Watermarking Citra Digital Menggunakan Teknik Amplitude Modulation

Semuil Tjiharjadi, Sanwill
Jurusan Teknik Elektro
Fakultas Teknik, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
Email: semuil.tjiharjadi@eng.maranatha.edu

Abstract

Watermarking is a way to save some information into other digital data and the existences of information cannot be realized by human sense. It can also survive from digital signal processing in adequate processing. There are some applications of watermarking, for examples Broadcast Monitoring, Owner Identification, Proof of Ownership, Authentication, Fingerprinting, Copy control and Covert Communication. Some techniques, which have already been developed, are LSB, Amplitude Modulation, Patchwork, Spread Spectrum etc.

This report will describe the Bitmap image format, the concept of watermarking, the concept of watermarking using Amplitude Modulation technique, the realization of watermarking concept through a watermarking program, and the test results of the program.

Through some tests, which have been done to the digital image watermarking program using amplitude modulation technique, it can be concluded that the program has been realized successfully. The test also shows that the increment of the value of q, make the ability of program to identify the characters, which were watermarked, to the image become better.

Keywords: Watermarking, Digital imaging, Amplitude modulation, Digital Signal Processing, Broadcast Monitoring, Owner Identification, Proof of Ownership, Authentication, Fingerprinting, Copy control, Covert Communication

1. Pendahuluan

W*atermarking* merupakan suatu cara penyembunyian atau penyimpanan informasi tertentu ke dalam suatu data digital lainnya, tetapi tidak diketahui kehadirannya oleh indera manusia dan mampu menghadapi proses-proses pengolahan sinyal digital sampai pada tahap tertentu. Beberapa aplikasi *watermarking* antara lain, *Broadcast Monitoring, Owner Identification, Proof of Ownership, Authentication, Fingerpriting, Copy control,* dan *Covert Communication*.

Teknik-teknik *watermarking* untuk citra digital yang telah dikembangkan, misalnya LSB, *Amplitude Modulation, Patchwork, Spread Spectrum* dan lain-lain. Dalam makalah ini penulis akan merealisasikan konsep *watermarking* melalui sebuah program *watermarking* yang menggunakan teknik *Amplitude Modulation*.

2. Citra Digital

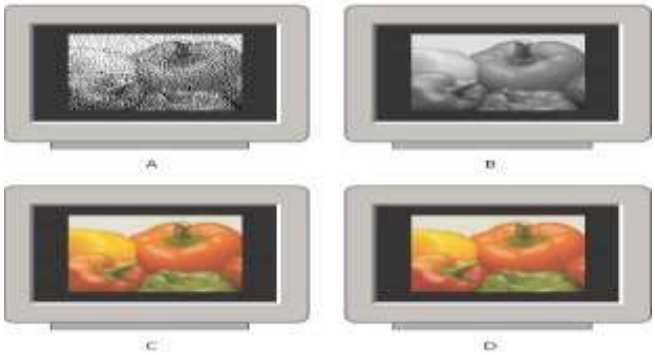
2.1. Vektor dan Bitmap

Citra digital dalam komputer grafis terbagi menjadi 2 kategori, yaitu bitmap

dan vektor. Sebuah citra digital yang berbasis vektor (citra vektor/*vector image*) merupakan kumpulan garis dan kurva yang didefinisikan oleh persamaan matematika yang disebut vektor-vektor (*vectors*). Karena citra vektor tersusun dari vektor-vektor, maka citra tersebut dapat dimanipulasi tanpa kehilangan kualitas. Citra vektor merupakan jenis citra yang *resolution-independent*, artinya citra yang tidak bergantung pada resolusi. Citra jenis ini dapat diperbesar dan dicetak tanpa kehilangan kualitasnya. Namun citra vektor memerlukan banyak perhitungan matematika dalam proses.

Citra digital bitmap (citra bitmap) secara teknis sering disebut sebagai *raster images*. Citra digital jenis ini tersusun atas titik-titik yang disebut piksel (*pixel*). Manipulasi pada citra bitmap adalah manipulasi yang dilakukan pada tingkat piksel. Citra bitmap banyak digunakan pada media elektronik yang berbasis pada warna yang kontinyu atau citra dengan gradasi warna, misalnya dalam dunia fotografi digital ataupun menggambar digital.

Citra bitmap adalah citra digital yang *resolution-dependent*, artinya citra yang bergantung pada resolusi. Resolusi adalah jumlah piksel persatuan luas, misalnya 72 piksel/inci^2 (72 piksel/inci^2) yang merupakan ukuran standar dari citra bitmap. Sebuah citra bitmap tersusun atas piksel-piksel yang posisinya tertentu dengan nilai kedalaman warna tertentu. Jika resolusi piksel terlalu rendah maka citra bitmap tampak bergerigi. Dalam dunia komputer grafis, citra bitmap murni ditulis dengan ekstension bmp (*.bmp), misal pemandangan.bmp.



Gambar 1. Kedalaman Warna. A. 1-bit. B. 8-bit (Grayscale).
C. 8-bit (Indexed Color). D. 24-bit (RGB).

2.2. Watermarking

W*atermark* merupakan cabang ilmu dari *steganography*. *Steganography* merupakan ilmu yang mempelajari tentang bagaimana menyembunyikan suatu informasi rahasia di dalam suatu informasi lainnya. *Steganography* mempunyai kemiripan dengan *cryptograhpy.* Perbedaannya terletak pada proses penyembunyian data dan hasil akhir dari proses tersebut. *Cryptography* melakukan proses pengacakan (enkripsi) data aslinya sehingga menghasilkan data hasil enkripsi yang benar-benar acak dan berbeda dengan aslinya, sedangkan *stegranography* menyembunyikan dalam data lain yang akan ditumpanginya tanpa mengubah data yang ditumpanginya tersebut sehingga data yang ditumpanginya sebelum dan setelah proses penyembunyian hampir sama.

2.2.1. *Watermarking* Citra Digital Teknik A*mplitude Modulation*

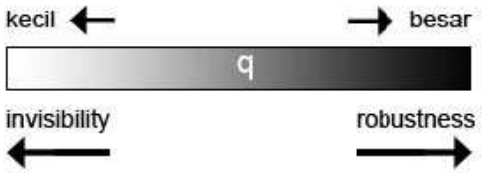
Pada watermarking yang menggunakan teknik *Amplitude Modulation* (*watermarking* AM), informasi yamg disimpan berupa data bit. Data bit ini disimpan di citra digital dengan memodifikasi nilai dari kanal biru, karena mata manusia kurang sensitif terhadap warna biru. Modifikasi ini bisa berupa penjumlahan ataupun pengurangan nilai warna, tergantung bit yang disimpan. Proses *decoding* (*retrieve*) data tidak memerlukan citra asli yang belum dimodifikasi/belum di-*watermark*.

Untuk peyimpanan informasi yang berupa karakter, informasi tersebut harus diubah dulu ke kode biner, dalam bentuk bit-bit, agar bisa di-*watermark* ke citra. Misal, *s* adalah bit tunggal yang ingin di-*watermark* ke citra *I*, dengan *I= {R,G,B}*, dan *p = (x,y)* adalah titik kartesian pada citra *I*. Bit di-*watermark* ke citra pada posisi *p* di kanal biru dengan memodifikasi tingkat keabuan citra (*L = lb= Luminance*)

$$L = lb = 0.299\ R + 0.587\ G + 0.114\ B \quad \text{..... (1)}$$

$$B_{x,y} = B_{x,y} + (2s-1) \times q \times lb \quad \text{..... (2)}$$

Nilai *q* adalah nilai faktor kualitas. Nilai *q* menunjukkan tingkat kekuatan dari *watermark*. Nilai *q* dipilih berdasarkan tujuan dari proses penyimpanan data pada citra tersebut. Dengan mengatur nilai *q*, dapat dipilih apakah citra lebih mengutamakan perlindungan atas deteksi (*invisibility*) ataupun perlindungan atas modifikasi (*robustness*).



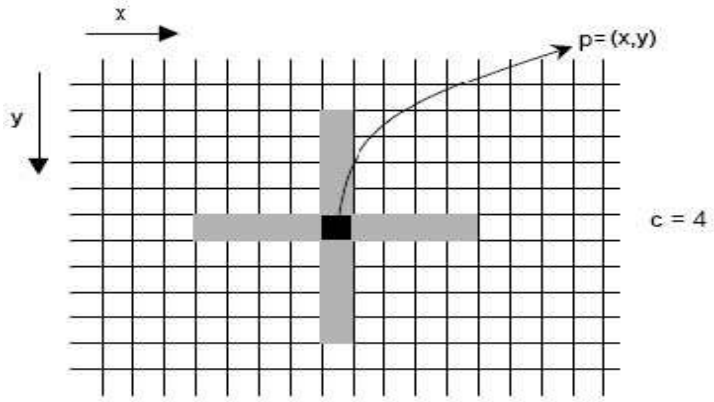
Gambar 2. Pengaruh Nilai *q* terhadap *invisibility* dan *robustness*.

Jika nilai *q* kecil maka perubahan akibat proses *watermarking* pada citra lebih sulit terdeteksi oleh mata, *invisibility* lebih baik, namun *robustness* lebih buruk. *Robustness* yang buruk menyebabkan kemungkinan terjadinya kesalahan deteksi bit s pada proses *retrieve* menjadi lebih besar dan data yang di-*watermark* ke citra lebih mudah rusak/berubah jika citra dimodifikasi.. Jika nilai *q* besar maka perubahan akibat proses *watermarking* pada citra lebih mudah terdeteksi oleh mata, *invisibility* lebih buruk, namun *robustness* lebih baik. Contoh modifikasi pada citra adalah *bluring* (mengaburkan citra), *rotate* (memutar citra) dan lain-lain. Ini adalah proses *encoding* atau *watermarking.*

Proses *decoding* atau *retrieve* dilakukan dengan memprediksi nilai bit yang di-*watermark* ke citra. Prediksi tersebut berdasarkan kombinasi linear antara piksel-piksel kanal biru yang berada disekitar *p*. Daerah sekitar yang berbentuk silang memberikan hasil yang terbaik. Nilai prediksi kanal biru $\hat{B}_{xy}$ dihitung dengan persamaan berikut.

$$\hat{B}_{xy} = \frac{1}{4c}\left(\sum_{k=-c}^{c} B_{x+k,y} + \sum_{k=-c}^{c} B_{x,j+y} - 2B_{xy} \right) \quad \text{.....(3)}$$

Nilai c menunjukkan jangkauan prediksi. Perhatikan Gambar 3 pada daerah sekitar piksel yang berbentuk silang



Gambar 3. Daerah Sekitar Piksel yang Berbentuk Silang

Untuk mendapatkan bit yang di-*watermark*, beda nilai antara B_{xy} dan $\hat{B}_{xy}$, δ dihitung dengan menggunakan persamaan berikut.

$$\delta = B_{xy} - \hat{B}_{xy} \quad \text{..... (4)}$$

Nilai δ merupakan nilai bit yang di-*watermark* ke citra.

2.2.1.1. *Multiple Embedding*

Proses *watermark* dan proses *retrieve* bit pada *watermarking* AM merupakan proses yang asimetris, karena persamaan untuk proses *retrieve* bukan *inverse* (kebalikan) dari proses *watermark*. Akibatnya terdapat kemungkinan terjadinya kesalahan indentifikasi nilai bit pada proses *retrieve*. Untuk mengatasinya digunakan metode *multiple embedding*.

Prinsip kerja metode *multiple embedding* adalah me-*watermark* atau menyisipkan bit yang sama ke citra sebanyak *n* kali. Lokasi *p* untuk tiap penyisipan ke-*k* berbeda. Pada proses *retrieve* yang menggunakan metode *multiple embedding*.

$$\delta_k = B_{pk} - \hat{B}_{pk} \quad \text{..... (5)}$$

$k = 1,2,3,\ldots n$

Nilai δ_k adalah beda nilai antara B dan $\hat{B}$ pada penyisipan ke-k. Nilai B_{pk} adalah nilai B pada posisi p pada penyisipan ke-k. Nilai $\hat{B}_{pk}$ adalah nilai B_{xy} pada posisi p pada penyisipan ke-k. Setelah mendapatkan nilai δ untuk seluruh penyisipan, nilai δ_k dirata-rata.

$$\bar{\delta} = \frac{1}{n}\sum_{k} \delta_k \quad \text{..... (6)}$$

Untuk mendapatkan hasil maksimal δ dibandingkan dengan nilai threshold. Nilai

threshold didapat dengan menghitung rata-rata nilai δ^0 dan δ^1. Nilai δ^0 didapat jika seluruh lokasi piksel p pada citra diisi dengan bit 0. Nilai δ^1 didapat jika seluruh lokasi piksel p pada citra diisi dengan bit 1.

$$\bar{\delta}^0 = \frac{1}{|I|n} \sum_i \delta_i^0 \quad \text{.....(7)}$$

$$\bar{\delta}^1 = \frac{1}{|I|n} \sum_i \delta_i^1 \quad \text{..... (8)}$$

Nilai $|I|$ adalah banyaknya bit yang di-*watermark* ke citra *I* pada penyisipan ke *k*.

$$threshold = \frac{\bar{\delta}^0 + \bar{\delta}^1}{2} \quad \text{.....(9)}$$

Identifikasi bit dilakukan dengan menggunakan persamaan berikut,

$$\bar{s} = \begin{cases} 1, & If\ \bar{\delta} > threshold \\ 0, & lainnya \end{cases} \quad \text{..... (10)}$$

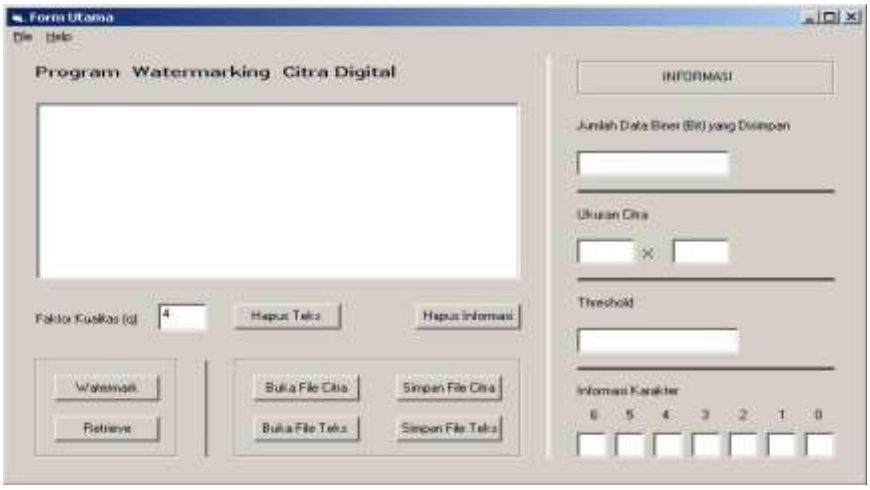
3. Perangkat lunak watermarking

3.1. Program *Watermarking*

Pada form utama, pengguna dapat memasukkan nilai kualitas (*q*), pada kotak teks di bawah label "Faktor Kualitas" dari proses *watermarking* dan data yang berupa teks ke kotak teks plainteks. Pengguna harus memasukkan citra digital yang akan di-*watermark* dengan menekan tombol "Buka File Citra" (cmdbukacitra). Setelah pengguna memilih citra yang ingin di-*watermark*, program akan menampilkan citra tersebut ke kotak citra (picbx) pada "Form Citra".

Ketika pengguna menekan tombol "Watermarking", maka program akan memeriksa apakah citra yang akan di-*watermark* telah tersedia di form "Form Citra". Jika citra belum tersedia program akan menghentikan proses eksekusi, menampilkan pesan "ERROR", berisi pesan "Tidak Ada Citra!"

Untuk memasukkan teks ke kotak teks plainteks, pengguna dapat memasukkannya dengan cara mengetikkan teks kotak teks plainteks atau dengan menggunakan tombol "Buka File Teks" untuk membuka file teks yang sudah ada. Jika teks belum tersedia program akan menghentikan perintah proses *watermarking*, mengeluarkan kotak pesan "ERROR", yang berisi pesan "Masukkan Teks!"



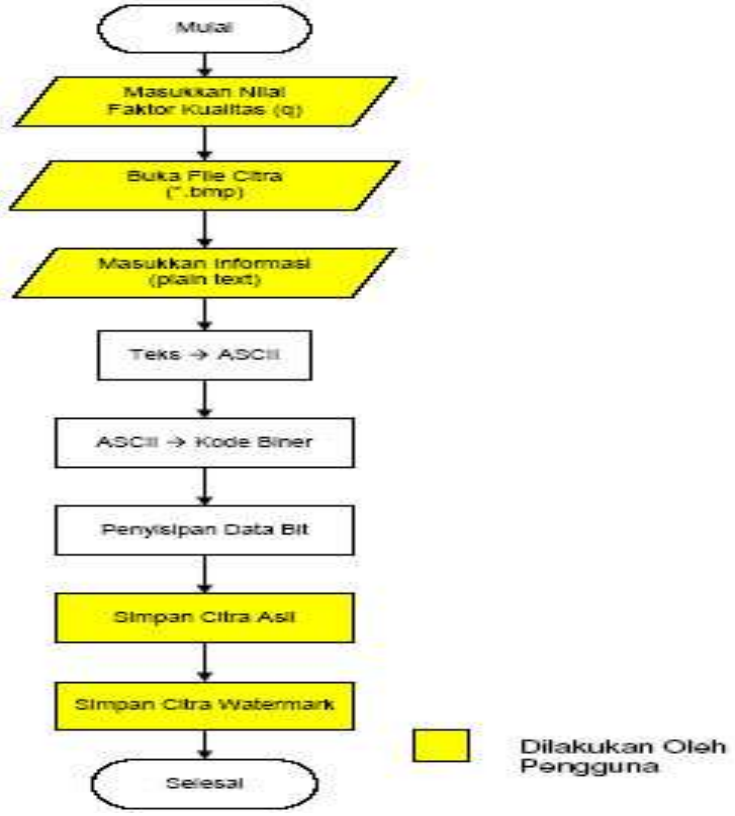
Gambar 4. Form utama program watermarking citra digital metoda AM

Setelah membuka file citra dan memasukkan data teks ke kotak teks plainteks, pengguna harus menekan tombol "Watermarking", untuk melakukan proses *watermarking*. Jika kotak teks *q* tidak diubah oleh pengguna, program akan menggunakan nilai *q default* yang terdapat pada teks boks, $q = 4$. Jika nilai *q* dihapus oleh pengguna, akan terjadi kesalahan ketika melakukan eksekusi perintah proses *watermarking* dan proses *retrieve*, namun karena ada perintah "On Error GoTo", program akan menghentikan proses eksekusi tersebut, membatalkan perintah tersebut dan menuju ke akhir perintah ("exit sub"), sehingga kesalahan tidak menyebabkan program *crash*. Secara umum langkah-langkah proses *watermarking* diperlihatkan oleh Gambar 5.

3.1.1. Proses Perubahan Data Teks ke Kode Biner

Setelah pengguna membuka citra, mengisi teks pada kotak teks plainteks, mengisi nilai *q*, dan menekan tombol "Watermarking", program akan melakukan eksekusi proses *watermarking*. Proses eksekusi dimulai dengan memindahkan citra pada kotak citra yang terdapat pada form citra ke variabel pixHDC, kemudian dilakukan penghitungan ukuran citra.

Perintah "sx = picwindow.picbx.ScaleWidth Mod 7" berfungsi untuk menghitung sisa dari ukuran citra pada sumbu kartesian x dibagi dengan 7. Perintah "nx = picwindow.picbx.ScaleWidth – sx", mengurangi ukuran citra pada sumbu kartesian x dengan variabel sx. Kedua perintah ini akan menghasilkan variabel nx yang merupakan kelipatan 7. Ukuran citra pada sumbu kartesian x (panjang) dibagi dengan 7, tujuannya adalah untuk memudahkan penyisipan bit-bit karakter karena sebuah bit karakter ASCII (American Standard Code for Informatin Exchange) terdiri dari 7 bit biner, sehingga dengan membagi panjang citra dengan 7, satu baris sumbu kartesian x dari citra dapat menampung sejumlah n karakter, n adalah bilangan bulat (1,2,3,4, …). Misalnya, ukuran citra pada sumbu kartesian x = 57, maka sx = 57 mod 7 =1, dan nx = x – sx = 56.



Gambar 5 Flowchart program utama

Perintah "sy = picwindow.picbx.ScaleHeight Mod 5" berfungsi untuk menghitung sisa dari ukuran citra pada sumbu kartesian y dibagi dengan 5. Perintah "pnjg = picwindow.picbx.ScaleHeight – sy", mengurangi ukuran citra pada sumbu kartesian y dengan variabel sy. Kedua perintah ini akan menghasilkan variabel pnjg yang merupakan kelipatan 5. Perintah "ny = pnjg / 5" akan membagi variabel pnjg dengan 5. Misalnya, ukuran citra pada sumbu kartesian y = 57, maka sx = 57 mod 5 = 2, dan pnjg = x – sx = 55, nx = 55 / 5 = 11.

Perintah "bitsize = nx * ny / 5" untuk menghitung variabel bitsize, yang merupakan jumlah bit pada citra yang dapat disisipi dengan data. Nilai variabel bitsize jauh lebih kecil dari ukuran citra sesungguhnya, karena ada alokasi tempat untuk melakukan *multiple embedding*. Pada Gambar 6 diperlihatkan ilustrasi dari sebuah citra yang telah dicari nilai nx, dan ny-nya. Daerah pada citra yang diberi nomor tersebut disebut sebagai sektor. Daerah tersebut adalah posisi piksel (0,0) ke (nx, ny) untuk sektor 1 , kemudian (0,ny + 5) ke (nx, 2ny) untuk sektor 2, begitu seterusnya hingga (0, 4ny + 5) ke (nx, 5ny) untuk sektor 5. Sektor 1 sampai 5 akan dimasukkan data bit yang sama, jadi 1 bit data akan dimasukkan sebanyak 5 kali kedalam citra digital. Inilah metode *multiple embedding* yang digunakan pada program ini.

Sesudah melakukan penghitungan ukuran citra, program akan memproses karakter yang telah dimasukkan ke kotak teks plainteks. Program akan memindahkan data pada kotak teks plainteks ke variabel *string* chtr. Kemudian menghitung jumlah karakter yang dimasukkan. Program akan membandingkan

variabel bitsize dengan jumlah karakter x 7, jumlah karakter harus dikali 7 karena sebuah bit karakter ASCII terdiri dari 7 bit biner. Jika variabel bitsize lebih besar proses *watermarking* akan diteruskan, jika lebih kecil program akan mengeluarkan kotak pesan "Jumlah karakter terlalu panjang!" dan proses *watermarking* akan dihentikan.

Gambar 6. Ilustrasi file citra digital yang telah dihitung nilai nx dan ny dan pembagian sektor

Langkah berikutnya, program akan mengubah data plainteks ke karakter ASCII. Karakter ASCII yang diijinkan untuk diproses adalah yang berada pada daerah, 10 ≤ASCII ≤126, karena di daerah tersebut terdapat karakter-karakter yang paling sering digunakan. Jika ada salah satu karakter dari data plainteks yang berada di luar daerah tersebut, maka program akan mengeluarkan kotak pesan " Ada karakter yang tidak dikenal!" dan proses *watermarking* akan dihentikan. Jika karakter berada pada daerah yang diijinkan, program akan meneruskan proses *watermarking* dengan mengubah karakter ASCII yang didapat ke kode biner dan disimpan ke *array* "arraybit( )". Variabel *array* arraybit( ) dapat menampung data *array* sebanyak variabel bitsize ditambah satu, karena array dimulai dari urutan ke nol, jadi jika variabel bitsize adalah 35, maka *array* arraybit( ) bisa menampung sebanyak 35 + 1, 36 data.

3.1.2. Proses Penyisipan Data Bit ke Citra Digital

Karena adanya proses *multiple embedding*, maka proses penyisipan data bit ke citra digital dilakukan berkali-kali. Dalam program ini data bit diw*atermark* sebanyak 5 kali.

Proses *watermark* dimulai dengan menentukan piksel mana saja yang akan dipakai sebagai piksel yang akan disisipi dengan data bit. Pola posisi piksel yang digunakan untuk sumbu kartesian x adalah dari piksel 0 hingga piksel nx dengan selisih 7 antar piksel. Pola posisi piksel yang digunakan untuk sumbu kartesian y adalah dari piksel 0 hingga piksel ny dengan selisih 5 antar piksel. Tujuan adanya selisih ini adalah agar tidak ada interferensi antar piksel yang disisipi informasi ketika proses *retrieve.*

Setelah menentukan pola posisi piksel, langkah selanjutnya adalah memindahkan informasi kanal warna dari piksel sesuai dengan pola posisi piksel ke variabel newcol dengan perintah "newcol = GetPixel(pixHdc, X, Y)". Kemudian menghitung nilai R, G, B ($r_{x,y}$, $g_{x,y}$, $b_{x,y}$) dari piksel tersebut dan menghitung tingkat keabuan dari kanal warna biru (*lb*). Setelah mendapatkan kanal biru dan

tingkat keabuan biru, program akan memodifikasi kanal biru dengan cara memasukkan nilai bit pada *array* arraybit( ) ke persamaan " $b_{x,y} = b_{x,y} + (2s - 1)\ x\ q\ x\ lb$", s bit adalah arraybit( ), $b_{x,y}$ adalah nilai yang merepsentasikan kanal biru pada posisi piksel (x,y) dan q adalah faktor kualitas. Nilai $b_{x,y}$, sama dengan nilai $B_{x,y}$,

(0,0)	(7,0)	(14,0)	…	(nx,0)
(0,5)	(7,5)	(14,5)	…	(nx,5)
(0,10)	(7,10)	(14,10)	…	(nx,10)
…	…	…	…	…
(0,ny)	(7, ny)	(14, ny)	…	(nx, ny)
(0,ny + 5)	(7, ny + 5)	(14, ny + 5)	…	(nx, ny + 5)
…	…	…	…	…
(0, 2ny)	(7, 2ny)	(14, 2ny)	…	(nx, 2ny)
(0,2ny + 5)	(7, 2ny + 5)	(14, 2ny + 5)	…	(nx, 2ny + 5)
…	…	…	…	…
(0,5ny)	(7, 5ny)	(14, 5ny)	…	(nx, 5ny)

Gambar 7. Matriks Pola Posisi Piksel Citra untuk Penyisipan Data Bit

Jika piksel yang tersedia lebih besar daripada jumlah arraybit yang ada, maka sisa tempat akan diisi dengan bit 1, karakter ASCII dari biner 1111111 adalah 127. Hal ini dilakukan untuk mempermudah identifikasi karakter ketika proses retrieve. Pada saat perubahan data teks ke karakter ASCII, karakter dibatasi antara 10 hingga 126 sehingga bisa dijamin program tidak akan menyisipkan 30 karakter ASCII 127 ke citra digital. Nilai $b_{x,y}$ yang telah dimodifikasi tidak boleh kurang dari 0 atau lebih dari 256, $0 \leq b_{x,y} \leq 256$, jika kurang dari nol nilai $b_{x,y}$ akan dibuat menjadi nol, jika lebih dari 256 nilai $b_{x,y}$ akan dibuat menjadi 256, begitupun dengan nilai $r_{x,y}$, dan $g_{x,y}$. Kemudian nilai $b_{x,y}$, $r_{x,y}$, $g_{x,y}$ akan dikembalikan lagi ke citra, dengan perintah "SetPixelV pixHdc, X, Y, RGB(r, g, b)".

Setelah selesai melakukan penyisipan data bit ke citra digital, program akan menampilkan kode biner dari karakter pertama (7 bit), ukuran citra dan jumlah bit informasi yang telah disisipkan ke citra ke kolom "INFORMASI".

3.2. Proses *Retrieve*

Untuk mulai melakukan proses *retrieve*, pengguna harus mengisi nilai faktor kualitas (q) dari citra yang akan di-*retrieve* datanya, nilai q ini harus sama dengan nilai q ketika data di-*watermark* ke citra, jika nilai q tidak sama, data yang di-*retrieve* dari citra tidak akan sama dengan data yang di-*watermark* ke citra. Jika kotak teks q tidak diubah oleh pengguna, program akan menggunakan nilai q *default* pada teks boks, $q = 4$.

Proses retrieve dimulai dengan menekan tombol "Retrieve". Program mula-mula akan meminta pengguna memilih file citra asli, yaitu file citra yang belum di-*watermark* (file citra asli), kemudian program akan melakukan penghitungan nilai *threshold*. Setelah selesai menghitung *threshold* program akan meminta pengguna memasukkan file citra yang telah di-*watermark* (file citra *watermark*). Kemudian program akan melakukan identifikasi karakter yang telah di-*watermark* ke citra, dan menampilkan hasilnya ke kotak teks plainteks. Jika pengguna tidak memiliki file citra asli pengguna dapat memasukkan file citra *watermark*.

3.2.1. Menghitung Nilai *Threshold*

Setelah proses "Retrieve" dan pemasukkan file citra asli selesai dilakukan pada proses terdahulu, maka file citra asli ini ditampilkan ke kotak citra pada form citra. Kemudian file citra tersebut dipindahkan ke variabel pixHDC.

Langkah selanjutnya adalah memindahkan informasi kanal warna dari piksel sesuai dengan pola posisi piksel ke variabel newcol dengan perintah "newcol = GetPixel(pixHdc, X, Y)". Kemudian menghitung nilai R, G, B ($r_{x,y}$, $g_{x,y}$, $b_{x,y}$) dari piksel dengan pola posisi piksel sama dengan proses penyisipan data bit pada proses *watermark* yaitu pada lokasi *p*. Kemudian tiap piksel terpakai tersebut disisipi dengan data biner 0,1,0,1,0,1,… (data 0101).

Proses penyisipan ini sama dengan proses penyisipan data bit pada proses *watermark*, bedanya tidak ada proses konversi data teks ke biner dan data yang akan disisipkan tadi sudah ditentukan sebelumnya yaitu data 0101. Tujuan penyisipan ini adalah untuk menghitung nilai ddari *y x b* , jika disisipi data bit 0 dan 1. Nilai d kemudian dirata-rata dan didapatlah nilai *threshold*. Fungsi nilai *threshold* ini adalah sebagai detektor bit (0 atau 1) pada proses identifikasi bit pada file citra watermark

Setelah memasukkan data 0101, program akan melakukan penghitungan nilai *threshold*. Mula-mula program akan menghitung nilai d dari *y x b* , untuk tiap piksel yang telah disisipi dengan data 0101. Nilai d untuk sektor 1 sampai 5 dijumlahkan, kemudian dibagi dengan 5, didapat nilai rata-rata dari kelima sektor dan disimpan ke variabel bufferlong. Kemudian nilai d untuk tiap piksel dari seluruh sektor dijumlahkan dan dibagi dengan variabel bitsize, hasil akhir dari proses ini adalah nilai *threshold*, nilai ini disimpan ke variabel bitmean.

.3.2.2. Proses Identifikasi Bit dan Identifikasi Karakter Teks

Sama seperti proses penghitungan nilai bitmean, program akan menghitung nilai d dari *y x b* , dari tiap piksel terpakai. Program akan menghitung nilai d dari *y x b* , untuk tiap piksel pada lokasi *p*. Nilai d untuk sektor 1 sampai 5 dijumlahkan, kemudian dibagi dengan 5, didapat nilai rata-rata dari kelima sektor (d) dan disimpan ke variabel bufferlong. Nilai d dari tiap piksel akan dibandingkan dengan nilai bitmean, jika d lebih besar dari bitmean maka bit yang tersimpan pada piksel tersebut adalah bit 1, jika d lebih kecil dari bitmean maka bit yang tersimpan pada piksel tersebut adalah bit 0. Bit-bit tersebut kemudian disimpan ke *array* arraybit( ).

Setelah mengidentifikasi bit, program akan melakukan identifikasi karakter. Bit-bit yang tersimpan pada arraybit( ) dikonversi ke karakter ASCII dan kemudian ke bentuk karakter. Agar bit-bit tersebut dapat dikonversi ke bentuk karakter ASCII, bit-bit tersebut harus dikelompokkan menjadi per 7 bit kode biner. Kemudian tiap-tiap 7 bit kode biner ini disimpan ke *array* arraychrascii( ). Kemudian bit yang telah dikelompokkan diubah ke bentuk kode desimal.

Berikutnya adalah konversi kode desimal yang tersimpan di arraychrascii( ) menjadi karakter teks. Karakter ASCII terdiri dari 7 bit kode biner, dengan perintah Chr$( ), akan didapat karakter teks. Hasil dari konversi ke karakter teks disimpan ke variabel chtr. Variabel chtr berbentuk string. Variabel chtr kemudian ditampilkan ke kotak teks plainteks dengan perintah "rtbplaintext.Text = chtr".

Misalnya arraybit( ) = (1,1,1,0,1,0,0,1,1,0,0,1,0,1,1,1,1,0,0,1,1) , bit –bit ini akan dikelompokkan menjadi 1110100, 1100101 dan 1110011. Kemudian bit yang telah dikelompokkan diubah ke bentuk desimal menjadi 116, 101, 115. Bentuk desimal yang didapat disimpan ke arraychrascii( ), arraychrascii( ) = (116,101,115). Kode desimal kemudian diubah ke karakter teks, Chr$(arraychrascii (1)) = Chr$(116) = t, Chr$(arraychrascii (2)) = Chr$(101) = e, Chr$(arraychrascii (3)) = Chr$(115) = s. Variabel chtr berisi 3 karakter yaitu karakter "t", karakter "e" dan karakter "s". Karakter ini kemudian ditampilkan ke kotak teks plainteks menjadi "tes".

Terdapat kemungkinan bahwa data bit yang disisipkan ke citra digital jumlahnya lebih kecil dari tempat yamg tersedia di citra digital. Sebelumnya telah dijelaskan bahwa sisa tempat yang terisi akan diisi dengan bit 1. Maka pada proses identifikasi citra, jika program berhasil mengidentifikasi bit dengan tepat, program akan mendapatkan kode biner 1111111, yang berarti karakter ASCII 127. Pada kode program telah ditambahkan jika kode menemukan karakter ASCII 127 atau 0, maka program akan mengidentifikasi karakter ASCII tersebut sebagai spasi. Perintah untuk melakukan hal tersebut adalah, "If arraychrascii(i) = 0 Or arraychrascii(i) = 127 Then chtr01 = Chr$(32)", bilangan desimal 32 adalah karakter ASCII untuk spasi. Tindakan tersebut akan membuat sisa identifikasi karakter diisi dengan spasi, tidak diisi dengan karakter teks dari karakter ASCII 127 ataupun 0, sehingga kotak teks plainteks terlihat kosong dan hanya menampilkan isi dari variabel chtr, yang merupakan data teks (informasi) yang tersimpan pada citra.

Sebagai informasi tambahan program akan menampilkan informasi ukuran citra dan karakter ASCII dari 7 karakter teks pertama pada kolom "INFORMASI"

1. Pengujian

Pengujian meliputi proses *watermarking* dan proses *retrieve* data teks plainteks pada citra digital dengan format Bitmap. Pengujian dilakukan 3 kali, yaitu Pengujian 1, Pengujian 2 dan Pengujian 3. Untuk Data teks (informasi) pada Pengujian 1 terdiri dari 156 karakter teks plainteks dan dilakukan pengujian untuk faktor kualitas, $q = 2$, $q = 4$ dan $q = 6$.

Data teks pada Pengujian 2 terdiri dari 86 karakter teks plainteks dan dilakukan pengujian untuk faktor kualitas, $q = 4$. Sedangkan data teks pada Pengujian 3 terdiri dari 43 karakter teks plainteks dan dilakukan pengujian untuk faktor kualitas, $q = 4$. Masing-masing pengujian dilakukan terhadap 4 buah citra digital yang berbeda. Jadi total ada 20 kali pengujian yaitu, 12 pengujian untuk Pengujian 1, 4 pengujian untuk Pengujian 2 dan 4 pengujian untuk Pengujian 3.

Pengujian dilakukan dengan melakukan me-*watermark* data teks plainteks ke citra digital, kemudian setelah berhasil di-*watermark* file citra asli dan file citra yang telah di-*watermark* (2 file) disimpan. Kemudian file citra *watermark* yang telah disimpan data teks plainteks diambil kembali melalui proses *retrieve*. Untuk proses *retrieve*, program akan meminta pengguna memasukkan file citra sebanyak 2 kali, yaitu file citra asli dan file citra *watermark*. Lalu pada semua pengujian ketika diminta membuka file citra asli, file yang dibuka adalah file citra *watermark*, untuk menunjukkan bahwa program tidak memerlukan file citra asli dalam proses *retrieve*.

Tabel 1. Analisa tingkat Kesalahan (*Error*) program untuk pengujian 1

Faktor Kualitas (q) = 2								
No	**Nama File Citra**	**Karakter (Jumlah = 156)**						
		R	**B**	**%B**	**H**	**%H**	**TE**	**%TE**
1	01ALL-p1q2.bmp	146	11	7,53%	10	6%	21	13%
2	02ALL-p1q2.bmp	90	8	8,89%	66	42%	74	47%
3	03ALL-p1q2.bmp	94	12	12,77%	62	40%	74	47%
4	04ALL-p1q2.bmp	107	10	9,35%	49	31%	59	38%
Faktor Kualitas (q) =4								
No	**Nama File Citra**	**Karakter (Jumlah = 156)**						
		R	**B**	**%B**	**H**	**%H**	**TE**	**%TE**
1	01ALL-p1q4.bmp	146	3	2,05%	10	6%	13	8%
2	02ALL-p1q4.bmp	90	0	0%	66	42%	66	42%
3	03ALL-p1q4.bmp	94	1	1,06%	62	40%	63	40%
4	04ALL-p1q4.bmp	107	0	0,68%	49	31%	49	31%
Faktor Kualitas (q) =6								
No	**Nama File Citra**	**Karakter (Jumlah = 156)**						
		R	**B**	**%B**	**H**	**%H**	**TE**	**%TE**
1	01ALL-p1q6.bmp	146	1	0,68%	8	5%	9	6%
2	02ALL-p1q6.bmp	90	0	0%	66	42%	66	42%
3	03ALL-p1q6.bmp	94	0	0%	62	40%	62	40%
4	04ALL-p1q6.bmp	107	0	0%	49	31%	49	31%

Tabel 2. Analisa tingkat Kesalahan (*Error*) program untuk pengujian 2

Faktor Kualitas (q) = 4								
No	**Nama File Citra**	**Karakter (Jumlah = 86)**						
		R	**B**	**%B**	**H**	**%H**	**TE**	**%TE**
1	01ALL-p2.bmp	84	1	1,19%	2*	1%	3	2%
2	02ALL-p2.bmp	86	0	0%	0	0%	0	0%
3	03ALL-p2.bmp	86	0	0%	0	0%	0	0%
4	04ALL-p2.bmp	86	0	0%	0	0%	0	0%

*Terjadi Penambahan karakter

Tabel 3. Analisa tingkat Kesalahan (*Error*) program untuk pengujian 3

Faktor Kualitas (q) = 4								
No	**Nama File Citra**	**Karakter (Jumlah = 43)**						
		R	**B**	**%B**	**H**	**%H**	**TE**	**%TE**
1	01ALL-p3.bmp	39	0	0%	4*	3%	4	3%
2	02ALL-p3.bmp	43	0	0%	0	0%	0	0%
3	03ALL-p3.bmp	41	1	2,44%	2*	1%	3	2%
4	04ALL-p3.bmp	43	0	0%	0	0%	0	0%

*Terjadi Penambahan karakter

Berdasarkan hasil analisa tersebut, didapatkan bahwa semakin kecil nilai *q,* maka kemampuan program untuk mengidentifikasi bit semakin berkurang, menyebabkan kesalahan identifikasi karakter. Semakin besar nilai *q,* kemampuan program untuk mengenali karakter semakin baik.

5. Kesimpulan

Berdasarkan pengujian setelah program watermarking citra digital menggunakan teknik Amplitude Modulation telah berhasil direalisasikan, ternyata didapatkan semakin besar nilai q, kemampuan program dalam mengidentifikasi karakter yang di-watermark ke citra semakin baik

Untuk pengembangan lebih lanjut dapat diteliti untuk dapat melakukan watermarking pada berbagai citra digital dengan menggunakan berbagai jenis input yang berbeda.

Daftar Pustaka

[Pop98] Popa, Richard. (1998). *An Analysis of Steganographic Techniques*. The "Politehnica" University of Timisoara, Faculty of Automatics and Computers, Department of Computer Science and Software Engineering, Timisoara.

[Sch04] Schneiner B. (1994). *Applied Cryptography: Protocols, Algorithm, and Source Code in C*. Wiley, New York.

[Sup00] Supangkat, Suhono H., Kuspriyanto, Juanda. (2000). Watermarking sebagai Teknik Penyembunyian Label Hak Cipta pada Data Digital. Departemen Teknik Elektro, Institut Teknologi, Bandung.