Audio Watermarking Menggunakan *Discrete Wavelet Transform*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

oleh:
AHMAD AZWAR ANAS
0710960035-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2012


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

AUDIO WATERMARKING MENGGUNAKAN
DISCRETE WAVELET TRANSFORM

Oleh :
Ahmad Azwar Anas
0710960035-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 09 Agustus 2012
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana
Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs.Muh Arif Rahman, M.kom	Drs. Marji, M.T
NIP. 196604231991111001	NIP. 196708011992031001

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Ahmad Azwar Anas
NIM	: 0710960035-96
Jurusan	: Matematika
Program Studi	: Ilmu Komputer
Penulis Skripsi berjudul	: *Audio Watermarking* Menggunakan *Discrete Wavelet Transform*

Dengan ini menyatakan bahwa :

1. Skripsi ini karya saya sendiri dan tidak menjiplak karya orang lain selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila dikemudian hari ternyata skripsi yang ditulis terbukti hasil jiplakan maka saya bersedia menanggung segala resiko yang akan diterima.

Demikian pernyataan dibuat dengan segala kesadaran.

Malang, 09 Agustus 2012
Yang menyatakan,

Ahmad Azwar Anas
NIM. 0710960035-96

Audio Watermarking Menggunakan *Discrete Wavelet Transform*

ABSTRAK

Audio watermarking merupakan salah satu cara untuk melindungi data *audio* digital dari pelanggaran hak cipta. Informasi yang ditempelkan ke dalam media asli dalam *audio watermarking* digunakan untuk mempertahankan identitas unik pemiliknya.

Banyak algoritma *watermarking* telah diusulkan dan diimplementasikan untuk *audio watermarking* yang menunjukkan bahwa sistem pendengaran manusia lebih sensitif dan kompleks dibandingkan dengan sistem penglihatan manusia.

Teknik *audio watermarking* yang digunakan pada penelitian adalah *discrete wavelet transform* yang bekerja dalam domain frekuensi. Metode *discrete wavelet transform* melakukan proses transformasi untuk melakukan proses dekomposisi sinyal *audio* asli menjadi beberapa *multi-resolution sub-band* yang memungkinkan pembuat algoritma untuk menentukan beberapa *sub-band* dengan tingkat kesesuaian yang tinggi untuk penyisipan *bit watermark*. *Bit watermark* disisipkan pada beberapa *sub-band* dengan resolusi yang tinggi pada sinyal *audio* sehingga kepuasan pencapaian performa *robustness* dan *imperceptibility* (*inaudibility*) dapat diperoleh.

Pengujian yang dilakukan menggunakan berbagai serangan yang menunjukkan bahwa *audio* yang disisipi *watermark* juga mampu terdeteksi dengan baik dan tahan terhadap beberapa serangan dari *Adobe Audition 3*.

Pengujian terakhir yang dilakukan adalah dengan menunjukkan hasil *audio* yang disisipi *watermark* terhadap 10 responden dan diperoleh hasil semua responden tidak mendengar adanya perbedaan kualitas *audio* yang disisipi *watermark* dengan *audio* asli setelah dilakukan penyerangan.

Kata kunci: Multimedia, *Watermarking*, *Discrete Wavelet Transform*, Transformasi Domain

Audio Watermarking Using Discrete Wavelet Transform

ABSTRACT

Audio watermarking is one way to protect digital audio data from copyright infringement. Information embedded into the original media in audio watermarking is used to maintain the unique identity of its owner.

Many watermarking algorithms have been proposed and implemented for audio watermarking which suggests that the human auditory system is more sensitive and complex than the human visual system.

Audio watermarking technique used in this study was discrete wavelet transform that works in the frequency domain. Discrete wavelet transform method to the process of transformation to make the process of decomposition of the original audio signal into a multi-resolution sub-band algorithm that allows the manufacturer to determine the number of sub-band with a high level of suitability for embedding watermark bits. Watermark bit is inserted in a sub-band high-resolution audio signals so that the satisfaction in the achievement of performance robustness and imperceptibility (inaudibility) can be obtained.

Tests were performed using a variety of attacks that show that the inserted audio watermark can also be detected well and is resistant to some attacks from the Adobe Audition 3.

The last test performed is to show the results of the inserted audio watermark of 10 respondents and the results of all respondents did not hear any difference in audio quality with the original audio watermark inserted after the attack.

Keywords : Multimedia, Watermarking, Discrete Wavelet Transform, Domain Transformation

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur kehadirat Allah SWT atas segala rahmat dan hidayah-Nya yang memberikan kekuatan pada penulis sehingga penulis dapat mengerjakan skripsi yang berjudul "***Audio Watermarking* Menggunakan *Discrete Wavelet Transform*"**. Skripsi disusun dan diajukan sebagai syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Shalawat dan salam atas Nabi Muhammad SAW yang senantiasa memberikan cahaya Islam atas keluarganya dan sahabat-sahabatnya.

Penulis mendapat banyak bantuan baik secara moral dan materi dari berbagai pihak dalam penyelesaian skripsi. Penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang tulus atas bantuan yang diberikan kepada:

1. Dr. Abdul Rouf Alghofari, M.Sc. selaku Ketua Jurusan Matematika Universitas Brawijaya.
2. Drs. Muh. Arif Rahman, M.Kom. selaku Dosen Pembimbing skripsi yang memberikan saran, kritik, waktu, bimbingan, dan dorongan semangat selama penulisan skripsi.
3. Drs.Marji M.T. selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang dan selaku Dosen Pembimbing skripsi yang memberikan saran, kritik, waktu, bimbingan, dan dorongan semangat selama penulisan skripsi.
4. Kasyful Amron, S.T. selaku Penasihat Akademik yang memberikan bimbingan akademik dan dorongan semangatnya.
5. Segenap bapak dan ibu dosen yang mendidik dan mengamalkan ilmunya kepada penulis.
6. Segenap staf dan karyawan di Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
7. Ayah, ibu, kakak, dan adik atas doa, dukungan, dan semangat yang diberikan.
8. Jimmy dan sahabat-sahabatku yang memberikan semangat, inspirasi, dan dukungan.
9. Pihak lain yang membantu terselesaikannya skripsi.

Semoga penulisan laporan skripsi bermanfaat bagi para pembaca. Penulis sadar bahwa skripsi masih jauh dari kesempurnaan dan mengandung banyak kekurangan sehingga penulis dengan kerendahan hati mengharapkan kritik dan saran dari pembaca.

Malang, 09 Agustus 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

DAFTAR *SOURCE CODE*


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ketersediaan koneksi internet dengan *bandwidth* tinggi yang terjangkau serta peningkatan kualitas proses pelayanan konsumen menghasilkan kemampuan untuk memproses, mentransmisi, dan menyimpan data dalam berbagai format media, khususnya *audio* dan *video* yang tersedia untuk berbagai tujuan (Al-Haj, 2011). Hal tersebut ditunjang dengan adanya komputer dengan kemampuan tinggi untuk mengeksekusi algoritma kompresi *lossy* dengan mereduksi *bandwidth* dan kapasitas penyimpanan yang dibutuhkan untuk menyimpan informasi rahasia.

Hal tersebut belum terkolerasi secara baik dengan mekanisme distribusi hak cipta modern yang bertindak sebagai tindakan preventif pembajakan. Distribusi tersebut memberikan keuntungan berupa insentif tambahan bagi distributor dari pembuat media digital. Hak cipta bagi pembuat media digital berdasarkan aturan konvensi Berne terdiri atas eksklusifitas *intangible rights* sebagai berikut:

a. *Reproduction right*: ijin untuk melakukan duplikasi hasil media digital yang diproteksi.
b. *Derivative right*: ijin untuk membuat media baru secara derivatif ataupun adaptasi dari media digital yang diproteksi.
c. *Distribution right*: ijin untuk menjual ataupun mendistribusikan kepada publik.
d. *Performance and display rights*: ijin untuk mempertunjukkan media digital yang diproteksi seperti komposisi musikal atau pelaksanaan pameran seperti pameran fotografi publik. Ijin tersebut dapat bervariasi berdasarkan hukum dan jenis pekerjaan.

Problematika utama adalah keberadaan sistem komputer terkoneksi yang memudahkan pertukaran data antar individu bahkan dalam beberapa kondisi terkesan otomatis. Hal tersebut menyebabkan distribusi hasil pembajakan menjadi sederhana dan proses otomatis untuk melacak hingga level internasional menjadi sulit (Al-Haj, 2011).

Watermarking merupakan teknik yang memungkinkan penyediaan representasi media digital untuk proteksi dari proses penghilangan penanda. Hal tersebut mengimplikasikan bahwa penambahan properti dengan ciri sinyal yang ditanamkan harus *redundant* sehingga dapat bertahan terhadap degradasi secara selektif dan penghilangan penanda. Penambahan tersebut harus dilakukan dengan syarat bahwa penanda tidak dapat diganti dengan *fraudulent message* ataupun dihapus secara keseluruhan.

Salah satu media digital yang sering dilakukan proses *watermarking* adalah *audio* digital. *Audio* digital merupakan variasi tekanan suara di udara terhadap kumpulan nilai representasi interval waktu antar sinyal dengan proses perubahan sinyal analog menjadi diskrit dan diubah dalam bentuk digital melalui proses kuantisasi (Rusdianto, 2009).

Audio watermarking menggunakan *psychoacoustic auditory model* dengan metode *spread spectrum* (Rusdianto, 2009) menjelaskan bahwa *Human Auditory System* (HAS) memiliki *masking threshold* sehingga memiliki kemungkinan untuk diberikan *watermark*. Penelitian tersebut memiliki kekurangan karena proses pemberian *watermark* dilakukan dengan metode *spread spectrum* yang dikenal dengan kelemahan dalam penyisipan *watermark* dalam jumlah besar. Hal tersebut dibuktikan dengan jenis informasi yang disisipkan hanya berupa *string* dengan panjang tidak lebih dari 30 karakter.

Alternatif *audio watermarking* yang efektif dapat diperoleh dengan menggunakan algoritma berdasarkan metode *discrete wavelet transform*. Metode *discrete wavelet transform* melakukan transformasi untuk proses dekomposisi sinyal *audio* asli menjadi beberapa *multi-resolution sub-band* yang memungkinkan pembuat algoritma untuk menentukan beberapa *sub-band* dengan tingkat kesesuaian yang tinggi untuk penyisipan *bit watermark*. *Bit watermark* disisipkan pada beberapa *sub-band* dengan resolusi yang tinggi pada sinyal *audio* sehingga kepuasan pencapaian performa, *robustness*, dan *imperceptibility* (*inaudibility*) dapat diperoleh (Al-Haj, 2011). Penulis melakukan penelitian dan pengujian terhadap metode *discrete wavelet transform* pada teknik *watermarking* untuk *audio* digital berdasarkan penjelasan latar belakang tersebut. Penulis

mengambil judul "***Audio Watermarking* Menggunakan *Discrete Wavelet Transform***".

1.2 Rumusan Masalah

Rumusan permasalahan yang menjadi titik pembahasan dalam skripsi antara lain:

1. Implementasi penyisipan data dengan metode *Discrete Wavelet Transform* (DWT).
2. Kualitas *audio* digital yang dihasilkan setelah disisipi *watermark* jika dibandingkan dengan *file* asli.
3. Ketahanan *file audio* digital berdasarkan *Mean Opinion Square* (MOS) dan *Peak Signal to Noise Ratio* (PSNR) terhadap beberapa serangan dari Adobe Audition 3.

1.3 Batasan Masalah

Batasan permasalahan yang dipilih pada skripsi untuk menghindari salah persepsi antara lain:

1. Metode yang digunakan untuk penyisipan *watermark* adalah *discrete wavelet transform.*
2. Berkas *audio* yang digunakan adalah *audio* dengan format WAV berupa musik instrumental dan klasik dengan panjang sekitar 26 detik yang memiliki *sample rate* 44,1 kHz dan dikuantisasi 16 bit per *sample* dalam kualitas CD.
3. Berkas gambar yang disisipkan adalah gambar *gray scale* berformat BMP dengan ukuran 20x20 *pixel* dan 50x50 *pixel*.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian antara lain:

1. Implementasi *audio watermarking* menggunakan *discrete wavelet transform*.
2. Menghitung serta menganalisis kualitas *audio* digital dari proses *watermarking* menggunakan *signal to noise ratio* dan *mean opinion score*.

1.5 Manfaat

Penulisan skripsi diharapkan dapat digunakan sebagai alternatif bagi pembuat *audio* digital untuk melakukan proteksi terhadap kepemilikan *audio* digital.

1.6 Metodologi Pemecahan Masalah

Metodologi yang digunakan dalam penulisan skripsi untuk mencapai tujuan dirumuskan yaitu:

1. Studi literatur
   Mempelajari teori-teori, konsep-konsep dasar, dan identifikasi masalah yang berhubungan dengan *watermarking*, khususnya mengenai metode *discrete wavelet transform* melalui pustaka yang bersangkutan yaitu buku, jurnal ilmiah, ataupun melalui internet.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat perancangan sistem *audio watermarking* dan mengimplementasikan hasil rancangan tersebut dengan metode *discrete wavelet transform*.
4. Uji coba dan analisis hasil implementasi
   Menguji sistem dan menganalisis hasil dari implementasi tersebut untuk mengetahui kesesuaian dengan tujuan yang dirumuskan sebelumnya untuk dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Pembuatan skripsi dilakukan dengan pembagian bab sebagai berikut :

BAB 1: PENDAHULUAN

Bab 1 membahas mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi pemecahan masalah, dan sistematika penulisan skripsi.

BAB II: TINJAUAN PUSTAKA

Bab 2 menjelaskan tentang teori dasar *watermarking*, teknik *watermarking* menggunakan metode *discrete wavelet transform*, data digital berupa *audio* dan gambar secara umum

berdasarkan literatur yang meliputi buku referensi serta dokumentasi internet.

BAB III: METODOLOGI DAN PERANCANGAN

Bab 3 menjelaskan metode-metode yang digunakan untuk menyelesaikan masalah teknik *watermarking* menggunakan metode *discrete wavelete transform* dengan menjelaskan tahapan deskripsi umum sistem, perancangan kerja sistem, perancangan uji coba, dan evaluasi hasil.

BAB IV:HASIL DAN PEMBAHASAN

Bab 4 menjelaskan proses implementasi dari rancangan penelitian yang dijelaskan pada bab 3. Implementasi terdiri dari implementasi program, penerapan aplikasi, uji coba, dan analisa hasil percobaan.

BAB V: KESIMPULAN DAN SARAN

Bab 5 berisi kesimpulan dari pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan skripsi lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Konsep Dasar *Digital Watermarking*

Teknik proteksi kekayaan intelektual dibutuhkan selama kekayaan intelektual tersebut didistribusikan untuk dijual tanpa adanya dukungan secara organisasi. Beberapa mekanisme proteksi kekayaan intelektual secara teknis diklasifikasikan dalam taksonomi sebagai berikut:

a. *Copy protection*
   Sebagian besar bentuk langsung dari *control exertion.* Sebuah entitas dijual atau diberikan lisensi dengan sejumlah duplikasi tertentu yang tetap. Mekanisme tersebut memastikan tidak adanya replikasi tambahan. Beberapa *subtaxa* diidentifikasikan sebagai berikut:
   - *Analog physical media*
     Media harus mengijinkan reproduksi hasil pekerjaan pada beberapa alat yang ditentukan dan melakukan modifikasi signal sehingga tidak bisa direproduksi.
   - *Analog ephemeral data*
     Signal-signal *boardcast* harus ditransmisikan hanya pada beberapa alat yang ditentukan untuk melakukan reproduksi hasil pekerjaan.
   - *Digital physical media*
     Duplikasi media harus menggunakan operasi tambahan disamping proses reproduksi sederhana pada hasil pekerjaan saat menambahkan informasi tambahan atau fitur untuk reproduksi.
   - *Digital ephemeral data*
     Data yang dibutuhkan untuk reproduksi hasil pekerjaan harus diproses menggunakan alat yang tidak mengijinkan replikasi pada hasil pekerjaan yang diproteksi.

b. *Usage monitoring*
   Setiap *instance* parsial dari suatu operasi ataupun sekumpulan operasi yang didefinisikan sebagai penggunaan harus dilakukan proses *record* atau dikomunikasikan sehingga informasi secara *subsequent* dapat digunakan oleh pemilik media ataupun agen dari pemilik media tersebut.

c. *Distribution tracing*
   Pembuatan duplikat dan transmisi *subsequent* untuk alat yang lain, individu, kelanjutan dari pelaksanaan pekerjaan awal harus menghasilkan pembuatan proses *record* informasi sebuah fitur yang mengidentifikasi *source*, dan kemungkinan identifikasi tujuan dari transmisi.
d. *Usage control*
   Setiap *instance* parsial dari suatu operasi ataupun sekumpulan operasi yang didefinisikan sebagai penggunaan harus menjadi subjek untuk melakukan proses *approve* pemilik media ataupun agen dari pemilik media tersebut (Arnold, 2003).

Beberapa mekanisme proteksi fitur fisik media ataupun pembatas duplikasi yang dapat berkolaborasi dengan skema perijinan digital untuk melacak distribusi duplikasi hasil pembajakan secara individual adalah teknik *finger printing* dan *digital watermarking*. *Digital watermarking* (misalnya, penanaman sinyal *payload* ke dalam sinyal *carrier*) dan *finger printing* (misalnya, penurunan pola karakteristik identifikasi sinyal data media) merepresentasikan berbagai solusi untuk skenario aplikasi dengan independensi berbagai format media dan *encoding* serta dapat diimplementasikan secara obtrusif. *Watermark* digital juga dapat membantu proses pelacakan distribusi jika proses penandaan pada media mengidentifikasi jalur pada graf distribusi (Arnold, 2003).

Digital watermarking adalah proses penambahan kode identifikasi secara permanen ke dalam data digital. Kode identifikasi tersebut dapat berupa teks, gambar, *audio*, atau *video*. Selain tidak merusak data digital produk yang akan dilindungi, kode yang disisipkan seharusnya memiliki ketahanan (*robustness*) dari berbagai pemrosesan lanjutan seperti pengubahan, transformasi, geometri, kompresi, enkripsi, dan sebagainya. Sifat *robustness* berarti data *watermark* tidak rusak akibat pemrosesan lebih lanjut.

Watermarking sedikit berbeda dengan *watermark* pada uang kertas. *Watermark* pada uang kertas masih dapat terlihat secara kasat mata manusia (mungkin dalam posisi kertas yang tertentu) tetapi *watermarking* pada media digital dimaksudkan untuk tidak dirasakan kehadirannya oleh manusia tanpa alat bantu mesin pengolah digital seperti komputer dan sejenisnya. *Watermarking* memanfaatkan kekurangan-kekurangan sistem indera manusia seperti mata dan

telinga. Metode *watermarking* dapat diterapkan pada berbagai media digital dengan adanya kekurangan tersebut. Oleh karena itu, *watermarking* merupakan suatu cara untuk penyembunyian atau penanaman data atau informasi tertentu (baik hanya berupa catatan umum maupun rahasia) ke dalam suatu data digital lainnya tetapi tidak diketahui kehadirannya oleh indera manusia (indera penglihatan atau indera pendengaran) dan mampu menghadapi proses pengolahan sinyal digital sampai pada tahap tertentu (Rusdianto, 2009).

2.1.1 Sejarah *Watermarking*

Watermarking sudah ada sejak 700 tahun yang lalu. Pabrik kertas di Fabriano, Italia, membuat kertas yang diberi *watermark* atau tanda-air dengan cara menekan bentuk cetakan gambar atau tulisan pada kertas yang baru setengah jadi pada akhir abad 13. Kertas dikeringkan sehingga suatu kertas memiliki *watermark*. Kertas biasanya digunakan oleh seniman atau sastrawan untuk menulis karya seniman tersebut. Kertas yang sudah dibubuhi *watermark* tersebut dijadikan identifikasi bahwa karya seni adalah milik seniman tersebut.

Ide *watermarking* pada data digital sehingga disebut *digital watermarking* dikembangkan di Jepang tahun 1990 dan Swiss tahun 1993. *Digital watermarking* semakin berkembang seiring dengan semakin meluasnya penggunaan internet, objek digital seperti *video*, gambar, dan *audio* yang dapat dengan mudah digandakan dan disebarluaskan (Friskayanti, 2011).

2.1.2 Klasifikasi dan Kebutuhan *Watermarking*

Digital watermarking dapat diklasifikasikan dan diukur atas dasar beberapa karakteristik termasuk pertahanan dari berbagai macam serangan, kapasitas penyimpanan dari perlawanan dari serangan jahat, kecocokan dengan *watermark* lain, dan kompleksitas dari metode *watermarking*. Digital *watermarking* harus memenuhi beberapa persyaratan yang disebutkan pada beberapa jurnal ilmiah dan penelitian antara lain ketahanan *(robustness)*, tidak terlihat *(nonperceptibility)*, tidak terdeteksi *(nondetecable)*, kapasitas, dan kompleksitas sebagai berikut (Friskayanti,2011):

1. Ketahanan *(Robustness)*

Robustness dapat diartikan pembatasan hak akses terhadap perubahan informasi *watermarking* dan memodifikasi *file* asli. Modifikasi yang dimaksud adalah mengubah ukuran, kompresi *file*, dan sebagainya.

2. Tidak terlihat *(Nonperceptibility)*
   Klasifikasi didasarkan pada *Human Visual System* (HVS) dan *Human Audio System* (HAS). *Watermark* tidak terlihat jika manusia normal tidak dapat membedakan antara *file* asli dan pembawa *watermark*.
3. Tidak terdeteksi *(Nondetecable)*
   Materi data dengan membawa informasi *watermark* tidak terdeteksi jika materi data konsisten dengan sumber data. *Nondetecability* berhubungan dengan sumber data dan komponen-komponennya.
4. Kapasitas
   Kapasitas mengacu pada kemampuan sumber data dalam menyimpan informasi *watermark*.
5. Kompleksitas
   Kompleksitas menjelaskan cara yang diperlukan untuk mendeteksi dan menyandikan informasi *watermark*.

2.1.3 Tipe *Digital Watermarking*

Watermark dapat diklasifikasikan dalam beberapa kategori yaitu:

1. Tipe sumber atau aplikasi yang digunakan
   Tipe sumber atau aplikasi yang dimaksud adalah objek yang digunakan dalam proses *watermark*. Objek tersebut yang akan disisipi oleh suatu tanda identitas unik pemiliknya. Objek yang dapat disisipi *watermark* antara lain gambar, *video*, *audio*, *text*, dan sebagainya.
2. Persepsi manusia
   Watermark dibedakan menjadi dua berdasarkan persepsi manusia yaitu:
   a. *Visible watermark* merupakan suatu tanda yang disisipkan sehingga dapat diketahui oleh indera manusia.
   b. *Invisible watermark* merupakan suatu tanda yang disisipkan sehingga tidak dapat diketahui oleh indera manusia.

3. *Blindness*

   Watermark dibagi menjadi dua berdasarkan *blindness* yaitu:

   a. *Blind watermark* merupakan teknik mendapatkan hasil ekstraksi *watermark* yang dapat dilakukan tanpa data asli.
   b. *Nonblind watermark* merupakan teknik mendapatkan data asli dan data *watermark* diperlukan dalam ekstraksi *watermark*. Teknik tersebut membuat data sumber diperiksa untuk memperoleh informasi *watermark* (Friskayanti, 2011).

2.1.4 Teknik-Teknik dalam *Watermarking*

Beberapa penelitian mengenai teknik-teknik yang digunakan dalam *watermarking* banyak dilakukan. Teknik-teknik tersebut dapat diklasifikasikan berdasarkan domain kerja menjadi tiga kelompok yaitu:

1. Teknik *watermarking* yang bekerja pada domain spasial (*spatial domain watermarking*).
2. Teknik *watermarking* yang bekerja pada domain *transform/* frekuensi (*transform domain watermarking*).
3. Teknik *watermarking* yang bekerja pada kedua domain (*hybrid techniques watermarking*) (Friskayanti, 2011).

Teknik *audio watermarking* berdasarkan literatur dapat dikelompokkan menjadi dua bagian yaitu teknik berdasarkan *time-domain* dan *frequency-transform* (Al-Haj, 2011). Dua domain tersebut memiliki perbedaan karakteristik dan performa terhadap kebutuhan *robustness* dan *imperceptibility* (*inaudibility*) dalam *audio watermarking*. *Inaudibility* merupakan kondisi *watermark* yang disisipkan seharusnya tidak memunculkan distorsi yang dapat didengar berdasarkan *audio* digital asli dengan syarat bahwa *audio* digital versi *watermarking* dan *audio* digital asli tidak mudah untuk dapat dibedakan. *Robustness* merupakan ketahanan *watermark* terhadap penghapusan ataupun degradasi. *Watermark* dapat bertahan dari berbagai macam serangan seperti *random cropping* dan *noise adding*. Proses penghapusan tidak mungkin dilakukan kecuali dengan penggabungan sinyal yang sesuai.

Teknik *time-domain* (Al-Haj, 2011) terdiri dari substitusi *Least Significant Bit* (LSB), teknik *echo hiding*, dan yang lainnya. LSB menanamkan informasi *watermark* pada bit *audio* digital

dengan signifikansi terkecil dengan melakukan proses penindihan *bit* asli. Hal tersebut merupakan kelebihan dari kesalahan kuantisasi yang sering diturunkan dari proses digitalisasi sinyal *audio*. *Echo watermarking* di sisi lain berusaha untuk menanamkan informasi ke dalam sinyal diskrit dari *audio* asli melalui pengulangan versi dari komponen sinyal *audio* dengan *offset* yang kecil, amplitudo yang terinisinialisasi, dan *decay rate* membuat hal tersebut menjadi tidak terdengar. *Audio watermarking* berbasis *time-domain* umumnya relatif mudah untuk diimplementasikan dan membutuhkan sedikit sumber daya komputasi namun teknik tersebut lemah terhadap serangan pemrosesan sinyal seperti kompresi dan *filtering*.

Teknik *audio watermarking* berbasis *frequency-domain* menggunakan properti *human perceptual* dan karakteristik *frequency masking* dari sistem pendengaran manusia untuk *watermarking* yang efektif (Al-Haj, 2011). Fase dan amplitudo dari koefisien *transform domain* dimodifikasi dengan cara membawa informasi *watermark* yang diinginkan. Beberapa transformasi yang populer adalah *Discrete Fourier Transform* (DFT), *Discrete Cosine Transform* (DCT), dan *Discrete Wavelets Transform* (DWT). Koefisien magnitudo dari *fourier transform* antara 2.4 KHz sampai dengan 6.4 KHz diganti dengan *watermark sequence* saat pengabaian sensitivitas manusia terhadap *peak* sekitar 1 KHz. Pendengaran manusia relatif tidak sensitif terhadap distorsi fase khususnya ketidakmampuan untuk menerima nilai fase absolut bahkan *watermark* direpresentasikan sebagai fase relatif pada beberapa koefisien. Permasalahan yang timbul dengan skema *watermarking* tersebut antara lain kurang bertahan terhadap pemrosesan sinyal dan beberapa serangan seperti kompresi *audio*.

Metode *spread-spectrum watermarking* juga sangat populer. Metode tersebut menanamkan sebuah *narrow-band signal* (*watermark*) ke dalam *wide-band channel* (*file audio*) dengan menyebarkan data *watermark* pada *large frequency band* yang sering disebut *audible spectrum* (Al-Haj, 2011). Pendeteksian *watermark* dapat dilakukan dengan melakukan kalkulasi korelasi antara sinyal *audio* yang ditanam *watermark* dengan sinyal *watermark*.

Metode yang terakhir yaitu metode *patchwork* (Al-Haj, 2011) yang menggunakan *pseudorandom* untuk melakukan proses penanaman kumpulan statisitik tertentu ke dalam kumpulan data dan

dideteksi melalui proses pembacaan dengan bantuan *index* secara numerik misalnya *mean* untuk menjelaskan distribusi secara spesifik. Kompleksitas komputasi metode tersebut sangat tinggi dan sinkronisasi sangat sulit untuk diterapkan.

2.1.5 Transform Domain *Watermarking*

Transform domain watermarking sering juga disebut dengan *frequency domain watermarking* yang merupakan penyisipan *watermark* pada koefisien frekuensi hasil transformasi *audio* asalnya. Beberapa transformasi yang umum digunakan oleh para peneliti antara lain *Discrete Cosine Transform* (DCT), *Discrete Fourier Transform* (DFT), *Discrete Wavelet Transform* (DWT), ataupun *Discrete Laguerre Transform* (DLT) (Rusdianto, 2009).

Teknik yang berbasiskan *wavelet* juga populer digunakan dalam *watermarking* digital seperti penggunaan *wavelet* pada *watermarking video* yang diusulkan oleh Swanson pada tahun 1997. Salah satu alasan pemanfaatan *wavelet* dalam *watermarking* adalah kemampuan *watermark* untuk bertahan dalam berbagai skala resolusi gambar (Rusdianto, 2009).

2.2 *Discrete Wavelet Transform* (DWT)

Wavelet merupakan suatu fungsi khusus dalam bentuk analog dengan sinus dan cosinus dalam analisis Fourier yang digunakan untuk fungsi dasar untuk mewakili sinyal. *Wavelet* menawarkan alat dengan resolusi yang banyak untuk analisis sinyal-sinyal dinamis dengan informasi lokalisasi waktu yang baik.

Koefisien dari transformasi *wavelet* diskrit dapat dikalkulasi secara rekursif maupun diterapkan pada algoritma piramida Mallat. Suatu koefisien dari transformasi *wavelet* diskrit berdimensi satu berdasarkan algoritma tersebut dapat dihitung dari koefisien pada beberapa tahap sebelum koefisien tersebut. DWT menggunakan persamaan iterasi yang dapat dilihat pada persamaan 2.1 dan 2.2 sebagai berikut (Al-Haj, 2011):

$$W_L(n,j) = \sum_m W_L(m, j-1) h_0(m-2n) \quad (2.1)$$

$$W_H(n,j) = \sum_m W_L(m, j-1) h_1(m-2n) \quad (2.2)$$

dimana $W_L(n, j)$ adalah koefisien *scaling* n di tahap j. $W_H(n, j)$ adalah koefisien *wavelet* n di tahap j dan m adalah induk n sedangkan $h_0(n)$ dan $h_1(n)$ adalah koefisien pelebaran sesuai dengan skala dan fungsi *wavelet* masing-masing. Persamaan 2.1 digunakan untuk memperoleh koefisien *wavelet* untuk tahap berikutnya. Proses dekomposisi dalam penerapan hanya dilakukan dalam sebagian kecil dari seluruh tahapan.

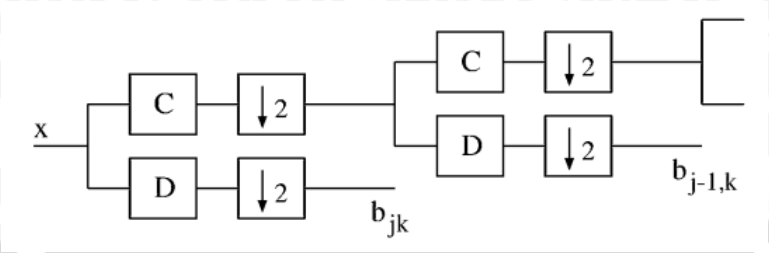
Koefisien transformasi *wavelet* diskrit dalam rangka untuk merekonstruksi data asli melewati proses *up-sample* untuk diteruskan melalui kumpulan *filter low-pass* dan *high-pass* seperti yang dinyatakan sebagai berikut (Al-Haj, 2011):

$$W_L(n,j) = \sum_k W_L(k, j+1) g_0(n-2k) + \sum_l W_H(l, j+1) g_1(n-2l) \quad (2.3)$$

dimana $g_0(n)$ dan $g_1(n)$ masing-masing adalah *filter* sintesis *low-pass* dan *high-pass* yang sesuai dengan *wavelet* induk ($W_L(n, j)$). $W_L(k, j+1)$ adalah koefisien *scaling* k di tahap j+1. $W_H(l, j)$ adalah koefisien *wavelet* l di tahap j+1, k setara l, dan k adalah *sub* n. Hal tersebut dapat diketahui dari persamaan 2.3 bahwa koefisien pada level ke j dapat diperoleh dari level ke j+1.

2.2.1 Transformasi *Wavelet* berdasarkan Multi Resolusi

Transformasi *wavelet* dioperasikan dalam waktu yang berlangsung kontinu dalam fungsi dan waktu tertentu dalam vektor. *Input* direpresentasikan sebagai f(t) atau x(n). *Ouput* direpresentasikan sebagai kumpulan dari koefisien b_{jk} yang mewakili *input* dalam bentuk *wavelet*. Koefisien b_{jk} dapat diturunkan dalam suatu bentuk formula. Koefisien tersebut diperoleh dengan proses integral dari f(t) terhadap $w_{jk}(t)$ jika dilihat dalam sudut pandang waktu yang berlangsung kontinu. Koefisen tersebut dapat diperoleh sebagai representasi sistem linier jika dilihat dalam sudut pandang waktu tertentu. *Discrete Wavelet Transform* (DWT) dapat diubah dalam bentuk *Fast Wavelet Transform* (FWT) dengan penggunaan koefisien tersebut. Metode *fast recursion* tersebut dikenal sebagai multi resolusi. Proses rekursi dalam *wavelet* dapat dijelaskan dengan struktur pohon dari *filter bank* pada gambar 2.1 (Strang, 1996).



Gambar 2. 1 Pohon *Filter Bank* (Strang, 1996)

High-pass filter D ditunjukkan sebagai hasil perhitungan selisih dari *input*. Langkah *down-sampling* digunakan untuk menetapkan selisih sebesar $\big(x(2k) - x(2k-1)\big)/\sqrt{2}$. Hasil tersebut ditetapkan sebagai *output* final karena *filter* tersebut tidak akan ditransformasi kembali. *Filter* tersebut menjadi akhir dari setiap cabang pohon *logarithmic tree*. Faktor $r = 1/\sqrt{2}$ diikutsertakan untuk menghasilkan vektor unit dalam matrik *sample* (Strang, 1996).

Low-pass filter C digunakan sebagai rata-rata dari *input*. *Filter* tidak dianggap sebagai *output* final karena dapat dilakukan proses *filter* kembali mengunakan *filter* D dan C. Rata-rata dan selisih dari semua level dapat dilihat dengan persamaan berikut (Strang, 1996) :

Rata-rata atau representasi koefisien *scaling* (*low-pass filter*):

$$a_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} + a_{j,2k+1}) \qquad (2.4)$$

Selisih atau representasi koefisien *wavelet* (*high-pass filter*):

$$b_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} - a_{j,2k+1}) \qquad (2.5)$$

dimana a dan b merupakan representasi koefisien *scaling* dan *wavelet* yang dapat diperoleh *wavelet* induk. Hal tersebut dapat diketahui dari persamaan 2.4 dan 2.5 bahwa koefisien pada level ke j-1 dapat diperoleh dari level ke j dengan k merupakan urutan *sample*.

Koefisien transformasi *wavelet* diskrit dalam rangka untuk merekonstruksi data asli melewati proses *up-sample* untuk diteruskan melalui kumpulan koefisien *scaling* dan *wavelet* seperti yang dinyatakan pada persamaan 2.6.

$$a_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,k} + b_{j,k}) \quad (2.6)$$

dimana a dan b merupakan representasi koefisien *scaling* dan *wavelet* yang dapat merekonstruksi *wavelet* induk pada urutan ke k. Hal dapat diketahui dari persamaan 2.6 bahwa koefisien pada level ke j-1 dapat diperoleh dari level ke j (Strang, 1996).

2.2.2 Penyisipan *Watermark* ke dalam *Audio*

Prosedur *embed* dilakukan dengan tiga operasi utama yaitu *watermark* pra-pengolahan, DWT berbasis dekomposisi frekuensi dari sinyal *audio*, dan penanaman *watermark* dalam sinyal *audio* yang diubah bentuk DWT.

Operasi dapat dijelaskan dalam langkah-langkah berikut:

a. Proses dilakukan dengan menyatakan *watermark* dalam bentuk gambar *gray-scale* sebagai matriks dua dimensi yang berukuran M1xM2.

$$Img = \{Img(k,j), 0 \leq k \leq M1, 0 \leq j \leq M2\} \quad (2.7)$$

b. Proses dilanjutkan dengan mengkonversi matriks gambar dua dimensi ke dalam ke suatu vektor satu dimensi dengan panjang M1xM2.

$$W = \{W_i = Img(k,j), i = k \times M2 + j, 0 \leq j \leq M2\} \quad (2.8)$$

Proses dilanjutkan dengan menormalkan vektor satu dimensi W dengan membagi setiap elemen dengan 255.

$$Wn_i = W_i/255\,; 1 \leq i \leq (M1 \times M2) \quad (2.9)$$

c. Proses dilanjutkan dengan menerapkan DWT level dua ke *left channel* dari sinyal *audio* stereo. Operasi DWT menghasilkan satu dimensi *sub-band* yang ditunjukkan pada gambar di bawah dengan A2 adalah pendekatan *sub-band* dengan D1 dan D2 adalah tingkat pertama dan kedua sebagai rincian dari tingkat *sub-band* masing-masing. Keputusan untuk

mengadopsi DWT level dua karena memberikan hasil yang lebih baik dari DWT dengan tingkat yang lebih tinggi.

A2	D2	D1

Gambar 2. 2 *Output Sub-Band* dari Operasi DWT Level Dua (Al-Haj, 2011)

d. Proses dilanjutkan dengan menyisipkan *watermark* vektor Wn yang dinormalisasi dinyatakan oleh persamaan 2.9 ke dalam tingkat kedua dari rincian *sub-band* D2 dari *left channel*. Prosedur *embed* menghasilkan tingkat kedua yang diberi *watermark* dengan rincian *sub-band* (WD2) seperti yang diberikan dalam persamaan 2.10 dan 2.11.

$$WD2 = \begin{cases} \left(D2_i + Wt_j\right);\ i = \{1, 120, 240, \dots, L2\}, dan \\ \qquad 1 \le j \le (M1 \times M2) \\ D2_i; i = \{(2:119), (121:239), \dots, (L2-119:L2)\} \\ \qquad 1 \le j \le (M1 \times M2) \end{cases} \quad (2.\ 10)$$

$$dimana\ Wt_i = Wn_i \times \alpha; 1 \le i \le (M1 \times M2) \quad (2.\ 11)$$

Wn_i adalah *bit watermark* yang dapat bernilai antara 0 hingga 1 dan α adalah faktor amplifikasi *watermark*. WD2 akan bernilai (D2+ 0.2) jika α ditentukan sebesar 0.2 ketika Wn_i bernilai 1 dan bernilai D2 ketika Wn_i bernilai 0.

2.2.3 Ekstraksi *Audio* yang Memiliki *Watermark*

Prosedur ekstraksi *watermark* memungkinkan pemilik dari klip *audio* untuk mengekstrak *watermark* yang disisipkan. Prosedur tersebut membutuhkan pengetahuan dari *file audio* asli, intensitas *watermark*, dan ukuran *watermark* untuk mengekstrak *watermark*. Ekstraksi *watermark* adalah langkah pembalikan langsung dari langkah-langkah yang dilakukan dalam prosedur *embed*. Langkah-langkah ekstraksi dijelaskan sebagai berikut:

a. Proses dilakukan dengan menerapkan operasi DWT level dua pada sinyal *audio* asli yang diberi *watermark*.

b. Proses dilanjutkan dengan menghitung *watermark* vektor W_t menurut rumus persamaan 2.12

$$Wt_i = WD2_i - D2_i;\ i = \{1,120,240,\dots,L2\} \quad dan\ 1 \le i \le (M1 \times M2) \qquad (2.\ 12)$$

c. Proses dilanjutkan dengan membagi W_t dengan faktor amplifikasi *watermark* (α) untuk memperoleh vektor *watermark* W_n yang dinormalisasi.

$$Wn_i = Wt_i/\alpha\ ;\ 1 \le i \le (M1 \times M2) \qquad (2.\ 13)$$

d. Proses dilanjutkan dengan mengalikan W_n dengan 255 untuk merekonstruksi *watermark* gambar *gray-scale* yang asli.

$$W_i = Wn_i \times 255;\ 1 \le i \le (M1 \times M2) \qquad (2.\ 14)$$

e. Proses dilanjutkan dengan mengkonversi W kembali ke matriks dua dimensi yang mewakili *watermark* gambar *gray-scale*.

Metode DWT berdasarkan properti lokalisasi frekuensi spasial yang sangat baik tersebut sangat cocok untuk mengidentifikasi daerah-daerah dimana sinyal *audio watermark* dapat tertanam secara efektif (Al-Haj, 2011).

2.3 Serangan terhadap *Digital Watermarking*

Skema global yang digambarkan sebagai suatu kumpulan pelayanan secara fungsionalitas terhadap suatu *assurance* secara serentak untuk diterapkan. Setiap tingkatan *assurance* diperlukan tingkat kemampuan yang berbeda untuk diterapkan. Oleh karena itu, suatu evaluasi yang sesuai harus dapat memastikan semua kebutuhan untuk suatu tingkatan *assurance* tertentu.

Jumlah tingkatan *assurance* tidak dapat ditentukan secara pasti. Kebanyakan dari tingkatan tersebut menunjukkan adanya ketidakteraturan evaluasi dan tidak dapat digunakan untuk tujuan parsial. Di sisi lain, hanya terdapat sedikit tingkatan yang dapat mencegah penyedia skema untuk menemukan suatu evaluasi yang cukup sesuai kepada kebutuhan. Pemilik *audio* digital juga dibatasi

oleh ketelitian metode yang tersedia untuk menilai. Evaluasi keamanan teknologi informasi masih digunakan dan keamanan tersebut disebutkan menjadi 6 atau 7 level berdasarkan alasan historis. Hal tersebut menunjukkan jumlah yang masuk akal untuk uji ketahanan.

Berdasarkan realitas yang ada, direkomendasikan untuk menggunakan lebih sedikit level untuk segmentasi pasar peralatan elektronik. Hal tersebut dikarenakan adanya kecenderungan seseorang untuk menambah jumlah *assurance* (Fabien, 2000).

2.3.1 Serangan berdasarkan Ketahanan (*Robustness*)

Serangan terhadap *audio* yang disisipi *watermark* dapat digunakan untuk mendapatkan tingkat ketahanan. Ketahanan dapat ditentukan dengan mengukur kemungkinan pendeteksian tanda dan *bit* laju galat untuk kumpulan kriteria relevan yang diinginkan. Tingkatan ketahanan dapat dimulai dari tingkat tanpa ketahanan hingga ketahanan yang dapat dibuktikan (Fabien, 2000).

Level 0 tidak memiliki fitur ketahanan dalam skema terpisah yang diperlukan untuk memenuhi tujuan dasar dan lingkungan operasional skema tersebut. Oleh karena itu, mengacu kembali pada contoh *radio-monitoring*, fitur minimal dari ketahanan perlu dipastikan bahwa *watermark* bertahan terhadap distorsi *radio-link* pada kondisi normal.

Level *low* sesuai dengan beberapa fitur tambahan yang ditambahkan tetapi ketahanan masih dapat dihilangkan dengan menggunakan sistem sederhana dan murah yang tersedia untuk umum. Fitur-fitur tersebut mencegah orang-orang 'jujur' untuk menonaktifkan *watermark* selama penggunaan normal. *Watermark* yang digunakan untuk mengidentifikasi pemilik lagu (*end user*) harus dapat diberikan proses penyimpanan, pengkompresan lagu, mengubah ukuran, dan memotongnya tanpa menghapus *watermark*.

Level *moderate* dapat dicapai ketika sistem yang lebih mahal diperlukan serta beberapa pengetahuan dasar dalam *watermarking*. Jika diasumsikan seperti masalah sebelumnya maka *end-user* akan memerlukan sistem seperti *Adobe Audition* dan menerapkan proses tambahan ke dalam *audio* untuk menonaktifkan *watermark*.

Level *moderately high* dapat dicapai saat sistem yang diperlukan tersedia tetapi keterampilan khusus dan pengetahuan

sangat diperlukan sehingga serangan memiliki kemungkinan tidak berhasil. Beberapa upaya dan operasi mungkin diperlukan dan penyerang harus meluang waktu untuk menyerang.

Level *high* dapat dicapai ketika semua serangan diketahui gagal sehingga diperlukan beberapa penelitian oleh tim spesialis. Biaya serangan itu mungkin jauh lebih tinggi dari biasanya dan keberhasilan belum tentu diperoleh.

Level *provable* dapat dicapai saat proses untuk menonaktifkan *watermark* harus dilakukan secara komputasi atau bahkan secara teoritis sehingga tidak mudah bagi lawan secara sengaja untuk menonaktifkan *watermark*. Hal tersebut memiliki kemiripan seperti kriptografi dengan beberapa algoritma yang didasarkan pada beberapa masalah matematika yang sulit (Fabien, 2000).

Level pertama dari level ketahanan dapat dinilai secara otomatis dengan menerapkan algoritma acuan sederhana yaitu:

a. Perlakuan untuk setiap media dalam satu set yang ditentukan:
   - Ditanamkan suatu muatan acak dengan kekuatan terbesar yang tidak memperkenalkan efek yang menjengkelkan. Dengan kata lain, proses penanaman *watermark* dengan kualitas *output* tersebut untuk metrik kualitas yang diberikan lebih besar dari minimum yang diberikan.
   - Diterapkan satu set transformasi yang diberikan ke media yang diberikan *watermark*.

b. Perlakuan untuk setiap media yang terdistorsi diterapkan proses ekstraksi *watermark* dan mengukur kepastian ekstraksi. Metode sederhana mungkin hanya menggunakan pendekatan keberhasilan/kegagalan, hal tersebut digunakan untuk mempertimbangkan kesuksesan ekstraksi jika dan hanya jika *payload* sepenuhnya pulih tanpa kesalahan. Ukuran untuk ketahanan adalah kepastian deteksi atau kesalahan *bit rate* setelah ekstraksi.

Prosedur tersebut harus diulang beberapa kali selama informasi yang tersembunyi masih acak dan proses pengujian mungkin bisa berhasil secara kebetulan.

Tingkat ketahanan berbeda dengan jumlah dan kekuatan serangan yang diterapkan dan sejumlah media yang diukur. Sekumpulan pengujian dan media akan tergantung pada tujuan dari skema *watermarking* yang didefinisikan dalam profil evaluasi.

Skema yang digunakan sebagai contoh dalam sistem medis hanya perlu diuji pada gambar medis sementara algoritma *watermarking* untuk identifikasi pemilik harus diuji pada sebuah panel besar dari media.

Tingkat pertama dari ketahanan dapat didefinisikan dengan menggunakan sebuah himpunan berhingga dan tepat dari kriteria ketahanan misalnya SDMI, IFPI atau persyaratan EBU, dan cukup satu yang diperlukan untuk memeriksa hal tersebut (Fabien, 2000).

2.3.2 *False Positives*

False positives sulit untuk mengukur serta menjadi solusi dengan menggunakan model untuk memperkirakan tingkatan. Hal tersebut memiliki dua masalah utama yaitu skema 'dunia nyata' *watermarking* sangat sulit untuk suatu model secara akurat dan pemodelan suatu skema membutuhkan akses secara rinci dari suatu algoritma. Terlepas dari kenyataan bahwa berbagai algoritma tidak mempublikasikan pelanggaran prinsip-prinsip Kerckhoffs, rincian algoritma masih dianggap sebagai rahasia perdagangan dan mendapatkan akses tidak selalu memungkinkan.

Salah satu cara naif memperkirakan *false alarm rate* dalam menghitung jumlah *false alarm* yaitu menggunakan *sample* data yang besar. Hal tersebut dapat berubah menjadi masalah lain yang sangat sulit karena beberapa aplikasi membutuhkan 1 kesalahan dalam 10^8 atau bahkan 10^{12} (Fabien, 2000).

Beberapa serangan dapat digunakan dalam pengujian kualitas ketahanan *watermark*. Beberapa serangan yang dilakukan adalah *amplify*, penambahan *noise*, *smooth*, dan *stat* (Al-Haj, 2011).

2.3.3 Serangan berdasarkan *Perceptibility*

Perceptibility dapat dinilai untuk level *assurance* yang berbeda. *Perceptibility* memiliki masalah yang sangat mirip dengan evaluasi algoritma kompresi. *Watermark* hanya bisa sedikit dimengerti tetapi tidak mengganggu atau tidak tampak di dalam studio atau sudut pandang konsumen maupun kondisi mendengarkan. Level lain adalah *nonperceptibility* dibandingkan dengan *file* asli dalam kondisi studio. Jaminan terbaik diperoleh ketika media *watermark* dinilai oleh sebuah panel individu yang

diminta untuk melihat atau mendengarkan dengan seksama di media berdasarkan kondisi yang dapat dilihat pada tabel 2.1.

Perceptibility tidak dapat diotomatisasi dan salah satu mungkin ingin menggunakan level kurang ketat. Berbagai level *assurance* juga dapat dicapai dengan menggunakan berbagai *quality measure* berdasarkan model persepsi manusia. Metrik tersebut tidak memperhitungkan distorsi geometrik secara tetap sehingga memiliki tantangan untuk memperhitungkan serangan terhadap skema *watermarking*.

Tingkatan pada tabel 2.1 mungkin tampak samar namun merupakan tingkatan terbaik yang dapat dicapai jika tidak memiliki metrik kualitas yang baik dan memuaskan (Fabien, 2000).

Tabel 2. 1 Ringkasan Level *Perceptibility Assurance* (Fabien, 2000)

Level *assurance*	Kriteria
Low	• *Peak Signal-to-Noise Rasio* ketika diberlakukan • Sedikit jelas namun tidak mengganggu
Moderate	• Metrik berdasarkan model persepsi • Tidak jelas di bawah kondisi domestik
Moderate high	• Tidak jelas dibandingkan dengan yang asli dalam kondisi studio
High	• Evaluasi tinggi oleh sebuah panel besar di bawah kondisi yang ketat.

2.4 *Peak Signal-to-Noise Ratio* (PSNR)

Istilah *Peak Signal-to-Noise Ratio* (PSNR) adalah sebuah istilah dalam bidang teknik yang menyatakan perbandingan antara kekuatan sinyal maksimum dari suatu sinyal digital dengan kekuatan *noise* yang mempengaruhi kebenaran sinyal tersebut. Oleh karena banyak sinyal memiliki rentang dinamis yang luas maka PSNR biasanya diekspresikan dalam skala *logarithmic decibel*. PSNR didefinisikan melalui *Signal-to-Noise Ratio* (SNR). SNR digunakan untuk mengukur tingkat kualitas sinyal. Nilai tersebut dihitung berdasarkan perbandingan antara sinyal dengan nilai kebisingan (*noise*). Kualitas sinyal berbanding lurus dengan nilai SNR. Semakin besar nilai PSNR semakin baik kualitas sinyal yang dihasilkan. SNR dari kumpulan nilai cuplikan adalah jumlah dari kuadrat nilai

kuantisasi dibagi dengan jumlah kuadrat nilai kuantisasi derau (Rusdianto, 2009). Pernyataan *Signal-to-Noise Ratio* (SNR) dapat dirumuskan melalui persamaan berikut :

$$SNR = \frac{\Sigma_{i-1}^{N} x^2(i)}{\Sigma_{i-1}^{N} \theta^2(i)} \quad (2.\ 15)$$

dengan N menyatakan banyaknya sinyal suara yang dicuplik, x(i) menyatakan nilai cuplikan ke-i, dan θ(i) menyatakan kesalahan cuplikan ke-i. SNR dapat pula dinyatakan sebagai perbandingan kuadrat sinyal rata-rata berkas WAV setelah disisipi *watermark* dengan kuadrat selisih kekuatan sinyal sebelum dan setelah disisipi *watermark*. Pernyataan di atas secara matematis ditulis menjadi persamaan 2.16.

$$SNR = \frac{\Sigma_{t=0}^{n} x^2{}_w(t)}{\Sigma_{t=0}^{n} (x(t) - x_w(t))^2} \quad (2.\ 16)$$

Nilai PSNR dapat dihitung berdasarkan persamaan dengan nilai PSNR direpresentasikan dalam skala desibel (dB) sebagai berikut:

$$PSNR = 10 \times \log_{10}(SNR) \quad (2.\ 17)$$

PSNR adalah metrik kualitas yang sangat ketat dan tidak memperhitungkan sifat dari model visual manusia. Hal tersebut termasuk sifat *masking* biasa tapi juga toleransi besar untuk distorsi geometris. Proses pengecualian skema *watermarking* dengan menggunakan PSNR dilakukan berdasarkan distorsi geometris namun tidak diketahui adanya metrik yang mengambil distorsi ke dalam suatu *account* (Fabien, 2000).

2.5 *Mean Opinion Score* (MOS)

SNR merupakan cara sederhana untuk mengukur kebisingan yang diperkenalkan oleh *watermark* yang tertanam dan dapat memberikan ide umum mengenai *imperceptibility* namun hal

tersebut tidak mempertimbangkan karakteristik yang spesifik dari sistem pendengaran manusia sehingga dipergunakan *Perceptual Audio Quality Measure* (PAQM) (Arnold, 2003). Hal tersebut menunjukkan bahwa korelasi antara PAQM dan *Mean Opinion Score* (MOS) adalah 0,98 (Al-Haj, 2011). Oleh karena itu, nilai PAQM akan dipetakan dengan skala penilaian MOS yang ditunjukkan pada tabel 2.2.

Tabel 2. 2 Skala Tingkatan MOS (Arnold, 2003)

Tingkatan MOS	Deskripsi
5	Tidak terlihat
4	Terlihat namun tidak mengganggu
3	Sedikit mengganggu
2	Mengganggu
1	Sangat Mengganggu

2.6 *Adobe Audition* 3

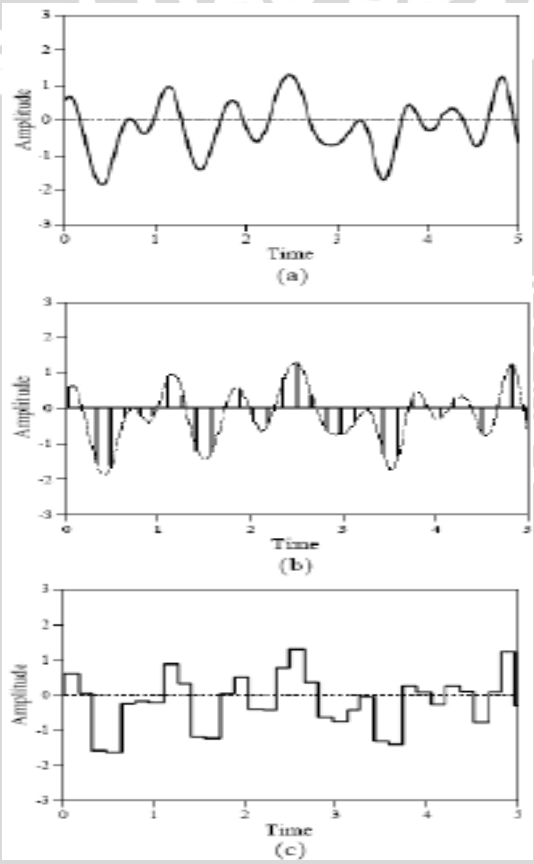
Adobe Audition 3 memberikan gambaran visual yang baik mengenai mono ataupun stereo dari *file audio* dalam tampilan berbentuk gelombang. Hal tersebut memungkinkan untuk melakukan proses *edit* dan penyempurnaan *file* asli dalam beberapa cara, baik sebagai kisaran yang dipilih ataupun *file audio* secara keseluruhan. *Adobe Audition 3* dapat beroperasi dalam mode destruktif dimana pengeditan apapun secara langsung akan mempengaruhi *file audio* asli yang dalam banyak kasus adalah hal yang diinginkan oleh konsumen. *Adobe Audition 3* juga dilengkapi dengan sebuah spektral pendamping untuk melihat *audio* dengan gaya *Adobe Photoshop*. Perubahan yang dilakukan untuk setiap pengeditan *file audio* dapat dibatalkan dengan menggunakan fungsi *undo* standar sampai ketika *file* disimpan.

2.7 Konsep *Audio* Digital

Telinga manusia peka terhadap suara yang secara fisik merupakan variasi cepat dalam tekanan udara (Rusdianto, 2009). Suara dapat direpresentasikan sebagai fungsi :

Audio : Waktu → tekanan

dimana tekanan adalah nilai yang merepresentasikan tekanan udara dan waktu merupakan kumpulan nilai yang merepresentasikan interval waktu antar sinyal. Sinyal suara secara umum dapat dibedakan menjadi tiga jenis yaitu sinyal analog, sinyal diskrit, dan sinyal digital (Rusdianto, 2009). Ketiga sinyal tersebut dapat dilihat pada gambar 2.3 (a), (b), dan (c). Gambar tersebut menunjukkan sinyal analog pada gambar (a), sinyal diskrit pada gambar (b), sinyal digital pada gambar (c).

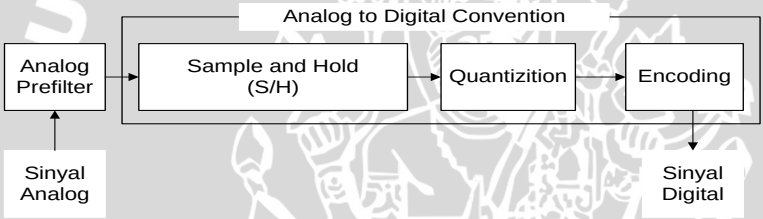


Gambar 2. 3 Sinyal Analog (a), Diskrit (b), Digital (c) (Rusdianto, 2009)

Sinyal analog adalah sinyal yang nilainya terdefinisi pada setiap titik interval waktu secara kontinu. Karakteristik sinyal masih utuh karena belum mengalami perubahan. Sinyal diskrit merupakan hasil proses pencuplikan dari sinyal analog. Parameter yang berpengaruh dalam proses pencuplikan adalah *sampling rate* yang merupakan jumlah *sample* per unit waktu dan disebut juga *sampling periode*. Sinyal digital adalah sinyal diskrit yang melalui proses kuantisasi (Rusdianto, 2009).

2.7.1 Proses Pengubahan Sinyal Analog ke Digital

Proses pengubahan sinyal analog ke sinyal digital terdiri atas *prefiltering*, *sampling*, *quantization*, dan *encoding*. *Analog prefilter* berfungsi membatasi *bandwidth* frekuensi sinyal analog (*prefiltering*). (Rusdianto, 2009).



Gambar 2. 4 Pengubahan Sinyal Analog ke Digital (Rusdianto, 2009)

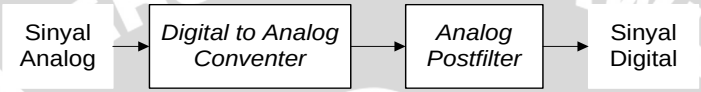
Sampling and Hold (S/H) berfungsi untuk mencuplik sinyal analog dan menyimpan nilainya sampai *Analog to Digital Converter* (ADC) mengubah sinyal analog menjadi sinyal digital. Pencuplikan sinyal (*sampling*) dilakukan pada laju tertentu yang disebut dengan *sampling rate*. Penentuan waktu *sampling* (T_s) jika salah dapat menyebabkan sinyal yang dicuplik mengalami distorsi. Suatu sinyal analog mempunyai frekuensi tertinggi f_h dan dicuplik dengan frekuensi pencuplikan $f_s=1/T_s$ (Rusdianto, 2009).

Proses selanjutnya adalah kuantisasi (*quantization*). Jumlah tingkat kuantisasi berkaitan dengan resolusi sistem pengukuran. Pengukuran sinyal dengan amplitudo 1 dengan sebuah ADC beresolusi 4 bit memilik 16 tingkat kuantisasi dengan besar tingkat kuantisasi masing-masing 1/16 dengan setiap perubahan 1 tingkat dari 16 tingkat yang mengakibatkan perubahan sebesar 1/16.

Encoding adalah proses pengubahan sinyal diskrit hasil kuantisasi menjadi sinyal digital (Rusdianto, 2009).

2.7.2 Prinsip Pengubahan Sinyal Digital ke Analog

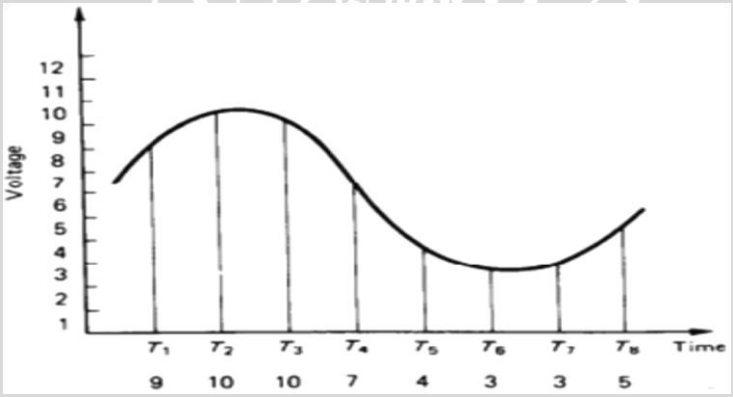
Sinyal digital dari komputer dapat diubah menjadi sinyal dengan *Digital to Analog Converter* (DAC). DAC mengubah sinyal digital menjadi sinyal analog yang masih merupakan gabungan *step*. Sinyal tersebut dilewatkan pada *analog postfilter* sehingga terbentuk sinyal analog kontinu (Rusdianto, 2009).



Gambar 2. 5 Pengubahan Sinyal Digital ke Sinyal Analog (Rusdianto, 2009)

2.7.3 Representasi *Audio* Analog dalam Sinyal Digital

Suatu sistem yang disebut dengan *Pulse Code Modulation* (PCM) digunakan sebagai alat bantu untuk memahami operasi *audio* digital. PCM dapat digambarkan dengan grafik pada gambar 2.6 (Watkinson, 1994).



Gambar 2. 6 Grafik *Pulse Code Modulation* (PCM) (Watkinson, 1994)

Time direpresentasikan secara diskrit dan *waveform* ditunjukkan oleh angka-angka dengan interval yang beraturan. Hal

tersebut disebut dengan *sampling*. Frekuensi *sample* berada disebut dengan *sampling rate* atau *sampling frequency* (Fs). Frekuensi adalah jumlah *sample* yang diambil dalam satu detik. Kualitas media *audio* dengan frekuensi 44,1 KHz sebagai contoh dapat diartikan dalam satu detik sebagai *sample* data sebanyak 44.100 *sample*. Mono atau stereo menunjukkan jumlah *layer* yang digunakan.

Panjang *bit* menentukan ketepatan pengukuran *sample* (*precision*). Nilai *sample* dalam media *audio* ada diantara rentang [-1, +1]. Panjang *bit* yang digunakan adalah 8 *bit* dengan *bit* yang pertama menunjukkan nilai (+) atau (-) dan 7 *bit* sisanya merepresentasikan nilai *sample*. Penggunaan *bit* sebesar 16 bit, berarti 1 *bit* pertama untuk menunjukkan nilai + (0) atau - (1) dan 15 *bit* sisanya merepresentasikan nilai *sample*.

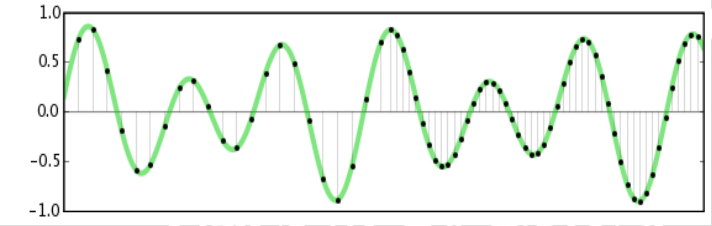
2.7.4 Kualitas *Audio* Digital

Kualitas rekaman *audio* digital sangat bergantung pada dua faktor yaitu *sample rate* dan *sample format* atau *bit depth*. Proses meningkatkan tingkat *sample* atau jumlah *bit* pada setiap *sample* selain dapat meningkatkan kualitas rekaman tetapi juga meningkatkan jumlah ruang yang digunakan oleh *file audio* pada komputer atau disk (Gale, 2010).

Sample rate diukur dalam hertz (Hz) atau siklus per detik. Nilai hanya mewakili jumlah *sample* yang diambil per detik untuk mewakili bentuk gelombang. Pertambahan *sample* per detik berarti peningkatan resolusi yang menyebabkan pengukuran lebih tepat dapat diwakili dalam bentuk gelombang. Telinga manusia peka terhadap pola suara dengan frekuensi antara sekitar 20 Hz sampai 20.000 Hz. Suara pada kisaran tak terdengar meskipun para peneliti subyektif membuktikan adanya konsistensi *pysychoacoustic* yang dapat didengar di atas batas yang seharusnya lebih dari 20.000 kHz (Gale, 2010).

Proses menangkap suara pada frekuensi tertentu memerlukan sampling *rate* dengan batas minimal dua kali frekuensi yang dikenal sebagai frekuensi *Nyquist*. Oleh karena itu, *sample rate* bernilai 40.000 Hz adalah batas minimum mutlak yang diperlukan untuk mereproduksi suara dalam rentang pendengaran manusia. Proses pada tingkat lebih tinggi yang diketahui dapat meningkatkan kualitas lebih jauh dengan menghindari artefak *aliasing* sekitar frekuensi

Nyquist disebut *over sampling*. Tingkat *sample* yang digunakan oleh CD *audio* adalah 44100 Hz. Suara manusia dapat dipahami bahkan jika frekuensi di atas 4.000 Hz dieliminasi karena dalam telepon sebenarnya hanya mengirimkan frekuensi antara 200 Hz dan 4.000 Hz. Oleh karena itu *sample rate* umum untuk rekaman *audio* adalah 8.000 Hz yang disebut dengan kualitas pidato. Penyaringan secara ketat yang disebut *anti-aliasing filter* diperlukan atas frekuensi *Nyquist* untuk melarang sinyal di atas titik *cutoff* dari yang terlipat kembali pada kisaran terdengar oleh *converter* digital sehingga menciptakan distorsi artefak kebisingan *aliasing*.



Gambar 2. 7 Gelombang dengan *Sample Rate* Rendah dan Tinggi (Gale, 2010)

Tingkat *sample* yang umum diukur dalam kilo *hertz* (kHz atau 1000 Hz) adalah 8 kHz, 16 kHz, 22,05 kHz, 22,25 kHz, 44,1 kHz, 48 kHz, 96 kHz, dan 192 kHz. *Sample rate* yang paling umum adalah 44,1 kHz (44.100 Hz) (Gale, 2010).

Gambar 2.7 menunjukkan bagian sebelah kiri memiliki *sample rate* yang rendah dan bagian kanan memiliki *sample rate* tinggi yang disebut juga resolusi tinggi.

Parameter lain dari kualitas *audio* adalah format *sample* atau kedalaman *bit* yang diukur dengan jumlah *bit* untuk mewakili setiap *sample* sehingga ketepatan representasi setiap *sample* dapat diperoleh. Proses meningkatkan jumlah *bit* mengakibatkan peningkatan rentang dinamis maksimum rekaman *audio* sehingga perbedaan *volume* antara suara paling keras dan paling lembut yang dapat diwakili (Gale, 2010).

Rentang dinamis diukur dalam skala desibel (dB). Telinga manusia dapat melihat suara dengan *dynamic range* paling sedikit 90 dB. Kemungkinan ide yang baik dapat dimunculkan untuk merekam *audio* digital dengan *dynamic range* lebih dari 90 dB sehingga suara

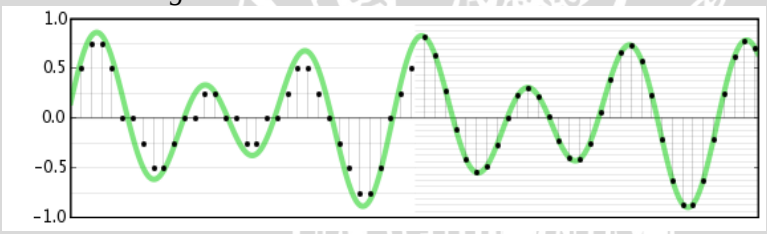
yang terlalu lunak dapat diperkuat untuk konsistensi yang maksimal. Sinyal yang dicatat pada tingkat rendah dapat ditingkatkan menjadi normal untuk mengambil keuntungan dari jangkauan dinamis yang tersedia karena rekaman sinyal tingkat rendah tidak akan menggunakan semua kedalaman *bit* yang tersedia dengan konsekuensi kehilangan resolusi yang tidak dapat diselesaikan dengan normalisasi keseluruhan tingkat gelombang digital.

Format *sample* umum dan jangkauan dinamis meliputi:

- 8 *bit integer*: 48 dB
- 16 *bit integer*: 96 dB
- 24 *bit integer*: 145 dB
- 32 *bit floating point*: mendekati nilai tak terbatas dalam dB

Pembatasan praktis terdapat pada rentang dinamis karena kemampuan perangkat keras, perangkat *input*, dan konverter *output* mengakibatkan batas menjadi lebih praktis seperti 90 dB untuk 16 *bit*.

Bagian setengah kiri pada gambar 2.8 memiliki format *sample* dengan beberapa *bit* dan bagian kanan memiliki format *sample* dengan *bit* lebih banyak dengan tingkat *sample* sebagai jarak antara *vertical grid lines* yang menunjukkan format *sample* adalah jarak antara *horizontal grid lines*.



Gambar 2. 8 Gelombang dengan *Sample Format* (Gale, 2010)

2.7.5 *Waveform Audio File Format* (WAV)

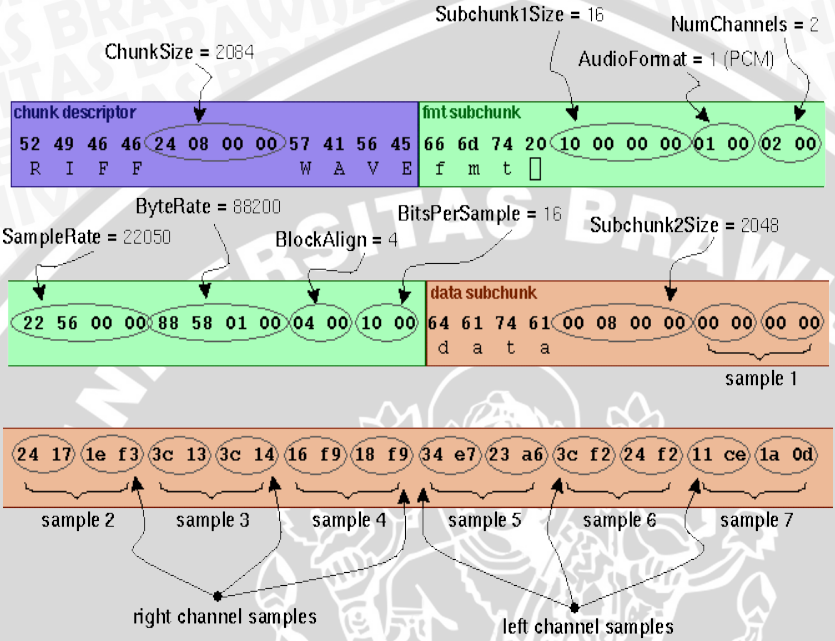
Waveform Audio File Format atau disingkat dengan WAV adalah sebuah standar format *file audio* Microsoft dan IBM untuk menyimpan *audio* dalam sebuah *Personal Computer* (PC). WAV adalah sebuah varian dari metode format *bit stream* RIFF untuk menyimpan data dalam potongan-potongan (*chunks*) dan mirip dengan format IFF dan AIFF yang digunakan oleh komputer-komputer Macintosh. Format WAV ataupun format AIFF cocok

dengan sistem operasi Windows dan Macintosh. Format RIFF berperan sebagai pembungkus untuk berbagai macam kompresi *audio*. RIFF adalah bentuk utama pada sistem Windows untuk raw *audio*. Format WAV yang paling umum berisi *audio* yang belum dikompresi dalam bentuk *Pulse-Code Modulation* (PCM) meskipun sebuah *file* WAV dapat menahan *audio* yang dikompresi. *Audio* PCM adalah format *audio* standar untuk *Compact Disc* (CD) pada 44.100 *sample* per detik dan 16 *bit* per *sample*. *PCM* menggunakan metode penyimpanan yang tidak terkompresi (*lossless*) untuk menyimpan semua *sample* dari sebuah *track audio* sehingga pengguna ahli (*professional user*) atau ahli *audio* dapat menggunakan format WAV untuk *audio* kualitas maksimum. *Audio* WAV juga dapat dirubah dan dimanipulasi dengan sistem.

Format WAV terbatas pada *file* yang berukuran kurang dari 4 GB karena penggunaan 32 *bit integer* yang belum tertanda untuk merekam ukuran *header file* dan beberapa program membatasi ukuran *file* sampai 2 GB. Hal tersebut sama dengan sekitar 6,6 jam *CD* kualitas *audio* (44,1 kHz, 16-*bit* stereo).

Sebuah *file* WAV merupakan kumpulan sejumlah potongan yang berbeda jenis. Potongan format yang diperlukan diinisialisasi sebagai *fmt* berisi parameter penting untuk menggambarkan bentuk gelombang (*waveform*) seperti tingkat *sample*. Potongan data yang berisi data aktual *waveform* juga diperlukan. Potongan yang lain adalah pilihan. Potongan pilihan dapat terdiri atas poin isyarat, daftar parameter instrumen, menyimpan informasi aplikasi spesifik, dan lainnya. Semua aplikasi yang menggunakan WAV harus dapat membaca 2 potongan yang diperlukan dan dapat memilih untuk mengacuhkan potongan (*chunks*) pilihan secara selektif. Sebuah program yang meniru sebuah WAV seharusnya menggandakan semua potongan dalam WAV meskipun memilih untuk tidak memunculkannya.

Interpretasi dari tiap *byte* pada *file wave* tersebut dijelaskan pada gambar berikut:



Gambar 2. 9 Interpretasi *Byte* pada WAV (Gale, 2010)

Tabel 2. 3 Struktur *File* WAV (Kurniawan, 2010)

Offset	Size	Nama Field	Deskripsi
0	4	*ChunkID*	Terdiri atas kata "RIFF" dalam bentuk ASCII (0x52494646 dalam bentuk *big-endian*).
4	4	*Chunksize*	36 + *SubChunk2Size* atau lebih tepatnya : 4 + (8 + *SubChunk1Size*) + (8 + *SubChunk2Size*). *Chunksize* adalah besar seluruh *file* dalam *byte* dikurangi 8 *byte* untuk 2 *field* yang tidak termasuk dalam hitungan: *ChunkID* dan *ChunkSize*.
8	4	*Format*	Terdiri atas kata "*Wave*" (0x57415645 dalam bentuk *big-endian*).
12	4	*SubChunk1ID*	Terdiri atas kata "fmt" (0x666d7420

			dalam bentuk *big-endian*).
16	4	*SubChunk1Size*	16 untuk jenis PCM
20	2	*AudioFormat*	PCM = 1 (*linear quantizatization*). Nilai lebih dari 1 mengidentifikasikan *file* WAV kompresi.
22	2	*NumChannels*	Mono = 1, Stereo = 2 dan seterusnya
24	4	*SampleRate*	800, 44100, dan seterusnya dalam satuan Hz
28	4	*ByteRate*	*=SampleRate*NumChannels*BitsPer Sample*/8
32	2	*`BlockAlign*	*= NumChannels * BitsPerSample* / 8 Jumlah *byte* untuk satu *sample* termasuk semua *channel*
34	2	*BitsPerSample*	8 bit = 8, 16 bit = 16, dan seterusnya
36	4	*SubChunk2ID*	Terdiri atas kata "*data*" (0x64617461 dalam bentuk *big-endian*)
40	4	*SubChunk2Size*	*=NumSamples*NumChannels*BitsPer Sample*/8
44	*	*Data*	*Data sound* sebenarnya

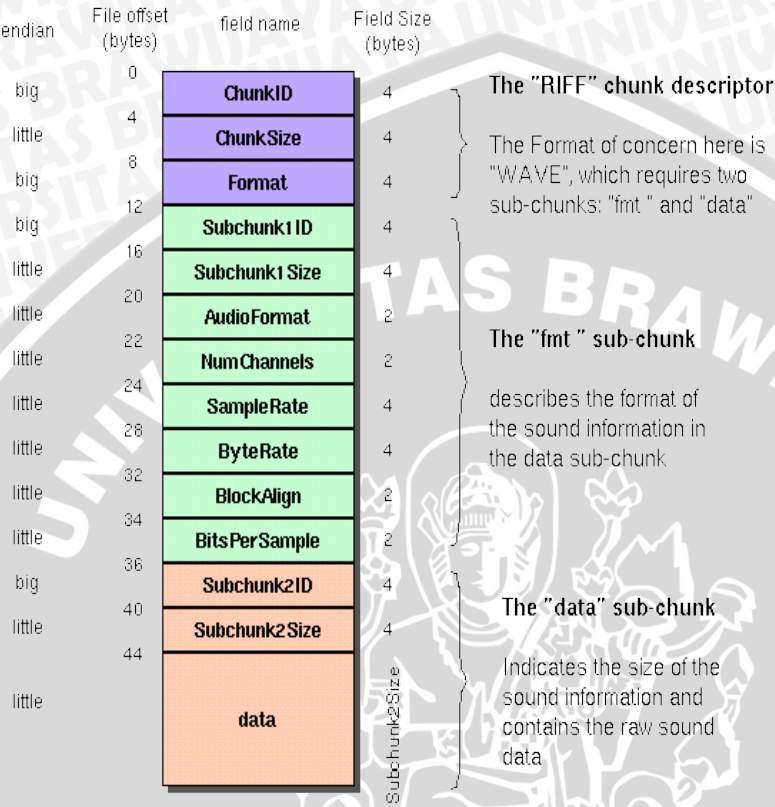
Keterangan:

a. Format WAV terdiri atas 2 *SubChunk2* yaitu *fmt* dan *data*.
b. *SubChunkfmt* menggambarkan format *data sound.*
c. *SubChunkdata* terdiri atas ukuran besar *data* dan *sound* sebenarnya.

Format *Pulse Code Modulation* (*PCM*) adalah *file* WAV yang tidak terkompresi sehingga ukuran *file* sangat besar jika *file* mempunyai durasi yang panjang. Proses penyisipan data dilakukan pada *file* WAV dimulai dari *byte* ke-44 dari *file offset*. Penjelasan format *file* WAV ditunjukkan pada tabel 2.3.

Berikut adalah 72 *byte* pada *file* WAV sebagai contoh dengan menampilkan *byte* sebagai heksa desimal:

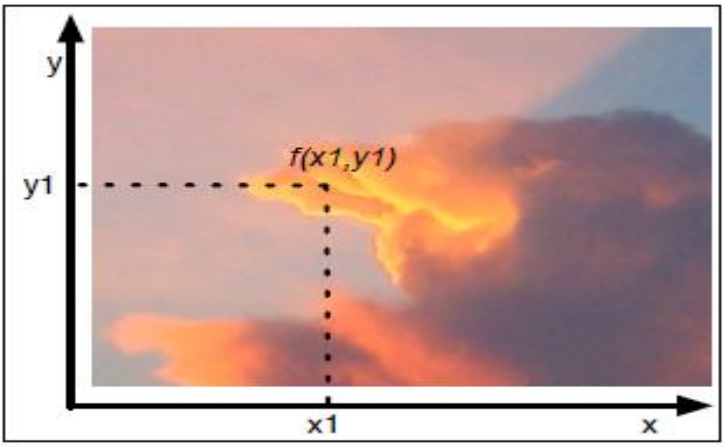
```
52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02
00 22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00
00 00 24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce
1a 0d
```



Gambar 2. 10 Format WAV *File*
(Wilson, 2003)

2.8 Konsep Gambar Digital

Gambar digital dapat didefinisikan sebagai fungsi dua variabel *f(x,y)* dimana *x* dan *y* adalah koordinat spasial dan nilai *f (x,y)* adalah tingkat kecerahan atau derajat keabuan (*brightness/gray level*) dan informasi gambar pada koordinat tersebut. Hal tesebut diilustrasikan pada gambar 2.11. Teknologi dasar untuk menciptakan dan menampilkan warna pada gambar digital didasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (Friskayanti, 2011).



Gambar 2. 11 Gambar Digital (Friskayanti, 2011)

Sebuah gambar diubah ke bentuk digital dan disimpan dalam memori komputer atau media lain. Proses mengubah gambar ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya *scanner*, kamera digital, dan *handy cam.* Sebuah gambar diubah ke dalam bentuk digital selanjutnya disebut gambar digital memungkinkan bermacam-macam proses pengolahan gambar dapat diperlakukan terhadap gambar tersebut.

Pengolahan gambar digital dapat dilakukan dengan cara-cara sebagai berikut:

a. Representasi dan pemodelan gambar.
b. Peningkatan kualitas gambar.
c. Restorasi gambar.
d. Analisis gambar.
e. Rekonstruksi gambar.
f. Kompresi gambar.

2.8.1 Warna dan Ruang Warna

Friskayanti (2011) menyebutkan bahwa "warna merupakan hasil persepsi dari warna cahaya dalam spektrum wilayah yang terlihat oleh retina mata dengan panjang gelombang antara 400/nm sampai dengan 700/nm" sedangkan penulis yang lain menyebut warna sebagai "reaksi otak dalam menerima rangsangan visual".

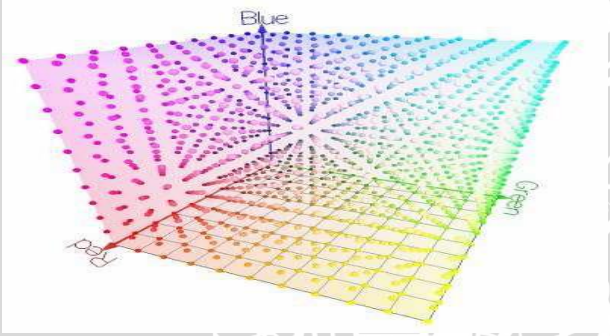
Ruang warna atau model warna merupakan sebuah metode yang dapat digunakan untuk menentukan, membuat, dan memvisualisasikan warna. Beberapa riset hanya membahas beberapa ruang warna yang biasa digunakan untuk aplikasi *watermarking*. Beberapa ruang warna tersebut antara lain adalah sebagai berikut :

a. *Red Green Blue* (RGB).
b. *Hue Saturation Intensity* (HSI).
c. *Luminance-Chrominance* (YC_bC_r).

Ruang warna yang digunakan dalam skripsi adalah ruang warna RGB dan YC_bC_r.

2.8.2 *Red-Green-Blue* (RGB)

Gambar berwarna umumnya memiliki ruang warna RGB. Ruang warna RGB dapat divisualisasikan sebagai sebuah kubus dengan tiga sumbunya yang mewakili komponen warna merah (*red*) R, hijau (*green*) G, dan biru (*blue*) B. Ruang warna RGB ditunjukkan pada gambar 2.12.



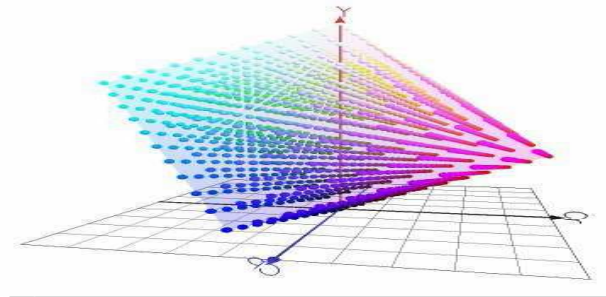
Gambar 2. 12 Ruang Warna RGB
(Friskayanti, 2011)

RGB sering digunakan di dalam sebagian besar aplikasi komputer karena dengan ruang warna tersebut tidak diperlukan transformasi untuk menampilkan informasi di layar monitor. Hal tersebut juga menyebabkan RGB banyak dimanfaatkan sebagai ruang warna dasar bagi sebagian besar aplikasi (Friskayanti, 2011).

2.8.3 *Luminance-Chrominance* (YC_bC_r)

YC_bC_r merupakan standar internasional bagi pengkodean *video* dan televisi yang didefinisikan di CCIR *Recommended* 601

(Friskayanti, 2011). Y merupakan komponen *luminance* sedangkan C_b dan C_r adalah komponen *chrominance*. Ruang warna YC_bC_r ditunjukkan pada gambar 2.13.



Gambar 2. 13 Ruang Warna *Luminance* dan *Chrominance* (Friskayanti, 2011)

YC_bC_r dapat diperoleh dari RGB dengan menggunakan persamaan berikut:

$$Y = 0.299*R + 0.587*G + 0.114*B \quad (2.18)$$

$$C_b = -0.1687*R - 0.3313*G + 0.5*B + 128 \quad (2.19)$$

$$C_r = 0.5*R-0.418*G-0.0813*B + 128 \quad (2.20)$$

Konversi YC_bC_r ke RGB dapat dilakukan dengan persamaan berikut (Friskayanti, 2011) :

$$R = Y+1.402(C_r-128) \quad (2.21)$$

$$G = Y-0.34414(C_b-128)-0.71414(C_r-128) \quad (2.22)$$

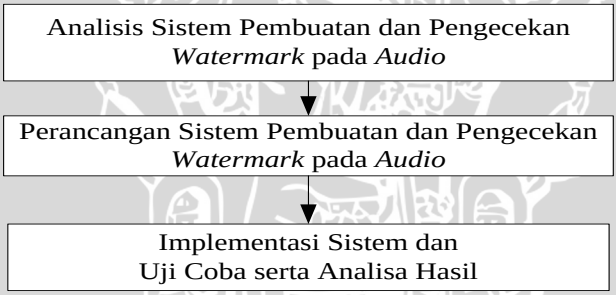
$$B = Y+1.772(C_b-128) \quad (2.23)$$

BAB III
METODOLOGI DAN PERANCANGAN

Bab metodologi dan perancangan menjelaskan tentang penggunaan metode dalam pembuatan sistem untuk proses *watermarking*, pengujian secara manual, dan penjelasan langkah-langkah dalam pembuatan sistem.

Tahapan pembuatan sistem dapat dijelaskan sebagai berikut:

a. Proses diawali dengan mempelajari metode yang digunakan pada jurnal yang pernah ada dengan penjelasan pada bab sebelumnya.
b. Pembuatan sistem berdasarkan analisis dan perancangan.
c. Uji coba sistem dengan menggunakan *audio* berjenis instrumental dan klasik.
d. Evaluasi hasil pengujian yang dilakukan oleh sistem untuk mengetahui ketahanan *audio* yang diberi *watermark*.



Gambar 3. 1 Diagram Alir Pembuatan Sistem

3.1 Deskripsi Umum Sistem

Sistem dirancang untuk dapat melakukan suatu penyisipan *watermark* ke dalam sebuah *audio* digital. *Watermark* yang dipilih berupa gambar *gray-scale* yang berperan sebagai *input* pada sistem.

Sistem dibangun pada satu lingkungan pengembangan yaitu *Personal Computer* (PC) sehingga proses *watermarking* dan ekstraksi dilakukan pada satu lingkungan pengembangan. Oleh karena itu, *user* harus memilih proses saat sistem mulai dijalankan.

Proses *watermarking* bertujuan untuk melakukan *watermarking* pada *audio* asli sedangkan proses ekstraksi bertujuan

untuk melakukan proses ekstraksi *watermark* dari *audio* digital yang sudah memiliki *watermark*.

Ekstraksi *watermark* dirancang untuk perbandingan antara *audio* asli dengan *audio* yang diberikan *watermark* untuk mendapatkan *watermark*. *Audio* yang diberikan *watermark* dimasukkan ke dalam sistem sehingga dapat dilakukan proses ekstraksi *watermark*.

3.2 Batasan Sistem

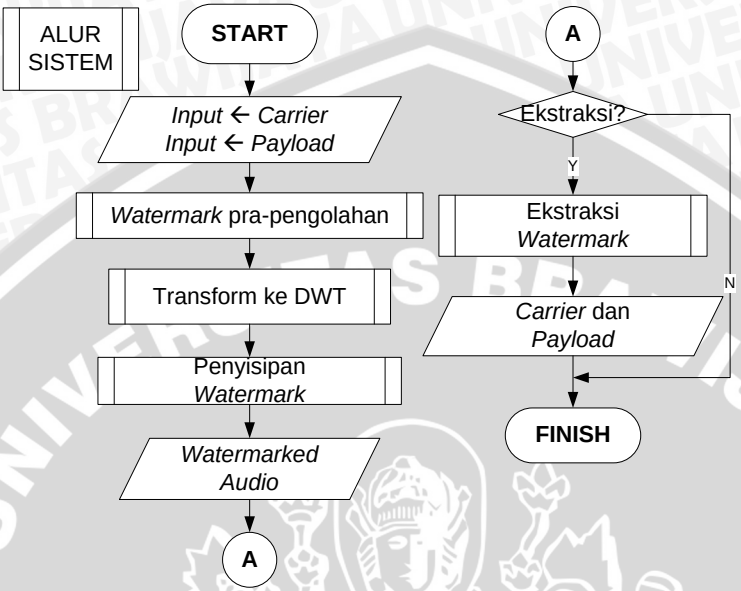
Batasan dari sistem yang dikembangkan yaitu:

a. Sistem hanya memproses *audio* berjenis musik dengan ekstensi WAV.
b. Frekuensi maksimal *audio* digital yang diperbolehkan adalah 192 KHz.

3.3 Perancangan Kerja Sistem

Perancangan kerja sistem membahas arsitektur dan proses yang terjadi pada sistem berdasarkan deskripsi umum sistem yang dijelaskan. Beberapa hal mengenai *flowchart* yang perlu diketahui bahwa terdapat beberapa istilah untuk memudahkan penyampaian informasi antara lain:

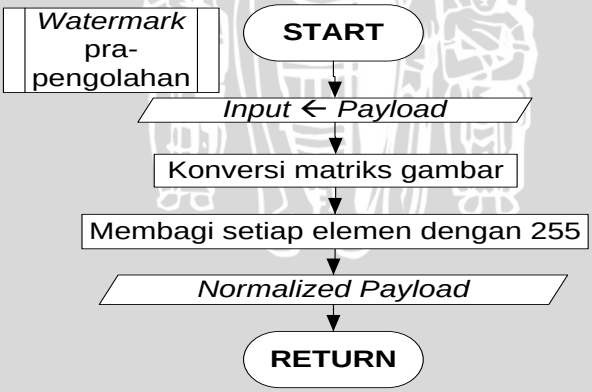
a. *Carrier* digunakan untuk mewakili representasi dari *input* berupa *audio* asli.
b. *Payload* digunakan untuk mewakili representasi dari *input* berupa gambar dan *audio* yang memiliki *watermark*.
c. Semua variabel yang berhubungan masing-masing dengan *carrier* dan *payload* direpresentasikan sesuai dengan hal tersebut.



Gambar 3. 2 Diagram Alir Sistem

3.3.1 Perancangan Proses Pembuatan *Input* Sistem

Audio yang diproses di dalam sistem adalah *audio* instrumental dan klasik yang berekstensi WAV. Penyisipan untuk *watermark* akan diujikan pada sinyal analog berupa *wavelet*.



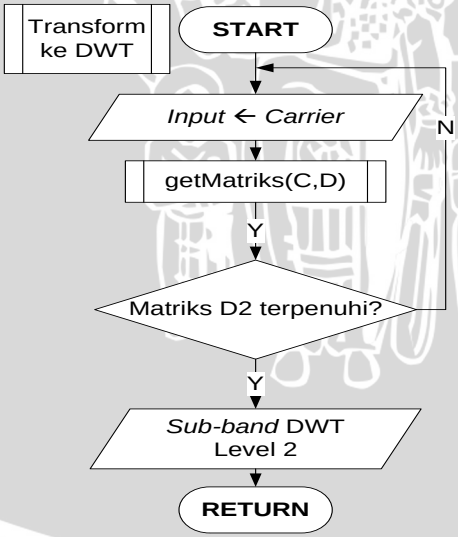
Gambar 3. 3 Diagram Alir Vektorisasi

Audio yang digunakan pada proses *input* dalam sistem adalah musik instrumental dan klasik. Penyisipan *watermark* pada *audio* akan diujikan pada sinyal analog untuk sebagian sinyal digital dengan menyisipkan *watermark* dari gambar *gray-scale* sebagai matriks dua dimensi yang berukuran M1xM2. Gambar tersebut dikonversi menjadi matriks gambar dua dimensi dan diubah menjadi vektor satu dimensi dengan panjang M1xM2. Vektor tersebut dinormalkan dengan membagi setiap elemen dengan 255.

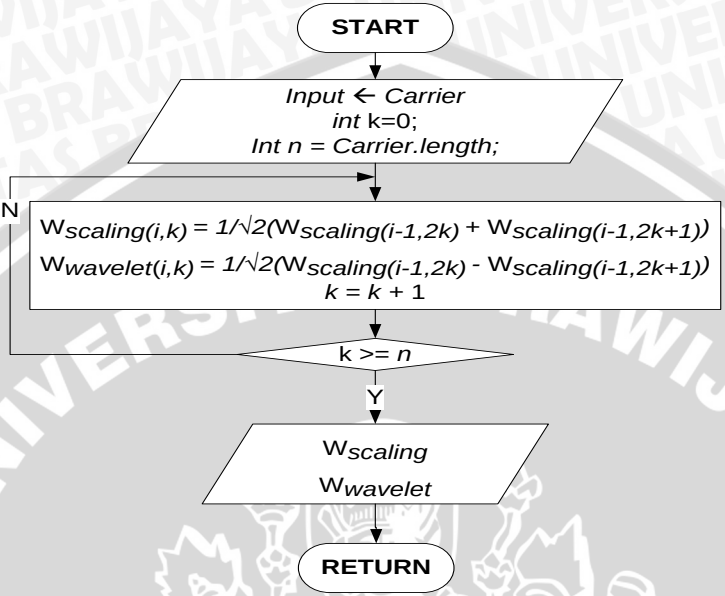
3.3.2 Perancangan Proses Pembuatan *Watermark* DWT

Proses pembuatan *watermark* dengan struktur DWT dilakukan dengan mengubah *watermark* ke dalam domain frekuensi menggunakan perhitungan yang dijelaskan pada persamaan 2.7 sampai dengan 2.9.

Sinyal pada *discrete wavelet transform* dianalisis pada frekuensi berbeda dengan resolusi yang berbeda melalui dekomposisi sinyal sehingga menjadi detil informasi dan taksiran kasar. *Discrete wavelet transform* bekerja pada dua kumpulan fungsi yang disebut fungsi penskalaan dan fungsi *wavelet* yang masing-masing berhubungan dengan *low-pass filter* dan *high-pass filter*.



Gambar 3. 4 Diagram Alir *Down-Sampling*

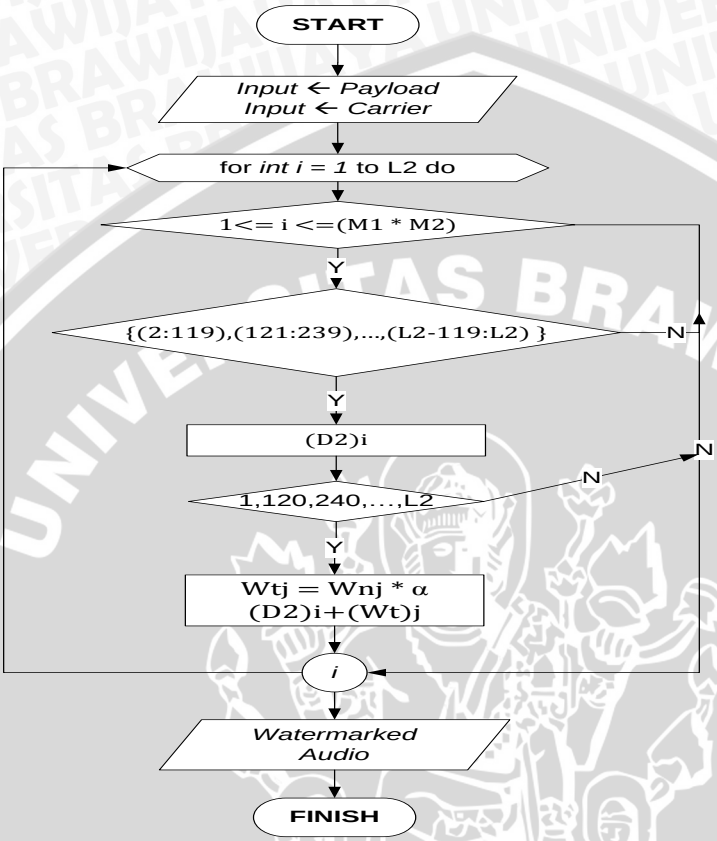


Gambar 3. 5 Diagram Alir Penentuan Nilai Fungsi *Sampling* dan *Wavelet*

3.3.3 Perancangan Proses Penyisipan *Watermark*

Proses penyisipan *watermark* dapat dijelaskan dalam langkah-langkah berikut:

a. *Watermark* diterapkan menggunakan DWT level dua ke *left channel* setelah proses *watermark* pra-pengolahan dari sinyal *audio* stereo.
b. *watermark* vektor ternormalisasi yang dinyatakan oleh persamaan 2.9 disisipkan ke dalam level kedua dari rincian *sub-band* D2 dari *left channel*.
c. Prosedur *embed* menghasilkan level kedua yang diberi *watermark* dengan rincian *sub-band* (WD2) seperti persamaan 2.10 dan 2.11.



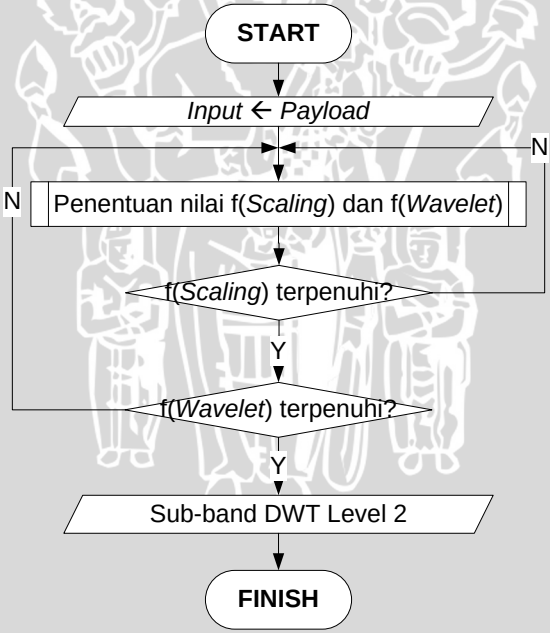
Gambar 3. 6 Diagram Alir Proses Penyisipan *Watermark*

3.3.4 Perancangan Proses Ekstraksi Gambar *Watermark*

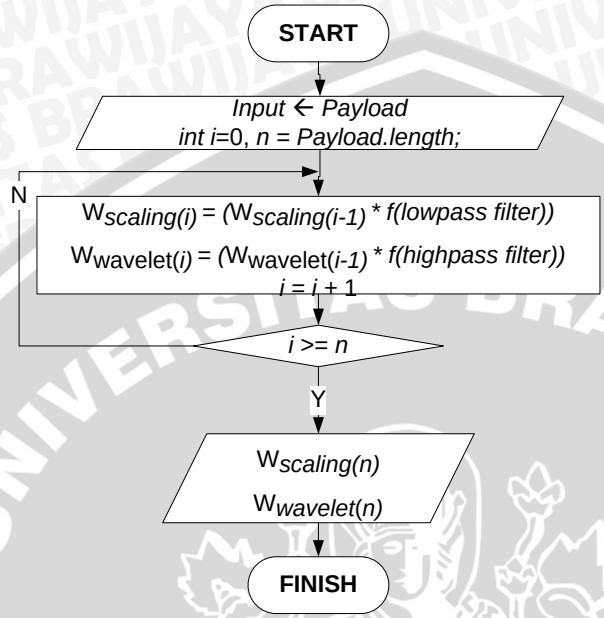
Prosedur ekstraksi *watermark* memungkinkan pemilik dari klip *audio* untuk mengekstrak *watermark* yang disisipkan. Prosedur tersebut membutuhkan pengetahuan dari *file audio* asli, intensitas *watermark*, dan ukuran *watermark* untuk mengekstrak *watermark*.

Ekstraksi *watermark* adalah langkah pembalikan langsung dari langkah-langkah yang dilakukan dalam prosedur *embed*. Diagram alir untuk proses ekstraksi *watermark* ditunjukkan pada gambar 3.7.

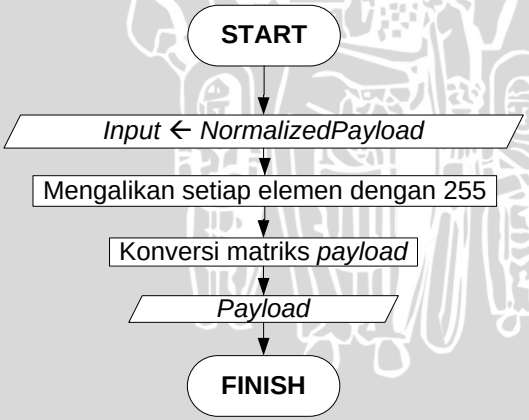
Gambar 3. 7 Diagram Alir Proses Ekstraksi *Watermark*



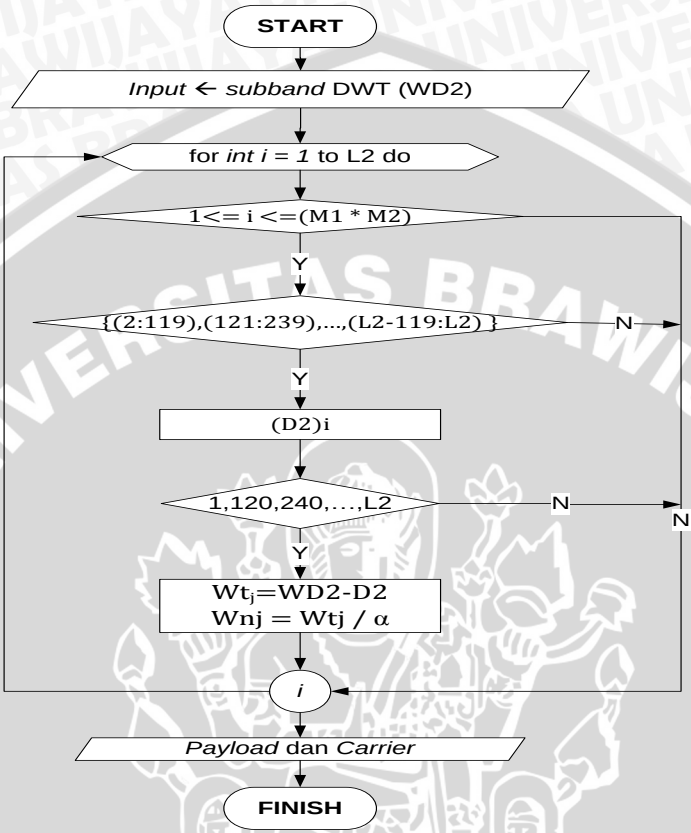
Gambar 3. 8 Diagram Alir Transformasi ke DWT



Gambar 3. 9 Diagram Alir Penentuan Nilai Fungsi *Sampling* dan *Wavelet*



Gambar 3. 10 Diagram Alir Proses *Invers* Vektorisasi



Gambar 3. 11 Diagram Alir Proses Ekstraksi *Watermark*

Prosedur ekstraksi *watermark* memungkinkan pemilik dari klip *audio* untuk mengekstrak *watermark* yang disisipkan. Prosedur tersebut membutuhkan pengetahuan dari *file audio* asli, intensitas *watermark*, dan ukuran *watermark* untuk mendapatkan *watermark*. Ekstraksi *watermark* adalah langkah pembalikan langsung dari langkah-langkah yang dilakukan dalam prosedur *embed*. Langkah-langkah ekstraksi dijelaskan sebagai berikut:

a. Operasi DWT level dua diterapkan pada sinyal *audio* asli dan *audio* yang diberi *watermark*, masing-masing memproduksi satu perkiraan dan dua rincian *sub-band*.
b. Proses ekstraksi dilakukan berdasarkan persamaan 2.12, 2.13, dan 2.14.

c. *Watermark* dikonversi kembali ke matriks dua dimensi yang mewakili *watermark* gambar *gray-scale*.

3.4 Perancangan Uji Coba

Pengujian yang dilakukan dalam sistem dilihat dari kualitas *audio watermark* digital yang dihasilkan dibandingkan dengan *audio* asli dan ketahanan *watermark* dalam *audio* terhadap beberapa serangan *audio* berdasarkan *Adobe Audition 3*. Pengujian dari sisi *imperceptibility* dijelaskan pada subbab 2.3.3.

3.4.1 *Audio* Uji

Spesifikasi *audio* yang akan diujikan yaitu:

a. Ukuran *audio* untuk perhitungan manual maksimal 5 detik.
b. Ukuran *audio* untuk *audio* yang disisipi *watermark* adalah musik dengan rentang waktu sekitar 26 detik.
c. Jenis *audio* yang diuji adalah musik digital berjenis instrumental dan klasik berekstensi WAV.
d. Empat *audio* untuk mendengarkan kualitas *audio* setelah diberikan *watermark*.
e. Gambar yang digunakan sebagai tanda *watermark* adalah gambar dengan format BMP berukuran 20x20 dan 50x50 *pixel*.

3.4.2 Lingkungan Pengujian

Sistem yang digunakan adalah *Microsoft Excel* untuk pengolahan data secara manual dan *Adobe Audition 3* untuk memanipulasi *audio* yang diberi *watermark* (*watermarked audio*). Perangkat keras yang digunakan dalam pengujian sama dengan proses implementasi.

3.4.3 Pengujian Kualitas *Watermarked Audio*

Pengujian kualitas *watermarked audio* menggunakan SNR *dan* PSNR. SNR akan melihat *audio* asli sebagai sinyal $x(t)$ sedangkan untuk *watermarked audio* disimbolkan $x_w(t)$ dengan ukuran *audio* sepanjang t detik. Pengukuran SNR menunjukkan rata-rata perbedaan antara sinyal *audio* asli $x(t)$ dengan sinyal *watermarked audio* $x_w(t)$. Variabel t adalah ukuran panjang waktu *audio*. Perhitungan SNR ditunjukkan dalam persamaan 2.16.

Pengujian nilai PSNR menggunakan persamaan 2.17. Nilai PSNR yang besar menunjukkan semakin baik hasil *watermarking*. *Input* dari pengujian berupa rentang waktu dan *size* dari *audio* asli dan *audio* hasil *watermarking*. Keluaran dari pengujian berupa nilai kesalahan. Semakin besar kesalahannya maka makin kecil nilai PSNR yang dihasilkan dan sebaliknya. Rancangan tabel hasil uji kualitas *audio* yang disisipi *watermark* ditunjukkan pada tabel 3.1.

Tabel 3. 1 Rancangan Tabel Hasil Uji Kualitas

Watermarked audio	*Water-mark*	SNR	PSNR
Audio yang diberi *watermark*	Gambar disisipkan ke *audio* asli	SNR dihitung sesuai persamaan (2.16)	PSNR dihitung sesuai persamaan (2.17)

3.4.4 Pengujian terhadap Ketahanan *Watermark*

Uji coba ketahanan dilakukan dengan memberikan beberapa serangan kepada *audio* yang disisipi *watermark* dan dilakukan pengujian untuk mengetahui *watermark* masih dapat dideteksi atau tidak terdeteksi. Variasi pemrosesan *audio* dapat dilakukan dengan variasi berdasarkan *Adobe Audition 3* antara lain:

a. *Noise reduction effect* pada *audio* beresolusi 32 bit tanpa *DC offset*.
b. Pengubahan kedalaman *bit* dari sebuah *file* dengan proses *dithering*. Penentuan *dither depth* dalam *bit* dengan nilai random antara 0,2 sampai dengan 0,7.
c. Pemberian fungsi distribusi kemungkinan (*probability distribution function*) menggunakan jenis *triangular*.

Rancangan tabel uji ketahanan *audio* yang disisipi *watermark* ditunjukkan pada tabel 3.2.

Tabel 3. 2 Rancangan Tabel Uji Ketahanan

Watermarked audio	Perlakuan	α	Deteksi *watermark*
Nama dari *audio*	Serangan yang dilakukan	*Alpha* yang diberikan	Status *watermark* pada *audio*

Hasil perhitungan untuk uji ketahanan akan disimpan dalam tabel uji ketahanan dengan penyimpanan *watermark* yang dimanipulasi ke dalam *field audio* yang disisipi *watermark* dan *field* perlakuan untuk menyimpan jenis manipulasi yang digunakan. *Field* deteksi *watermark* digunakan untuk menyimpan hasil pendeteksian *watermark* dan α merupakan kekuatan penyisipan yang mengontrol tingkat kekuatan penyisipan *watermark*.

3.4.5 Pengujian *Watermark* terhadap *Nonperceptibility*

Pengujian *watermark* terhadap *nonpercertibility* didasarkan pada *human auditory system. audio* asli yang disisipi *watermark* diujikan pada pendengaran telinga manusia normal untuk membedakan *audio* asli dengan *audio* yang disisipi *watermark*. Pengujian dilakukan dengan melakukan tes terhadap 10 orang berusia 20 - 30 tahun. Rancangan tabel hasil uji *nonperceptibility* dapat dilihat pada tabel 3.3.

Tabel 3. 3 Rancangan Tabel Hasil Uji *Nonperceptibility*

Nama Responden	*Audio*	Hasil
Nama dari penilai	*Audio* yang diberi *watermark*	Nilai diberikan berdasarkan pendengaran penilai

3.5 Contoh Perhitungan Manual Penyisipan *Watermark*

Matriks 3x3 digunakan pada contoh perhitungan manual untuk mempermudah proses perhitungan. *Pixel* gambar dipisahkan menjadi komponen-komponen *red*, *green*, dan *blue* yang ditunjukkan pada tabel 3.4 sampai dengan tabel 3.6.

Tabel 3. 4 Warna *Red* (R)

	0	**1**	**2**
0	38	57	35
1	37	102	40
2	25	35	23

Tabel 3. 5 Warna *Green* (G)

	0	**1**	**2**
0	35	56	39

1	39	106	45
2	29	40	33

Tabel 3. 6 Warna *Blue* (B)

	0	**1**	**2**
0	37	58	40
1	40	107	46
2	30	41	33

3.5.1 Perhitungan Manual Vektorisasi

Perhitungan vektorisasi diawali dengan transformasi RGB ke ruang warna YC_bC_r untuk mendapatkan nilai *gray-scale* dari *watermark* menggunakan persamaan 2.18.

Contoh Perhitungan:

$Y_{(0,0)} = 0.299 * R_{(0,0)} + 0.587 * G_{(0,0)} + 0.114 * B_{(0,0)}$

$Y_{(0,0)} = 0.299 * 38 + 0.587 * 35 + 0.114 * 37$

$Y_{(0,0)} = 11.362 + 20.545 + 4.218$

$\mathbf{Y_{(0,0)} = 36.125}$

Hasil perhitungan transformasi RGB ke Y dengan mengabaikan matriks C_b dan C_r ditunjukkan pada tabel 3.7 sampai dengan 3.9.

Tabel 3. 7 Warna *Luminance* (Y)

	0	**1**	**2**
0	36.125	56.527	37.918
1	38.516	104.918	43.619
2	27.918	38.619	30.01

Struktur warna Y diubah dalam bentuk vektor 1 dimensi dengan mengumpulkan data per kolom menjadi satu kolom seperti tabel 3. 8.

Tabel 3. 8 Vektor 1 Dimensi

Urutan	**Data**
0	36.125
1	38.516
2	27.918
3	56.527

4	104.918
5	38.619
6	37.918
7	43.619
8	30.01

Struktur vektor akan diubah vektor yang dinormalisasi dengan membagi setiap elemen dengan 255 seperti tabel 3.9.

Tabel 3. 9 Vektor Ternormalisasi

Urutan*Watermark*	Data(Wn_i)
0	0.141667
1	0.151043
2	0.109482
3	0.221675
4	0.411443
5	0.151447
6	0.148698
7	0.171055
8	0.117686

Hasil dari vektorisasi akan digunakan untuk menentukan nilai *sampling* dan *wavelet* dalam proses transformasi ke dalam bentuk DWT.

3.5.2 Perhitungan Manual Transformasi ke DWT

Proses transformasi ke DWT diawali dengan menentukan nilai D2 dari *left channel audio* asli sehingga menghasilkan kumpulan *sub-band* berdimensi satu berdasarkan persamaan 2.10 dan 2.11.

Tabel 3. 10 Contoh *Left Channel Audio*

Urutan*Sample(n)*	Data dalam hexa	Data dalam desimal
0	34	52
1	e7	231
2	23	35
3	a6	166
4	3c	60
5	f2	242
6	24	36

7	f2	242
8	11	17
9	ce	206
10	1a	26
11	0d	13

Contoh Perhitungan:

a. Perhitungan koefisien *scaling* data original ke C1

$$a_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} + a_{j,2k+1})$$

Dimana a_j = x (data original), a_{j-1} = C1, dan jika k = 0 maka,

$$C1_k = \frac{1}{\sqrt{2}} * (a_{j,2k} + a_{j,2k+1})$$

$$C1_0 = \frac{1}{\sqrt{2}} * (a_{j,2*0} + a_{j,2*0+1})$$

$$C1_0 = \frac{1}{\sqrt{2}} * (a_{j,0} + a_{j,1})$$

$$C1_0 = \frac{1}{\sqrt{2}} * (52 + 231)$$

$$C1_0 = \frac{1}{\sqrt{2}} * (283)$$

$$C1_0 = 0.7071 * (283)$$

$$C1_0 = 200.1093$$

Perhitungan tersebut menunjukkan $C1_0$ sebesar 200.1093

b. Perhitungan koefisien *wavelet* data original ke C1

$$b_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} - a_{j,2k+1})$$

Dimana a_j = x (data original), b_{j-1} = D1, dan jika k = 0 maka,

$$D1_k = \frac{1}{\sqrt{2}} * (a_{j,2k} - a_{j,2k+1})$$

$$D1_0 = \frac{1}{\sqrt{2}} * (a_{j,2*0} - a_{j,2*0+1})$$

$$D1_0 = \frac{1}{\sqrt{2}} * (a_{j,0} - a_{j,1})$$

$$D1_0 = \frac{1}{\sqrt{2}} * (52 - 231)$$

$$D1_0 = \frac{1}{\sqrt{2}} * (-179)$$
$$D1_0 = 0.7071 * (-179)$$
$$D1_0 = -126.571$$

Perhitungan tersebut menunjukkan $D1_0$ sebesar -126.571

c. Perhitungan koefisien *scaling* C1 ke C2

$$a_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} + a_{j,2k+1})$$

Dimana a_j = C1, a_{j-1} = C2, dan jika k = 0 maka,

$$C2_k = \frac{1}{\sqrt{2}} * (a_{j,2k} + a_{j,2k+1})$$
$$C2_0 = \frac{1}{\sqrt{2}} * (a_{j,2*0} + a_{j,2*0+1})$$
$$C2_0 = \frac{1}{\sqrt{2}} * (a_{j,0} + a_{j,1})$$
$$C2_0 = \frac{1}{\sqrt{2}} * (200.1093 + 142.1271)$$
$$C2_0 = \frac{1}{\sqrt{2}} * (342.2364)$$
$$C2_0 = 0.7071 * (342.2364)$$
$$C2_0 = 241.9954$$

Perhitungan tersebut menunjukkan $C2_0$ sebesar 241.9954

d. Perhitungan koefisien *wavelet* C1 ke D2

$$b_{j-1,k} = \frac{1}{\sqrt{2}} * (a_{j,2k} - a_{j,2k+1})$$

Dimana a_j = C1, b_{j-1} = D2 dan jika k = 0 maka,

$$D2_k = \frac{1}{\sqrt{2}} * (a_{j,2k} - a_{j,2k+1})$$
$$D2_0 = \frac{1}{\sqrt{2}} * (a_{j,2*0} - a_{j,2*0+1})$$
$$D2_0 = \frac{1}{\sqrt{2}} * (a_{j,0} - a_{j,1})$$
$$D2_0 = \frac{1}{\sqrt{2}} * (200.1093 - 142.1271)$$

$$D2_0 = \frac{1}{\sqrt{2}} * (57.9822)$$
$$D2_0 = 0.7071 * (57.9822)$$
$$D2_0 = 40.9992$$

Perhitungan tersebut menunjukkan $D2_0$ sebesar 40.9992

Tabel nilai C1, D1, C2, D2, W_{ti} yang diperoleh dari perkalian W_{ni} dengan $\alpha = 1$, dan WD2 untuk dijumlahkan dengan W_{ti} dapat dilihat pada tabel 3.11 dan 3.12.

Tabel 3. 11 Hasil Contoh Perhitungan C1, D1,C2, dan D2

No *Sample*	C1	D1	C2	D2
0	200,1093	-126,5709	241,9953584	40,99921
120	188,0886	138,5916	245,9952817	19,99962
240	142,1271	-92,6301	251,4951762	-50,499
480	159,8046	74,9526	251,9951666	-25,9995
960	213,5442	-128,6922	289,9944378	11,99977
1920	196,5738	145,6626	268,4948502	9,499818
3840	196,5738	-145,6626	250,4951954	27,49947
7680	183,1389	159,0975	129,4975162	129,4975
15360	157,6833	-133,6419	111,4978614	111,4979

Tabel 3. 12 Hasil Contoh Perhitungan W_{ti} dan WD2

Urutan (i)	W_{ti}	WD2
0	0.141667	41.1408803
1	0.151043	20.1506595
2	0.109482	-50.3895491
3	0.221675	-25.7778268
4	0.411443	12.411213
5	0.151447	9.65126485
6	0.148698	27.6481706
7	0.171055	129.668571
8	0.117686	111.615548

3.5.3 Hasil Rekonstruksi WD2 ke Sinyal Digital

Proses transformasi ke DWT diakhiri dengan menentukan nilai S dari rekonstruksi nilai *left channel audio* yang diberi *watermark*

sehingga menghasilkan kumpulan *sample* berdimensi satu yang dapat dilihat pada tabel 3.13. Tabel 3.13 menunjukkan hasil bahwa nilai *audio* yang disisipi *watermark* tidak jauh berbeda dengan *audio* asli.

Tabel 3. 13 *Audio* yang Disisipi *Watermark*

No *Sample*	S
0	52.067121
120	231.06854
240	35.052141
480	166.10548
960	60.201671
1920	242.06841
3840	36.070991
7680	242.0784
15360	17.056377

3.5.4 Analisa Ketahanan *Watermark*.

Perhitungan *Signal-to-Noise-Ratio* (SNR) dapat ditunjukkan pada tabel 3.14.

Tabel 3. 14 Perbedaan *Audio* Asli dengan *Watermarked Audio*

No *Sample*	Selisih
0	-0.06712
120	-0.06854
240	-0.05214
480	-0.10548
960	-0.20167
1920	-0.06841
3840	-0.07099
7680	-0.37251
15360	-0.05638

Perhitungan nilai SNR dapat dilakukan dengan menggunakan persamaan 2.16.

$$SNR = \frac{\sum_{t=0}^{n} x^2{}_w(t)}{\sum_{t=0}^{n}\left(x(t) - x_w(t)\right)^2}$$

$$SNR = \frac{120.22925^2}{-1.06325^2}$$

$$SNR = \frac{14455.0725555625}{1.1305005625}$$

$$SNR = 12786.4355$$

Perhitungan nilai PSNR dapat dilakukan dengan menggunakan persamaan 2.17.

$$PSNR = 10 \times \log_{10}(SNR)$$

$$PSNR = 10 \times \log_{10}(12786.4355)$$

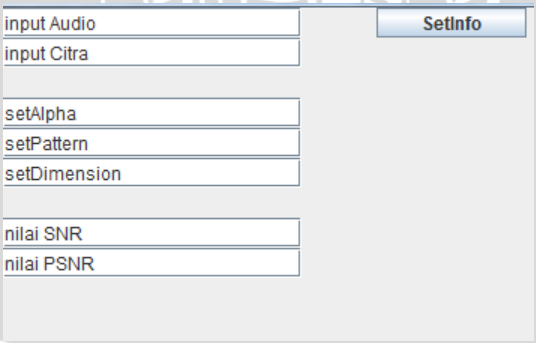
$$PSNR = 10 \times 4.1067$$

$$PSNR = 41.067$$

Tabel 3.14 didapatkan hasil bahwa nilai PSNR adalah 41.067 dB yang menandakan bahwa kualitas *audio* yang disisipi *watermark* cukup baik untuk permisalan penanaman watermark terhadap semua *left channel* dengan *audio* asli yang belum diberikan *watermark.*

3.6 Perancangan Antarmuka

Perancangan antarmuka sistem dapat dilihat pada gambar 3.12 dan 3.13.

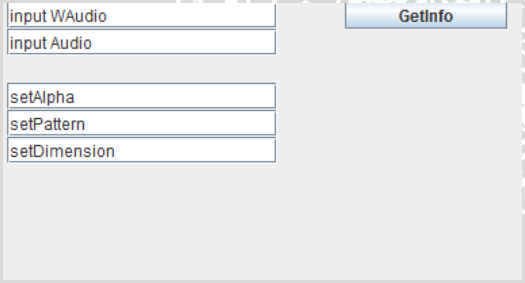


Gambar 3. 10 Antarmuka Penyisipan

Gambar 3.12 menunjukkan antarmuka penyisipan dengan *button* setInfo dengan penjelasan sebagai berikut:

- Input *Audio* digunakan untuk memasukkan berkas *audio* digital dengan ekstensi WAV yang berada dalam *local storage*.
- Input Citra digunakan untuk memasukkan pesan berupa berkas gambar *gray-scale* yang berada dalam *local storage*.
- setAlpha digunakan untuk memasukkan nilai *alpha* yang digunakan sebagai faktor amplifikasi sinyal *audio*.
- setPattern digunakan untuk memasukkan nilai *pattern* yang digunakan sebagai pola letak penyimpanan informasi pada sinyal *audio*.
- setDimension digunakan untuk memasukkan nilai *dimension* yang digunakan sebagai jumlah informasi disisipkan pada sinyal *audio*.
- Nilai SNR merupakan nilai SNR yang dikeluarkan oleh sistem sebagai respon dari *input* yang diberikan.
- Nilai PSNR merupakan nilai PSNR yang dikeluarkan oleh sistem sebagai respon dari *input* yang diberikan.

Proses dilakukan dengan menekan tombol setInfo dan proses menyimpan berkas *audio* dengan *watermark* ke dalam *local storage* dilakukan secara otomatis.



Gambar 3. 11 Antarmuka Ekstraksi

Gambar 3.13 menunjukkan antarmuka ekstraksi dengan *button* getInfo dengan penjelasan sebagai berikut:

- Input *WAudio* digunakan untuk memasukkan berkas *audio* digital yang memiliki watermark dengan ekstensi WAV yang berada dalam *local storage*.
- Input *Audio* digunakan untuk memasukkan berkas *audio* digital sebagai *audio* asli dengan ekstensi WAV yang berada dalam *local storage*.
- setAlpha digunakan untuk memasukkan nilai *alpha* yang digunakan sebagai faktor amplifikasi sinyal *audio*.
- setPattern digunakan untuk memasukkan nilai *pattern* yang digunakan sebagai pola letak penyimpanan informasi pada sinyal *audio*.
- setDimension digunakan untuk memasukkan nilai *dimension* yang digunakan sebagai jumlah informasi disisipkan pada sinyal *audio*.

Proses dieksekusi dilakukan dengan menekan tombol getInfo dan proses menyimpan berkas hasil proses ekstrak berupa gambar ke dalam *local storage* akan dilakukan secara otomatis.


UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Sistem

Bab implementasi dan pembahasan menjelaskan lingkungan sistem dalam cakupan lingkungan perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi *watermarking* sebagai berikut :

1. *Processor* AMD *Phenom*(tm) II X4 955 3.20 Ghz.
2. RAM 2 GB.
3. Monitor LCD 17".
4. *Keyboard.*
5. *Mouse.*
6. *Sound System.*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem pembuatan *watermarked audio* adalah :

1. Sistem Operasi *Microsoft Windows 7 Ultimate.*
2. Bahasa Pemograman *Java SE.*
3. IDE *Netbeans 7.1.2.*
4. *Wave Editor Adobe Audition 3.*

4.2 Implementasi Program

Implementasi dirancang berdasarkan perancangan perangkat lunak pada subbab 3.1 sampai dengan 3.3.

4.2.1 *Input Audio*

Proses pembacaan *audio* akan dilakukan dengan pola PCM. PCM dipilih karena *channel* berjumlah 2. PCM memiliki sifat *loseless* yang berarti mampu menghasilkan *audio* dengan kualitas maksimal. Proses mendapatkan pola tersebut dapat ditunjukkan dalam *source code* 4.1.

```
void read(String AudioPath) {
   myData = null;
```

```
    byte[] tmpLong = new byte[4];
    byte[] tmpInt = new byte[2];
    try {
//      read audio as carrier
     inFile = new DataInputStream(new FileInputStream( AudioPath ));
//      for debugging only
     System.out.println("Reading wav file...\n");
     String chunkID = "" + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte();
     inFile.read(tmpLong);
//      read the ChunkSize
     myChunkSize = Converter.byteArrayToLong(tmpLong);
     String format = "" + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte();
//      print what we've read so far
//      for debugging only
     System.out.println("chunkID:" + chunkID
       + " chunk1Size:" + myChunkSize
       + " format:" + format);
     String subChunk1ID = "" + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte()
       + (char) inFile.readByte();
     inFile.read(tmpLong);
//      read the SubChunk1Size
     mySubChunk1Size = Converter.byteArrayToLong(tmpLong);
//      read the audio format.
//     This should be 1 for PCM
     inFile.read(tmpInt);
     myFormat = Converter.byteArrayToInt(tmpInt);
     inFile.read(tmpInt);
//      read the # of channels (1 or 2)
     myChannels = Converter.byteArrayToInt(tmpInt);
     inFile.read(tmpLong);
//      read the samplerate
     mySampleRate = Converter.byteArrayToLong(tmpLong);
     inFile.read(tmpLong);
//      read the byterate
```

```
        myByteRate = Converter.byteArrayToLong(tmpLong);
        inFile.read(tmpInt);
//      read the blockalign
        myBlockAlign = Converter.byteArrayToInt(tmpInt);
        inFile.read(tmpInt);
//      read the bitspersample
        myBitsPerSample = Converter.byteArrayToInt(tmpInt);
//      print what we've read so far
        System.out.println("SubChunk1ID:" + subChunk1ID
            + " SubChunk1Size:" + mySubChunk1Size
            + " AudioFormat:" + myFormat
            + " Channels:" + myChannels
            + " SampleRate:" + mySampleRate);
//      read the data chunk header - reading this IS necessary,
//      because not all WAV files will have the data chunk here
//      - for now, we're just assuming that the data chunk is here
        String dataChunkID = "" + (char) inFile.readByte()
            + (char) inFile.readByte()
            + (char) inFile.readByte()
            + (char) inFile.readByte();
        inFile.read(tmpLong);
//      read the size of the data
        myDataSize = Converter.byteArrayToLong(tmpLong);
//      read the data chunk
        myData = new byte[(int) myDataSize];
        inFile.read(myData);
//      end of read audio as carrier
//      close the input stream
        inFile.close();
      } catch (Exception e) {
        Logger.getLogger(MyEngine.class.getName()).log(Level.SEVERE,
null, e);
      }
    }
```

***Source Code* 4. 1** Inisialisasi *Audio*

4.2.2 *Input* Gambar

Proses inisialisasi *watermark* dilakukan pada ruang warna YC_bC_r. Proses transformasi untuk pengubahan ruang warna RGB ke ruang warna *luminance* (Y) ditunjukkan pada *source code* 4.2.

```
static double[] grayInput(int[] RGBarray, double myAlpha) throws
```

```
IOException {
    double luminance[] = new double[RGBarray.length];
    double AlphaNormalizedWatermark [] = new double[RGBarray.length];
    int red;
    int green;
    int blue;
    double[] normalizedLuminance = new double[RGBarray.length];
    for (int i = 0; i < RGBarray.length; i++) {
      red = (RGBarray[i] & 0x00ff0000) >> 16;
      green = (RGBarray[i] & 0x0000ff00) >> 8;
      blue = (RGBarray[i] & 0x000000ff);
      luminance[i] = (0.299 * red + 0.587 * green + 0.114 * blue);
//      luminance dengan normalisasi
      normalizedLuminance[i] = (luminance[i]) / 255;
      AlphaNormalizedWatermark[i]=normalizedLuminance[i] * myAlpha;
    }
    return Alphanormalizedwatermark;
  }
```

***Source Code* 4. 2** Transformasi RGB ke YC_bC_r

4.2.3 Dekomposisi Transformasi *Wavelet*

Dekomposisi pada *audio* menghasilkan 2 *sub-band* yang mengandung 4 informasi rentang frekuensi yang berbeda yaitu C1, D1, C2, dan D2. Proses dalam dekomposisi *audio* ditunjukkan dalam *source code* 4.3.

```
static int[] makeDWT(double[] luminance, int[] channel) {
    int ctr = 0;
    int length = luminance.length;
    int pat = MyEngine.WAV.myPattern;
    int cha = (int) MyEngine.WAV.myChannels;
    double[] C1a = new double[length];
    double[] C1b = new double[length];
    double[] C2 = new double[length];
    double[] D1 = new double[length];
    double[] D2 = new double[length];
    double[] WD2 = new double[length];
    double[] WS = new double[length];
    int a;
    int b;
    for (int i = 0; i < channel.length; i++) {
```

```
    if (i % pat == 0) {
      if (ctr < channel.length) {
        if (ctr < length) {
          a = (ctr * (cha * pat));
          b = ((ctr + 1) * (cha * pat));
          C1a[ctr] = cnstnt * ((channel[a]) + channel[a + 1]);
          C1b[ctr] = cnstnt * ((channel[b]) + channel[b + 1]);
          D1[ctr] = cnstnt * ((channel[a]) - channel[a + 1]);
          C2[ctr] = cnstnt * (C1a[ctr] + C1b[ctr]);
          D2[ctr] = cnstnt * (C1a[ctr] - C1b[ctr]);
          WD2[ctr] = (D2[ctr] + luminance[ctr]);
          WS[ctr] = cnstnt * ((cnstnt * (WD2[ctr] + C2[ctr])) + D1[ctr]);
          int x = WS [ ctr ]-channel [ a ] < 0.49 * MyEngine.WAV.myAlpha
? 0 : 1;
          if (channel[a] < 255) {
            channel[a] = channel[a] + x;
          } else {
            channel[a] = channel[a] + 0;
          }
          if (x == 1) {
            penyebutSNR += Math.pow(1, 2);
          }
//          channel[a] = (int) WS[ctr];
//          channel[a] = channel[a] + x;
          System.out.println("cha "+ctr+" " + channel[a]);
         System.out.println("ws "+ctr+" " + WS[ctr]);
          ctr += 1;
        }
       }
     }
    }
    return channel;
  }
```

***Source Code* 4. 3** Dekomposisi pada *Audio*

4.2.4 Penyisipan dengan *Discrete Wavelet Transform*

Proses pembuatan dan proses pengecekan *watermarked audio* dibutuhkan perhitungan *discrete wavelet transform* yang bekerja dalam domain frekuensi. Prosedur penghitungan proses *discrete wavelet transform* ditunjukkan pada *source code* 4.4.

```
static int[] makeDWT(double[] luminance, int[] channel) {
```

```
    int ctr = 0;
    int length = luminance.length;
    int pat = MyEngine.WAV.myPattern;
    int cha = (int) MyEngine.WAV.myChannels;
    double[] C1a = new double[length];
    double[] C1b = new double[length];
    double[] C2 = new double[length];
    double[] D1 = new double[length];
    double[] D2 = new double[length];
    double[] WD2 = new double[length];
    double[] WS = new double[length];
    int a;
    int b;
    for (int i = 0; i < channel.length; i++) {
      if (i % pat == 0) {
        if (ctr < channel.length) {
          if (ctr < length) {
            a = (ctr * (cha * pat));
            b = ((ctr + 1) * (cha * pat));
            C1a[ctr] = cnstnt * ((channel[a]) + channel[a + 1]);
            C1b[ctr] = cnstnt * ((channel[b]) + channel[b + 1]);
            D1[ctr] = cnstnt * ((channel[a]) - channel[a + 1]);
            C2[ctr] = cnstnt * (C1a[ctr] + C1b[ctr]);
            D2[ctr] = cnstnt * (C1a[ctr] - C1b[ctr]);
            WD2[ctr] = (D2[ctr] + luminance[ctr]);
            WS[ctr] = cnstnt * ((cnstnt * (WD2[ctr] + C2[ctr])) + D1[ctr]);
            int x = WS[ ctr ]-channel [ a ] < 0.49 * MyEngine.WAV.myAlpha
? 0 : 1;
            if (channel[a] < 255) {
              channel[a] = channel[a] + x;
            } else {
              channel[a] = channel[a] + 0;
            }
            if (x == 1) {
              penyebutSNR += Math.pow(1, 2);
            }
//            channel[a] = (int) WS[ctr];
//            channel[a] = channel[a] + x;
            System.out.println("cha "+ctr+" " + channel[a]);

          System.out.println("ws "+ctr+" " + WS[ctr]);
            ctr += 1;
```

```
        }
      }
    }
  }
  return channel;
}
```

***Source Code* 4. 4** Perhitungan DWT

4.2.5 Ekstraksi dengan *Discrete Wavelet Transform*

Proses ekstraksi dibutuhkan perhitungan *discrete wavelet transform* yang bekerja dalam domain frekuensi pada dua *file audio*. Prosedur penghitungan proses *discrete wavelet transform* ditunjukkan pada *source code* 4.5.

```
static double[] getDWT(int[] channel, int[] channel1) {
  int length = MyEngine.WAV1.myDimension * MyEngine.WAV1.
myDimension ;
  int pat1 = MyEngine.WAV1.myPattern;
  int pat2 = MyEngine.WAV2.myPattern;
  int cha1 = (int) MyEngine.WAV1.myChannels;
  int cha2 = (int) MyEngine.WAV2.myChannels;
  double[] wi = new double[length];
  int ctr = 0;
  double[] C1a = new double[length];
  double[] C1b = new double[length];
  double[] C2 = new double[length];
  double[] D1 = new double[length];
  double[] D2 = new double[length];
  double[] C1aP = new double[length];
  double[] C1bP = new double[length];
  double[] C2P = new double[length];
  double[] D1P = new double[length];
  double[] D2P = new double[length];
  int a;
  int b;
  for (int i = 0; i < channel.length; i++) {
    if (i % pat1 == 0) {
      if (ctr < channel.length) {
        if (ctr < length) {
          a = (ctr * (cha1 * pat1));
          b = ((ctr + 1) * (cha1 * pat1));
```

```
            C1a[ctr] = cnstnt * ((channel[a]) + channel[a + 1]);
            C1b[ctr] = cnstnt * ((channel[b]) + channel[b + 1]);
            D1[ctr] = cnstnt * ((channel[a]) - channel[a + 1]);
            C2[ctr] = cnstnt * (C1a[ctr] + C1b[ctr]);
            D2[ctr] = cnstnt * (C1a[ctr] - C1b[ctr]);
            a = (ctr * (cha2 * pat2));
            b = ((ctr + 1) * (cha2 * pat2));
            C1aP[ctr] = cnstnt * ((channel1[a]) + channel1[a + 1]);
            C1bP[ctr] = cnstnt * ((channel1[b]) + channel1[b + 1]);
            D1P[ctr] = cnstnt * ((channel1[a]) - channel1[a + 1]);
            C2P[ctr] = cnstnt * (C1aP[ctr] + C1bP[ctr]);
            D2P[ctr] = cnstnt * (C1aP[ctr] - C1bP[ctr]);
            double wti = D2[ctr] - D2P[ctr];
          System.out.println("cha "+ctr+" " + channel[a]);
          System.out.println("cha1 "+ctr+" " + channel1[a]);
          System.out.println("d2 "+ctr+" " + D2[ctr]);
          System.out.println("d2p "+ctr+" " + D2P[ctr]);
//          double wni = wti < 0.49 * MyEngine.WAV1.myAlpha ? 0 : 1;
//          wi[ctr] = wni;// * 255;
          wi[ctr] = wti;
          ctr += 1;
         }
        }
       }
      }
     return wi;
    }
```

***Source Code* 4. 5** Perhitungan Ekstraksi dengan DWT

4.2.6 Perhitungan SNR dan PSNR

Perhitungan nilai kualitas *audio* digital yang merupakan hasil *watermarking* terhadap *audio* digital asli dilakukan dengan menggunakan nilai *Signal-to-Noise Ratio* (SNR) dan nilai *Peak Signal-to-Noise Ratio* (PSNR). Perhitungan nilai SNR dari *audio* digital berukuran NxM dilakukan sesuai persamaan 2.16. Nilai PSNR didapat dengan menghitung perkalian logaritma SNR yang dimiliki suatu *audio* dengan 10. Prosedur penghitungan SNR dan PSNR ditunjukkan oleh *source code* 4.6 dan 4.7.

```
int x = WS[ctr] - channel[a] < 0.49 * MyEngine.WAV.myAlpha ? 0 : 1;
          if (channel[a] < 255) {
            channel[a] = channel[a] + x;
          } else {
            channel[a] = channel[a] + 0;
          }
          if (x == 1) {
            penyebutSNR += Math.pow(1, 2);
          }
byte[] b = new byte[2];
      for (int i = 0; i < WAV.myData.length; i++) {
//        System.arraycopy(WAV.myData, i, a, 0, 1);
        System.arraycopy(watermarkedMyData, i, b, 0, 1);
        pembilangSNR += Math.pow(Converter.byteArrayToInt(b), 2);
//                penyebutSNR  +=  Math.pow(Converter.byteArrayToInt(a)  -
Converter.byteArrayToInt(b), 2);
      }
      SNR = pembilangSNR / DwtEngine.penyebutSNR;
```

***Source Code* 4. 6** Perhitungan SNR

```
double getPSNR() {
    double PSNR;
    PSNR = 10.0 * Math.log10(SNR);
    return PSNR;
  }
```

***Source Code* 4. 7** Perhitungan PSNR

4.2.7 Penyimpanan *Audio* yang Disisipi *Watermark*

Audio yang disisipi *watermark* disimpan dalam format WAV. Prosedur penyimpanan *audio* yang disisipi *watermark* ditunjukkan pada *source code* 4.8.

```
boolean saveAudio() {
    try {
      DataOutputStream outFile = new DataOutputStream(new FileOutput
Stream( "outOf" + myAudioPath));
//      write the WAV file per the WAV file format
//      00 - RIFF
      outFile.writeBytes("RIFF");
//      04 - how big is the rest of this file?
      outFile.write(Converter.intToByteArray((int) WAV.myChunkSize), 0,
```

```
4);
//      08 - WAVE
    outFile.writeBytes("WAVE");
//      12 - fmt
    outFile.writeBytes("fmt ");
//      16 - size of this chunk
    outFile.write(Converter.intToByteArray((int)
WAV.mySubChunk1Size), 0, 4);
//      20 - what is the audio format? 1 for PCM = Pulse Code Modulation
    outFile.write(Converter.shortToByteArray((short)    WAV.myFormat),
0, 2);
//      22 - mono or stereo? 1 or 2? (or 5 or ???)
    outFile.write(Converter.shortToByteArray((short) WAV.myChannels),
0, 2);
//      24 - samples per second (numbers per second)
    outFile.write(Converter.intToByteArray((int)    WAV.mySampleRate),
0, 4);
//      28 - bytes per second
    outFile.write(Converter.intToByteArray((int)   WAV.myByteRate),   0,
4);
//      32 - # of bytes in one sample, for all channels
    outFile.write(Converter.shortToByteArray((short)
WAV.myBlockAlign), 0, 2);
//      34 - how many bits in a sample(number)? usually 16 or 24
    outFile.write(Converter.shortToByteArray((short)
WAV.myBitsPerSample), 0, 2);
//      36 - data
    outFile.writeBytes("data");
//      40 - how big is this data chunk
    outFile.write(Converter.intToByteArray((int)   WAV.myDataSize),   0,
4);
//      44 - the actual data itself - just a long string of numbers
    outFile.write(watermarkedMyData);
   } catch (Exception e) {
    System.out.println(e.getMessage());
    return false;
   }
   return true;
  }
```

***Source Code* 4. 8** Penyimpanan *Audio* dengan *Watermark*

4.2.8 Penyimpanan Gambar hasil Ekstraksi

Gambar yang diperoleh dari hasil ekstraksi disimpan dalam format bmp. Prosedur penyimpanan gambar ditunjukkan pada *source code* 4.8.

```
static void rgbOutput(String tmpPath, double[] rgb, double myAlpha) {
    try {
      int a = (int) Math.sqrt(rgb.length);
      BufferedImage buffer=new BufferedImage(a,a,BufferedImage.TYPE
_BYTE_GRAY);
      int x = 0;
      for (int i = 0; i < a; i++) {
        for (int j = 0; j < a; j++) {
          double rgb_a = rgb[x] / myAlpha;
//          double rgb_255 = rgb_a * 255;
//          buffer.setRGB(i, j, (int) -rgb_255);
          System.out.println("rgb " +x+" "+ rgb_a);
          buffer.setRGB(i, j, (int) -rgb_a);
          x++;
        }
      }
//      to save image
      File imageFile = new File(tmpPath);
      System.out.println("SELESAI");
      ImageIO.write(buffer, "bmp", imageFile);
    }catch(IOException
ex){Logger.getLogger(BmpEngine.class.getName()).log(Level.SEVERE,
null, ex ) ;
    }
  }
```

***Source Code* 4. 9** Penyimpanan Gambar *Watermark*

4.3 Implementasi Antarmuka

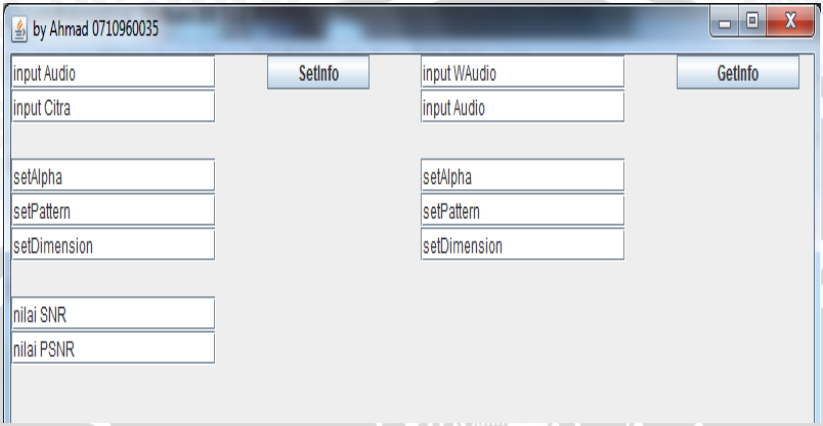
Antarmuka dibuat berdasarkan rancangan antarmuka pada subbab 3.6.

4.3.1 Tampilan Utama Aplikasi *Audio Watermarking*

Tampilan utama implementasi program *Audio Watermarking* ditunjukkan pada gambar 4.1.

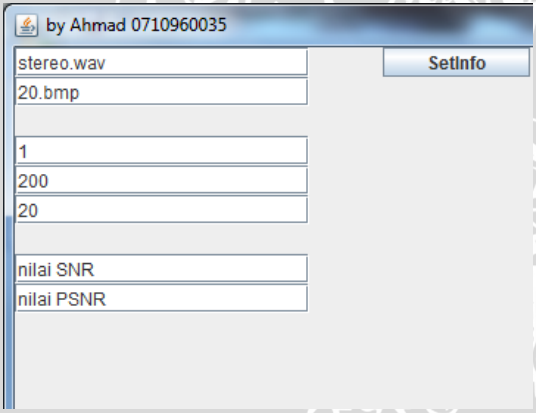
Tampilan utama memiliki 2 *button* untuk proses penyisipan dan ekstraksi *watermark*. tampilan program juga memiliki *text field* untuk menerima parameter dari kedua proses antara lain *input Audio*, *input* Citra, *Input* WAudio, setAlpha, setPattern, dan setDimension.

Selain itu, tampilan utama memiliki *text field* nilai SNR dan PSNR sebagai respon terhadap parameter yang diberikan.



Gambar 4. 1 Tampilan Utama

4.3.2 Tampilan Proses Penyisipan *Watermark*

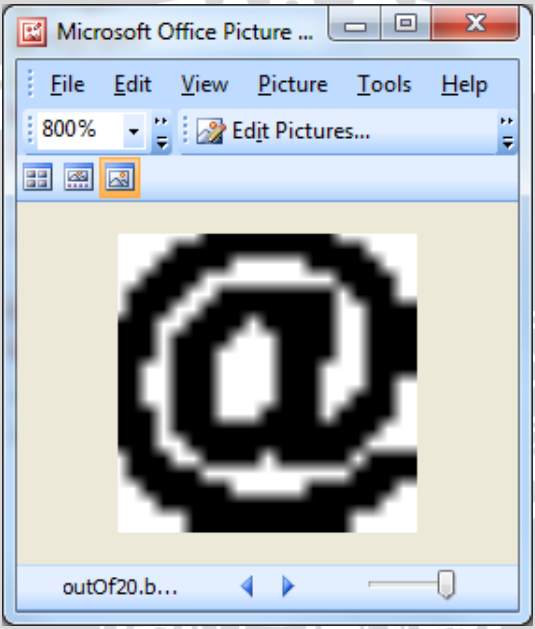


Gambar 4. 2 Tampilan Pemberian Parameter Proses Penyisipan

Langkah pertama proses penyisipan *watermark* adalah proses memasukkan parameter yang berhubungan dengan proses penyisipan antara lain *input Audio* untuk menujukkan letak *file audio* pada *local storage*, *input* Citra untuk menujukkan letak *file* gambar *watermark* pada *local storage*, setAlpha untuk menentukan nilai amplifikasi

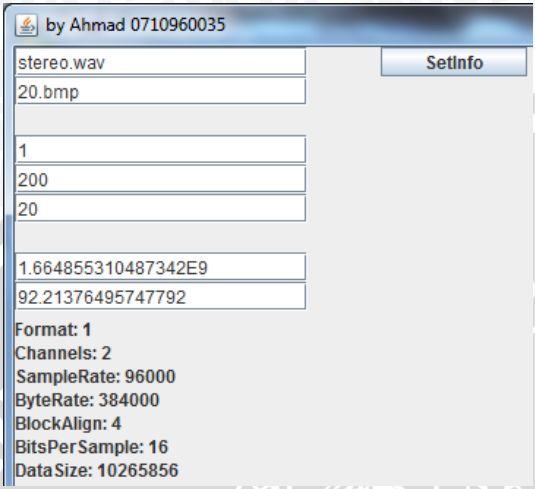
yang diinginkan dari sinyal *audio*, setPattern untuk menentukan pola letak penyisipan informasi *watermark* pada sinyal *audio*, dan setDimesion untuk menentukan jumlah informasi yang ingin disisipkan pada sinyal *audio* yang ditunjukkan pada gambar 4.2.

Hasil eksekusi dari proses penyisipan adalah nilai SNR, PSNR, dan *audio* yang disisipi *watermark*. *Text field* nilai SNR dan PSNR digunakan untuk menampilkan nilai SNR dan PSNR tersebut.



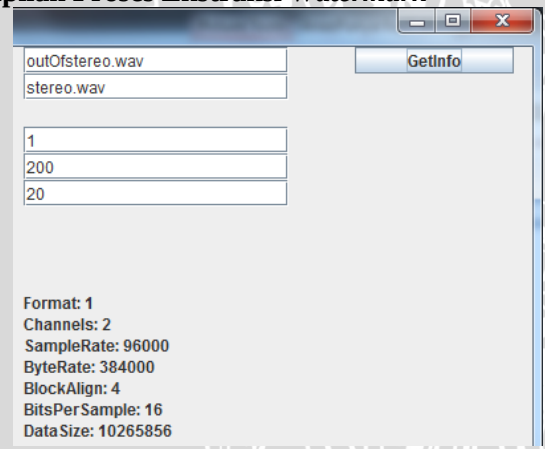
Gambar 4. 3 Contoh Tanda *Watermark*

Proses dilanjutkan dengan menekan *button* setInfo sehingga sistem akan menerima parameter-parameter dan mengeluarkan nilai PSNR sebagai respon dari parameter tersebut yang ditunjukkan pada gambar 4.4.



Gambar 4. 4 Tampilan Hasil Proses Penyisipan *Watermark*

4.3.3 Tampilan Proses Ekstraksi *Watermark*



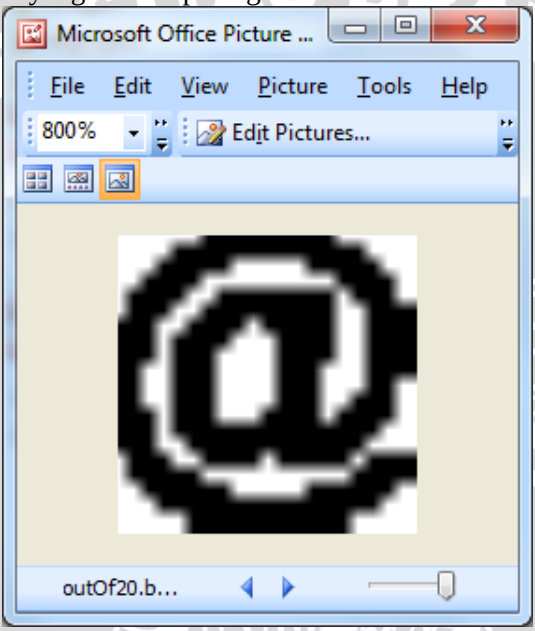
Gambar 4. 5 Tampilan Proses Ekstraksi

Proses ekstraksi *watermark* diawali dengan proses memasukkan parameter yang berhubungan dengan proses ekstraksi antara lain *input* WAudio untuk menujukkan letak *file audio* yang diberi *watermark* pada *local storage*, *input Audio* untuk menujukkan letak *file audio* asli sebagai pembanding pada *local storage*, setAlpha untuk menentukan nilai amplifikasi informasi dari sinyal *audio*,

setPattern untuk menentukan pola letak informasi *watermark* pada sinyal *audio*, dan setDimesion untuk menentukan jumlah informasi pada sinyal *audio* yang ditunjukkan pada gambar 4.5.

Proses dilanjutkan dengan menekan *button* getInfo sehingga sistem akan menerima parameter-parameter dan mengeluarkan *file* gambar sebagai respon dari parameter tersebut yang ditunjukkan pada gambar 4.6.

Hasil tanda *watermark* yang disisipkan dapat dilihat *image viewer* seperti yang terlihat pada gambar 4.6.



Gambar 4. 6 Tampilan Watermark Hasil Ekstraksi

4.4 Implementasi Uji Coba

Subbab implementasi uji coba digunakan sebagai pembahasan mengenai pengujian yang dilakukan sistem dan hasil evaluasi yang dikeluarkan sistem.

4.4.1 Evaluasi Kualitas *Audio*

Kualitas *audio* dihitung dengan SNR untuk melihat pengaruh pemberian *watermark*. Penyisipan *watermark* dilakukan terhadap *audio* bernama stereo.wav dengan durasi 26 detik dalam format 16

bit. Setiap *audio* ditandai *watermark* dengan menggunakan *alpha* (α) = 1 ; 5 ; 10 dan *pattern* penyisipan per 200 pada *left channel*.

Watermark yang digunakan sebagai tanda *watermark* adalah 20.bmp dan 50.bmp dengan ukuran *pixel* masing-masing adalah 20x20 dan 50x50 *pixel*.

Data penyisipan *watermark* yang dilakukan pada *audio* yang sama untuk *alpha* (α) = 1 ditunjukkan pada tabel 4.1.

Tabel 4. 1 Hasil Penyisipan *Watermark* untuk *Alpha* (α) = 1

Watermarked Audio	*Watermark*	Pola	SNR	PSNR
outOfstereo.wav	20.bmp	Per 20	$1{,}66x10^{9}$	92,2137
outOfstereo.wav	50.bmp	Per 20	$1{,}95x10^{8}$	82,9195
outOfstereo.wav	20.bmp	Per 50	$1{,}66x10^{9}$	92,2137
outOfstereo.wav	50.bmp	Per 50	$1{,}95x10^{8}$	82,9195
outOfstereo.wav	20.bmp	Per 100	$1{,}66x10^{9}$	92,2137
outOfstereo.wav	50.bmp	Per 100	$1{,}95x10^{8}$	82,9195
outOfstereo.wav	20.bmp	Per 200	$1{,}66x10^{9}$	92,2137
outOfstereo.wav	50.bmp	Per 200	$1{,}95x10^{8}$	82,9195

Data penyisipan *watermark* yang dilakukan pada *audio* yang sama untuk *alpha* (α) = 5 ditunjukkan pada tabel 4.2.

Tabel 4. 2 Hasil Penyisipan *Watermark* untuk *Alpha* (α) = 5

Watermarked Audio	*Watermark*	Pola	SNR	PSNR
outOfstereo.wav	20.bmp	Per 20	$8{,}32x10^{8}$	89,2034
outOfstereo.wav	50.bmp	Per 20	$9{,}79x10^{7}$	79,9092
outOfstereo.wav	20.bmp	Per 50	$8{,}32x10^{8}$	89,2034
outOfstereo.wav	50.bmp	Per 50	$9{,}79x10^{7}$	79,9092
outOfstereo.wav	20.bmp	Per 100	$8{,}32x10^{8}$	89,2034
outOfstereo.wav	50.bmp	Per 100	$9{,}79x10^{7}$	79,9092
outOfstereo.wav	20.bmp	Per 200	$8{,}32x10^{8}$	89,2034
outOfstereo.wav	50.bmp	Per 200	$9{,}79x10^{7}$	79,9092

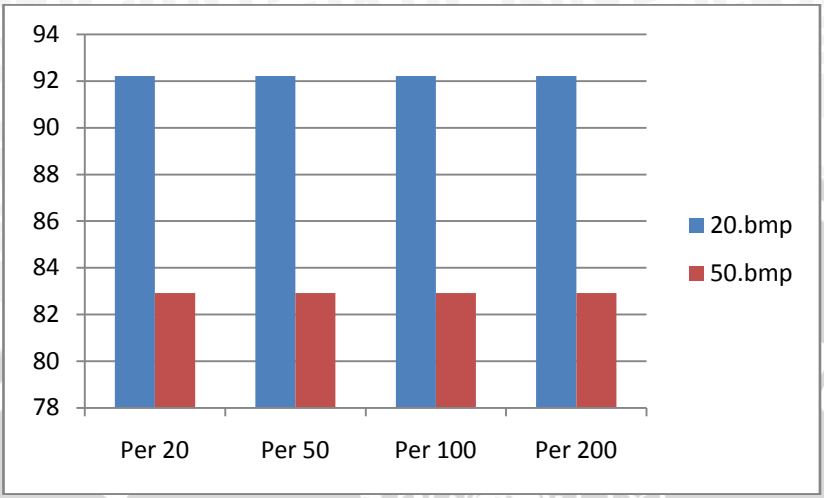
Data penyisipan *watermark* yang dilakukan pada *audio* yang sama untuk *alpha* (α) = 10 ditunjukkan pada tabel 4.3

Tabel 4. 3. Hasil Penyisipan *Watermark* untuk *Alpha* (α) = 10

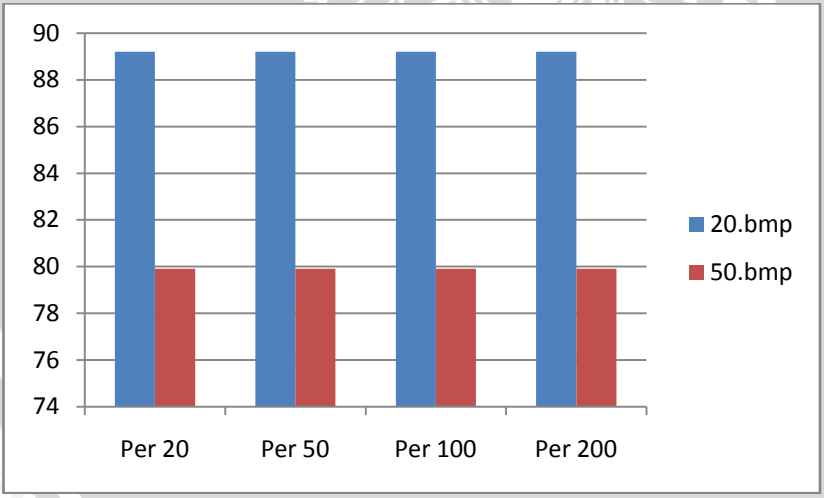
Watermarked Audio	*Watermark*	Pola	SNR	PSNR
outOfstereo.wav	20.bmp	Per 20	$5{,}54x10^{8}$	87,4425
outOfstereo.wav	50.bmp	Per 20	$6{,}52x10^{7}$	78,1483
outOfstereo.wav	20.bmp	Per 50	$5{,}54x10^{8}$	87,4425
outOfstereo.wav	50.bmp	Per 50	$6{,}52x10^{7}$	78,1483
outOfstereo.wav	20.bmp	Per 100	$5{,}54x10^{8}$	87,4425
outOfstereo.wav	50.bmp	Per 100	$6{,}52x10^{7}$	78,1483
outOfstereo.wav	20.bmp	Per 200	$5{,}54x10^{8}$	87,4425
outOfstereo.wav	50.bmp	Per 200	$6{,}52x10^{7}$	78,1483

Nilai PSNR diperoleh dengan menghitung nilai SNR. Hasil penyisipan *watermark* menunjukkan bahwa nilai SNR berbanding terbalik dengan nilai PSNR. Semakin kecil nilai SNR yang dihasilkan maka semakin besar nilai PSNR sehingga semakin baik kualitas *audio* yang disisipi *watermark*.

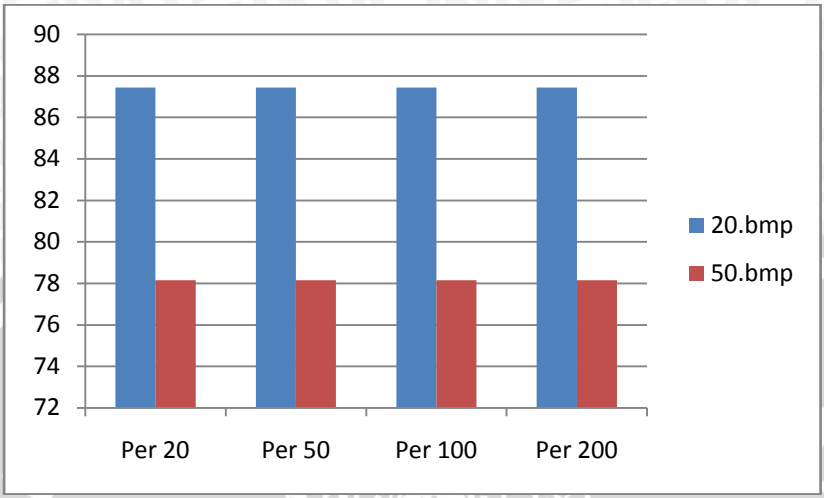
Hasil penyisipan *watermark* pada tabel 4.1 menunjukkan bahwa dengan menggunakan *alpha* yang kecil yaitu 1 maka SNR yang dihasilkan akan kecil dan memiliki nilai PSNR yang besar. Peningkatan *alpha* menjadi 5 dan 10 maka SNR yang dihasilkan mengalami kecenderungan semakin besar dan memiliki nilai PSNR yang semakin kecil. Hal tersebut ditunjukkan dengan rata-rata SNR untuk *alpha* 1 sebesar $9{,}27x10^{9}$ sedangkan rata-rata SNR dengan nilai *alpha* 5 dan 10 adalah sebesar $4{,}64x10^{8}$ dan $3{,}09x10^{8}$. Rata-rata PSNR dengan nilai *alpha* 1 sebesar 87,5666 sedangkan rata-rata PSNR dengan nilai *alpha* 5 dan 10 adalah 84,5563 dan 82,7954. Perhitungan tersebut menunjukkan bahwa nilai SNR dan PSNR yang baik dihasilkan oleh *audio* yang disisipi *watermark* dengan nilai *alpha* yang kecil. Grafik pengaruh *alpha* untuk nilai PSNR terhadap *pattern* yang diberikan pada masing-masing ukuran tanda *watemark* ditunjukkan pada gambar 4.7 sampai dengan 4.9.



Gambar 4. 7 Grafik Pengaruh Ukuran *Watermark* dan Pola terhadap Nilai PSNR untuk *Alpha* Sebesar 1



Gambar 4. 8 Grafik Pengaruh Ukuran *Watermark* dan Pola terhadap Nilai PSNR untuk *Alpha* Sebesar 5



Gambar 4. 9 Grafik Pengaruh Ukuran *Watermark* dan Pola terhadap Nilai PSNR untuk *Alpha* Sebesar 10

Gambar grafik 4.7 sampai dengan 4.9 menunjukkan bahwa nilai PSNR yang dihasilkan akan semakin menurun ketika ukuran *watermark* dinaikkan untuk semua pola yang digunakan. Nilai PSNR tertinggi dihasilkan oleh *audio* yang disisipi *watermark* dengan ukuran *watermark* yang digunakan adalah 20x20 *pixel* yaitu gambar 20.bmp dengan nilai SNR $1{,}66 \times 10^{9}$ dan nilai PSNR 92.2137.

4.4.2 Evaluasi Hasil *Audio* yang Disisipi *Watermark*

Evaluasi hasil *audio* yang disisipi *watermark* mengacu pada subbab 2.7.4 mengenai kualitas *watermarked audio*. Sebuah *audio* yang disisipi harus memiliki persyaratan ketahanan (*robustness*) dan tidak terlihat (*nonpercepbility*) sehingga diperlukan suatu evaluasi hasil terhadap hasil *audio* yang disisipi *watermark*.

4.4.2.1 Hasil Evaluasi Tanpa Serangan

Uji coba pertama yang dilakukan terhadap 2 *watermarked audio* dengan menggunakan ukuran gambar *watermark* 20x20 dan 50x50 *pixel* dan menggunakan α (1;5;10) tanpa serangan pemrosesan *audio*. Proses tersebut merupakan proses penyisipan *watermark* pada *audio* asli dan langsung dilakukan proses ekstraksi *watermark* untuk

mengetahui apakah tanda *watermark* yang disisipkan dapat terdeteksi atau tidak.

Hasil pengujian tanpa serangan yang dilakukan menunjukkan bahwa semua *audio* yang disisipi *watermark* untuk semua nilai *alpha*, semua nilai *pattern*, dan untuk semua ukuran tanda *watermark* mampu terdeteksi dengan baik.

4.4.2.2 Evaluasi Hasil Robustness *Watermarked Audio*

Evaluasi ketahanan *watermarked audio* dilakukan dengan memberikan beberapa serangan terhadap *audio* tersebut. Serangan tersebut antara lain *noise reduction effect* pada *audio* beresolusi 32 *bit* tanpa *DC offset*, pengubahan kedalaman *bit* dari sebuah *file* dengan proses *dithering* (penentuan *dither depth* dalam *bit* dengan nilai antara 0,2 sampai dengan 0,7), dan pemberian fungsi distribusi kemungkinan (*probability distribution function*) menggunakan jenis *triangular*.

4.4.2.2.1 *Noise Reduction Effect*

Percobaan dengan mereduksi signal *audio* atau *noise reduction effect* dengan mereduksi bagian *watermarked audio* sebesar 7 %, dan 15% dari ukuran semula. Hasil pengujian serangan *audio* yang disisipi *watermark* dengan *noise reduction effect* ditunjukkan dengan tabel 4.4.

Tabel 4. 4 Hasil Pendeteksian *Watermark* dengan Serangan *Noise Reduction Effect*

20.bmp		7%	14%
Alpha	1	√	√
	5	√	√
	10	√	√
50.bmp		7%	14%
Alpha	1	√	√
	5	√	√
	10	√	√

Keterangan :
√ = *watermark* terdeteksi
X = *watermark* tidak terdeteksi

Tabel 4.4 menunjukkan uji coba yang dilakukan terhadap *audio* yang disisipi *watermark* dengan serangan *noise reduction effect* untuk penggunaan semua nilai *alpha* memungkinkan *watermark* yang disisipkan masih dapat dideteksi meskipun mengalami reduksi sebesar 14% dari ukuran semula.

4.4.2.2.2 Penentuan *Dither Depth*

Penentuan *dither depth* secara umum menggunakan nilai berkisra antara 0,2 sampai dengan 0,7 yang memberikan hasil terbaik tanpa menambahkan terlalu banyak *noise*. Selain nilai standar akan mengakibatkan suara distorsi harmonik yang tidak diinginkan muncul. Hasil pengujian serangan *audio* yang disisipi *watermark* dengan penentuan *dither depth* ditunjukkan dengan tabel 4.5.

Tabel 4. 5 Hasil Pendeteksian *Watermark* dengan Serangan *Dither Depth Scaling*

20.bmp		0,1	0,2
Alpha	1	√	√
	5	√	√
	10	√	√
50.bmp		0,3	0,6
Alpha	1	√	√
	5	√	√
	10	√	√

Keterangan :
√ = *watermark* terdeteksi
X = *watermark* tidak terdeteksi

Tabel 4.5 menunjukkan uji coba yang dilakukan terhadap *audio* yang disisipi *watermark* dengan serangan penentuan *dither depth* untuk penggunaan semua nilai *alpha* memungkinkan *watermark* yang disisipkan masih dapat dideteksi meskipun mengalami penentuan *dither depth* sebesar 0,2 dari *audio*.

4.4.2.2.3 *Probability Distribution Function* (PDF)

Probability distribution fuction (PDF) digunakan untuk mengontrol *dithered noise* yang didistribusikan pada nilai *sample audio*. *Triangular* PDF merupakan pilihan yang umum untuk

memberikan nilai distribusi SNR, distorsi, dan modulasi *noise*. *Triangular* PDF memilih nilai acak yang mendekati 0 daripada -1 atau 1 (hal tersebut menunjukkan kemungkinan 0 dua kali lebih besar daripada -0,5 atau 0,5). PDF digunakan bersamaan dengan konversi *audio* berbasis kualitas berskala 30 sampai dengan 999. Hasil pengujian serangan *audio* yang disisipi *watermark* dengan *probability distribution fuction* ditunjukkan dengan tabel 4.6.

Tabel 4. 6. Hasil Pendeteksian *Watermark* dengan *Probability Distribution Fuction*

20.bmp		100	400
Alpha	1	√	√
	5	√	√
	10	√	√
50.bmp		100	400
Alpha	1	√	√
	5	√	√
	10	√	√

Keterangan :
√ = *watermark* terdeteksi
X = *watermark* tidak terdeteksi

Tabel 4.6 menunjukkan uji coba yang dilakukan terhadap *audio* yang disisipi *watermark* dengan *probability distribution fuction* untuk penggunaan semua nilai *alpha* memungkinkan *watermark* yang disisipkan masih dapat dideteksi meskipun mengalami *probability distribution fuction* sebesar 400 dari *audio*.

4.4.2.2.4 Evaluasi Hasil Tidak Terlihat (*Nonpercepbility*)

Salah satu persyaratan yang harus dimiliki oleh *audio watermarking* adalah *nonpercepbility* dengan kondisi bahwa *audio* asli dan *audio* yang disisipi oleh *watermark* tidak bisa dibedakan oleh pendengaran manusia normal. Evaluasi *nonpercepbility* dilakukan tes terhadap 10 orang usia 20-30 tahun. Hasil tes terhadap *audio* yang disisipi *watermark* menunjukkan bahwa semua responden melihat tidak adanya perbedaan antara *audio* asli dan *audio* pembawa *watermark* untuk semua *alpha* pada tanda *watermark* dengan ukuran 20x20 dan 50x50 *pixel*. Tabel hasil *audio*

yang disisipi *watermark* tidak terlihat (*nonpercepbility*) ditunjukkan pada tabel 4.7.

Tabel 4. 7 Hasil *Nonpercepbility* untuk *Alpha* (α) = 1

Nama	*Audio*	*Pattern*	*Watermark*	Hasil
Afif	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Sandro	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Saiful	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Arief	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√

Yudi	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Ade	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Madya	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Agung	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Ivan	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√

	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√
Soni	outOfstereo.wav	20	20.bmp	√
	outOfstereo.wav	20	50.bmp	√
	outOfstereo.wav	50	20.bmp	√
	outOfstereo.wav	50	50.bmp	√
	outOfstereo.wav	100	20.bmp	√
	outOfstereo.wav	100	50.bmp	√
	outOfstereo.wav	200	20.bmp	√
	outOfstereo.wav	200	50.bmp	√

Keterangan :

√ = tidak terdengar perbedaan

X = terdengar perbedaan

4.5 Analisa Hasil

Pengujian dilakukan terhadap *audio* dengan *watermark* yang berbeda dengan memberikan *alpha* sebesar 1, 5, dan 10. Kualitas *audio* dihitung dengan menggunakan SNR dan PSNR. Nilai SNR dan PSNR untuk masing-masing *audio* yang disisipi *watermark* dapat dilihat pada subbab 4.4.1.

Hasil penyisipan tanda *watermark* kedua *audio* didapatkan hasil dengan menggunakan *alpha* yang kecil yaitu nilai *alpha* 1 akan dihasilkan nilai SNR yang kecil dan nilai PSNR yang besar. Penambahan *alpha* mengakibatkan nilai SNR yang dihasilkan akan semakin besar dan nilai PSNR yang dihasilkan akan semakin kecil. Penambahan nilai SNR dan pengurangan nilai PSNR ditunjukkan dengan rata-rata SNR untuk *alpha* 1 sebesar $9{,}27 \times 10^{9}$ sedangkan rata-rata SNR dengan nilai *alpha* 5 dan 10 adalah sebesar $4{,}64 \times 10^{8}$ dan $3{,}09 \times 10^{8}$. Rata-rata PSNR dengan nilai *alpha* 1 sebesar 87,5666 sedangkan rata-rata PSNR dengan nilai *alpha* 5 dan 10 adalah 84,5563 dan 82,7954. Hal tersebut dapat diartikan bahwa proses mendapatkan nilai *error* dari *audio* yang disisipi *watermark* yang kecil dan menghasilkan kualitas *audio* yang bagus diperlukan penggunaan nilai *alpha* yang kecil. Hal tersebut terjadi karena besaran nilai *alpha* yang digunakan saat penyisipan *watermark* ke dalam *byte audio*. *Alpha* yang semakin besar mengakibatkan nilai *watermark* yang disisipkan menjadi semakin besar sehingga

memperbesar nilai *byte audio* setelah ditambahkan. Nilai *byte audio* yang terlalu besar akan mengurangi kualitas *audio* yang disisipi *watermark*.

Pengujian penambahan ukuran tanda *watermark* mulai dari 20x20 *pixel* kemudian ditingkatkan menjadi 50x50 *pixel* menunjukkan hasil yang didapatkan yaitu *audio* yang disisipi tanda *watermark* dengan ukuran 20x20 *pixel* memiliki nilai SNR yang kecil dan nilai PSNR yang besar sehingga dapat disimpulkan memiliki kualitas *audio* yang bagus. *Audio* yang disisipi tanda *watermark* dengan ukuran 50x50 *pixel* pada semua nilai *alpha* menunjukkan SNR yang dihasilkan cukup besar yaitu mulai dari $6{,}52x10^{7}$ sampai dengan $1{,}95x10^{8}$ dan nilai PSNR yang dihasilkan berkisar antara 82 ke bawah yang berarti kualitas *watermark* yang dihasilkan kurang baik dibandingkan dengan penyisipan tanda *watermark* ukuran 20x20 *pixel*. Hal tersebut disebabkan karena semakin besar ukuran tanda *watermark* yang digunakan maka semakin memperbesar jumlah *byte audio* yang disisipi *watermark* sehingga mengurangi kualitas *audio* yang disisipi *watermark*.

Percobaan evaluasi terhadap *audio* yang disisipi *watermark* dilakukan pada subbab 4.4.2. Evaluasi terhadap *audio* yang disisipi *watermark* terdiri dari uji coba *audio* yang disisipi *watermark* tanpa serangan dengan hasil uji coba dapat lihat pada subbab 4.4.2.1. Pengujian terhadap audio yang disisipi *watermark* tanpa serangan menunjukkan tanda *watermark* yang disisipkan dapat dideteksi dengan baik.

Pengujian dilanjutkan dengan menggunakan serangan terhadap *audio* yang disisipi *watermark* meliputi *noise reduction effect*, penentuan *dither depth*, dan *probability distribution function* menggunakan jenis *triangular*. Hasil pengujian ketiga serangan terhadap *audio* tersebut dapat dilihat pada subbab 4.4.2.1 sampai dengan subbab 4.4.2.3. Hasil pengujian menunjukkan tanda *watermark* yang disisipkan ke dalam *audio* yang disisipi *watermark* mampu dideteksi dengan baik walaupun diberikan serangan. Tanda *watermark* yang disisipkan pada domain frekuensi dengan metode *discrete wavelet transform* dapat bertahan dari beberapa macam serangan. Hal tersebut dikarenakan penggunaan nilai *alpha* sebagai penguat *watermark* dan pembuatan *audio* dengan *watermark* dalam

domain frekuensi membuat *attacker* sulit untuk melepaskan *watermark* tersebut.

Pengujian terakhir *audio* yang dilakukan dengan mengujikan *audio* yang disisipi *watermark* dari sisi *nonpercepbility*. Pengujian melibatkan 10 orang dari berbagai usia. Hasil penelitian menunjukkan bahwa semua responden menyatakan tidak mendengar pebedaan antara *audio* asli terhadap *audio* yang disisipi *watermark*. Hal tersebut menunjukkan bahwa *audio* yang disisipi *watermark* memenuhi syarat tidak terlihat (*nonperceptibility*) untuk ukuran *watermark* 20x20 *pixel* dan 50x50 *pixel*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Hasil uji coba yang dilakukan menunjukkan kesimpulan sebagai berikut:

1. Kualitas *audio* dengan pemberian *alpha* 1 terlihat lebih baik dibandingkan *audio* dengan pemberian *alpha* 5 dan 10. Hal tersebut dapat dilihat dari nilai rata-rata *Signal Noise-to-Ratio* **(**SNR) dan rata-rata nilai *Peak Signal Noise-to-Ratio* **(**PSNR) yang dihasilkan. Pemberian *alpha* 1 menghasilkan nilai rata-rata SNR terbesar sebesar $9{,}27 \times 10^{9}$ dan nilai PSNR terbesar sebesar 87,5666.
2. Kualitas *audio* yang dihasilkan dengan pemberian tanda *watermark* dengan ukuran 20x20 *pixel* lebih bagus dibandingkan dengan pemberian tanda *watermark* dengan ukuran 50x50 *pixel*. Hal tersebut ditunjukkan dengan nilai SNR dan PSNR yang dihasilkan.
3. Tanda *watermark* yang disisipkan berhasil dideteksi dengan baik setelah dilakukan serangan pemrosesan *audio* yaitu *noise reduction effect*, penentuan *dither depth*, dan *probability distribution function* menggunakan jenis *triangular*.
4. Hasil *audio* yang disisipi *watermark* yang diujikan kepada responden menunjukkan bahwa seluruh responden tidak mendengar pebedaan antara *audio* asli ataupun *audio* yang disisipi *watermark* untuk semua *alpha* pada tanda *watermark* dengan ukuran maksimal 50x50 *pixel*. Hal tersebut menunjukkan bahwa *audio* yang disisipi *watermark* memenuhi syarat tidak terdengar (*nonperceptibility*) untuk ukuran *watermark* 50x50 *pixel* dan nilai *pixel* yang lebih kecil dari 50x50 *pixel*.

5.2 Saran

Beberapa saran untuk pengembangan penelitian lebih lanjut adalah:

1. Penelitian dapat dikembangkan dengan menggunakan nilai *pattern* yang lebih besar dan ukuran *audio* yang lebih besar.

2. Penelitian dapat dikembangkan dengan menambahkan *attack* terhadap *audio* yang disisipi *watermark*.
3. Penelitian dapat dikembangkan untuk dokumen lain yang dapat diberikan *watermark* seperti gambar, *video*, ataupun *formatted text*.

DAFTAR PUSTAKA

ADOBE® Audition. 2009. http://www.adobe.com/products/audition/

Al-Haj Ali, Mohammad Ahmad, and Bata Lama . 2011.*DWT–Based Audio Watermarking*. The International Arab Journal of Information Technology, Vol. 8, No. 3.

Arnold M., Wolthusen S., and Schmucker M. 2003. *Techniques and Applications of Digital Watermarking and Content Protection.Artech House, Psychoacoustics: Facts and models*. Springer-Verlag.

Brown, Antony. 2006. *The Focal Easy Guide to Adobe® Audition™ 2.0*. Focal Press. Elsevier Ltd. United Kingdom.

Fabien A. P. Petitcolas, Ross J. Anderson, Markus G. Kuhn. 1998. *Attacks On Copyright Marking Systems*, in David Aucsmith (Ed), Information Hiding, Second International Workshop, IH'98, Portland, Oregon, U.S.A., April 15-17, Proceedings, LNCS 1525, Springer-Verlag, ISBN 3-540-65386-4, pp. 219-239.

Fabien A. P. Petitcolas. 2000. *Watermarking Schemes Evaluation*. I.E.E.E. Signal Processing, vol. 17, no. 5, pp. 58–64, September.

Friskayanti, Dinda. 2011. *Watermarking untuk Gambar Digital dengan Menggunakan Metode Discrete Wavelet Transform (DWT)*. Universitas Brawijaya. Malang.

Gale. 2010. *Digital Audio*. http://manual.audacityteam.org/man/Digital_*Audio*. Terakhir diakses pada tanggal 11 Januari 2012 pada pukul 19.45 WIB.

Juergen, Seitz. 2004. *Digital Watermarking for Digital Media*, Information Science Publishing, Germany.

Katzenbeisser S. and Fabien, Petitcolas. 2000. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Boston, USA.

Kurniawan, Harris. 2010. *Audio Steganografi pada File Wav Menggunakan Metode Low Bit Coding dengan Modifikasi Low Significant Bit*. Universitas Brawijaya. Malang.

Rusdianto, Denny, S. 2009. *Watermarking Audio Digital Menggunakan Psychoacoustic Auditory Model dengan Metode Spread Spectrum*. Universitas Brawijaya. Malang.

Strang, G, dan Nguyen Truong. 1996. *Wavelet and Filter Banks*. MIT.

Watkinson, John. 2002. *An Introduction to Digital Audio 2nd Edition*. Focal Press, Inc.

Wilson, Scott. 2003. *The Cannonical Wave File Format*. http://www.ccrma.stanford.edu/courses/422/projects/*Wavefo rmat*.html. Terakhir diakses pada tanggal 11 Januari 2012 pada pukul 19.45 WIB.