IMPLEMENTASI METODE *LOG-POLAR MAPPING* DAN *PHASE CORRELATION* PADA *WATERMARKING* CITRA DIGITAL YANG INDEPENDEN TERHADAP ROTASI, SKALA, DAN TRANSLASI

TUGAS AKHIR

Disusun Oleh :

Trisno Wijayanto
NRP. 5198 100 087

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

IMPLEMENTASI METODE *LOG-POLAR MAPPING* DAN *PHASE CORRELATION* PADA *WATERMARKING* CITRA DIGITAL YANG INDEPENDEN TERHADAP ROTASI, SKALA, DAN TRANSLASI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan

Untuk Memperoleh Gelar Sarjana Komputer

Pada

Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui,

Dosen Pembimbing

Rully Soelaiman, M.Kom
NIP. 132 085 802

S U R A B A Y A

Juli, 2005

ABSTRAK

Watermarking adalah proses *embedding* (penambahan) dan ekstraksi data yang disebut *watermark* ke dalam sebuah media. *Watermark* ini bisa berbentuk sinyal, citra, atau barisan bit. Sedangkan media yang digunakan bisa berupa citra atau video atau audio. Fungsi utama dari *watermarking* adalah sebagai proteksi hak cipta dari media yang diberi *watermark.*

Banyak metode telah digunakan dalam *watermarking* ini. Salah satunya menggunakan metode *Log-Polar Mapping* (LPM) dan *Phase Correlation.* Metode inilah yang digunakan dalam tugas akhir ini. Metode LPM dan *Phase Correlation* digunakan dalam proses *embedding* dan ekstraksi *watermark.* Proses-proses ini dilakukan pada *magnitude* dari domain Fourier sebuah citra digital untuk membuat *watermarking* independen terhadap translasi. Metode LPM digunakan pada proses *embedding* untuk membuat *watermark* yang independen terhadap rotasi dan skala. Sedangkan metode *Phase Correlation* dalam proses ekstraksi *watermark* bersamaan dengan metode LPM digunakan dalam proses penyesuaian (*rectification*) citra digital. Dengan kedua metode ini posisi *watermark* dalam *magnitude* tetap terjaga sehingga *watermark* mudah diekstraksi. Selain itu *watermark* terhindar dari resiko pengurangan kualitas yang disebabkan oleh LPM.

Uji coba dilakukan menggunakan 5 citra digital dan berbagai macam *watermark.* Hasil uji coba menunjukkan terdeteksinya *watermark* pada hampir semua kasus, hal ini menunjukkan *watermarking* ini independen terhadap rotasi, skala, dan translasi (RST).

Kata Kunci : *Watermarking*, *Watermark*, Transformasi Fourier, *Log-Polar Mapping* dan *Phase Correlation.*

KATA PENGANTAR

Puji dan syukur ke hadirat Allah SWT atas rahmat dan hidayah-Nya, penulis dapat menyelesaikan Tugas Akhir yang berjudul :

Implementasi Metode *Log-Polar Mapping* dan *Phase Correlation* pada *Watermarking* Citra Digital yang Independen Terhadap Rotasi, Skala, dan Translasi

Tugas akhir ini dikerjakan sebagai salah satu persyaratan akademik guna mendapatkan gelar sarjana S-1 pada jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam kesempatan ini penulis ingin mengucapkan terima kasih dan penghargaan sebesar – besarnya bagi semua pihak yang membantu dalam pelaksanaan Tugas Akhir baik langsung maupun tidak langsung. Secara khusus penulis ingin menyampaikan terima kasih kepada :

Rully Soelaiman, S.Kom., M.Kom. selaku Dosen Pembimbing yang telah memberi bimbingan, petunjuk, dan arahan dalam pembuatan tugas akhir.

Dr. Ir. Joko Lianto B. M.Sc selaku dosen wali dan seluruh dosen jurusan Teknik Informatika atas bimbingan semasa perkuliahan.

Kedua orang tua dan ketiga saudara penulis yang selalu memberikan dorongan, nasihat, dan doa.

Decky FR, Kamil Zubair, Widyo Wicaksono, dan semua teman – teman, khususnya rekan – rekan C0E, terima kasih atas semua bantuan, kebersamaan dan dukungannya.

Akhir kata, selamat membaca dan semoga laporan Tugas Akhir ini dapat memberikan manfaat yang seluas-luasnya bagi kita semua.

Surabaya, Juli 2005

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Dewasa ini citra digital menjadi semakin umum dan dan mulai menggantikan citra analog dalam berbagai keperluan. Ada berbagai macam keuntungan dari citra digital, antara lain :

1. Kemudahan melakukan reproduksi terhadap citra digital, cukup dengan mengkopi file dari citra digital tersebut.
2. Kemudahan dalam menyimpan citra digital, misalnya sebuah CD bisa berisi citra yang apabila dalam bentuk analog membutuhkan ruang yang lebih besar daripada sebuah CD.
3. Kemudahan dalam melakukan distribusi citra digital, dengan adanya internet distribusi bisa dilakukan dengan mengirimkan *e-mail* yang akan sampai dalam waktu singkat.

Kemajuan dalam bidang Internet juga memicu makin luasnya penggunaan citra digital. Namun perluasan dalam penggunaan citra digital ini juga membuat pembajakan terhadap citra digital tersebut semakin marak. Ditambah dengan kemajuan internet, maka distribusi dari pembajakan tersebut juga semakin mudah.

Pembajakan inilah yang menjadi faktor utama yang memicu riset dibidang *watermark* mendapat perhatian dari kalangan penerbit, artis dan pihak-pihak yang tertarik dengan perlindungan hak cipta. Oleh karena itu sifat *robust* (tahan) terhadap serangan oleh pembajak dari *watermark* adalah faktor utama untuk sebuah *watermark* menjadi berguna [1], hal ini termasuk *robust* terhadap kompresi seperti JPG, skala dan perubahan rasio aspek, rotasi, pemotongan, pembuangan baris atau kolom, penambahan noise, *filtering*, kriptografi dan serangan statistikal, dan juga *embedding watermark* lain.

Ada banyak metode *watermarking* yang telah dikembangkan, namun beberapa metode lama yang tidak *robust* [6], antara lain metode dari L.F.Turner yang menggunakan metode *watermarking* yang memasukkan kode identifikasi ke dalam sinyal digital audio dengan cara mensubstitusikan bit yang tidak signifikan

dari sampel yang dipilih secara random dengan bit dari kode identifikasi. Bit dianggap tidak signifikan jika dilakukan perubahan terhadap bit tersebut maka perubahan tersebut tidak terdengar. Metode tersebut bisa diaplikasikan ke dalam citra dua dimensi. Sayangnya metode ini mudah dikalahkan, apabila diketahui metode ini hanya mempengaruhi 2 bit yang paling tidak signifikan, maka hanya dengan melakukan *flip* terhadap bit tersebut maka *watermark* dapat dihancurkan. Metode lain adalah metode dari Caronni yang menambahkan sebuah pola geometrik kecil ke dalam bagian yang memiliki keterangan tinggi sehingga tidak terlihat, metode ini tidak *robust* terhadap perubahan geometrik umum. Begitu pula dengan metode dari Brassil, yang menggunakan teks yang ada dalam citra. *Watermark* dikodekan dengan menggeser secara vertikal baris teks, menggeser secara horizontal kata-kata, atau merubah sifat teks seperti garis vertikal dari tiap-taip huruf. Metode inipun tidak *robust*.

Salah satu metode *watermarking* yang *robust* adalah metode *watermarking* yang menggunakan domain frekuensi (pada umumnya menggunakan transformasi Fourier) sebagai tempat *embedding watermark*. Untuk dapat mengerti keuntungan dari penggunaan domain frekuensi dalam pemasangan *watermark*, perlu dilihat efek dari sebuah serangan terhadap spektrum frekuensi dari sinyal [8]. Serangan tersebut antara lain : Kompresi jenis *Lossy* adalah operasi yang biasanya membuang bagian yang tidak terlihat (*perceptually insignificant*) dari sebuah citra. Jika seseorang ingin melindungi *watermark* dari serangan tersebut, dia harus menaruh *watermark* di bagian yang terlihat (*perceptually significant*) dari data. Pada umumnya operasi-operasi jenis ini dilakukan dalam domain frekuensi. Pada kenyataannya, hilangnya data biasanya terjadi di bagian frekuensi tinggi dari data. Oleh karena itu *watermark* harus ditambahkan pada daerah bagian frekuensi yang signifikan (bagian frekuensi rendah) dari gambar. Distorsi geometrik bersifat spesifik untuk tiap citra. Dengan menentukan secara manual maksimum dari empat atau sembilan antara *watermark* yang terdistorsi dan *watermark* asli, adalah mungkin untuk membuang transformasi *affine* dari dua atau tiga dimensi. Namun, sebuah skala *affine* (pengecilan) dari citra akan mengarah pada hilangnya data pada daerah frekuensi tinggi. *Cropping* merupakan ancaman serius bagi

watermark yang ditambahkan pada domain spasial, tapi kemungkinan untuk mempengaruhi *watermark* yang menggunakan domain frekuensi adalah lebih kecil. Distorsi sinyal umum, yang memiliki sifat *nonlinear*, sulit untuk dianalisa efeknya terhadap baik domain frekuensi maupun spasial.

Metode Fourier-Mellin adalah metode transformasi yang independen terhadap serangan rotasi, skala dan translasi. Metode ini diusulkan pertama kali oleh O'Ruanaidh [7]. Pada pendekatannya *Discrete Fourier Transform* (DFT) dari citra dihitung kemudian transfomasi Fourier-Mellin dilakukan pada *magnitude*-nya, *watermark* ditambahkan pada hasil transformasi tersebut. Citra ber-*watermark* kemudian direkonstruksi dengan menggunakan transformasi *invers* (*invers* DFT dan *invers* Fourier-Mellin) setelah digabungkan kembali dengan *phase* dari DFT sebelumnya. Transformasi Fourier-Mellin adalah *log-polar mapping* (LPM) yang diikuti oleh *invers* dari tranformasi Fourier, sedangkan *invers* dari transformasi Fourier-Mellin adalah *invers* dari LPM yang diikuti oleh *invers* dari transformasi Fourier. Dalam domain *log-polar* ini sebuah serangan rotasi dan skala hanya akan menjadi pergeseran dari citra sehingga bisa dikatakan independen terhadap rotasi dan skala. Ekstraksi *watermark* dilakukan dengan cara melakukan transformasi citra ber-*watermark* ke dalam domain log-polar.

Metode O'Ruanaidh memiliki masalah dalam implementasinya, LPM dan *invers*-nya menghasilkan efek *ringing* pada citra hasil *embedding watermark* sehingga tidak dapat diterima kualitasnya. Untuk mengatasi hal itu maka D.Zheng [1] mengusulkan untuk melakukan *embedding watermark* tanpa melakukan LPM pada citra asli, tetapi yang mendapatkan LPM adalah *watermark*-nya. Untuk mengurangi efek negatif dari LPM maka LPM untuk *watermark* ini dilakukan dengan aproksimasi, disebut sebagai *approximate invers log-polar*. Dengan metode ini sifat independen *watermark* terhadap rotasi, skala dan translasi dapat dipertahankan dan efek negatif dari LPM dapat dikurangi.

Telah diterima secara luas bahwa *phase* memegang peranan penting pada representasi citra dan vision. *Phase* yang diambil dari DFT sebuah citra ini merujuk pada bentuk dari sebuah citra, sehingga bisa digunakan untuk mencari pinggiran dan pengenalan bentuk. Metode *phase correlation*, yaitu korelasi antar

dua *phase*, telah digunakan untuk mencocokkan dua citra yang secara relatif bergeser antara satu sama lain. Pencarian *watermark* selama ini menggunakan metode *exhaustive search* yang mencari tiap-tiap *pixel*, metode ini sangat memakan waktu dan juga ada kemungkinan menghasilkan nilai positif adanya *watermark* yang palsu. Metode *phase correlation* bisa diimplementasikan pada domain LPM dari *watermarking* untuk menggantikan metode *exhaustive search.*

1.2 PERMASALAHAN

Dalam *Watermarking* dengan menggunakan teknik transformasi Fourier ini permasalahannya terutama pada :

1. Mendapatkan *watermark* yang *invariant* (independen) terhadap rotasi, skala, maupun translasi.
2. Memasang *watermark* pada citra asli.
3. Mengekstraksi *watermark* dari citra.
4. Perubahan koordinat pada LPM, Interpolasi digunakan dalam permasalahan ini.
5. Penggunaan *Phase Correlation* dalam menyesuaikan posisi *watermark*

1.3 BATASAN MASALAH

Tugas Akhir ini mempunyai beberapa batasan masalah, yaitu sejauh mana program *watermarking* itu dibuat :

1. Masukkan berupa 1 citra. (2 citra untuk ekstraksi)
2. Program dapat memasang maupun mengenali *watermark* dalam citra.
3. Interpolasi yang digunakan interpolasi *bilinear.*
4. *Watermark* berbentuk spread spektrum dua dimensi.
5. Menggunakan kode Reed Solomon untuk kontrol error

1.4 TUJUAN

Tujuan dari tugas akhir ini adalah untuk membangun program yang dapat melakukan *embedding watermarking* pada citra dan ekstraksi *watermarking* yang ditambahkan tersebut.

1.5 METODOLOGI

Pembuatan tugas akhir ini dilakukan dengan mengikuti metodologi sebagai berikut :

- Studi Literatur

  Pada tahap ini dipelajari konsep-konsep transformasi Fourier, *Log-polar mapping*, *phase correlation*, spreaid *spectrum* dan *reed solomon error control*. Konsep-konsep ini didapat baik dari buku-buku yang relevan maupun dari paper-paper dan artikel-artikel dari internet. Dalam tahap ini juga dipelajari pemrograman dengan Matlab yang akan dipakai dalam pembuatan program.

- Perancangan program *watermarking*

  Pada tahap ini dilakukan perancangan dari program *watermarking* dengan merancang tahap-tahap yang harus dilalui dan menentukan batasan-batasan dari program.

- Pembuatan program *watermarking*

  Pada tahap ini dilakukan implementasi dari rancangan program *watermark*

- Uji coba dan evaluasi

  Pada tahap ini dilakukan uji coba program dengan menggunakan citra-citra yang tidak mengalami serangan dan dengan citra-citra yang mengalami serangan rotasi, skala dan translasi

- Penyusunan buku tugas akhir

  Pada tahap ini dilakukan penyusunan dokumentasi terhadap dasar teori dan metode yang digunakan, serta hasil-hasil uji coba dari program.

1.6 SISTEMATIKA PEMBAHASAN

Pembahasan yang akan disajikan dalam tugas akhir ini, akan dibagi dalam beberapa bab sebagai berikut :

BAB I **PENDAHULUAN,** menjelaskan tentang hal-hal yang mendasar serta memotivasi perancangan dan pembuatan perangkat lunak,

meliputi latar belakang, permasalahan, tujuan, metodologi penelitian, dan sistematika pembahasan.

BAB II **DASAR TEORI,** dibahas dasar – dasar ilmu yang menunjang pembahasan tugas akhir, dan konsep mengenai *watermarking* sebagai proteksi hak cipta.

BAB III **DESAIN PERANGKAT LUNAK,** mengulas secara lengkap program *watermarking* yang digunakan dalam tugas akhir.

BAB IV **IMPLEMENTASI PERANGKAT LUNAK,** mengulas perancangan dan pembuatan skema untuk program *watermarking* yang dikerjakan selama tugas akhir.

BAB V **UJI COBA DAN EVALUASI,** membahas hasil uji coba program *watermarking* yang sudah dikerjakan, dan menganalisa ketahanan *watermarking* terhadap berbagai macam serangan.

BAB VI **KESIMPULAN DAN SARAN,** berisi kesimpulan – kesimpulan dari bab – bab sebelumnya serta saran – saran yang berkaitan dengan arah pengembangan sistem selanjutnya.

BAB II

DASAR TEORI

BAB II
DASAR TEORI

2.1.TERMINOLOGI

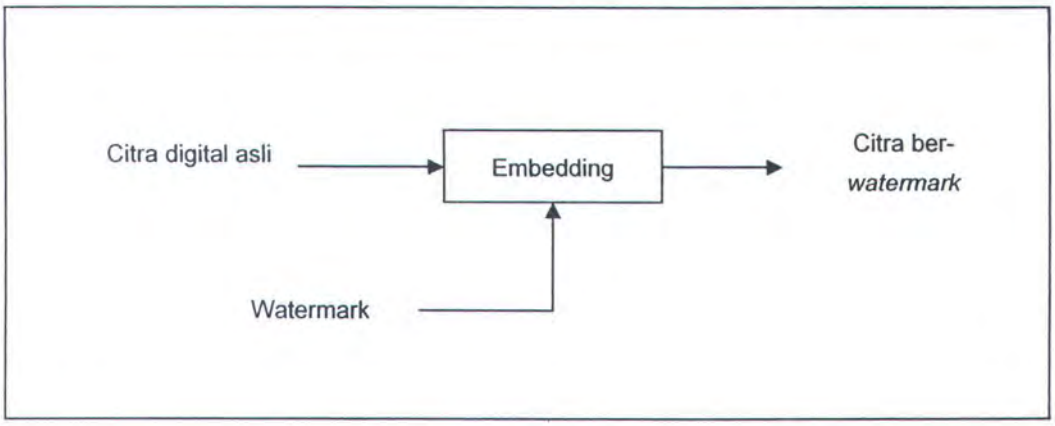
Watermarking adalah proses *embedding* data yang disebut *watermark* atau tanda tangan digital kedalam multimedia sedemikian hingga *watermark* dapat dideteksi dan diekstraksi di lain waktu untuk mendapatkan kepastian dari suatu obyek [8]. Obyek tersebut bisa berupa citra atau video atau audio. Contoh sederhana dari *watermark* adalah sebuah segel yang terlihat pada citra untuk menunjukkan identitas dari pemilik hak cipta. Namun *watermark* juga dapat berisi informasi tambahan, termasuk identitas pembeli dari sebuah kopi hak cipta.

Secara umum *watermarking* dibagi menjadi tiga bagian :

1. *Watermark*
2. Proses *Embedding*
3. Proses Ekstraksi atau Korelasi

Setiap pemilik memiliki *watermark* yang unik atau pemilik tersebut dapat memberikan *watermark* yang berbeda untuk obyek-obyek yang berbeda menggunakan algoritma *embedding*. Algoritma ekstraksi melakukan verikasi terhadap kepemilikan dan keabsahan suatu obyek.

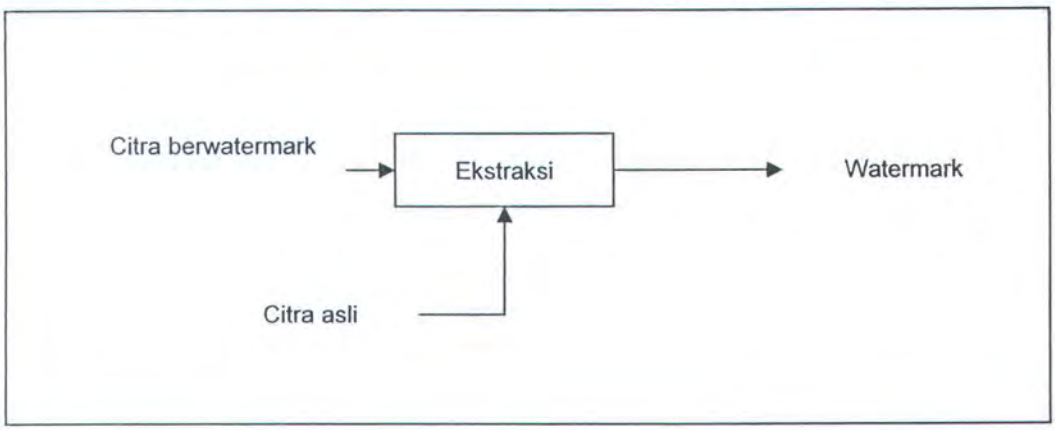
Proses *embedding* secara umum direpresentasikan oleh gambar dibawah:



Gambar 2.1 Proses umum *embedding watermark*

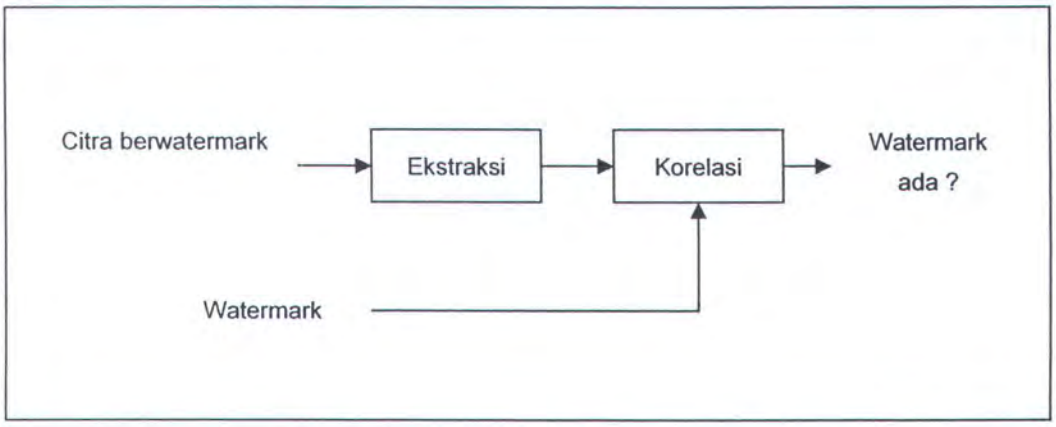
Data *watermark*, bisa dalam bentuk *spread spektrum*, citra, atau barisan bit, ditambahkan dalam citra digital asli dengan metode *embedding*, Hasil dari penambahan tersebut adalah citra ber-*watermark*.

Sedangkan proses ekstraksi secara umum digambarkan sebagai berikut :



Gambar 2.2 Ekstraksi *watermark* menggunakan citra asli

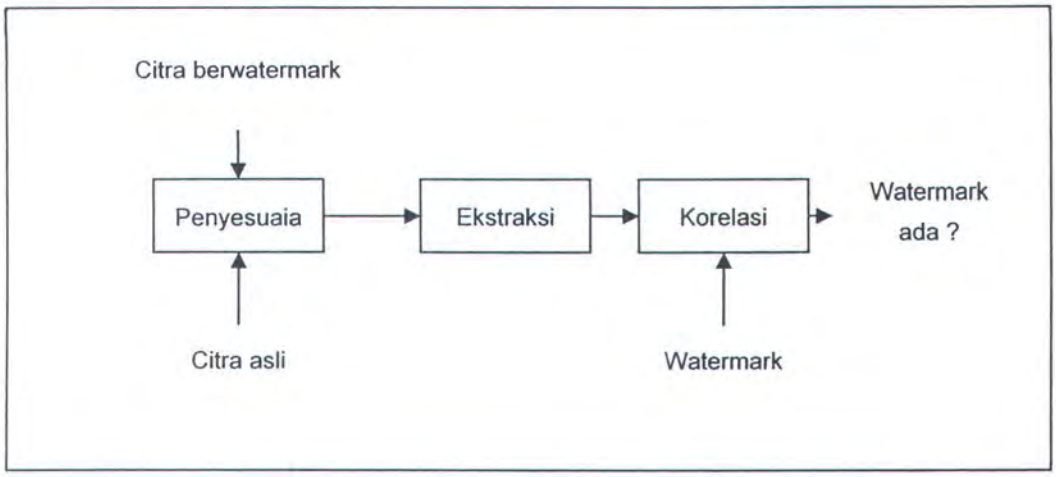
Ekstraksi dilakukan dengan membandingkan citra asli dan citra ber-*watermark* untuk mendapatkan *watermark* yang terkandung didalamnya. Pada umumnya proses ekstraksi membutuhkan gambar asli untuk menghasilkan *watermark* yang terkandung didalamnya, namun ada pula proses ekstraksi yang tidak menggunakan citra asli tapi dengan data *watermark* yang ingin dicari, hal ini digambarkan sebagaimana berikut :



Gambar 2.3 Ekstraksi *watermark* tanpa citra asli

Metode yang tidak menggunakan citra asli biasanya menggunakan korelasi untuk mencari *watermark* di dalam citra ber-*watermark*. Hasil dari proses ini

adalah keterangan mengenai ada tidaknya *watermark* yang dicari dalam citra. Dalam tugas akhir ini metode *watermarking* yang digunakan membutuhkan citra asli dan *watermark* asli, yang digambarkan sebagaimana berikut :



Gambar 2.4 Ekstraksi *watermark* dengan citra asli dan *watermark*

Metode diatas menggunakan citra asli dalam menyesuaikan (*Rectification*) citra ber-*watermark*. Menyesuaikan dalam hal ini adalah membalikkan hasil serangan terhadap citra ber-*watermark* agar kembali ke bentuk semula sama seperti citra asli, citra yang tidak mendapatkan serangan apapun.

2.2.APLIKASI *WATERMARK*

Aplikasi *watermark* dalam dunia nyata membutuhkan tingkat *robustness watermark* yang berbeda [9], aplikasi itu antara lain :

1. *Watermarking* untuk proteksi hak cipta

   Perlindungan terhadap hak cipta mungkin merupakan aplikasi utama untuk *watermark* saat ini. Tujuannya adalah untuk menambahkan informasi tentang asal, umumnya pemilik hak cipta, dari data untuk menghalangi pihak lain mengklaim hak cipta suatu data. Jadi *watermark* digunakan untuk membuktikan kepemilikan atas suatu hak cipta, sehingga membutuhkan tingkat *robustness* yang sangat tinggi. Yang menjadi motor utama dari aplikasi ini adalah adanya Web yang berisi jutaan citra yang tersedia yang ingin dilindungi oleh pemiliknya.
2. Sidik jari untuk penelusuran penghianat

Ada aplikasi lain dimana tujuannya adalah untuk membawa informasi tentang penerima yang legal daripada sumber dari data, utamanya untuk mengidentifikasikan satu kopi dari sebuah data. Hal ini berguna untuk melakukan monitor atau penelusuran kembali sebuah kopi ilegal suatu data. Dan ini mirip dengan sistem *serial number* dari suatu *software*. Aplikasi jenis ini biasanya disebut "*fingerprinting*" dan melibatkan *embedding watermark* yang berbeda-beda untuk masing-masing kopi yang berbeda. Karena didistribusi secara individual maka *watermark* harus *robust* terhadap serangan jenis *collusion*. Juga untuk beberapa aplikasi *fingerprinting* diperlukan *watermark* dengan tingkat kompleksitas yang rendah dan mudah diekstraksi. Aplikasi ini membutuhkan tingkat *robustness* yang tinggi terhadap pemrosesan data standar dan juga serangan dari luar.

3. *Watermarking* untuk perlindungan terhadap penyalinan (*copy protection*)

   Sifat yang diinginkan dari distribusi multimedia adalah mekanisme perlindungan terhadap penyalinan yang tidak berizin. *Copy protection* ini sukar diaplikasikan pada sistem yang terbuka, namun untuk sistem yang tertutup hal ini dapat dilakukan. *Watermark* digunakan untuk menandakan status *copy* dari data. Misalnya untuk sebuah sistem DVD dimana data *copy protection* ditambahkan dengan *watermarking*. DVD player yang mengaplikasikan tidak akan memutar DVD yang memiliki *watermark* "*copy never*". Data yang membawa *watermark* "*copy once*" dapat dikopi satu kali, tetapi tidak dapat dikopi lebih dari satu kali.

4. *Watermarking* untuk autentifikasi citra

   Tujuan dari aplikasi ini untuk mendeteksi adanya modifikasi terhadap suatu data. Aplikasi ini membutuhkan *watermark* yang memiliki tingkat *robustness* yang rendah, bahkan aplikasi ini membutuhkan tingkat *robustness* terendah. Namun aplikasi ini membutuhkan tingkat *robustness* yang lebih tinggi jika identifikasi daerah yang dimodifikasi adalah hal utama yang dicari.

frekuensi ini dieksploitasi untuk mendapatkan data enkoding dengan *bitrate* yang rendah. Adalah jelas pada kedua sistem, baik pendengaran maupun penglihatan, memiliki resolusi yang lebih baik di daerah spektral yang memiliki energi tinggi dan frekuensi rendah. Lebih lagi, analisa spektrum dari citra menunjukkan bahwa sebagian besar dari data semacam itu terletak pada daerah berfrekuensi rendah.

Ada 2 teknik dasar *Spread Spectrum*: *Direct Sequencing* (*DS*) dan *Frequency Hopping* (*FH*). Juga metode hibrida lainnya. Dengan *DS*, sinyal asli dikalikan oleh sinyal yang diketahui memiliki *bandwidth* yang lebih besar. Metode ini yang digunakan dalam pembentukan *watermark* di tugas akhir ini Dengan *FH*, pusat frekuensi dari sinyal yang ditransmisikan berbeda-beda dalam pola *pseudorandom*.

2.4.*DIRECT SEQUENCES SPREAD SPECTRUM* (DSSS) *SYSTEM*

Dalam *Direct Sequences* (DS), bentuk sinyal adalah barisan dari pulsa persegi yang lebih kecil disebut *chip*. *Chip* adalah barisan *pseudorandom* dari +1 dan -1 yang dikenal oleh penerima dan sering memiliki periode pengulangan yang sama dengan periode simbol.

Dalam *spreading*, data biner d_t dikalikan secara langsung dengan barisan PN pn_t, yang merupakan data biner yang independen, untuk menghasilkan sinyal *baseband* yang ditransmisi [3] :

$$tx_b = d_t . pn_t \quad \text{..... (2.1)}$$

Efek dari pengalian tersebut adalah tersebarnya *bandwidth baseband* dari d_t ke dalam *bandwidth baseband* R_c.

Dalam *Despreading*(pengumpulan), Sinyal *spread spectrum* tidak dapat dideteksi dengan penerima *narrowband* konvensional. Dalam penerima, sinyal *baseband* yang diterima rx_b dikalikan dengan barisan PN pn_r.

Jika $pn_r = pn_t$ dan disinkronkan dengan barisan PN dalam data yang diterima, maka data biner yang didapat dihasilkan dalam d_r. Efek dari pengalian dari sinyal *spread spectrum* rx_b dengan barisan *PN* pn_t yang digunakan oleh transmiter adalah untuk men-*despread bandwidth* rx_b ke R_s.

Jika $pn_r \neq pn_t$ maka tidak ada *despreading*. Sinyal d_r memiliki *spread spectrum*, penerima yang tidak mengetahui barisan *PN* dari transmiter tidak dapat mereproduksi data yang dikirim.

Dalam implementasinya perkalian tersebut bisa dilakukan dengan melakukan operasi biner XOR.

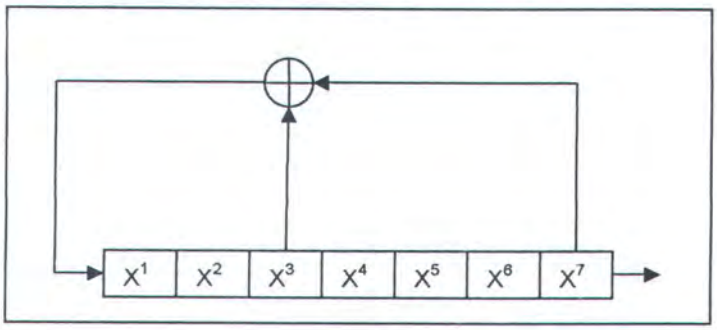
2.4.1 *Pseudorandom Noise*

Kode Barisan PN adalah sebuah *Pseudonoise* atau barisan *Pseudorandom* yang terdiri atas 1 dan 0, tetapi bukan barisan random sebenarnya (karena periodik) sedangkan sinyal random tidak dapat diperkirakan [3]. *Autocorrelation* dari kode PN memiliki sifat yang mirip dengan *noise* putih (*white-noise*):

1. Tidak random, namun terlihat random bagi user yang tidak mengetahui kodenya.
2. Dapat ditentukan, sinyal periodik diketahui oleh penerima dan pengirim, semakin panjang periode dari kode *spreading* PN makin dekat sinyal yang dikirim mendekati gelombang biner random sebenarnya, dan makin silit dideteksi
3. memiliki sifat statistik dari *white-noise*.

2.4.2 *Pseudorandom Noise* dengan *M-Sequences*

Secara ideal diinginkan barisan yang ortogonal pada semua perpindahan (*shift*), namun hanya aproksimasi yang mungkin dilakukan.



Gambar 2.5 Generator *Shift Register* untuk *M-Sequence*

Secara teori barisan yang dipilih secara random memiliki sifat *Autocorrelation* yang secara umum untuk aplikasi komunikasi juga diperlukan barisan yang bisa digenerate secara mudah oleh penerima maupun pengirim. Salah satunya *M-Sequences*. Barisan ini bisa dihasilkan dengan menggunakan *shift register* biner dengan *feedback*. Seperti gambar 2 diatas dimana menunjukkan penambahan modulo 2 (XOR). Jika memori dari *shifting* sebesar *m*, maka hanya ada 2^m kondisi yang mungkin dan *register* masukkan dari *shift register* adalah penjumlahan modulo 2 (XOR) dari *register* output dan macam-macam *register* internal. Jika *feedback* dipilih dengan benar maka *shift register* akan berputar ke seluruh state, kecuali 0, sebelum mengulangi sebuah state. Dengan kata lain, proses akan menghasilkan barisan dengan panjang 2^m-1 sebelum mengulangi lagi (*shift Register* menghasilkan barisan 0 dan 1, dan barisan ini dipetakan ke barisan -1 dan 1 ketika digunakan sebagai kode *spreading*). Untuk memori *m*, tidak semua *feedback* akan menghasilkan *M-Sequence*, *feedback* tersebut bisa didapat dengan teori bidang Galois. Ada korespondensi *shift register* panjang maksimum dan polinominal tak terfaktorkan dari Galois. Contoh polinominal x^2+1 bisa difaktorkan sebagai $(x+1)^2$ namun x^2+x+1 tidak dapat difaktorkan, yang tidak dapat difaktorkan itulah yang bisa digunakan untuk menghasilkan *M-Sequence*, sedangkan yang dapat tidak bisa digunakan.

Polinom tak tereduksi dengan koefisien biner :

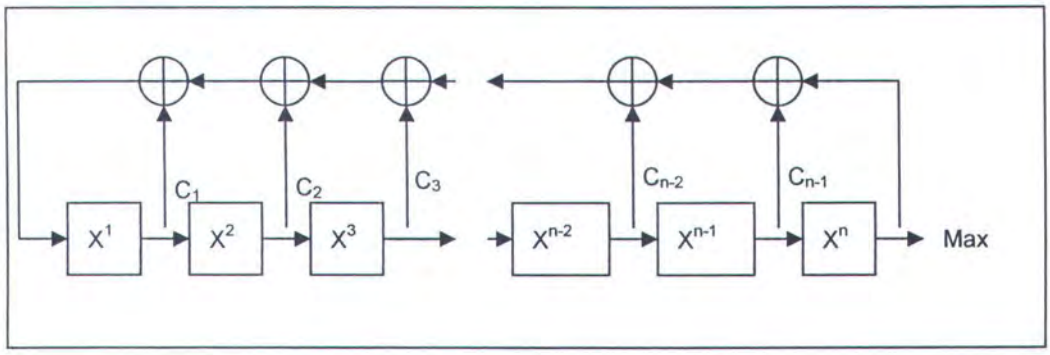
Tabel 2.1 Polinom untuk *M-Sequences*

Order	Polinominal tak tereduksi	Posisi Feedback
2	x^2+x+1	[2,1]
3	x^3+x+1, x^3+x^2+1	[3,1]
4	x^4+x+1	[4,1]
5	x^5+x^2+1, $x^5+x^4+x^3+x^2+1$	[5,3], [5,4,3,2]
6	x^6+x+1, $x^6+x^5+x^2+x+1$	[6,1], [6,5,2,1]
7	x^7+x^3+1, $x^7+x^5+x^4+x^3+x^2+x+1$	[7,3], [7,5,4,3,2,1]
8	$x^8+x^4+x^3+x^2+1$, $x^8+x^6+x^5+x^3+1$	[8,4,3,2], [8,6,5,3]
9	x^9+x^4+1, $x^9+x^6+x^4+x^3+1$	[9,4], [9,6,4,3]
10	$x^{10}+x^3+1$, $x^{10}+x^7+x^3+x^2+1$	[10,3], [10,7,3,2]
11	$x^{11}+x^2+1$,	[11,2],
12	$x^{11}+x^{10}+x^9+x^7+x^6+x^5+x^3+x^2+x+1$	[11,10,9,7,6,5,3,2,1]
	$x^{12}+x^6+x^4+x+1$,	[12,6,4,1],
	$x^{12}+x^{11}+x^9+x^8+x^7+x^5+x^2+x+1$	[12,11,9,8,7,5,2,1]

Contoh : polinom :

$$x^m + c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \ldots + c_1 x + 1 = 0$$

bisa dibentuk *shift register* :



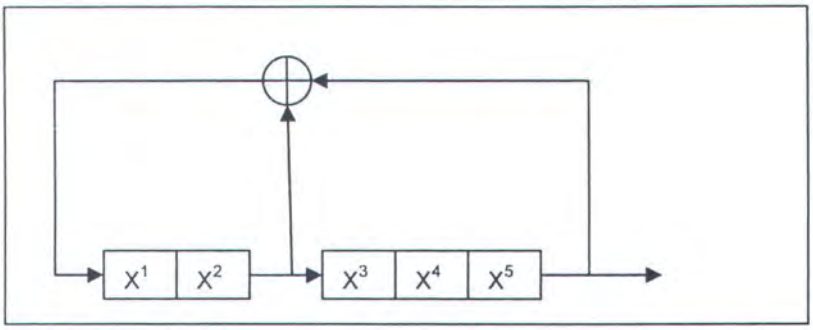
Gambar 2.6 *Shift register* untuk polinom

$$x^m + c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \ldots + c_1 x + 1 = 0$$

Contoh lain polinom :

$x^5 + x^2 + 1 = 0$

bisa dibentuk *shift register* :



Gambar 2.7 *Shift register* untuk polinom $x^5 + x^2 + 1 = 0$

Dan menghasilkan barisan :

-1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 1 1 -1 1 -1 1 -1 1

2.5 *ERROR CONTROL CODE REED SOLOMON*

Peningkatkan *robustness* dari *watermark* dengan menggunakan *error control code* diperlukan dalam penggunaan citra sebagai *carrier* sinyal. Hal ini disebabkan dalam proses ekstraksi *watermark* akan selalu terjadi pengurangan kualitas *watermark*. Pengurangan ini dapat menyebabkan tidak akuratnya dekoding *watermark*. Dalam tugas akhir ini digunakan kode Reed Solomon sebagai *error control code*.

2.5.1 Definisi Kode Reed Solomon

Sebuah kode Reed Solomon *(RS Codes)* yang meliputi $GF(p^m)$ adalah sebuah kode BCH dengan panjang $n = p^m-1$, berdimensi $k = n-d+1$, dan jarak minimum d. Oleh Karena itu ideal dalam $GF(p^m)[x]/(x^{p^m-1} - 1)$ [4]

Generator Polinominalnya :

$$g(x) = (x - \alpha^a)(x - \alpha^{a+1})\ldots(x - \alpha^{a+d-2}) \quad \ldots\ldots (2.2)$$

dimana α adalah elemen primitif dalam $GF(p^m)$ dan a adalah sebuah integer.

RS Codes dikatakan sebagai *maximum distance separable (MDS) codes*. Sebuah istilah yang diperkenalkan oleh Singleton, karena jarak minimum dari *RS Codes* adalah sebesar mungkin untuk *n* dan *k* yang tetap.

2.5.2 *Encoding* Kode Reed Solomon

Karena *RS Codes* bersifat siklik, mereka dapat di-*encode* oleh produk dari *g(x)* dan polinominal yang berasosiasi dengan vektor informasi, atau sistem *encoding*. Tetapi, *encoding* asli yang dibuat oleh Reed dan Solomon dilakukan dengan cara yang lain.

Misalkan $r = (r_0, r_1, \ldots, r_{k-1}), r_i \in GF(p^m)$, adalah vektor informasi. Dan $r(x) = \sum_{i=0}^{k-1} r_i x^i$, adalah representasi polinominal, maka *codeword*-nya adalah :

$$v = (r(1), r(\alpha).r(\alpha^2), \ldots, r(\alpha^{n-1})) \quad \text{..... (2.3)}$$

Dengan menggunakan polinominal Mattson dan Solomon dibuktikan bahwa $\alpha, \alpha^2, \ldots, \alpha^{d-1}$ adalah akar dari *v(x)* dan *v* adalah *codeword*. Perlu dicatat bahwa sistem *encoding* ini tidak sistematik.

2.5.3 Matriks *Parity Check*

Bentuknya :

$$\begin{pmatrix} 1 & \alpha^a & (\alpha^a)^2 & \cdots & (\alpha^a)^{n-1} \\ 1 & \alpha^{a+1} & (\alpha^{a+1})^2 & \cdots & (\alpha^{a+1})^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{a+d-2} & (\alpha^{a+d-2})^2 & \cdots & (\alpha^{a+d-2})^{n-1} \end{pmatrix} \quad \text{..... (2.4)}$$

Contoh 1 :

Dalam *GF(7), α = 3* adalah elemen primitif, jadi kode RS dengan parameter n = 6, d = 4, k = 3 ditentukan oleh α = 3 dan a = 1, memiliki generator polinominal :

$$g(x) = (x-3)(x-2)(x-6) = x^3 + 3x^2 + x + 6$$

Matriks Generatornya :

$$G = \begin{pmatrix} 6 & 1 & 3 & 1 & 0 & 0 \\ 0 & 6 & 1 & 3 & 1 & 0 \\ 0 & 0 & 6 & 1 & 3 & 1 \end{pmatrix}$$

dan matriks *parity check* :

$$H = \begin{pmatrix} 1 & 3 & 2 & 6 & 4 & 5 \\ 1 & 2 & 4 & 1 & 2 & 4 \\ 1 & 6 & 1 & 6 & 1 & 6 \end{pmatrix}$$

Contoh 2 :

Dalam $GF(2^3)$, bisa diambil akar α dari polinominal tak tereduksi $x^3 + x + 1$ sebagai elemen primitif. Kode RS dengan panjang n = 7, jarak minimum d = 3 dan dimensi k = 5 memiliki generator polinominal $g(x) = (x - \alpha^3)(x - \alpha^4)$. Jika a = 3, maka :

Matriks Generatornya :

$$G = \begin{pmatrix} 1 & \alpha^6 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \alpha^6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \alpha^6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \alpha^6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & \alpha^6 & 1 \end{pmatrix}$$

dan Matriks *Parity Check*-nya :

$$H = \begin{pmatrix} 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^2 \\ 1 & \alpha^4 & \alpha & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix}$$

2.6 TRANSFORMASI FOURIER

Transformasi Fourier dalam dunia *signal processing* merupakan sebuah transformasi dari domain waktu ke domain frekuensi. Dalam *watermarking* ini transformasi Fourier ini memegang peranan penting, karena proses utama dalam *embedding* dan ekstraksi *watermark* terjadi dalam *magnitude* dari Transformasi Fourier ini. Persamaan transformasi Fourier untuk satu dimensi adalah sebagai berikut :

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx \quad \text{..... (2.5)}$$

di mana j = √-1. Sebaliknya *inverse* Fourier bisa didapatkan dengan mencari harga *f(x)* pada persamaan 2.5 menjadi :

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}\,du \quad \text{..... (2.4)}$$

Sedangkan jika ditulis dalam bentuk diskrit maka persamaan menjadi :

$$F(u)=\frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/M} \quad \text{..... (2.5)}$$

dimana M adalah jumlah elemen dari *f(x)*.

Sedangkan harga fungsi asli *f(x)* pada persamaan 2.5 (atau juga *inverse* Fourier) adalah :

$$f(x)=\sum_{u=0}^{M-1} F(u)e^{j2\pi ux/M} \quad \text{.....(2.6)}$$

dimana M adalah jumlah elemen dari *F(u)*.

Untuk transformasi berdimensi banyak digunakan pemikiran bahwa transformasi dihitung dengan menghitung transformasi untuk masing-masing dimensi.

2.6.1 *Discrete Fourier Transform (DFT)* dan *Inverse Discrete Fourier Transform (IDFT)* Dua Dimensi

Dalam *watermarking* ini digunakan transformasi dalam bentuk *Discrete Fourier Transform* (*DFT*) dan *Inverse Discrete Fourier Transform* (*IDFT*) dua dimensi. DFT dan IDFT dari citra didefinisikan sebagaimana berikut [1]:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \quad \text{..... (2.7)}$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \quad \text{..... (2.8)}$$

Hal ini selalu dilakukan dalam *watermarking* ini. Jika contoh diatas digunakan sebagai contoh IDFT, maka matriks dftshift perlu digeser kembali seperti semula menjadi matriks dft, dan hasil dari IDFT akan menjadi matriks tes.

Matriks dftshift jika dihitung *magnitude*-nya, maka akan menghasilkan :

$$mag = \begin{pmatrix} 5 & 12.207 & 3 & 12.207 \\ 6.7082 & 12.042 & 14.318 & 1 \\ 1 & 6.7082 & 73 & 6.7082 \\ 6.7082 & 1 & 14.318 & 12.042 \end{pmatrix}$$

dan *phase* yang dihasilkan :

$$phase = \begin{pmatrix} 3.1416 & -2.1815 & 0 & 2.1815 \\ -2.6779 & -3.0585 & 2.0032 & -1.5708 \\ 0 & -1.1071 & 0 & 1.1071 \\ 2.6779 & 1.5708 & -2.0032 & 3.0585 \end{pmatrix}$$

2.6.2 Simetrisitas DFT

Hasil perhitungan *magnitude* DFT selalu memiliki bentuk yang simetris. Hal ini bisa dilihat dari percobaan dengan menggunakan matlab sebagaimana berikut :

$$\text{tes} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$$

Dari matriks tes diambil DFT-nya dengan fungsi *fft2* dari matlab kemudian diambil *magnitude*-nya dengan fungsi *abs* dari matlab setelah komponen berfrekuensi nol digeser ke tengah dengan fungsi *fftshift* dari matlab,

tes2 = abs(fftshift(fft2(tes)))

$$tes2 = \begin{pmatrix} 0 & 0 & 65.7164 & 0 & 0 \\ 0 & 0 & 106.3314 & 0 & 0 \\ 13.1433 & 21.2663 & 325.0000 & 21.2663 & 13.1433 \\ 0 & 0 & 106.3314 & 0 & 0 \\ 0 & 0 & 65.7164 & 0 & 0 \end{pmatrix}$$

dari *magnitude* diatas terlihat *magnitude* memiliki bentuk yang simetris.

2.7 *LOG-POLAR MAPPING*

Log-Polar Mapping (LPM) adalah pemetaan dari sistem koordinat kartesian ke sistem koordinat *log-polar*. Sistem koordinat log-polar ini direpresentasikan oleh jari-jari r dan sudut θ. Persamaan dari pemetaan ini adalah :

$$x = e^{\rho} * \cos\theta \quad \text{.....(2.11)}$$

$$y = e^{\rho} * \sin\theta \quad \text{.....(2.12)}$$

$$r = \ln\rho \quad \text{.....(2.13)}$$

di mana $\rho \in R$ dan $0 \leq \theta \leq 2\pi$, dengan batasan-batasan :

$$r_{\min} = 1$$

$$r_{\max} = \sqrt{x_c^{\,2} + y_c^{\,2}} \quad \text{.....(2.14)}$$

$$\theta_{\min} = 0$$

$$\theta_{\max} = 2 * \pi * ((numa - 1)/numa) \quad \text{.....(2.15)}$$

dimana : numa = resolusi sudut LPM

r = jari-jari sampling

θ = sudut sampling

LPM merupakan penyebab utama terjadinya berkurangnya kualitas citra. Untuk mengurangi efek dari LPM ini digunakan interpolasi *bilinear* dalam pengambilan titik atau *pixel* di domain kartesian. Interpolasi *bilinear* mendapatkan nilai *pixel* baru berdasarkan pada rata-rata yang diberatkan (*weighted average*) dari 4 *pixel* yang terdekat 2x-2 tetangga dari *pixel* di citra asli. Persamaan dari interpolasi *bilinear* ini ditulis sebagaimana berikut :

$$\begin{aligned} P(\rho,\theta) = {} & C(x,y).(1-a).(1-b) + C(x,y+1).(1-a).b \\ & + C(x+1,y).a.(1-b) + C(x+1,y+1).a.b \end{aligned} \quad \text{..... (2.16)}$$

dimana $C(x,y), C(x,y+1), C(x+1,y)$, dan $C(x+1,y+1)$ merupakan empat titik dalam domain kartesian sedangkan $P(\rho,\theta)$ merupakan sebuah titik yang berada dalam kotak yang dibentuk oleh empat titik tersebut.

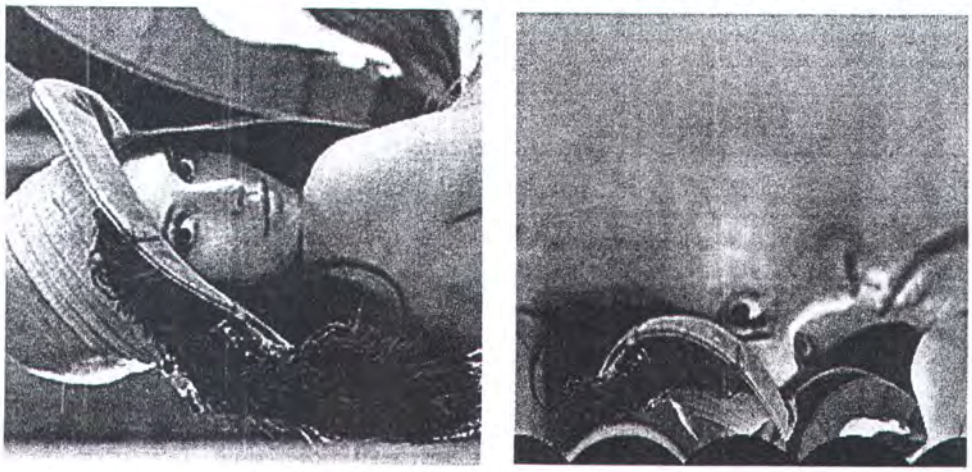
Domain LPM memiliki sifat-sifat yang menjadi penyebab utama independennya *watermarking* terhadap rotasi dan skala. Hal ini terjadi karena

dalam domain LPM rotasi dan skala hanya akan menyebabkan pergeseran yang dapat dengan mudah diatasi, tidak seperti halnya dengan rotasi dan skala dalam domain kartesian. Untuk memperlihatkan pergeseran itu, digunakan contoh dengan citra lena_org.bmp berikut :

(a) (b)

Gambar 2.8 (a) citra lena_org.bmp (b) LPM dari lena_org.bmp

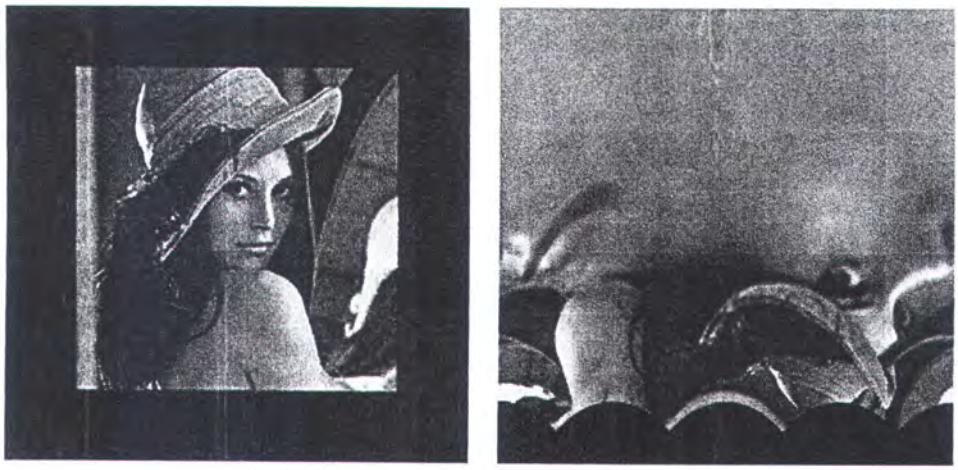
Apabila citra lena_org.bmp itu dirotasi 90 derajat berlawanan arah jarum jam :



(a) (b)

Gambar 2.9 (a) citra lena_org.bmp yang dirotasi (b) LPM dari lena_org.bmp yang dirotasi

Dari hasil LPM gambar 2.8 dan gambar 2.9 terlihat adanya pergeseran kekiri. Hal ini menunjukkan rotasi hanya akan menimbulkan pergeseran di domain LPM. Sedangkan untuk skala, digunakan citra yang mengalami skala 0,7 dan diberi nilai nol agar ukurannya tetap (*zeropad*):



(a) (b)

Gambar 2.10 (a) citra lena_org.bmp yang diskala (b) LPM dari lena_org.bmp yang diskala

Dari hasil LPM gambar 2.8 dan gambar 2.10 terlihat adanya pergeseran keatas. Hal ini menunjukkan skala juga hanya akan menimbulkan pergeseran di domain LPM.

2.8 SIFAT INDEPENDEN TERHADAP ROTASI, SKALA DAN TRANSLASI

Relasi dari citra asli dengan citra yang mengalami serangan RST dapat ditulis sebagaimana berikut :

$$i_1(x,y) = i_0(\sigma(x\cos\alpha + y\sin\alpha) - x_0, \sigma(-x\sin\alpha + y\cos\alpha) - y_0) \quad \text{.....(2.17)}$$

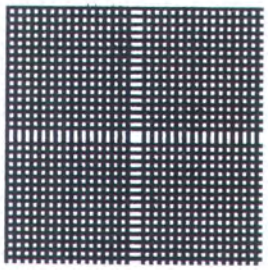
kemudian *magnitude* :

$$|I_1(u,v)| = |\sigma|^{-2} |I_0(\sigma^{-1}(u\cos\alpha + v\sin\alpha), \sigma^{-1}(-u\sin\alpha + y\cos\alpha))| \quad \text{.....(2.18)}$$

Persamaan diatas independen terhadap translasi. Hal ini dibuktikan dengan sebuah citra sederhana seperti di bawah ini :

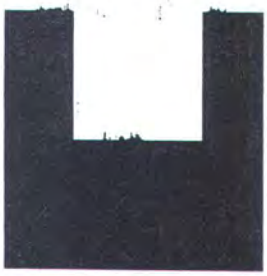
Gambar 2.11 citra sederhana asli

Apabila dilakukan DFT dan diambil *magnitude* dari DFT tersebut akan menghasilkan gambar sebagaimana berikut :



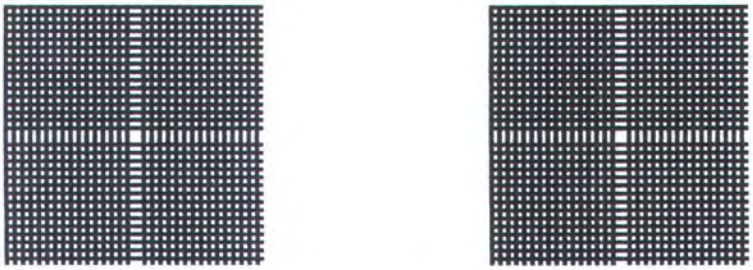
Gambar 2.12 *magnitude* sebuah citra sederhana

Lalu citra asli digeser secara horizontal sebanyak 16 *pixel* menjadi :



Gambar 2.13 citra sederhana yang telah digeser

Jika diambil *magnitude* dari citra yang telah digeser,



Gambar 2.14 *magnitude* citra sederhana asli (kiri) dan yang telah mengalami pergeseran (kanan)

Dari gambar 2.14 bisa terlihat bahwa kedua *magnitude* adalah sama, hal ini diperkuat dengan melakukan korelasi pada kedua *magnitude* yang menghasilkan nilai 1 yang berarti kedua *magnitude* adalah sama.

Dalam sistem koordinat *log-polar*, maka *magnitude* ditulis sebagai :

$$\left|I_1(u,v)\right| = \left|\sigma\right|^{-2}\left|I_0(\sigma^{-1}e^{\rho}\cos(\theta-\alpha)),\sigma^{-1}e^{\rho}\sin(\theta-\alpha))\right| \quad \text{..... (2.19)}$$

atau

$$\left|I_1(\rho,\theta)\right| = \left|\sigma\right|^{-2}\left|I_0(\rho-\log\sigma,\theta-\alpha)\right| \quad \text{.....(2.20)}$$

Persamaan (2.20) menunjukkan *amplitudo* dari spektrum *log-polar* diskala sebesar $\left|\sigma^{-2}\right|$, sehingga skala pada citra menghasilkan pergeseran translasi sebesar $\log\sigma$ sepanjang sumbu radius log-polar ρ. Rotasi dari citra menghasilkan pergeseran siklik sebesar α sepanjang sumbu sudut θ.

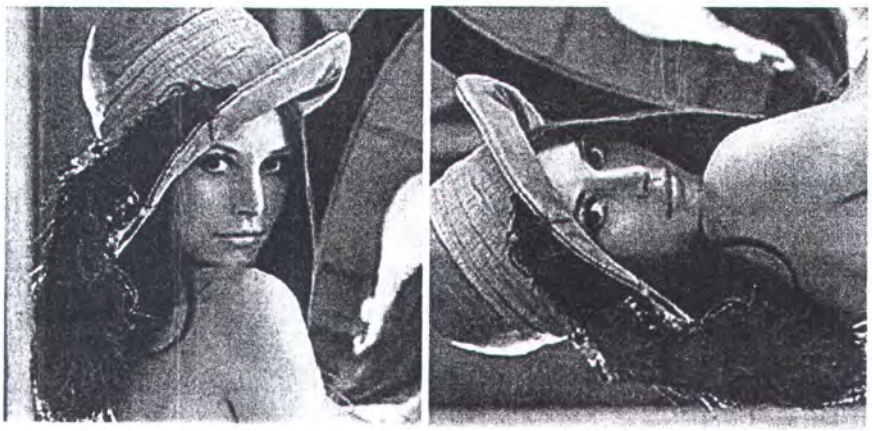
Menurut sifat translasi dari transformasi Fourier, transformasi Fourier I_1 dan I_0 direlasikan menjadi :

$$F_1(\omega_\rho,\omega_\theta) = \left|\sigma\right|^{-2} e^{-j(\omega_\rho.\log\sigma+\omega_\theta.\alpha)} F_0(\omega_\rho,\omega_\theta) \ldots. \qquad (2.21)$$

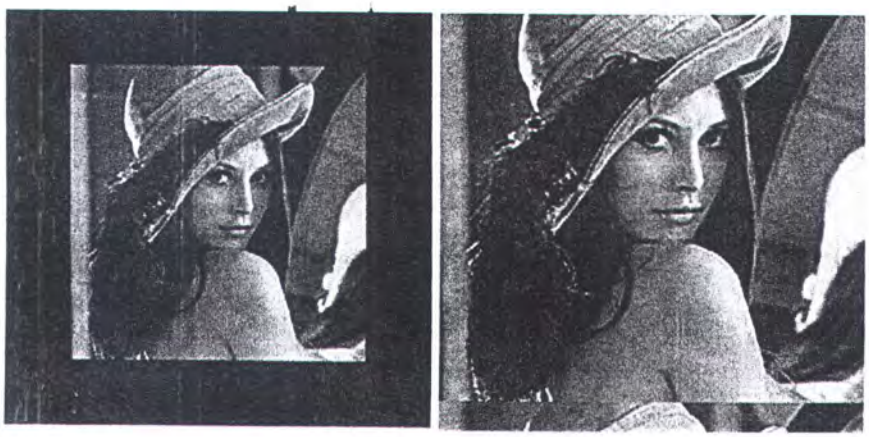
magnitude Fourier dari dua *LPM* direlasikan :

$$\left|F_1(\omega_\rho,\omega_\theta)\right| = \left|\sigma\right|^{-2}\left|F_0(\omega_\rho,\omega_\theta)\right| \quad \text{.....} \qquad (2.22)$$

dimana F_1 dan F_0 merupakan DFT dari I_1 dan I_0 .Perbedaan *Phase* antara kedua pemetaan LPM berelasi secara langsung dengan *displacement* (pergeseran) mereka, yang diberikan oleh: $e^{j(\omega_\rho . \log \sigma + \omega_\theta . \alpha)}$. Persamaan 2.22 diatas setara dengan perhitungan Fourier-Mellin. Karena penggunaan korelasi yang dinormalkan maka persamaan ini menjadi independen terhadap rotasi, skala dan translasi. Hal ini dibuktikan dengan menggunakan citra lena_org.bmp, citra lena_org.bmp yang mengalami rotasi 90 derajat berlawanan arah jarum jam, lena_org.bmp yang mengalami rotasi skala 0,7 kemudian di-*zeropad.* dan citra yang mengalami translasi keatas 20 *pixel.* pembuktian tersebut :



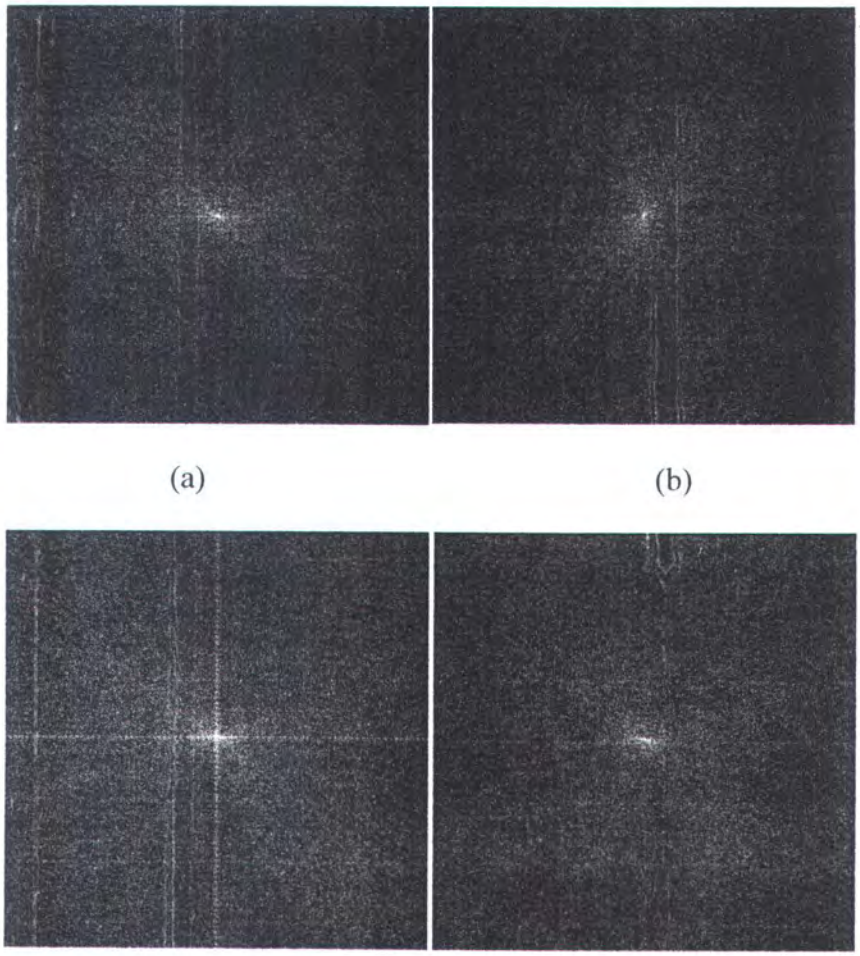
(a) (b)



(c) (d)

Gambar 2.15 (a) citra lena_org.bmp, (b) citra lena_org.bmp yang mengalami rotasi, (c) citra lena_org.bmp yang mengalami skala, (d) citra lena_org.bmp yang mengalami translasi.

Kemudian diambil *magnitude* Fourier keempat citra :

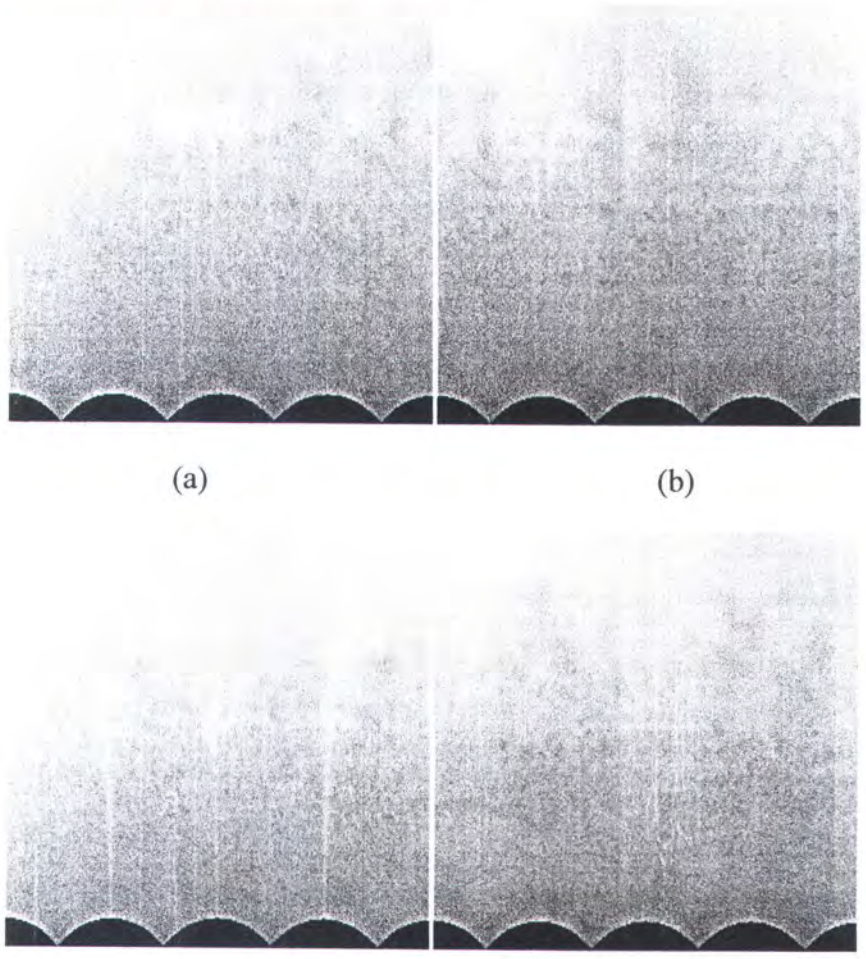


(a) (b)

(c) (d)

Gambar 2.16 *Magnitude* dari : (a) citra lena_org.bmp, (b) citra lena_org.bmp yang mengalami rotasi, (c) citra lena_org.bmp yang mengalami skala, (d) citra lena_org.bmp yang mengalami translasi.

Kemudian *magnitude* tersebut di LPM :



(a) (b)

(c) (d)

Gambar 2.17 LPM dari *magnitude* :(a) citra lena_org.bmp, (b) citra lena_org.bmp yang mengalami rotasi, (c) citra lena_org.bmp yang mengalami skala, (d) citra lena_org.bmp yang mengalami translasi.

Dari korelasi antara LPM-LPM tersebut dengan LPM citra lena_org.bmp didapat : citra yang mengalami rotasi menghasilkan korelasi sebesar 0,9706, citra yang mengalami skala menghasilkan korelasi sebesar 0.9125, dan citra yang mengalami translasi menghasilkan korelasi sebesar 1. Dari hasil korelasi terlihat LPM dari keempat *magnitude* memiliki kemiripan yang tinggi. Jika dimasukkan kemungkinan berkurangnya kualitas karena LPM, maka bisa dikatakan

pengurangan hasil korelasi disebabkan oleh LPM. Hasil ini menunjukkan independensi terhadap rotasi, skala, dan translasi (RST).

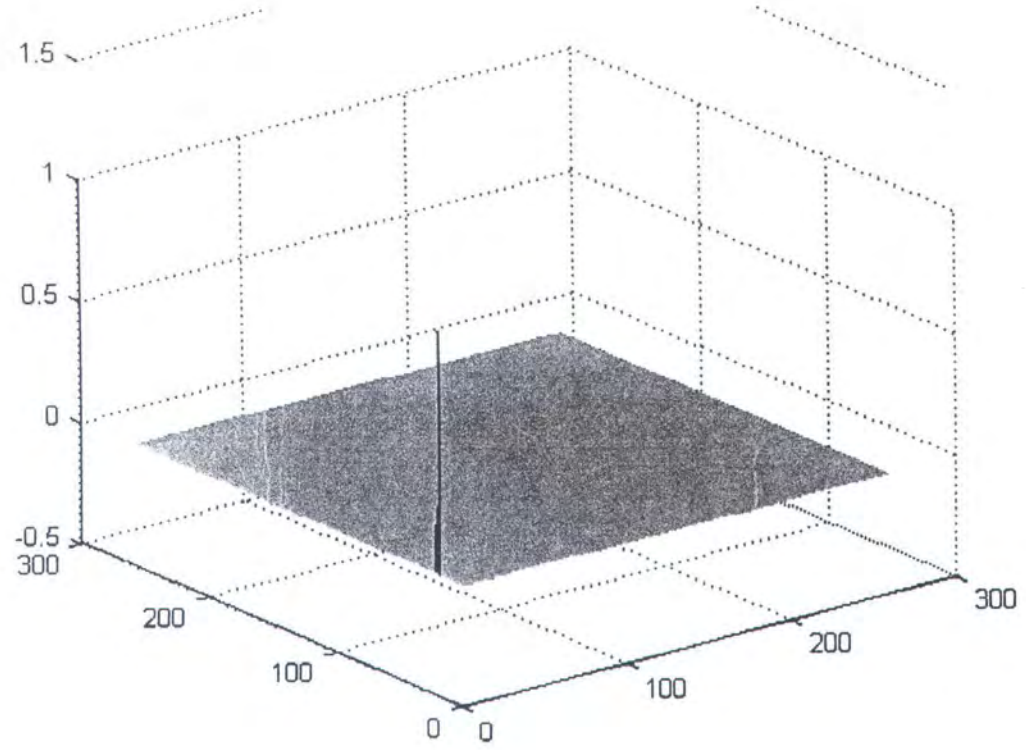
2.9 *PHASE CORRELATION*

Phase Correlation menghitung perbedaan *Phase* antara dua citra. Persamaan dari *Phase Correlation*:

$$C_{10} = \frac{F_1(\omega_\rho, \omega_\theta).* F_0^*(\omega_\rho, \omega_\theta)}{\left|F_1(\omega_\rho, \omega_\theta)\right|.*\left|F_0^*(\omega_\rho, \omega_\theta)\right|} \quad \text{.....} \quad (2.23)$$

dimana F_1 dan F_0 merupakan DFT dari citra yang dibandingkan dan citra pembandingnya dan F^* merupakan konjugasi kompleks.

Hasil persamaan diatas kemudian di IDFT untuk menghasilkan sebuah fungsi yang berbentuk *impulse* pada lokasi pergeseran. Bentuk *impulse* berarti fungsi tersebut kiri-kira nol(0) di semua tempat kecuali pada pergeseran. *Phase Correlation* digunakan untuk menghitung pergeseran (translasi) pada dua citra, citra asli dan citra yang mengalami pergeseran. Sebagai pembuktian pendeteksian pergeseran ini digunakan citra yang ada pada gambar 2.15 (a) citra lena_org.bmp dan 2.15 (d) citra lena_org.bmp yang mengalami translasi 20 *pixel* keatas, *Phase Correlation* dari kedua citra menjadi :



Gambar 2.18 Hasil *Phase Correlation* dari citra lena_org.bmp dengan citra lena_org.bmp yang mengalami translasi.

Apabila diambil posisi (dengan perintah *find* dari matlab) dari puncaknya di dapatkan :

```
[x y] = find(pc == max(max(pc)))
x =    21
y =     1
```

dari posisi puncak terlihat puncak berada pada baris ke 21 dan kolom ke 1, jadi terdapat pergeseran baris sebesar 20 *pixel* keatas.

BAB III
DESAIN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai desain perangkat lunak *Embedding Watermark* dan Ekstraksi *Watermark* dengan *Log-polar Mapping* (LPM) dan *Phase Correlation*. Selain itu dijelaskan pula desain perangkat lunak pendukung *Watermark*, penghitung nilai *Peak Signal to Noise Ratio* (PSNR), dan pembuatan tabel lokasi *Watermark*. Pembahasan ini meliputi lingkungan desain perangkat lunak, masukan dan luaran, serta desain proses. Desain perangkat lunak ini menggunakan notasi *Flowchart*. *Flowchart* dipergunakan untuk menjelaskan alur proses yang terjadi.

3.1 LINGKUNGAN DESAIN

Pada bagian ini akan dijelaskan mengenai lingkungan pendesainan aplikasi yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam *Embedding* dan Ekstraksi *Watermark* dengan *Log-polar Mapping(LPM)* dan *Phase Correlation* ini tampak pada tabel berikut:

Tabel 3.1 Lingkungan pendesainan aplikasi

Perangkat Keras	Prosesor : AMD Athlon 800MHz Memori : 256 MB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pendesain : Microsoft Word 2002

3.2 MASUKAN DAN LUARAN

Masukkan sistem pendukung pembuatan *watermark* berupa sebuah *string* yang menjadi data *watermark*, panjang *pseudo-random noise* (*PN*) hasil *PN generator*, polinom dari *PN generator*, nilai inisialisasi *register PN generator*, panjang codeword *Reed-Solomon (RS)*, panjang data *RS*, panjang *watermark*, dan lebar *watermark*. Luaran sistem ini adalah *watermark* biner.

Masukkan dari sistem *Embedding Watermark* berupa citra, *watermark* yang ingin di *embed* ke dalam citra, kekuatan *watermark*, posisi *watermark* dalam domain *Log-Polar Mapping* (baris dan kolom), dan resolusi domain *Log-Polar Mapping* dari *watermark*. Luaran dari sistem ini berupa citra yang ber-*watermark*.

Sedangkan masukkan dari sistem Ekstraksi *Watermark* berupa citra ber-*watermark*, citra asli, *watermark* asli, posisi *watermark* dalam domain *Log-Polar Mapping* (baris dan kolom)), resolusi domain *Log-Polar Mapping* dari *watermark*, dan resolusi domain *Log-Polar Mapping* untuk *phase correlation*. Luaran dari sistem ini berupa hasil korelasi (*Similiarity*) terbaik antara *watermark* asli dengan *watermark* hasil ekstraksi, nilai puncak penyesuaian terhadap translasi dimana nilai korelasinya terbaik, dan indeks puncak tersebut.

Untuk masukkan sistem pendukung penghitung *Peak Signal to Noise Ratio (PSNR)* berupa dua matriks yang akan dibandingkan. Luaran dari sistem ini adalah nilai *PSNR* dalam dB.

Untuk masukkan sistem pendukung pembuatan tabel lokasi *watermark* berupa citra, *watermark*, resolusi domain *Log-Polar Mapping*, kekuatan *minimal* dan *maksimal watermark*, dan posisi kolom *minimal* dan *maksimal watermark*. Luaran dari sistem ini berupa tabel pasangan nilai *PSNR* dan hasil korelasi.

3.3 DESAIN PEMBUATAN WATERMARK

Secara umum pada Pembuatan *Watermark* ini terdapat 5 langkah, yaitu :

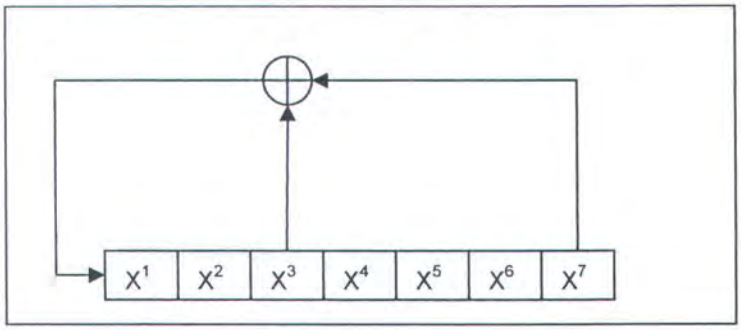
a. Langkah pertama yang disebut dengan mengambil data biner dari *string* sebagai data *watermark*.

   Pada langkah ini *string* yang menjadi masukkan diubah menjadi bilangan biner, bilangan biner ini didapat dari nilai *ascii* dari masing-masing huruf dalam *string*. Hasil dari langkah ini berupa sebuah *array* satu dimensi yang memiliki panjang 7 x jumlah huruf dalam *string*. *Array* ini dinamai *wData*.

b. Langkah kedua yang disebut dengan membuat barisan *PseudoRandom Noise*.

   Pada langkah ini digunakan metode *M-sequence (M-seq)* untuk menghasilkan *PN* yang digunakan dalam *spread spectrum (SS)*. Metode *M-seq* menggunakan

sebuah *shift register* dengan *feedback*[3], posisi *feedback* tersebut ditentukan sesuai dengan polinominal yang didapat sebagai masukkan.



Gambar 3.1 Generator *Shift Register* untuk M-Sequence dengan polinominal [3,1]

Nilai inisialisasi dari *register* didapat dari masukkan. Proses dalam *shift register* ini dimulai dengan mengeluarkan *register* paling kanan sebagai luaran satu bit, kemudian *register* di-*shift* kekanan satu bit. Setelah *shift* terjadi maka dihitung pengaruh *feedback* pada bagian yang memilikinya. Lalu proses kembali ke permulaan (mengeluarkan *register*), dan diulangi lagi hingga mendapatkan *PN* dengan panjang yang diinginkan. Hasil dari generator ini memiliki panjang sesuai dengan panjang *Pseudorandom Noise* yang menjadi masukkan dan dimasukkan ke dalam variabel *PN*.

c. Langkah ketiga yang disebut dengan melakukan *spreading* data asli dengan barisan *PN* yang dihasilkan oleh proses sebelumnya.

   Pada langkah ini dilakukan *spreading* terhadap data *watermark* dengan melakukan operasi *XOR* antara masing-masing bit dari data asli dengan barisan *PN* yang menjadi masukkan. Hasil dari operasi ini adalah sebuah *array* dua dimensi yang berukuran panjang *Pseudorandom Noise* kali ukuran data *watermark* sebelumnya. *Array* ini dinamai *wsData*.

d. Langkah keempat yang disebut dengan menambahkan *Error Control Code* pada *watermark* dengan metode *reed-solomon*.

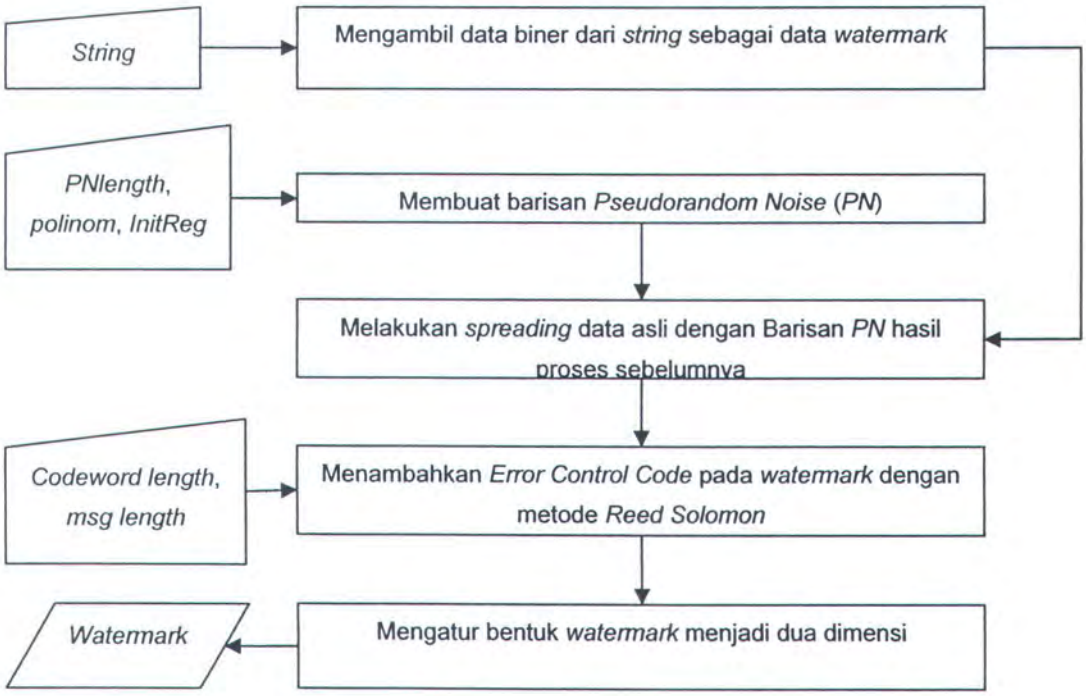
   Pada langkah ini dilakukan penambahan *Error Control Code* Reed Solomon dalam data *watermark* hasil proses sebelumnya. Metode *Reed-Solomon (RS)*

yang meliputi $GF(p^m)$ adalah sebuah kode *BCH* dengan panjang $n = p^m-1$, berdimensi $k = n-d+1$, dan jarak minimum d [4].

e. Langkah kelima yang disebut dengan mengatur bentuk *Watermark* menjadi dua dimensi.

   Pada langkah ini hasil proses sebelumnya yang berupa *array* satu dimensi diubah menjadi sebuah matriks dua dimensi dengan ukuran panjang kali lebarnya sama dengan panjang *array*. Panjang dan lebar ini merupakan masukkan. Hasil perubahan ini menjadi luaran dari sistem.

Alur dari langkah-langkah pembuatan *watermark* tersebut secara umum dapat digambarkan dengan menggunakan *Flowchart* yang tampak pada gambar 3.2.



Gambar 3.2 *Flowchart* pembuatan *watermark*

3.4 DESAIN *EMBEDDING WATERMARK* DENGAN *LOG-POLAR MAPPING* DAN *PHASE CORRELATION*

Secara umum pada *Embedding Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation* dasar ini terdapat 5 langkah utama, yaitu :

a. Inisialisasi

Pada langkah ini diinisialisasi variabel *wmCartesian*. Variabel *wmCartesian* ini diinisialisasikan dengan matriks berukuran sama dengan ukuran citra masukkan dan bernilai nol (0).

b. Langkah pertama yang disebut dengan menghitung *magnitude* dan *phase* Fourier.

Pada langkah ini dihitung transformasi Fourier dari citra (variabel masukkan *oImg*), lalu matriks digeser sedemikian hingga komponen yang berfrekuensi nol dari hasil transformasi berada di tengah matriks. Perhitungan transformasi ini dilakukan sesuai dengan persamaan *Discrete Fourier Transform* (persamaan 2.7). Dari hasil transformasi ini dihitung *magnitude* (persamaan 2.9), dan *phase*(persamaan 2.10) yang kemudian dimasukkan ke dalam variabel *magnitudeImage* dan *phaseImage*.

c. Langkah kedua yang disebut pemetaan *Watermark* ke sistem koordinat kartesian.

Pada langkah ini dibuat sebuah matriks berukuran sama dengan citra yang berisi *watermark* dari masukkan (variabel masukkan *WM*) dalam domain kartesian. Lokasi dari *watermark* dalam domain kartesian ini dihitung dengan menggunakan persamaan *Log-Polar Mapping* (persamaan 2.11 hingga 2.15), dengan anggapan posisi (1,1) dari *watermark* berada pada posisi *watermark* dalam domain *Log-Polar Mapping* yang menjadi masukkan (variabel masukkan *wmPositionRows* dan *wmPositionCols*). Ukuran domain *Log-Polar Mapping* ini ditentukan oleh masukan resolusi *Log-Polar Mapping* (variabel masukkan *lpmResolution*). Kemudian masing-masing elemen dari *watermark* dihitung posisinya dalam domain kartesian dan nilai dari masing-masing elemen dimasukkan dalam *container*(variabel penyimpan) berukuran sama dengan citra sesuai dengan posisinya dalam domain kartesian tersebut. Kemudian *container* tersebut ditambahkan dengan pencerminan dari *container* terhadap sumbu diagonal untuk simetrisitas. *Container* ini berupa variabel *wmCartesian*.

d. Langkah ketiga yang disebut dengan *Embedding Watermark* ke dalam *Magnitude*.

Pada langkah ini dilakukan penambahan *watermark* dengan persamaan [1]:

$$|F'(u,v)| = |F(u,v)| + \alpha * W \quad \text{..... (3.1)}$$

dimana $|F'(u,v)|$ adalah *magnitude* ber-*watermark*, $|F(u,v)|$ adalah *magnitude* citra masukkan, α kekuatan *watermark* dan W adalah *watermark* dalam domain kartesian. α merupakan masukkan dari sistem (variabel masukkan *alpha*). $|F(u,v)|$ dan W merupakan hasil dari langkah-langkah sebelumnya, yaitu variabel *magnitudeImage* dan *wmCartesian*. Perkalian α dengan W adalah perkalian konstanta dengan matriks, sedangkan penjumlahan *magnitude* dengan hasil perkalian itu adalah penjumlahan matriks. Hasil dari langkah ini dimasukkan ke variabel *wmMagnitudeImage*.

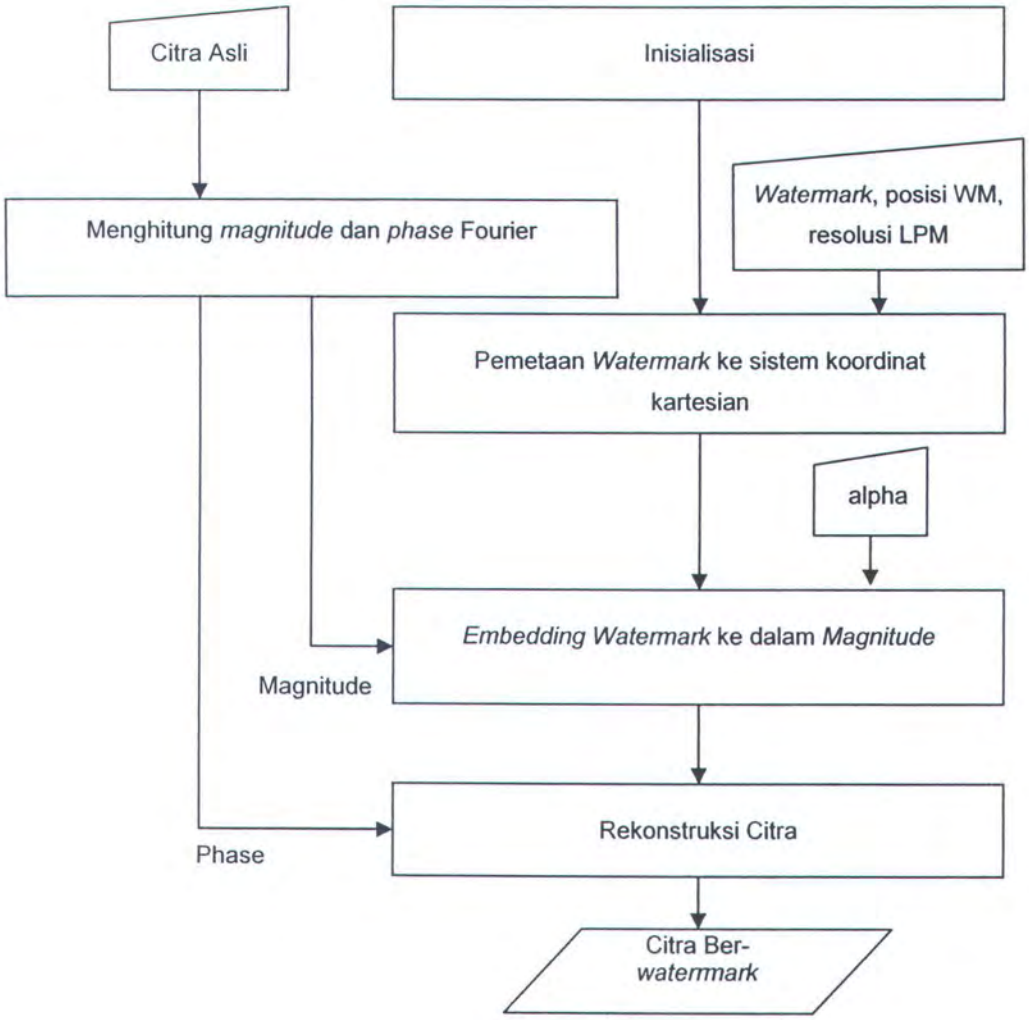
e. Langkah keempat yang disebut dengan Rekonstruksi Citra.

Pada langkah ini hasil transformasi *Fourier* yang telah berubah menjadi *magnitude* dan *phase* dikembalikan lagi dengan menggunakan *magnitude* ber-*watermark* dan *phase* citra masukkan. Pengembalian ini dilakukan dengan persamaan [2]:

$$F(u,v) = |F(u,v)| .* e^{i*\phi(u,v)} \quad \text{..... (3.2)}$$

dimana $F(u,v)$ adalah hasil transformasi Fourier, $|F(u,v)|$ adalah *magnitude* dan $\Phi(u,v)$ adalah *phase*. $|F(u,v)|$ dan $\Phi(u,v)$ merupakan hasil dari langkah-langkah sebelumnya, yaitu variabel *wmMagnitudeImage* dan *phaseImage*, i merupakan bilangan imajiner, sedangkan e adalah bilangan natural (eksponensial). Perkalian $|F(u,v)|$ dengan hasil eksponensial adalah perkalian antar elemen matriks. Lalu $F(u,v)$ ditransformasi balik dengan persamaan *Inverse Discrete Fourier Transform* (persamaan 2.8), untuk mendapatkan citra ber-*watermark*.

Alur dari langkah-langkah *Embedding Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation* tersebut secara umum dapat digambarkan dengan menggunakan *Flowchart* yang tampak pada gambar 3.3.



Gambar 3.3 *Flowchart Embedding Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation*

3.5 DESAIN EKSTRAKSI *WATERMARK* DENGAN *LOG-POLAR MAPPING* DAN *PHASE CORRELATION*

Secara umum pada Ekstraksi *Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation* ini terdapat 13 langkah, yaitu :

a. Inisialisasi.

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *bestResult, dscale, Result, idx, lastPeak, lastIdx*. Variabel *bestResult,*

Result, idx, lastPeak, dan *lastIdx* diinisialisasikan dengan nol (0). Sedangkan *dscale* diinisialisasikan dengan satu (1).

b. Langkah pertama yang disebut dengan menyamakan ukuran kedua gambar.
Pada langkah ini ukuran kedua citra dibuat sama dengan menambahkan nol(*zeropad*) pada citra yang lebih kecil. Hasilnya dimasukkan ke dalam variabel *oImgP* untuk citra asli (*oImg*) dan *wImgP* untuk citra ber-*watermark* (*wImg*). Menyamakan ukuran kedua gambar ini merupakan syarat dari proses perhitungan *Phase Correlation.*

c. Langkah kedua yang disebut dengan menghitung *Magnitude* Fourier kedua citra.
Pada langkah ini dilakukan perhitungan transformasi menggunakan persamaan *Discrete Fourier Transform* (persamaan 2.7), terhadap citra asli dan citra ber-*watermark* hasil proses sebelumnya (variabel *oImgP* dan *wImgP*). Hasil transformasi ini digeser sedemikian hingga komponen berfrekuensi nol berada di tengah matriks. Lalu diambil *magnitude* (variabel *oMag* untuk citra asli dan *wMag* untuk citra ber-*watermark*) dari masing-masing hasil transformasi *Fourier* kedua citra tersebut dengan persamaan *Magnitude* (persamaan 2.9).

d. Langkah ketiga yang disebut *high-pass* filter kedua *magnitude*.
Pada langkah ini dilakukan *filtering* terhadap kedua *magnitude* (variabel *oMag* dan *wMag*) untuk mengurangi nilai komponen berfrekuensi rendah sehingga tidak mengganggu proses perhitungan selanjutnya. Nilai komponen berfrekuensi rendah ini biasanya bernilai jauh lebih tinggi dari komponen berfrekuensi menengah ke tinggi, hal ini mengakibatkan perubahan pada komponen berfrekuensi rendah ke tinggi terlihat tidak signifikan, oleh sebab itu nilai komponen berfrekuensi rendah ini perlu dikurangi.Untuk membuat filter ini digunakan persamaan :

$$eta = \cos(\pi\theta_x) \quad \text{(3.3)}$$

$$neta = \cos(\pi\theta_y) \quad \text{(3.4)}$$

$$X = eta^T * neta \quad \text{(3.5)}$$

$$H = (1 - X).*(2 - X) \quad \text{(3.6)}$$

dengan θ_x sudut antara -½π hingga ½π berjumlah ½ panjang *Magnitude* dan θ_y sudut antara -½π hingga ½π berjumlah ½ lebar *Magnitude*

e. Langkah keempat yang disebut dengan *Log-Polar Mapping* dari kedua *Magnitude*.

   Pada langkah ini dilakukan transformasi *Log-Polar Mapping* (persamaan 2.11 sampai persamaan 2.15) dari kedua *magnitude* yang sudah difilter sebelumnya. Transformasi ini dilakukan untuk masing-masing titik dari domain *Log-Polar Mapping* dengan resolusi untuk *Phase Correlation* yang menjadi masukkan (variabel masukan *PCResolution*). Pengambilan nilai dari domain kartesian dilakukan dengan menggunakan interpolasi *bilinear*. Hasil transformasi ini dimasukkan dalam variabel *oLpm* untuk citra asli dan *wLpm* untuk citra ber-*watermark*.

f. Langkah kelima yang disebut dengan *Phase Correlation* kedua hasil *Log-Polar Mapping*.

   Pada langkah ini dilakukan perhitungan *Phase Corelation*. *Phase Correlation* adalah operasi pencarian perbedaan *Phase* dari *Discrete Fourier transform* antar dua matriks. *Phase* dari *Discrete Fourier transform* merepresentasikan bentuk geometrik dari sebuah citra, maka dengan melakukan perbandingan antara kedua *Phase* bisa didapatkan pergeseran yang terjadi pada citra yang dibandingkan (Hasil *Log-Polar Mapping* citra ber-*watermark*). Hasil dari *Phase Correlation* ini akan berupa sebuah matriks berbentuk *impulse* dimana secara ideal akan terjadi puncak pada posisi yang mengalami pergeseran. Persamaan dari *Phase Correlation* ini adalah persamaan 2.23. Hasil persamaan diatas di transformasi balik dengan *Inverse Discrete Transform* (persamaan 2.8), dari hasil transformasi diambil nilai-nilai puncak yang akan digunakan dalam proses penyesuaian. Nilai-nilai ini dimasukkan ke dalam variabel *peaksVal*.

g. Langkah keenam yang disebut dengan evaluasi puncak-puncak.

   Pada langkah ini dilakukan penyesuaian untuk masing masing puncak hasil *Phase Correlation*. Bagian ini sebenarnya adalah bagian awal dari proses *looping* yang dilakukan sebanyak 200 kali atau apabila nilai variabel *lastPeak*

sudah melewati *threshold* 0,3. Nilai 200 dan 0,3 didapat dari uji coba. Sedangkan variabel *lastPeak* akan dijelaskan di bagian selanjutnya. Pada bagian awal ini juga dicari posisi dari puncak (*dPeaksS, dPeaksR*). Dan *increment* untuk variabel *idx* untuk penunjuk jumlah proses *looping*.

h. Langkah ketujuh yang disebut dengan penyesuaian terhadap rotasi.

Pada langkah ini dilakukan perhitungan terhadap posisi puncak yang sedang dievaluasi hasil pencarian sebelumnya. Perubahan yang terjadi di domain *Log-Polar Mapping* ketika terjadi rotasi adalah pergeseran baris (variabel *dPeaksR*). Parameter rotasi dihitung dengan persamaan [5]:

$$\alpha' = \frac{360 * \Delta\theta}{N'} \quad \text{.....(3.7)}$$

dimana : $\Delta\theta$ adalah besar pergeseran

N' adalah resolusi baris dari LPM

dan hasilnya adalah besar rotasi searah jarum jam yang merupakan rotasi yang terjadi pada citra. Parameter ini digunakan untuk melakukan penyesuaian pada citra hasil proses menyamakan ukuran citra (*oImgP* dan *wImgP*) sebesar negatif dari derajat hasil. Ada dua operasi penyesuaian terhadap rotasi: yang pertama rotasi dengan nilai rotasi $-\alpha'$, sedangkan yang kedua rotasi dengan nilai rotasi $-(\alpha'+180)$. Hal ini disebabkan untuk rotasi > 180 derajat, nilai yang dihasilkan adalah sama dengan rotasi dikurangi 180. Karena kesamaan nilai inilah maka diperlukan dua hasil penyesuaian terhadap rotasi. Hasil penyesuaian terhadap rotasi yang benar akan diketahui ketika melakukan penyesuaian terhadap translasi. Kedua hasil ini dimasukkan ke dalam variabel *wImg1* untuk rotasi < 180 derajat dan *wImg2* untuk rotasi > 180 derajat.

i. Langkah kedelapan yang disebut dengan penyesuaian terhadap skala.

Pada langkah ini dilakukan perhitungan terhadap posisi puncak yang sedang dievaluasi hasil pencarian sebelumnya. Perubahan di domain *Log-Polar Mapping* apabila terjadi perubahan skala adalah pergeseran kolom (variabel *dPeaksS*). Parameter skala ini dihitung dengan menggunakan persamaan [5] :

$$\sigma' = \frac{\ln(r_{max})\Delta\rho}{M'} \quad \text{..... (3.8)}$$

$$r_{max} = \sqrt{(M/2)^2 + (N/2)^2} \quad \text{..... (3.9)}$$

dimana : $\Delta\rho$ = besar pergeseran

M' = resolusi kolom dari LPM

M = resolusi kolom citra

N = resolusi baris citra

Hasil dari perhitungan tersebut adalah nilai skala yang terjadi pada citra. Nilai yang digunakan untuk penyesuaian terhadap skala ini adalah 1/hasil perhitungan. Citra yang disesuaikan adalah dua citra hasil penyesuaian terhadap rotasi sebelumnya. Hasil dari penyesuaian dimasukkan ke dalam variabel yang sama (*wImg1* dan *wImg2*).

j. Langkah kesembilan yang disebut dengan *Phase Correlation* antara hasil penyesuaian dengan citra asli.

Pada langkah ini dilakukan perhitungan *Phase Correlation*. *Phase Correlation* yang dilakukan disini adalah untuk mendapatkan pergeseran yang terjadi akibat translasi. Translasi adalah pergeseran di dalam domain kartesian, oleh sebab itu perlu melakukan *Phase Correlation* citra hasil penyesuaian sebelumnya dengan citra asli dalam domain kartesian. Sebelum dilakukan *Phase Correlation*, ukuran citra perlu disamakan dengan menambahkan nol (*zeropad*) pada citra asli yang menjadi masukkan sehingga memiliki ukuran yang sama dengan citra hasil penyesuaian sebelumnya. Hanya citra asli yang perlu ditambahkan karena ukuran citra asli selalu lebih kecil dari citra hasil penyesuaian. Namun jika citra hasil penyesuaian sebelumnya memiliki ukuran lebih kecil maka proses kembali ke langkah keenam. Hasil *Phase Correlation* dimasukkan ke dalam variabel *d11* untuk citra yang mengalami penyesuaian terhadap rotasi < 180 dan *d12* untuk citra yang mengalami penyesuaian terhadap rotasi > 180.

k. Langkah kesepuluh yang disebut dengan penyesuaian terhadap translasi.

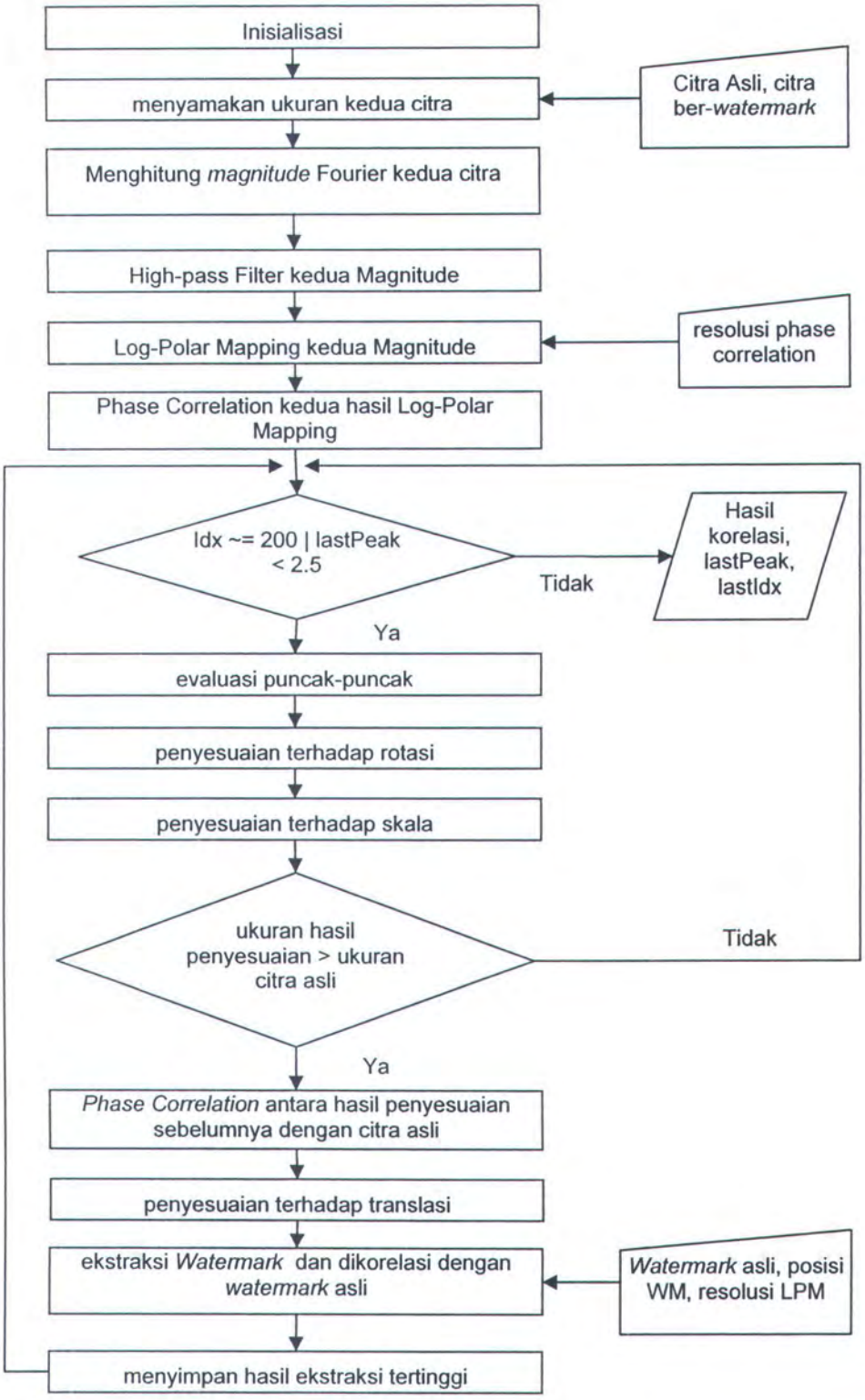
Pada langkah ini dilakukan perhitungan terhadap hasil *Phase Correlation* dalam domain kartesian. Perhitungan parameter translasi dimulai dengan pengambilan puncak tertinggi dari dua hasil *Phase Correlation*, dimana yang satu (*d11*) disesuaikan terhadap rotasi secara normal (variabel puncak

Operasi $(.)^T$ adalah operasi transpose.

Hasil korelasi dimasukkan ke dalam variabel *Result*.

m. Langkah keduabelas yang disebut dengan menyimpan hasil ekstraksi tertinggi. Pada langkah ini dilakukan penyimpanan nilai puncak dan nilai hasil perhitungan Korelasi sebelumnya. Apabila nilai puncak yang disimpan (variabel *lastPeak*) lebih kecil dari nilai puncak yang sedang dievaluasi (variabel *peakVal3*) maka nilai *peakVal3* menjadi nilai dari *lastPeak*, nilai *idx* menjadi *lastIdx,* dan nilai *Result* menjadi nilai *bestResult*. Variabel *bestResult, lastPeak,* dan *lastIdx* ini menjadi luaran dari sistem ketika *looping* selesai, yaitu hasil korelasi terbaik yang merupakan luaran utama, puncak yang memiliki hasil korelasi terbaik, dan indeks puncak tersebut. Setelah langkah ini maka proses kembali ke langkah keenam.

Alur dari langkah-langkah Ekstraksi *Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation* tersebut secara umum dapat digambarkan dengan menggunakan *Flowchart* yang tampak pada gambar 3.4.



Gambar 3.4 *Flowchart* Ekstraksi *Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation*

3.6 DESAIN PENGHITUNG *PEAK SIGNAL TO NOISE RATIO* (PSNR)

Penghitung *Peak Signal to Noise Ratio (PSNR)* ini merupakan implementasi dari persamaan [10]:

$$PSNR = 10\log_{10}\left(\frac{N * \max(x_i^2)}{\sum_{i=1}^{N}(x_i - x_i')^2}\right) \quad \text{.....(3.11)}$$

dimana : N adalah jumlah titik dalam citra

x_i adalah nilai *pixel* asli (dari citra asli)

x_i' adalah nilai *pixel* yang mengalami perubahan (dari citra ber-*watermark*)

Proses ini tidak bisa dilakukan jika kedua citra adalah sama. Luaran dari persamaan diatas bersatuan dB.

3.7 DESAIN PEMBUATAN TABEL LOKASI *WATERMARK*

Secara umum pada pembuatan tabel lokasi *watermark* ini terdapat 4 langkah, yaitu :

a. Langkah pertama yang disebut dengan perhitungan tiap-tiap pasangan kekuatan *watermark* dan posisi kolom dari *watermark*.

   Langkah ini merupakan awal dari *nested loop* dua kali, yang pertama *loop* untuk kekuatan *watermark*, sedangkan yang kedua adalah *loop* untuk posisi kolom dari *watermark* di dalam domain *Log-Polar Mapping*. *Loop* pertama dikerjakan sebanyak nilai minimum kekuatan *watermark* dikurangi nilai maksimum kekuatan *watermark*. Nilai minimum dan maksimum ini didapat dari masukkan. *Loop* pertama ini dimulai dari nilai variabel X sama dengan nilai minimum kekuatan *watermark*. Sedangkan *Loop* kedua dikerjakan sebanyak posisi kolom minimum dikurangi posisi kolom maksimum dari *watermark* di dalam domain *Log-Polar Mapping*. Posisi kolom maksimum dan minimum ini didapat dari masukkan. *Loop* pertama dimulai dari nilai variabel Y sama dengan posisi kolom minimum *watermark* dalam domain *Log-Polar Mapping*.

b. Langkah kedua yang disebut dengan *embedding watermark.*

   Pada langkah ini dilakukan *embedding watermark* terhadap citra asli pada posisi (1,*Y*) dengan kekuatan *X*. Resolusi *Log-Polar Mapping* dan *watermark* ditentukan dari masukkan. Hasilnya dimasukkan dalam variabel *wImg*.

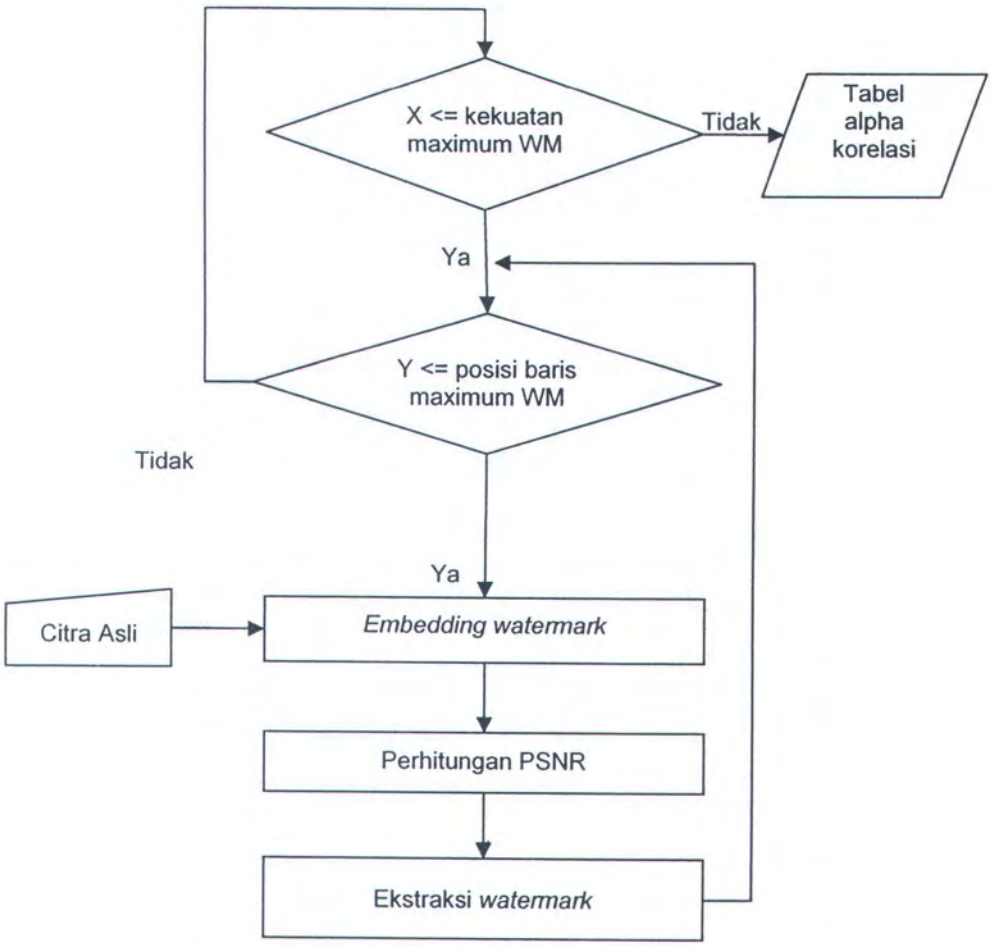
c. Langkah ketiga yang disebut perhitungan *PSNR*.

   Pada langkah ini dihitung nilai *Peak Signal to Noise Ratio (PSNR)* dari citra asli dengan citra ber-*watermark* hasil *embedding* sebelumnya (*wImg*). Hasil dari perhitungan ini dimasukkan ke dalam tabel *PResult.*

d. Langkah keempat yang disebut ekstraksi *watermark.*

   Pada langkah ini dilakukan ekstraksi *watermark* dari citra ber-*watermark* hasil proses sebelumnya (*wImg*). Posisi *watermark* adalah (1,*Y*) dengan kekuatan *watermark X*. Resolusi *Log-Polar Mapping* dan *watermark* ditentukan dari masukkan, sedangkan resolusi *Phase Correlation* ditentukan 10 untuk mempercepat proses ekstraksi dari sebuah citra ber-*watermark* yang normal. Hasil ekstraksi ini dimasukkan ke dalam tabel *PResult.* Tabel *PResult* ini menjadi luaran dari sistem. Setelah proses ini maka dilanjutkan ke pasangan kekuatan *watermark* dan posisi kolom *watermark* selanjutnya.

Alur dari langkah-langkah pembuatan tabel lokasi *watermark* tersebut secara umum dapat digambarkan dengan menggunakan *Flowchart* yang tampak pada gambar 3.5.



Gambar 3.5 *Flowchart* pembuatan tabel lokasi *watermark*

BAB IV

IMPLEMENTASI PERANGKAT LUNAK

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai implementasi perangkat lunak *Embedding Watermark* dan Ekstraksi *Watermark* dengan *Log-polar Mapping (LPM)* dan *Phase Correlation*. Selain itu dijelaskan pula implementasi perangkat lunak pendukung pembuatan *Watermark*, penghitung nilai *Peak Signal to Noise Ratio (PSNR)*, dan pembuatan tabel lokasi *Watermark*. Pembahasan ini meliputi lingkungan implementasi perangkat lunak, *pseudocode* program untuk setiap proses, serta antarmuka perangkat lunak.

4.1 LINGKUNGAN IMPLEMENTASI

Pada bagian ini akan dijelaskan mengenai lingkungan pembangunan aplikasi yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pembangunan aplikasi *watermarking* ini tampak pada tabel berikut:

Tabel 4.1 Lingkungan pembangunan aplikasi

Perangkat Keras	Prosesor : AMD Athlon 800 MHz Memori : 256 MB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pembangun : Matlab 6.1.0

4.2 PEMBUATAN *WATERMARK*

Terdapat lima proses pada Pembuatan *Watermark*, yaitu:

1. mengambil data biner dari *string* sebagai data *watermark*
2. membuat barisan *PseudoRandom Noise*
3. melakukan *spreading* data asli dengan *Barisan PN* yang dihasilkan oleh proses sebelumnya
4. menambahkan *Error Control Code* pada *watermark* dengan metode *reed-solomon*

4.2.3 Melakukan *Spreading* Data Asli dengan *Barisan PN* yang Dihasilkan oleh Proses Sebelumnya

Pada langkah ini dilakukan *spread*, dengan fungsi *spread,* terhadap data dengan melakukan operasi *XOR* antara masing-masing bit dari data asli dengan *Barisan PN* yang menjadi masukkan. *Pseudocode* program melakukan *spreading* data asli dengan *Barisan PN* yang dihasilkan oleh proses sebelumnya tampak pada gambar 4.3.

```
5  wsData = spread(wData,PN);
```

Gambar 4.3 Kode program melakukan *spreading* data asli dengan *Barisan PN* yang dihasilkan oleh proses sebelumnya dalam Pembuatan *Watermark*

4.2.4 Menambahkan *Error Control Code* (ECC) pada *Watermark* dengan Metode Reed-Solomon

Pada langkah ini ditambahkan *ECC* Reed Solomon dalam *watermark.* Metode *Reed-Solomon (RS)* yang meliputi $GF(p^m)$ adalah sebuah kode *BCH* dengan panjang $n = p^m-1$, berdimensi $k = n-d+1$, dan jarak minimum d [4]. *Pseudocode* program menambahkan *Error Control Code* pada *watermark* dengan metode *reed-solomon* ini tampak pada gambar 4.4.

```
6  WM = rsencorow(wsData,7,6);
```

Gambar 4.4 Kode program menambahkan *Error Control Code* pada *watermark* dengan metode *reed-solomon* dalam pembuatan *watermark*

4.2.5 Mengatur Bentuk *Watermark* Menjadi Dua Dimensi

Pada langkah ini digunakan fungsi *reshape* dari *matlab* yang membentuk sebuah *array* menjadi matriks yang memiliki ukuran yang berbeda namun jumlah elemennya tetap sama. *Pseudocode* program mengatur bentuk *Watermark* menjadi dua dimensi ini tampak pada gambar 4.5.

```
7   WM = reshape(WM,9,7);
```

Gambar 4.5 Kode program mengatur bentuk *Watermark* menjadi dua dimensi dalam pembuatan *watermark*

4.3 *EMBEDDING WATERMARK* DENGAN *LOG-POLAR MAPPING (LPM)* DAN *PHASE CORRELATION.*

Proses yang terjadi pada *Embedding Watermark* dengan *Log-polar Mapping (LPM)* dan *Phase Correlation* adalah sebagai berikut :

1. Inisialisasi
2. Menghitung *Magnitude* dan *Phase Fourier*
3. Pemetaan *Watermark* ke sistem koordinat kartesian
4. *Embedding Watermark* ke dalam *Magnitude*
5. Rekonstruksi citra.

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

4.3.1 Inisialisasi

Pada langkah ini diinisialisasi *header* fungsi (fungsi *embed*) dan variabel *wmCartesian* dengan ukuran sama dengan citra asli dan bernilai nol(0). *Pseudocode* untuk proses inisialisasi yaitu:

```
1   function wImg = embed(oImg,WM,wmPositionRows,wmPositionCols,alpha,lpmResolution)
2
3   %creating result container for watermark in LPM domain
4   wmCartesian = zeros(size(oImg));
```

Gambar 4.6 Kode program inisialisasi *Embedding Watermark*

4.3.2 Menghitung *Magnitude* dan *Phase* Fourier

Pada langkah ini dilakukan perhitungan *Discrete Fourier Transform* dengan menggunakan fungsi *fft2* dari *matlab*, kemudian komponen berfrekuensi nol digeser ke tengah dengan fungsi *fftshift*. Setelah itu diambil *magnitude* dan *phase* dengan fungsi *abs* dan *angle* dari *matlab. Pseudocode* untuk proses menghitung *magnitude* dan *phase* Fourier tampak pada gambar 4.7.

```
5   %generate dfts
6   dftImage = fftshift(fft2(oImg));
7   magnitudeImage = abs(dftImage);
8   phaseImage = angle(dftImage);
```

Gambar 4.7 Kode program menghitung *magnitude* dan *phase* Fourier dalam *Embedding watermark*

4.3.3 Pemetaan *Watermark* ke Sistem Koordinat Kartesian

Pada langkah ini *container wmCartesian* diisi *watermark* dengan menggunakan fungsi *appilpm* (gambar 4.9). Fungsi *appilpm* ini merupakan implementasi persamaan 2.11 pada baris ke-32 dari fungsi *appilpm,* persamaan 2.12 pada baris ke-33 dari fungsi *appilpm,* persamaan 2.13 pada baris ke-15 dari fungi *appilpm,* persamaan 2.14 pada baris ke-6 dan ke-7 dari fungsi *appilpm,* persamaan 2.15 pada baris ke-16 dari fungsi *appilpm.*

Yang digunakan untuk melakukan transformasi pada *watermark* (*WM*) yang berada di posisi (*wmPositionRows,wmPositionCols*) dalam domain *Log-Polar Mapping* ke domain kartesian. Penambahan simetrisitas dari *watermark* dimulai dari baris ke-55 dari fungsi *appilpm* hingga ditambahkan pada baris ke-78. *Pseudocode* program pemetaan *watermark* ke sistem koordinat kartesian tampak pada gambar 4.8 dan gambar 4.9.

```
9   %embed wm into container
10  wmCartesian = appilpm(wmCartesian,WM,wmPositionRows, ...
11                wmPositionCols,lpmResolution);
```

Gambar 4.8 Kode program pemetaan *watermark* ke sistem koordinat kartesian dalam *embedding watermark*

```
function invpolarMap = appilpm(mat,WM,wc,wr,res)

[rows, cols] = size(mat);
[wrows, wcols] = size(WM);

Rmin  = 1;
Rmax = sqrt(((rows/2)^2)+((cols/2)^2));

Midx = cols/2;
Midy = rows/2;

numr = res;
numa = res;

rrange = linspace(log(rmin), log(rmax), numr);
arange = linspace(0, 2*pi*(numa-1)/numa, numa);

rTab = exp(rrange);
sinTab = sin(arange);
cosTab = cos(arange);

invpolarMap = zeros(rows, cols);
ax = zeros(wrows,wcols);
ay = ax;

for m = 1:wrows
   for n = 1:wcols

      r = wr+m-1;
      a = wc+n-1;

      x = midx + rTab(r)* cosTab(a);
      y = midy + rTab(r)* sinTab(a);

      if (x < 1)
        x = 1;
      End
      if (y < 1)
        y = 1;
      End
```

Gambar 4.9 Kode program fungsi *appilpm*

```
41
42          imVal = WM(m,n);
43
44          x1 = floor(x);
45          x2 = x1 + 1;
46          y1 = floor(y);
47          y2 = y1 + 1;
48
49          invpolarMap(x1, y1) = imVal;
50          ax(m,n) = x1;
51          ay(m,n) = y1;
52      End
53  end;
54
55  %simetricity
56  if (mod(rows,2) ~= 1)
57      if (mod(cols,2) ~= 1)
58          %both even
59          mirMap = invpolarMap(2:rows,2:cols);
60          mirMap = rot90(rot90(mirMap));
61          invpolarMap(2:rows,2:cols) = invpolarMap(2:rows,2:cols) + mirMap;
62      End
63      %row even col odd
64      mirMap = invpolarMap(2:rows,1:cols);
65      mirMap = rot90(rot90(mirMap));

66      invpolarMap(2:rows,1:cols) = invpolarMap(2:rows,1:cols) + mirMap;
67
68  Else
69      if (mod(cols,2) ~= 1)
70          %row odd col even
71          mirMap = invpolarMap(1:rows,2:cols);
72          mirMap = rot90(rot90(mirMap));
73          invpolarMap(1:rows,2:cols) = invpolarMap(1:rows,2:cols) + mirMap;
74      End
```

Gambar 4.10 Lanjutan pertama kode program fungsi *appilpm*

```
75      %both odd
76      mirMap = invpolarMap;
77      mirMap = rot90(rot90(mirMap));
78      invpolarMap = invpolarMap + mirMap;
79  End
80  invpolarMap = mat + invpolarMap;
```

Gambar 4.11 Lanjutan kedua kode program fungsi *appilpm*

4.3.4 *Embedding Watermark* ke Dalam *Magnitude*

Pada langkah ini dilakukan implementasi terhadap persamaan 3.1 pada baris ke-dua. Langkah ini menambahkan *container* domain kartesian berisi *watermark* (*wmCartesian*) ke dalam *magnitude* (*magnitudeImage*) setelah dikalikan dengan kekuatan *watermark alpha. Pseudocode* program *Embedding Watermark* ke dalam *Magnitude* tampak pada gambar 4.12.

```
12  %adding wm into magnitude
13  wmMagnitudeImage = double(wmCartesian)*alpha + magnitudeImage;
```

Gambar 4.12 Kode program *Embedding Watermark* ke dalam *Magnitude* dalam *embedding watermark*

4.3.5 Rekonstruksi Citra

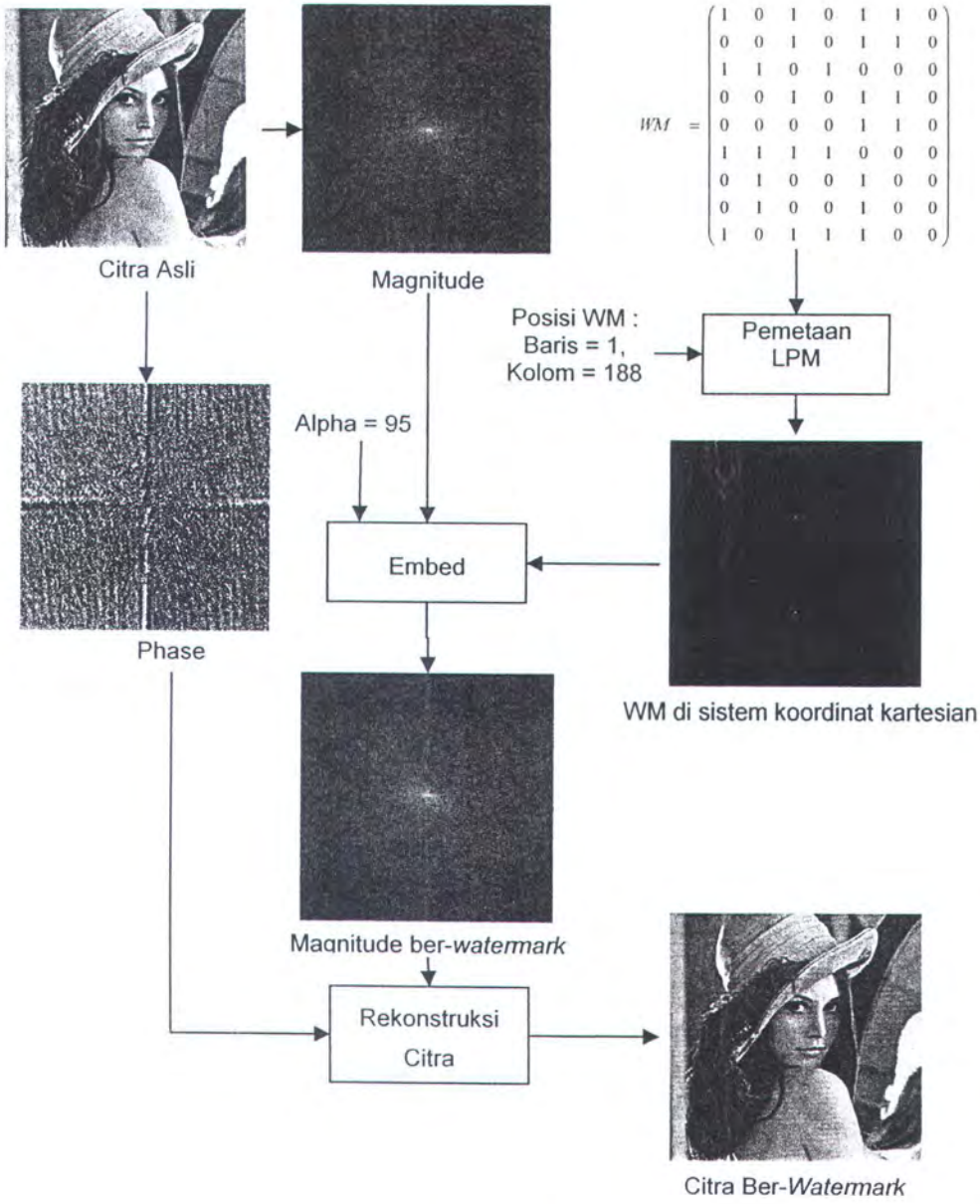
Pada langkah ini dilakukan implementasi persamaan 3.2 pada baris ke-dua. Hasil dari implementasi persamaan tersebut ditransformasi balik dengan menggunakan fungsi *ifft2* dari *matlab* setelah komponen berfrekuensi nol dikembalikan ke posisinya semula dengan fungsi *ifftshift* dari *matlab*. Hasil transformasi balik tersebut (*wImg*) menjadi nilai luaran dari fungsi *embed* ini *Pseudocode* program Rekontruksi citra ini tampak pada gambar 4.13.

```
14  %invers dft
16  wImg = wmMagnitudeImage.*exp(i*phaseImage);
16  wImg = ifft2(ifftshift(wImg));
```

Gambar 4.13 Kode program Rekonstruksi Citra dalam *Embedding Watermark*

4.3.6 Penggambaran Proses *Embedding Watermark*

Penggambaran langkah-langkah diatas dengan menggunakan sebuah citra tampak pada gambar 4.14.



Gambar 4.14 Penggambaran langkah-langkah *Embedding Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation* dengan menggunakan sebuah citra (lena_org.bmp).

4.4 EKSTRAKSI *WATERMARK* DENGAN *LOG-POLAR MAPPING (LPM)* DAN *PHASE CORRELATION*

Terdapat tigabelas proses pada Ekstraksi *Watermark* dengan *Log-polar Mapping (LPM)* dan *Phase Correlation*, yaitu:

1. Inisialisasi
2. menyamakan ukuran kedua gambar
3. menghitung *magnitude* Fourier kedua citra
4. *high-pass filter* kedua *magnitude*
5. *Log-Polar Mapping* dari kedua *Magnitude*
6. *Phase Correlation* kedua hasil *Log-Polar Mapping*
7. evaluasi puncak-puncak
8. penyesuaian terhadap rotasi
9. penyesuaian terhadap skala
10. *Phase Correlation* antara hasil penyesuaian dengan citra asli
11. penyesuaian terhadap translasi
12. Ekstraksi *Watermark* dan dikorelasi dengan *watermark* asli
13. menyimpan hasil ekstraksi tertinggi

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

4.4.1 Inisialisasi

Pada langkah ini diinisialisasikan *header* fungsi dari ekstraksi *watermark* ini (fungsi *extract*) lebih dulu, kemudian diinisialisasi variabel-variabel *bestResult, dscale, Result, idx, lastPeak, lastIdx*. Variabel-variabel *bestResult, Result, idx, lastPeak,* dan *lastIdx* diinisialisasikan dengan nol (0). Sedangkan *dscale* diinisialisasikan dengan satu (1). *Pseudocode* untuk proses inisialisasi tampak pada gambar 4.15.

```
1  function [bestResult,lastPeak,lastIdx]=extract(wImg,oImg,WM,...
2  wmPositionRows,wmPositionCols, wmStrength,lpmResolution,PCResolution)
3  bestResult = 0;
4  dscale = 1;
5  Result = 0;
6  idx = 0;
7  lastPeak = 0;
8  lastIdx = 0;
```

Gambar 4.15 Kode program inisialisasi ekstraksi *watermark*

4.4.2 Menyamakan Ukuran Kedua Gambar

Pada langkah ini ukuran kedua citra dibuat sama dengan menambahkan nol (*zeropad*) pada citra yang lebih kecil. Penambahan ini dilakukan dengan fungsi *zeroPad.* Pada baris ke-13 hingga ke-22 gambar 4.16 dilakukan penentuan mana citra yang akan di *zeropad.* Hasilnya dimasukkan ke dalam variabel *oImgP* untuk citra asli (*oImg*) dan *wImgP* untuk citra ber-*watermark* (*wImg*). Ukuran yang akan digunakan oleh proses-proses selanjutnya diambil dengan menggunakan fungsi *size* dari *matlab*. *Pseudocode* untuk menyamakan ukuran kedua gambar adalah :

```
10  %make both image size equal
11  soImg = size(oImg);
12  [wImgRows wImgCols] = size(wImg);
13  if (wImgRows > soImg(1))
14      wImgP = wImg;
15      oImgP = zeroPad(oImg, size(wImgP));
16  elseif (wImgRows < soImg(1))
17      wImgP = zeroPad(wImg, soImg);
18      oImgP = oImg;
19  Else
20      wImgP = wImg;
21      oImgP = oImg;
22  End
23  soMag = size(oImgP);
24  [wImgRows wImgCols] = size(wImgP);
```

Gambar 4.16 Kode program menyamakan ukuran kedua gambar dalam ekstraksi

4.4.3 Menghitung *Magnitude* Fourier Kedua Citra

Pada langkah ini dihitung *Discrete Fourier Transform* dari citra hasil proses sebelumnya (*oImgP* dan *wImgP*) dengan menggunakan fungsi *fft2* dari *matlab* lalu digeser sehingga komponen berfrekuensi nol berada ditengah dengan fungsi *fftshift* dari *matlab*. Kemudian diambil *magnitude* untuk masing-masing citra (*oMag* dan *wMag*).*Pseudocode* program menghitung *magnitude* Fourier kedua citra tampak pada gambar 4.17.

```
26  %DFT
27  oMag = abs(fftshift(fft2(oImgP)));
28  wMag = abs(fftshift(fft2(wImgP)));
```

Gambar 4.17 Kode program menghitung *magnitude* Fourier kedua citra dalam ekstraksi *watermark*

4.4.4 *High-pass Filter* Kedua *Magnitude*

Pada langkah ini terjadi *filtering* terhadap kedua *magnitude* untuk membuang komponen-komponen berfrekuensi rendah. Hal ini dilakukan agar nilai dari komponen-komponen berfrekuensi rendah (yang biasanya besar) tidak mempengaruhi hasil dari *phase correlation*, sehingga didapat hasil yang lebih bersih. Fungsi *hipass_filter* membentuk *array* yang jika dikalikan ke *magnitude*, akan memfilter bagian tengah yang membentuk lingkaran dari *magnitude* tersebut. Diperlukan satu filter untuk masing-masing *magnitude*. Persamaan untuk *fitering* ini adalah persamaan 3.3 hingga 3.6, diimplementasikan dalam fungsi *hipass_filter* dari baris ke-7 hingga baris ke-11 (gambar 4.18). *Pseudocode* program *high-pass filter* kedua citra ini tampak pada gambar 4.18.

```
30  %high pass filter
31  filter = hipass_filter(soMag(1),soMag(2));
32  oMag = filter.*oMag;
33  filter = hipass_filter(wImgRows,wImgCols);
34  wMag = filter.*wMag;
```

Gambar 4.18 *high-pass filter* kedua *magnitude* dalam ekstraksi *watermark*

```
1  function H = hipass_filter(ht,wd)
2  % hi-pass filter function
3  % ...designed for use with Fourier-Mellin stuff
4  res_ht = 1/(ht-1);
5  res_wd = 1/(wd-1);
6
7  eta = cos(pi*(-0.5:res_ht:0.5));
8  neta = cos(pi*(-0.5:res_wd:0.5));
9  X = eta'*neta;
10
11 H=(1.0-X).*(2.0-X);
12 %figure,mesh(H),title('The high-pass filter')
```

Gambar 4.19 fungsi *hipass_filter*

4.4.5 *Log-Polar Mapping* dari kedua *Magnitude*

Pada langkah ini dilakukan perhitungan *Log-Polar Mapping* untuk masing-masig *magnitude* dengan resolusi *PCResolution* dari masukkan. Perhitungan ini dilakukan dengan menggunakan fungsi *rlpm* yang merupakan implementasi dari persamaan 2.11 hingga 2.15. *Pseudocode* program *Log-Polar Mapping* dari kedua *Magnitude* ini tampak pada gambar 4.20.

```
36 %LPM
37 oLpm = rlpm(oMag,PCResolution);
38 wLpm = rlpm(wMag,PCResolution);
```

Gambar 4.20 Kode program *Log-Polar Mapping* dari kedua *Magnitude* dalam ekstraksi *watermark*

```
function polarMap = rlpm(mat,res)

[rows, cols] = size(mat);

midx = cols/2;
midy = rows/2;

rmin  = 1;
rmax = sqrt(((res/2)^2)+((res/2)^2));

numr = res;
numa = res;

rrange = linspace(log(rmin), log(rmax), numr);
arange = linspace(0, 2*pi*(numa-1)/numa, numa);

rTab = exp(rrange);
sinTab = sin(arange);
cosTab = cos(arange);

polarMap = zeros(numr, numa);

for r = 1:numr
   for a = 1:numa

      x = midx + rTab(r)* cosTab(a);
      y = midy + rTab(r)* sinTab(a);

      imVal = biLinInterp(mat, x, y);

      polarMap(r, a) = imVal;
   End
end;
```

Gambar 4.21 Kode program fungsi *rlpm*

4.4.6 *Phase Correlation* kedua hasil *Log-Polar Mapping*

Pada langkah ini dihitung *Phase Correlation* antara hasil *Log-Polar Mapping* dari *magnitude* citra asli dengan hasil *Log-Polar Mapping* dari *magnitude* citra ber-*watermark*. Perhitungan ini dilakukan dalam fungsi

PhaseCorr yang merupakan implementasi dari persamaan 2.23, seperti yang terlihat pada baris ke-4 dari fungsi *PhaseCorr* (gambar 22). Hasil dari *Phase Correlation* kemudian diambil nilai-nilai puncaknya dengan menggunakan fungsi *sort* dari matlab. *Pseudocode* program *Phase Correlation* kedua hasil *Log-Polar Mapping* ini tampak pada gambar 4.22.

```
40  %calculate displacement & rectify watermark position
41  d10 = PhaseCorr(oLpm,wLpm);
42  peaksVal = sort(d10(:));
```

Gambar 4.22 Kode program *Phase Correlation* kedua hasil *Log-Polar Mapping* dalam ekstraksi *watermark*

```
1  function d10 = PhaseCorr(oin1,oin2)
2  in1 = fft2(oin1);
3  in2 = fft2(oin2);
4  c10 = (in1.*conj(in2))./(abs(in1).*abs(conj(in2)));
5  d10 = real(ifft2(c10));
```

Gambar 4.23 Kode program fungsi *PhaseCorr*

4.4.7 Evaluasi Puncak-Puncak

Pada langkah ini dimulai *looping* dalam bentuk *while* selama *idx* tidak sama dengan 200 dan *lastPeak* lebih kecil dari 0.3. Nilai batas *idx* dan *lastPeak* ini didapat dari uji coba. Bagian awal dari *Looping* ini adalah pencarian posisi puncak dengan menggunakan fungsi *find* dari *matlab*. Kemudian dilakukan *increment* terhadap *idx* sebanyak 1. *Pseudocode* program Evaluasi puncak-puncak ini tampak pada gambar 4.24.

```
44  while  (idx ~= 200)&(lastPeak < 0.3)
45      [dPeaksS dPeaksR] = find(d10 == peaksVal(length(peaksVal)- idx))
46      Idx = idx + 1;
```

Gambar 4.24 Kode program evaluasi puncak-puncak dalam ekstraksi *watermark*

4.4.8 Penyesuaian Terhadap Rotasi

Pada langkah ini dilakukan pencarian parameter rotasi dengan persamaan 3.7 seperti yang terlihat pada baris ke-50. Penyesuaian terhadap rotasi ini dilakukan dengan menggunakan fungsi *imrotate* dari *matlab* dengan interpolasi *bilinear*. Penyesuaian dilakukan untuk rotasi lebih kecil atau sama dari 180 derajat dan lebih besar dari 180 derajat. *Pseudocode* program Penyesuaian terhadap rotasi ini tampak pada gambar 4.25.

```
48      %calculating Rotation parameter
49      %changed
50      dtheta = (((dPeaksR(1)-1)/PCResolution)*360)
51
52      %synchronize Rotated watermarked image
53      wImg1 = imrotate(wImgP,-dtheta,'bilinear');
54      wImg2 = imrotate(wImgP,-(dtheta+180),'bilinear');
```

Gambar 4.25 Kode program penyesuaian terhadap rotasi dalam ekstraksi *watermark*

4.4.9 Penyesuaian Terhadap Skala

Pada langkah ini dilakukan perhitungan parameter skala dengan persamaan 3.8 dan 3.9, seperti yang terlihat pada baris ke-62 dan baris ke-66 dari *Pseudocode*. Hasil dari perhitungan tersebut adalah nilai skala yang terjadi pada citra. Nilai tersebut sering hanya tepat satu angka dibelakang koma, oleh karena itu nilai tersebut dibulatkan untuk mendapatkan hasil yang akurat. Selain itu diperlukan penambahan konstanta yang berbeda untuk perubahan < 1 dan > 1 seperti yang tampak pada baris ke-63 dan baris ke-67. perubahan <1 ditunjukkan dengan nilai pergeseran kolom yang > ½ resolusi LPM, dan perubahan >1 ditunjukkan dengan kebalikannya (baris ke-61). Terkadang parameter skala yang dihasilkan terlalu besar, sehingga perlu diberi *threshold* agar tidak terjadi out-of-memory. Nilai *threshold* ditentukan sebesar 3,5. Nilai yang digunakan untuk penyesuaian terhadap skala ini adalah 1/hasil perhitungan. Fungsi yang digunakan untuk penyesuaian terhadap skala ini adalah fungsi *imresize* dengan metode

interpolasi *bilinear*. *Pseudocode* program Penyesuaian terhadap skala ini tampak pada gambar 4.26.

```
%calculating Scaling parameter
if (dPeaksS(1) ~= 1)
    %changed

    rmax = sqrt(((wImgRows/2)^2)+((wImgCols/2)^2));
    if (dPeaksS(1) > PCResolution/2)
        oscale = exp(((log(rmax))*(dPeaksS(1)- ...
                 PCResolution))/PCResolution)
        dscale = (10/round(oscale*10)) + (2/size(wImg1,1))
    Else
        oscale = exp(((log(rmax))*(dPeaksS(1)-1))/PCResolution)
        dscale = 10/floor(oscale*10) + (1/size(wImg1,1))
    End
    if (dscale < 3.5)
        %synchronize Scaled watermarked image
        wImg1= imresize(wImg1,dscale,'bilinear');
        wImg2 = imresize(wImg2,dscale,'bilinear');
    End
End
```

Gambar 4.26 Kode program penyesuaian terhadap skala dalam ekstraksi *watermark*

4.4.10 *Phase Correlation* antara Hasil Penyesuaian dengan Citra Asli

Pada langkah ini dihitung *Phase Correlation* yang dilakukan untuk mendapatkan pergeseran yang terjadi akibat translasi. Translasi adalah pergeseran di dalam domain kartesian, oleh sebab itu perlu membandingkan citra yang mengalami pergeseran dalam domain kartesian. Pembatasan ukuran dari citra ber-*watermark* diperlukan karena apabila ukuran citra ber-*watermark* lebih kecil dari citra asli maka parameter-parameter sebelumnya ada yang salah sehingga tidak perlu dilanjutkan lagi. Selain itu ditambahkan juga *threshold* untuk skala. *Pseudocode* program penyesuaian terhadap skala ini tampak pada gambar 4.27.

```
78      szW = size(wImg1);
79      if (szW(1) >= soImg(1))&(szW(2) >= soImg(2))&(dscale < 3.5)
80          oPadded = zeroPad(oImg,szW);
81          d11 = PhaseCorr(oPadded,wImg1);
82          d12 = PhaseCorr(oPadded,wImg2);
```

Gambar 4.27 Kode program *Phase Correlation* antara hasil penyesuaian dengan citra asli dalam ekstraksi *watermark*

4.4.11 Penyesuaian Terhadap Translasi

Pada langkah ini diawali dengan perhitungan parameter translasi. Perhitungan ini dimulai dengan pengambilan puncak tertinggi di antara dua hasil *phase correlation* dari proses sebelumnya.Pengambilan puncak tertinggi ini dilakukan dengan fungsi *max* dari *matlab.* Posisi dari puncak (yang dicari dengan fungsi *find* dari *matlab*) bisa diartikan pergeseran sebesar posisi puncak, untuk nilai-nilai posisi yang lebih kecil dari ½ ukuran citra hasil penyesuaian sebelumnya pergeseran yang terjadi sebesar selisih dari posisi puncak dengan ukuran citra hasil penyesuaian sebelumnya (terlihat pada baris ke-104 dan ke-112). Operasi penyesuaian dari translasi ini dilakukan dengan melakukan translasi balik sebesar parameter yang didapatkan. Kemudian dengan asumsi citra telah selesai disesuaikan, citra di bersihkan dari efek penambahan nol (*zeropad*) dangan fungsi *unPad.* Fungsi ini mengambil bagian tengah dari citra seukuran citra asli. *Pseudocode* program penyesuaian terhadap translasi ini tampak pada gambar 4.28.

```
83          peakVal1 = max(max(d11));
84          peakVal2 = max(max(d12));
85
86          if (peakVal1 >= peakVal2)
87              wImgT = wImg1; %using normal rotation
88              [peaku peakl] = find(d11 == peakVal1)
89              peakVal3 = peakVal1;
90          else
91              wImgT = wImg2; %using 180 deg added rotation
92              [peaku peakl] = find(d12 == peakVal2)
93              peakVal3 = peakVal2;
94          end
95
96          if (peaku ~= 1)|(peakl ~= 1)
97              peaku = peaku - 1;
98              peakl = peakl - 1;
99              %translate back
100             if (peaku > (szW(1)/2))
101                 peaku = szW(1) - peaku; %translate back up
102                 wImgT = [wImgT(peaku+1:szW(1),1:szW(2));wImgT(1:peaku,...
103                         1:szW(2))];
104             else    %translate back down
105                 wImgT = [wImgT(szW(1)-peaku+1:szW(1),1:szW(2)); ...
106                         wImgT(1:szW(1)-peaku,1:szW(2))];
107             end
108             if (peakl > (szW(2)/2)) %translate back left
109                 peakl = szW(2) - peakl;
110                 wImgT = [wImgT(1:szW(1),peakl+1:szW(2)) ...
111                         wImgT(1:szW(1),1:peakl)];
112             else    %translate back right
113                 wImgT = [wImgT(1:szW(1),szW(2)-peakl+1:szW(2)) ...
114                         wImgT(1:szW(1),1:szW(2)-peakl)];
115             end
116
117         end
118
119         wImg1= unPad(wImgT,soImg);
```

Gambar 4.28 Kode program penyesuaian terhadap translasi dalam ekstraksi *watermark*

4.4.12 Ekstraksi *Watermark* dan Dikorelasi dengan *Watermark* Asli

Pada langkah ini ekstraksi dilakukan terhadap *magnitude* citra yang telah disesuaikan. Pembuatan *magnitude* dilakukan dengan menggunakan fungsi *fft2*, *fftshift* dan *abs* dari *matlab* sama dengan yang ada pada langkah *Discrete Fourier*

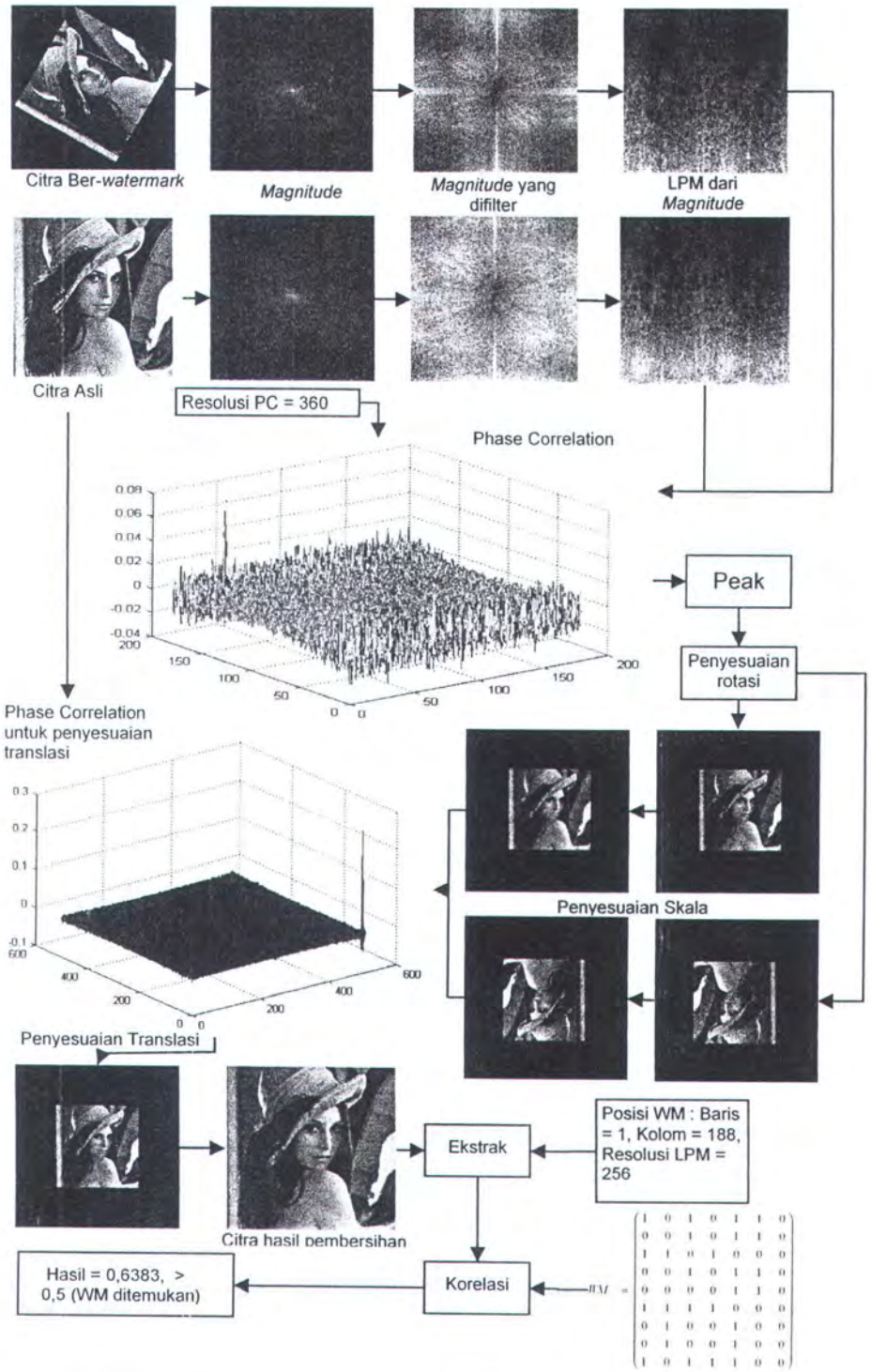
Transform di atas. Ekstraksi *watermark* dilakukan dengan menggunakan fungsi *applpm*, fungsi ini adalah kebalikan fungsi *appilpm* yang ada di operasi *embedding*, namun tetap menggunakan persamaan yang sama persamaan 2.11 hingga 2.15 dengan kata lain tetap menggunakan koordinat di domain *Log-Polar Mapping* sebagai acuan. Dalam fungsi *applpm* ini tidak mempedulikan simetrisitas. Setelah mengekstraksi *watermark* diambil korelasi dari *watermark* hasil ekstraksi dengan *watermark* asli, rumus dari perhitungan korelasi ini adalah persamaan 3.10, yang memiliki kesesuaian dengan fungsi *corr2* dari *matlab*. *Pseudocode* program ekstraksi *watermark* dan dikorelasi dengan *watermark* asli ini tampak pada gambar 4.29.

```
120         %creating magnitude of synchronized image
121         wDft2 = fftshift(fft2(wImg1));
122         wMag = abs(wDft2);
123
124         sizeWM = size(WM);
125         WM = WM*wmStrength;
126
127         %nonexhaustive search
128         %extracting
129         eWM = applpm(wMag,sizeWM(1),sizeWM(2),wmPositionRows,...
130               wmPositionCols,lpmResolution,wmStrength);
131         Result = corr2(WM,eWM);
```

Gambar 4.29 Kode program ekstraksi *watermark* dan dikorelasi dengan *watermark* asli dalam ekstraksi *watermark*

4.4.13 Menyimpan Hasil Ekstraksi Tertinggi

Pada langkah ini nilai tertinggi dari puncak hasil *phase correlation* translasi dipilih yang tertinggi untuk disimpan sebagai hasil terbaik dari evaluasi puncak-puncak yang ada. Kemudian kembali ke langkah ke-7 (dilakukan pada baris ke-138). Setelah keseluruhan puncak selesai dievaluasi, nilai *bestResult, lastPeak* dan *lastIdx* akan dikembalikan sebagai luaran program ekstraksi. Nilai *bestResult* merupakan luaran utama dari program. *Pseudocode* program menyimpan hasil ekstraksi tertinggi ini tampak pada gambar 4.30.



Gambar 4.31 Penggambaran langkah-langkah Ekstraksi *Watermark* dengan *Log-Polar Mapping* dan *Phase Correlation*

4.5 PENGHITUNG *PEAK SIGNAL TO NOISE RATIO* (PSNR)

Proses penghitung ini berupa fungsi bernama psnr yang merupakan implementasi persamaan 3.11, baris ke-12 merupakan bagian pembilang dari pembagian dalam persamaan diatas. Sedangkan baris ke-14 merupakan bagian penyebut dari pembagian dalam persamaan diatas. Baris ke-16 merupakan operasi pembagian dan log untuk mendapatkan hasil *PSNR*. *Pseudocode* program penghitung *Peak Signal to Noise Ratio* (PSNR) ini tampak pada gambar 4.32.

```
1  function [A] = psnr(image,image_prime,M,N)
2
3      % convert to doubles
4      image=double(image);
5      image_prime=double(image_prime);
7      % avoid divide by zero nastiness
8      if ((sum(sum(image-image_prime))) == 0)
9          error('Input vectors must not be identical')
10     else
11         % calculate numerator
12         psnr_num=M*N*max(max(image.^2));
13         % calculate denominator
14         psnr_den=sum(sum((image-image_prime).^2));
15         % calculate PSNR
16         A=log10(psnr_num/psnr_den)*10;
17   end
19 return
```

Gambar 4.32 Kode program penghitung *PSNR*

4.6 PEMBUATAN TABEL LOKASI WATERMARK

Terdapat empat proses pada Pembuatan tabel lokasi *watermark*, yaitu:

1. perhitungan tiap-tiap pasangan kekuatan *watermark* dan posisi kolom dari *watermark*
2. *embedding watermark*
3. perhitungan *PSNR*
4. ekstraksi *watermark*

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

4.6.1 Perhitungan Tiap-tiap Pasangan Kekuatan *Watermark* dan Posisi Kolom dari *Watermark*

Pada langkah ini diawali proses *nested loop* dua variabel dengan menggunakan *for*. Variabel x di-*loop* dari nilai kekuatan *watermark* minimum hingga maksimum, sedangkan variabel y di-*loop* dari nilai posisi kolom *watermark* minimum hingga maksimum. *Pseudocode* untuk proses inisialisasi tampak pada gambar 4.33.

```
5  for x = minStr:maxStr
6      for y = minY:maxY
```

Gambar 4.33 Kode program perhitungan tiap-tiap pasangan kekuatan *watermark* dan posisi kolom dari *watermark* dalam pembuatan tabel lokasi *watermark*

4.6.2 *Embedding Watermark*

Pada langkah ini dilakukan *embedding watermark* terhadap citra asli pada posisi (1,*Y*) dengan kekuatan *X*. Resolusi *Log-Polar Mapping* dan *watermark* ditentukan dari masukkan. Proses ini dilakukan dengan fungsi *embed*. *Pseudocode* untuk *embedding watermark* adalah sebagai berikut:

```
7          wImg = embed(oImg,WM,1,y,x,256);
```

Gambar 4.34 Kode program *embedding watermark* dalam pembuatan tabel lokasi *watermark*

4.6.3 Perhitungan *PSNR*

Pada langkah iniPada langkah ini dihitung nilai *Peak Signal to Noise Ratio (PSNR)* dari citra asli dengan citra ber-*watermark* hasil *embedding* sebelumnya (*wImg*). Hasil dari perhitungan ini dimasukkan ke dalam tabel *PResult*. *Pseudocode* perhitungan *PSNR* sebagai berikut:

```
8          PResult(y-(minY-1),((x-(minStr-1))*2)-1) = psnr(wImg,oImg,256,256);
```

Gambar 4.35 Kode program perhitungan *PSNR* dalam pembuatan tabel lokasi *watermark*

4.6.4 Ekstraksi *Watermark*

Pada langkah ini dilakukan ekstraksi *watermark* dari citra ber-*watermark* hasil proses sebelumnya (*wImg*). Posisi *watermark* adalah (1,*Y*) dengan kekuatan *watermark X*. Resolusi *Log-Polar Mapping* dan *watermark* ditentukan dari masukkan, sedangkan resolusi *Phase Correlation* ditentukan 10 untuk mempercepat proses ekstraksi dari sebuah citra ber-*watermark* yang normal. Hasil ekstraksi ini dimasukkan ke dalam tabel *PResult.* Tabel *PResult* ini menjadi luaran dari sistem. Setelah proses ini maka dilanjutkan ke pasangan kekuatan *watermark* dan posisi kolom *watermark* selanjutnya. *Pseudocode* ekstraksi *watermark* ini tampak pada gambar 4.35.

```
 9          wImg = real(wImg);
10          wImg = im2uint8(wImg);
11          PResult(y-(minY-1),(x-(minStr-1))*2)=extractNormal(im2double(...
12                                                  wImg),oImg,WM,1,y,1,256,10);
13      end
14  end
```

Gambar 4.36 Kode program ekstraksi *watermark* dalam pembuatan tabel lokasi *watermark*

4.7 ANTARMUKA

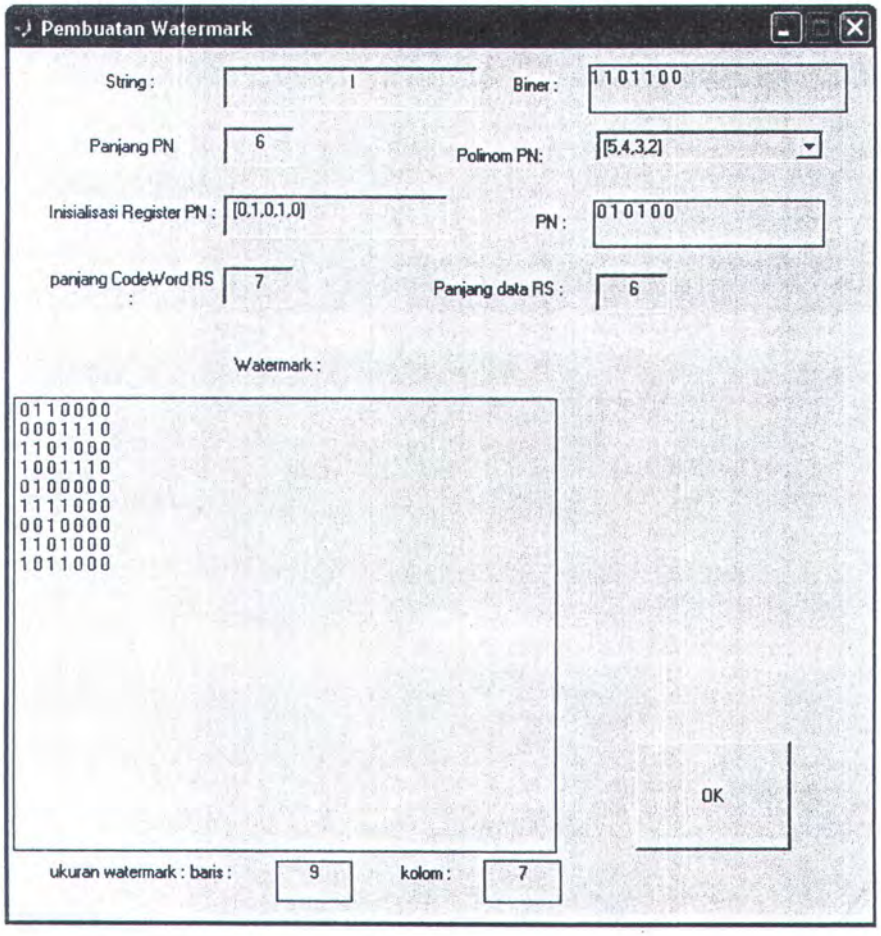
Ada 4 antarmuka, masing-masing untuk :

1. Pembuatan *Watermark*
2. *Embedding Watermark*
3. Ekstraksi *Watermark*
4. Pencarian Lokasi *Watermark*

Sedangkan perhitungan PSNR dimasukkan ke dalam antarmuka tersebut sesuai kebutuhan.

4.7.1 Pembuatan *Watermark*

Pada Pembuatan *Watermark* ini digunakan sebuah *figure* seperti yang digambarkan pada gambar 4.37



Gambar 4.37 *figure* antarmuka Pembuatan *Watermark*

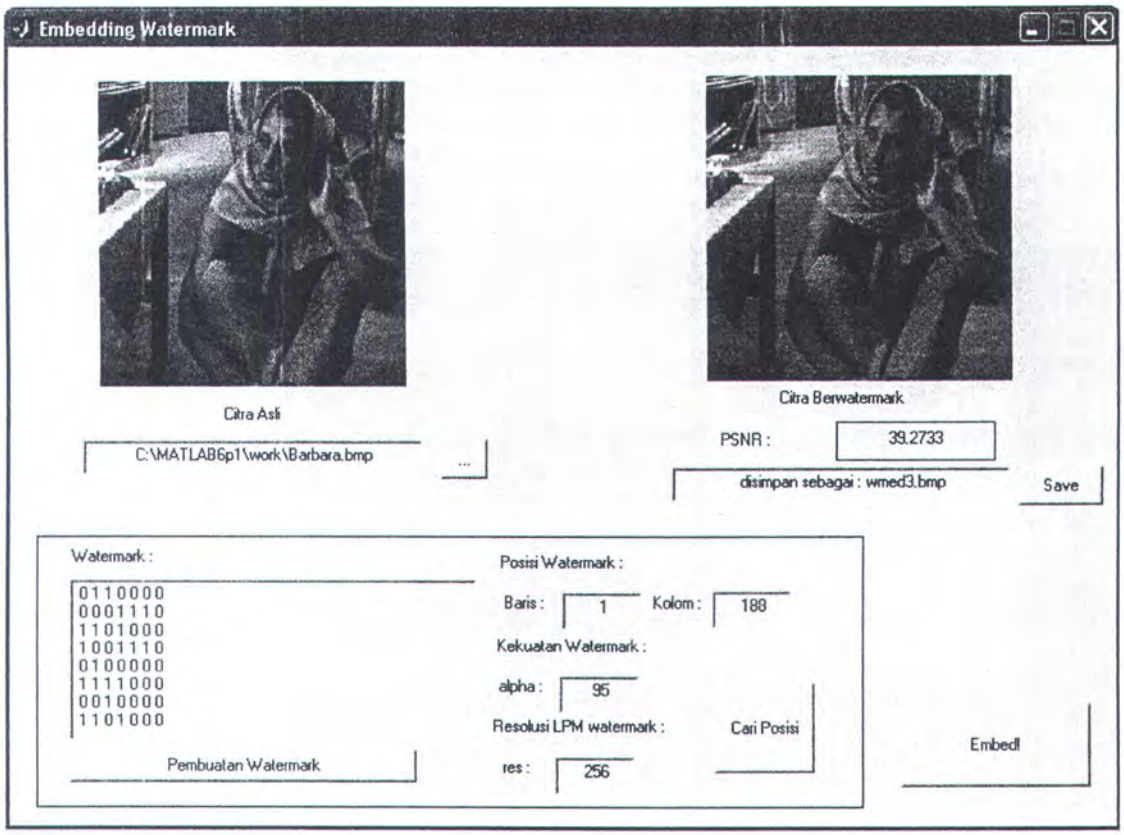
Bagian-bagian dari *figure* tersebut adalah :

1. "String" merupakan masukkan *string* yang digunakan sebagai data *watermark.*
2. "Biner" merupakan tampilan bentuk biner dari *string.*
3. "Panjang PN" merupakan masukkan panjang dari *Pseudonoise* (PN) yang akan dihasilkan oleh generator PN.
4. "Polinom PN" merupakan masukkan polinom untuk generator PN.

5. "Inisialisasi *Register* PN" merupakan masukkan nilai-nilai inisialisasi untuk *register* dalam generator PN.
6. "panjang Codeword RS" merupakan masukkan panjang codeword (n) yang akan digunakan dalam ECC Reed Solomon.
7. "Panjang data RS " merupakan masukkan panjang data (k) yang akan digunakan dalam ECC Reed Solomon.
8. "Watermark" merupakan tampilan *watermark* hasil Pembuatan *Watermark.*
9. "Ukuran Watermark: Baris" merupakan tampilan ukuran baris dari *watermark.*
10. "Kolom" merupakan tampilan ukuran kolom dari *watermark.*
11. tombol "OK" merupakan tombol untuk menandakan pembuatan *watermark* selesai dan kembali ke *figure* sebelumnya jika dipanggil oleh *figure* lain atau keluar.

4.7.2 *Embedding Watermark*

Pada *Embedding Watermark* digunakan *figure* seperti yang tampak pada gambar 4.38 :



Gambar 4.38 *figure* antarmuka *Embedding Watermark*

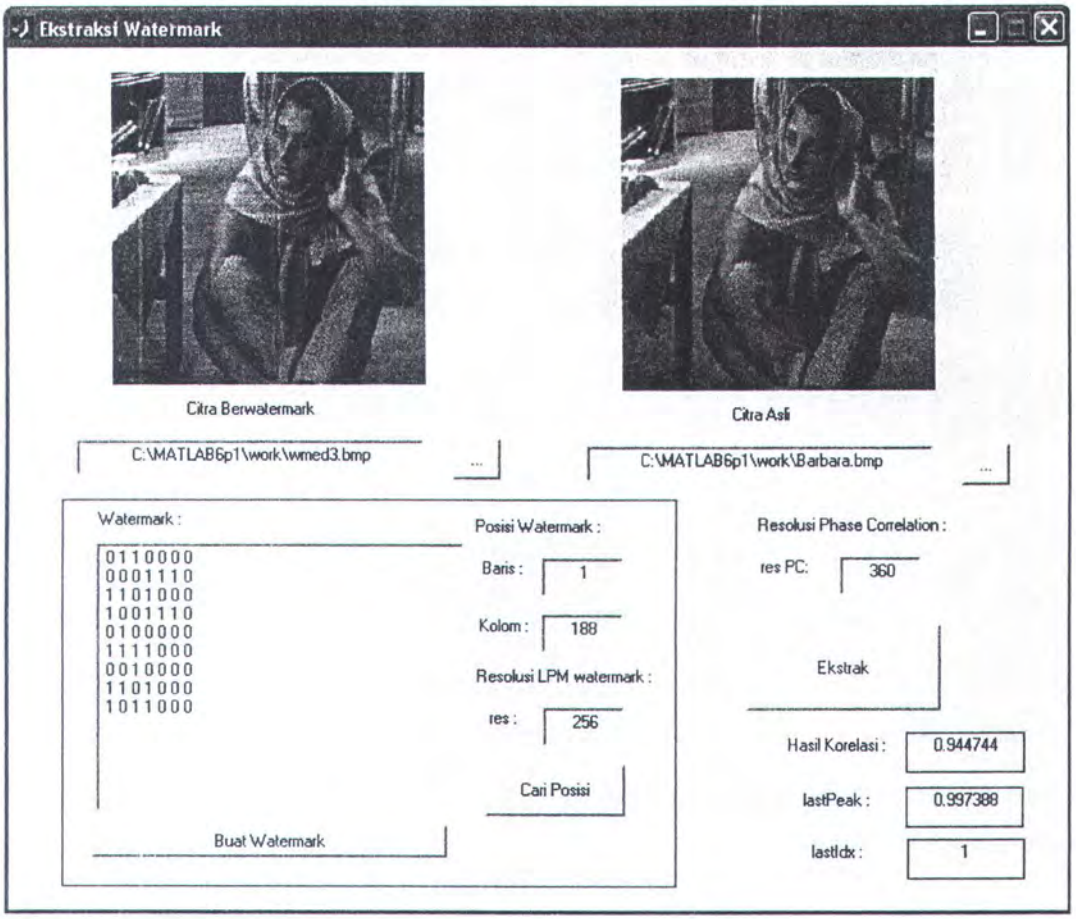
Bagian-bagian dari *figure* tersebut adalah :

1. "Citra Asli" merupakan masukkan citra asli yang akan ditambah *watermark*. Tombol "..." merupakan tombol pemanggil kotak untuk *browse* file citra, sedangkan nama dari file ditampilkan di kotak *edit* sebelah kiri tombol yang juga sebagai tempat pengisian nama file secara manual.
2. "Watermark" merupakan masukkan *watermark* yang akan ditambahkan ke dalam citra. Bisa dibuat dengan program "Pembuatan Watermark"
3. "Pembuatan Watermark" berupakan tombol untuk memanggil program "Pembuatan *Watermark*"
4. "Baris" dan "Kolom" merupakan masukkan posisi *watermark* dalam domain LPM.
5. "Alpha" merupakan masukkan kekuatan *watermark*.
6. "res" merupakan masukkan resolusi dari domain LPM tempat *watermark*.

7. "Cari Posisi" merupakan tombol untuk memanggil program "Pencarian Posisi *Watermark*".
8. "Embed!" merupakan tombol untuk memulai proses *embedding watermark*.
9. "Citra berwatermark" merupakan tampilan hasil dari *embedding* berupa citra ber-*watermark*.
10. "PSNR" merupakan tampilan nilai PSNR (dalam dB) antara citra asli dan citra ber-*watermark*. Merupakan hasil dari program "Perhitungan PSNR"
11. "Save" merupakan tombol untuk menyimpan citra ber-*watermark* ke dalam file. Tombol ini membuka kotak *browse* untuk penyimpanan file. Nama dari file akan ditampilkan di kotak sebelah kiri dari tombol.

4.7.3 Ekstraksi *Watermark*

Pada Ekstraksi *Watermark* digunakan figure seperti yang tampak pada gambar 4.39 :



Gambar 4.39 *figure* antarmuka Ekstraksi *Watermark*

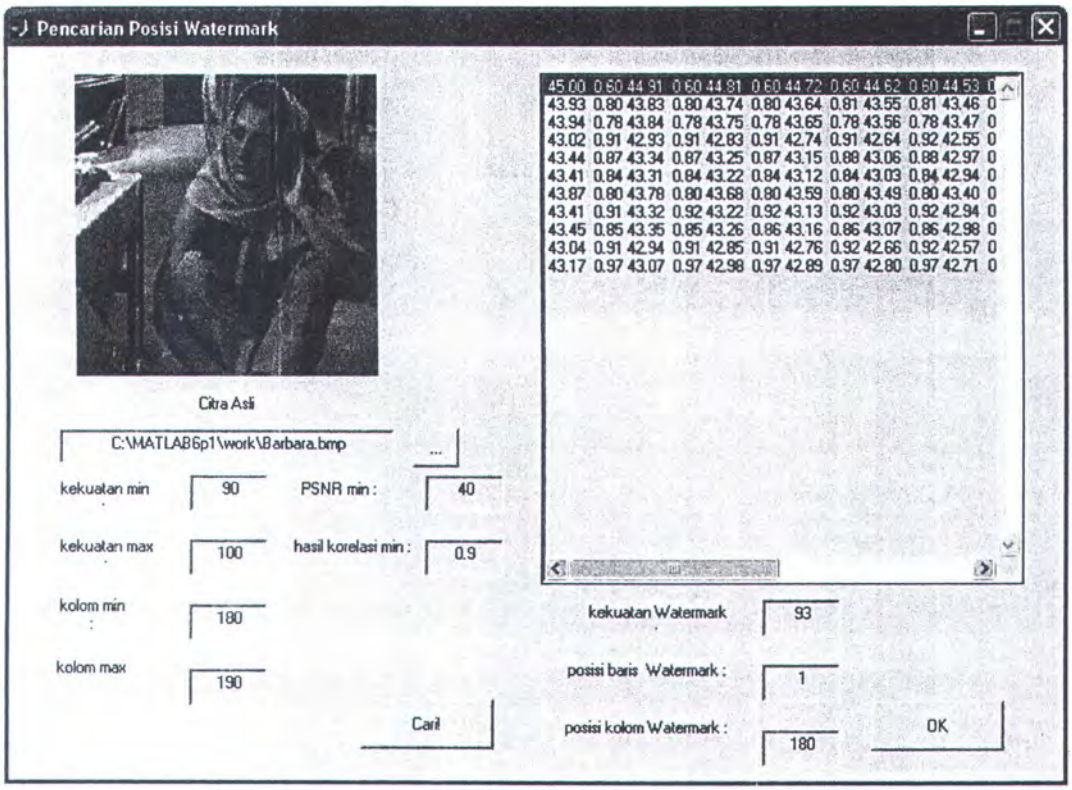
Bagian-bagian dari *figure* tersebut adalah :

1. "Citra Berwatermark" merupakan masukkan citra ber-*watermark* hasil *embedding watermark*. Tombol "..." merupakan tombol pemanggil kotak untuk *browse* file citra, sedangkan nama dari file ditampilkan di kotak *edit* sebelah kiri tombol yang juga sebagai tempat pengisian nama file secara manual.
2. "Citra Asli" merupakan masukkan citra asli yang akan digunakan dalam proses ekstraksi. Tombol "..." merupakan tombol pemanggil kotak untuk *browse* file citra, sedangkan nama dari file ditampilkan di kotak *edit* sebelah kiri tombol yang juga sebagai tempat pengisian nama file secara manual.

3. "Watermark" merupakan masukkan *watermark* asli yang akan dikorelasi dengan *watermark* hasil ekstraksi. Bisa dibuat dengan program "Pembuatan Watermark"
4. "Pembuatan Watermark" berupakan tombol untuk memanggil program "Pembuatan *Watermark*"
5. "Baris" dan "Kolom" merupakan masukkan posisi *watermark* dalam domain LPM.
6. "res" merupakan masukkan resolusi dari domain LPM tempat *watermark.*
7. "Cari Posisi" merupakan tombol untuk memanggil program "Pencarian Posisi *Watermark*".
8. "res PC" merupakan masukkan resolusi dari LPM untuk *Phase Correlation.*
9. "Ekstrak" merupakan tombol untuk memulai proses ekstraksi *watermark.*
10. "Hasil Korelasi" merupakan tampilan hasil korelasi antara *watermark* asli dengan *watermark* hasil ekstraksi.
11. "lastPeak" dan "lastIdx" merupakan tampilan hasil sampingan ekstraksi *lastPeak* dan *lastIdx.*

4.7.4 Pencarian Lokasi *Watermark*

Pada Pencarian Lokasi *Watermark* digunakan figure seperti yang tampak pada gambar 4.40 :



Gambar 4.40 *figure* antarmuka Pencarian Lokasi *Watermark*

Bagian-bagian dari *figure* tersebut adalah :

1. "Citra Asli" merupakan masukkan citra yang akan dicari posisi *watermark*-nya. Tombol "..." merupakan tombol pemanggil kotak untuk *browse* file citra, sedangkan nama dari file ditampilkan di kotak *edit* sebelah kiri tombol yang juga sebagai tempat pengisian nama file secara manual.
2. "kekuatan min" merupakan masukkan kekuatan minimal dari *watermark* yang menjadi kolom terkiri dari tabel alpha-korelasi yang tampak pada sebelah kanan tampilan citra.
3. "kekuatan max" merupakan masukkan kekuatan maksimal dari *watermark* yang menjadi kolom terkanan dari tabel alpha-korelasi.
4. "kolom min" merupakan masukkan posisi kolom minimal dari *watermark* yang menjadi baris paling atas dari tabel alpha-korelasi.
5. "kolom max" merupakan masukkan posisi kolom maksimal dari *watermark* yang menjadi baris paling bawah dari tabel alpha-korelasi.

6. "PSNR min" merupakan masukkan nilai PSNR minimum yang menjadi kriteria pencarian posisi *watermark*.
7. "hasil korelasi minimum" merupakan masukkan nilai hasil korelasi antara *watermark* asli dengan *watermark* hasil ekstraksi minimum yang menjadi kriteria pencarian posisi *watermark*.
8. "Cari" merupakan tombol untuk memulai pencarian posisi *watermark*. Dan hasilnya ditampilkan dalam tabel alpha-korelasi.
9. "Kekuatan watermark", "posisi baris watermark", dan "posisi kolom watermark" merupakan hasil pencarian posisi *watermark*.
10. "OK" merupakan merupakan tombol untuk menandakan pencarian posisi *watermark* selesai dan kembali ke *figure* sebelumnya jika dipanggil oleh *figure* lain atau keluar.

BAB V

UJI COBA DAN EVALUASI

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dijelaskan mengenai uji coba dan evaluasi yang telah dilakukan. Pembahasan yang dikemukakan meliputi lingkungan uji coba, dan uji coba dengan menggunakan citra-citra dan *watermark-watermark* yang berbeda-beda.

5.1 LINGKUNGAN UJI COBA

Pada bagian ini akan dijelaskan mengenai lingkungan pengujian aplikasi, yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pembangunan aplikasi *watermarking* ini tampak pada tabel berikut:

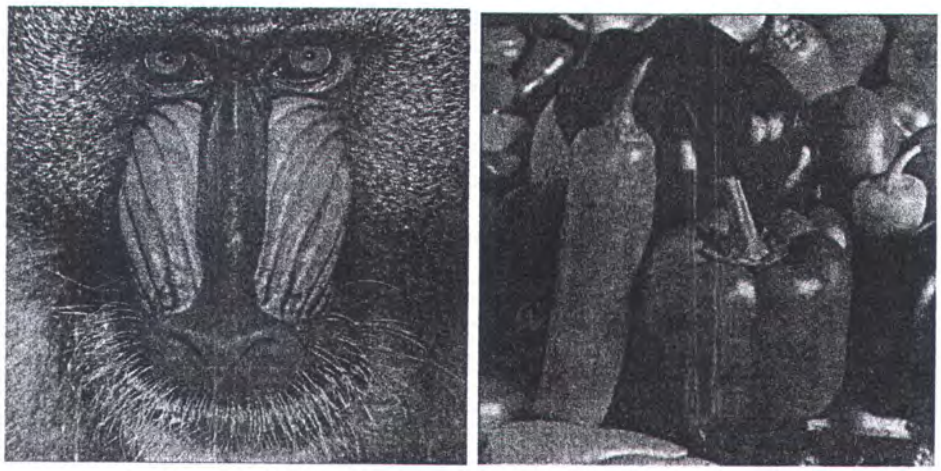
Tabel 5.1 Lingkungan pengujian aplikasi

Perangkat Keras	Prosesor : AMD Athlon 800 MHz Memori : 256 MB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pembangun : Matlab 6.1.0

5.2 DATA UJI COBA

Citra-citra yang menjadi bahan uji coba berasal dari berbagai sumber dan secara umum memiliki karakteristik kedalaman warna 8 bit, ukuran panjang dan lebar sama, dan bertipe file bmp. Citra-citra yang digunakan dalam uji coba adalah Barbara.bmp dengan ukuran 256x256, Bikes.bmp dengan ukuran 256x256, mandrill.bmp dengan ukuran 512x512, peppers.bmp dengan ukuran 384x384, dan lena_org.bmp dengan ukuran 256x256. Pada uji coba ini dievaluasi ketahanan *watermark* terhadap serangan-serangan rotasi, skala, translasi, *cropping* dan kompresi jpg. Juga dievaluasi ketahanan pada berbagai macam *watermark*. Selain ketahanan *watermark* juga dilakukan uji coba untuk mencari batas-batas *idx* dan *lastPeak* yang digunakan dalam proses ekstraksi *watermark* dan uji coba untuk mencari lokasi *watermark*.

(a) (b)



(c) (d)

Gambar 5.1 Citra-citra yang dipakai dalam uji coba ketahanan RST:(a) Barbara.bmp, (b)Bikes.bmp, (c)mandrill.bmp, (d)peppers.bmp

Tabel 5.2 Tabel posisi dan kekuatan *watermark* (*WM*)

citra	Resolusi citra	posisi baris WM	posisi kolom WM	kekuatan WM	resolusi LPM
Barbara.bmp	256x256	1	185	79	256x256
Bikes.bmp	256x256	1	197	90	256x256
mandrill.bmp	512x512	1	210	150	256x256
peppers.bmp	384x384	1	285	140	384x384

Terdapat tujuh jenis uji coba ketahanan, yaitu:

- Uji coba dengan citra ber-*watermark* tidak berubah,
- Uji coba dengan citra ber-*watermark* mengalami rotasi,
- Uji coba dengan citra ber-*watermark* mengalami skala,
- Uji coba dengan citra ber-*watermark* mengalami rotasi dan skala,
- Uji coba dengan citra ber-*watermark* mengalami skala dan translasi
- Uji coba dengan citra ber-*watermark* mengalami rotasi, skala dan translasi
- Uji coba dengan citra ber-*watermark* mengalami kompresi jpg.

Uji coba yang dilakukan dengan ketujuh kriteria diatas diujicobakan pada kelima citra. *Watermark* dibentuk dengan program "pembuatan *watermark*" dengan menggunakan huruf 'I' sebagai data *watermark* dan *pseudorandom noise* sepanjang 6 bit dengan polinominal [5,4,2,1] dan inisialisasi *register* [1,0,1,1]. Kemudian ditambahkan kode Reed Solomon dengan panjang *codeword* 7 dan panjang data 6. Hasilnya di bentuk kembali menjadi matriks 9x7.

Watermark hasil :

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Selain itu dilakukan uji coba :

- Uji coba dengan *watermark* yang berbeda-beda
- Uji coba pencarian batas *idx* dan *lastPeak*
- Uji coba pencarian lokasi *watermark*

Ketiga uji coba ini diujicobakan pada citra lena_org.bmp (gambar 5.2). Uji coba dilakukan pada citra ber-*watermark* dan citra asli. Citra asli yang dimaksud adalah citra yang belum diberi *watermark* sama sekali.

Gambar 5.2 Citra lena_org.bmp

5.3 UJI COBA DENGAN CITRA BER-*WATERMARK* TIDAK MENGALAMI PERUBAHAN

Uji coba ini dilakukan dengan melakukan ekstraksi citra ber-*watermark* yng tidak mengalami serangan apapun. Hasil dari ekstraksi ini tampak pada Tabel 5.3. dengan keterangan : Korelasi adalah Hasil korelasi antara *watermark* asli dengan *watermark* hasil ekstraksi, Ber-*watermark* berarti citra yang diekstraksi adalah citra ber-*watermark,* Asli berarti citra yang diekstraksi adalah citra tidak ber-*watermark,* dan PSNR adalah *Peak Signal to Noise Ratio* antara citra ber-*watermark* dengan citra tak ber-*watermark.*

Tabel 5.3 Hasil uji coba dengan citra ber-*watermark* tidak mengalami perubahan

citra	Korelasi		PSNR
	Ber-*watermark*	Asli	
Barbara.bmp	0.9084	0.0738	40.6040 + 0.0000i
Bikes.bmp	0.9141	0.1758	40.1505 + 0.0000i
mandrill.bmp	0.9361	0.1865	46.8742 + 0.0000i
peppers.bmp	0.9676	0.0666	42.1627 + 0.0000i

(a) (b)

(c) (d)

Gambar 5.3 Citra-citra ber-*watermark*:(a) Barbara.bmp, (b)Bikes.bmp, (c)mandrill.bmp, (d)peppers.bmp

5.4 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI ROTASI

Uji coba ini menggunakan citra ber-*watermark* yang dilewatkan pada fungsi *imrotate* milik *matlab* untuk menghasilkan citra yang mengalami rotasi sebesar *Derajat Rotasi*. Untuk yang mengalami *cropping*, opsi 'crop' dalam fungsi *imrotate* dimasukkan sedangkan yang tidak mengalami *cropping* opsi tersebut tidak dimasukkan. Hasil dari uji coba dengan citra ber-*watermark* mengalami

rotasi tampak pada gambar 5.4 dan gambar 5.5 dan detail hasil uji coba pada Tabel A.1 hingga tabel A.8 dalam Lampiran. Keterangan gambar untuk gambar uji coba adalah :

1B = citra Barbara.bmp ber-*watermark*,

1A = citra Barbara.bmp tak ber-*watermark*,

2B = citra Bikes.bmp ber-*watermark*,

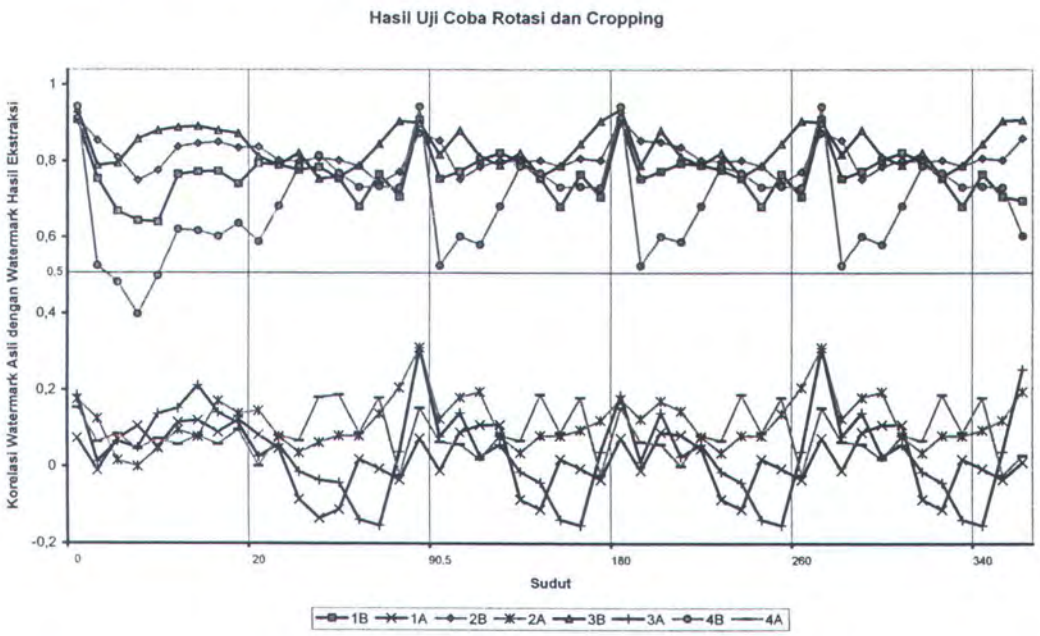
2A = citra Bikes.bmp tak ber-*watermark*,

3B = citra mandrill.bmp ber-*watermark*,

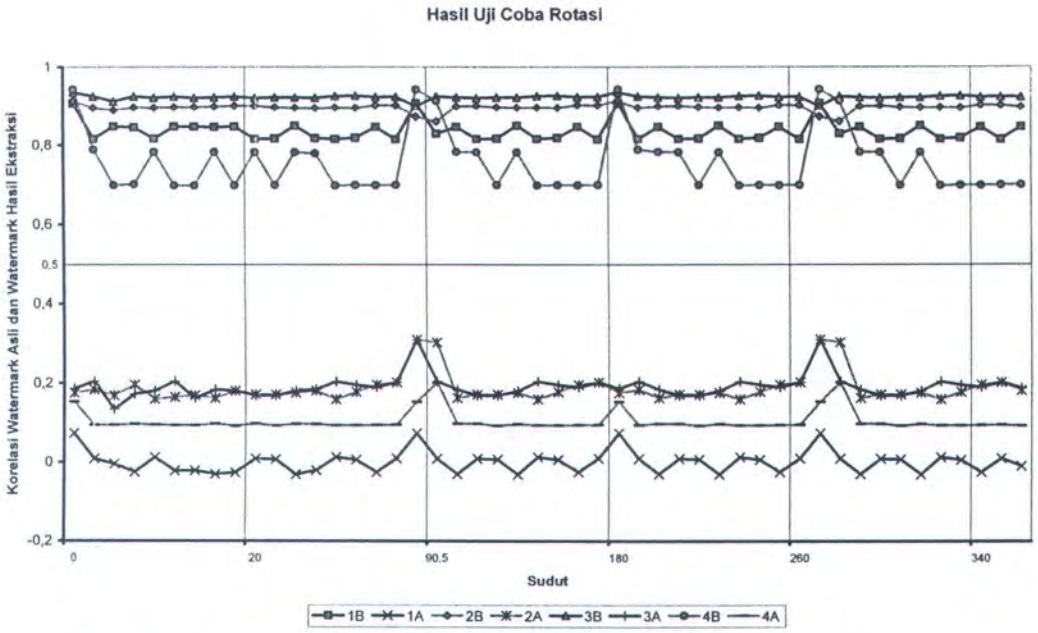
3A = citra mandrill.bmp tak ber-*watermark*,

4B = citra peppers.bmp ber-*watermark*,

4A = citra peppers.bmp tak ber-*watermark*,



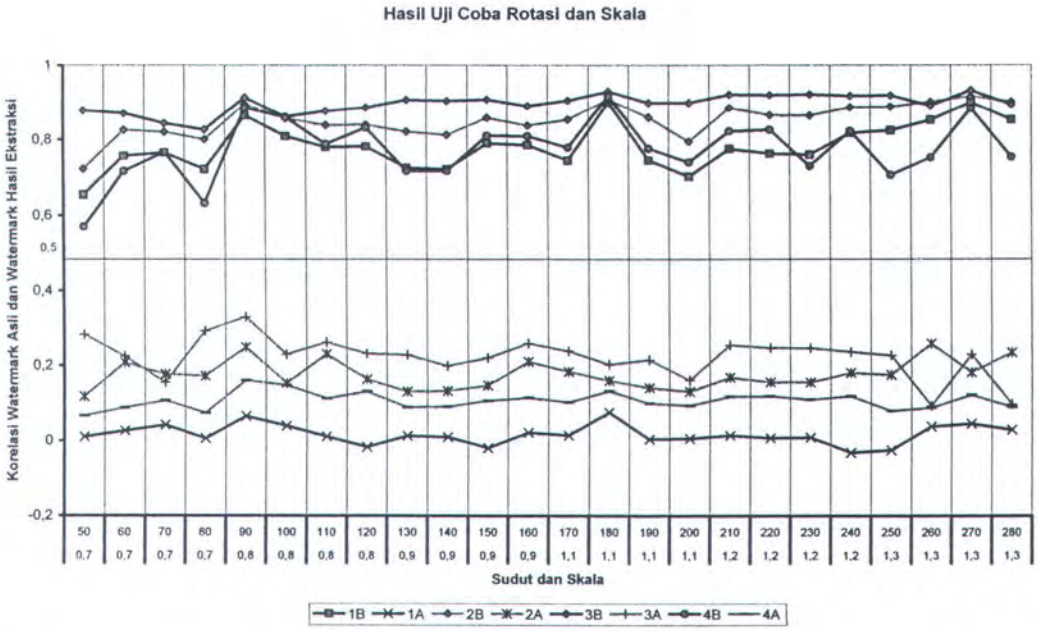
Gambar 5.4 Hasil uji coba dengan citra ber-*watermark* mengalami rotasi dan *cropping*



Gambar 5.5 Hasil uji coba dengan citra ber-*watermark* mengalami rotasi

5.5 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI SKALA

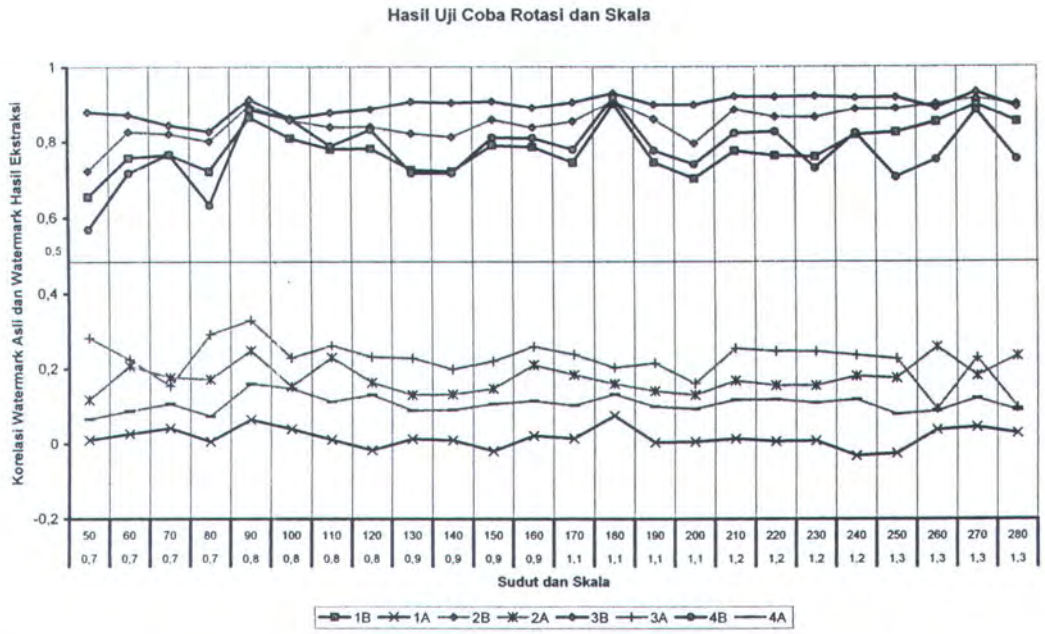
Uji coba ini menggunakan citra ber-*watermark* yang dilewatkan pada fungsi *imresize* milik *matlab* untuk menghasilkan citra yang mengalami perubahan skala sebesar *Skala* kali. Hasil dari uji coba dengan citra ber-*watermark* mengalami skala tampak gambar 5.6 dengan detail tampak pada Tabel A.9 dan Tabel A.10 dalam Lampiran.



Gambar 5.7 Hasil uji coba dengan citra ber-*watermark* mengalami rotasi dan skala

5.7 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI SKALA DAN TRANSLASI

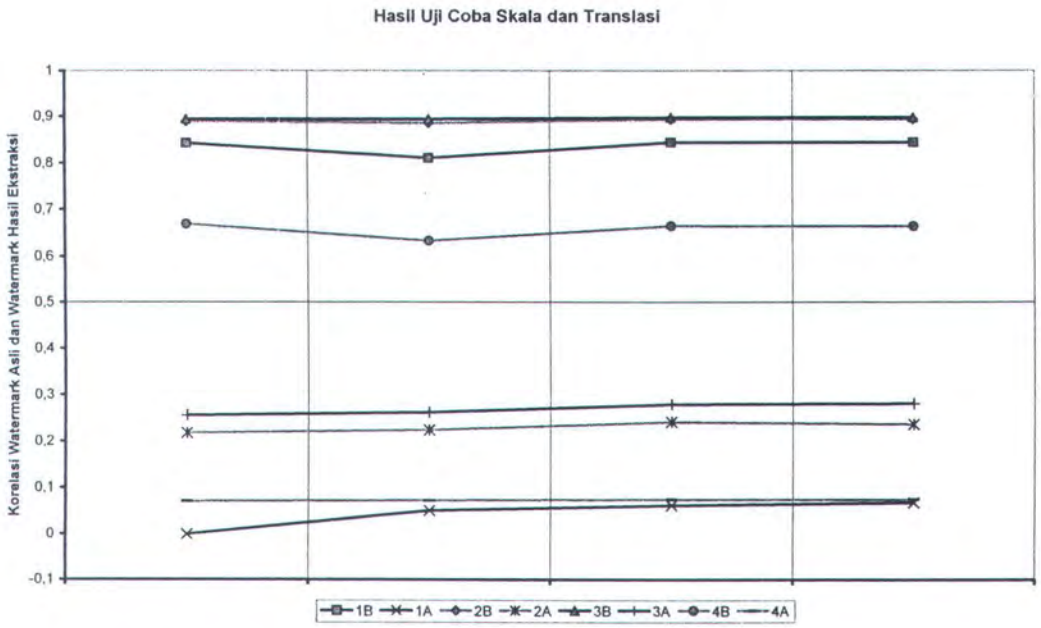
Uji coba ini menggunakan citra ber-*watermark* yang dilewatkan pada fungsi *imresize* milik *matlab* untuk menghasilkan citra yang mengalami perubahan skala dan di-*shift* dengan *swapping* sehingga misalkan di-*shift* keatas 10 *pixel*, maka 10 *pixel* yag di-*shift* akan dipindah (*swap*) kebawah dari citra. Skala 0,7 digunakan untuk menghindari terpotongnya citra yang disebabkan oleh proses *shift* yang terjadi. Hasil dari uji coba dengan citra ber-*watermark* mengalami skala dan translasi tampak pada gambar 5.8 dibawah dan Tabel A.13 dan tabel A.14 dalam Lampiran.



Gambar 5.7 Hasil uji coba dengan citra ber-*watermark* mengalami rotasi dan skala

5.7 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI SKALA DAN TRANSLASI

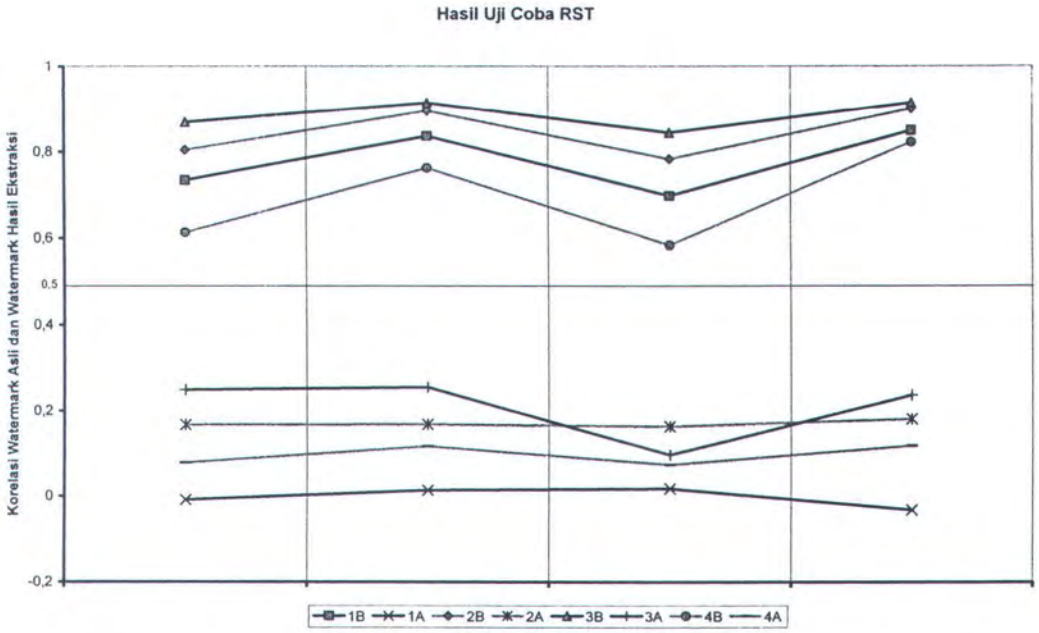
Uji coba ini menggunakan citra ber-*watermark* yang dilewatkan pada fungsi *imresize* milik *matlab* untuk menghasilkan citra yang mengalami perubahan skala dan di-*shift* dengan *swapping* sehingga misalkan di-*shift* keatas 10 *pixel*, maka 10 *pixel* yag di-*shift* akan dipindah (*swap*) kebawah dari citra. Skala 0,7 digunakan untuk menghindari terpotongnya citra yang disebabkan oleh proses *shift* yang terjadi. Hasil dari uji coba dengan citra ber-*watermark* mengalami skala dan translasi tampak pada gambar 5.8 dibawah dan Tabel A.13 dan tabel A.14 dalam Lampiran.



Gambar 5.8 Hasil uji coba dengan citra ber-*watermark* mengalami skala dan translasi

5.8 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI ROTASI, SKALA DAN TRANSLASI (RST)

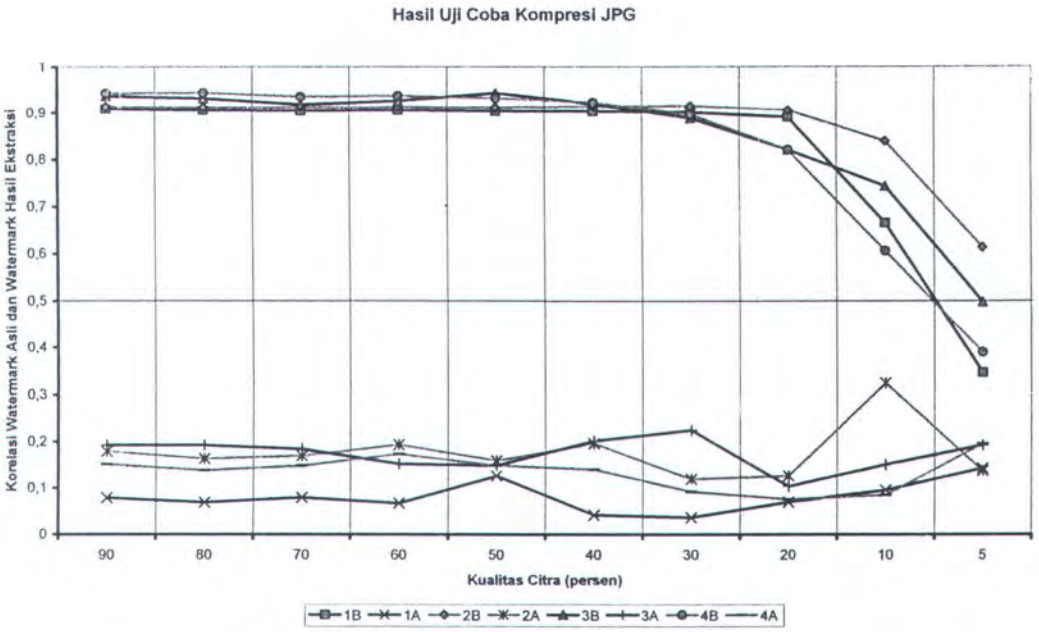
Uji coba ini menggunakan citra ber-*watermark* yang dilewatkan pada fungsi *imrotate* milik *matlab* untuk menghasilkan citra yang mengalami rotasi sebesar *theta* derajat, kemudian dilewatkan fungsi *imresize* milik *matlab* untuk menghasilkan citra yang mengalami skala *Scale* kali. Dan hasil kedua fungsi di-*shift* dengan *swapping* sehingga misalkan di-*shift* keatas 5 *pixel*, maka 5 *pixel* yang di-*shift* akan dipindah (*swap*) kebawah dari citra. Hasil dari uji coba dengan citra ber-*watermark* mengalami rotasi, skala dan translasi (RST) tampak pada gambar 5.9 dibawah dan Tabel A.15 hingga tabel A.16 dalam Lampiran.



Gambar 5.9 Hasil uji coba dengan citra ber-*watermark* mengalami rotasi, skala dan translasi

5.9 UJI COBA DENGAN CITRA BER-*WATERMARK* MENGALAMI KOMPRESI JPG

Uji coba ini mengunakan fungsi *matlab imwrite* untuk menghasilkan gambar bertipe jpg dengan kualitas yang bervariasi. Hasil dari Uji coba dengan citra ber-*watermark* mengalami kompresi jpg tampak pada gambar 5.10 dibawah dan pada Tabel A.17 dan tabel A.18 dalam Lampiran.



Gambar 5.10 Hasil uji coba dengan citra ber-*watermark* mengalami kompresi JPG

5.10 EVALUASI UJI COBA KETAHANAN

Hasil ekstraksi *watermark* ini dipengaruhi oleh sangat jumlah serangan yang terjadi pada citra ber-*watermark*. Semakin banyak serangan yang terjadi pada citra ber-*watermark* semakin buruk hasil ekstraksi. Untuk menunjukkan ada tidaknya *watermark*, ditentukan batas minimal hasil korelasi sebesar 0,5. Batas ini didapat dari paper milik D.Zheng [1]. Berikut ini evaluasi terhadap hasil uji coba yang telah dilakukan. :

1. Uji coba dengan citra ber-*watermark* tidak berubah,

   Dalam uji coba ini dapat dilihat hasil korelasi *watermark* selalu dibawah 1, Hal ini menunjukkan lokasi *watermark* tidaklah kosong, sehingga ada nilai-nilai yang mempengaruhi nilai *watermark-watermark* yang ditambahkan. Hasil perhitungan *Peak Signal to Noise Ratio (PSNR)* berada diatas 40 dB untuk masing-masing citra, hal ini menunjukkan hasil *embedding watermark* tetap menghasilkan citra yang baik. Dari ekstraksi

terhadap citra asli menunjukkan nilai hasil korelasi yang sangat rendah yang berarti *watermark* tidak ditemukan.

2. Uji coba dengan citra ber-*watermark* mengalami rotasi,

   Dari hasil yang didapatkan terlihat untuk citra yang mengalami *cropping*(pemotongan) memiliki hasil korelasi yang lebih rendah dari yang tidak mengalami *cropping*. Hal ini disebabkan dengan adanya *cropping*, data yang ada mengalami pengurangan sehingga *magnitude* dari citra mengalami perubahan, yang selanjutnya mempengaruhi *watermark* yang ada. Namun jika dilihat terdeteksi atau tidaknya *watermark* maka bisa dilihat *watermark* terdeteksi untuk hampir di semua kasus rotasi, kecuali pada rotasi 1 hingga 2 derajat dan di *crop*. Hal ini menunjukkan *watermarking* tahan terhadap rotasi, tetapi tidak tahan terhadap *cropping*

3. Uji coba dengan citra ber-*watermark* mengalami skala,

   Dari uji coba didapatkan nilai hasil korelasi diatas 0,5 untuk semua kasus. Hal ini menunjukkan terdeteksinya *watermark* dan *watermark* tahan terhadap skala.

4. Uji coba dengan citra ber-*watermark* mengalami rotasi dan skala,

   Dari uji coba didapatkan nilai hasil korelasi diatas 0,5 untuk semua kasus uji coba. Hal ini menunjukkan terdeteksinya *watermark* dan *watermark* tahan terhadap serangan rotasi dan skala yang bersamaan.

5. Uji coba dengan citra ber-*watermark* mengalami skala dan translasi

   Dari uji coba didapat hasil korelasi diatas 0,5. Hal ini menunjukkan terdeteksinya *watermark* dan *watermark* tahan terhadap translasi

6. Uji coba dengan citra ber-*watermark* mengalami rotasi, skala dan translasi

   Dari uji coba didapat hasil korelasi diatas 0,5. Hal ini menunjukkan terdeteksinya *watermark* dan *watermark* tahan terhadap rotasi, skala dan translasi

7. Uji coba dengan citra ber-*watermark* mengalami kompresi jpg

   Dari uji coba didapat hasil korelasi diatas 0,5 untuk kualitas diatas 5 persen, dari sini bisa disimpulkan *watermark* tahan terhadap kompresi jpg hingga kualitas 10 persen

5.11 UJI COBA DENGAN *WATERMARK* YANG BERBEDA-BEDA

Uji coba ini dilakukan pada citra lena_org.bmp dengan sejumlah karakter random dan jumlah karakter yang berbeda-beda, sedangkan parameter lainnya sama untuk tiap-tiap uji coba. Parameter tersebut adalah panjang 3, polinom [5,4,2,1] dan inisialisasi *register* [1,0,1,1] untuk generator *M-Sequence*, n = 7,k = 5 untuk RS encoder. Posisi *watermark* yang adalah (1,188) di domain *LPM* beresolusi 256x256 dengan kekuatan *watermark* 95. Selain dengan karakter random *watermark* yang diuji juga berupa *array* random biner satu dimensi untuk baris dan kolom dari domain *LPM. Array* random ini berukuran *panjang* untuk kolom dan *lebar* untuk baris. Hasil dari uji coba dengan berbagai macam *watermark* tampak pada gambar 5.12 dan 5.13, detail uji coba tampak pada Tabel A.19 hingga tabel A.21 dalam Lampiran, sedangkan uji coba panjang dan lebar *watermark* tampak pada gambar 5.14 dan 5.15, dengan detail pada tabel A.22 hingga tabel A.30. Dengan keterangan gambar :

NB = citra lena_org ber-*watermark* normal,

NA = citra lena_org tak ber-*watermark* normal,

KJB = citra lena_org ber-*watermark* dikompresi JPG 70 persen,

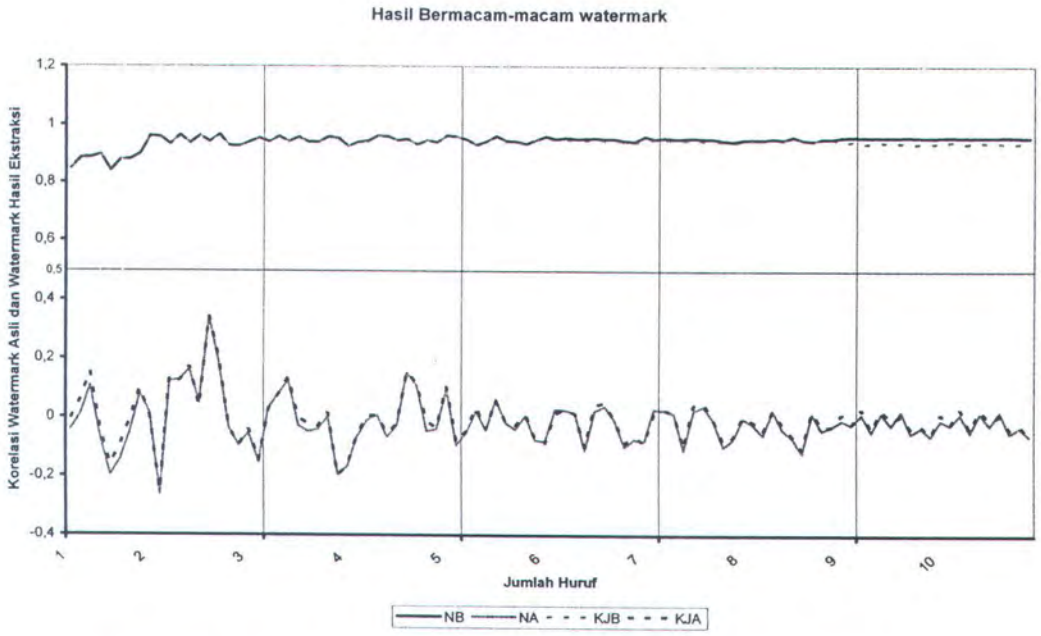
KJA = citra lena_org tak ber-*watermark* dikompresi JPG 70 persen.

Ukuran masing-masing *watermark* ada pada tabel 5.4 dibawah.

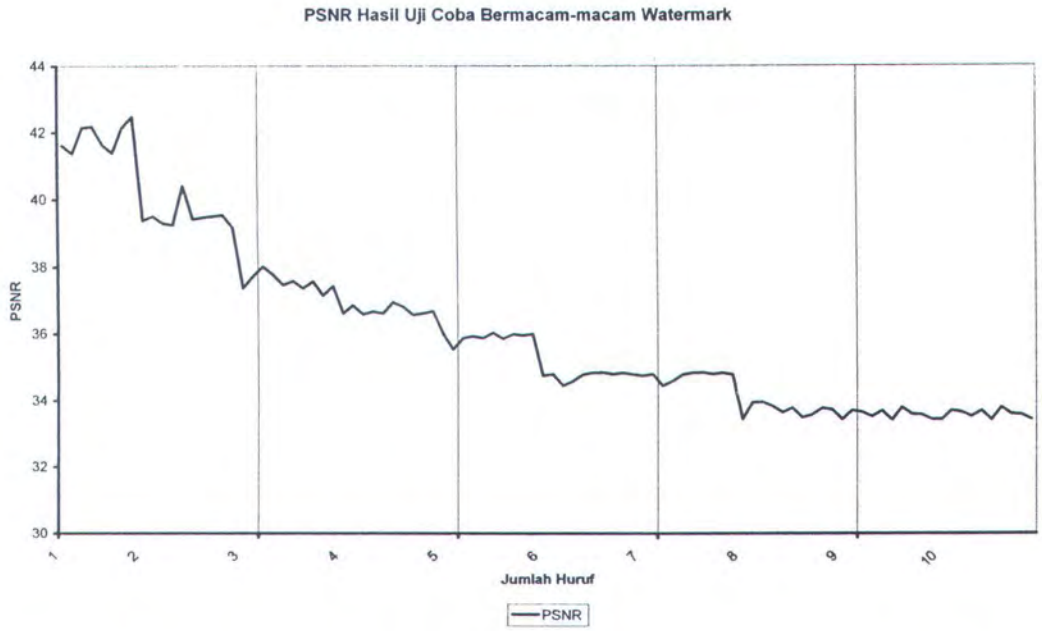
Tabel 5.4 Tabel ukuran *watermark* untuk uji coba berbagai macam *watermark*

Jumlah Karakter	Ukuran *Watermark*
1	7x6
2	9x7
3	15x7
4	21x6
5	21x7
6	21x9
7	35x6
8	21x12
9	21x13
10	49x6

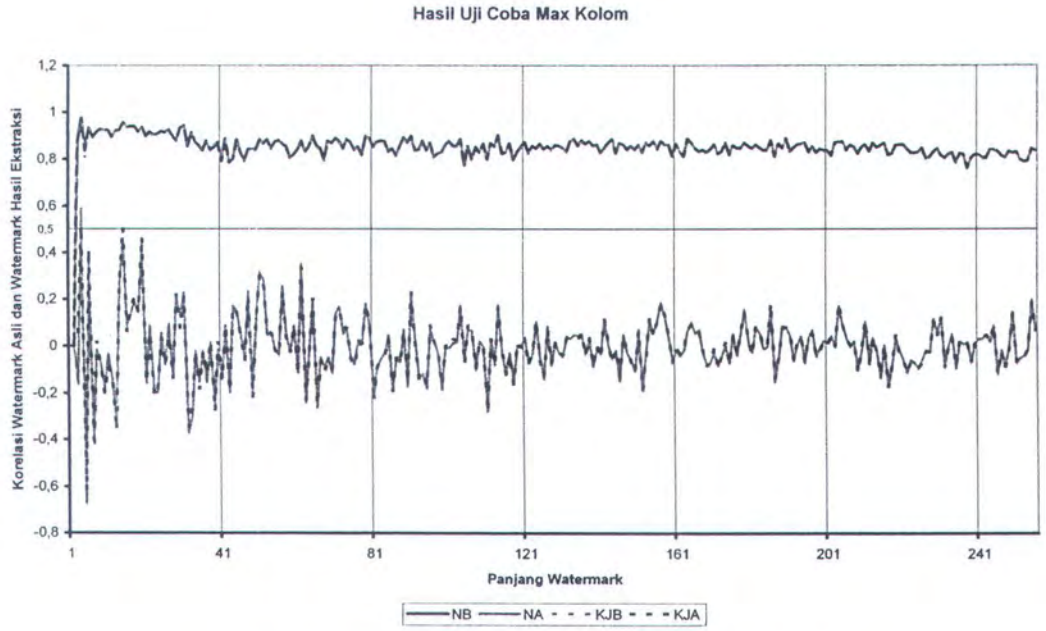
Gambar 5.11 Citra lena_org.bmp ber-*watermark* dengan *watermark* dari 1 huruf (berukuran 7x6) dan PSNR 43.4697 - 0.0000i.



Gambar 5.12 Hasil uji coba dengan *watermark* yang berbeda-beda



Gambar 5.13 PSNR hasil uji coba dengan *watermark* yang berbeda-beda



Gambar 5.14 Hasil uji coba kolom maksimum *watermark*

187 terjadi pada citra Bikes.bmp ber-*watermark* yang mengalami skala 0,7 dan rotasi 70 derajat. dari nilai – nilai tersebut dibulatkan keatas untuk digunakan sebagai batas. *lastPeak* dibulatkan menjadi 0,3 keatas dengan kelipatan 0,1 terdekat, sedangkan *lastIdx* dibulatkan keatas dengan kelipatan 100 terdekat menjadi 200 yang merupakan batas *lastPeak* dan *idx* dari langkah evaluasi puncak-puncak dalam ekstraksi *watermark*. *lastPeak* dibulatkan keatas karena pernah ada hasil yang salah tapi memiliki nilai puncak antara 2,4 dan 2,7. Hasil ini tidak dicatat dimana pernah terjadinya karena hasil yang salah tidak pernah disimpan.

5.15 UJI COBA PENCARIAN LOKASI *WATERMARK*

Posisi *watermark* dicari dengan cara melakukan operasi *embedding* dan ektraksi *watermark* di sebagian titik dalam daerah yang diperkirakan sebagai daerah *midband* dari *magnitude*, misalnya daerah (1,200) sampai dengan (1,220). Untuk masing-masing titik dihitung pula *psnr* citra hasil *embedding* dibandingkan dengan citra asli untuk tingkat kekuatan *watermark* yang berbeda-beda, misalnya antara 40 sampai dengan 60. dari pasangan korelasi *watermark* asli dengan *watermark* hasil ekstraksi dan nilai *psnr* yang didapat diambil salah satu pasangan titik dan kekuatan yang memiliki nilai *psnr* dan hasil ekstraksi yang bagus, misalnya nilai *psnr* mendekati 40 dengan hasil korelasi lebih besar dari 0,9. Pasangan inilah yang akan menjadi posisi *embedding* dari *watermark*. Proses ini tercakup dalam program "pembuatan tabel lokasi *watermark*". Hasil pencarian posisi *watermark* dari citra lena_org.bmp dalam daerah (1,205)-(1,215) dengan kekuatan antara 48 – 52 : (kor = hasil korelasi *watermark* asli dengan *watermark* hasil korelasi)

Tabel 5.6 Tabel hasil program "Pembuatan Tabel Lokasi *Watermark*"

Titik (1,x)	Kekuatan									
	48	48	49	49	50	50	51	51	52	52
	psnr	kor	psnr	kor	psnr	kor	psnr	kor	psnr	kor
205	45,48	0,93	45,31	0,93	45,13	0,93	44,96	0,93	44,80	0,94
206	45,53	0,94	45,36	0,94	45,19	0,94	45,02	0,94	44,86	0,95
207	45,46	0,94	45,29	0,94	45,12	0,94	44,95	0,95	44,78	0,95
208	45,48	0,94	45,31	0,94	45,14	0,94	44,97	0,94	44,81	0,94
209	45,49	0,93	45,32	0,94	45,15	0,94	44,98	0,94	44,81	0,94
210	45,50	0,94	45,33	0,94	45,16	0,95	44,99	0,95	44,82	0,95
211	45,50	0,95	45,33	0,95	45,15	0,95	44,98	0,95	44,82	0,95
212	45,48	0,96	45,31	0,97	45,14	0,97	44,97	0,97	44,80	0,97
213	45,52	0,96	45,35	0,96	45,18	0,96	45,01	0,97	44,84	0,97
214	45,46	0,96	45,29	0,96	45,12	0,96	44,95	0,96	44,78	0,97
215	45,50	0,97	45,33	0,97	45,16	0,97	44,99	0,97	44,82	0,97

5.16 EVALUASI UJI COBA PENCARIAN LOKASI *WATERMARK*

Dari tabel pasangan posisi dan kekuatan diatas diambil posisi (1,210) dengan kekuatan 50. Pada posisi tersebut nilai *PSNR* lebih besar daripada 40 dB dan nilai hasil korelasi diatas 0,9 yang menjadikan posisi *watermark* tersebut menjadi salah satu posisi *watermark* yang baik, karena memiliki kualitas citra hasil *embedding* yang baik dengan hasil ekstraksi yang baik pula.

BAB VI

KESIMPULAN DAN SARAN

BAB VI
KESIMPULAN DAN SARAN

Pada bab terakhir ini akan dijelaskan tentang kesimpulan yang didapat dari Tugas Akhir ini dan saran-saran yang perlu untuk diperhatikan untuk pengembangan selanjutnya.

6.1 KESIMPULAN

Berdasarkan aplikasi yang telah dibuat beserta uji coba yang telah dilakukan terhadapnya, dapat diambil beberapa kesimpulan sebagai berikut:

1. *Watermarking* independen terhadap rotasi, skala dan translasi, namun terhadap *cropping* memiliki ketahanan yang terbatas. Sedangkan terhadap kompresi *JPG watermarking* tahan terhadap kompresi hingga kualitas gambar 10 persen.
2. Semakin besar *watermark* semakin kecil nilai hasil korelasi antara *watermark* asli dengan *watermark* hasil ekstraksi dan *PSNR*. Dengan kata lain ukuran *watermark* berbanding terbalik dengan kualitas *watermark* yang diekstraksi dan kualitas citra.
3. Kondisi terbaik ekstraksi dicapai ketika melakukan proses terhadap citra normal, yaitu citra yang tidak mengalami serangan. Sedangkan kondisi terburuk terjadi di kasus-kasus yang bervariasi, dengan persamaan nilai *lastPeak* berada di bawah batas.
4. Ukuran maksimal *watermark* ditentukan oleh resolusi domain *LPM* dari *watermark* untuk kolom dan terpotongnya *watermark* oleh kompresi *JPG* untuk barisnya.

6.2 SARAN

Adapun beberapa saran yang dianggap perlu untuk pengembangan dari Tugas Akhir ini adalah sebagai berikut :

1. Algoritma dari *embedding watermarking* harus dikembangkan lebih lanjut untuk mendapatkan hasil ekstraksi yang lebih baik.

2. Algoritma dari ekstraksi *watermark* harus dikembangkan lebih lanjut untuk mengatasi *cropping* dan penambahan *pixel* nol yang disebabkan oleh rotasi dan skala.
3. Program ekstraksi dan *embedding watermark* dapat dikembangkan lebih lanjut untuk melakukan *embedding* dan ekstraksi pada citra berwarna, dan citra yang memiliki ukuran panjang dan lebar yang tidak sama.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1] D.Zheng; J.Zhao; A.El Saddik: "RST Invariant Digital Image Watermarking Based on Log-Polar Mapping and Phase Correlation". IEEE Trans. Circuits Syst. Video Technol.,vol 13, no.8, 2003; p.753-765.

[2] Mathworks, Inc: "MATLAB Function Reference". Matlab 6.1 Help;2001.

[3] J.Meel: "Spread Spectrum (SS) – An Introduction". 1999;avail. from: http://www.sss-mag.com/pdf/Ss_jme_denayer_intro_print.pdf.

[4] Poli, Alain; Hoguet, Lloranc : "Error Correcting Codes : Theory and Applications". Prentice Hall And Masson; 1992.

[5] Y.Liu; D.Zheng; J.Zhao: "A Rectification Scheme for RST Invariant Image Watermarking". 2005.

[6] I.J. Cox; J. Killian; T. Leighton; T. Shamoon: "Secure Spread Spectrum Watermarking for Multimedia". Technical Report 95-10, NEC Research Institute, 1995.

[7] J.J.K.O'Ruanaidh; T. Pun: "Rotation, Scale and Translation Invariant Digital Image Watermarking". Signal Processing, Vol. 66, no. 3, 1998; p. 303-317.

[8] S.P. Mohanty: "Digital Watermarking: A Tutorial Review". 1999.

[9] S. Katzenbeisser; F.A.P. Petitcolas: "Information Hiding Techniques for Steganography and Digital Watermarking". Artech House; 2000.

[10] Petitcolas, Fabien.A.P; Steinebach, Martin;Raynal, Frédéric;Dittman, Jana; Fontaine, Caroline; Fatés, Nazim: "A public automated web-based evaluation service for watermarking schemes: StirMark Benchmark". In Ping Wah Wong and Edward J. Delp, editors, proceedings of electronic imaging, security and watermarking of multimedia contents III, vol. 4314, San Jose, California, U.S.A., 20-26 January 2001. The Society for imaging science and technology (I.S.&T.) and the international Society for optical engineering (SPIE). ISSN 0277-786X.

[11] "Phase (Waves)". Wikipedia, the free encyclopedia, 2005; avail. from: http://en.wikipedia.org/.

[12] Wilmer, Adam: "Fourier-Mellin based Image Registration (with GUI)". MATLAB Central File Exchange; 2003.

Tabel A.4 Lanjutan tabel hasil uji coba rotasi dengan *cropping* untuk mandrill.bmp dan peppers.bmp

Derajat Rotasi	Korelasi			
	mandrill.bmp		peppers.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
90,5	0,8173	0,0783	0,5238	0,0642
100,0	0,8807	0,1405	0,6023	0,0578
110,0	0,8124	0,0261	0,5807	0,0189
120,0	0,7895	0,0574	0,6823	0,0804
130,0	0,8221	-0,0137	0,7868	0,0673
140,0	0,7594	-0,0421	0,7691	0,1876
150,0	0,7888	-0,1382	0,7316	0,0874
160,0	0,8454	-0,1530	0,7333	0,1798
170,0	0,9051	0,0379	0,7308	-0,0230
180,0	0,9361	0,1865	0,9431	0,1533
180,5	0,7893	0,0122	0,5238	0,0642
190,0	0,8807	0,1405	0,6023	0,0578
200,0	0,8124	0,0261	0,5882	0,0014
210,0	0,7895	0,0574	0,6823	0,0804
220,0	0,8221	-0,0137	0,7868	0,0673
230,0	0,7594	-0,0421	0,7691	0,1876
240,0	0,7888	-0,1382	0,7316	0,0874
250,0	0,8454	-0,1530	0,7333	0,1798
260,0	0,9051	0,0379	0,7308	-0,0230
270,0	0,9010	0,3098	0,9431	0,1533
270,5	0,8173	0,0783	0,5238	0,0642
280,0	0,8807	0,1405	0,6023	0,0578
290,0	0,8124	0,0261	0,5807	0,0189
300,0	0,7895	0,0574	0,6823	0,0804
310,0	0,8221	-0,0137	0,7868	0,0673
320,0	0,7594	-0,0421	0,7691	0,1876
330,0	0,7888	-0,1382	0,7316	0,0874
340,0	0,8454	-0,1530	0,7333	0,1798
350,0	0,9051	0,0379	0,7308	-0,0230
359,5	0,9071	0,2537	0,6033	0,0282

Tabel A.5 Tabel hasil uji coba rotasi untuk Barbara.bmp dan Bikes.bmp

Derajat Rotasi	Korelasi			
	Barbara.bmp		Bikes.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
0,0	0,9084	0,0738	0,9141	0,1758
0,5	0,8162	0,0091	0,8964	0,1837
1,0	0,8487	-0,0043	0,8907	0,1689
1,5	0,8462	-0,0248	0,8978	0,1960
2,0	0,8171	0,0125	0,8974	0,1602
5,0	0,8483	-0,0213	0,8985	0,1651
5,5	0,8482	-0,0212	0,8990	0,1695
10,0	0,8474	-0,0300	0,9004	0,1627
10,5	0,8481	-0,0258	0,9019	0,1806
20,0	0,8164	0,0092	0,9017	0,1718
30,0	0,8174	0,0072	0,8983	0,1716
40,0	0,8507	-0,0309	0,8970	0,1760
45,0	0,8185	-0,0200	0,8960	0,1815
50,0	0,8165	0,0131	0,8973	0,1598
60,0	0,8193	0,0066	0,8961	0,1780
70,0	0,8470	-0,0252	0,9035	0,1960
80,0	0,8155	0,0101	0,9030	0,2022
90,0	0,9084	0,0738	0,8734	0,3102
90,5	0,8298	0,0091	0,8607	0,3031
100,0	0,8474	-0,0300	0,9004	0,1627
110,0	0,8164	0,0092	0,9017	0,1718
120,0	0,8174	0,0072	0,8983	0,1716
130,0	0,8507	-0,0309	0,8970	0,1760
140,0	0,8165	0,0131	0,8973	0,1598
150,0	0,8193	0,0066	0,8961	0,1780
160,0	0,8470	-0,0252	0,9035	0,1960
170,0	0,8155	0,0101	0,9030	0,2022
180,0	0,9084	0,0738	0,9141	0,1758
180,5	0,8162	0,0091	0,8964	0,1837
190,0	0,8474	-0,0300	0,9004	0,1627
200,0	0,8164	0,0092	0,9017	0,1718
210,0	0,8174	0,0072	0,8983	0,1716
220,0	0,8507	-0,0309	0,8970	0,1760
230,0	0,8165	0,0131	0,8973	0,1598
240,0	0,8193	0,0066	0,8961	0,1780
250,0	0,8470	-0,0252	0,9035	0,1960
260,0	0,8155	0,0101	0,9030	0,2022
270,0	0,9084	0,0738	0,8734	0,3102
270,5	0,8298	0,0091	0,8607	0,3031
280,0	0,8474	-0,0300	0,9004	0,1627
290,0	0,8164	0,0092	0,9017	0,1718
300,0	0,8174	0,0072	0,8983	0,1716

Tabel A.6 Lanjutan tabel hasil uji coba rotasi untuk Barbara.bmp dan Bikes.bmp

Derajat Rotasi	Korelasi			
	Barbara.bmp		Bikes.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
310,0	0,8507	-0,0309	0,8970	0,1760
320,0	0,8165	0,0131	0,8973	0,1598
330,0	0,8193	0,0066	0,8961	0,1780
340,0	0,8470	-0,0252	0,9035	0,1960
350,0	0,8155	0,0101	0,9030	0,2022
359,5	0,8473	-0,0095	0,8987	0,1819

Tabel A.7 Tabel hasil uji coba rotasi untuk mandrill.bmp dan peppers.bmp

Derajat Rotasi	Korelasi			
	mandrill.bmp		peppers.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
0,0	0,9361	0,1865	0,9431	0,1533
0,5	0,9255	0,2046	0,7887	0,0947
1,0	0,9138	0,1343	0,7002	0,0946
1,5	0,9254	0,1730	0,7026	0,0968
2,0	0,9227	0,1808	0,7828	0,0956
5,0	0,9251	0,2049	0,6993	0,0936
5,5	0,9219	0,1625	0,6993	0,0935
10,0	0,9234	0,1842	0,7830	0,0982
10,5	0,9251	0,1808	0,6996	0,0922
20,0	0,9217	0,1692	0,7822	0,0982
30,0	0,9235	0,1689	0,7002	0,0927
40,0	0,9230	0,1803	0,7823	0,0970
45,0	0,9227	0,1851	0,7788	0,0966
50,0	0,9267	0,2040	0,6992	0,0937
60,0	0,9272	0,1958	0,7002	0,0938
70,0	0,9243	0,1899	0,6995	0,0945
80,0	0,9247	0,2024	0,7005	0,0950
90,0	0,9010	0,3098	0,9431	0,1533
90,5	0,9255	0,2046	0,9148	0,2022
100,0	0,9234	0,1842	0,7830	0,0982
110,0	0,9217	0,1692	0,7822	0,0982
120,0	0,9235	0,1689	0,7002	0,0927
130,0	0,9230	0,1803	0,7823	0,0970
140,0	0,9267	0,2040	0,6992	0,0937
150,0	0,9272	0,1958	0,7002	0,0938
160,0	0,9243	0,1899	0,6995	0,0945
170,0	0,9247	0,2024	0,7005	0,0950
180,0	0,9361	0,1865	0,9431	0,1533
180,5	0,9255	0,2046	0,7887	0,0947
190,0	0,9234	0,1842	0,7830	0,0982

Tabel A.8 Lanjutan tabel hasil uji coba rotasi untuk mandrill.bmp dan peppers.bmp

Derajat Rotasi	Korelasi			
	mandrill.bmp		peppers.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
200,0	0,9217	0,1692	0,7822	0,0982
210,0	0,9235	0,1689	0,7002	0,0927
220,0	0,9230	0,1803	0,7823	0,0970
230,0	0,9267	0,2040	0,6992	0,0937
240,0	0,9272	0,1958	0,7002	0,0938
250,0	0,9243	0,1899	0,6995	0,0945
260,0	0,9247	0,2024	0,7005	0,0950
270,0	0,9010	0,3098	0,9431	0,1533
270,5	0,9255	0,2046	0,9148	0,2022
280,0	0,9234	0,1842	0,7830	0,0982
290,0	0,9217	0,1692	0,7822	0,0982
300,0	0,9235	0,1689	0,7002	0,0927
310,0	0,9230	0,1803	0,7823	0,0970
320,0	0,9267	0,2040	0,6992	0,0937
330,0	0,9272	0,1958	0,7002	0,0938
340,0	0,9243	0,1899	0,6995	0,0945
350,0	0,9247	0,2024	0,7005	0,0950
359,5	0,9240	0,1876	0,7011	0,0933

Tabel A.9 Tabel hasil uji coba skala untuk Barbara.bmp dan Bikes.bmp

Skala	Korelasi			
	Barbara.bmp		Bikes.bmp	
	Ber-*watermark*	Asli	Ber-*watermark*	Asli
0,6	0,7770	-0,0014	0,7978	0,1159
0,7	0,8447	0,0699	0,8936	0,2393
0,8	0,8636	0,0641	0,8959	0,2557
0,9	0,8458	0,1291	0,8680	0,2534
1,0	0,9084	0,0738	0,9141	0,1758
1,1	0,8996	0,0753	0,9077	0,1600
1,2	0,9031	0,0484	0,9116	0,1739
1,3	0,8994	0,0451	0,9157	0,1817

Tabel A.21 Lanjutan kedua tabel hasil uji coba untuk berbagai macam *watermark*

Panjang Karakter	Karakter	Korelasi				psnr
		Normal		jpg 70 persen		
		Ber-*watermark*	Asli	Ber-*watermark*	Asli	
7	WCQzul]	0,9614	-0,0805	0,9565	-0,0873	34,7340
	rdA\Djl	0,9523	0,0274	0,9464	0,0123	34,7743
	pTodPqf	0,9559	0,0251	0,9485	0,0276	34,4359
	vbD^enk	0,9527	0,0142	0,9462	0,0139	34,5728
	quRyhzA	0,9532	-0,1115	0,9464	-0,1010	34,7669
	s^TOkbu	0,9555	0,0224	0,9493	0,0422	34,8241
	mYpgwUg	0,9524	0,0383	0,9443	0,0508	34,8304
	ZoAldVC	0,9519	-0,0091	0,9476	-0,0081	34,7804
	Zt]Xkz[	0,9459	-0,0996	0,9394	-0,0855	34,8200
	N^dletJ	0,9434	-0,0739	0,9368	-0,0655	34,7721
9	tRKvfH[Wd	0,9491	-0,0025	0,9488	0,0051	33,4378
	jlQmWLdtX	0,9495	-0,0220	0,9476	-0,0136	33,9431
	LwhYSJlXc	0,9490	-0,0591	0,9490	-0,0460	33,9602
	GhcPPnzVN	0,9535	0,0236	0,9512	0,0244	33,8284
	kYWe^avFq	0,9493	-0,0406	0,9476	-0,0268	33,6412
	mqwz[gByE	0,9601	-0,0689	0,9601	-0,0644	33,7788
	QJpepsWvN	0,9486	-0,1212	0,9468	-0,1082	33,4888
	rZCN\dg[]	0,9450	0,0138	0,9450	0,0222	33,5717
	elqn_wBax	0,9529	-0,0425	0,9519	-0,0333	33,7662
	edWALQd]D	0,9531	-0,0298	0,9514	-0,0298	33,7245
10	nNzEyXlzAt	0,9581	-0,0093	0,9372	0,0125	33,4328
	pUCccuSWA[	0,9592	-0,0243	0,9430	-0,0221	33,6982
	lizNLJORkX	0,9583	0,0110	0,9351	0,0303	33,6492
	eASybBkOFc	0,9575	-0,0482	0,9371	-0,0412	33,5221
	ba[XycZCmy	0,9588	0,0145	0,9408	0,0270	33,6899
	Ldj^hbSals	0,9580	-0,0250	0,9408	-0,0085	33,4162
	i[w^BbufIk	0,9585	0,0209	0,9382	0,0073	33,7945
	zB`cXTiMAV	0,9588	-0,0528	0,9339	-0,0387	33,5877
	pzGVdalNCq	0,9576	-0,0331	0,9375	-0,0246	33,5731
	mC_GPXPDW^	0,9570	-0,0620	0,9368	-0,0589	33,4283

Tabel A.22 Tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
1	NaN	NaN	NaN	NaN	314,4413	NaN
2	NaN	NaN	NaN	NaN	54,0846	NaN
3	0,8716	-0,1635	0,9043	0,0380	57,0782	-0,0327
4	0,9760	0,5878	0,9736	0,5185	54,0851	0,0025
5	0,8356	-0,6738	0,8039	-0,6228	52,3319	0,0317
6	0,9340	0,3488	0,9364	0,4149	51,0800	-0,0024
7	0,8939	-0,4092	0,9060	-0,4389	54,0848	-0,0121
8	0,9155	-0,0172	0,9194	0,0220	52,3398	-0,0040
9	0,9275	-0,0746	0,9191	-0,0857	50,1096	0,0084
10	0,9225	-0,1834	0,9236	-0,2044	50,1217	-0,0010
11	0,9252	-0,0346	0,9250	-0,0865	49,3500	0,0002
12	0,8948	-0,1394	0,8962	-0,1381	47,6017	-0,0015
13	0,9210	-0,3415	0,9213	-0,3629	49,3355	-0,0003
14	0,9239	0,2667	0,9258	0,2654	46,7268	-0,0019
15	0,9557	0,4605	0,9519	0,5088	47,1341	0,0038
16	0,9364	0,0864	0,9369	0,0580	47,5960	-0,0005
17	0,9396	0,1362	0,9371	0,1568	47,6041	0,0026
18	0,9389	0,1903	0,9303	0,2100	48,1013	0,0086
19	0,9174	0,1418	0,9200	0,1426	48,1081	-0,0025
20	0,9361	0,4416	0,9344	0,4697	49,3453	0,0017
21	0,8976	-0,1579	0,9023	-0,1586	46,3611	-0,0048
22	0,9167	0,0869	0,9056	0,0578	47,5996	0,0111
23	0,9043	-0,1881	0,8986	-0,2166	46,0566	0,0057
24	0,9067	-0,2002	0,9056	-0,1827	46,0216	0,0011
25	0,9189	0,0544	0,9097	0,0382	46,3522	0,0092
26	0,9135	-0,0800	0,9079	-0,0831	45,7265	0,0056
27	0,9255	0,0936	0,9260	0,0411	45,7091	-0,0005
28	0,8987	-0,1366	0,8989	-0,1415	44,3791	-0,0002
29	0,8784	0,2202	0,8782	0,2240	46,7332	0,0002
30	0,9326	0,1202	0,9320	0,0713	46,4087	0,0006
31	0,9425	0,2299	0,9413	0,2135	45,3942	0,0012
32	0,8536	-0,3714	0,8565	-0,3481	45,6920	-0,0029
33	0,9128	-0,3031	0,9119	-0,2610	44,3999	0,0008
34	0,8738	-0,0221	0,8663	-0,0188	45,6960	0,0075
35	0,8580	-0,1548	0,8625	-0,1863	46,0191	-0,0045
36	0,8365	-0,0234	0,8385	-0,0216	45,4054	-0,0021
37	0,8778	-0,1560	0,8782	-0,1189	45,1673	-0,0004
38	0,8386	0,0141	0,8313	0,0213	44,8781	0,0074
39	0,8520	-0,2578	0,8607	-0,2795	44,8939	-0,0087
40	0,8723	-0,0128	0,8733	0,0196	43,8020	-0,0011
41	0,7892	-0,1431	0,7860	-0,1457	45,4130	0,0032

Tabel A.23 Lanjutan pertama tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
42	0,8869	0,0868	0,8855	0,0890	43,9467	0,0013
43	0,7842	-0,1979	0,7839	-0,1949	46,0340	0,0003
44	0,8002	0,1717	0,7961	0,1664	43,7912	0,0041
45	0,8835	0,1466	0,8768	0,1429	43,9857	0,0067
46	0,8280	0,0458	0,8294	0,0347	43,9929	-0,0014
47	0,7920	-0,0550	0,7909	-0,0647	42,6232	0,0011
48	0,8421	0,2341	0,8390	0,2121	42,5948	0,0031
49	0,8432	-0,2027	0,8434	-0,2239	42,7336	-0,0002
50	0,8379	0,0812	0,8349	0,0846	44,6358	0,0029
51	0,8848	0,3153	0,8830	0,3060	43,4169	0,0018
52	0,8662	0,2846	0,8685	0,2774	43,7684	-0,0022
53	0,8838	0,0473	0,8834	0,0568	43,7978	0,0003
54	0,8447	0,0545	0,8444	0,0585	43,7534	0,0003
55	0,8738	-0,0155	0,8710	-0,0204	42,7252	0,0028
56	0,8799	-0,0388	0,8819	-0,0406	43,4156	-0,0020
57	0,8640	0,2556	0,8591	0,2355	43,9892	0,0049
58	0,8553	0,0442	0,8536	0,0360	41,9780	0,0018
59	0,8054	-0,0278	0,8061	-0,0293	42,7316	-0,0007
60	0,8202	0,0854	0,8140	0,0863	42,8243	0,0062
61	0,8393	-0,1142	0,8434	-0,1230	43,3978	-0,0041
62	0,8777	0,3496	0,8856	0,3405	41,6679	-0,0080
63	0,8317	-0,2399	0,8325	-0,2315	42,3064	-0,0007
64	0,8530	-0,0216	0,8532	-0,0378	42,8385	-0,0002
65	0,8993	0,1945	0,9005	0,2062	43,0870	-0,0011
66	0,8569	-0,2618	0,8482	-0,2523	41,7668	0,0087
67	0,8468	-0,0479	0,8429	-0,0622	42,4410	0,0039
68	0,7945	-0,0942	0,7942	-0,1112	41,3401	0,0004
69	0,8779	-0,0495	0,8713	-0,0684	42,2059	0,0066
70	0,8689	-0,1170	0,8675	-0,1023	41,5380	0,0014
71	0,8899	0,1403	0,8888	0,1310	41,8045	0,0011
72	0,8756	0,1684	0,8704	0,1553	41,9380	0,0052
73	0,8450	0,0517	0,8419	0,0700	42,0675	0,0031
74	0,8858	0,0751	0,8866	0,0823	41,3223	-0,0008
75	0,8729	-0,0457	0,8736	-0,0578	42,0218	-0,0008
76	0,8392	-0,0795	0,8395	-0,0847	40,5435	-0,0003
77	0,8531	0,0263	0,8501	0,0106	41,1022	0,0030
78	0,8171	0,0158	0,8141	0,0065	42,0434	0,0030
79	0,8982	0,1825	0,8978	0,1794	42,1943	0,0003
80	0,8900	0,0861	0,8869	0,1008	41,0476	0,0031
81	0,8517	-0,2136	0,8474	-0,2240	40,6762	0,0042
82	0,8740	-0,0878	0,8705	-0,0896	41,5535	0,0035

Tabel A.24 Lanjutan kedua tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
83	0,8776	-0,0556	0,8696	-0,0558	41,0118	0,0080
84	0,8760	-0,0279	0,8666	-0,0346	40,4723	0,0094
85	0,8315	0,0451	0,8316	0,0336	40,6374	-0,0001
86	0,8452	-0,1879	0,8391	-0,1947	41,2637	0,0062
87	0,8149	-0,0533	0,8132	-0,0464	42,0355	0,0016
88	0,8543	-0,0704	0,8530	-0,0724	41,4802	0,0013
89	0,8930	0,0705	0,8857	0,0655	41,1666	0,0073
90	0,8702	-0,1695	0,8700	-0,1540	40,5943	0,0002
91	0,8980	0,2258	0,8964	0,2341	40,7381	0,0016
92	0,8369	0,0238	0,8379	0,0060	41,1639	-0,0009
93	0,8400	-0,1345	0,8342	-0,1460	41,1376	0,0058
94	0,8780	-0,1194	0,8732	-0,1217	40,5342	0,0048
95	0,8371	-0,1814	0,8380	-0,1696	40,3196	-0,0009
96	0,8697	0,0727	0,8677	0,0918	40,5396	0,0020
97	0,8082	0,0204	0,8076	0,0189	40,2631	0,0006
98	0,8231	-0,0286	0,8221	-0,0220	40,3493	0,0010
99	0,8292	-0,1852	0,8330	-0,1692	39,7822	-0,0038
100	0,8614	-0,0010	0,8656	0,0047	39,8698	-0,0041
101	0,8772	-0,0099	0,8720	-0,0060	40,1319	0,0052
102	0,8557	0,0128	0,8518	0,0321	41,0022	0,0039
103	0,8562	0,0134	0,8518	0,0232	39,5638	0,0044
104	0,8862	0,1742	0,8820	0,1716	40,3365	0,0042
105	0,7737	-0,0628	0,7781	-0,0787	39,6285	-0,0044
106	0,8552	0,0728	0,8519	0,0916	40,1776	0,0033
107	0,7981	0,0646	0,7942	0,0634	39,7483	0,0039
108	0,8514	-0,1048	0,8453	-0,0977	40,9054	0,0060
109	0,8271	0,0205	0,8270	0,0162	40,4189	0,0001
110	0,8655	-0,0025	0,8604	-0,0086	40,3126	0,0051
111	0,7987	-0,2792	0,7945	-0,2798	39,8328	0,0043
112	0,8683	0,0318	0,8693	0,0327	40,0501	-0,0010
113	0,8532	-0,0822	0,8492	-0,0827	40,2105	0,0040
114	0,9024	0,1749	0,9017	0,1675	39,7701	0,0007
115	0,8231	-0,0332	0,8267	-0,0229	39,7505	-0,0036
116	0,8297	-0,1236	0,8293	-0,1246	39,4849	0,0004
117	0,8670	-0,0285	0,8625	-0,0380	40,1393	0,0044
118	0,7971	-0,1611	0,8014	-0,1663	40,2485	-0,0043
119	0,8314	0,0031	0,8307	0,0157	40,0805	0,0007
120	0,8524	-0,0003	0,8451	-0,0082	40,3888	0,0073
121	0,8723	0,0377	0,8670	0,0105	39,8972	0,0053
122	0,8245	-0,0748	0,8262	-0,0798	40,1699	-0,0017
123	0,8562	-0,0337	0,8541	-0,0253	40,3570	0,0021

Tabel A.25 Lanjutan ketiga tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
124	0,8450	0,1030	0,8422	0,1128	39,7037	0,0028
125	0,8648	-0,0509	0,8573	-0,0632	38,9893	0,0075
126	0,8420	-0,1400	0,8376	-0,1300	39,7783	0,0044
127	0,8552	0,0844	0,8533	0,0768	39,4965	0,0019
128	0,8390	-0,0843	0,8372	-0,0756	38,7570	0,0018
129	0,8590	0,0183	0,8573	0,0165	40,3257	0,0017
130	0,8572	-0,0299	0,8514	-0,0249	40,0190	0,0058
131	0,8479	-0,0477	0,8503	-0,0378	39,2254	-0,0024
132	0,8307	0,0417	0,8282	0,0336	38,9908	0,0026
133	0,8772	0,0305	0,8771	0,0328	38,6639	0,0001
134	0,8868	0,0473	0,8845	0,0350	39,1242	0,0023
135	0,8559	0,0439	0,8513	0,0260	39,8030	0,0046
136	0,8806	0,0526	0,8779	0,0578	39,1655	0,0027
137	0,8646	-0,0348	0,8634	-0,0419	39,1365	0,0012
138	0,8766	0,0297	0,8746	0,0408	39,6685	0,0020
139	0,8574	-0,0896	0,8528	-0,0841	38,9395	0,0046
140	0,8621	-0,0072	0,8614	0,0028	38,4208	0,0008
141	0,8276	-0,0349	0,8281	-0,0486	38,5312	-0,0005
142	0,8496	0,1165	0,8506	0,1180	38,8042	-0,0010
143	0,8699	0,0020	0,8685	0,0094	38,5235	0,0014
144	0,8853	-0,0537	0,8822	-0,0702	39,0099	0,0031
145	0,8262	-0,0179	0,8248	-0,0135	38,5695	0,0014
146	0,8681	-0,1470	0,8648	-0,1322	38,5815	0,0033
147	0,8627	0,0483	0,8606	0,0427	38,7076	0,0021
148	0,8393	-0,0329	0,8372	-0,0269	38,3829	0,0022
149	0,8563	-0,0557	0,8546	-0,0675	38,6410	0,0017
150	0,8404	-0,1098	0,8350	-0,1119	38,6935	0,0054
151	0,8606	0,0704	0,8609	0,0677	39,0335	-0,0003
152	0,8281	-0,1902	0,8272	-0,1882	39,1438	0,0009
153	0,8706	-0,0277	0,8701	-0,0435	38,1493	0,0005
154	0,8576	0,1073	0,8576	0,1222	38,3598	0,0001
155	0,8763	0,0550	0,8766	0,0418	38,7157	-0,0002
156	0,8437	0,1087	0,8433	0,1083	38,2114	0,0004
157	0,8766	0,1860	0,8736	0,1840	37,8732	0,0031
158	0,8617	0,1226	0,8630	0,1300	38,3072	-0,0013
159	0,8615	0,0371	0,8613	0,0446	38,2719	0,0003
160	0,8123	-0,0710	0,8118	-0,0708	38,1267	0,0005
161	0,8653	-0,0151	0,8630	-0,0087	38,1754	0,0023
162	0,8315	-0,0433	0,8323	-0,0337	38,1208	-0,0009
163	0,8116	-0,0191	0,8111	-0,0067	38,5998	0,0005
164	0,8878	0,0655	0,8840	0,0718	38,8584	0,0038

Tabel A.26 Lanjutan keempat tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
165	0,8695	0,1018	0,8681	0,0910	38,1107	0,0014
166	0,8459	0,0536	0,8474	0,0611	38,7076	-0,0015
167	0,8320	0,0697	0,8311	0,0621	38,5483	0,0009
168	0,8389	-0,0278	0,8360	-0,0169	38,1293	0,0029
169	0,8300	-0,0788	0,8258	-0,0879	37,9567	0,0042
170	0,8412	-0,0717	0,8392	-0,0682	38,3260	0,0020
171	0,8625	-0,0258	0,8641	-0,0069	38,0683	-0,0016
172	0,8235	-0,0870	0,8226	-0,0961	37,9562	0,0008
173	0,8321	-0,0439	0,8313	-0,0403	38,4333	0,0008
174	0,8694	0,0026	0,8686	0,0196	38,2451	0,0007
175	0,8190	-0,0741	0,8217	-0,0772	38,0744	-0,0027
176	0,8583	0,0518	0,8559	0,0460	37,8372	0,0024
177	0,8462	-0,0407	0,8442	-0,0334	37,8090	0,0021
178	0,8463	0,0506	0,8443	0,0551	37,9383	0,0019
179	0,8695	0,1556	0,8694	0,1463	37,6366	0,0001
180	0,8551	0,0197	0,8534	0,0279	38,3053	0,0017
181	0,8445	-0,0285	0,8437	-0,0370	37,8628	0,0009
182	0,8604	0,0771	0,8583	0,0865	38,0486	0,0020
183	0,8476	0,0561	0,8488	0,0611	37,8381	-0,0012
184	0,8371	-0,0113	0,8328	-0,0020	38,2876	0,0043
185	0,8386	-0,0046	0,8337	0,0009	37,8434	0,0050
186	0,8801	0,1713	0,8797	0,1870	38,2323	0,0004
187	0,8106	-0,1529	0,8081	-0,1406	37,9710	0,0025
188	0,8661	-0,0553	0,8631	-0,0528	37,5025	0,0029
189	0,8460	0,0846	0,8460	0,0722	37,7446	0,0000
190	0,8887	0,0823	0,8863	0,0806	37,8288	0,0024
191	0,8321	0,0441	0,8292	0,0522	37,3757	0,0029
192	0,8475	-0,0645	0,8443	-0,0605	37,4842	0,0032
193	0,8613	0,0062	0,8584	0,0170	38,0823	0,0029
194	0,8636	0,0412	0,8624	0,0397	37,5116	0,0011
195	0,8701	-0,0726	0,8681	-0,0678	37,6950	0,0020
196	0,8299	0,0365	0,8335	0,0452	37,7742	-0,0036
197	0,8530	0,0677	0,8518	0,0537	37,7445	0,0013
198	0,8271	-0,0469	0,8254	-0,0445	37,4200	0,0017
199	0,8429	-0,0053	0,8396	-0,0202	37,7032	0,0033
200	0,8365	0,0215	0,8322	0,0234	37,2110	0,0043
201	0,8352	0,0130	0,8348	0,0058	37,2222	0,0004
202	0,8143	0,0436	0,8142	0,0443	37,1728	0,0001
203	0,8703	-0,0037	0,8667	-0,0005	37,2762	0,0037
204	0,8732	0,1710	0,8721	0,1832	36,9645	0,0011
205	0,8743	0,0885	0,8722	0,0892	37,3843	0,0021

Tabel A.27 Lanjutan kelima tabel uji coba ukuran kolom *watermark*

Panjang *Watermark*	Korelasi				PSNR	Perbe-daan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
206	0,8374	0,0340	0,8392	0,0293	37,5569	-0,0018
207	0,8601	-0,0062	0,8617	0,0034	37,2831	-0,0016
208	0,8508	0,0211	0,8482	0,0178	36,9143	0,0027
209	0,8222	-0,1031	0,8193	-0,1140	37,7585	0,0029
210	0,8500	-0,0260	0,8492	-0,0154	36,7982	0,0008
211	0,8610	0,1031	0,8560	0,1098	37,3453	0,0050
212	0,8321	-0,0792	0,8313	-0,0663	37,5872	0,0007
213	0,8677	0,0344	0,8645	0,0384	37,1872	0,0032
214	0,8488	-0,0249	0,8462	-0,0251	37,2100	0,0026
215	0,8585	-0,1351	0,8523	-0,1222	37,4025	0,0062
216	0,8680	0,0037	0,8661	0,0061	37,3126	0,0019
217	0,8170	-0,1749	0,8144	-0,1759	36,6225	0,0026
218	0,8229	-0,0917	0,8164	-0,0910	37,5005	0,0065
219	0,8590	0,0372	0,8569	0,0491	37,3447	0,0020
220	0,8607	-0,0209	0,8595	-0,0345	37,7533	0,0012
221	0,8604	-0,0599	0,8572	-0,0548	36,9928	0,0032
222	0,8428	-0,1136	0,8399	-0,1185	37,3008	0,0029
223	0,8257	-0,0616	0,8248	-0,0682	36,5543	0,0009
224	0,8265	-0,0727	0,8235	-0,0798	36,8940	0,0029
225	0,8286	-0,0970	0,8266	-0,0958	37,0984	0,0020
226	0,8465	-0,0612	0,8463	-0,0750	36,7238	0,0002
227	0,8025	-0,0170	0,8025	-0,0259	37,1362	0,0000
228	0,8081	-0,0288	0,8048	-0,0242	36,4760	0,0033
229	0,8296	0,1152	0,8282	0,1086	36,7490	0,0014
230	0,8436	0,0523	0,8446	0,0435	36,6273	-0,0010
231	0,8147	0,1230	0,8106	0,1215	36,9909	0,0041
232	0,8040	-0,0938	0,8011	-0,0924	36,5546	0,0029
233	0,8200	0,0014	0,8166	-0,0049	36,4759	0,0035
234	0,8275	0,0502	0,8257	0,0363	36,9990	0,0018
235	0,7841	-0,0969	0,7832	-0,0899	37,2779	0,0009
236	0,8234	0,0193	0,8247	0,0282	37,5972	-0,0014
237	0,8200	0,0082	0,8205	0,0115	37,0427	-0,0005
238	0,7612	0,0154	0,7586	0,0114	36,6718	0,0026
239	0,8046	-0,0692	0,8046	-0,0689	37,2929	0,0000
240	0,8188	0,0064	0,8157	0,0132	36,4367	0,0031
241	0,8174	0,0306	0,8160	0,0186	36,4964	0,0013
242	0,8062	0,0304	0,8017	0,0307	36,6115	0,0044
243	0,8046	0,0503	0,8010	0,0477	36,3155	0,0036
244	0,8393	0,0180	0,8382	0,0208	36,2678	0,0011
245	0,8379	0,0867	0,8384	0,0892	36,8291	-0,0005
246	0,8334	-0,1220	0,8300	-0,1209	36,7408	0,0034

Tabel A.30 Lanjutan tabel uji coba ukuran baris *watermark*

Lebar *Watermark*	Korelasi				PSNR	Perbedaan
	Normal		Jpg 70 Persen			
	Ber-*watermark*	Asli	Ber-*watermark*	Asli		
28	0,9368	0,0589	0,9247	0,1035	45,7190	0,0122
29	0,9435	0,1604	0,9410	0,1586	45,4423	0,0025
30	0,9372	-0,0061	0,9342	0,0026	45,1254	0,0029
31	0,9429	0,1387	0,9293	0,1005	46,0467	0,0136
32	0,9450	0,1764	0,9298	0,1507	45,7174	0,0152
33	0,9384	0,0269	0,9144	0,0506	45,1195	0,0240
34	0,9359	-0,1391	0,9221	-0,1800	45,4086	0,0138
35	0,9350	-0,2077	0,9200	-0,2001	45,4005	0,0150
36	0,9307	-0,2461	0,8995	-0,2155	45,6948	0,0312
37	0,9318	-0,2267	0,8977	-0,2194	43,5572	0,0341
38	0,9412	-0,0164	0,9078	-0,0315	44,3922	0,0335
39	0,9415	-0,0336	0,9009	-0,0207	44,3958	0,0406
40	0,9514	0,2044	0,9339	0,1894	44,8549	0,0175
41	0,9369	-0,2817	0,8985	-0,2839	44,1793	0,0384
42	0,9458	0,1247	0,8875	0,0718	42,8936	0,0583
43	0,9392	-0,1393	0,8958	-0,1497	43,0383	0,0434
44	0,9415	-0,0838	0,8659	-0,0732	43,0342	0,0756
45	0,9421	-0,2015	0,8355	-0,1253	43,7668	0,1066
46	0,9466	0,0247	0,8552	0,0318	43,8061	0,0914
47	0,9426	-0,0388	0,8828	-0,0075	44,1742	0,0598
48	0,9529	0,2313	0,8395	0,2818	44,6289	0,1134
49	0,9470	-0,0464	0,8241	-0,0403	43,5774	0,1229
50	0,9497	0,1278	0,8255	0,0746	43,0221	0,1241
51	0,9422	-0,0746	0,7874	-0,1047	42,1721	0,1548