VIDEO WATERMARKING UNTUK PERLINDUNGAN HAK CIPTA DENGAN ALGORITMA KOCH ZHAO

Tigus Juni Betri
Jurusan Informatika
FMIPA Universitas Sebelas Maret
Ir. Sutami No.36 A Surakarta
57126
juni.betry@gmail.com

Esti Suryani
Jurusan Informatika
FMIPA Universitas Sebelas Maret
Ir. Sutami No.36 A Surakarta
57126
suryapalapa@yahoo.co.id

Abdul Aziz
Jurusan Informatika
FMIPA Universitas Sebelas Maret
Ir. Sutami No.36 A Surakarta
57126
aaziz@staff.uns.ac.id

ABSTRAK

Penggunaan data digital sekarang ini sering disalahgunakan. Hal ini disebabkan selain kemudahan dalam penyebaran dengan menggunakan internet, juga dikarenakan penggandaan data digital semakin mudah dan murah. Kemudahan tersebut akhirnya dapat digunakan secara negatif tanpa memperhatikan aspek hak cipta. Oleh karena itu , maka produser data digital seperti gambar, audio, serta video mencari sebuah solusi teknis untuk permasalahan terkait perlindungan hak cipta.

Video watermarking adalah suatu teknologi yang bertujuan untuk menjaga dan melindungi hak cipta dan kepemilikan suatu video. Watermarking bisa dikatakan suatu teknik penyembunyian informasi tambahan ke dalam suatu data lainnya, tetapi orang lain tidak dapat melihat adanya data tambahan tersebut. Discrete Cosine Transform atau DCT merupakan salah satu teknik klasik dalam kompresi gambar yang digunakan dalam penyisipan watermark. Salah satu penerapan metode DCT adalah dengan algoritma Koch Zhao.

Hasil penelitian menunjukkan bahwa video watermarking dengan algoritma Koch zhao dapat diterapkan dengan cukup baik pada video dengan format Mp4. Hal ini ditunjukkan dengan hasil dari pengujian terhadap data sampel yang menghasilkan nilai PSNR (Peak Signal Noise Ratio) tinggi. Video yang sudah ditanami watermark mempunyai nilai yang terendah 30,12 dB dan tertinggi adalah 36,98 dB. Sementara pengujian terhadap ketahanan watermark menghasilkan tingkat kesamaan 100%.

Kata Kunci
Koch Zhao, Mp4,Watermark

1. PENDAHULUAN

Perkembangan teknologi yang pesat di dunia saat ini mendorong pertumbuhan banyak bidang. Terutama dalam bidang komputerisasi termasuk multimedia, yang meliputi pembuatan file dalam bentuk gambar, suara ataupun video.
Kemudahan untuk memanipulasi, reproduksi dan distribusi dokumen *digital* seperti file-file multimedia dapat menciptakan masalah dan merugikan suatu pihak. Karena itu maka pelaku atau produser data *digital* seperti gambar, audio, serta video mencari sebuah solusi teknis untuk permasalahan terkait perlindungan hak cipta.

Digital watermark adalah suatu teknologi yang bertujuan untuk menjaga dan melindungi hak cipta dan kepemilikan suatu data multimedia. *Digital watermark* menyisipkan informasi ke dalam data *digital* dengan cara yang rahasia dan tidak mencurigakan. *Digital watermark* merupakan kode identifikasi atau pola *bit* terintegrasi ke dalam data multimedia sebagai hak cipta yang membantu dalam mengidentifikasikan audio, video, atau citra yang didistribusikan secara ilegal. *Digital watermark* dapat disebut juga suatu bentuk *steganography*. *Steganography* sendiri adalah seni dan ilmu komunikasi dalam` menyembunyikan keberadaan dari komunikasi. Hasil yang ingin dicapai dari teknik ini adalah menyembunyikan pesan tanpa mengubah pesan sehingga tidak terdeteksi oleh pihak yang tidak dikehendaki [6].

Teknik *watermarking* dibedakan menjadi beberapa domain berdasarkan tempat penyisipan. Diantara teknik *watermarking* yang sering diterapkan adalah teknik *watermarking* dengan domain *Discrete Cosine transform* (DCT) dan *Discrete Wavelet Transform* (DWT). Sementara dalam teknik penyisipannya sendiri ada bermacam-macam, diantaranya dengan teknik menumpuk *frame*/*image* dengan *image watermark* [3], teknik memotong *frame* menjadi *image* dari video [13], atau dengan teknik *bit plane coding* [14].

Setiap teknik *watermarking* tersebut memiliki kelebihan dan kekurangan sendiri-sendiri. Misalnya kelebihan teknik *watermarking* dengan *Discrete Cosine Transform* (DCT) adalah DCT menghitung kuantitas *bit-bit image* dimana pesan tersebut disembunyikan di dalamnya. Teknik ini menerapkan penyisipan *watermark* pada domain frekuensi di dalam *image*, bukan pada domain spasial sehingga tidak akan ada perubahan mencolok yang terlihat di cover *image*[10]. Sedangkan kekurangannya adalah tidak tahan terhadap perubahan suatu obyek [13]. Pada teknik *watermarking Discrete Wavelet Transform*, ketahanan terhadap perubahan obyek lebih teruji, tapi penyisipan citra *watermark* dilakukan ke dalam citra asli dengan menyisipkan citra *watermark* ke dalam koefisien *wavelet* dari citra asli sehingga ada potensi *watermark* menjadi *visible* [3].

Algoritma Koch Zhao adalah salah satu teknik penyisipan *watermark* yang bekerja di domain *Discrete Cosine Transform*. Algoritma ini adalah algoritma penyisipan metode DCT yang sudah diperbaiki [9]. Pada *watermark* yang dilakukan dengan DCT ketahanan *watermark* menjadi kelemahan utama, tapi algoritma Koch Zhao menjadi solusi kelemahan tersebut dengan memperbaiki proses penyisipan. Oleh karena itu, algoritma Koch Zhao dipilih dalam penelitian ini karena algoritma ini bisa mengatasi permasalahan dalam *watermarking* yang diterapkan pada domain DCT, yaitu mengenai ketahanan *watermark* pada file video[7].

2. LANDASAN TEORI

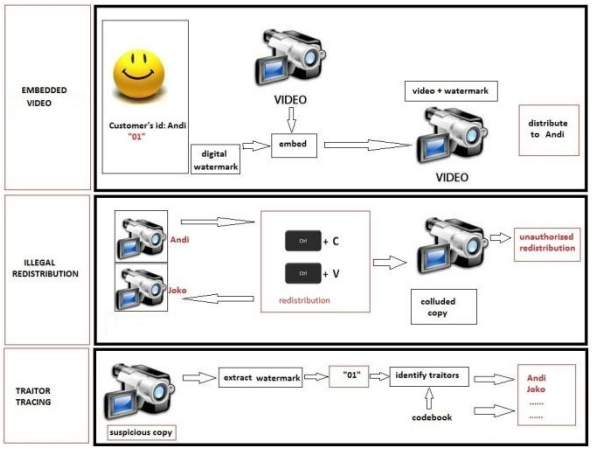
2.1 Watermarking

Watermark adalah kode yang berisi informasi mengenai pemilik hak cipta, pencipta, pembeli yang sah dan segala sesuatu yang diperlukan untuk menangani hak kepemilikan pada media digital [8]. *Watermark* sengaja ditanamkan secara permanen pada data digital sedemikian hingga pengguna yang tidak berwenang tidak dapat membacanya, disisi lain *watermark* tersebut haruslah tidak mengubah isi media kecuali sedikit atau perubahan tersebut tidaklah tampak atau kurang begitu tampak bagi indera manusia [2].

2.2 Prinsip Kerja Watermarking

Suatu data (misalnya file video) hendak diterbitkan suatu *content digital* kepada konsumen. Untuk menghindari kecurangan dalam penyebaran file yang mungkin dapat dilakukan konsumen, maka dibuat *watermark* untuk file

tersebut. Gambar 1 menjabarkan skema penyebaran *watermark*.



Gambar 1. Skema Penyebaran Watermark

Misalnya *provider* menjual *file* kepada Andi dimana *file* tersebut berisi bit 01. Kemudian *file* tersebut digandakan secara ilegal oleh Joko. Proses *watermarking* menggunakan *robust watermarking* (kokoh), sehingga ia bisa mendapatkan *watermark* kembali untuk melacak sumber pendistribusian ilegal. *File* hasil penggandaan milik Joko bisa dideteksi asal kepemilikannya dengan cara mendeteksi isi *watermark* tersebut yaitu *file* dengan isi *watermark* bit 01 milik Andi. Selanjutnya tinggal melakukan perhitungan dengan konsumen terkait [11].

2.3. Karakteristik Watermarking

Ada beberapa karakteristik sistem *watermarking* seperti *robustness*, *tamper resistance*, *fidelity*, dan *computational cost*. Dimana setiap karakteristik tersebut terdapat *trade-off* diantaranya. Evaluasi terhadap karakteristik sistem *watermarking* tidak sama untuk semua aplikasi, sehingga pemilihan *trade-off* yang sesuai harus benar-benar dipertimbangkan berdasarkan aplikasi *watermarking*. Teknik *watermarking* diterapkan pada *file text*, *image*, audio dan video [1].

2.4 MP4

MPEG telah dikenal sebagai *codec* standar dunia dalam bidang video *digital*. Perkembangan MPEG terkini telah mencapai generasi MPEG-4 atau biasa lebih familiar disebut Mp4. Mp4 memiliki beberapa kelebihan. Pertama, MP4 mampu memberi kualitas yang sama baiknya seperti MPEG-2 dengan *bitrate* yang jauh lebih kecil dari MPEG-2. Efisiensi tinggi ini didapat berkat teknologi *Scalable Video Coding*. *File* Mp4 dengan *bitrate* yang kecil memiliki ukuran *file* yang juga termasuk kecil. Berkat teknologi tersebut sebuah *file* video yang dibuat memakai MP4 hanya memiliki ukuran *file* seperempat dari video yang dibuat memakai MPEG-2 sehingga memungkinkan sekeping DVD dapat menampung beberapa film yang dikompres memakai format MP4 dengan kualitas yang tetap baik.

Kedua, MP4 menghasilkan gambar yang lebih baik daripada MPEG-2 pada pemakaian *bit rate* yang rendah. Pada kompresi MPEG-2, kualitas gambar hanya dapat dipertahankan apabila kita memakai pilihan *bit rate* tinggi, dan begitu nilai *bit rate* diturunkan maka kualitasnya akan langsung turun. Namun, MP4 mampu memberi kualitas gambar lebih baik pada *bit rate* rendah sehingga cocok untuk ruang simpan yang terbatas seperti pada peranti genggam [2]

2.5 *Discrete Cosine Transform*

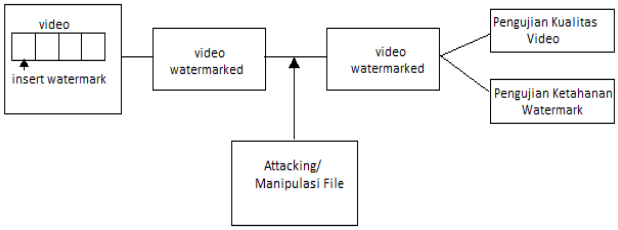
Discrete Cosine Transform atau DCT adalah salah satu teknik klasik dalam kompresi gambar. DCT memecah gambar menjadi tiga frekuensi band yang berbeda. Sub band tersebut adalah frekuensi tinggi, frekuensi rendah dan frekuensi tengah yang digunakan untuk menyisipkan informasi atau lainnya. Metode ini bisa diterapkan pada gambar atau video. Frekuensi tengah dipilih sebagai tempat penyisipan *watermark* karena jika dilakukan perubahan pada frekuensi ini tidak akan mempengaruhi kualitas gambar (12)

2.6 Algoritma Koch Zhao

Algoritma Koch-Zhao sebenarnya merupakan algoritma yang dikembangkan dari DCT , diterapkan pada media *digital* dalam bentuk citra. Algoritma ini bekerja blok demi blok dengan ukuran 8 piksel x 8 piksel dalam domain DCT. Melihat domain kerja algoritma ini, berarti algoritma Koch Zhao ini memiliki kemiripan dengan metode kompresi JPEG untuk citra. Hasil perbaikan algoritma Koch-Zhao ini diterapkan dalam video berdasarkan kemiripan setiap *frame* dari video seolah-olah merupakan sebuah citra. Dalam *watermarking* video, setiap *frame* video dianggap sebagai sebuah citra. (7)

3 METODE PENELITIAN

Penerapan *watermarking* terdiri dari tiga tahapan utama yaitu proses *embedding*, *extracting* dan pengujian, dimana dalam setiap tahapan ada beberapa proses yang terjadi. Seperti pada *embedding*, sebelum dilakukan penyisipan video dipecah terlebih dahulu menjadi *image* setelah itu baru dilakukan penyisipan *watermark* terhadap *image*. Gambar 2 di bawah ini merupakan skema penelitian secara umum *video watermarking:*



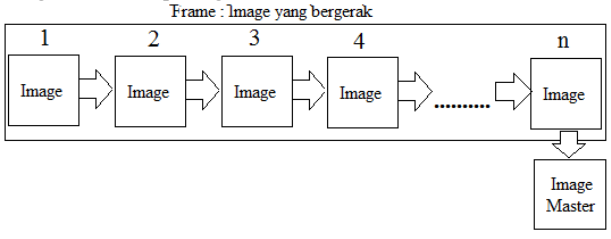
Gambar 2. Skema Penelitian (4)

Penerapan video *watermarking* dengan algoritma koch zhao, secara garis besar terdapat beberapa tahap, yakni;

1. Proses pemecahan video menjadi *image*.
2. *Embedding*
3. Penggabungan *image* menjadi video
4. *Extracting*
5. Tahap pengujian

3.1 Pemecahan Video Menjadi Image

Proses pemecahan video menjadi image dilakukan dengan skema seperti gambar 3 berikut ini:

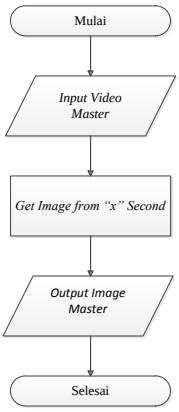


Gambar 3 Skema Pemecahan Video Menjadi Image

Penjelasan dari skema pemecahan video menjadi image pada gambar 3 di atas adalah sebagai berikut:

1. File video terdiri dari frame atau satu gambar tunggal yang menjadi satu rentetan gambar yang membentuk video
2. Dalam pemecahan video menjadi image. Video dipecah menjadi beberapa image
3. Image pada frame pertama diambil untuk dijadikan sebagai image yang akan ditanam *watermark* (*image master*)

Gambar 4 berikut adalah alur proses pemecahan video menjadi *image* :



Gambar 4 Alur Proses Pemecahan Video Menjadi Image

Algoritma dalam pemecahan video menjadi *image* dijabarkan dalam algoritma 3.1 berikut ini;

Algoritma 3.1

Input : Video mp4 (video *master*)
Output : *image* (disebut *image master*)
Proses :

1. Proses pemecahan video menjadi *image* dilakukan dengan memasukkan video mp4 yang akan dipecah.
2. Proses pengambilan *image* dilakukan dengan model *get image from x second*, memecah *frame* video dengan cara mengambil *screenshoot* video pada setiap *frame per second*. Jadi misalnya video dengan durasi 3 detik dengan 29 fps maka *image* yang diambil adalah pada setiap 0.034 detik (3 detik : (29 fps x 3 detik))

3.2 *Embedding Watermark*

Proses *embedding* / penyisipan *watermark* dilakukan untuk menghasilkan nilai *watermark* berdasarkan metode DCT. *Flowchart* proses *embedding* dijabarkan seperti *flowchart* pada gambar 5:

Gambar 5 Flowchart Alur Proses Penyisipan Watermark ke Image

Algoritma dalam penyisipan *watermark* dijabarkan seperti pada algoritma 3.2 berikut ini:

Algoritma 3.2

Input : - I*mage master*
- *Key*
- Teks *watermark*

Output : - *Image watermarked*
Proses :

1. Proses transformasi *image* dengan DCT 8x8 piksel kemudian dipetakan dengan Persamaan 1(5):

$$T(i,j) = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{if } i \neq 0 \end{cases}$$

……(1)

i,j = (0, . . . N-1) (0 . . . M-1)
N = panjang matriks , M = lebar matriks

2. Proses kuantisasi dengan persamaan 2 (5):

$$C_{ij} = \text{round } \frac{Dij}{Qij}$$ ……(2)

D: Matriks koefisien DCT
Q: Matriks kuantisasi

3. Hasil koefisien tersebut, digunakan untuk menentukan daerah-daerah yang cocok untuk penyisipan *watermark.*
4. Proses DCT akan memecah masing-masing gambar menjadi blok-blok berukuran 8x8 piksel. Setelah mengalami pemecahan tersebut, gambar-gambar akan dicek nilainya (nilai bobot intensitas warna), kemudian akan mulai dilakukan penyisipan nilai-nilai *bit* dari gambar ke pesan ke frekuensi tengah dari blok-blok yang dicek dari gambar asli, dengan ketentuan sebagai berikut:

-jika nilai *bit* dari gambar berisi 0 maka sisipkan nilai_0 ke dalam frek_tengah dari blok_dct seperti ditunjukkan *code* penyisipan pada gambar 6 sampai 8

```
If{vector_pesan(kk)==0}
                    for
ii=1:ukuran_blok
        for jj=1 :ukuran_blok
        if{frek_tengah(jj,ii)==1}
        blok_dct(jj,ii)=blok_dct(jj
,ii)+k*nilai_0(11);
                    11=11+1;
End
```

Gambar 6. Code Embed Watermark

Jika tidak sisipkan nilai_1 ke dalam frek_tengah komponen dari blok_dct

```
Else
For ii=1:ukuran_blok
    For jj=1:ukuran_blok

If{frek_tengah(jj,11)==1}
        Blok_dct(jj,ii);

Blok_dct(jj,ii)=blok_dct(jj,ii)+k*
nilai_1(11);
        11=11+1
    End
```

Gambar 7 Code Embed Watermark

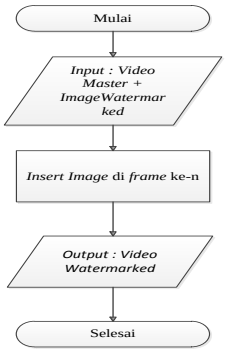
Setelah tiap frekuensi tengah dan gambar asli mengalami penyisipan nilai-nilai bit gambar pesan,maka blok-blok tersebut kembali diubah ke dalam bentuk domain spasial dengan menggunakan invers dari DCT. Kemudian lanjutkan proses tadi ke blok-blok pada baris berikutnya.

```
If (ukuran_blok) >=Nc
  X=1;
Else
  X= X+ukuran_blok;
end
```

Gambar 8 Code Embed Watermark

3.3 Penggabungan Image Ke Video

Proses penggabungan image ke video dilakukan dengan skema seperti ditunjukkan pada gambar 9;



Gambar 9. Skema Penggabungan Image ke Video

Algoritma dalam penggabungan *image* ke video dijabarkan pada algoritma 3.3 berikut ini:

Algoritma 3.3

Input : - *Video master*
- *Image watermarked*

Output : - *Video watermarked*

Proses :

1. Memasukkan kembali *image* yang sudah ditanam *watermark* ke dalam video dengan menggunakan software aplikasi VideoPad Editor.
2. Posisi *image* diletakkan pada *frame* ke-n atau *frame* yang sama dengan pada saat pemecahan sehingga posisinya kembali seperti pada video semula.

3.4 Extracting

Proses ekstrak *watermark* dilakukan setelah video *watermarked* (video berisi *watermark*) dipecah lagi menjadi *image*. *Flowchart* proses *extracting* dijabarkan seperti pada gambar 10:

Gambar 10 Alur Proses Ekstrak Watermark

Algoritma dalam *extracting watermark* dijabarkan pada algoritma 3.4 berikut ini:

Algoritma 3.4

Input : - *Video watermarked*
- *Key*

Output : - *Watermark (txt)*

Proses :

1. Koefisien *image* ditransformsi dengan DCT menggunakan Persamaan 4.1
2. Matriks hasil proses DCT dilakukan kuantisasi dengan Persamaan 4.2 :
3. Proses ekstrak *watermark* hampir sama dengan proses penyisipan. Perbedaanya pada ekstraksi, nilai-nilai *bit* dari gambar di frekuensi yang ditambahi *watermark* akan dihilangkan dengan operasi pengurangan. *Code* ekstrak *watermark* ditunjukkan pada gambar 11 dan 12:

```
If{vector_pesan(kk)==0}
        for
ii=1:ukuran_blok
                    for    jj=1
:ukuran_blok
  if{frek_tengah(jj,ii)==1}
```

Gambar 11. Alur Proses Ekstrak Watermark

Setelah tiap frekuensi tengah dan gambar sudah dibersihkan dari nilai-nilai bit *watermark* kemudian diubah kembali ke dalam bentuk domain spasial dengan menggunakan invers dari DCT. Kemudian lanjutkan proses tadi ke blok-blok pada baris berikutnya.

```
If (ukuran_blok) >=Nc
        X=1;
Else
        X= X+ukuran_blok;
end
```

Gambar 12. Alur Proses Ekstrak Watermark

3.5 Tahap Pengujian

Tahap pengujian terdiri dari beberapa tahap,yaitu:

1. Pengujian terhadap kualitas video *watermarked*. Pengujian ini meliputi;
   a. Kualitas video *watermarked* (dB)
      Pengujian kualitas audio dan kualitas visual dilakukan dengan menghitung PSNR . PSNR (*Peak Signal Noise Ratio*) merupakan nilai perbandingan antara nilai maksimum hasil rekonstruksi dengan nilai *Mean Square Error* (MSE). PSNR dihitung dengan Persamaan 3 (5):

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE} dB$$

……(3)

   b. Tingkat kesamaan video *master* dengan video *watermarked*.
      Pengujian kesamaan video *master* dan video *watermarked* dilakukan dengan cara melihat dengan kasat mata apakah video yang sudah ditanam *watermark* terdapat perbedaan dengan video sebelum ditanam *watermark*.
2. Pengujian terhadap kualitas *watermark.* Pengujian ini dilakukan dengan memberikan serangan untuk menguji *watermark*. Video yang telah disisipi *watermark* diberi serangan berupa *convert*, *compression*, dan pemberian *noise gaussian*.

4 HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi

Penerapan video *watermarking* dengan algoritma koch zhao, secara garis besar terdapat beberapa tahap, yakni

1. Proses pemecahan video menjadi *image*.
2. *Embedding*
3. Penggabungan *image* menjadi video
4. *Extracting*
5. Tahap pengujian

4.1.1 Hasil Implementasi Pemecahan Video menjadi *Image*

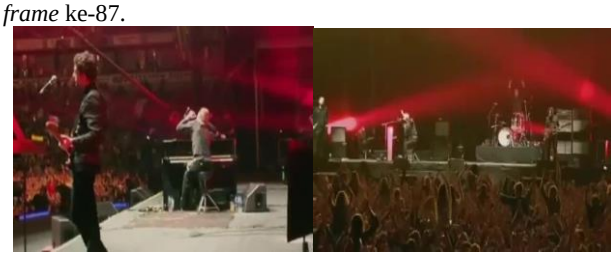
Implementasi pemecahan video menjadi *image* dilakukan dengan memotong tiap *frame* dari video. Proses pengambilan *image* dilakukan dengan model *get image from x second*, memecah *frame* video dengan cara mengambil *screenshoot* video pada setiap *frame per second*. Sebagai contoh, misalnya video coldplay.mp4 dengan durasi 3 detik dengan 29 fps maka *image* yang dipecah adalah pada setiap 0.0344 detik (3 detik :

(29 fps x 3 detik). Gambar 8 menunjukkan code pemecahan video menjadi *image*.

```
$ffmpeg = "C:\\ffmpeg\\bin\\ffmpeg";
        $videoFile = $_FILES["file"]["tmp_name"];
        $size = "640x480";
        for($num = 1; $num <= 87; $num++)
        {
         $interval = $num * 0.0344;
         shell_exec("$ffmpeg -i $videoFile -an -ss
$interval -s $size $num.jpg");
        echo "berhasil dipecah! - $num.jpg<br />";
        }
        echo "<br />$num berhasil dipecah!";
```

Gambar 13. Code Pemecahan Video Menjadi Image

Proses tersebut menghasilkan *image* berjumlah 87 dan *image* yang diambil untuk penyisipan *watermark* adalah *image* terakhir. Gambar 14 menunjukkan *image frame* ke-1 dan *image frame* ke-87.



(a) (b)

Gambar 14. (a) *Image* di *frame* ke-1 (b) *Image* di *frame* ke-87

4.1.2 Hasil Implementasi *Embedding*

Implementasi *watermarking* dilakukan dengan penyisipan *watermark* ke *image* dilakukan dengan algoritma Koch Zhao. Algoritma ini merupakan modifikasi dari teknik kompresi DCT. Algoritma DCT yang digunakan adalah DCT 2D (2 dimensi) karena obyeknya adalah *image*. DCT bekerja dengan memisahkan gambar ke bagian frekuensi yang berbeda. Frekuensi yang kurang penting dibuang. Kemudian, hanya frekuensi yang paling penting yang tetap digunakan mengambil gambar dalam proses dekompresi. Akibatnya, gambar direkonstruksi mengandung beberapa distorsi. Hasil implementasi menunjukkan proses dari penerapan algortma DCT 2D dalam penelitian ini. Berikut adalah urutan proses yang terjadi dalam algoritma DCT:

1. Gambar dibagi menjadi beberapa blok, dan masing-masing blok memiliki 8 piksel x 8 piksel. Hal ini dilakukan karena pemilihan 8x8 akan menghasilkan 64 blok sebagai tempat penyisipan. Pemilihan 64 *bit* informasi memiliki jumlah redundansi yang cukup banyak sehingga dapat bertahan meskipun dilakukan kompresi.
2. Data Matriks *original* dikurangi dengan 128 karena algoritma DCT bekerja pada rentang -128 sampai 127 sesuai dengan ketentuan pengolahan citra digital pada citra berwarna. Matriks *original* ditunjukkan pada matriks O pada gambar 15.

$$O = \begin{pmatrix} 104 & 101 & 99 & 97 & 96 & 96 & 96 & 99 \\ 101 & 96 & 95 & 95 & 96 & 98 & 99 & 99 \\ 99 & 92 & 91 & 92 & 97 & 102 & 103 & 99 \\ 98 & 90 & 89 & 91 & 98 & 104 & 104 & 98 \\ 98 & 90 & 89 & 92 & 97 & 103 & 104 & 98 \\ 99 & 91 & 90 & 92 & 97 & 102 & 103 & 98 \\ 99 & 92 & 91 & 93 & 97 & 102 & 102 & 98 \\ 100 & 93 & 92 & 94 & 98 & 102 & 102 & 98 \end{pmatrix}$$

Gambar 15. Matriks *Original Image* (Matriks O)

Matriks O atau *original* yang sudah dikurangi dengan 128 menghasilkan matriks M pada Gambar 16

$$M = \begin{pmatrix} -24 & -27 & -29 & -31 & -33 & -33 & -32 & -29 \\ -27 & -32 & -33 & -33 & -32 & -30 & -29 & -29 \\ -29 & -36 & -37 & -36 & -31 & -36 & -25 & -29 \\ -30 & -38 & -39 & -37 & -30 & -24 & -24 & -30 \\ -30 & -38 & -39 & -37 & -30 & -24 & -24 & -30 \\ -29 & -37 & -38 & -36 & -31 & -26 & -26 & -30 \\ -29 & -36 & -37 & -35 & -31 & -26 & -26 & -30 \\ -28 & -35 & -36 & -34 & -30 & -26 & -26 & -30 \end{pmatrix}$$

Gambar 16 Matriks *Original* Setelah dikurangi 128 (Matriks M)

Selanjutnya adalah mencari nilai untuk matriks *Discrete Cosine Transform* untuk matriks T dan buat matriks *transpose* nya untuk matriks T^t

$$T(i,j) = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{if } i \neq 0 \end{cases} \quad \ldots\ldots(4)$$

Maka dengan menggunakan rumus persamaan 4.1 dapat dihitung nilai matriks T mulai dari T (0,0) sampai T (7,7).

T (0,0) = $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{8}}$ = 0.3536

.

.

T (0.7) = $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{8}}$ = 0.3536

.

.

T (7.0) = $\sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} = \sqrt{\frac{2}{8}} \cos \frac{(2.0+1)7.180^{\circ}}{2N}$ = 0.0975

.

.

T (7.7) = $\sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} = \sqrt{\frac{2}{8}} \cos \frac{(2.7+1)7.180^{\circ}}{2N}$ = -0.0975

Maka dari perhitungan persamaan 1 didapatkan nilai untuk matriks T seperti ditunjukkan pada Gambar 17 dan matriks *transpose* T^t seperti ditunjukkan pada Gambar 18.

T=

0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
0.4904	0.4157	0.2778	0.0975	−0.0975	−0.2778	−0.4157	−0.4904
0.4619	0.1919	−0.1913	−0.4619	−0.4619	−0.1913	0.1913	0.4619
0.4157	−0.0975	−0.4904	−0.2778	0.2778	0.4904	0.0975	−0.4157
0.3536	−0.3536	−0.3536	0.3536	0.3536	−0.3536	−0.3536	0.3536
0.2778	0.4904	0.0975	0.4157	−0.4157	−0.0975	0.4904	−0.4904
0.1913	−0.4619	0.4619	−0.1913	−0.1913	0.4619	−0.4619	−0.4619
0.0975	−0.2778	0.4157	−0.4904	0.4904	−0.4157	0.2778	−0.0975

Gambar 17. Matriks T

T^t=

0.3536	0.4904	0.4619	0.4157	0.3536	0.2778	0.1913	0.0975
0.3536	0.4157	0.1919	−0.0975	−0.3536	0.4904	−0.4619	−0.2778
0.3536	0.2778	−0.1913	−0.4904	−0.3536	0.0975	0.4619	0.4157
0.3536	0.0975	−0.4619	−0.2778	0.3536	0.4157	−0.1913	−0.4904
0.3536	−0.0975	−0.4619	0.2778	0.3536	−0.4157	−0.1913	0.4904
0.3536	−0.2778	−0.1913	0.4904	−0.3536	−0.0975	0.4619	−0.4157
0.3536	−0.4157	0.1913	0.0975	−0.3536	0.4904	−0.4619	0.2778
0.3536	−0.4904	0.4619	−0.4157	0.3536	−0.4904	−0.4619	−0.0975

Gambar 18. Matriks T^t

3. Langkah berikutnya dicari matriks D dimana D akan digunakan untuk proses perhitungan lanjutan dengan matriks *image* (matriks M) . Matriks D dicari dengan menggunakan persamaan 5

$$D = T \,.M\, . T^t \quad \text{.....(5)}$$

D=

−148.00	−16.1592	11.3996	19.2935	0.50	6.1145	2.0431	1.5029
2.2777	8.2394	1.8936	−6.5075	0.8284	−1.8108	0.0814	−0.2827
3.9989	11.1516	0.6036	−7.3684	1.5771	−1.8480	0.1036	−0.2827
0.2381	4.3461	0.8372	−2.9047	0.6091	−0.2061	−0.2008	−0.4761
1.0000	1.3390	−0.3266	−0.1420	−0.0000	−0.1829	−0.1353	−0.1688
0.0710	0.1907	0.1665	0.3974	−0.4070	−0.5022	0.2587	0.0355
0.5084	0.2214	0.6036	0.1661	−0.1121	−0.0122	−0.1036	0.3436
0.0179	−0.1792	0.2387	−0.3785	−0.1648	0.2781	−0.1218	−0.3328

Gambar 19 Matriks Hasil Perhitungan Matriks M , T dan T^t (Matriks D)

Matriks D yang ditunjukkan pada Gambar 19 berisi koefisien DCT, akan digunakan pada langkah selanjutnya yaitu kuantisasi dengan menggunakan tabel kuantisasi dengan level terpilih.

4. Matriks D sekarang berisi dengan koefisien DCT, dimana data yang terletak pada kiri atas merupakan korelasi dari frekuensi - frekuensi rendah dari data *original*. Sedangkan yang terletak pada kanan bawah merupakan korelasi dari frekuensi – frekuensi tinggi dari data *original*. Setelah itu dilakukan proses kuantisasi dengan *Quality level* 50 seperti diperlihatkan pada gambar 20.

Q_{50}=

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	69	56
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Gambar 20. Matriks untuk Kuantisasi (Matriks Q50)

Persamaan 6 merupakan persamaan matriks kuantisasi, dimana *round* berarti mendekatkan nilai hasil pembagian ke pembulatan bilangan integer terdekat.

C_{ij} = round

$$\frac{Dij}{Qij} \quad \text{.....(6)}$$

D: Matriks koefisien DCT

Q: Matriks kuantisasi

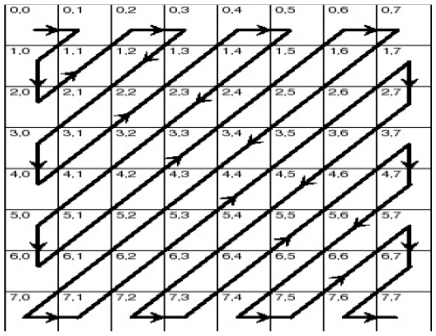
5.

C =

−15	−1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Gambar 21. Matriks Hasil Kuantisasi (Matriks C)

Matriks C pada gambar 21 adalah hasil kuantisasi D dan Q_{50}. Langkah selanjutnya disusun bilangan menggunakan fungsi *zig zag scanning* dimana ini merupakan langkah terakhir pada proses kompresi. Algoritma penyisipan dan pendeteksian *watermark* memerlukan pengurutan *zig-zag* sebagaimana yang dipergunakan dalam algoritma kompresi JPEG. Pengurutan secara *zig-zag* ini dimaksudkan untuk membuat sebuah vektor yang menyatakan urutan koefisien DCT mulai dari koefisien DCT dengan frekuensi rendah sampai dengan koefisien DCT yang memiliki frekuensi tinggi. Indeks pengurutan berjalan mulai dari koefisien paling atas bagian kiri dan bergerak searah anak panah didalam gambar, sampai kemudian berakhir pada ujung kanan bawah matriks koefisien DCT. Pengurutan secara *zig-zag* ini dimaksudkan untuk membuat sebuah vektor yang menyatakan urutan koefisien DCT mulai dari koefisien DCT dengan frekuensi rendah sampai dengan koefisien DCT yang memiliki frekuensi tinggi. Pengurutan *zig-zag* yang digunakan di dalam algoritma ini dapat dilihat dalam gambar 22.



Gambar 22. Pengurutan zig zag scanning

6. Algoritma Koch Zhao bekerja dengan menyisipkan *watermark* X = {x1,x2,…,xn}ke dalam sekelompok koefisien DCT citra terpilih C = { $C_{L+1}, C_{L+2}, \ldots C_{L+n}$ } .
Kemudian hasilnya adalah C' yang akan disisipkan kembali ke dalam urutan *zig-zag*nya kembali. Sebagai contoh untuk file 87.jpg yang disisipi *watermark* bertuliskan "*WATERMARK*" dengan kapasitas 9 kb akan menghasilkan X yaitu deret matriks baru yang akan diinisialisasikan ke deret hasil *zig-zag*. *Watermark* tersebut disisipkan dengan *key* angka "9" . *Key* berupa angka "9" dan kata "*WATERMARK*" akan dikonversikan ke dalam biner berdasarkan Tabel ASCII (*American Standard Code For Information Interchange*) sehingga hasilnya ditunjukkan Tabel 1:

Tabel 1. Konversi Watermark ke ASCII

Konversi ke ASCII			
Huruf yang Disisipkan	**Hasil Konversi**	**Kunci yang Disisipkan**	**Hasil Konversi**
W	01010111	9	00001001
A	01000001		
T	01010100		
E	01000101		
R	01010010		
M	01001101		
A	01000001		
R	01010010		
K	01001011		

Hasil dari konversi biner *key* angka " 9 " kemudian disisipkan pada hasil konversi biner kata "*WATERMARK*" untuk menghasilkan angka baru seperti ditunjukkan Tabel 2

Tabel 2. Hasil konversi ke ASCII

Konversi ke ASCII			
Huruf yang Disisipkan	**Hasil Konversi**	**Bilangan Biner yang Disisipkan**	**Hasil**
W	01010111	00001001	01011111
A	01000001	00001001	01001001
T	01010100	00001001	01011001
E	01000101	00001001	01001101
R	01010010	00001001	01011011
M	01001101	00001001	01001101
A	01000001	00001001	01001001
R	01010010	00001001	01011011
K	01001011	00001001	01001011

Hasil dari penyisipan pada Tabel 1 dipetakan menjadi matriks 8x8. *Watermark* yang berisi lebih dari 8 karakter, maka karakter berikutnya dijumlahkan kembali dengan urutan atas ke bawah. Sebagai contoh, hasil penyisipan karakter K pada Tabel 1 dijumlahkan kembali dengan hasil penyisipan karakter W, sehingga matriks akhir adalah matriks X seperti diperlihatkan pada gambar 23:

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

Gambar 23. Matriks Sisipan *Watermark* (Matriks X)

Matriks C adalah nilai akhir dari proses konversi DCT.
Matriks X adalah nilai input *watermark*
Matriks C' adalah hasil inisialisasi matriks C dengan matriks X dengan ketentuan :

$$C'(ij) = \begin{cases} Cij, & if\ Cij = 0 \\ Xij, & if\ Cij \neq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

Sehingga dengan perhitungan persamaan 4.4, hasilnya adalah matriks C' seperti pada gambar 7:

$$C' = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Gambar 24. Matriks Hasil Perhitungan *Watermark* (Matriks C')

Langkah terakhir adalah menggabungkan kembali nilai matriks C' ke dalam matriks O pada gambar 4.3. Hasil pernjumlahan tersebut adalah matriks O' yang ditunjukkan gambar 25 berikut:

$$O' = \begin{pmatrix} 104 & 102 & 100 & 97 & 95 & 95 & 96 & 99 \\ 110 & 97 & 95 & 95 & 95 & 98 & 99 & 99 \\ 99 & 93 & 91 & 92 & 97 & 102 & 103 & 99 \\ 98 & 90 & 89 & 91 & 98 & 103 & 103 & 98 \\ 98 & 90 & 89 & 92 & 97 & 103 & 104 & 98 \\ 99 & 91 & 90 & 92 & 97 & 102 & 103 & 98 \\ 99 & 92 & 91 & 93 & 97 & 102 & 102 & 98 \\ 100 & 93 & 92 & 94 & 94 & 102 & 102 & 98 \end{pmatrix}$$

Gambar 25. Matriks Hasil Akhir Koefisien Image (Matriks O')

Matriks O' adalah nilai akhir koefisien yang berhasil disisipi dengan *watermark*. Nilai ini akan otomatis masuk untuk menggantikan koefisien *image* di posisi yang diambil sebagai tempat penyisipan *watermark*.

4.1.3 Hasil Implementasi Penggabungan *Image* ke Video

Implementasi penggabungan *image* ke dalam video dilakukan dengan memasukkan *image* yang sudah disisipi *watermark* (*image watermarked*) ke dalam video yang belum berisi *watermark* (video *master*) sesuai dengan posisi *frame* nya. Hasil dari proses ini adalah video yang sudah berisi *watermark* (video *watermarked*). Proses penggabungan ini dilakukan dengan fungsi *ffmpeg* yang bisa didownload secara gratis. Perintah yang digunakan untuk proses ini ditunjukkan gambar 26:

```
ffmpeg -itsoffset 5 -i in.mp4 -r 0.034 -loop
1   -i   87_wtm.jpg   -filter_complex   "[1:v]
fade=out:125:25:alpha=1 [intro]; [0:v][intro]
overlay  [v]"  -map  "[v]"  -map  0:a  -acodec
```

Gambar 26. *Code* Penggabungan *Image* Ke Video

4.1.4 Hasil Implementasi *Extracting*

Hasil implementasi *extracting* dilakukan dengan mencari koefisien awal urutan dengan cara mendeteksi dan memisahkan *watermark*. Langkah awal yang dilakukan adalah dengan menggunakan metode yang sama dengan *embedding*, yakni dengan menerapkan metode DCT untuk mencari nilai koefisien akhir. Selanjutnya dengan memasukkan kunci yang sama maka akan menghasilkan deret yang sama. Nilai koefisien akhir akan dikurangkan dengan matriks input *watermark* sehingga hasil akhir adalah matriks *watermark*. Berikut urutan tahapan yang terjadi dalam proses *extracting*:

1. Mengambil komponen yang berisi *watermark* dengan metode yang sama seperti *embedding*. Matriks O' pada gambar 27 adalah koefisien matriks *image* yang sudah berisi *watermark* .

$$O' = \begin{pmatrix} 104 & 102 & 100 & 97 & 95 & 95 & 96 & 99 \\ 110 & 97 & 95 & 95 & 95 & 98 & 99 & 99 \\ 99 & 93 & 91 & 92 & 97 & 102 & 103 & 99 \\ 98 & 90 & 89 & 91 & 98 & 103 & 103 & 98 \\ 98 & 90 & 89 & 92 & 97 & 103 & 104 & 98 \\ 99 & 91 & 90 & 92 & 97 & 102 & 103 & 98 \\ 99 & 92 & 91 & 93 & 97 & 102 & 102 & 98 \\ 100 & 93 & 92 & 94 & 94 & 102 & 102 & 98 \end{pmatrix}$$

Gambar 27. Matriks Koefisien *Image Watermarked* (Matriks O')

2. Mengurangi masing-masing nilai matriks O'dengan 128. Hasilnya adalah matriks M' pada Gambar 28

$$M' = \begin{pmatrix} -28 & -27 & -29 & -31 & -33 & -33 & -32 & -29 \\ -27 & -32 & -33 & -33 & -32 & -30 & -29 & -29 \\ -29 & -35 & -37 & -36 & -31 & -36 & -25 & -29 \\ -30 & -38 & -39 & -37 & -30 & -24 & -24 & -30 \\ -30 & -38 & -39 & -36 & -31 & -25 & -24 & -30 \\ -29 & -37 & -38 & -36 & -31 & -26 & -25 & -30 \\ -29 & -36 & -37 & -35 & -31 & -26 & -26 & -30 \\ -28 & -35 & -36 & -34 & -30 & -26 & -26 & -30 \end{pmatrix}$$

Gambar 28 Matriks Setelah dikurangi 128 (Matriks M')

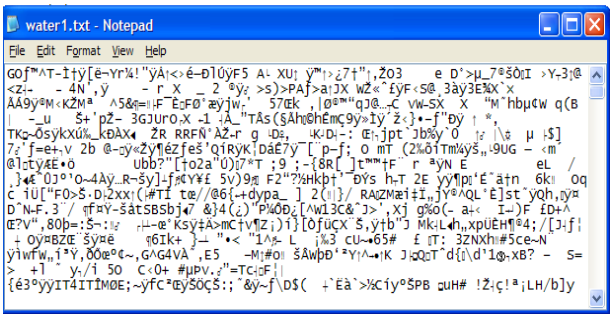
3. Selanjutnya adalah mencari nilai untuk matriks *Discrete Cosine Transform* untuk matriks T dan buat matriks *transpose* nya untuk matriks T^t seperti pada langkah *embedding*. Perhitungan ini dilakukan dengan menggunakan persamaan 4.1. Hasilnya adalah matriks T seperti ditunjukkan pada gambar 29 dan matriks *transpose* T^t yang ditunjukkan pada gambar 30
4. Kemudian dicari matriks D dimana matriks ini akan digunakan untuk kuantisasi lanjutan. Perhitungan dilakukan dengan persamaan 2. Hasil dari perhitungan ini menghasilkan matriks D seperti ditunjukkan pada gambar 4.7.
5. Matriks D yang ditunjukkan pada Gambar 4.7 berisi koefisien DCT, yang kemudian akan dikuantisasi dengan matriks Q_{50}(Quality level 50) pada gambar 4.8 Persamaan proses kuantisasi ditunjukkan pada formula 4.3, dimana *round* berarti mendekatkan nilai hasil pembagian ke pembulatan bilangan integer terdekat.
6. Kunci yang sama digunakan untuk menghasilkan *watermark* yang sama yakni *watermark* bertuliskan *"WATERMARK"*dengan kapasitas 9 kb dan *key* angka " 9 ". Kombinasi tersebut menghasilkan matriks X seperti pada gambar 4.11:
7. Kemudian dengan persamaan 4.4 akan menghasilkan matriks C'.

   Langkah terakhir adalah dengan mengurangi nilai matriks O' dengan matriks C' sehingga nilai akhir adalah matriks O seperti ditunjukkan gambar 4.15.

$$O = \begin{pmatrix} 104 & 101 & 99 & 97 & 95 & 95 & 96 & 99 \\ 101 & 96 & 95 & 95 & 96 & 98 & 99 & 99 \\ 99 & 92 & 91 & 92 & 97 & 102 & 103 & 99 \\ 98 & 90 & 89 & 91 & 98 & 104 & 104 & 98 \\ 98 & 90 & 89 & 92 & 97 & 103 & 104 & 98 \\ 99 & 91 & 90 & 92 & 97 & 102 & 103 & 98 \\ 99 & 92 & 91 & 93 & 97 & 102 & 102 & 98 \\ 100 & 93 & 92 & 94 & 98 & 102 & 102 & 98 \end{pmatrix}$$

Gambar 29. Matriks O

Apabila kunci yang dimasukkan cocok, maka *watermark* akan terekstrak ke file water.txt



Gambar 30. File Hasil Ekstrak *watermark*

Hasil dari proses *extracting* adalah file *watermark* yang berupa .txt dan video yang sudah terlepas dari *watermark*.

4.1.5 Hasil Implementasi Tahap Pengujian

Hasil implementasi tahap pengujian mengukur kinerja fungsionalitas pada *video watermarking*. Disini akan dijelaskan mengenai data sampel yang digunakan, hasil kinerja fungsionalitas saat proses penyisipan (*embedding*) yang terdiri dari performa pemecahan video menjadi *image* dan performa penyisipan *watermark*, hasil kinerja fungsionalitas saat proses deteksi (*extracting*) serta hasil pengujian *watermark*.

4.1.5.1 Data Sampel

Percobaan dilakukan terhadap 10 berkas video format mp4 dengan atribut yang berbeda-beda. File video yang digunakan untuk data sampel memilik durasi, reolusi, *bitrate* dan *frame rate* agar lebih bervariasi. Pemilihan data sampel yang bervariasi dipilih agar supaya hasil antara video satu dan lainnya dapat dibandingkan dengan obyektif. Video yang digunakan untuk sampel adalah coldplay.mp4, sampel.mp4,

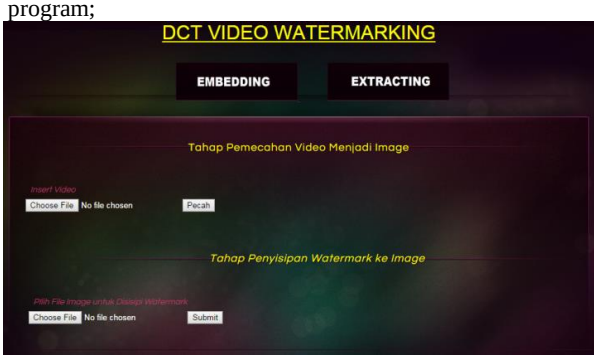
Litle.mp4, Nice face.mp4, Ball.mp4, tenis.mp4, Freeze.mp4, Glory.mp4, Merdeka.mp4, dan Menari.mp4. Tabel 3 merupakan penjelasan mengenai data file yang digunakan sebagai sampel;

Tabel 3. Data Sampel Tahap Pengujian

No	Nama File	Durasi (menit)	Ukuran Resolusi	Total *Rate*	*Frame rate*
1	Sample.mp4	00.07	320x240	383 Kbps	15 fps
2	Coldplay.mp4	00.03	640x480	1629 Kbps	29 fps
3	Litle.mp4	00.06	480x320	1441 Kbps	30 fps
4	Niceface.mp4	00.04	640x480	1441 Kbps	29 fps
5	Pict.mp4	00.21	320x240	578 Kbps	29 fps
6	Tom.mp4	00.27	1280x720	476 Kbps	29 fps
7	Freeze.mp4	00.11	480x320	1408 Kbps	30 fps
8	Glory.mp4	00.10	640x480	1240 Kbps	29 fps
9	Cartoon.mp4	00.09	854x480	476 Kbps	29 fps
10	Itunes.mp4	00.15	640x480	1030 Kbps	29 fps

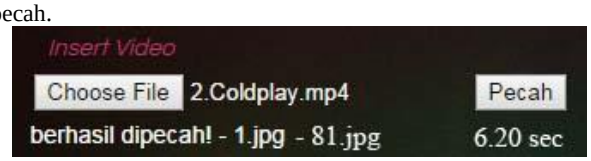
4.1.5.2 Performa Proses Pemecahan Video Menjadi *Image*

Performa proses pemecahan video menjadi *image* menjelaskan kecepatan yang dibutuhkan *video watermarking* dalam proses memecah video menjadi *image* pada setiap percobaan data sampel. Gambar 31 menunjukkan screenshot program;



Gambar 31. Tampilan Program Video *Watermarking*

Dalam proses *embedding*, video terlebih dahulu dipecah menjadi *image*. Gambar 32 menunjukkan proses pemecahan video dengan cara memilih video kemudian tekan tombol pecah.



Gambar 32. Tampilan Pemecahan Video menjadi *Image*

Tabel 4 merupakan data performa pemecahan video menjadi *image* terhadap data sampel;

Tabel 4. Tabel Performa Pemecahan Video Menjadi *Image*

No	Berkas Video	Ukuran	Dimensi	Waktu
1	Sample.mp4	0.3 mb	320x240	2.1 detik
2	Coldplay.mp4	0.6 mb	640x480	6.2 detik
3	Litle.mp4	1.0 mb	480x320	7.7 detik
4	Niceface.mp4	0.5 mb	640x480	7.8 detik
5	Pict.mp4	1.4 mb	320x240	8.2 detik
6	Tom.mp4	1.5 mb	1280x720	24.02 detik
7	Freeze.mp4	1.8 mb	480x320	28.12 detik
8	Glory.mp4	1.5 mb	640x480	30.10 detik
9	Cartoon.mp4	1.3 mb	854x480	32.16 detik
10	Itunes.mp4	1.8 mb	640x480	36.50 detik

Kecepatan pemecahan video ke *image* tergantung dari ukuan file video dan dimensi *image*. Berdasarkan Tabel 4.4, proses pemecahan video menjadi *image* pada percobaan 1 sampai 10 diperoleh rata-rata kecepatan yang dibutuhkan adalah berkisar antara 0.142 mb/detik dan yang paling lama adalah mencapai 0.049 mb/detik. Hal tersebut bisa diambil kesimpulan bahwa proses pemecahan video menjadi *image* tidak membutuhkan waktu yang lama.

4.1.5.3 Performa Proses *Embedding Watermark* Ke *Image*

Performa proses penyisipan video menjadi *image* menjelaskan proses penyisipan *watermark* ke *image* (hasil pemecahan) dan kecepatan yang dibutuhkan *video watermarking* dalam proses penyisipan *watermark* ke *image* pada setiap percobaan data sampel. Gambar 33 berikut ini adalah tampilan screenshoot program proses penyisipan *watermark*, dimana dalam proses yang pertama dilakukan adalah memilih *image* yang akan ditanam *watermark*/ *image master* kemudian memasukkan input berikutnya yaitu *key*,dan teks *watermark*. Setelah itu tekan tombol submit dan *watermark* akan disisipkan ke *image master*.

Gambar 33. Halaman Penyisipan *Watermark* ke *Image*

Tabel 4.5 adalah data performa penyisipan *watermark* ke *image* terhadap data sampel. Berkas *image* di Tabel 4.5 adalah *image* hasil pemecahan atau *image* yang belum berisi *watermark*. Performa penyisipan *watermark* diukur dari kecepatan penyisipan tersebut bila disisipi dengan isi *watermark* yang berbeda-beda. Hasilnya diperlihatkan pada Tabel 5 :

Tabel 5. Tabel Performa Proses Penyisipan *Watermark* Ke *Image*

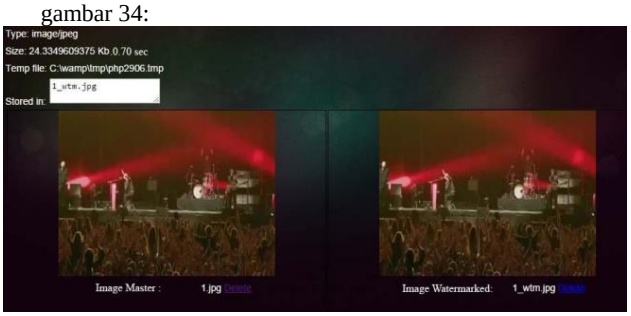
No	Berkas Video	Ukuran	Dimensi	Waktu
1	Sample.mp4	0.3 mb	320x240	2.1 detik
2	Coldplay.mp4	0.6 mb	640x480	6.2 detik
3	Litle.mp4	1.0 mb	480x320	7.7 detik
4	Niceface.mp4	0.5 mb	640x480	7.8 detik
5	Pict.mp4	1.4 mb	320x240	8.2 detik
6	Tom.mp4	1.5 mb	1280x720	24.02 detik
7	Freeze.mp4	1.8 mb	480x320	28.12 detik
8	Glory.mp4	1.5 mb	640x480	30.10 detik
9	Cartoon.mp4	1.3 mb	854x480	32.16 detik
10	Itunes.mp4	1.8 mb	640x480	36.50 detik

Berdasarkan Tabel 5, proses penyisipan *watermark* pada percobaan 1 sampai 10 dengan kapasitas *watermark* berbeda-beda menghasilkan kecepatan yang tidak signifikan. Hal ini diperlihatkan pada percobaan 1 yakni sample.jpg dengan kapasitas 19.5 kb dan kapasitas *watermark* 9 kb diperoleh kecepatan 21.60 kb/detik serta pada percobaan 10 yakni itunes.jpg dengan kapasitas 25 kb dan kapasitas *watermark* 13 kb diperoleh kecepatan 24,75 kb/detik . Hal tersebut bisa diambil kesimpulan bahwa perbedaan kapasitas *watermark* tidak menghasilkan perbedaan kecepatan yang signifikan dan secara keseluruhan proses penyisipan *watermark* ke *image* tidak membutuhkan waktu yang lama.

4.1.5.4 Hasil Proses *Embedding*

Hasil proses penyisipan menjelaskan perbandingan data hasil penyisipan yang meliputi :

a. Perbandingan antara berkas *image* sebelum dibubuhi *watermark* (*image master*) dan berkas *image* hasil penyisipan (*image watermarked*). Gambar 4.22 menunjukkan perbandingan antara *image master* dan *image* hasil penyisipan *watermark*. Berikut tampilannya seperti gambar 34:



Gambar 34. Perbandingan Gambar Sebelum dan Sesudah Penyisipan *Watermark*

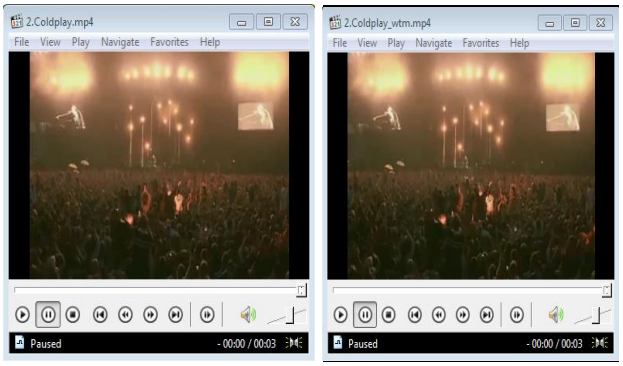
Perbandingan data file *image master* dan *image watermarked* ditampilkan dalam Tabel 6 dengan asumsi *watermark* yang disisipkan sama:

Tabel 6. Tabel Perbandingan Sebelum dan Sesudah Proses Penyisipan *Watermark* Ke *Image*

No	Nama File	Ukuran(kb)		*Bit* Depth (dpi)		Vertical& Horizontal Resolution (dpi)	
		Sebelum	Sesudah	Sebelum	Sesudah	Sebelum	Sesudah
1	Sample.jpg	19.5	23	96	96	96	96
2	Coldplay. Jpg	20	24	96	96	96	96
3	Tom. Jpg	20	24	108	108	108	108
4	Niceface. Jpg	22	24	210	210	210	210
5	Pict. Jpg	23	24	108	108	108	108
6	Freeze. Jpg	23	24	200	200	200	200
7	Glory. Jpg	24	25	96	96	96	96
8	Cartoon. Jpg	24	23	108	108	108	108
9	Itunes. jpg	25	25	96	96	96	96
10	Litle. Jpg	25	26	96	96	96	96

Berkas *image* yang telah ditanami *watermark* memiliki nilai *Vertical& Horizontal Resolution* (dpi) yang sama seperti saat sebelum ditanami *watermark*. Perbedaan hanya terletak di ukuran file yang sedikit berubah ukurannya dan *bit depth*. Perbandingan deskripsi antara *image master* dengan *image watermarked* dari Tabel 4.6 mengindikasikan bahwa file *watermark* tidak merubah file *image* secara signifikan sekaligus bisa disimpulkan proses penyisipan *watermark* ke *image* bisa berjalan dengan baik.

b. Perbandingan antara berkas video sebelum dibubuhi *watermark* (*video master*) dan berkas video hasil penyisipan (*video watermarked*). Setelah penyisipan *watermark* terhadap *image* selesai, maka yang dilakukan selanjutnya adalah menggabung *image* ke video. Gambar 35 di bawah ini menunjukkan perbandingan antara *image master* dan *image* hasil penyisipan *watermark*.



Gambar 35. Perbandingan Video Sebelum dan Sesudah diberi *watermark*

Hasil perbandingan data file antara berkas video yang belum berisi *watermark (video master*) dan video yang telah berisi *watermark* (*video watermarked*) terhadap data sampel ditunjukkan dalam Tabel 7:

Tabel 7. Tabel Perbandingan Sebelum dan Sesudah Proses Penyisipan *Watermark* Ke Video

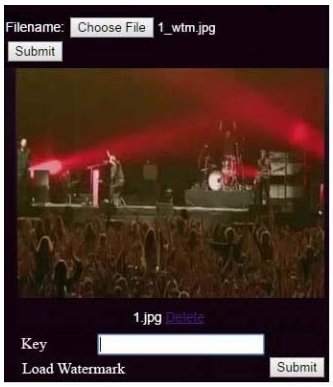
No	Nama File	Ukuran (mb)		Durasi (menit)		Total *Bitrate* (Kbps)		*Frame Rate* (Kbps)		Data *Rate* (Kbps)	
		Sebelum	Sesudah	Sebelum	Sesudah	Sebelum	Sesudah	Sebelum	Sesudah	Sebelum	Sesudah
1	Sample.mp4	0.33	0.41	00.07	00.07	383	482	15	15	256	353
2	Coldplay.mp4	0.60	0.49	00.03	00.03	1629	1332	29	29	1501	1205
3	Litle.mp4	1.05	0.90	00.06	00.06	1441	681	30	30	1441	677
4	Niceface.mp4	0.53	0.48	00.04	00.04	1441	1052	29	29	1404	1023
5	Pict.mp4	1.43	1.49	00.21	00.21	578	731	29	29	446	600
6	Tom.mp4	1.52	1.30	00.27	00.27	476	392	29	29	320	260
7	Freeze.mp4	1.81	1.05	00.11	00.11	1408	783	30	30	1408	779
8	Glory.mp4	1.53	1.45	00.10	00.10	1240	1596	29	29	1160	1467
9	Cartoon.mp4	1.31	1.22	00.09	00.09	476	1104	29	29	1001	966
10	Itunes.mp4	1.82	1.84	00.15	00.15	1030	1000	29	29	975	915

Berkas video yang telah berisi *watermark* memiliki kesamaan pada durasi seperti saat sebelum ditanami *watermark*. Perbedaan terletak pada ukuran, total *bitrate*, *framerate*, dan *data rate*. Perbandingan video sebelum dan sesudah dibubuhi *watermark* , dapat dilihat bahwa nilai *total bitrate* video mengalami penurunan. Hal ini disebabkan karena sebelum dibubuhi *watermark*, video terlebih dahulu dilakukan pemotongan *frame* pada video untuk mengambil *image* yang digunakan sebagai tempat menanam *watermark*, sehingga akan merubah nilai *bit rate*. Sedangkan untuk jumlah *frame*, proses ini tidak mengakibatkan perubahan. Perbedaan *bit rate* tersebut hanya kecil dan tidak berpengaruh banyak. Perbandingan deskripsi antara *video master* dengan video *watermarked* dari Tabel 7 menunjukkan bahwa penyisipan *watermark* di video tidak membuat perubahan yang signifikan pada video. Berdasarkan hal tersebut, bisa disimpulkan proses penyisipan *watermark* ke video bisa berjalan dengan baik.

4.1.5.5 Performa Proses *Extracting Watermark*

Performa proses *extracting* menjelaskan hasil dari proses *extracting* terhadap video yang sudah berisi *watermark*.

Gambar 36 adalah tampilan screenshoot program proses *extracting watermark.*



Gambar 36. Halaman Proses *Extracting Watermark*

4.1.5.6 Hasil Proses *Extracting*

Hasil proses penyisipan menjelaskan perbandingan *watermark* sebelum disisipkan dan *watermark* hasil *extracting* serta perbandingan antara video sebelum dibubuhi *watermark* dan video setelah dibubuhi *watermark*. Hasil dari proses *extracting* adalah file *watermark* yang berupa .txt dan video.

4.1.5.7 Hasil Pengujian Kualitas Video *Watermarked*

Pengujian kualitas video dengan menghitung nilai PSNR dengan persamaan 7:

$$\text{PSNR} = 10 \times \log_{10} = \frac{255^2}{MSE} dB \quad \ldots(7)$$

Pengujian dilakukan dengan 10 file data sampel dan hasil pengujian ditunjukkan pada Tabel 8 :

Tabel 8. Hasil Perhitungan PSNR Video Sebelum dan Sesudah diberi *Watermark*

No	Nama File	Sebelum		Setelah	
		MSE	PSNR(dB)	MSE	PSNR (dB)
1	Coldplay.mp4	870.47	34.12	849.09	32.81
2	Sampel.mp4	890.12	31.22	900.42	30.12
3	Baby.mp4	910.01	36.12	901.31	32.14
4	Satu.mp4	990.21	33.12	920.13	32.02
5	Bola.mp4	1012.20	31.12	1001.01	30.03
6	Tenis.mp4	1012.95	34.62	1009.15	34.11
7	Freeze.mp4	1020.42	35.10	1000.43	32.12
8	Glory.mp4	1021.90	36.12	1020.50	34.74
9	Merdeka.mp4	1041.12	38.25	1021.09	36.98
10	Menari.mp4	1050.40	32.18	1031.29	32.01

Nilai PSNR untuk video yang terbaik adalah antara 30-50 dB. Nilai PSNR yang kurang dari 30 dB mengindikasikan kualitas rendah, dimana distorsi menyebabkan penyisipan terlihat jelas. Nilai PSNR yang lebih dari 50 dB menghasilkan kualitas gambar yang kurang baik. Hasil pengujian yang ditunjukkan Tabel 8 memperlihatkan bahwa video Merdeka.mp4 mempunyai nilai PSNR tertinggi (36.98 dB).

4.1.5.8 Hasil Pengujian Kualitas Audio Visual

Pengujian terhadap kualitas visual dilakukan dengan membandingkan antara video sebelum mendapat *watermark* dan video setelah mendapatkan *watermark.* Pengujian ini dilakukan karena *watermark* yang baik adalah yang tidak merubah isi/tampilan file yang manampungnya, dalam hal ini adalah video. Pengujian ini dilakukan dengan subyektif (*Mean Opinion Square*(MOS)).Penilaian subyektif berkenaan dengan seberapa bagus kualitas suatu gambar dimana kriterianya ditentukan berdasarkan pengamatan mata manusia.

Pengujian MOS dilakukan dengan menyebarkan angket terhadap responden berjumlah 10 orang.Responden tersebut terdiri dari berbagai macam profesi termasuk dari kalangan yang awam dengan obyek pengamatan seperti anak sekolah dan guru.Selain itu responden terdiri dari kalangan yang sudah ahli dan familiar dengan obyek pengamatan seperti mahasiswa informatika atau dosen informatika.Responden tersebut diminta untuk menilai apakah terdapat perbedaan pada video sebelum diberi *watermark* dengan video setelah diberi *watermark*dengan cara melihat 2 buah video tersebut secara bersamaan. Video player yang digunakan adalah K-Lite Media Player Classic dan narasumber melakukan pengamatan dalam keadaan mengetahui lokasi *watermark***.**

Tabel 9. Hasil MOS (*Mean Opinion Square*)

No	Responden / Nama File	1	2	3	4	5	6	7	8	9	10
1	Sample.mp4	√	√	√	√	√	√	√	√	√	√
2	Coldplay.mp4	√	√	√	√	√	√	√	√	√	√
3	Litle.mp4	√	√	√	√	√	√	√	√	√	√
4	Niceface.mp4	√	√	√	√	√	√	√	√	√	√
5	Pict.mp4	√	√	√	√	√	√	√	√	√	√
6	Tom.mp4	√	√	√	√	√	√	√	√	√	√
7	Freeze.mp4	√	√	√	√	√	√	√	√	√	√
8	Glory.mp4	√	√	√	√	√	√	√	√	√	√
9	Cartoon.mp4	√	x	√	x	√	√	x	√	x	x
10	Itunes.mp4	x	√	x	√	x	√	√	x	√	x

Hasil pengamatan ditunjukkan Tabel 9 dimana tanda (√) menunjukkan kualitas visual antara video sebelum berisi *watermark* dan seteleh berisi *watermark* tidak mempunyai perbedaan.Tanda (x) menunjukkan terdapat perbedaan kualitas visual antara video sebelum berisi *watermark* dan seteleh berisi *watermark*. Nomor di kolom responden menunjukkan responden ke 1 sampai responden 10.Hasil dari pengamatan menunjukkan bahwa secara kasat mata tidak ada perubahan antara video yang belum berisi *watermark* dengan video yang telah berisi*watermark* .

4.1.5.9 Hasil Pengujian Serangan Terhadap *Watermark*

Pengujian terhadap *watermark* ini meliputi pengujian terhadap ketahanan *watermark* setelah dilakukan *convert* (Audacity&GoldWave) serta pengujian untuk melihat kesamaan *watermark.*

4.1.5.9.1 *Convert*

Pengujian terhadap ketahanan *watermark* dilakukan terhadap video *watermarked*. Tujuannya adalah untuk mengetahui apakah *watermark* masih tersimpan dengan baik pada video setelah dilakukan proses *convert* menjadi 3 format yang berbeda. Hasil dari pengujian ditunjukkan pada Tabel 10,11 dan 12 :

Tabel 10. Hasil Pengujian Sebelum dan Sesudah Proses *Convert* Menjadi Mpeg-2

No	Nama File	Setalah Dilakukan *Convert* Menjadi Mpeg-2			
		Total *Rate* (kbps)		*Frame Rate* (fps)	
		Sebelum	Sesudah	Sebelum	Sesudah
1	Sample_wtm.mp4	1410	1490	29	29
2	Coldplay_wtm.mp4	570	578	23	23
3	Litle_wtm.mp4	1428	1680	23	23
4	Niceface_wtm.mp4	1510	1980	25	25

5	Pict_wtm.mp4	760	798	23	23
6	Tom_wtm.mp4	1024	1044	32	32
7	Freeze_wtm.mp4	1550	1680	29	29
8	Glory_wtm.mp4	1556	1699	29	29
9	Cartoon_wtm.mp4	1028	1208	23	23
10	Itunes. _wtm mp4	1210	1240	25	25

Tabel 11. Hasil Pengujian Sebelum dan Sesudah Proses *Convert* Menjadi Avi

No	Nama File	Setalah Dilakukan *Convert* Menjadi Avi			
		Total *rate*		*Frame Rate*	
		Sebelum	Sesudah	Sebelum	Sesudah
1	Sample_wtm.mp4	1410	1488	29	29
2	Coldplay_wtm.mp4	570	589	23	23
3	Litle_wtm.mp4	1428	1289	23	23
4	Niceface_wtm.mp4	1510	1790	25	25
5	Pict_wtm.mp4	760	782	23	23
6	Tom_wtm.mp4	1024	1110	32	32
7	Freeze_wtm.mp4	1550	1560	29	29
8	Glory_wtm.mp4	1556	1590	29	29
9	Cartoon_wtm.mp4	1028	1212	23	23
10	Itunes. _wtm mp4	1210	1318	25	25

Tabel 12. Hasil Pengujian Sebelum dan Sesudah Proses *Convert* Menjadi 3gp

No	Nama File	Setalah Dilakukan *Convert* Menjadi 3gp			
		Total *rate*		*Frame Rate*	
		Sebelum	Sesudah	Sebelum	Sesudah
1	Sample_wtm.mp4	1410	1300	29	29
2	Coldplay_wtm.mp4	570	561	23	23
3	Litle_wtm.mp4	1428	1290	23	23
4	Niceface_wtm.mp4	1510	1190	25	25
5	Pict_wtm.mp4	760	760	23	23
6	Tom_wtm.mp4	1024	1010	32	32
7	Freeze_wtm.mp4	1550	1400	29	29
8	Glory_wtm.mp4	1556	1551	29	29
9	Cartoon_wtm.mp4	1028	989	23	23
10	Itunes. _wtm mp4	1210	1202	25	25

Berdasarkan Tabel 10, 11 dan 12 menunjukkan bahwa *watermark* video tersimpan dengan baik setelah dilakukan *convert* menjadi 3 buah format berbedadan dilakukan ekstraksi serta perubahan data file yang terjadi tidak signifikan. Perubahan pada hasil pengujian terdapat pada nilai *bit rate* sedangkan nilai *frame rate* tetap. Perubahan *bit rate* tersebut dikarenakan perbedaan format dari MP4 dengan format yang lainnya adalah dikarenakan teknik coding *rate* tiap format berbeda. Semakin tinggi nilai *bit rate* maka semakin baik kualitas video. Pada video, semakin besar *bit rate*-nya semakin halus pula tampilan videonya. *Frame rate* tidak mengalami perubahan karena pada dasarnya *frame* adalah citra/gambar yang tersusun berurutan sehingga membentuk video, sehingga perubahan format tidak akan merubah jumlah *frame* tersebut.

4.1.5.9.2 *Compression*

Pengujian dengan melakukan *compression* dilakukan pada file video yang berisi *watermark*. Tiap video dirubah nilai total *bit rate*nya. Setelah itu dilakukan *extracting* pada video hasil kompresi tersebut. Hasil dari pengujian tingkat kesamaan *watermark* ditunjukkan pada Tabel 13 :

Tabel 13. Hasil Pengujian Sebelum dan Sesudah Proses *Compression Watermark*

No	Nama File	Setalah Dilakukan *Compression*		
		Total *rate*		Ketahanan *Watermark*
		Sebelum	Sesudah	
1	Sample_wtm.mp4	1410	820	√
2	Coldplay_wtm.mp4	570	256	√
3	Litle_wtm.mp4	1428	728	√
4	Niceface_wtm.mp4	1510	625	√
5	Pict_wtm.mp4	760	360	√
6	Tom_wtm.mp4	1024	560	√
7	Freeze_wtm.mp4	1550	256	√
8	Glory_wtm.mp4	1556	860	√
9	Cartoon_wtm.mp4	1028	500	√
10	Itunes. _wtm mp4	1210	420	√

Tanda √ menunjukkan kecocokan/kesamaan antara *watermark* asal dan*watermark* hasil *extracting*. Dari hasil pengujian dapat disimpulkan bahwa tingkat kesamaan *watermark* asal dan hasil deteksi adalah 100% atau semua sama. Hal ini menunjukkan bahwa penerapan teknik *watermarking* dengan algoritma Koch zhao pada video mp4 tahan terhadap *compression*.

4.1.5.9.3 Pengujian Dengan *Noise gaussian*

Pengujian metode *noise gaussian* dilakukan pada video hasil *watermark*. Tabel 14, 15, 16 dan 17 menunjukkan hasil pengujian *noise Gaussian* dengan besar skala noise,5%, 10% ,40 %,60 dan 75% :

Tabel 14. Tabel Hasil Pengujian *Noise gaussian* Skala 5%

No	Nama File	MSE Awal	MSE Akhir	Selisih	PSNR Awal	PSNR Akhir	Selisih
1	Sample_wtm.mp4	870.47	860.35	-10.12	34.12	34.56	0.44
2	Coldplay_wtm.mp4	890.12	825.12	-65	31.22	36.12	4.9
3	Litle_wtm.mp4	910.01	865.35	-44.66	36.12	33.80	-2.32
4	Niceface_wtm.mp4	990.21	928.50	-61.71	33.12	32.80	-0.32
5	Pict_wtm.mp4	1012.20	1006.36	-5.84	31.12	33.55	2.43
6	Tom_wtm.mp4	1012.95	1010.12	-2.83	34.62	34.52	-0.1
7	Freeze_wtm.mp4	1020.42	995.25	-25.17	35.10	35.60	0.5
8	Glory_wtm.mp4	1021.90	1016.00	-5.9	36.12	36.52	0.4
9	Cartoon_wtm.mp4	1041.12	1009.89	-31.23	38.25	38.80	0.55
10	Itunes. _wtm mp4	1050.40	1010.20	-40.2	32.18	33.20	1.02

Tabel 15. Tabel Hasil Pengujian *Noise gaussian* Skala 10%

No	Nama File	MSE Awal	MSE Akhir	Selisih	PSNR Awal	PSNR Akhir	Selisih
1	Sample_wtm.mp4	870.47	850.30	-20.17	34.12	35.56	1.44
2	Coldplay_wtm.mp4	890.12	885.85	-4.27	31.22	37.14	5.92
3	Litle_wtm.mp4	910.01	856.37	-53.64	36.12	35.40	-0.72
4	Niceface_wtm.mp4	990.21	910.12	-80.09	33.12	33.81	0.69
5	Pict_wtm.mp4	1012.20	1020.75	8.55	31.12	32.57	1.45
6	Tom_wtm.mp4	1012.95	1000.17	-12.78	34.62	35.50	0.88
7	Freeze_wtm.mp4	1020.42	920.20	-100.22	35.10	36.40	1.3
8	Glory_wtm.mp4	1021.90	1001.00	-20.9	36.12	37.82	1.7
9	Cartoon_wtm.mp4	1041.12	1001.89	-39.23	38.25	38.75	0.5
10	Itunes. _wtm mp4	1050.40	1002.56	-47.84	32.18	34.27	2.09

Tabel 16. Tabel Hasil Pengujian *Noise gaussian* 40%

No	Nama File	MSE Awal	MSE Akhir	Selisih	PSNR Awal	PSNR Akhir	Selisih
1	Sample_wtm.mp4	870.47	780.35	-90.12	34.12	36.55	2.43
2	Coldplay_wtm.mp4	890.12	825.12	-65	31.22	36.17	4.95
3	Litle_wtm.mp4	910.01	785.24	-124.77	36.12	34.41	-1.71
4	Niceface_wtm.mp4	990.21	921.40	-68.81	33.12	37.80	4.68
5	Pict_wtm.mp4	1012.20	988.36	-23.84	31.12	33.45	2.33
6	Tom_wtm.mp4	1012.95	989.12	-23.83	34.62	36.40	1.78
7	Freeze_wtm.mp4	1020.42	998.25	-22.17	35.10	35.93	0.83
8	Glory_wtm.mp4	1021.90	1020.70	-1.2	36.12	38.81	2.69
9	Cartoon_wtm.mp4	1041.12	1010.19	-30.93	38.25	39.77	1.52
10	Itunes. _wtm mp4	1050.40	1005.210	-45.19	32.18	33.21	1.03

Tabel 17. Tabel Hasil Pengujian *Noise gaussian* 60%

No	Nama File	MSE Awal	MSE Akhir	Selisih	PSNR Awal	PSNR Akhir	Selisih
1	Sample_wtm.mp4	870.47	813.35	-57.12	34.12	37.86	3.74
2	Coldplay_wtm.mp4	890.12	877.12	-13	31.22	39.15	7.93
3	Litle_wtm.mp4	910.01	899.37	-10.64	36.12	38.44	2.32
4	Niceface_wtm.mp4	990.21	988.57	-1.64	33.12	41.82	8.7
5	Pict_wtm.mp4	1012.20	988.30	-23.9	31.12	44.75	13.63
6	Tom_wtm.mp4	1012.95	1012.14	-0.81	34.62	50.76	16.14
7	Freeze_wtm.mp4	1020.42	989.25	-31.17	35.10	49.34	14.24
8	Glory_wtm.mp4	1021.90	1023.55	1.65	36.12	44.81	8.69
9	Cartoon_wtm.mp4	1041.12	1012.84	-28.28	38.25	48.34	10.09
10	Itunes. _wtm mp4	1050.40	988.70	-61.7	32.18	50.33	18.15

Tabel 18. Tabel Hasil Pengujian *Noise gaussian* 75%

No	Nama File	MSE Awal	MSE Akhir	Selisih	PSNR Awal	PSNR Akhir	Selisih
1	Sample_wtm.mp4	870.47	820.18	-50.29	34.12	50.68	16.56
2	Coldplay_wtm.mp4	890.12	886.15	-3.97	31.22	48.26	17.04
3	Litle_wtm.mp4	910.01	887.08	-22.93	36.12	49.46	13.34
4	Niceface_wtm.mp4	990.21	975.24	-14.97	33.12	50.12	17
5	Pict_wtm.mp4	1012.20	998.00	-14.2	31.12	51.80	20.68
6	Tom_wtm.mp4	1012.95	1004.14	-8.81	34.62	50.14	15.52
7	Freeze_wtm.mp4	1020.42	997.27	-23.15	35.10	49.08	13.98
8	Glory_wtm.mp4	1021.90	1001.05	-20.85	36.12	50.61	14.49
9	Cartoon_wtm.mp4	1041.12	1015.75	-25.37	38.25	50.36	12.11
10	Itunes. _wtm mp4	1050.40	1004.25	-46.15	32.18	52.83	20.65

Berdasarkan hasil pengujian penambahan *noise gaussian* yang ditunjukkan Tabel di atas menunjukkan bahwa berdasarkan Tabel 14 terlihat bahwa penerapan *noise gaussian* memberikan hasil MSE (rata-rata selisih antara MSE awal dan MSE akhir) = -29,266 dan PSNR (rata-rata selisih antara PSNR awal dan PSNR akhir = 0,75 dengan besar *noise* 5%, sedangkan pada Tabel 15 dengan pengujian dengan besar *noise* 10%, memberikan hasil MSE = -37,059 dan PSNR = 1,525. Pada pengujian di Tabel 16 dengan besar *noise* 40% memberikan hasil MSE = -49,586 dan PSNR = 2,463 , pada Tabel 17 dengan besar *noise* 60% memberikan hasil MSE = -22,661 dan PSNR = 10,363 serta terakhir Tabel 18 dengan besar *noise* 75% memberikan hasil MSE = -16,137 dan PSNR = 23,069

Dari hasil pengujian yang ditunjukkan pada Tabel di atas menunjukkan bahwa MSE pada empat macam penambahan noise bernilai negatif sedangkan PSNR keduanya bernilai positif, terjadi penurunan nilai MSE dan peningkatan nilai PSNR yang artinya tidak terjadi perubahan kualitas citra/*image* video dikarenakan kualitas video dikatakan baik bila memiliki nilai PSNR yang tinggi (30-50 dB), sementara dari hasil pengujian setelah penambahan *noise gaussian*, nilai PSNR tidak berkurang justru ada penambahan walaupun hanya sedikit dan tidak signifikan.

4.2 Kesimpulan Hasil Pengujian

Berdasarkan hasil dari pengujian kualitas video dan kualitas *watermark*, maka dapat disimpulkan:

1. Nilai PSNR untuk masing-masing video yang ditanami *watermark* umumnya menurun dibandingkan video asal (sebelum ditanam *watermark*) , akan tetapi penurunan nilai sinyal (dB) rata-rata hanya sekitar 1 – 3 dB sehingga tidak terlalu besar. Karena perubahan nilai PSNR kecil dan nilai akhir masih termasuk tinggi (yang tertinggi Merdeka.mp4: 36,98dB), maka kualitas video tidak banyak terpengaruh.
2. Pengujian terhadap kualitas video yang dilakukan dengan cara Mean Opinion Square (MOS) menghasilkan kesimpulan bahwa video yang sudah ditanami *watermark* memiliki kualitas yang baik. Hal ini berdasarkan hasil pengamatan yang dilakukan terhadap 10 responden.
3. Pengujian terhadap ketahanan *watermark* yang dilakukan dengan *convert* menjadi 3 format yang berbeda menghasilkan kesimpulan bahwa *watermark* tetap terjaga dengan baik. Hal ini dibuktikan dengan dengan hasil deteksi setelah proses *convert* menghasilkan isi *watermark* yang sama.
4. Pengujian dengan *compression* tidak merubah /menghilangkan *watermark*.
5. Pengujian dengan penambahan *noise gaussian* dilakukan dengan skala 5%, 10%, 40%, 60 %, serta 75 %. Hasilnya adalah nilai PSNR berkisar diantara 30 dB – 45 dB kecuali untuk skala 60% dimana ada beberapa video menghasilkan nilai PSNR lebih dari 50 dB. Sedangkan untuk skala 75% rata-rata nilai PSNR sudah lebih dari 50 dB.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil dari pengujian, maka dapat disimpulkan Algoritma Koch Zhao dapat diterapkan sebagai teknik *watermark*ing untuk video dengan format mp4 dengan cukup baik. Hal ini ditunjukkan salah satunya berdasarkan pengamatan visual. Video yang berisi *watermark* secara visual tidak mengalami perubahan dibandingkan dengan sebelum diberi *watermark*.

Video yang sudah ditanami *watermark* menghasilkan nilai PSNR (*Peak Signal Noise Ratio*) dengan nilai terendah adalah 30.18 dB dan tertinggi 36,98 dB. Ini menunjukkan penerapan teknik *watermark*ing pada video dengan algoritma Koch Zhao bisa berjalan dengan baik karena untuk nilai PSNR masih berkisar 30-50 dB. *Watermark* yang ditanam pada video tidak hilang setelah dilakukan proses *convert* dan *compression*. Kualitas video yang berisi *watermark* tersebut baru mengalami

perubahan setelah dilakukan manipulasi file *Noise Gaussian* dengan skala nilai di atas 60%.

5.2 Saran

Berdasarkan hasil atau kesimpulan, berikut ini adalah beberapa saran sebagai pertimbangan untuk penelitian selanjutnya;

1. Mengembangkan aplikasi video *watermarking* supaya dapat diterapkan untuk file video yang berkapasitas besar.
2. Mengembangkan algoritma supaya *watermark* tahan terhadap manipulasi file berupa *Noise Gaussian* dengan skala nilai lebih tinggi.

DAFTAR PUSTAKA

[1] Ardyana & Juarna , 2008 , *Validasi Keaslian Ijazah dan Transkip Nilai Digital dengan Watermarking Menggunakan Dicrete Cosine Transform*, Sekolah Tinggi Teknologi Telkom, Bandung

[2] Baroni, 2008, *Teknologi Watermarking pada Citra Digital*, Sekolah Tinggi Teknologi Telkom, Bandung

[3] Mar, E, Koredianto, U & Magdalena, R, 2009, *Implementasi dan Analisis Video Watermarking dengan Format Video MPEG-4 Berbasis Wavelet Transform.* Makalah disampaikan pada Seminar Nasional Aplikasi Teknologi Informasi. Yogyakarta.

[4] Harjito, B, Potdar, V & Singh, J, 2012, *Watermarking Technique For Wireless Multimedia Sensor Network : A State of The Art Technology*, CUBE International Information Technology Conference, Pune, India

[5] Jain, 1989, *Watermarking of uncompressed and compressed video*. Signal Process., 66: 283-301.

[6] Johnson, F, Zoran, D & Sushi, J, 1998, *Information Hiding: Steganography and Watermarking-Attack and Counter measures.* USA : Kluwer Academic

[7] Langelaar, 2000, *Independent Component Analysis: Algorithms and Aplications*. Neural Network, 13(4-5):411-430,2000

[8] Langham, 1995, *Digital Image Watermarking using Wavelet Transform*. IEEE, International Conference on Consumer Electronics

[9] Nursupangkat & Suhono, 2011, *Watermarking sebagai Teknik Penyembunyian Label Hak Cipta pada Data Digital.* Institut Teknologi Bandung: Bandung

[10] Ortega, D, 2009, *Simulasi Real Time Video Stream Watermarking dengan Menggunakan DWT Discrete Wavelet Transform.* Fakultas Teknik. Universitas Indonesia : Jakarta

[11] Pensiska, F, 2006, *Prinsip Desain untuk Active Audio and Video Fingerprinting*, Institut Teknologi Bandung: Bandung

[12] Persada, F, Pramono, R & Adirama, K, 2005. "*Teknologi watermarking yang Kuat pada Video MPEG",* Institut Teknologi Bandung : Bandung

[13] Zhang, Y & Bi, H, 2011, *Transparent Video Watermarking Exploiting Spatio-Temporal Masking in 3D DCT Domain.* Journal of Computational Information System 7:5.

[14] Zhou, Z, Tao, L & Zhang, C, 2011, *A Video Watermarking Algorithm Based on the Bit-plane Coding*. Journal of Computational Information System 7:6.