**TUGAS AKHIR - KI141502**

# STEGANOGRAFI VIDEO MENGGUNAKAN METODE *MOTION-BASED LEAST SIGNIFICANT FRAME* DAN *BLOCK-BASED DISSIMILAR HISTOGRAM*

**FAUZAN ADIM**
**NRP 5113100101**

Dosen Pembimbing I
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019


**TUGAS AKHIR – KI141502**

# STEGANOGRAFI VIDEO MENGGUNAKAN METODE *MOTION-BASED LEAST SIGNIFICANT FRAME* DAN *BLOCK-BASED DISSIMILAR HISTOGRAM*

**FAUZAN ADIM**
**NRP 5113100101**

**Dosen Pembimbing I**
**Henning Titi Ciptaningtyas, S.Kom., M.Kom.**

**Dosen Pembimbing II**
**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

**Jurusan Teknik Informatika**
**Fakultas Teknologi Informasi**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2019**

***[Halaman ini sengaja dikosongkan]***


**UNDERGRADUATE THESIS – KI141502**

# VIDEO STEGANOGRAPHY USING MOTION-BASED LEAST SIGNIFICANT FRAME AND BLOCK-BASED DISSIMILAR HISTOGRAM

**FAUZAN ADIM**
**NRP 5112100101**

**First Advisor**
**Henning Titi Ciptaningtyas, S.Kom., M.Kom.**

**Second Advisor**
**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

**Department of Informatics**
**Faculty of Information Technology**
**Sepuluh Nopember Institute of Technology**
**Surabaya 2019**

***[Halaman ini sengaja dikosongkan]***

# LEMBAR PENGESAHAN

## STEGANOGRAFI VIDEO MENGGUNAKAN METODE MOTION-BASED LEAST SIGNIFICANT FRAME DAN BLOCK-BASED DISSIMILAR HISTOGRAM

## TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
**FAUZAN ADIM**
**NRP: 5113100101**

Disetujui oleh Pembimbing Tugas Akhir:

1. Hening Titi Ciptaningtyas, S.Kom., M.Kom.
   (NIP. 1984 0708 2010 122 004)

2. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
   (NIP. 1984 1016 2008 121 002)

**SURABAYA**
**JANUARI, 2019**

***[Halaman ini sengaja dikosongkan]***

# STEGANOGRAFI VIDEO MENGGUNAKAN METODE MOTION-BASED LEAST SIGNIFICANT FRAME DAN BLOCK-BASED DISSIMILAR HISTOGRAM

**Nama Mahasiswa : FAUZAN ADIM**
**NRP : 51131001101**
**Jurusan : Teknik Informatika FTIF-ITS**
**Dosen Pembimbing 1 : Henning Titi Ciptaningtyas, S.Kom., M.Kom.**
**Dosen Pembimbing 2 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

## Abstrak

*Steganografi, salah satu teknik penyembunyian data ke dalam suatu media digital, memiliki ragam metode dengan tujuan mendapatkan kapasitas penyimpanan yang besar, robustness yang tinggi serta tingkat keamanan yang tinggi. Penelitian ini menggunakan video sebagai media penyimpanan data rahasia dengan metode Motion-based Least Significant Frame dan Block-based Dissimilar Histogram.*

*Video yang telah dibagi menjadi sejumlah frame dipilih Least Sifnificant Frame-nya, yaitu frame dengan Optical Flow diatas rata-rata Optical Flow seluruh frame. LSF yang kemudian dibagi menjadi sejumlah blok. Blok dengan Histogram Difference diatas rata-rata Histogram Diffirence seluruh blok dipilih untuk disisipkan menggunakan algoritme penyisipan RGB Weighted Coding.*

*Hasil uji coba menunjukkan bahwa kombinasi metode LSF dan BBDH menghasilkan Stego-Video dengan PSNR yang cukup baik (36 dB s.d. 53 dB).*

***Kata kunci: Steganografi, Optical flow, Histogram Warna***

***[Halaman ini sengaja dikosongkan]***

# VIDEO STEGANOGRAPHY USING MOTION-BASED LEAST SIGNIFICANT FRAME AND BLOCK-BASED DISSIMILAR HISTOGRAM


| | |
|---|---|
| **Student's Name** | **: FAUZAN ADIM** |
| **Student's ID** | **: 5113100101** |
| **Department** | **: Teknik Informatika FTIF-ITS** |
| **First Advisor** | **: Henning Titi Ciptaningtyas, S.Kom., M.Kom.** |
| **Second Advisor** | **: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.** |



***Abstract***

*Steganography is a data hiding technique which cover hidden information inside a carrier such as text, images, audio or video. In this research, we use video as the carrier, text message file as the secret message, Motion-based Least Significant Frame for frames selction and Block-based Dissimilar Histogram method for blocks selection and also RGB Weighted Coding algorithm as embedded algorithm.*

*Least significant frames are insignificant frames in a video. LSFs in this case can be altered and do not change the overall quality of all frames. In this research, optical flow is used to help determining the frames which is the least significant. Those LSFs will be divided into blocks. Block-based dissimilar histogram will use those blocks to determine appropriate blocks.*

*Upon all of the appropriate blocks is found, they will be embedded with ASCII code which previously obtained by convertion. The embedding algorithm used in this research is RGB weighted coding which use the last digit of each color channelin each pixels of the appropriate blocks.*

*We obtain 36 dB - 53 dB PSNR values which is quite good.*



***Keywords: Steganography, Optical Flow, Color Histogram.***

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji baga Allah SWT yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan penelitian yang berjudul "Steganografi Video menggunakan Metode *Motion-based Least Significant Frame* dan *Block-based Dissimilar Histogram*". Penelitian ini diharapkan dapat memberikan manfaat dalam pengembangan steganografi terutama steganografi pada video.

Penulis mengucapkan banyak terima kasih kepada:

1. Keluarga yang senantiasa memberikan dukungan kepada penulis.
2. Henning Titi Ciptaningtyas, S.Kom., M.Kom selaku dosen pembimbing I tugas akhir yang memberi ide dan petunjuk.
3. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
4. Bapak Dr. Radityo Anggoro, S.Kom., M.Sc. selaku dosen pembimbing II tugas akhir serta koordinator tugas akhir.
5. Seluruh dosen dan karyawan Teknik Informatika ITS.
6. Keluarga penulis.
7. Pihak-pihak yang selalu membantu, menghibur, menjadi tempat bertukar ilmu yang membuat buku ini selesai.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak memiliki kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca.

Surabaya, Januari 2019

***[Halaman ini sengaja dikosongkan]***

# DAFTAR ISI

# DAFTAR TABEL

***[Halaman ini sengaja dikosongkan]***

# DAFTAR PSEUDOCODE

***[Halaman ini sengaja dikosongkan]***

# DAFTAR GAMBAR

*[Halaman ini sengaja dikosongkan]*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Banyak metode yang dapat dilakukan untuk menyembunyikan data. Tempat data disembunyikan harus aman dari segala kemungkinan kebocoran informasi serta data yang disembunyikan harus dapat diekstrak dengan metode yang telah ditentukan saat dilakukan penyembunyian data.

Salah satu metode penyembunyian data adalah Steganografi. Steganografi dapat menyembunyikan data ke dalam suatu media berupa teks, citra, audio, video serta berkas-berkas unik lainnya. Sebagian besar steganografi pada video, mengimplementasikan steganografi pada citra yang dilakukan pada *video frame*.

Penelitian ini mengusulkan Steganografi dengan metode *Motion-based Least Significant Frame* dilanjutkan dengan *Block-based Dissimilar Histogram*. Di samping itu penelitian ini menggunakan algoritme penyematan *RGB Weighted Coding*. Dengan menggunakan usulan tersebut, diharapkan dapat meningkatkan keamanan serta nilai evaluasi (MSE dan PSNR).

## 1.2 Rumusan Masalah

Penelitian ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana caranya mendapatkan frame dengan metode *motion-based least significant frame*?
2. Bagaimana caranya mendapatkan blok dengan metode *block-Based dissimilar histogram*?
3. Bagaimana caranya menyisipkan data menggunakan algoritme *RGB Weighted Coding*?

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas pada penelitian ini memiliki batasan sebagai berikut:

1. Perangkat Lunak yang digunakan adalah Matlab 2018a

2. Bahasa pemrograman yang digunakan adalah matlab.
3. Masukan untuk media penyimpanan pesan rahasia berupa video dengan ekstensi *.avi* atau *.mp4.*
4. Masukan data untuk penyisipan berupa berkas teks dengan ekstensi *.txt*.

## 1.4 Tujuan

Tujuan penelitian ini adalah sebagai berikut:

1. Melakukan pengamanan data melalui penyisipan pesan rahasia ke dalam berkas video.
2. Melakukan implementasi *motion-based least significant frame* dan block-based dissimilar histogram yang efisien sehingga hasil video steganografi tidak mengalami perubahan yang signifikan.

## 1.5 Manfaat

Manfaat penelitian ini adalah untuk mendapatkan *robustness* yang lebih baik serta meningkatkan keamanan.

## 1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam penelitian ini adalah sebagi berikut:

### 1.6.1 Studi Literatur

Dilakukan pengumpulan data dan studi terhadap sejumlah referensi yang diperlukan dalam penelitian. Referensi berupa sejumlah artikel yang dipublikasikan oleh jurnal dan internet.

### 1.6.2 Analisis dan Desain Perangkat Lunak

Pada tahap ini disusun perancangan sistem steganografi video. Perancangan ini berisi tentang alur kerja sistem dan data yang digunakan. Selain itu, terdapat desain antarmuka dari sistem yang akan dibangun.

### 1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk mengimplementasikan metodemetode yang sudah diajukan pada proposal penelitian. Untuk membangun algoritme yang telah dirancang sebelumnya, implementasi dilakukan dengan menggunakan perangkat lunak Matlab 2018a.

### 1.6.4 Uji Coba dan Evaluasi

Dilakukan pengujian sistem menggunakan parameter yang berbeda. Masukan yang berbeda diuji coba pula untuk menghasilkan data evaluasi yang beragam.

## 1.7 Sistematika Penulisan Laporan

Penelitian ini diharapkan mampu untuk membuat pembaca tertarik. Dengan ketertarikan tersebut, diharapkan penilitian ini dapat dikembangkan lebih lanjut baik oleh penulis maupun pembaca. Secara garis besar, penelitian terdiri atas beberapa bagian seperti berikut:

**Bab I Pendahuluan**

Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan penelitian.

**Bab II Tinjauan Pustaka**

Bab ini berisi kajian teori dari metode dan algoritme yang digunakan dalam menyusun penelitian. Secara garis besar, bab ini berisi tentang video digital, steganografi, *motion-based least significant frame*, *block-based dissimilar histogram*, RGB *coding*, bahasa pemrograman matlab dan *mean square error*dan *peak signal-to-noise ratio.*

**Bab III Perancangan Perangkat Lunak**

Bab ini berisi pembahasan mengenai perancangan dari metode-metode yang diusulkan yang digunakan untuk melakukan

penyembunyian data dan perancangan *GUI* atau antarmuka menggunakan matlab GUIDE.

**Bab IV Implementasi**

Bab ini menjelaskan implementasi yang berupa *pseudocode* dari metode-metode yang diusulkan.

**Bab V Pengujian dan Evaluasi**

Bab ini berisikan hasil uji coba dari metode-metode yang digunakan untuk melakukan penyembunyian data yang sudah diimplementasikan. Uji coba dilakukan dengan menggunakan berkas video yang sesuai dengan batasan masalah. Hasil evaluasi mencakup parameter dan performa dari masing-masing data masukan.

**Bab VI Kesimpulan dan Saran**

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami dalam mengerjakan penelitian, dan saran untuk pengembangan solusi ke depannya.

**Daftar Pustaka**

Daftar Pustaka berisi sejumlah referensi yang dijadikan literatur dalam penelitian.

**Lampiran**

Dalam lampiran terdapat kode program secara keseluruhan.

# BAB II
# TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam penelitian, diantaranya adalah *motion-based least significant frame*, *block-based dissimilar histogram*, *RGB coding* dan beberapa teori lain yang mendukung penyusunan penelitian. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan.

## 2.1 Video Digital.

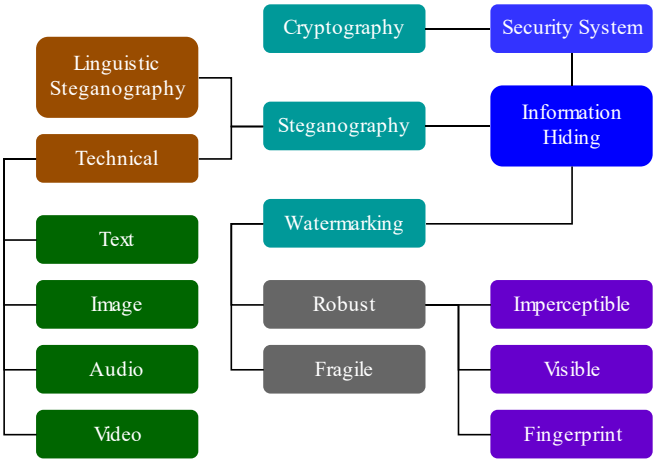
Video Digital adalah video (gambar bergerak) dalam bentuk digital [1]. Penelitian ini menggunakan unsur-unsur yang ada pada video seperti *frame rate*, resolusi video, durasi video dan kedalaman bit (*bit depth*). *Frame rate* adalah jumlah *frame* terpisah yang ditayangkan dalam kurun waktu tertentu. Sering kali diukur dengan satuan *frame* per detik (*frame per second* atau *fps*) [2]. Durasi video dan nilai *frame rate* akan mempengaruhi banyaknya *frame* yang dapat disisipi data rahasia.

*Bit depth* atau *color depth* (kedalaman warna) pada video—*biasanya merujuk pada bit per color*—merupakan jumlah kemungkinan warna tiap piksel di satu saluran warna pada setiap piksel di *frame*. Jadi video dengan *bit depth* 8-bit, tiap saluran warnanya—*yang ada di setiap piksel*—dapat menyimpan $2^8$ atau 256 jenis warna dan mampu menyimpan $2^8 x 2^8 x 2^8$ atau 16.777.216 jenis warna di setiap piksel. Video dengan kedalaman warna 8-bit pada tiap *frame*-nya (citra dengan tiga *color channel*) memiliki kedalaman warna—*dalam hal ini merujuk pada bit per pixel*—24-bit. [3] [4]

Penelitian ini menggunakan video sebagai media tempat disisipkannya data rahasia karena telah banyak digunakannya video secara umum dan kapasitas penyimpanan data (*hiding data capacity*) yang lebih besar jika dibandingkan dengan berkas audio, citra maupun teks. Terlebih, teknik steganografi pada citra dapat pula digunakan pada setiap *frame*.

## 2.2 Steganografi

Steganografi adalah teknik penyembunyian dan pemindahan data melalui media kasat mata (atau bisa juga berupa suara) secara disengaja untuk menyembunyikan keberadaan data rahasia di dalamnya. Hasil steganografi yang hampir mirip dengan media asli membuat berkurangnya kecurigaan—*dibandingkan dengan hasil dari kriptografi*—akan adanya pesan rahasia yang disisipkan sehingga pihak-pihak tak diinginkan yang ingin membaca pesan




**Gambar 2.1** Klasifikasi teknik penyembunyian Data **[42]**

rahasia juga berkurang. [5]

## 2.3 Motion-based Least Significant Frame

*Least Significant Frame* (LSF) adalah *frame* yang paling 'tidak signifikan'. *Motion-based* LSF adalah LSF yang diperoleh melalui komputasi aliran optis (*optical flow*) . Dari perolehan nilai *optical flow*, didapatkan sejumlah LSF yang memiliki *optical flow* di atas rata-rata *optical flow* seluruh *frame*. [6]

*Optical flow* adalah distribusi kecepatan objek yang terlihat pada *frame* (*images*). Dengan mengestimasi *optical flow* antar-*frame*, kecepatan objek di dalam video dapat dikalkulasikan. Pada umumnya, dengan kecepatan gerak yang sama, objek yang lebih dekat dengan kamera—*tampak lebih besar*—akan menghasilkan pergerakan yang lebih jelas dibanding objek yang lebih jauh. [7] [8] [9]

Terdapat banyak metode untuk memperoleh *optical flow*. Di antaranya adalah *Horn-Schunck.* Penelitian ini menggunakan algoritme *Horn-Schunck* karena mampu menghasilkan *flow vectors* yang lebih pekat dibdandingkan dengan algoritme *Lucas-Kanade*. Sehingga lebih mudah dalam proses perolehan *optical flow*-nya. [10]

Metode Horn-Schunck, yang dapat dipelajari lebih lanjut dari penelitian berjudul "*Determining Optical Flow*" (1981) oleh Horn dan Schunck, menggambarkan kehalusan (*smoothness*) dalam (*flow*) aliran di seluruh frame. Metode ini mencoba meminimalisasi distorsi dalam arus dan memilih solusi dengan kehalusan yang lebih. Untuk frame dua dimensi:

$$E = \iint \left[ (I_x u + I_y v + I_t)^2 + \alpha^2 (|\nabla_u|^2 + |\nabla_v|^2) \right] \mathrm{d}x\mathrm{d}y \quad (1)$$

Pada persamaan (1), $I_x, I_y$ dan $I_t$adalah turunan nilai intensitas *frame* sepanjang dimensi *x, y* dan *t* masing-masing.

$$\vec{V} = [u(x,y), v(x,y)]^{\mathrm{T}} \quad (2)$$

Persamaan (2) adalah *optical flow vector*. Sementara itu, $\alpha$ adalah suatu konstanta yang makin besar nilainya, makin besar kehalusan alirannya.

Dari persamaan sebelumnya, dapat pula kita sederhanakan menjadi persamaan (3) di bawah ini:

$$\begin{aligned}(I_x^2 + \alpha^2)u + I_x I_y v &= \alpha^2 \bar{u} - I_x I_t \\ \left(I_y^2 + \alpha^2\right)v + I_x I_y u &= \alpha^2 \bar{v} - I_y I_t\end{aligned} \tag{3}$$

Namun, karena persamaan tersebut sangat tergantung dengan nilai *flow* tetangga, perulangan harus dilakukan saat kondisi tetangga telah diperbarui. Untuk itu:

$$\begin{aligned}u^{k+1} &= \bar{u}^k - \frac{I_x\left(I_x \bar{u}^k + I_y \bar{u}^k + I_t\right)}{\alpha^2 + I_x^2 + I_y^2} \\ v^{k+1} &= \bar{v}^k - \frac{I_x(I_x \bar{u}^k + I_y u^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}\end{aligned} \tag{4}$$

Pada persamaan (4), notasi $k + 1$ yang merupakan iterasi selanjutnya dan $k$ yang merupakan hasil akhir. [11]

$$r_{x,y} = \sqrt{{u_{x,y}}^2 + {u_{x,y}}^2} \tag{5}$$

Pada persamaan (5) di atas, nilai *optical flow* tiap *frame* terurut (*concecutive frame*) yang diperoleh berwujud komponen kompleks vektor horizontal ($u_{x,y}$) dan vektor vertikal ($v_{x,y}$). Kemudian dilakukan perhitungan vektor diagonal ($r_{x,y}$) yang diperoleh dengan akar kuadrat jumlah vektor horizontal dan vertikal.

$$\vec{r}_t = \frac{\sum_{x=1}^{w}\sum_{y=1}^{h}(r_{x,y})}{w \times h} \quad (6)$$

Kemudian dilakukan perhitungan rata-rata panjang vektor diagonal seluruh piksel di tiap *concecutive frame* ($\vec{r}_t$) sesuai persamaan (6).

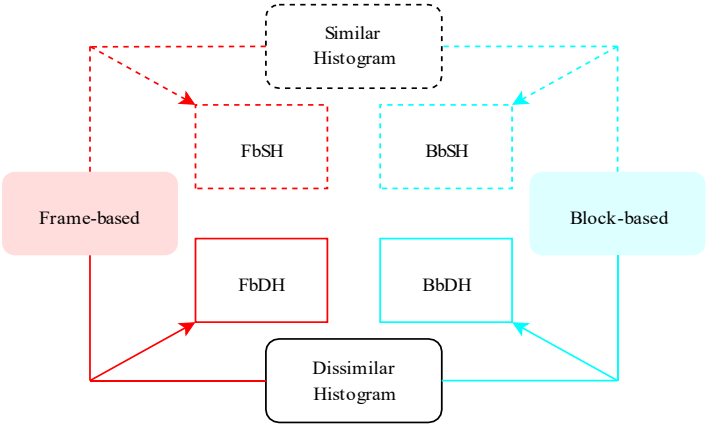
$$\vec{r}_{tot} = \frac{\sum_{t=k}^{N}(\vec{r}_t)}{N} \quad (7)$$

Kemudian dilakukan penjumlahan seluruh nilai rata-rata panjang vektor diagonal pada setiap *frame* ( $\sum_{t=k}^{N}(\vec{r}_t)$ ) lalu dibagi dengan jumlah *frame* (*N*) untuk memperoleh nilai rata-rata panjang vektor *r* seluruh *concecutive frame* ($\vec{r}_{tot}$) sesuai persamaan (7). Didapat LSF yakni *concecutive frame* dengan $\vec{r}_t \geq \vec{r}_{tot}$. [6].

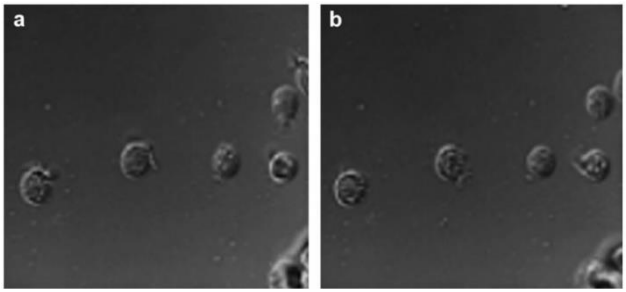
## 2.4 Block-Based Dissimilar Histogram

*Block-based dissimilar histogram* merupakan satu dari empat metode pencarian piksel sesuai (*appropriate pixel*) yang berdasar pada histogram warna yang diusulkan oleh *Cetin. O* [12]. Metode *similar histogram* serta *dissimilar histogram* diaplikasikan untuk 'histogram pada *frame*' dan 'histogram pada blok' sehingga diperoleh empat metode yaitu *frame-based similar histogram* (FbSH), *block-based similar histogram* (BbSH), *frame-based dissimilar histogram* (FbDH) dan *block-based dissimilar histogram* (BbDH). Diilustrasikan pada Gambar 2.2.

*Cetin. O.* menyimpulkan bahwa hasil *dissimilar histogram* pada pendekatan blok (BbDH) lebih baik dari pada pendekatan *frame* (FbDH). Begitu pula dengan hasil *similar histogram* pada pendekatan *frame* (FbSH) yang lebih baik dari pada pendekatan blok (BbSH). [12]



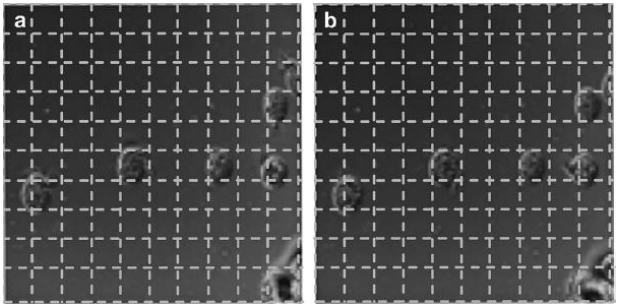

**Gambar 2.2** Metode berdasarkan histogram warna **[12]**




**Gambar 2.3** *Frame* terurut *100 x 100* piksel **[12] :**

a) *Frame* Pertama b) *Frame* Kedua

*Frame* terurut (Gambar 2.3) akan dipecah menjadi blok (Gambar 2.4). Blok tetangga adalah blok yang bersebelahan (beda letak) dalam satu *frame*. Sementara blok terurut adalah blok sama letak, beda *frame* (*frame* selanjutnya).




**Gambar 2.4** Blok *10 x 10* piksel pada *frame* terurut **[12] :**

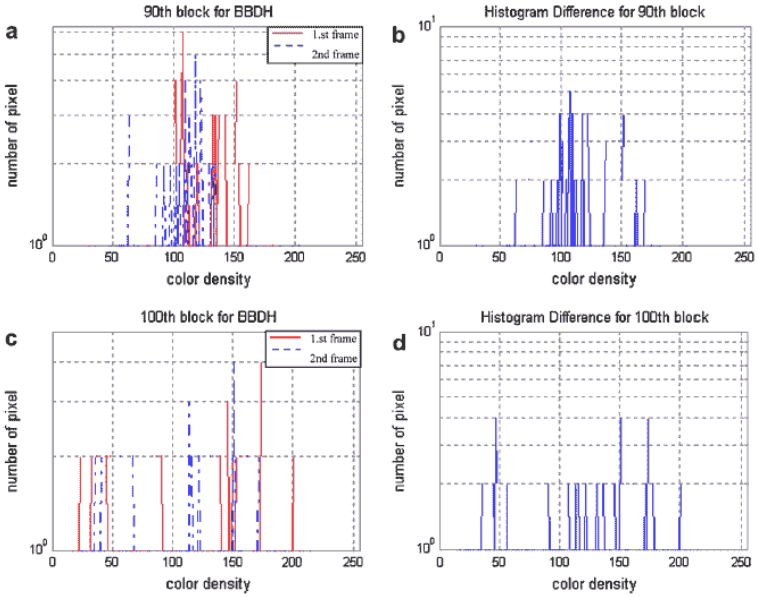
a) *Frame* Pertama b) *Frame* Kedua

Dalam penelitian ini histogram warna (*hist.*) setiap blok di setiap *frame* (LSF) akan dihitung. Setelah *hist.* pada setiap blok diperoleh, dilakukan perhitungan perbedaan histogram warna (*hist. diff.*) di setiap blok terurut. Perolehan *hist.diff.* dapat dilihat pada persamaan (4). Untuk contoh penggunaannya terdapat pada Gambar 2.5.

$$D(i,i+1) = \sum_{j=0}^{255} |H_i(j) - H_{i+1}(j)| \quad (4)$$

Pada persamaan (5), $j$ adalah *color density* dan $i$ adalah *current frame* (jika *frame-based*) atau *current block* (jika *block-based*). [13]

Setelah *hist. diff.* seluruh blok terurut terhitung, dilakukan perhitungan rata-rata *hist. diff.* keseluruhan. Nilai rata-rata tersebut dijadikan sebagai HCV (*histogram constant value*). Blok dengan nilai *hist. diff.* lebih besar dari HCV dipilih sebagai tempat disisipkannya data rahasia.




**Gambar 2.5** Grafik Histogram Blok yang Bersangkutan **[12]**

(a) Grafik Histogram blok #90,
(b) Grafik Hist-Diff blok #90,
(c) Grafik Histogram blok #100,
(d) Grafik Hist-Diff blok #100.

## 2.5 RGB Weighted Coding

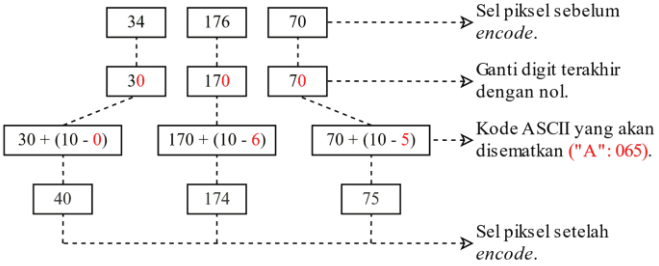
Algoritme RGB *weighted coding* menggunakan bobot RGB dari *frame* piksel sebagai media penyisipan informasi rahasia. Algoritme ini hanya dapat digunakan jika piksel memiliki tiga saluran warna dengan *color depth* 24-bit. Algoritme RGBW memungkinkan penyisipan kode ASCII—*hasil konversi karakter dari informasi rahasia*—ke piksel RGB—*digit terakhir tiap color channel*—. Dipilihnya algoritme ini karena suatu *frame* dengan *M x N* piksel dapat menyimpan data rahasia berukuran paling besar *M x N* bita. Dapat diakatakan tiap satu piksel mampu menyimpan 1 bita (8 bit) data rahasia. [14]

ASCII adalah kependekan dari *American Standard Code for Information and Interchange*. ASCII *Code* adalah *character encoding* berukuran 7-bit yang dipakai komputer untuk berkomunikasi dengan manusia. Komputer mengkonversikan karakter ke bentuk angka, kemudian angka-angka tersebut dikonversikan ke bentuk biner. Tiga puluh dua karakter pertama kode ASCII (0 s.d. 31) merupakan *control characters* yang tidak dapat ditampilkan namun berfungsi untuk mengontrol perangkat keras seperti mesin cetak (*printer*). Sisanya, sejumlah sembilan puluh enam karakter—*indeks 32-127*—adalah *printable characters* yang berupa angka, huruf kecil, huruf kapital, tanda baca dan simbol yang semuanya dapat ditemukan di papan tombol (*keyboard*). [15] [16]

RGB, kependekan dari *Red Green Blue*, adalah tiga warna cahaya yang dapat digabung untuk membentuk segala warna. Jika intensitas tertinggi setiap warna dicampur maka akan terbentuk cahaya putih. Jika intensitas setiap warna yang dicampur diatur menjadi nol maka hasilnya adalah hitam. Dalam menampilkan gambar, peralatan elektronik, seperti televisi, layar monitor menggunakan RGB. Sementara itu mesin cetak (*printer*) menggunakan CMYK (*cyan, magenta, yellow, key—black—*). RGB yang memiliki 8-bit bps (*bit per samle*)—*istilah umum pada video*—atau bpc (*bit per color*)—*istilah yang digunakan oleh Adobe*—dapat membuat $2^8 x 2^8 x 2^8 = 16.777.216 = 2^{24}$ varian

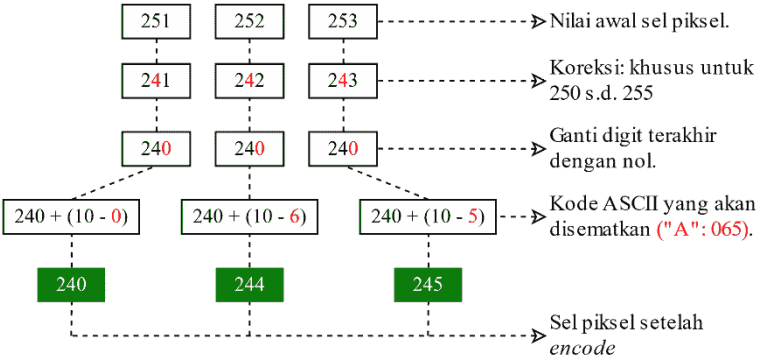
warna. Nilai $2^{24}$atau 24-bit tersebut—*disebut pula sebagai true color*—adalah kedalaman warna atau *bit-depth—istiliah umum pada citra—* atau bpp (*bit per pixel*). [17] [18] [19]

Algoritme RGB *Weighted Coding* mampu mengolah *frame* video dengan kedalaman warna hingga 8-bit *per sample* dan komponen warnanya harus ada tiga. Jadi meski video memiliki bps rendah—*bahkan 4-bit bps*—, proses penyisipan akan tetap berlangsung. Sebaliknya, *frame* di atas *true color* (24-bit), yaitu *deep color* (30/36/48-bit) atau *frame* yang hanya memiliki satu *color channel* tidak akan diproses. Algoritme ini tidak mampu mengkonversi karakter yang bukan ASCII 7-bit. Jadi apabila berkas teks berisi karakter yang tidak termasuk ASCII 7-bit, konversi karakter tersebut tidak dilakukan yang beakibat hilangnya karakter tersebut.




**Gambar 2.6** Encode: RGB(34-176-70) dan 'A'(0-6-5) **[14].**

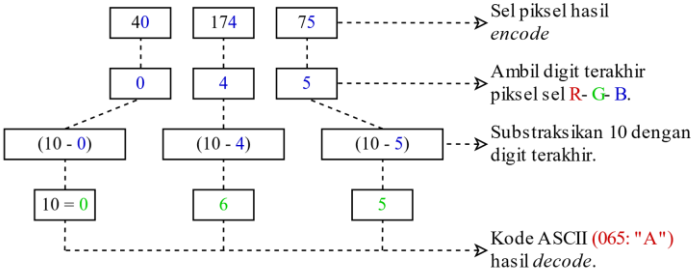
Algoritme RGB *Weighted Coding* membaca bobot RGB pada sel piksel. Sebagai contoh pada Gambar 2.6, sel piksel RGB(34,176,70) akan disisipkan karakter A (kode ASCII 65). Digit terakhir sel piksel akan diubah menjadi nol. Kemudian kode ASCII yang disubtraksikan dengan 10, akan disisipkan ke sel piksel dengan ketentuan digit pertama ke R, digit kedua ke G dan digit ketiga ke B.




**Gambar 2.7** Encode: RGB(251-252-253) dan 'A'(0-6-5) **[14].**

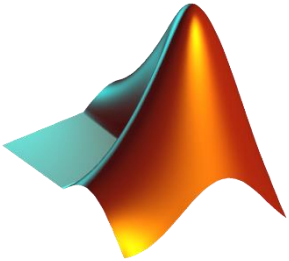
Berbeda jika nilai bobot RGB di atas 249. Sebagai contoh pada Gambar 2.7, sel piksel RGB(251,252,253) yang akan disisipi oleh karakter ‘A’ (kode ASCII 65). Bobot di antara 250 hingga 255 akan secara otomatis diubah menjadi 240. Hal ini disebabkan apabila hasil *encode* memiliki bobot sel di atas 255, sel piksel tersebut akan menghasilkan karakter yang berbeda dengan aslinya saat dilakukan *decode*. Setelah bobot disesuaikan menjadi 240, proses selanjutnya masih sama seperti pada Gambar 2.6.

Jika ingin membaca pesan rahasia, maka proses pengembaliannya dapat dilakukan sesuai dengan Gambar 2.8.




**Gambar 2.8** Decode: RGB(40-174-75) hasilnya 'A'(0-6-5) **[14].**

## 2.6 Matlab



**Gambar 2.9** Logo Matlab **[41]**

Matlab adalah peranti lunak pengembangan pemrograman (dikenal sebagai: *Integrated Development Environment* atau IDE) dengan fasilitas dan fitur untuk pemrosesan data menggunakan angka-angka. Matlab dikembangkan oleh Math Works. Matlab juga dikenal sebagai bahasa pemrograman yang paling banyak digunakan oleh matematikawan atau ilmuwan. Penelitian ini menggunakan Matlab versi R2018a—*yang dirilis oleh Mathworks pada tanggal 15 Maret, 2018*—karena fitur di dalam Matlab sangat lengkap untuk topik yang dimuat dalam penelitian ini. [20] [21]

## 2.7 Mean Squared Error

Untuk mengevaluasi kualitas citra/*frame* terubah kita dapat menghitung nilai rata-rata kuadrat eror. Perhitungan MSE (pada *grayscaled image*) dapat dilihat pada persamaan di bawah.

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \qquad (1)$$

Pada penelitian ini *frame* yang digunakan memiliki ukuran 8-bit per piksel atau $20 \log 10(255) = 46$ dB. Maka nilai MSE 0.224. Makin kecil nilai MSE, makin baik kualitas citra berkehilangan (*lossy images*). Hasil kalkulasi MSE digunakan untuk perolehan PSNR. [22]

## 2.8 Peak signal-to-noise Ratio

PSNR adalah suatu ukuran standar dalam kompresi citra. PSNR telah umum digunakan dalam mengukur kualitas citra terkompresi.

PSNR dapat digunakan untuk melambangkan beda antara dua video. Sehingga persamaan ini dipakai dalam penelitian ini untuk menghitung perbedaan video sebelum dan setelah disisipi pesan rahasia. *Peak signal-to-noise ratio* dapat dihitung menggunakan persamaan di bawah.

$$PSNR = \log_{10}(\frac{MAX_f}{\sqrt{MSE}}) \quad (2)$$

Nilai PSNR maksimal pada piksel dengan kedalaman 7-bit adalah 54 dB. Umumnya nilai *PSNR* pada citra berkehilangan (*lossy images*) dan kompresi video berkisar antara 30 dB hingga 50 dB. Makin tinggi angka PSNR, makin baik kualitas citra berkehilangan [23] [24] [25].

## 2.9 AES Crypt

Perangkat lunak *open source* pengenkripsi, yang dapat digunakan pada berbagai sistem operasi, yang menggunakan *Advanced Encryption Standard* (AES) untuk memudahkan pengenkripsian berkas. *Software* ini tergolong mudah digunakan dan tidak memerlukan *compiler*. AES *Crypt* hanya memerlukan Visual C+ untuk sistem operasi Windows 10. [26]

## 2.10 VLC Media Player

VLC *media player* adalah *multimedia player* untuk ragam macam format audio dan video (MPEG, DivX/Xvid, Ogg, dll) dan juga DVD, VCD dan macam-macam streaming protocols. Dalam penelitian ini, VLC digunakan untuk mengedit *metadata cover video* pada saat proses *encode* dan membaca *metadata* pada saat proses *decode*. Fiturnya yang mudah digunakan membuat VLC terkenal bagi para penggunanya. Disamping itu mengedit *metadata* di VLC sangat mudah. Satu lagi keunggulan VLC yaitu gratis. [27]

*[Halaman ini sengaja dikosongkan]*

# BAB III
# PERANCANGAN PERANGKAT LUNAK

Bab ini membahas tentang perancangan sistem perangkat lunak yang dibuat untuk mencapai tujuan dari penelitian. Perangkat lunak yang dirancang untuk penelitian ini adalah perangkat lunak penyisipan pesan tersembunyi pada berkas video. Perancangan perangkat lunak meliputi perancangan perangkat secara umum, perancangan data masukan dan data keluaran, dan perancangan antar muka.

Berikut ini daftar istilah yang telah digunakan dalam penelitian ini beserta penjelasannya:

| Istilah | Detail Informasi |
|---|---|
| Input (Masukan) | n hasil memasukkan; apa yang dimasukkan: [28] |
| *Output* (Keluaran) | n hasil mengeluarkan (menerbitkan, dan sebagainya): [28] |
| *Frame* | sebuah citra dalam kumpulan citra terurut di suatu video [29] |
| *Block* (Blok) | *frame* yang dipecah dengan ukuran tertentu [30] |
| *Character Encoding* | daftar nomor yang merepresentasikan sejumlah karakter [31] |
| LSF | Singkatan dari *Least Significant Frame* [6] |
| DB | Singkatan dari *Dissimilar Block* [12] |
| ASCII *Code* | ASCII: singkatan dari *American Standard Code for Information Interchange*<br>ASCII *Code* merupakan satu dari sekian banyak *character encoding* yang ada [15] |
| RGB (*Red*, *Green*, *Blue*) | salah satu dari sekian banyak ragam sistem warna atau model warna yang ada [18] |

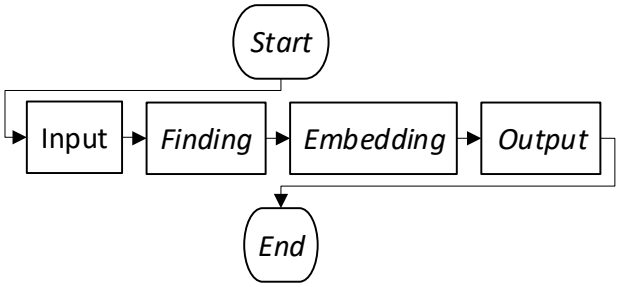
| | |
|---|---|
| *Weighted* | bobot untuk memberi nilai yang lebih pada suatu atribut dibanding atribut lain [32] |
| Algoritme | satu set instruksi yang didesain untuk menjalankan tugas tertentu [33] |
| *Optical Flow* | pergerakan yang tampak dari pola kecerahan yang diamati pada saat pengamat bergerak relatif ke objek yang sedang diamati [11] |
| *Color Histogram* | sebuah representasi dari distribusi warna dalam suatu citra yang diperoleh dengan menghitung banyak piksel di setiap rentang warna [34] |
| PSNR | suatu metode yang sering digunakan untuk mengukur kualitas rekonstruksi pada citra terkompresi [35] |
| *Pseudocode* | suatu cara untuk menyampaikan fungsi program secara umum [36] |
| *Hiding Data Capacity* | banyaknya data yang dapat disimpan per satu *byte*-nya [12] |
| *Robustness* | dalam steganografi *robustness* dapat diartikan sebagai metode yang diusulkan bagus (nilai PSNR yang tinggi) dan kuat (aman) [12] |
| *Color Depth* | jumlah warna yang dapat disimpan di dalam sebuah piksel yang ada pada suatu citra [19] |
| *Bit Depth* | jumlah bit yang dapat disimpan di dalam sebuah piksel [37] |
| *Color Channel* | suatu tempat untuk menyimpan informasi dari salah satu komponen warna primer dari suatu model warna [38] |
| *Cover Video* | video sebagai materi untuk *encode* [39] |
| *Stego Video* | video sebagai materi untuk *decode* [39] |

## 3.1 Deskripsi Umum Sistem

Perangkat lunak terdiri atas dua proses, yaitu proses *encode* dan proses *decode*.

### 3.1.1 Encode

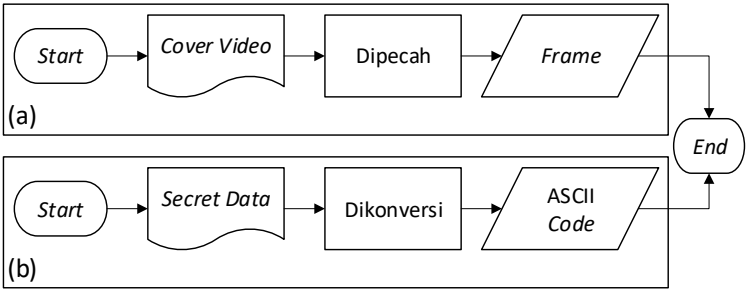
Terdapat empat bagian dalam tahap *encode* (Gambar 3.1) yaitu Input, *Finding*, *Embedding* dan Output.



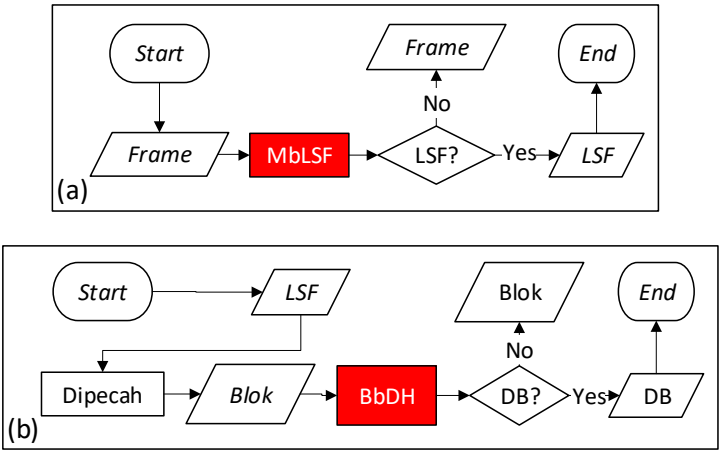

**Gambar 3.1** Proses *Encode*

Di bagian Input (Gambar 3.2), *cover video* akan dibaca untuk memperoleh *frame rate*, durasi, *bit depth* dan dimensi. Jumlah *frame* diperoleh dengan perkalian antara durasi dan *frame rate*. Tiap frame memiliki dimensi yang sama. Kedalaman warna pada *frame* juga diambil berdasarkan kedalaman warna pada video. Sementara itu, *Secret data* akan dikonversikan menjadi kode ASCII. *Frame* dan kode ASCII akan digunakan sebagai input untuk *Finding* (Pencarian tempat yang sesuai) (Gambar 3.3).

Pada bagian *Finding* (Gambar 3.3 (a) dan Gambar 3.3 (b)), terdapat dua metode yang digunakan. Metode pertama adalah *motion-based leaast significant frame* (Gambar 3.3 (a)) dengan input *cover frames* yang hasilnya adalah LSF beserta indeksnya. Kemudian setiap LSF dipecah menjadi *cover blocks*. *Cover frame* yang bukan LSF sudah tidak lagi digunakan kecuali untuk pembentukan *stego video* dengan penyatuan seluruh *cover frame*.

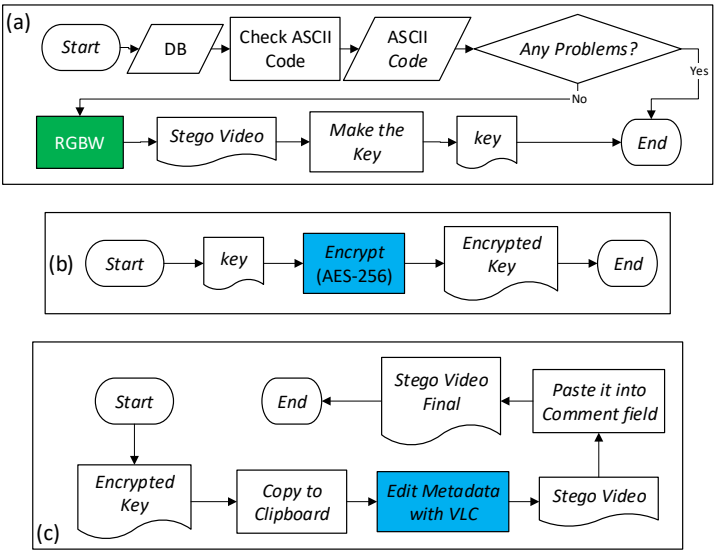



**Gambar 3.2** *Encode*: Input
(a) Input *Cover Video* (b) Input *Secret Data*




**Gambar 3.3** *Encode*: *Finding*
(a) *Find* LSF (b) *Find* DB

Setelah itu dilanjutkan dengan metode *block-based dissimilar histogram* (Gambar 3.3 (b)) yang menggunakan *cover blocks* sebagai input. Hasil BbDH adalah blok terpilih (*appropriate blocks*) beserta indeksnya. Blok yang tidak terpilih sudah tidak lagi digunakan kecuali untuk pembentukan *frame* dengan penyatuan seluruh blok pada LSF.




**Gambar 3.4** *Encode: Embedding*

(a) RGB Weighted Coding
(b) *Encrypt key* (*.mat file*)
(c) *Put Encrypted Key into Metadata*

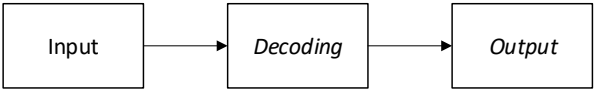
Proses selanjutnya yaitu *Embedding*. Proses ini terdiri dari RGBW (Gambar 3.4 (a)) kemudian dilanjutkan dengan enkripsi (Gambar 3.4 (b)) dan diakhiri dengan pengeditan *metadata* (Gambar 3.4 (c)).

*Dissimilar blocks* yang diperoleh kemudian disisipi dengan kode ASCII menggunakan RGB *Weighted Coding*. Hasil tahap pertama berupa *stego video* dan *key*. Pada tahap selanjutnya, *key*

akan dienkripsi menggunakan AES-256. Pada tahap terakhir, isi berkas *encrypted key* akan di-*copy* untuk di-*paste*-kan pada *comment field metadata* pada *stego video* dengan VLC.

### 3.1.2 Decode

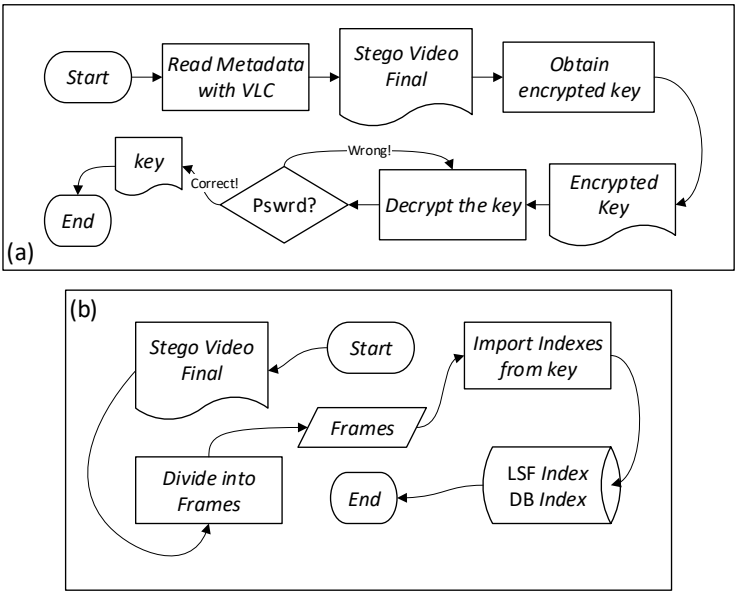
Ada tiga bagiandalam tahap *Decode* (Gambar 3.5) yaitu Input, *Decoding* dan Output.




**Gambar 3.5** Proses *Decode*

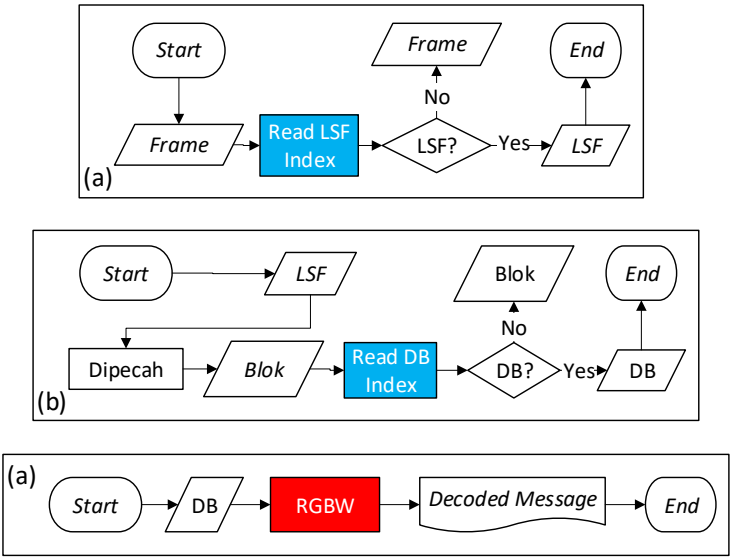
Diawali dengan tahap Input yang memiliki dua bagian. Pertama (Gambar 3.6 (a)), *Encrypted key* yang ada di dalam *comment field metadata* akan di-*decrypt*. Kedua (Gambar 3.6 (b)), *key* dan *stego video* akan dibaca. *Stego video* akan dipecah menjadi sejumlah *frame*. Sedangkan *key* akan diimpor agar diperoleh indeks LSF dan indeks DB.

Pada bagian *Decoding* (Gambar 3.7), setiap LSF—*cover frame dengan indeks LSF*—akan dipecah menjadi *blocks*. *Frame* yang bukan LSF tidak lagi digunakan. Setiap DB akan diproses menggunakan RGB *weighted coding*. *Blocks* yang bukan DB tidak lagi digunakan. Hasil RGBW berupa kode ASCII. Setiap kode ASCII akan dikonversikan menjadi karakter. Seluruh karakter hasil konversi tersebut akan ditulis dalam berkas teks.




**Gambar 3.6** *Decode*: Input

(a) *Obtain key* (b) *Obtain indexes*




**Gambar 3.7** *Decode*: *Decoding*

## 3.2 Perancangan Data

### 3.2.1 Data Masukan

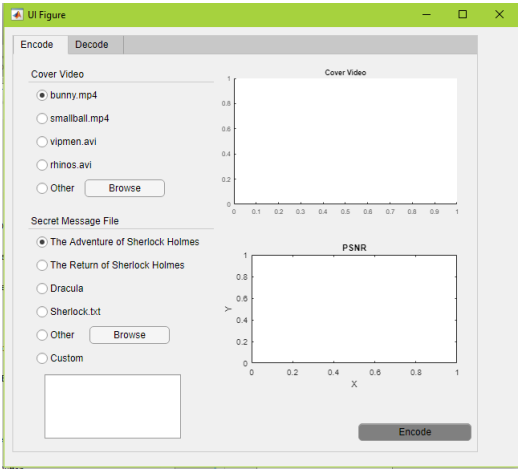
Input data pada proses *encode* adalah berkas *cover video* berekstensi *.avi* atau *.mp4* dan berkas teks berekstensi *.txt* berisi pesan rahasia. Sedangkan untuk proses *decode* menggunakan berkas *stego video* dan berkas indeks berekstensi *.mat*.

### 3.2.2 Data Keluaran

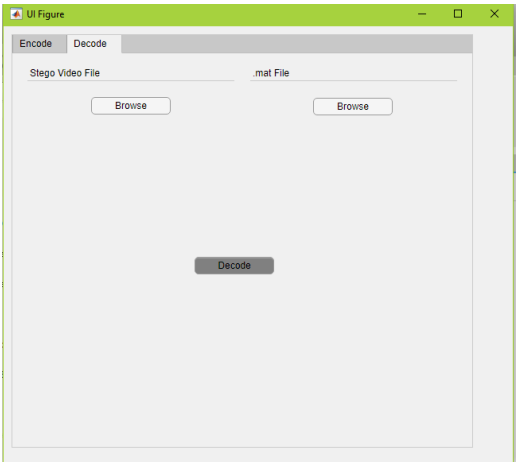
Output pada proses *encode* adalah *stego video* dan berkas berekstensi *.mat*. Sementara untuk proses *decode* berupa berkas teks berekstensi *.txt*.

## 3.3 Perancangan Antarmuka

Antarmuka perangkat lunak dibuat agar memudahkan pengguna menggunakan perangkat lunak. Desain antarmuka dibuat menggunakan *GUIDE* yang merupakan salah satu fasilitas penunjang perancangan sistem antar muka dari Matlab.




**Gambar 3.8** Antar Muka *Encode*




**Gambar 3.9** Antar Muka *Decode*

# BAB IV
# IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Imple-mentasi berupa pseudocode untuk membangun perangkat lunak.

## 4.1 Lingkungan Implementasi

Implementasi penyisipan pesan tersembunyi pada berkas audio dilakukan pada lingkungan implementasi yang ditunjukkan pada Tabel 4.1.

**Tabel 4.1** Lingkungan Implementasi Perangkat Lunak

| Perangkat | Jenis Perangkat | Spesifikasi |
|---|---|---|
| **Perangkat Keras** | Prosesor | Intel® Core™ i5 3230M 2.60 GHZ |
| | Memori | 4 GB 1333 MHz DDR3 |
| **Perangkat Lunak** | Sistem Operasi | Windows 10 Pro |
| | Perangkat Pengembang | Matlab R2018a |

## 4.2 Implementasi

Sub bab ini menjelaskan pembangunan perangkat lunak berdasarkan bab perancangan perang-kat lunak. *Pseudocode* yang bersesuaian dengan bab perancangan perangkat lunak akan ditampilkan pada bab ini.

### 4.2.1 Preprocessing

Pada tahap ini, video akan dibaca seperti halnya pada tahap Input di *Encode* dan *Decode* pada bab perancangan perangkat lunak. Informasi-informasi pada video yang terbaca, baik resolusi pada *frame*, jumlah *frame*, durasi, fps, jumlah *color channel* dan kedalaman warna adalah variabel-variabel penting yang diperoleh dari tahap ini. Video dipecah menjadi sejumlah *frame*. Setiap *frame* beserta variabel-variabel yang disebutkan sebelumnya akan disimpan pada suatu *array* variabel.

Tiap LSF akan dipecah menjadi *block* dalam variabel *array*. Tiap *block* memiliki resolusi yang sama dengan *block* lain. Tahap ini merupakan tahap penting sebelum dilakukannya proses *encoding* atau *decoding*. Detail tahap ini dapat dilihat pada *Pseudocode* 4.1 dan *Pseudocode* 4.2

#### 4.2.1.1 Fungsi *getFrame*

Fungsi ini membagi video menjadi sejumlah *frame*. Detail fungsi ini dapat dilihat pada *Pseudocode* 4.1

```
     FUNCTION getFrame(video)
       INPUTS        : video file
       OUTPUTS       : frames

1      FOR i = 1 to N DO //      N : number of frame
2        frames (i) = readFrames(video);
3      ENDFOR
     ENDFUNCTION
```

***Pseudocode* 4.1** Fungsi *getFrame*

#### 4.2.1.2 Fungsi getBlocks

Fungsi ini membagi *frame* menjadi blok dengan ukuran yang menyesuaikan dimensi *frame*. Tiap blok memiliki ukuran ukuran yang sama dengan blok lain di seluruh *frame*. Detail fungsi ini dapat dilihat pada *Pseudocode* 4.2.

```
    FUNCTION getBlocks(frames)
      INPUTS        : frames
      OUTPUTS       : blocks

      FOR i = 12 to 2 DO //Adjust blocks' height
1      IF frames.height % i IS == 0 THEN
2        blocks.height = i;
3        BREAK
       ENDIF
      ENDFOR

4     FOR i=15 to 2 DO //Adjust blocks' width
5      IF frames.width % i == 0 THEN
6        blocks.width = i;
         BREAK
       ENDIF
      ENDFOR

7     Initialize h, w, n;
      h = (frames.height / blocks.height)
      w = (frames.width / blocks.width)
      n = h x w //n as number of possible blocks

8     FOR EACH frames DO
9       ASSIGN blocks with  h as blocks.height;
10      ASSIGN blocks with  w as blocks.width;
      ENDFOR
    ENDFUNCTION
```

***Pseudocode* 4.2** Fungsi *getBlocks*

#### 4.2.1.3 Fungsi *msgConvert*

*Pseudocode* 4.3 membaca *secret data* dan mengonversikan tiap karakternya menjadi kode ASCII.

```
   FUNCTION msgConvert(secretFile)
     INPUTS          : secret file
     OUTPUTS         : ascii

1    FOR each character in secretFile
2      FOR i = 1 to 3                    // 1/2/3 = red/green/blue
3          ascii(i) = character(digit(i));
       ENDFOR
     ENDFOR
   ENDFUNCTION
```

***Pseudocode* 4.3** Fungsi *msgConvert*

### 4.2.2 Encoding

*Pseudocode* 4.4 di bawah ini adalah algoritme *encode* yang digunakan dalam penelitian. Di dalamnya akan dipanggil beberapa fungsi yang terdapat di halaman-halaman selanjutnya.

```
   INPUT  : - Cover Video
            - Secret Data
   OUTPUT : - Stego Video
            - Index File

1  Call function selectLSF(frames)
2  returning indexLSF
   Call function selectBBDH(frames,indexLSF)
3  returning indexBBDH
   Call function encode(blocks,indexBBDH,ascii)
4  returning modifiedBlocks that have hidden information embedded
5  Combine all blocks (including modifiedBlocks) into modifiedLSF
6  Combine all frames (including modifiedLSF) into video
7  Write video as Stego-Video
   Write index as .mat file
```

***Pseudocode* 4.4** *Encode*

#### 4.2.2.1 Fungsi *MbLSF*

Fungsi ini menggunakan input *frame* yang hasil outputnya berupa indeks LSF yang berupa *array* satu dimensi. Detail fungsi ini dapat dilihat di *Pseudocode* 4.5.

```
FUNCTION MbLSF(frames)
  INPUTS      : frames
  OUTPUTS     : indexLSF

1 FOR i = 1 to N - 1                  // N is number of frames
2   [u(i),v(i)] = getUV(frame(i), frame(i+1))
3   Compute r = sqrt(u(i)^2 + v(i)^2);
4   r_T(i) = average of the avarage of r;
5 ENDFOR
  r_Tot = average of r_T;

6 SET j = 1;                          // j as LSF index in frames
7 FOR i = 1 to N-1
8  IF r_T ≥ r_Tot THEN
9    indexLSF(j) = i;
     j++;
   ENDIF
  ENDFOR
ENDFUNCTION
```

***Pseudocode* 4.5** Fungsi *MbLSF*

#### 4.2.2.2 Fungsi *getUV*

Fungsi ini mencari *optical flow* vektor *u* dan *v* antara dua buah gambar. Vektor tersebut berupa data dua dimensi. Detail fungsi ini dapat dilihat pada *Pseudocode* 4.6.

**FUNCTION** *getUV*(*image1*, *image2*)
**INPUTS** : two *images*
**OUTPUTS** : *vector u* and *vector v*

1

Compute $I_x$, $I_y$ and $I_t$; //we use 2D convolution
$I_x$ = conv2(*image1*, $\frac{1}{4}\times\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}$)
+ conv2(*image2*, $\frac{1}{4}\times\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}$)

$I_y$ = conv2(*image1*, $\frac{1}{4}\times\begin{bmatrix}-1 & -1\\ 1 & 1\end{bmatrix}$)
+ conv2(*image2*, $\frac{1}{4}\times\begin{bmatrix}-1 & -1\\ 1 & 1\end{bmatrix}$)

$I_t$ = conv2(*image1*, $\frac{1}{4}\times\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$)
+ conv2(*image2*, $\frac{1}{4}\times\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$)

2

$kernel = \frac{1}{12}\cdot\begin{bmatrix}1 & 2 & 1\\ 2 & 0 & 2\\ 1 & 2 & 1\end{bmatrix}$; //this is averaging Kernel

// Threshold *T* is the maximum number of iterations, we use 10 to
// gain decent result, larger value gives slower time it will take,
// while smaller value gives worse accuracy

3
4
5

**Select** weight factor $\Lambda$; //Default value is 1
**SET** $T$ = 10, $n$ = 1;
**FOR** $n$ = 1 to last 2 *frame*, **DO**
// $N_{rows}$ and $N_{cols}$ is the dimension of $u$ and $v$

6
7

**FOR** $y$ = 1 to $N_{rows}$ **DO**
**FOR** $x$ = 1 to $N_{cols}$ **DO**

8

Compute $u_a$, $v_a$; //using 2D convolution
$u_a$ = conv2($u_{(x,y)}$, kernel)
$v_a$ = conv2($v_{(x,y)}$, kernel)

9

Compute $\alpha_{(x,y,n)}$;

$$\alpha_{(x,y,n)} = \frac{I_{x(x,y)}u_{a(x,y)} + I_{y(x,y)}v_{a(x,y)} + I_{t(x,y)}}{\Lambda + I^2_{x(x,y)} + I^2_{x(x,y)}}$$

10

Compute $u_{(x,y)}$ and $v_{(x,y)}$;

$u_{(x,y)} = \vec{u} - \alpha_{(x,y,n)} \cdot I_{x_{(x,y)}}$
$v_{(x,y)} = \vec{v} - \alpha_{(x,y,n)} \cdot I_{y_{(x,y)}}$

***Pseudocode* 4.6** Fungsi *getUV*

```
        END FOR
    END FOR
  END FOR
ENDFUNCTION
```

***Pseudocode* 4.6** Fungsi *getUV*

#### 4.2.2.3 Fungsi *BbDH*

Fungsi ini menghasilkan indeks *dissimilar blocks*. Indeks tersebut berupa *array* dua dimensi yang pada dasarnya merepresentasikan indeks blok terpilih (indeks *dissimilar blok*) di indeks *frame* (indeks LSF). Detail fungsi dapat dilihat pada Pseudocode 4.7.

```
   FUNCTION BbDH(frames,indexLSF)
      INPUTS    : frames
      OUTPUTS   : indexDB

1    SET i = 1; j = 1;
2    FOR each indexed LSF frames DO
3      FOR 1 to n DO
4        Divide each frames into array of blocks as blocks(j,i);
5        Compute histogram of blocks(j,i) as histblocks(j,i), an array;
6        Compute histogram difference of
           current block in the current frame and
           current block in the next frame
           as histdiffblocks(j,i), an array;
       ENDFOR
     ENDFOR
7    Assign maximum of histdiffblocks(j,i) as HCV;
8    SET bbdh = histdiffblocks ≤ HCV;
   ENDFUNCTION
```

**Pseudocode 4.7** Fungsi *BbDH*

#### 4.2.2.4 Fungsi *encodeWRGB*

Fungsi ini menyisipkan *secret data* ke piksel pada *dissimilar block*. Detail fungsi ini dapat dilihat pada *Pseudocode* 4.8.

```
     FUNCTION encodeWRGB(concecutive blocks, secret data)
     Inputs    : - frame(s) or block(s)
                 - secret data
     Outputs   : - encoded frame(s) or encoded block(s)
                 - unsued characters
1
2    FOR EACH LSF
3      FOR EACH DB
4        FOR EACH pixel
5          Reset the weight of the pixel
6          FOR EACH color channel
               Add the converted ASCII into weight
           ENDFOR
         ENDFOR
       ENDFOR
     ENDFOR
     ENDFUNCTION
```

***Pseudocode* 4.8** Fungsi encodeWRGB

### 4.2.3 Decoding

Pada tahap ini, informasi rahasia akan diekstrak dari berkas stego-video.

```
     Inputs:  - stego-video
              - .mat file variables
     Outputs: ..txt file extracted

1    Read stego-video file.
2    Read .mat file.
3    Extract all indexes in .mat.
4    Subtract the numbers gathered.
5    Combine each of the three subtracted number as one number.
6    Convert all numbers into ASCII characters.
7    Write the converted characters into a txt file
```

***Pseudocode* 4.9** *Decode*

# BAB V
# HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pada perangkat lunak penyisipan pesan tersembunyi pada berkas audio. Hasil uji coba didapatkan dari implementasi perangkat lunak yang dibahas pada bab empat. Uji coba dilakukan dengan berbagai skenario.

## 5.1 Lingkungan Pengujian

Deskripsi lingkungan pengujian untuk proses uji coba perangkat lunak penyisipan pesan tersembunyi pada berkas audio dapat dilihat pada Tabel 4.1.

**Tabel 5.1** Lingkungan Pengujian

| Perangkat | Jenis Perangkat | Spesifikasi |
|---|---|---|
| **Perangkat Keras** | Prosesor | Intel® Core™ i5 3230M 2.60 GHZ |
| | Memori | 4 GB 1333 MHz DDR3 |
| **Perangkat Lunak** | Sistem Operasi | Windows 10 Pro |
| | Perangkat Pengembang | Matlab R2018a |

## 5.2 Data Pengujian

Data pengujian yang digunakan untuk uji coba perangkat lunak penyisipan data rahasia adalah berkas video dengan ekstensi *.avi* atau *.mp4* dan berkas teks *.txt* yang berisi data rahasia. Detail tentang data pengujian dapat dilihat pada Tabel 5.2.

Berkas video yang digunakan diperoleh dari *test file* yang terdapat di dalam *folder* Matlab yaitu *vipmen.avi*, *rhinos.avi* dan *singleball.mp4*. Sementara itu, berkas secret data yang digunakan hanyalah 48320-0.txt berukuran 622.136 byte. Pada Tabel 5.3, 48320-0.txt digandakan dan diedit yang hasilnya berupa *a.txt*, *b.txt*, *c.txt* dan *d.txt*.

**Tabel 5.2** Data Berkas *Cover Video*

| *File* | *Depth* | *Resolution w* x *h* | | *Size* (*bytes*) | *Frames* | *Blocks* |
|---|---|---|---|---|---|---|
| *singleball .mp4* | 24 | 480 | 360 | 87.395 | 45 | 43.200 |
| *vipmen .avi* | 24 | 160 | 120 | 16.442.880 | 283 | 45.280 |
| *rhinos .avi* | 24 | 320 | 240 | 26.270.720 | 114 | 72.960 |

Berkas video yang digunakan diperoleh dari *test file* yang ada di *folder* Matlab. Sementara itu, berkas *secret data* yang digunakan hanyalah *48320-0.txt* berukuran 622.136 *byte*. Pada Tabel 5.3 dapat dilihat hasil *48320-0.txt* yang digandakan dan diedit.

**Tabel 5.3** Data Berkas Pesan Rahasia

| *File* | *Size* (*bytes*) | *Characters* |
|---|---|---|
| *a.txt* | 500.000 | 500.000 |
| *b.txt* | 250.000 | 250.000 |
| *c.txt* | 125.000 | 125.000 |
| *d.txt* | 50.000 | 50.000 |

## 5.3 Skenario Uji Coba

Uji coba dilakukan pada kedua tahap yaitu *encode* dan *decode*. Pada tahap *encode*, hasil akhir berupa *key* yang nantinya akan dienkripsi dan *stego video*. Setiap *encrypted key* terkait akan disisipkan ke dalam *metadata stego video*. Detail dapat dilihat pada Tabel 5.4.

Pada tahap *decode*, *key* diperoleh dengan *decrypt*. Setelah itu, *key* diimpor sebagai indeks LSF dan DB. Hasil akhir berupa *decoded message file*.

**Tabel 5.4** Skenario Uji Coba *Encode*

| *Cover Video* | *Secret* | *Stego Video* | *key* |
|---|---|---|---|
| (1) *vipmen.avi* | *a.txt* | *1a.avi* | *1a.mat* |
| | *b.txt* | *1b.avi* | *1b.mat* |
| | *c.txt* | *1c.avi* | *1c.mat* |
| | *d.txt* | *1.avi* | *1d.mat* |
| (2) *rhinos.avi* | *a.txt* | *2a.avi* | *2a.mat* |
| | *b.txt* | *2b.avi* | *2b.mat* |
| | *c.txt* | *23c.avi* | *2c.mat* |
| | *d.txt* | *2.avi* | *2d.mat* |
| (3) *singleball.mp4* | *a.txt* | *3a.avi* | *3a.mat* |
| | *b.txt* | *3b.avi* | *3b.mat* |
| | *c.txt* | *3c.avi* | *3c.mat* |
| | *d.txt* | *3d.avi* | *3d.mat* |

### 5.3.1 Hasil Uji Coba Encode

*Least significant frame* (LSF) dan *dissimilar blok* (DB) pada setiap berkas *cover video* akan dicari. Perolehan ini dapat dilihat pada Tabel 5.5.

**Tabel 5.5** Kapasitas Penyimpanan

| *File* | *Size* (*byte*) | *Hiding Data Capacity* | MbLSF | | BbDH | |
|---|---|---|---|---|---|---|
| | | | LSF | *Fra-mes* | DB | *Blocks* |
| *vipmen .avi* | 16.442.880 | 3.054.600 | 82 | 283 | 8.485 | 45.280 |
| *rhinos .avi* | 26.270.720 | 8.116.560 | 54 | 114 | 22.546 | 72.960 |

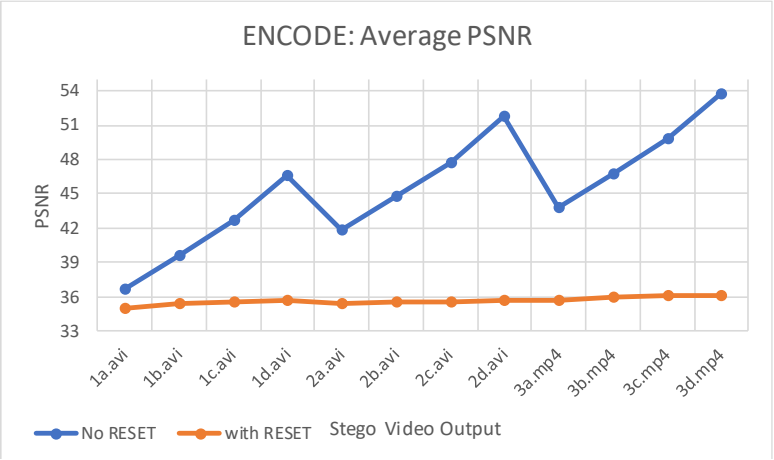
| *File* | *Size* (*byte*) | *Hiding Data Capacity* | MbLSF | | BbDH | |
|---|---|---|---|---|---|---|
| | | | LSF | *Fra-mes* | DB | *Blocks* |
| *singleball.mp4* | 87.395 | 9.061.740 | 28 | 45 | 16.781 | 43.200 |

Sementara itu, setiap karakter pada berkas pesan rahasia masing-masing dikonversikan menjadi angka yang bernilai antara 000 hingga 255. Dari hasil konversi tersebut, tiap angka dipecah menjadi tiga angka baru yang merepresentasikan tiap digit pada angka lama. Sehingga diperoleh jumlah sebesar tiga kali lipat dari jumlah karakter awal. Detail dapat dilihat pada Tabel 5.6.

**Tabel 5.6** Hasil Konversi Karakter Pesan Rahasia

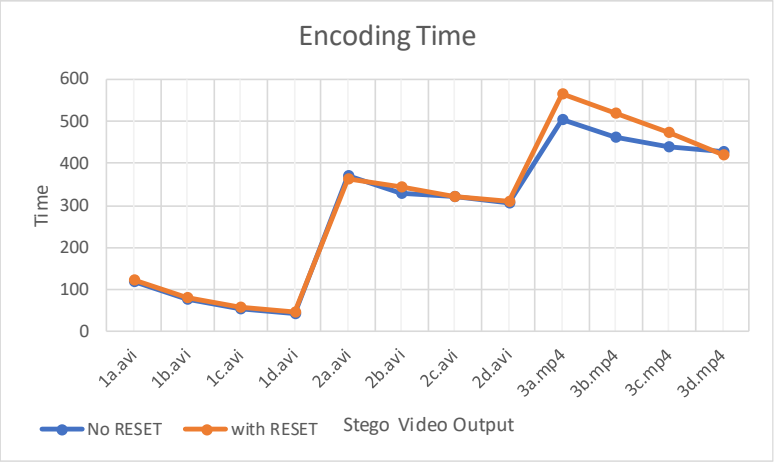
| *File* | *Size* (*byte*) | *Characters* | *Converted Characters* |
|---|---|---|---|
| *a.txt* | 500.000 | 500.000 | 1.500.000 |
| *b.txt* | 250.000 | 250.000 | 750.000 |
| *c.txt* | 125.000 | 125.000 | 375.000 |
| *d.txt* | 50.000 | 50.000 | 150.000 |

Sebelum dilakukan *embedding*, seluruh bobot piksel pada *dissimilar blocks* di*reset* sementara cara yang lain adalah tanpa *reset*. Hasilnya dapat dilihat pada .




**Gambar 5.1** Grafik PSNR *Encode*

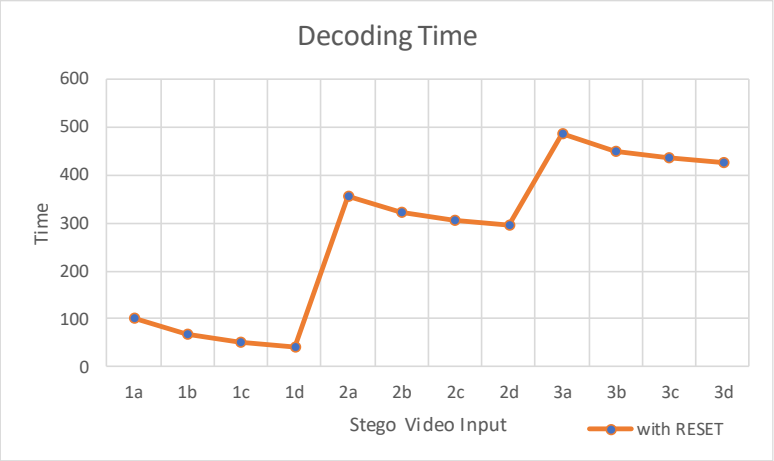
Reset dilakukan dengan mensubstraksi dengan 10 seluruh digit teakhir bobot piksel RGB di tiap *dissimilar blocks.*




**Gambar 5.2** Grafik *Encoding Time*

### 5.3.2 Hasil Uji Coba Decode

Waktu *decode* menggunakan RGBW dengan subtraksi seluruh blok (*reset*). Untuk *decode* yang tanpa menggunakan *reset*, perangkat keras belum mampu untuk sementara ini.




**Gambar 5.3** Grafik *Decoding Time*

*Dissimilar blocks* yang tidak dilakukan *reset* sebelum *embedded* akan membutuhkan *resource* dan *time* yang lebih besar dibandingkan *with reset* sebab pada *with reset*,bobot piksel tetangga yang tidak mengalami *embedded* akan di *ignore* (nilai 0 karena *reset*), sedangkan satu-satunya cara agar tanpa *reset* berhasil adalah dengan pemrosesan perpiksel bukan lagi per blok, yang pada saat dieksekusi mengalami masalah karena *memory* yang dibuthkan tidak mencukupi.

## 5.4 Evaluasi Skenario Uji Coba

*Encode* tanpa *reset* menghasilkan PSNR yang jauh lebih baik dari pada *with reset*. *Encoding Time* tanpa *reset* juga tampak lebih cepat dari pada *with reset*.

# BAB VI
# KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembangunan perangkat lunak dan hasil uji coba yang telah dilakukan. Hal-hal tersebut digunakan sebagai jawaban dari rumusan masalah yang telah dikemukakan. Terdapat juga saran yang dapat digunakan Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

## 6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba penyisipan pesan tersembunyi pada berkas audio adalah sebagai berikut:

1. Hasil PSNR *with reset* adalah 34 dB s.d. 36 dB (kurang bagus) Sementara 36 dB s.d. 53 dB adalah hasil *encode* tanpa *reset*.
2. Seleksi *frame* dilanjutkan dengan seleksi blok membuat berkurangnya HDC.

## 6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan perangkat lunak di masa yang akan datang berdasarkan hasil perancangan, implementasi, dan uji coba yang telah dilakukan.Saran yang diberikan terkait pengembangan pada penelitian ini adalah:

1. Optimasi perlu dilakukan terutama pada saat *decode without reset*. Salah satu cara yaitu dengan menambakan variabel ukuran/jumlah karakter *secret message* (buikan isinya) pada saat *key* ditulis. Sehingga terdapat perbedaan antara *tampered pixels* dengan *original pixels*.

*[Halaman ini sengaja dikosongkan]*

# DAFTAR PUSTAKA


[1] Merriam-Webster, "Video | Definion of Video by Merriam-Webster," [Online]. Available: https://www.merriam-webster.com.

[2] Technopedia, "What is Frame Rate?," [Online]. Available: https://www.techopedia.com.

[3] D. Photography, "Bit depth explained - what 8Bit, 10Bit and 12Bit really means," [Online]. Available: https://www.diyphotography.net.

[4] W. Fulton, "Understanding RGB Color, Bit Depth, Image Data Size. Calculators, image size and MB and GB conversion," 2018. [Online]. Available: https://www.scantips.com. [Diakses 2018].

[5] N. Johnson, Z. Duric dan S. Jajodia, Information Hiding: Steganography and Watermarking-Attacks and Countermeasures: Steganography and Watermarking : Attacks and Countermeasures, Springer Science & Business Media, 2001.

[6] A. Solichin dan Painem, "Motion-based Less Significant Frame for Improving LSB-based Video Steganography," *International Seminar on Application for Technology of Information and Communication,* pp. 179-183, 2016.

[7] Mathworks, "Optical Flow," [Online]. Available: https://www.mathworks.com.

[8] A. Burton dan J. Radford, "Thinking in Perspective: Critical Essays in the Study of Thought Processes," Routledge, 1978.

[9] D. H. Warren dan dkk, dalam *Electronic Spatial Sensing for the Blind*, D. H. Warren dan E. R. Sterlow, Penyunt., Springer.

[10] E. Patel dan D. Shukla, “Comparison of Optical Flow Algorithms for Speed Determination of Moving Objects,” *International Journal of Computer Applications,* vol. 63, no. 5, pp. 32-37, February 2013.

[11] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence,* vol. 17, no. 1-3, pp. 185-203, August 1981.

[12] O. Cetin dan T. A. Ozcerit, “A new steganography algorithm based on color histograms for data embedding into raw video streams,” *Computers & Security,* vol. 28, no. 6, pp. 670-682, 2010.

[13] W. Bender, D. Gruhl, N. Morimoto and A. Lu, "Techniques for Data Hiding," *IBM Systems Journal,* vol. 35, no. 313-36, pp. 3-4, 1996.

[14] S. H. Varol dan F. Akar, “A new RGB weighted encoding technique for Efficient Information Hiding in Images,” *Journal of Naval Science and Engineering,* no. 2, pp. 21-36, 2004.

[15] Injosoft AB, “ASCII Code - The extended ASCII table,” [Online]. Available: https://www.ascii-code.com.

[16] Sharpened Productions, “Techterms | ASCII,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[17] Quora, “Quora | Why is it CMYK and not CMYB?,” 2017. [Online]. Available: https://www.quora.com.

[18] Sharpened Productions, “Techterms | RGB,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[19] Adobe, “After Effects User Guide - Color - Color depth and high dynamic range color,” Adobe, [Online]. Available: https://helpx.adobe.com/.

[20] MathWorks, “MathWorks Announces Release 2018a of the MATLAB and Simulink Product Families,”

MathWorks, 15 Marh 2018. [Online]. Available: https://www.mathworks.com.

[21] Mathworks, “What is MATLAB?,” [Online]. Available: https://www.mathworks.com.

[22] E. L. Lechmann dan G. Casella, Theory of Point Estimation, 3nd penyunt., New York: Springer-Verlag New York, 1998.

[23] S. T. Welstead, Fractal and Wavelet Image Compression Techniques, Bellingham, Washington: SPIE, 1999.

[24] A. N. Netravali dan B. G. Haskell, Digital pictures: representation, compression, and standards, 2nd penyunt., New York: Plenum Press, 1995.

[25] M. Rabbani dan P. W. Jones, Digital image compression techniques, vol. VII, Bellvue, Washington: SPIE Optical Engineering Press, 1991.

[26] Packetizer, Inc., "AES Crypt," [Online]. Available: https://www.aescrypt.com.

[27] VideoLAN, "VLC media player," [Online]. Available: https://wiki.videolan.org/.

[28] Badan Pengembangan dan Pembinaan Bahasa, Kementerian Pendidikan dan Kebudayaan Republik Indonesia, "KBBI Daring," 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/.

[29] Sharpened Productions, “Techterms | Frame,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[30] Wikia, Inc., "FAQ | Matlab Wikia," [Online]. Available: http://matlab.wikia.com.

[31] Sharpened Productions, “Techterms | Character Encoding,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[32] R. Lemeden, "A Closer Look at Color Lightness," 13 March 2014. [Online]. Available: https://robots.thoughtbot.com.

[33] Sharpened Productions, “Techterms | Algorithm,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[34] IGI Global, "What is Color Histogram," [Online]. Available: https://www.igi-global.com.

[35] IGI Global, "What is PSNR," [Online]. Available: https://www.igi-global.com.

[36] Sharpened Productions, “Techterms | Pseudocode,” Sharpened Productions, 2018. [Online]. Available: https://techterms.com.

[37] G. Wyszecki dan D. B. Judd, “Color in Business, Science and Industry,” dalam *Color in Business, Science and Industry*, 3rd penyunt., New York, Wiley, 1975, p. 388.

[38] Federal Agencies Digital Guidelines Initiative, "Term: Color Channel," [Online]. Available: http://www.digitizationguidelines.gov.

[39] M. Dixit, N. Bhide, S. Khankhoje and R. Ukarande, "Video Steganography," *International Conference on Pervasive Computing (ICPC),* pp. 1-4, 2015.

[40] Jarekt, “File:Matlab Logo.png - Wikipedia,” [Online]. Available: https://en.m.wikipedia.org.

[41] C. P. Sumathi, t. Santanam dan G. Umamaheswari, “A Study of Various Steganographic Techniques Used for Information Hiding,” *IJCSES,* vol. 4, no. 6, 2013.

[42] N. Thomos, N. V. Boulgouris and M. G. Strintzis, "Optimized Transmission of JPEG2000 Streams Over Wireless Channels," *IEEE Transactions on Image Processing*, vol. 15, no. 1, January 2006.

# BIODATA PENULIS

Fauzan Adim adalah anak sulung Bapak Ir. Sudaryanto dan Ibu Wiwik Setiyowati S.Si., M.M. yang lahir di Surabaya pada tanggal 12 Desember 1995. Usai menempuh jenjang SD di SDN Tugu Jebres 120 Surakarta (2001 – 2006), penulis melanjutkan pendidikannya di SMP N 4 Surakarta (2007 – 2009) lalu SMA N 3 Surakarta (2010 – 2012) hingga dia menempuh pendidikan tinggi (2013 – 2018) di Institut Teknologi Sepuluh Nopember Jurusan Teknik Informatika bidang minat Komputasi Berbasis Jaringan (KBJ).

Penulis dapat dihubungi melalui alamat surel:
wishmeluck.amien@ymail.com