4.63.

```
if gbr(k,l,:)==0
    if l+1<=jklm
        lsb=bitget(uint8(gbr(k,l+1,fr)),1);
    else
        if k+1<=jbrs
            lsb=bitget(uint8(gbr(k+1,1,fr)),1);
        else
            lsb=bitget(uint8(gbr(k,l,fr)),1);
        end
    end
```

Gambar 4.63 *Listing* program pengambilan data

Seleksi kondisi pada Gambar 4.63 menyatakan bahwa jika nilai pixel pada **gbr** (gambar masukan sistem ekstraksi) bernilai nol, maka lsb akan diambil dari nilai pixel berikutnya `(k,l+1)` atau `(k+1,1)` jika pixel selanjutnya pada baris melebihi ukuran citra. Jika tidak bernilai nol maka nilai lsb pada pixel tersebut yang diambil. Baris terakhir adalah menyusun bit lsb yang diambil dalam sebuah urutan bit dan dimasukkan pada variabel **rowpesan**.

Pengambilan nilai pixel dilakukan hingga 4 buah karakter 'Q' secara berurutan ditemukan. Apabila ditemukan satu karakter 'Q' kemudian karakter berikutnya bukan karakter 'Q', maka perhitungan karakter akan kembali menjadi nol. Dianggap karakter tersebut adalah bagian dari pesan.

```
if mod(length(rowpesan),8)==0
    n=(length(rowpesan))/8;
    while i<n && q<4
        j=(8*i)+1;
        qui=rowpesan(j:j+7);
        if qui=='01010001'
            q=q+1;
        else
            q=0;
        end
```

Gambar 4.64 *Listing* program penyusunan pesan

Seleksi kondisi pertama pada *listing* program Gambar 4.64 menyatakan bahwa program tersebut hanya dilakukan saat susunan biner pesan yang telah telah dimasukkan dalam variabel **rowpesan** memiliki nilai kelipatan delapan. Seleksi kondisi kedua adalah pembandingan nilai dengan biner ASCII karakter huruf 'Q' yaitu `01010001`. Nilai variabel **a** adalah susunan baris satu kolom dari tiap delapan biner pesan yang diambil untuk memudahkan perubahan dari biner ASCII menjadi sebuah karakter.

Setelah pengambilan pesan selesai dilakukan, nilai biner pesan disusun ke dalam sebuah array. Dalam susunan tersebut, dilakukan

penghapusan 4 karakter terakhir. Karena karakter tersebut bukanlah karakter pesan, melainkan karakter 'Q' yang disisipkan sebagai penanda akhir pesan. Terakhir, susunan biner pesan diubah dalam bentuk karakter.

4.4.8 Keluaran Sistem Decoding

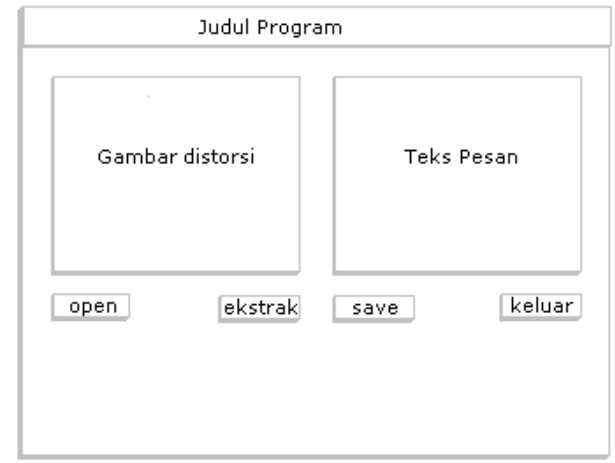
Keluaran sistem decoding adalah sebuah teks pesan. Nilai citra yang telah diproses tidak dikeluarkan karena hanya pixel dengan pesan yang kembali pada posisi yang sebenarnya. Tidak dilakukan pengembalian citra secara utuh karena hal tersebut akan memperlambat kinerja sistem. Tujuan utama sistem adalah membaca pesan yang tersembunyi dalam citra yang telah mendapat distorsi bukan mengembalikan citra yang telah mendapat distorsi kembali secara utuh pada posisi sebenarnya.

4.4.9 *User Interface* Sistem *Decoding*

Fungsi yang terdapat dalam *Interface decoding* meliputi:

1. Judul Program, digunakan agar dapat mengetahui nama program yang berjalan.
2. Panel gambar distorsi, berfungsi untuk meletakkan gambar distorsi yang diambil dari komputer.
3. Panel teks pesan, berfungsi untuk menampilkan hasil ekstraksi pesan yang telah disisipkan.
4. Tombol open, berfungsi untuk membuka file gambar distorsi dan diletakkan pada panel Gambar distorsi.
5. Tombol ekstrak, berfungsi untuk mengekstraksi teks pesan pada gambar distorsi.
6. Tombol save, berfungsi untuk menyimpan teks pesan kedalam komputer dengan ekstensi txt.
7. Tombol keluar, berfungsi untuk keluar dari program

Berikut adalah gambar bentuk rancangan *interface decoding*:



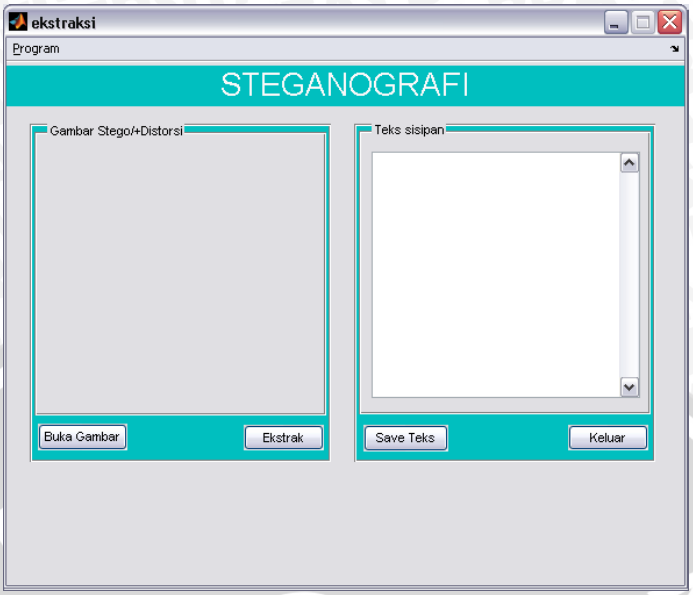
Gambar 4.65 Rancangan *Interface Decoding*

Beberapa hal yang perlu diperhatikan dalam pembuatan *user interface* Sistem *Decoding* menggunakan bahasa pemrograman matlab dapat dilihat dalam tabel 4.6.

no	Style	Name/String	Tag	visible
1	text	STEGANOGRAFI	text1	on
2	axes	-	gbrdistoaxes	off
3	text	-	txtsisip2	on
4	pushbutton	Save Teks	btnsavetxt	on
5	pushbutton	Keluar	btnoutekstrak	on
6	pushbutton	Buka Gambar	btnopendisto	on
7	pushbutton	Ekstrak	btnekstrak	on

Tabel 4.6 Tabel Implementasi Sistem *Encoding*

Sehingga dari tabel 4.6 dideroleh hasil implementasi dari perancangan sistem *User Interface Decoding* seperti pada Gambar 4.66



Gambar 4.66 Implementasi *Interface Encoding*

BAB V
PENGUJIAN DAN ANALISIS

Pada bab ini dibahas mengenai pengujian dan analisis sistem dimana akan terlihat kekurangan – kekurangan pada perangkat lunak untuk selanjutnya diadakan pengembangan sistem.

5.1 Pengujian dan Analisis Perubahan Citra Hasil Saat Penyisipan

Pada pengujian sistem terhadap perubahan citra saat penyisipan dilakukan dengan menggunakan beberapa jenis metode pengujian sebagai berikut,

1. Penyisipan 5 ukuran citra masukan yang berbeda berekstensi bmp dengan panjang karakter pesan yang sama.
2. Penyisipan dengan 5 macam panjang karakter pesan yang berbeda pada ukuran citra yang sama dan ekstensi citra bmp.
3. Penyisipan pada 5 macam ekstensi citra dengan ukuran citra dan panjang pesan yang sama.

Sehingga total pengujian adalah 15 kali dengan menggunakan 3 metode berbeda. Parameter keberhasilan pada sistem adalah besar persen perubahan citra sehingga dapat diketahui apakah perubahan citra masih dapat dipersepsi oleh visual manusia. Berikut adalah rumus singkat perhitungan besar perubahan nilai yang terjadi pada citra.

$$\%P = \frac{\sum | pixA - pixB |}{\sum pixA} \times 100\%$$

Keterangan:

%P : Nilai persentase perubahan citra

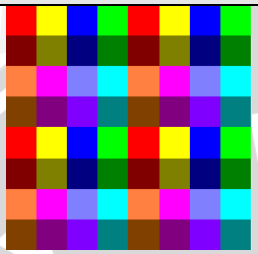
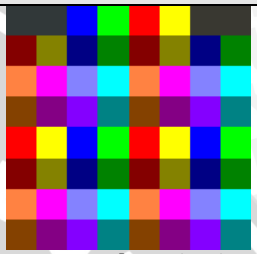
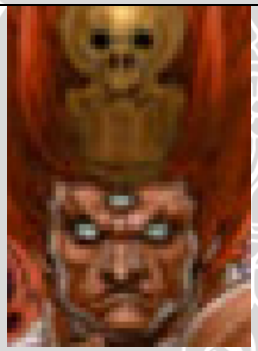
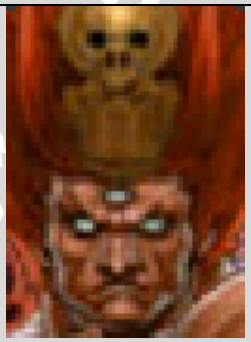
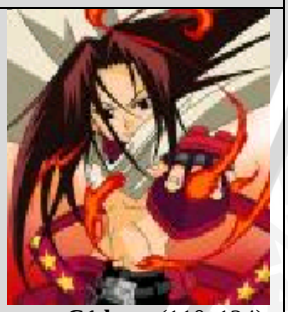
pixA : Nilai pixel citra asli

pixB : Nilai pixel citra stego

5.1.1 Perubahan Ukuran Citra

Dalam pengujian yang diperiksa adalah besar perubahan citra. Penyisipan dilakukan pada 5 ukuran citra masukan yang berbeda berekstensi bmp dengan panjang karakter pesan yang sama.

Pada Tabel 5.1 adalah tabel pengujian menggunakan metode pertama yaitu perubahan ukuran citra.

no	Citra Asli	Pesan	Pjg char	Citra Stego
1	 **A.bmp** (8x8) MAX char: 4	Halo	4	 **stegoA1.bmp** (8x8) **%P : 8.40361**
2	 **B.bmp** (50x70) MAX char: 433	Halo	4	 **stegoB1.bmp** (50x70) **%P : 0.20134**
3	 **C.bmp** (110x124) MAX char: 1701	Halo	4	 **stegoC1.bmp** (110x124) **%P : 0.05917**

4	 **D.bmp** (100x200) MAX char: 2496	Halo	4	 **stegoD1.bmp** (100x200) **%P : 0.02201**
5	 **E.bmp** (300x300) MAX char: 11246	Halo	4	 **stegoE1.bmp** (300x300) **%P : 0.01063**

Tabel 5.1 Tabel Pengujian Persen Perubahan Metode 1

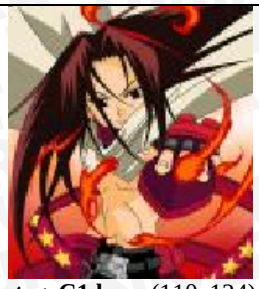
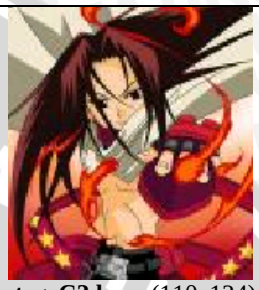
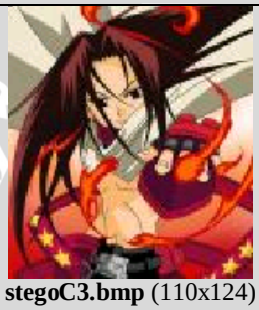
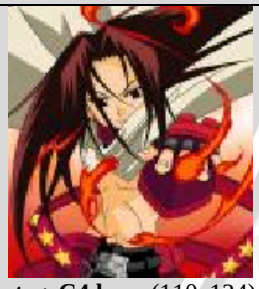
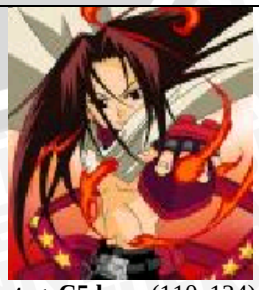
Dari Tabel 5.1 dapat dilihat bahwa nilai persen perubahan (%P) yang terjadi saat penyisipan pesan dilakukan memiliki nilai yang relatif kecil. Citra Stego yang didapat secara kasat mata tidak memiliki perbedaan dengan Citra Asli, kecuali mungkin pada citra dengan ukuran paling kecil yaitu 8x8 pixel. Namun apabila diteliti, perubahan paling mencolok adalah pada ujung atas kiri dan kanan Citra Stego. Dari hasil pengujian didapat bahwa apabila ukuran citra semakin besar, maka nilai persen perubahan semakin kecil.

5.1.2 Perubahan Ukuran Pesan

Dalam pengujian yang diperiksa adalah besar perubahan citra. Penyisipan dilakukan pada 5 macam panjang karakter pesan yang berbeda pada ukuran citra yang sama dan ekstensi citra bmp.

Pada Tabel 5.2 adalah tabel pengujian menggunakan metode kedua yaitu perubahan panjang karakter pesan.

no	Citra Asli	Pesan	Pjg char	Citra Stego

1	 **C.bmp** (110x124) MAX char: 1701	Halo	4	 **stegoC1.bmp** (110x124) **%P : 0.05917**
2	 **C.bmp** (110x124) MAX char: 1701	Hari ini cuaca CERAH	20	 **stegoC2.bmp** (110x124) **%P : 0.08003**
3	 **C.bmp** (110x124) MAX char: 1701	steganografi adalah teknik penyisipan pesan pada suatu media	60	 **stegoC3.bmp** (110x124) **%P : 0.13492**
4	 **C.bmp** (110x124) MAX char: 1701	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^&*()_+\ dan lainnya	125	 **stegoC4.bmp** (110x124) **%P : 0.23076**
5	 **C.bmp** (110x124) MAX char: 1701	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^&*()_+\ dan lainnya	250	 **stegoC5.bmp** (110x124) **%P : 0.40749**

		steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^&*()_+\ dan lainnya		

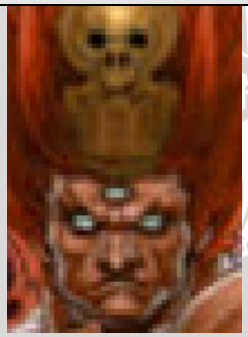
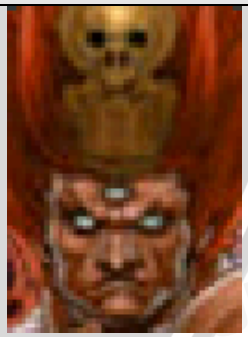
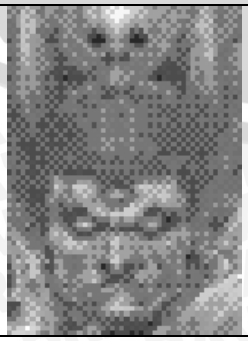
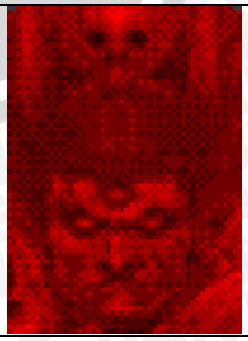
Tabel 5.2 Tabel Pengujian Persen Perubahan Metode 2

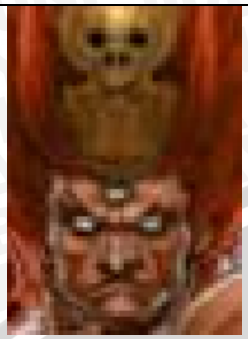
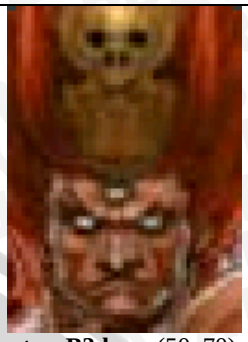
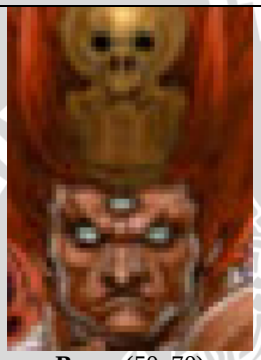
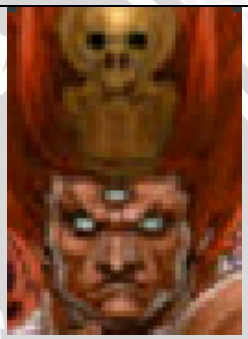
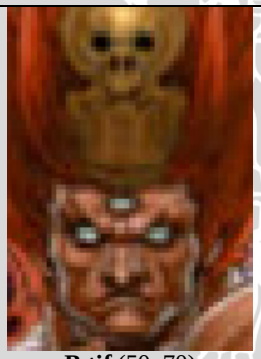
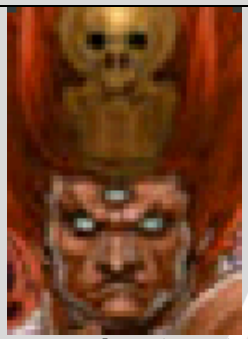
Dari Tabel 5.2 dapat dilihat bahwa Citra Stego yang didapat secara kasat mata tidak memiliki perbedaan dengan Citra Asli. Dari hasil pengujian didapat bahwa apabila ukuran pesan semakin panjang, maka nilai persen perubahan semakin besar.

5.1.3 Perubahan Ekstensi File

Dalam pengujian yang diperiksa adalah besar perubahan citra. Penyisipan dilakukan pada 4 macam ekstensi citra dengan ukuran citra dan panjang pesan yang sama.

Pada Tabel 5.3 adalah tabel pengujian menggunakan metode ketiga yaitu perubahan ekstensi file.

no	Citra Asli	Pesan	Pjg char	Citra Stego
1	 **B.bmp** (50x70) MAX char: 433	pengujian terhadap ekstensi file	32	 **stegoB1b.bmp** (50x70) **%P : 0.44799**
2		pengujian terhadap ekstensi file	32	

	B.gif (50x70) MAX char: 433			**stegoB2.bmp** (50x70) **%P : 0.20086**
3	 **B.jpg** (50x70) MAX char: 433	pengujian terhadap ekstensi file	32	 **stegoB3.bmp** (50x70) **%P : 0.44210**
4	 **B.png** (50x70) MAX char: 433	pengujian terhadap ekstensi file	32	 **stegoB4.bmp** (50x70) **%P : 0.44799**
5	 **B.tif** (50x70) MAX char: 433	pengujian terhadap ekstensi file	32	 **stegoB5.bmp** (50x70) **%P : 0.44799**

Tabel 5.3 Tabel Pengujian Persen Perubahan Metode 3

Dari Tabel 5.3 dapat dilihat bahwa terjadi perbedaan pada gambar dengan ekstensi gif. Citra dengan ekstensi gif oleh sistem hanya terbaca dalam sebuah frame sehingga pada saat dibuka menghasilkan citra grayscale. Hasil dari penyisipan akan disimpan dalam ekstensi bmp yang memiliki 3 frame sehingga penyimpanan satu frame gif diletakkan pada frame R saja. Nilai pada frame G dan B pada Citra hasil memiliki nilai default yaitu nol.

Secara umum selain pada citra dengan ekstensi gif, Citra Stego yang didapat secara kasat mata tidak memiliki perbedaan dengan Citra Asli. Nilai persen perubahan dari keempat citra memiliki nilai yang hampir sama. Sehingga dari hasil pengujian didapat bahwa masukan citra dapat berupa citra dengan ekstensi bmp, jpg, png dan tif, namun tidak disarankan menggunakan ekstensi file gif.

5.2 Pengujian dan Analisis Tingkat Keberhasilan Sistem Saat Ekstraksi

Pada pengujian sistem terhadap tingkat keberhasilan saat ekstraksi pesan dilakukan dengan menggunakan beberapa jenis metode pengujian sebagai berikut,

1. Pengujian perubahan ukuran citra, hasil citra dari penyisipan pada tabel 5.1 diberi 3 macam distorsi pada tiap citra kemudian dilakukan ekstraksi pesan.
2. Pengujian perubahan ukuran pesan, hasil citra dari penyisipan pada tabel 5.2 diberi 3 macam distorsi pada tiap citra kemudian dilakukan ekstraksi pesan.
3. Pengujian perubahan sudut rotasi, 2 macam citra stego diberi 15 variasi sudut rotasi yang dianggap mewakili seluruh sudut 360 derajat, kemudian dilakukan ekstraksi pesan

Sehingga total pengujian adalah 60 kali dengan menggunakan 3 metode berbeda. Parameter keberhasilan pada sistem adalah besar persen kesalahan pesan sehingga dapat diketahui kemampuan sistem dalam mengatasi distorsi. Berikut adalah rumus singkat perhitungan besar persen kesalahan saat ekstraksi pesan.

$$\%K = \frac{\sum Echar}{\sum char} \times 100\%$$

Keterangan:

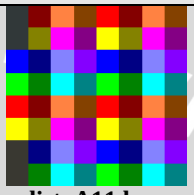
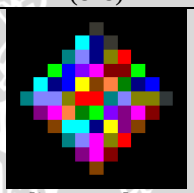
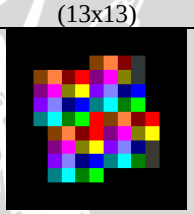
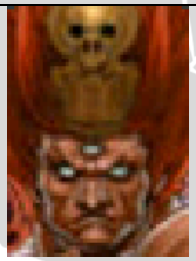
%K : Nilai persentase kesalahan

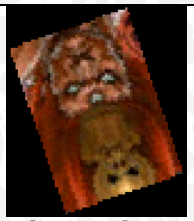
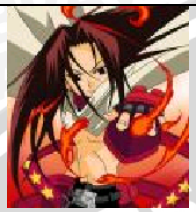
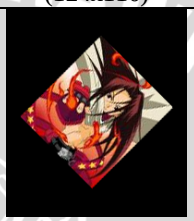
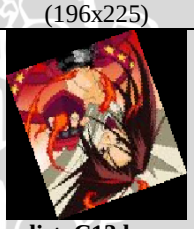
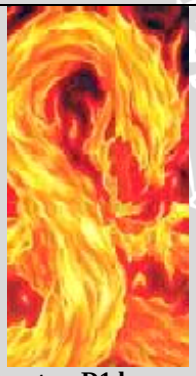
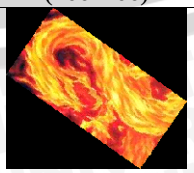
Echar : Karakter pesan yang mengalami error

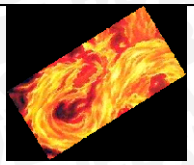
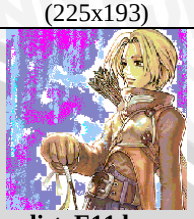
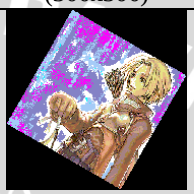
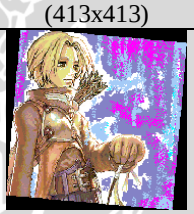
char : Karakter pesan asli

5.2.1 Perubahan Ukuran Citra Saat Ekstraksi

Dalam pengujian yang diperiksa adalah besar kesalahan pesan. Ekstraksi pesan dilakukan pada 5 ukuran citra masukan yang berbeda berekstensi bmp dengan panjang karakter pesan yang sama (hasil dari pengujian Tabel 5.1).

NO	Citra Stego	Pesan Asli	disto	Citra Disto	Pesan Hasil
1	 **stegoA1.bmp** (8x8)	Halo	R: 90 VF	 **distoA11.bmp** (8x8)	Halo **%K: 0**
		Halo	R: 48 HF	 **distoA12.bmp** (13x13)	Halo **%K: 0**
		Halo	R:284	 **distoA13.bmp** (13x13)	Halo **%K: 0**
2	 **stegoB1.bmp** (50x70)	Halo	R:270 HF	 **distoB11.bmp** (70x50)	Halo **%K: 0**
		Halo	R: 45 VF	 **distoB12.bmp** (85x85)	Halo **%K: 0**

		Halo	R:199	 **distoB13.bmp** (73x85)	Halo **%K: 0**
3	 **stegoC1.bmp** (110x124)	Halo	R: 90 VF eksternal Paint	 **distoC11.bmp** (124x110)	Halo **%K: 0**
		Halo	R:135 VF eksternal Photo-Shop CS	 **distoC12.bmp** (196x225)	ERROR Posisi mark tidak diketahui **%K: 100**
		Halo	R:880 eksternal Corel - Photo-Paint X4	 **distoC13.bmp** (146x156)	Halo **%K: 0**
4	 **stegoD1.bmp** (100x200)	Halo	R: 0 HF	 **distoD11.bmp** (100x200)	Halo **%K: 0**
		Halo	R:305 HF	 **distoD12.bmp** (221x197)	Halo **%K: 0**

		Halo	R:122	 **distoD13.bmp** (225x193)	Halo **%K: 0**
5	 **stegoE1.bmp** (300x300)	Halo	R:180 VF	 **distoE11.bmp** (300x300)	Halo **%K: 0**
		Halo	R:210 VF	 **distoE12.bmp** (413x413)	Halo **%K: 0**
		Halo	R:715	 **distoE13.bmp** (325x325)	Halo **%K: 0**
Rata-rata kesalahan (jumlah %K / jumlah percobaan)					**%K: 6,67**

Tabel 5.4 Tabel Pengujian Persen Kesalahan Metode 1

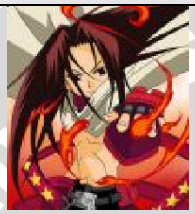
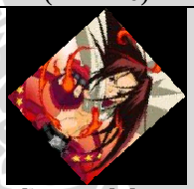
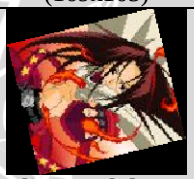
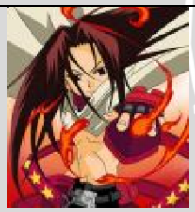
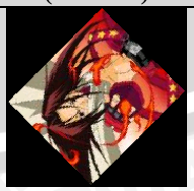
Dari Tabel 5.4 secara keseluruhan sistem dapat menangani distorsi pada citra dengan ukuran dan distorsi yang berbeda. Error yang terjadi pada Citra distoC12.bmp adalah karena sistem tidak dapat menemukan letak marking. Hal tersebut dikarenakan citra diberi distorsi eksternal yaitu pada photoshop CS. Rotasi pada program photoshop CS tidak memakai interpolasi terdekat, sehingga terjadi perubahan nilai pada tiap pixel citra. Perubahan nilai akan membuat sistem tidak dapat mengenali penanda dan pesan yang terdapat didalam citra juga akan rusak.

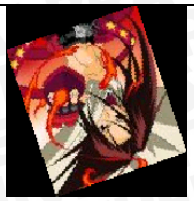
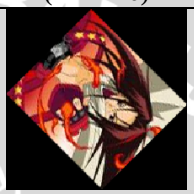
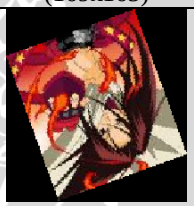
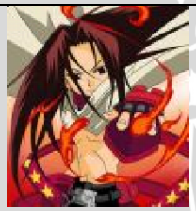
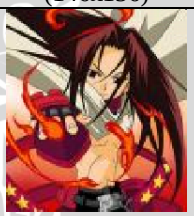
Kesalahan pesan yang terjadi pada citra distoD13.bmp terdapat pada penanda akhir pesan, yaitu karakter ’QQQQ’. Pesan asli tidak mengalami perubahan, namun jika kesalahan terjadi pada penanda akhir pesan maka sistem akan melanjutkan mengambil nilai LSB hingga akhir.

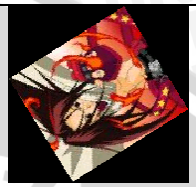
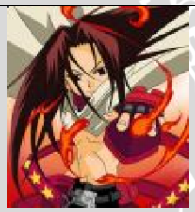
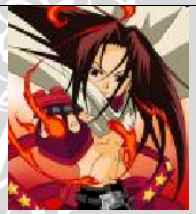
Sehingga karakter yang tidak termasuk dalam pesan akan diambil dan karakter penanda juga akan dianggap sebagai pesan.

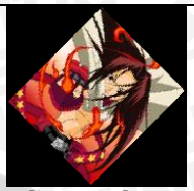
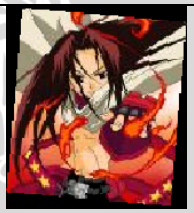
5.2.1 Perubahan Ukuran Pesan Saat Ekstraksi

Dalam pengujian yang diperiksa adalah besar kesalahan pesan. Ekstraksi pesan dilakukan pada 5 ukuran citra masukan yang sama berekstensi bmp dengan panjang karakter pesan sisipan yang berbeda (hasil dari pengujian Tabel 5.2).

NO	Citra Stego	Pesan Asli	disto	Citra Disto	Pesan Hasil
1	 **stegoC1.bmp** (110x124)	Halo	R: 90 VF	 **distoC11b.bmp** (124x110)	Halo **%K: 0**
		Halo	R: 48 HF	 **distoC12b.bmp** (169x165)	Halo **%K: 0**
		Halo	R:284	 **distoC13b.bmp** (149x137)	Halo **%K: 0**
2	 **stegoC2.bmp** (110x124)	Hari ini cuaca CERAH	R:270 HF	 **distoC21.bmp** (124x110)	Hari ini cuaca CERAH **%K: 0**
		Hari ini cuaca CERAH	R: 45 VF	 **distoC22.bmp** (169x169)	Hari ini cuaca CERAH **%K: 0**

		Hari ini cuaca CERAH	R:199	 **distoC23.bmp** (145x153)	Hari ini cuaca CERAH **%K: 0**
3	 **stegoC3.bmp** (110x124)	steganografi adalah teknik penyisipan pesan pada suatu media	R: 90 VF eksternal Paint	 **distoC21.bmp** (124x110)	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	R:135 VF eksternal Photo-Shop CS	 **distoC32.bmp** (169x165)	ERROR Posisi mark tidak diketahui **%K: 100**
		steganografi adalah teknik penyisipan pesan pada suatu media	R:880 eksternal Corel - Photo-Paint X4	 **distoC23.bmp** (146x156)	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
4	 **stegoC4.bmp** (110x124)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya	R: 0 HF	 **distoC41.bmp** (110x124)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^&*()_+\ dan lainnya **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media.	R:315 HF	 **distoC42.bmp**	steganografi adalah teknik penyisipan pesan pada suatu media. dapat

		dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya		(169x169)	menyisipkan berbagai karakter seperti !@#$%^&*()_+\ dan lainnya **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya	R:122	 **distoC43.bmp** (165x161)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya **%K: 0**
5	 **stegoC5.bmp** (110x124)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^	R:180 VF	 **distoC51.bmp** (110x124)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya

		&*()_+\ dan lainnya			**%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya	R:225 VF	 **distoC52.bmp** (169x169)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan	R:715	 **distoC53.bmp** (121x137)	steganografi adalah teknik penyisipan pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya steganografi adalah teknik penyisipan pesan pada suatu media.

		pesan pada suatu media. dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya			dapat menyisipkan berbagai karakter seperti !@#$%^ &*()_+\ dan lainnya **%K: 0**
Rata-rata kesalahan (jumlah %K / jumlah percobaan)					**%K: 6.67**

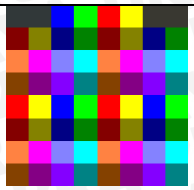
Tabel 5.5 Tabel Pengujian Persen Kesalahan Metode 2

Dari Tabel 5.5 secara keseluruhan sistem dapat menangani distorsi pada citra dengan ukuran pesan dan distorsi yang berbeda. Error yang terjadi pada Citra distoC32.bmp sama seperti pada pengujian sebelumnya yaitu karena sistem tidak dapat menemukan letak marking.

Kesalahan pesan yang terjadi pada citra distoC12b.bmp dan distoC21.bmp juga terdapat pada penanda akhir pesan, yaitu karakter ’QQQQ’. Pada pesan asli juga beberapa mengalami perubahan karena terdapat hilangnya pixel pada saat rotasi diberikan pada citra stego. Dalam penanganan pixel yang hilang dilakukan penyisipan satu bit pesan dalan dua pixel dan pembacaan data dilakukan pada pixel berikutnya jika pixel pertama hilang. Namun apabila dua pixel yang disisipi satu bit pesan ternyata hilang keduanya, maka sistem akan mengambil nilai nol. Jika bit yang hilang memiliki nilai satu, maka hasil saat pengubahan biner ke dalam karekter ASCII akan mengalami pergeseran. Sehingga pesan yang diperoleh berbeda dengan pesan asli.

5.2.1 Perubahan Besar Sudut Saat Ekstraksi

Dalam pengujian yang diperiksa adalah besar kesalahan pesan. Ekstraksi pesan dilakukan 2 macam citra stego dan diberi 15 macam sudut rotasi.

NO	Citra Stego	Pesan Asli	Sudut	Pesan Hasil	
1		Halo	0	Halo	**%K: 0**
		Halo	1	Halo	**%K: 0**
		Halo	2	Halo	**%K: 0**
		Halo	3	Halo	**%K: 0**

	stegoA1.bmp (8x8)	Halo	4	Halo **%K: 0**
		Halo	5	Halo **%K: 0**
		Halo	10	Halo **%K: 0**
		Halo	15	Halo **%K: 0**
		Halo	20	Halo **%K: 0**
		Halo	25	Halo **%K: 0**
		Halo	30	Halo **%K: 0**
		Halo	45	Halo **%K: 0**
		Halo	60	Halo **%K: 0**
		Halo	75	Halo **%K: 0**
		Halo	90	Halo **%K: 0**
	 **tesstegoD.bmp** (100x200)	steganografi adalah teknik penyisipan pesan pada suatu media	0	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	1	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	2	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	3	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	4	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	5	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan	10	ERROR 1 ??? Undefined function

		pesan pada suatu media		or variable "pos1". **%K: 100**
		steganografi adalah teknik penyisipan pesan pada suatu media	15	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	20	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	25	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	30	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	45	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	60	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	75	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
		steganografi adalah teknik penyisipan pesan pada suatu media	90	steganografi adalah teknik penyisipan pesan pada suatu media **%K: 0**
Rata-rata kesalahan (jumlah %K / jumlah percobaan)				**%K: 3.33**

Tabel 5.6 Tabel Pengujian Persen Kesalahan Metode 3

Pada Tabel 5.6 ERROR1 terjadi karena kesalahan 1 derajat pada saat pencarian nilai sudut dilakukan. Sehingga citra terjadi pergeseran dan tidak berbentuk kotak (masih terdapat sudut kemiringan 1 derajat)

dalam sistem penanganan citra miring. Karena masih terdapat sudut maka pada sistem pengembalian posisi awal posisi *mark*1 tidak dapat ditemukan. Dari pengujian didapat bahwa setiap citra stego memiliki daya tahan terhadap beberapa sudut distorsi. Hilangnya nilai penanda (*mark*1 dan *mark*2) tergantung pada besar sudut dan nilai ukuran citra.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian dan analisis yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Sistem Penyisipan pesan menggunakan metode LSB dapat dilakukan pada citra masukan yang dapat diubah menjadi citra RGB 24 bit misal BMP, JPG, PNG dan TIF.
2. Citra hasil penyisipan dengan ukuran paling kecil yaitu 8x8 pixel mengalami perubahan sebesar 8.4% dari citra asli. Sedangkan pada ukuran 300x300 pixel mengalami perubahan sebesar 0.01% sehingga hampir tidak dapat dibedakan antara citra asli dan citra hasil penyisipan.
3. Dari hasil pengujian ekstraksi pesan, terdapat 2 sebab yang mengakibatkan terjadinya error pada sistem yaitu:
   a. Kesalahan pembacaan sudut de-rotasi ($\pm 1^{o}$)
   b. Citra stego dirotasi tidak dengan metode interpolasi terdekat, misal menggunakan Photoshop.
4. Dari keseluruhan pengujian ekstraksi pesan, sistem memiliki tingkat keberhasilan minimum sebesar 93.33%.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem penanganan distorsi pada steganografi ini antara lain :

1. Sebaiknya dicoba dilakukan metode selain LSB. Untuk mengetahui apakah metode penanganan distorsi juga berlaku pada metode lain.
2. Ditambahkan penanganan beberapa jenis distorsi posisi lain seperti *skew* atau lainnya.
3. Digunakan jenis-jenis ekstensi file lainnya pada masukan dan keluaran sistem penyisipan. Karena citra bitmap umum digunakan dalam steganografi.

4. Ditambahkan fungsi untuk menangani jika terjadi kesalahan pembacaan nilai sudut dan penanganan hilangnya 2 pixel yang bersebelahan saat pemberian distorsi.

DAFTAR PUSTAKA

Anonim. *Bitmap*. http://id.wikipedia.org/wiki/Bitmap (diakses 18 Mei 2011).

Anonim. *Steganografi*. http://id.wikipedia.org/wiki/Steganografi (diakses 20 April 2011).

Anonim. *MATLAB*. http://id.wikipedia.org/wiki/MATLAB (diakses 18 Mei 2011).

Anonim. *User's Guide Matlab Documentation*. http://www.mathwork.com/help/techdoc/matlab_product_page.html (diakses tanggal 21 Mei 2011).

Putra, Darma. 2010. *Pengolahan Citra Digital. Yogyakarta*: Andi Offset.

Paulus, Erick, Yessica Nataliani. 2007. *Cepat Mahir GUI MATLAB*. Yogyakarta: Andi Offset.

Wijaya, Marvin Ch, Agus Prijono. 2007. *Pengolahan Citra Digital Menggunakan MATLAB*. Bandung: Informatika.