TUGAS AKHIR - KI141502

# PENGEMBANGAN METODE QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION UNTUK STEGANOGRAPHY PADA AUDIO

**DANANG ADI NUGROHO**
**NRP 5113100151**

Dosen Pembimbing I
Tohari Ahmad, S.Kom., MIT., Ph.D.

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

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

# PENGEMBANGAN METODE QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION UNTUK STEGANOGRAPHY PADA AUDIO

**DANANG ADI NUGROHO**
**NRP 5113100151**

**Dosen Pembimbing I**
**Tohari Ahmad, S.Kom., MIT., Ph.D.**

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

**Jurusan Teknik Informatika**
**Fakultas Teknologi Informasi**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2017**

*[Halaman ini sengaja dikosongkan]*


UNDERGRADUATE THESIS – KI141502

# DEVELOPING QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION FOR STEGANOGRAPHY IN AUDIO

**DANANG ADI NUGROHO**
**NRP 5112100151**

**First Advisor**
**Tohari Ahmad, S.Kom., MIT., Ph.D.**

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

**Department of Informatics**
**Faculty of Information Technology**
**Sepuluh Nopember Institute of Technology**
**Surabaya 2017**

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

## PENGEMBANGAN METODE QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION UNTUK STEGANOGRAPHY PADA AUDIO

**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:
**DANANG ADI NUGROHO**
**NRP: 5113100151**

Disetujui oleh Pembimbing Tugas Akhir:

1. Tohari Ahmad, S.Kom., MIT., Ph.D. .....
   (NIP. 197505252003121002) (Pembimbing 1)

2. Hening Titi Ciptaningtyas, S.Kom., M.Kom. .....
   (NIP. 198407082010122004) (Pembimbing 2)

**SURABAYA**
**JUNI, 2017**

*[Halaman ini sengaja dikosongkan]*

**PENGEMBANGAN METODE QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION UNTUK STEGANOGRAPHY PADA AUDIO**


| | |
|---|---|
| **Nama Mahasiswa** | **: DANANG ADI NUGROHO** |
| **NRP** | **: 51131001151** |
| **Jurusan** | **: Teknik Informatika FTIF-ITS** |
| **Dosen Pembimbing 1** | **: Tohari Ahmad, S.Kom., MIT., Ph.D.** |
| **Dosen Pembimbing 2** | **: Hening Titi Ciptaningtyas, S.Kom., M.Kom.** |


## Abstrak


*Steganography adalah salah satu bagian dari teknik penyembunyian data. Tujuan utama dari steganography adalah untuk menjaga orang lain agar tidak mengetahui informasi rahasia yang disembunyikan. Banyak penelitian yang telah dilakukan dalam bidang ini, khususnya steganography dengan media citra digital. Pada penelitian ini akan dilakukan penyembunyian data pada audio. Berkas audio yang akan digunakan adalah berkas audio dengan format WAV. Berkas audio tersebut memiliki panjang sampel 8-bit dan merupakan berkas audio mono atau single channel. Data yang disembunyikan pada audio merupakan berkas pesan tersembunyi berbentuk teks.*

*Metode-metode yang diimplementasikan untuk penyembunyian data pada audio adalah general reduced difference expansion, quad-based general rde, block overlap general rde, dan quad smoothness general rde. Perangkat lunak yang digunakan untuk melakukan penyembunyian data diimplementasikan dengan menggunakan bahasa pemrograman Python versi 3.6.0 dan Anaconda. Pengujian dilakukan pada empat buah berkas audio berbeda. Berkas audio yang digunakan merupakan potongan dari sebuah lagu dengan panjang 94 detik. Uji coba dilakukan dengan menyisipkan pesan tersembunyi berupa teks dengan ukuran-ukuran yang berbeda.*

*Dari hasil analisa uji coba didapatkan bahwa pemilihan coveraudio berpengaruh pada nilai PSNR stegoaudio. Panjang coveraudio tidak berpengaruh pada kapasitas penyisipan pesan tersembunyi. GRDE dan Quad-RDE memiliki kapasitas penyimpanan lebih besar dari metode lainnya. Metode RDE adalah metode yang memiliki hasil PSNR yang lebih besar dari metode lainnya.*

***Kata kunci: Steganography, difference expansion, reduced difference expansion, generalized difference expansion.***

**DEVELOPING QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION FOR STEGANOGRAPHY IN AUDIO**


| | |
|---|---|
| **Student's Name** | **: DANANG ADI NUGROHO** |
| **Student's ID** | **: 5113100151** |
| **Department** | **: Teknik Informatika FTIF-ITS** |
| **First Advisor** | **: Tohari Ahmad, S.Kom., MIT., Ph.D.** |
| **Second Advisor** | **: Hening Titi Ciptaningtyas, S.Kom., M.Kom.** |



***Abstract***

*Steganography is a method of data hiding techniques. The main purpose of steganography is to keep others from knowing the secret data that is hidden. Much research has been done in this field, especially steganography with digital image. In this research, a data will be concealed within an audio. The audio which used in this research is an audio with WAV format. This audio has 8-bit sample length and is a mono audio or a single channel audio. The data hidden within the audio is a text file.*

*The methods implemented for data hiding are general reduced difference expansion, quad-based general rde, block overlap general rde, and quad smoothness general rde. The software used to perform the data hiding is implemented using Python programming language version 3.6.0 and Anaconda. The test is performed on four different audio files. The audio file which is used is a snippet of a song. The snippet is 94 seconds long. The test is done by inserting a hidden message with different sizes.*

*The experiment result showed that the selection of the coveraudio has an effect on the stegoaudios PSNR value. The coveraudio length has no effect on the hidden message insertion capacity. GRDE and Quad-RDE have greater hiding capacity than the other methods. The RDE has higher PSNR results than the other methods.*

***Keywords: Steganography, difference expansion, reduced difference expansion, generalized difference expansion.***

## KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: "**PENGEMBANGAN METODE QUAD SMOOTHNESS REDUCED DIFFERENCE EXPANSION UNTUK STEGANOGRAPHY PADA AUDIO**". Tugas Akhir ini merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Kedua orang tua dan saudara penulis yang merupakan sumber dukungan, contoh, motivasi, dan doa bagi penulis dalam setiap perjalanan hidup yang penulis tempuh termasuk selama perkuliahan dan penyelesaian Tugas Akhir.
2. Bapak Tohari Ahmad, S.Kom., MIT., Ph.D. dan Ibu Hening Titi Ciptaningtyas, S.Kom., M.Kom. selaku dosen pembimbing penulis yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
4. Bapak Dr. Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir.

5. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
6. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
7. Atika Nur Kiptia yang selalu memberikan dukungan dan semangatnya baik secara langsung maupun tidak langsung.
8. Achmad Faisal Yanuar dan Faizal Anugrah Bhaswara yang selalu ada di saat suka maupun duka.
9. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
10. Teman-teman administrator NCC/KBJ, yang telah menemani dan menyemangati penulis selama penulis menjadi administrator, menjadi rumah kedua penulis selama penulis berkuliah.
11. Teman-teman angkatan 2013, yang sudah mendukung penulis selama perkuliahan.
12. Pihak-pihak yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juli 2017

# DAFTAR ISI

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

# DAFTAR TABEL

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SUMBER

*[Halaman ini sengaja dikosongkan]*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

*Steganography* adalah salah satu bagian dari teknik penyembunyian data [1]. Sebuah informasi rahasia dikodekan sedemikian rupa sehingga keberadaan informasi tersebut tidak diketahui. Tujuan utama *steganography* adalah untuk menjaga orang lain agar tidak mengetahui keberadaan informasi rahasia tersebut [2].

Banyak penelitian telah dilakukan dalam bidang ini, khususnya *steganography* dengan media citra digital. Salah satu metode yang terkenal adalah *difference expansion* oleh Tian [3]. Tian menyebut metodenya juga dapat diaplikasikan pada audio dan video digital.

Pada penelitian ini akan dilakukan penyembunyian data menggunakan metode yang diusulkan oleh Holil [4], yaitu: *general reduced difference expansion*, *quad-based general rde, block overlap general rde,* dan *quad smoothness general rde*. Metode-metode tersebut akan diaplikasikan pada audio. Diharapkan dengan pengaplikasian metode-metode tersebut dapat menambah metode pengaplikasian *steganography* pada audio dan menumbuhkan semangat untuk penelitian selanjutnya.

Dikemudian hari penyembunyian data dapat diaplikasikan pada media yang lebih beragam. Komunikasi data akan semakin aman.

## 1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara melakukan penyembunyian data pada audio?

2. Bagaimana cara melakukan implementasi metode *general reduced difference expansion* pada audio?
3. Bagaimana cara melakukan implementasi metode *quad-based general rde* pada audio?
4. Bagaimana cara melakukan implementasi metode *block overlap general rde* pada audio?
5. Bagaimana cara melakukan implementasi metode *quad smoothness general rde* pada audio?

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Bahasa pemrograman yang digunakan untuk penyembunyian data adalah Python dengan distribusi *package* Anaconda.
2. Berkas audio yang digunakan sebagai *cover* audio adalah berkas audio dengan format WAV
3. Berkas audio yang digunakan adalah berkas audio *mono* atau *single channel*
4. Panjang sampel dari berkas audio yang digunakan sebagai *cover* audio adalah *unsigned* 8-*bit*.
5. Keluaran dari aplikasi adalah berkas *stegoaudio* dan berkas *location map*.
6. Metode yang digunakan untuk adalah *general reduced difference expansion, quad-based general rde, block overlap general rde,* dan *quad smoothness general rde*.
7. Perangkat lunak yang digunakan adalah PyCharm Professional 2017.1.2 sebagai IDE.

## 1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Melakukan penyembunyian data pada audio.
2. Melakukan implementasi metode *general reduced difference expansion* pada audio.

3. Melakukan implementasi metode *quad-based general rde* pada audio.
4. Melakukan implementasi metode *block overlap general rde* pada audio.
5. Melakukan implementasi metode *quad smoothness general rde* pada audio.
6. Membuat aplikasi penyembunyian data pada audio dengan metode yang diusulkan.

## 1.5 Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Memberikan manfaat di bidang *steganography* terutama *steganography* pada audio.
2. Menerapkan ilmu yang dipelajari selama kuliah di Teknik Informatika ITS agar dapat berguna bagi masyarakat.

## 1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut.

### 1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal Tugas Akhir ini adalah menyusun proposal Tugas Akhir. Pada proposal, berisi tentang deskripsi pendahuluan dari Tugas Akhir yang akan dibuat. Proposal juga berisi tentang garis besar yang akan dikerjakan sehingga memberikan gambaran untuk dapat mengerjakan Tugas Akhir sesuai dengan *timeline* yang dibuat serta gagasan untuk melakukan penyembunyian data pada audio dengan menggunakan metode-metode yang diusulkan [4].

### 1.6.2 Studi Literatur

Tahap ini dilakukan untuk mencari informasi dan literatur apa saja yang dapat dijadikan sebagai referensi untuk membantu pengerjaan Tugas Akhir ini. Tahap ini merupakan tahap untuk memahami semua metode yang akan dikerjakan, sehingga memberi gambaran selama pengerjaan Tugas Akhir. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan. Informasi yang dicari adalah metode-metode *steganography*, dan metode-metode statistika seperti *variance*, *MSE*, dan *PSNR* [5] serta metode-metode untuk melakukan pengolahan audio dengan format WAV dan pengolahan berkas teks ke berkas biner [6], [7]. Tugas akhir ini juga mengacu pada literatur jurnal dengan judul "*Increasing the Performance of Difference Expansion-based Steganography when Securing Medical Data"* yang diterbitkan pada tahun 2014 [4].

### 1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, implementasi dilakukan dengan menggunakan suatu perangkat lunak yaitu Python versi 3.6.0 sebagai bahasa pemrograman utama dengan distribusi *package* Anaconda dan PyCharm Professional 2017.1.2 sebagai IDE.

### 1.6.4 Pengujian dan Evaluasi

Pada tahap ini algoritma yang telah diimplementasikan akan diuji coba dengan menggunakan data uji coba yang ada. Uji coba dilakukan dengan menggunakan suatu perangkat lunak dengan tujuan mengetahui kemampuan metode yang digunakan dan mengevaluasi hasil Tugas Akhir dengan jurnal pendukung yang digunakan. Hasil evaluasi juga mencakup kompleksitas, parameter

dan performa dari metode yang telah diimplementasikan dalam penyembunyian data pada audio.

### 1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

## 1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *steganography*, *difference expansion, reduced difference expansion, generalized difference expansion*, *variance*, *mean squared error*, *peak signal-to-noise ratio* dan PyQt.
3. 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 PyQt.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk *Pseudocode* yang berupa *Pseudocode* dari metode-metode yang diusulkan.
5. 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 audio yang sesuai dengan batasan masalah. Hasil evaluasi mencakup perbandingan kompleksitas, parameter dan performa dari masing-masing data masukan.

6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode program secara keseluruhan.

# BAB II
# TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut di antaranya adalah *difference expansion*, *reduced difference expansion*, dan beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

## 2.1 Audio Digital

Audio digital adalah cara alternatif untuk menyimpan informasi audio. Audio dapat direpresentasikan secara digital dengan berbagai cara, salah satunya adalah dengan *pulse code modulation* (*PCM*). Bentuk dari gelombang audio tidak direpresentasikan secara terus kontinu, tetapi bentuk gelombang diukur dengan interval yang tetap. Proses tersebut disebut dengan *sampling* dan frekuensi banyak sampel yang diambil disebut dengan *sampling rate* atau *sampling frequency* [8]. Audio digital dipilih karena kemudahan modifikasi sampelnya. Audio digital digunakan sebagai basis penyembunyian data pada penelitian ini.

## 2.2 *Steganography*

Sebuah sub disiplin penting dari penyembunyian data adalah *Steganography*. Sementara *Cryptography* adalah ilmu tentang melindungi isi dari suatu pesan, *steganography* adalah ilmu tentang menyembunyikan data. Contoh dari *steganography* adalah mengirimkan pesan kepada seorang mata-mata dengan menandai huruf-huruf tertentu pada koran dengan menggunakan tinta tidak kasat mata dan menambahkan gema yang tidak terdengar pada rekaman audio [9]. *Steganography* dipilih untuk mengembangkan

teknik penyembunyian data pada audio digital. *Steganography* adalah topik utama pada penelitian ini.

## 2.3 *Difference Expansion*

*Difference expansion* adalah metode *steganography* yang dikembangkan oleh Tian [3]. Metode ini menghitung nilai beda dari piksel yang bersebelahan dan memilih nilai beda yang memenuhi syarat untuk dilakukan *difference expansion* (*DE*) [3]. Metode ini dipilih karena mudah diimplementasikan dan hasil penyisipannya tidak mudah diketahui. *Difference expansion* merupakan metode dasar dari metode yang diimplementasikan pada penelitian ini.

## 2.4 *Reduced Difference Expansion*

*Reduced difference expansion* adalah metode *steganography* yang dikembangkan oleh Liu [10]. Metode ini merupakan pengembangan dari *difference expansion*. Metode *DE* dapat diaplikasikan pada sebuah citra lebih dari sekali untuk *multilayer embedding*. Karena operasi *DE* menambah nilai beda dari nilai piksel yang bersebelahan, apabila operasi *multilayer embedding* dilakukan, maka kualitas visual dari suatu citra akan turun dengan drastis. Terlebih lagi, operasi *difference expansion* dapat mengakibatkan *overflow* atau *underflow* dan mengurangi kapasitas penyembunyian data. Oleh karena itu, Liu [10] mengajukan *reduced difference expansion* untuk mengatasi masalah-masalah tersebut [10]. *Reduced difference expansion* merupakan metode dasar dari metode yang diimplementasikan pada penelitian ini.

## 2.5 *Generalized Difference Expansion*

*Generalized difference expansion* adalah metode *steganography* yang dikembangkan oleh Alattar [11]. Metode ini

merupakan pengembangan dari *difference expansion*. Alattar mengembangkan algoritma dari Tian [3] menggunakan *difference expansion* dari vektor-vektor piksel sebagai ganti dari pasangan-pasangan piksel untuk meningkatkan kemampuan penyembunyian data dan efisiensi komputasi algoritma. Pendekatan ini memungkinkan algoritma untuk menyisipkan beberapa bit sekaligus di setiap vektor dengan sekali jalan sepanjang data citra [11]. *Generalized difference expansion* merupakan metode dasar dari metode yang diimplementasikan pada penelitian ini.

## 2.6 Python

Python adalah salah satu bahasa pemrograman tingkat tinggi yang bersifat interpreter, interaktif, *object oriented* dan dapat beroperasi di hampir semua *platform* seperti UNIX, Mac, Windows ataupun yang lain. Sebagai bahasa pemrograman tingkat tinggi Python termasuk salah satu bahasa pemrograman yang mudah dipelajari karena sintaks yang jelas . Python menyediakan *module-module* yang siap dipakai dan juga Python memiliki struktur data tingkat tinggi yang efisien. Python menyediakan 2 versi yaitu versi 2 dan versi 3 di mana di setiap versi memiliki kekurangan dan kelebihan masing-masing [12]. Gambar 2.1 merupakan logo Python. Bahasa pemrograman ini dipakai untuk menulis kode sumber perangkat lunak yang dibangun untuk penelitian ini.


**Gambar 2.1 Logo Python**

## 2.7 Anaconda

Anaconda adalah *open data science platform* yang dibangun menggunakan bahasa pemrograman Python. Versi Anaconda *open-source* menyediakan distribusi dengan performa tingkat tinggi dari bahasa pemrograman Python dan R yang berisikan lebih dari 100 paket untuk *data science*. Anaconda menyediakan CLI (*command line interface*) berupa perintah "*conda*" sebagai manajemen lingkungan untuk bahasa pemrograman Python. Perintah "*conda*" bisa untuk *create, export, list, remove* dan *update* lingkungan yang bisa digunakan untuk versi bahasa pemrograman Python atau *package* yang dipasang di dalamnya. Anaconda dapat digunakan di berbagai *platform* seperti Windows, Linux dan Mac [13]. Pada Anaconda terdapat *module-module* yang berguna untuk pemrosesan data dan pembuatan *GUI*. Anaconda merupakan *platform* dipakai untuk mengimplementasikan metode-metode yang diusulkan pada penelitian ini. Gambar 2.2 merupakan logo Anaconda.


**Gambar 2.2 Logo Anaconda**

## 2.8 Numpy

NumPy atau *numeric Python* merupakan pustaka perangkat lunak bahasa pemrograman Python bersifat *open source* yang digunakan untuk pengolahan ilmiah matematika. Selain digunakan

untuk hal ilmiah, NumPy juga bisa digunakan untuk *container* multidimensi yang efisien untuk data generik [14]. Tipe data *arbitrary* dapat didefinisikan, ini memungkinkan NumPy secara lancar dan cepat mengintegrasikan dengan banyak tipe basis data. Pada penelitian ini, Numpy digunakan untuk memproses data sampel-sampel audio pada penelitian ini. Gambar 2.3 merupakan logo NumPy.


**Gambar 2.3 Logo Numpy**

## 2.9 PyQt

Qt merupakan sebuah IDE (*Integrated Development Environment*) yang dibuat pada tahun 1996. Qt adalah toolkit untuk membantu pengembangan aplikasi yang memiliki tampilan grafis yang dapat berjalan di berbagai *platform*. Sedangkan PyQt merukapan pustaka perangkat lunak Qt yang dapat membantu mengembangkan aplikasi yang memiliki tampilan grafis dengan menggunakan bahasa pemrograman Python [15]. Pada penelitian ini, PyQt digunakan untuk mengembangkan tampilan grafis dari perangkat lunak yang dibangun. Gambar 2.4 merupakan logo Qt.

**Gambar 2.4 Logo Qt**

## 2.10 *Variance*

*Variance* berhubungan erat dengan standar deviasi, yakni variabel yang digunakan untuk mengukur dan mengetahui seberapa jauh penyebaran data dalam distribusi data. Dalam artian lain, *variance* digunakan untuk mengukur variabilitas data, atau tingkat keragaman dalam data. Semakin tinggi nilai *variance*, maka semakin bervariasi dan beragam suatu data [4]. *Variance* digunakan dalam metode *quad smoothness general rde*. *Variance* dapat dihitung menggunakan Persamaan 2.1.

$$Variance = \frac{\sum(X - \mu)^2}{N} \tag{2.1}$$

## 2.11 *Mean Squared Error*

*Mean squared error* adalah metode untuk mengevaluasi dua buah matriks atau *array*. Masing-masing kesalahan atau sisa dikuadratkan. Kemudian dijumlahkan dan ditambahkan dengan jumlah observasi. Pendekatan ini mengatur kesalahan peramalan yang besar karena kesalahan-kesalahan itu dikuadratkan. Metode ini menghasilkan kesalahan-kesalahan sedang yang kemungkinan lebih baik untuk kesalahan kecil, tetapi kadang menghasilkan perbedaan yang besar [4].

*Mean squared error* digunakan dalam perhitungan *peak signal-to-noise ratio*. *Mean squared error* dapat dihitung menggunakan Persamaan 2.2.

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \qquad (2.2)$$

## 2.12 *Peak signal-to-noise Ratio*

*Peak signal to noise ratio* melambangkan rasio antara nilai maksimum (*power*) dari sinyal dan *power* dari *noise* distorsi yang mempengaruhi kualitas representasinya. Karena banyak sinyal memiliki rentang dinamis yang sangat lebar, (rasio antara nilai terkecil dan terkecil dari kuantitas yang dapat berubah) PSNR biasanya dinyatakan dalam skala-skala desibel logaritmik [16]. *Peak signal to noise ratio* dapat digunakan untuk melambangkan beda antara dua audio. Sehingga persamaan ini dipakai dalam penelitian ini untuk menghitung perbedaan audio setelah disisipi pesan rahasia. *Peak signal-to-noise ratio* dapat dihitung menggunakan Persamaan 2.3.

$$PSNR = \log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right) \qquad (2.3)$$

*[Halaman ini sengaja dikosongkan]*

# BAB III
# PERANCANGAN PERANGKAT LUNAK

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

## 3.1 Deskripsi Umum Sistem

Perangkat lunak penyisipan pesan terdiri atas dua proses, yaitu proses *embedding* dan proses *extraction*. Proses *embedding* adalah proses penyisipan pesan tersembunyi ke dalam berkas audio. Proses embedding memerlukan data masukan berupa berkas *coveraudio* dan berkas pesan tersembunyi. Setelah proses *embedding* selesai maka perangkat lunak akan mengeluarkan berkas *stegoaudio* dan berkas *location map*. Proses e*xtraction* adalah proses pengembalian pesan tersembunyi dari berkas *stegoaudio*. Data yang diperlukan proses *extraction* adalah berkas *stegoaudio* dan berkas *location map*. Setelah pesan tersembunyi dikembalikan dari *stegoaudio,* selanjutnya berkas *stegoaudio* akan dikembalikan ke bentuk semula (*coveraudio*). Data keluaran dari proses *extraction* adalah berkas audio yang identik dengan *coveraudio* dan berkas pesan tersembunyi.

Perangkat lunak mengaplikasikan enam metode penyisipan data. Metode-metode tersebut adalah *difference expansion* (*DE*), *reduced difference expansion* (*RDE*), *generalized reduced difference expansion* (*GRDE*), *quad-based reduced difference expansion* (*Quad-RDE*), *block overlap reduced difference expansion* (*Overlap-RDE*), dan *quad smoothness reduced difference expansion* (*Quad-Smoothness-RDE*). Setiap metode

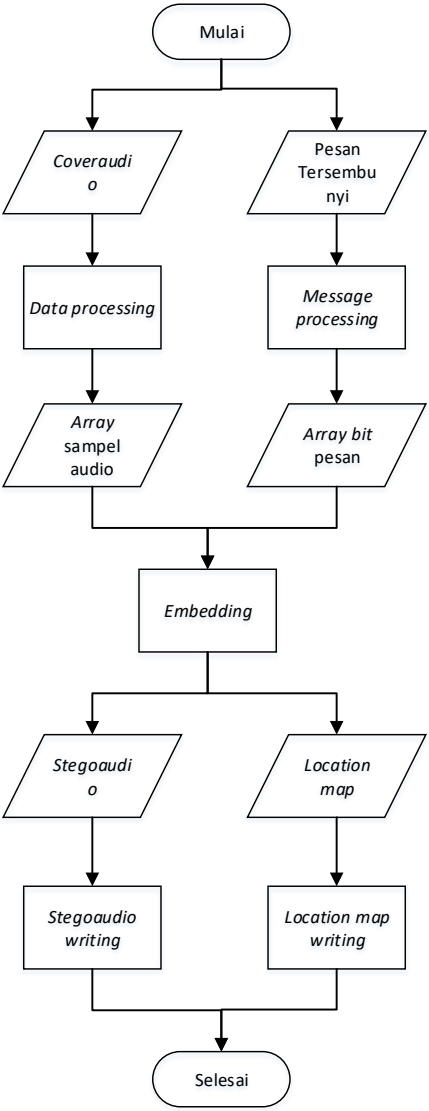
memiliki tahapan proses *embedding* dan proses *extraction* yang berbeda.

Secara umum proses *embedding* terdiri dari beberapa tahapan, yaitu: *data processing, message processing, embedding, stegoaudio writing,* dan *location map writing*. Tahap *data processing* dan *embedding* berbeda untuk setiap metode. Pada tahap *data processing* dilakukan pemrosesan berkas *coveraudio* menjadi bentuk *array*. *Array* tersebut berisikan sampel-sampel suara pada *coveraudio*. Bentuk *array* berbeda untuk setiap metode. Bentuk *array* dapat dilihat pada Tabel 3.1. Pemrosesan pesan tersembunyi dilakukan pada tahap *message processing*. Pesan tersembunyi diubah menjadi bentuk *bit* sehingga dapat disisipkan pada *coveraudio*. *Bit-bit* tersebut disimpan dalam *array* satu dimensi. Setelah itu, *bit-bit* pesan tersembunyi disisipkan pada sampel-sampel *coveraudio*. *Location map* dikumpulkan sejalan dengan proses *embedding*. Setelah proses *embedding* selesai maka tahap selanjutnya adalah *stegoaudio writing*, dan *location map writing*. Sampel-sampel *coveraudio* yang telah diubah akan ditulis menjadi berkas *stegoaudio*. *Location map* ditulis menjadi berkas *location map*. Diagram alir proses *embedding* dapat dilihat pada Gambar 3.1.

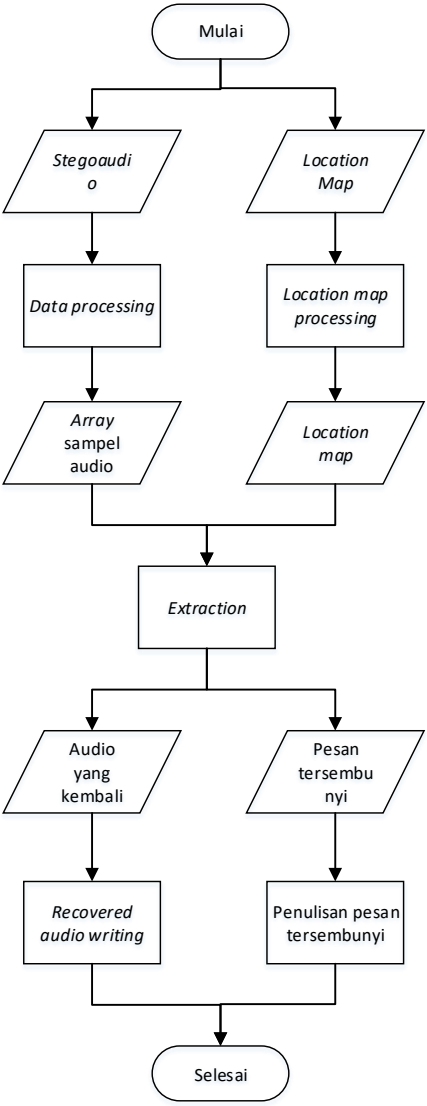
**Tabel 3.1 Bentuk *Array* pada *Data Processing***

| **Metode** | **Bentuk Array** |
|---|---|
| *DE* | (n, 2) |
| *RDE* | (n, 2) |
| *GRDE* | (n, 4) |
| *Quad-RDE* | (n, 4) |
| *Overlap-RDE* | (n, 1) |
| *Quad-Smoothness-RDE* | (n, 4, 4) |

Tahap-tahap proses *extraction* yaitu: *data processing*, *location map processing*, *extraction*, *recovered audio writing*, dan penulisan pesan tersembunyi. Pada tahap *data processing*, *stegoaudio* diproses menjadi *array* yang berisikan sampel-sampel *stegoaudio*. Bentuk *array* juga dapat dilihat pada Tabel 3.1. Berkas *location map* diproses dalam tahap *location map processing*. *Location map* diperlukan agar *stegoaudio* dapat dikembalikan seperti semula. Setelah proses *extraction* selesai, dilakukan penulisan berkas audio yang telah dikembalikan dan penulisan berkas pesan tersembunyi. Masing-masing pada tahap *recovered audio writing* dan tahap penulisan pesan tersembunyi. Diagram alir proses *extraction* dapat dilihat pada Gambar 3.2.




**Gambar 3.1 Diagram Alir Proses *Embedding***




**Gambar 3.2 Diagram Alir Proses *Extraction***

## 3.2 Perancangan Data

Pada sub bab ini akan dijelaskan mengenai berkas masukan yang diperlukan untuk proses penyisipan data dan berkas keluaran dari hasil proses penyisipan data.

### 3.2.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan awal dari perangkat lunak penyisipan pesan tersembunyi. Data masukan yang diperlukan pada perangkat lunak ini antara lain: berkas audio yang berekstensi .wav, berkas *location map*, dan berkas pesan tersembunyi yang berekstensi .txt.

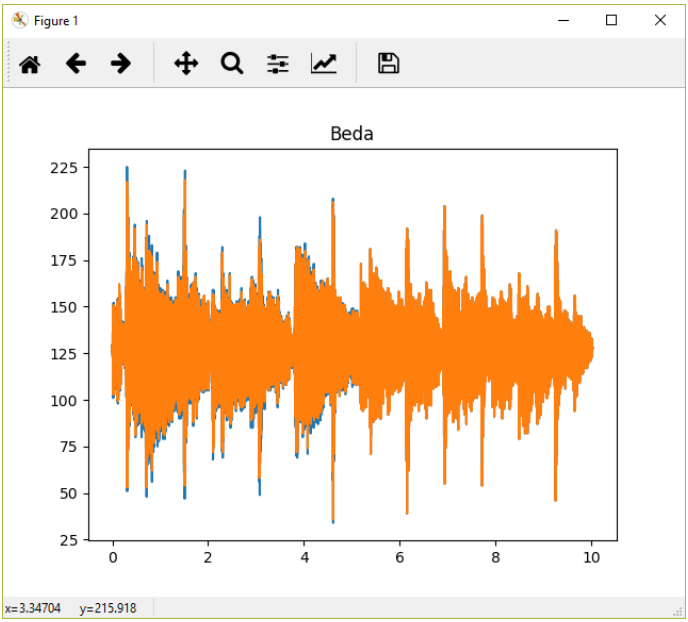
Berkas audio yang dipakai pada perangkat lunak ini adalah berkas audio berformat .wav *mono channel* yang memiliki kedalaman *bit* 8-*bit.* Format tersebut dipilih karena format tersebut adalah bentuk adaptasi dari format berkas citra yang dipakai metode yang menjadi referensi [4].

*Location Map* ditulis menjadi berkas biner. Berkas biner dipilih agar *location map* dapat ditulis dan dibaca dengan mudah oleh perangkat lunak penyisipan pesan tersembunyi.

Perangkat lunak penyisipan data terdiri atas dua proses yang membutuhkan dua data masukan yang berbeda. Proses *embedding* memerlukan masukan berkas *coveraudio* dan berkas pesan tersembunyi. Sedangkan proses *extraction* memerlukan berkas *stegoaudio* dan berkas *location map*.

### 3.2.2 Data Keluaran

Data keluaran perangkat lunak untuk proses *embedding* adalah *stegoaudio* dan *location map*. Perbedaan *stegoaudio* dan *coveraudio* dapat dilihat melalui grafik sampel-sampel suaranya. Perbedaan tersebut juga dapat terdengar jika berkas-berkas tersebut diputar pada aplikasi pemutar audio. Grafik perbedaan dapat dilihat pada Gambar 3.3.




**Gambar 3.3 Beda Coveraudio dengan Stegoaudio**

Proses *extraction* menghasilkan data keluaran audio yang telah dikembalikan dan pesan tersembunyi. Audio yang kembali identik dengan *coveraudio*. Kedua berkas audio tersebut dicocokkan dengan mencocokkan hasil fungsi *hash* kedua berkas tersebut.

## 3.3 Metode Penyisipan Data

### 3.3.1 *Difference Expansion*

*Difference Expansion* (*DE*) merupakan metode untuk menyisipkan data pada nilai beda antar sampel dengan nilai rata-rata tetap antar sampel [4]. Rata-rata antar sampel $m$ dihitung dengan menggunakan Persamaan 3.1. Pasangan sampel dinamakan

$u_1$ dan $u_2$, beda antar sampel $v$ dapat didapatkan dengan Persamaan 3.2.

$$\mathrm{m} = \left\lfloor \frac{\mathrm{u}_1 + \mathrm{u}_2}{2} \right\rfloor \tag{3.1}$$

$$v = u_1 + u_2 \tag{3.2}$$

Misal $\tilde{v}$ adalah nilai hasil dari proses *embedding*. Untuk sebuah bit pesan rahasia $b$ yang nilainya 0 atau 1. Proses penyisipan data dilakukan dengan mengaplikasikan Persamaan 3.3 atau Persamaan 3.4. Pemakaian dua persamaan tersebut tergantung oleh validitas dari hasil Persamaan 3.5. Jika hasil Persamaan 3.3 memenuhi Persamaan 3.5 maka pasangan sampel yang digunakan dikategorikan sebagai *expandable block* dan dalam *location map* akan bernilai 0. Jika tidak, maka digunakan Persamaan 3.4. Jika hasilnya memenuhi Persamaan 3.5 maka pasangan sampel tersebut dikategorikan sebagai *changeable block* dan dalam *location map* akan bernilai 1. Pasangan sampel yang tidak termasuk pada dua kategori di atas maka pasangan sampel tersebut dikategorikan sebagai *unchangeable block* dan dalam *location map* akan bernilai 2. Agar kategori-kategori ini dapat dikenali maka dibuatlah *location map*. *Location map* digunakan agar *stegoaudio* dan pesan tersembunyi dapat dikembalikan seperti semula.

$$\tilde{v} = 2\times v + b \tag{3.3}$$

$$\tilde{v} = 2\times \left\lfloor \frac{b}{2} \right\rfloor + b \tag{3.4}$$

$$\begin{matrix} |\tilde{v}| \leq 2\times(255 - m), \text{jika } 128 \leq \mathrm{m} \leq 255 \\ |\tilde{v}| \leq 2\times m + 1, \text{jika } 128 \leq \mathrm{m} \leq 255 \end{matrix} \tag{3.5}$$

Sampel-sampel baru setelah proses penyisipan, $u_1'$ dan $u_2'$ didapatkan dari Persamaan 3.6.

$$u_1' = m + \left\lfloor \frac{\tilde{v} + 1}{2} \right\rfloor$$
$$u_2' = m - \left\lceil \frac{\tilde{v}}{2} \right\rceil \quad (3.6)$$

### 3.3.2 *Reduced Difference Expansion*

Penggunaan metode *DE* dapat membuat beda antar sampel menjadi dua kali lebih besar. Sehingga digunakan metode *RDE* [4]. Beda sampel $v$ akan direduksi sebelum proses penyisipan data. Proses reduksi dilakukan dengan menggunakan Persamaan 3.7, dimana $\bar{v}$ merupakan hasil beda setelah direduksi.

$$\bar{v} = v, \text{ jika } v < 2$$
$$\bar{v} = v - 2^{\lfloor \log_2 v \rfloor - 1}, \text{ jika } v \geq 2 \quad (3.7)$$

$$v = \bar{v} + 2^{\log_2 |\bar{v}| - 1} \quad (3.8)$$
$$v = \bar{v} + 2^{\log_2 |\bar{v}|} \quad (3.9)$$

$$\textit{Location map} = 0, \text{ jika } 2^{\lfloor \log_2 \tilde{v} \rfloor} = 2^{\lfloor \log_2 v \rfloor}$$
$$\textit{Location map} = 1, \text{ jika } 2^{\lfloor \log_2 \tilde{v} \rfloor} \neq 2^{\lfloor \log_2 v \rfloor} \quad (3.10)$$

Agar sampel yang telah direduksi dapat kembali seperti semula maka dilakukan perhitungan *location map*. Pasangan sampel yang telah direduksi akan ditandai dengan *location map* bernilai 0 bila memenuhi Persamaan 3.10. Sedangkan bila tidak memenuhi persamaan di atas maka akan ditandai dengan *location map* bernilai 1. Nilai *location map* selanjutnya mengikuti *location map* dari metode *DE*, yaitu 2 untuk *expandable*, 3 untuk *changeable*, dan 4 untuk *unchangeable*.

Proses penyisipan bit dan proses perubahan sampel dilakukan dengan proses yang sama dengan metode *DE*. Proses penyisipan bit pesan tersembunyi dilakukan dengan menggunakan

Persamaan 3.3 dan Persamaan 3.4. Proses untuk mendapat sampel-sampel baru dapat dilihat pada Persamaan 3.6

Untuk mengembalikan nilai yang telah direduksi maka dilakukan Persamaan 3.8 atau Persamaan 3.9. Bila *location map* bernilai 0 maka dilakukan Persamaan 3.8, sedangkan bila *location map* bernilai 1 maka dilakukan Persamaan 3.9.

### 3.3.3 *Generalized Reduced Difference Expansion*

Metode ini adalah gabungan dari metode *Generalized Difference Expansion* dan *Difference Expansion*. Pada metode ini akan dibentuk *vector* $u = (u_0, u_1, u_2, u_3)$ yang berisi empat buah sampel. Dari *vector* tersebut akan dihitung nilai bedanya. Setelah itu, nilai beda itu akan direduksi. Proses penghitungan beda dan reduksi dapat dilihat pada Persamaan 3.11 dan Persamaan 3.12.

$$\begin{aligned} v_1 &= u_1 - u_0 \\ v_2 &= u_2 - u_0 \\ v_3 &= u_3 - u_0 \end{aligned} \tag{3.11}$$

$$\begin{aligned} \bar{v}_i &= v_i \text{ , jika } -2 < v_i < 2 \\ \bar{v}_i &= v_i + 2^{\lfloor \log_2 \lfloor v_i \rfloor \rfloor - 1} \text{, jika } -2 \geq v_i \\ \bar{v}_i &= v_i - 2^{\lfloor \log_2 \lfloor v_i \rfloor \rfloor - 1} \text{, jika } v_i \geq 2 \end{aligned} \tag{3.12}$$

*Location map* untuk metode ini sama dengan metode *RDE*. Detail location map dapat dilihat pada Tabel 3.2.

**Tabel 3.2 *Location Map* RDE**

| ***Location Map*** | **Nilai** |
|---|---|
| *RDE*-0 | 0 |
| *RDE*-1 | 1 |
| *Expandable* | 2 |
| *Changeable* | 3 |
| *Unchangeable* | 4 |

$$\begin{aligned} u_0' &= u_0 \\ u_1' &= \tilde{v}_1 + u_0 \\ u_2' &= \tilde{v}_2 + u_0 \\ u_3' &= \tilde{v}_3 + u_0 \end{aligned} \tag{3.13}$$

$$0 \leq u_i' \leq 255 \tag{3.14}$$

Penyisipan bit pesan tersembunyi dilakukan dengan Persamaan 3.3 atau Persamaan 3.4. Untuk memastikan kategori dari *location map* digunakan Persamaan 3.14. Bila ada satu beda $\tilde{v}$ yang masuk kategori *unchangeable*, maka semua beda sampel dalam *vector* $(\tilde{v}_1, \tilde{v}_2, \tilde{v}_3)$ masuk dalam kategori *unchangeable*.

### 3.3.4 *Quad-based General Reduced Difference Expansion*

Metode ini adalah modifikasi dari metode sebelumnya. Beda dari metode ini terdapat pada perhitungan beda dan perhitungan sampel baru yang dapat dilihat pada Persamaan 3.15 dan Persamaan 3.16.

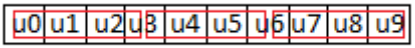
$$\begin{aligned} v_1 &= u_1 - u_0 \\ v_2 &= u_2 - u_1 \\ v_3 &= u_3 - u_2 \end{aligned} \tag{3.15}$$

$$\begin{aligned} u_0' &= u_0 \\ u_1' &= \tilde{v}_1 + u_0' \\ u_2' &= \tilde{v}_2 + u_1' \\ u_3' &= \tilde{v}_3 + u_2' \end{aligned} \tag{3.16}$$

### 3.3.5 *Block Overlap General Reduced Difference Expansion*

Metode ini juga merupakan modifikasi dari metode *GRDE*. Bila sampel pada metode *GRDE* diubah menjadi *array* bentuk (n, 4), maka pada metode ini sampel tidak diubah. Pada sampel tersebut akan dibentuk blok-blok yang saling menindih [4].

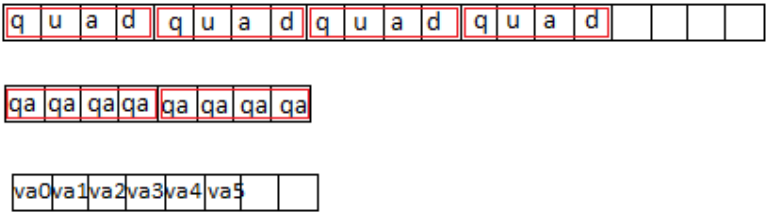
Sampel terakhir pada blok merupakan sampel pertama pada blok setelahnya. Contohnya dapat dilihat pada Gambar 3.4. Blok pertama terdiri dari u0, u1, u2, dan u3. Blok selanjutnya terdiri dari u3, u4, u5, dan u6. Penyisipan bit pesan tersembunyi dilakukan pada beda terakhir, dengan kata lain penyisipan hanya dilakukan pada $u_1'$ dan $u_2'$.




**Gambar 3.4 *Overlapping Block***

### 3.3.6 *Quad Smoothness General Reduced Difference Expansion*

Metode ini adalah pengembangan dari *Quad-based General Reduced Difference Expansion* [4]. Sebelum dilakukan penyisipan data, dilakukan pengurutan berdasarkan *smoothness*. *Smoothness* didapatkan dari mengurutkan *variance* dari empat buah rata-rata *quad*. *Quad* adalah *array* yang berisi empat buah sampel. Maka dari itu *array* yang digunakan pada metode ini adalah *array* yang berbentuk (n, 4, 4). Penyisipan data dimulai dari *smoothness* yang terkecil. Ilustrasi pengurutan *smoothness* dapat dilihat pada Gambar 3.5.




**Gambar 3.5 Penghitungan *Smoothness***

*Quad* dibentuk dari empat buah sampel. Setelah itu setiap *quad* akan dihitung rata-rata sampelnya (*qa*). Lalu *variance va*

akan dihitung dari empat buah *qa*. Dan akhirnya *va* akan diurutkan dari kecil ke besar.

## 3.4 Metode Pengembalian Data

*Location map* dibutuhkan dalam pengembalian data. Untuk melakukan pengembalian data pertama-tama dilakukan penghitungan beda sampel $\tilde{v}_i$ menggunakan Persamaan 3.2, Persamaan 3.11, atau Persamaan 3.15 sesuai dengan metode yang digunakan. Setelah itu bit pesan didapatkan dari mengambil bit terakhir atau *least significant bit* (*LSB*) dari $\tilde{v}_i$. Pengambilan *LSB* dapat dilihat pada Persamaan 3.17.

$$b_i = LSB(\tilde{v}_i) \tag{3.16}$$

Untuk mendapatkan $\bar{v}$ maka dilakukan Persamaan 3.17. Proses ini menggeser $\tilde{v}_i$ satu bit ke kanan.

$$\bar{v}_i = \left\lfloor \frac{\tilde{v}_i}{2} \right\rfloor \tag{3.17}$$

Setelah itu dilakukan pengembalian nilai $v_i$. Jika nilai $\bar{v}_i$ positif maka dapat dilakukan Persamaan 3.8 atau Persamaan 3.9 sesuai dengan *location map*-nya. Jika negatif maka dilakukan Persamaan 3.18 atau Persamaan 3.19 sesuai dengan *location map*-nya.

$$v = \bar{v} - 2^{\log_2|\bar{v}|-1} \tag{3.18}$$

$$v = \bar{v} - 2^{\log_2|\bar{v}|} \tag{3.19}$$

## 3.5 Perancangan Antarmuka

Antarmuka perangkat lunak dibuat agar mempermudah pengguna dalam menggunakan perangkat lunak. Desain antarmuka dibuat menggunakan *qtcreator* yang merupakan paket bawaan dari Anaconda3 [17]. Keluaran dari perangkat lunak *qtcreator* adalah

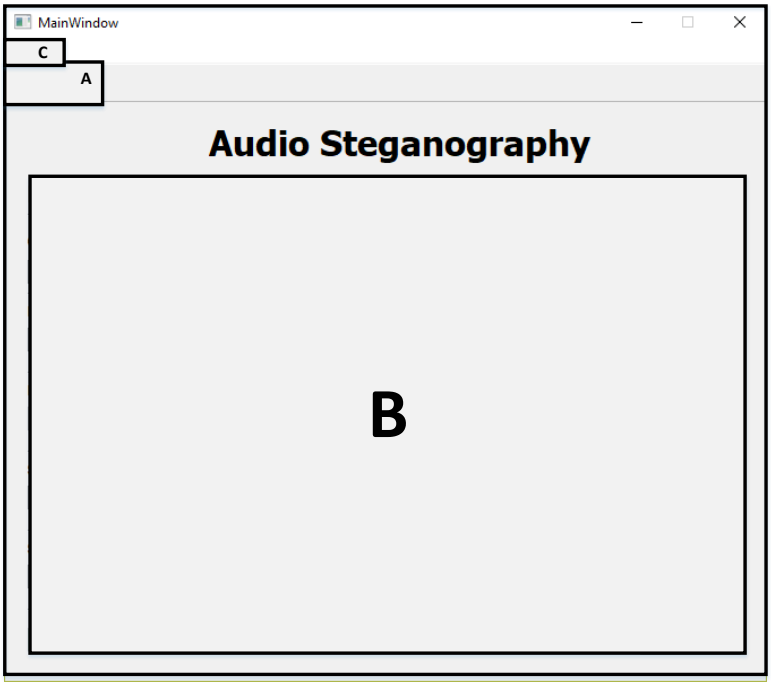
berkas berformat .ui yang setelah itu diubah menjadi berkas kode Python menggunakan perintah yang tersedia pada Anaconda3 [18]. Contoh perintah pengubahan desain antarmuka menjadi berkas kode Python dapat dilihat pada Gambar 3.6.


**Gambar 3.6 Perintah Pengubahan Berkas Desain**

Antarmuka terdiri dari sebuah *MainWindow* yang terdiri dari dua buah *frame*, sebuah *Dialog* yang berisi grafik hasil dan nilai *PSNR*, dan sebuah *Widget* yang berisi form untuk mencocokkan hasil fungsi *hash*. Desain antarmuka dapat dilihat pada Gambar 3.7, Gambar 3.8, dan Gambar 3.9.

Komponen desain antarmuka pada Gambar 3.7 antara lain:

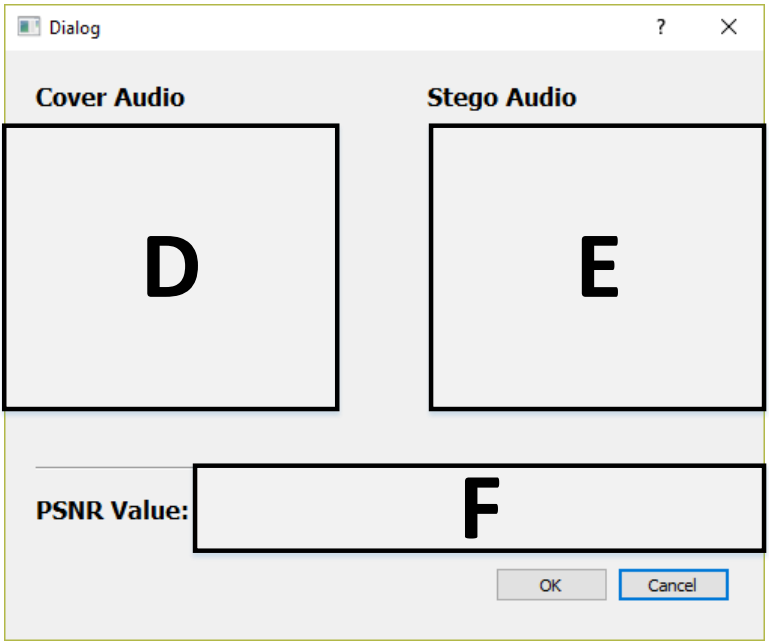
A. Bagian A merupakan *toolbox* yang berisi tombol untuk berpindah antar *frame*.
B. Bagian B merupakan dasar untuk *frame embed* dan *frame extract*.
    a. *Frame embed* berisi form-form data masukan dan data keluaran, daftar metode, tombol-tombol *Browse*, dan tombol *Embed*.
    b. *Frame extract* berisi form-form data masukan dan data keluaran, daftar metode, tombol-tombol *Browse*, dan tombol Extract.
C. Bagian C merupakan *menubar* yang berisi tombol untuk membuka *Widget* fungsi *hash*.




**Gambar 3.7 Desain *MainWindow***

Komponen desain antarmuka pada Gambar 3.8 antara lain:

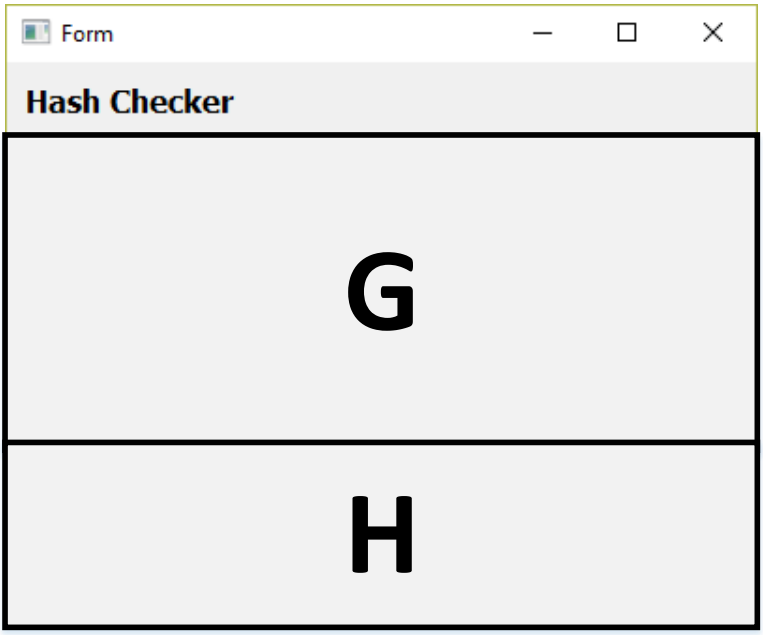
D. Bagian D merupakan gambar grafik sampel *coveraudio.*
E. Bagian E merupakan gambar grafik sampel *stegoaudio.*
F. Bagian F menampilkan hasil perhitungan *PSNR* antara *stegoaudio* dan *coveraudio.*




**Gambar 3.8 Desain *Dialog* Hasil**

Komponen desain antarmuka pada Gambar 3.9 antara lain:

G. Bagian G berisi form-form untuk memasukkan data masukan fungsi *hash* dan tombol-tombol terkait.
H. Bagian H akan menampilkan hasil fungsi *hash* dari kedua audio masukan.




**Gambar 3.9 Desain *Widget* Fungsi *Hash***

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

# BAB IV
# IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi 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 | 8 GB 1333 MHz DDR3 |
| **Perangkat Lunak** | Sistem Operasi | Windows 10 Education |
| | Perangkat Pengembang | PyCharm Professional 2017.1.3 |

## 4.2 Implementasi

Sub bab menjelaskan mengenai pembangunan perangkat lunak. Pembangunan perangkat lunak dilakukan berdasarkan bab perancangan perangkat lunak. *Pseudocode* akan ditampilkan pada bab ini. Bahasa pemrograman yang digunakan adalah Python dengan menggunakan konsep *Object-Oriented Programming*.

### 4.2.1 Modul *takhir*

Modul *takhir* berisi modul-modul yang digunakan untuk membantu proses yang dilakukan oleh perangkat lunak, seperti: pemrosesan berkas audio, pemrosesan pesan tersembunyi, dan penyisipan pesan tersembunyi.

#### 4.2.1.1 Modul *audio*

Modul *audio* berbentuk kelas. Kelas *audio* digunakan sebagai pemroses berkas audio, penyimpan hasil proses berkas audio, dan penulis ulang audio. Detail kelas *audio* dapat dilihat pada Pseudocode 4.1.

```
1   CLASS audio:
2       FUNCTION __init__(self, fileName):
3            fileName <- fileName
4       ENDFUNCTION
5
6       FUNCTION buka(self, bagi):
7           spf <- wave.open( fileName, 'r')
8           signal <- spf.readframes(-1)
9           signal <- np.fromstring(signal, 'uint8')
10          size <- signal.size
11          mod <- divmod(size, bagi)
12          headlen <- size - mod[1]
13           __head <- signal[:headlen]
14           tail <- signal[headlen:]
15      ENDFUNCTION
16
17      FUNCTION writeAudio(self, nama):
18          arr <- np.asarray( head, dtype=np.uint8)
19          tulis <- arr.reshape(1, -1)
20          tulis <- np.append(tulis,  tail)
21          tulis.astype(np.uint8)
22          write(nama, 44100, tulis
23      ENDFUNCTION
24
25  ENDCLASS
```

**Pseudocode 4.1 Kelas *audio***

Detail metode-metode pada kelas *audio* dapat dilihat pada Tabel 4.2.

**Tabel 4.2 Detail Metode Kelas *audio***

| Nama Metode | Keterangan |
|---|---|
| __init__(filename) | Berfungsi untuk menyimpan *path* berkas audio yang ingin dibuka |
| buka(bagi) | Berfungsi untuk membuka berkas audio dan membagi sampel audio menjadi *head* dan *tail* |
| writeAudio(nama) | Berfungsi untuk menulis kembali sampel audio menjadi berkas audio |

#### 4.2.1.2 Modul *pesan*

Modul *pesan* berbentuk kelas. Kelas *pesan* digunakan sebagai pemroses pesan tersembunyi. Di dalam kelas pesan terdapat metode-metode yang digunakan untuk mengubah berkas pesan tersembunyi menjadi *bit* dan juga sebaliknya. Detail kelas *pesan* dapat dilihat pada Pseudocode 4.2.

.

Detail metode-metode kelas *pesan* dapat dilihat pada Tabel 4.3.

**Tabel 4.3 Detail Metode Kelas *pesan***

| Nama Metode | Keterangan |
|---|---|
| __init__(filename) | Berfungsi untuk menyimpan *path* berkas pesan tersembunyi |
| getBinary() | Berfungsi untuk mengembalikan *bit-bit* pesan tersembunyi |

| | |
|---|---|
| bacaPesan() | Berfungsi untuk membaca berkas pesan tersembunyi |
| toBinary() | Berfungsi untuk mengubah berkas pesan tersembunyi menjadi biner-biner |
| writeReturn() | Berfungsi untuk menulis kembali pesan tersembunyi ke berkas pesan tersembunyi |
| toMessage(bitmessage) | Berfungsi untuk mengubah *bit-bit* pesan tersembunyi ke pesan tersembunyi |
| tulisPesan() | Berfungsi untuk menulis pesan tersembunyi ke berkas pesan tersembunyi |

```
1   CLASS Pesan:
2       FUNCTION __init__(self, filename):
3            fileName <- filename
4            ba <- bitarray.bitarray()
5       ENDFUNCTION
6
7       FUNCTION getBinary(self):
8            bacaPesan()
9            toBinary()
10          RETURN  bitmessage
11      ENDFUNCTION
12
13      FUNCTION bacaPesan(self):
14          with open( fileName, mode='rb') as file:
15               fileContent <- file.read()
16           fileContent <-
    bin(int(binascii.hexlify( fileContent), 16))
17      ENDFUNCTION
18
19      FUNCTION toBinary(self):
20           bitmessage <-  ba.fromstring(
    fileContent)
21           bitmessage <-  ba.tolist()
22           panjangbit <- len( bitmessage)
23      ENDFUNCTION
```

```
24
25      FUNCTION writeReturn(self, message):
26           toMessage(message)
27           tulisPesan()
28      ENDFUNCTION
29
30      FUNCTION toMessage(self, bitmessage):
31          lenmes <- len(bitmessage)
32          lenmes <- lenmes % 8
33          IF lenmes = 0:
34              n <-
    int(bitarray.bitarray(bitmessage).tostring(), 2)
35          ELSE:
36              n <-
    int(bitarray.bitarray(bitmessage[:-
    lenmes]).tostring(), 2)
37          ENDIF
38           writeTo <- binascii.unhexlify('%x' % n)
39      ENDFUNCTION
40
41      FUNCTION tulisPesan(self):
42          f <- open( fileName, 'wb')
43          f.write( writeTo)
44          f.close()
45      ENDFUNCTION
46
47  ENDCLASS
```

**Pseudocode 4.2 Kelas *pesan***

#### 4.2.1.3 Modul *transform*

Modul *transform* merupakan *base class* untuk kelas-kelas metode penyembunyian pesan. Kelas ini menyimpan variabel *location map*. Kelas *transform* juga memiliki metode untuk menulis *location map* ke berkas *location map*. Detail kelas *transform* dapat dilihat pada Pseudocode 4.3.

```
1   CLASS transform:
2       FUNCTION __init__(self):
3            __lm <- []
4       ENDFUNCTION
```

```
5
6       @property
7       FUNCTION lm(self):
8           RETURN  __lm
9       ENDFUNCTION
10
11      FUNCTION writelm(self, path):
12          LMFile <- open(path, "wb")
13          LMFileByteArray <- bytes( lm)
14          LMFile.write(LMFileByteArray)
15          LMFile.close()
16      ENDFUNCTION
17
18  ENDCLASS
```

**Pseudocode 4.3 Kelas *transform***

Detail metode-metode kelas *transform* dapat dilihat pada Tabel 4.4.

**Tabel 4.4 Detail Metode Kelas *transform***

| Nama Metode | Keterangan |
|---|---|
| __init__(filename) | Berfungsi untuk menyimpan *path* berkas *location map* |
| writelm(path) | Berfungsi untuk menulis berkas *location map* |

#### 4.2.1.4 Modul *de*

Modul *de* adalah modul yang digunakan untuk menghasilkan *location map* dan untuk menghitung nilai *difference expansion*. Kelas *de* mempunyai metode *expandable*, *changeable*, dan *unchangeable* yang mengeluarkan nilai *location map* dan nilai *difference expansion*. Nilai *location map* sesuai dengan metode yang menghasilkannya. Detail kelas *de* dapat dilihat pada Pseudocode 4.4

```
1   CLASS de(tr.transform):
2       FUNCTION expandable(self, v, m, b):
```

```
3           vtemp <- 2 * v + b
4           IF 128 <= m <= 255:
5               IF abs(vtemp) <= 2 * (255 - m):
6                    lm.append(0)
7                   RETURN vtemp
8               ELSE:
9                   RETURN  changeable(v, m, b)
10              ENDIF
11          ELSEIF 0 <= m <= 127:
12              IF abs(vtemp) <= 2 * m + 1:
13                   lm.append(0)
14                  RETURN vtemp
15              ELSE:
16                  RETURN  changeable(v, m, b)
17              ENDIF
18          ENDIF
19      ENDFUNCTION
20
21      FUNCTION changeable(self, v, m, b):
22          vtemp <- 2 * np.floor(v / 2) + b
23          IF 128 <= m <= 255:
24              IF abs(vtemp) <= 2 * (255 - m):
25                   lm.append(1)
26                  RETURN vtemp
27              ENDIF
28          ELSEIF 0 <= m <= 127:
29              IF abs(vtemp) <= 2 * m + 1:
30                   lm.append(1)
31                  RETURN vtemp
32              ENDIF
33          ELSE:
34              RETURN  unchangeable()
35          ENDIF
36      ENDFUNCTION
37      FUNCTION unchangeable(self):
38           lm.append(2)
39          RETURN "aduh"
40      ENDFUNCTION
41  ENDCLASS
```

**Pseudocode 4.4 Kelas *de***

#### 4.2.1.5 Modul *rde*

Modul *rde* berfungsi sama seperti modul *de*. Perbedaannya adalah tambahan masukan yang berfungsi sebagai penentu *location map* tambahan. Detail kelas *rde* dapat dilihat pada Pseudocode 4.5.

```
1   CLASS rde(tr.transform):
2       FUNCTION expandable(self, v, m, b, rde,
    location_map):
3           vtemp <- 2 * v + b
4           IF 128 <= m <= 255:
5               IF abs(vtemp) <= 2 * (255 - m):
6                   IF rde:
7                       IF location_map:
8                           lm.append(0)
9                       ELSE:
10                          lm.append(1)
11                      ENDIF
12                  ELSE:
13                       lm.append(2)
14                  ENDIF
15                  RETURN vtemp
16              ELSE:
17                  RETURN  changeable(v, m, b)
18              ENDIF
19          ELSEIF 0 <= m <= 127:
20              IF abs(vtemp) <= 2 * m + 1:
21                  IF rde:
22                      IF location_map:
23                          lm.append(0)
24                      ELSE:
25                          lm.append(1)
26                      ENDIF
27                  ELSE:
28                       lm.append(2)
29                  ENDIF
30                  RETURN vtemp
31              ELSE:
32                  RETURN  changeable(v, m, b)
33              ENDIF
34          ENDIF
```

```
35      ENDFUNCTION
36
37      FUNCTION changeable(self, v, m, b):
38          vtemp <- 2 * np.floor(v / 2) + b
39          IF 128 <= m <= 255:
40              IF abs(vtemp) <= 2 * (255 - m):
41                   lm.append(3)
42                  RETURN vtemp
43              ELSE:
44                  RETURN  unchangeable()
45              ENDIF
46          ELSEIF 0 <= m <= 127:
47              IF abs(vtemp) <= 2 * m + 1:
48                   lm.append(3)
49                  RETURN vtemp
50              ELSE:
51                  RETURN  unchangeable()
52          ENDIF
53              ENDIF
54      ENDFUNCTION
55
56      FUNCTION unchangeable(self):
57           lm.append(4)
58          RETURN "aduh"
59      ENDFUNCTION
60
61  ENDCLASS
```

**Pseudocode 4.5 Kelas *rde***

#### 4.2.1.6 Modul *grde*

Modul ini dipakai oleh metode penyembunyian pesan *generalized difference expansion*, *quad-based difference expansion*, dan *block-overlap difference expansion*. Ke tiga metode tersebut memakai rumus yang sama dalam menentukan nilai *difference expansion* sehingga modul ini dapat dipakai untuk metode-metode tersebut. Detail modul *grde* dapat dilihat pada Pseudocode 4.6.

```
1   CLASS grde(tr.transform):
2       FUNCTION expandable(self, v, b, u0, rde,
    location_map):
3           vtemp <- 2 * v + b
4           IF 0 <= vtemp + u0 <= 255:
5               IF rde:
6                   IF location_map:
7                       lm.append(0)
8                   ELSE:
9                       lm.append(1)
10                  ENDIF
11              ELSE:
12                   lm.append(2)
13                  RETURN vtemp
14              ENDIF
15          ELSE:
16              RETURN  changeable(v, b, u0)
17          ENDIF
18      ENDFUNCTION
19
20      FUNCTION changeable(self, v, b, u0):
21          vtemp <- 2 * np.floor(v / 2) + b
22          IF 0 <= vtemp + u0 <= 255:
23               lm.append(3)
24              RETURN vtemp
25          ELSE:
26              RETURN  unchangeable()
27          ENDIF
28      ENDFUNCTION
29
30      FUNCTION unchangeable(self):
31           lm.append(4)
32          RETURN "aduh"
33      ENDFUNCTION
34
35  ENDCLASS
```

**Pseudocode 4.6 Kelas *grde***

#### 4.2.1.7 Modul *smooth*

Modul *smooth* berisi metode-metode tambahan untuk mengaplikasikan metode *quad smoothness reduced difference expansion*. Kelas *smooth* merupakan *derived class* dari kelas *grde*. Detail kelas *smooth* dapat dilihat pada Pseudocode 4.7.

```
1   CLASS smooth(g.grde):
2       FUNCTION getquadavg(self, head):
3           quad <- head.reshape((-1, 4))
4           quadavg <- np.mean(quad, axis=1,
    dtype=np.float64)
5           quadavgsize <- quadavg.size
6           quadavgmod <- divmod(quadavgsize, 4)
7           quadavgheadlen <- quadavgsize -
    quadavgmod[1]
8            quadavghead <- quadavg[:quadavgheadlen]
9            quadavgtail <- quadavg[quadavgheadlen:]
10      ENDFUNCTION
11
12      FUNCTION getsmoothness(self):
13          smooth <-  quadavghead.reshape((-1, 4))
14          smoothness <- np.var(smooth, axis=1)
15          smoothness <- np.array([smoothness])
16          smoothnesssize <- smoothness.size
17          x <- np.arange(smoothnesssize)
18          x <- x.reshape((-1, 1))
19          smoothlist <- np.concatenate((x,
    smoothness.T), axis=1)
20          smoothlist <- smoothlist.tolist()
21          smoothlist.sort(key=lambda x: x[1],
    reverse=False)
22          urutan <- np.array(smoothlist)
23          urutan <- np.hsplit(urutan, 2)
24           urutan <- urutan[0].flatten()
25          RETURN smoothlist
26      ENDFUNCTION
27      FUNCTION saveurutan(self, path):
28          np.savetxt(path,  urutan, fmt='%i')
29      ENDFUNCTION
30  ENDCLASS
```

**Pseudocode 4.7 Kelas *smooth***

Variabel *urutan* didapat dari mengurutkan hasil perhitungan *variance* variabel *smoothness*. Variabel *smoothness* adalah sebuah *array* tiga dimensi berbentuk (-1, 4, 4). *Smoothness* sebuah *array* yang berisi kumpulan dari empat buah *quad*. Sebuah *quad* berisi empat sampel audio. Dengan adanya variabel *urutan* diharapkan dapat menaikkan nilai *PSNR*.

### 4.2.2 Modul *SIP*

Modul *SIP* berisi modul-modul yang memulai proses *embedding* dan *extraction.*

#### 4.2.2.1 Modul *embed*

Modul *embed* merupakan *base class* untuk semua kelas metode penyembunyian pesan tersembunyi. Kelas *embed* berisi metode-metode yang berfungsi untuk melakukan penyembunyian pesan tersembunyi. Detail kelas *embed* dapat dilihat pada Pseudocode 4.8.

```
1   CLASS embed:
2       FUNCTION getCover(self, path_cover, split):
3            cvr <- audio.audio(path_cover)
4            cvr.buka(split)
5            vectorArray <-  cvr.head
6       ENDFUNCTION
7
8       FUNCTION reshapeCover(self, shape):
9            vectorArray <-
    vectorArray.reshape(shape)
10      ENDFUNCTION
11
12      FUNCTION getPesan(self, path_message):
13          psn <- pesan.Pesan(path_message)
14           teks <- psn.getBinary()
15           panjangteks <- psn.panjangbit
16      ENDFUNCTION
17
18      FUNCTION writeStego(self, path_stego):
```

```
19          cvr.head <-  vectorArray
20          cvr.writeAudio(path_stego)
21      ENDFUNCTION
22
23      FUNCTION expandVector(self, m, vtemp, x):
24          uaksen1 <- m + np.floor((vtemp + 1) / 2)
25          uaksen2 <- m - np.floor(vtemp / 2)
26           vectorArray[x] <- [uaksen1, uaksen2]
27      ENDFUNCTION
28
29      FUNCTION reduce(self, v):
30          IF -2 < v < 2:
31              vr <- v
32              rde <- False
33          ELSEIF v <= -2:
34              vr <- v + 2 **
    (np.floor(np.log2(np.absolute(v))) - 1)
35              rde <- True
36          ELSEIF v >= 2:
37              vr <- v - 2 **
    (np.floor(np.log2(np.absolute(v))) - 1)
38              rde <- True
39          ENDIF
40          RETURN vr, rde
41      ENDFUNCTION
42      FUNCTION cekLM(self, vr, v):
43          IF 2 **
44  (np.floor(np.log2(np.absolute(vr)))) = 2 **
    (np.floor(np.log2(np.absolute(v)))):
                RETURN True
45          ELSE:
46              RETURN False
47          ENDIF
48      ENDFUNCTION
49      FUNCTION cekTeks(self, j):
50          IF j <  panjangteks:
51              b <- int( teks[j])
52          ELSE:
53              b <- 0
54          ENDIF
55          RETURN b
56      ENDFUNCTION
57  ENDCLASS
```

**Pseudocode 4.8 Kelas *embed***

Detail metode-metode kelas *embed* dapat dilihat pada Tabel 4.5.

**Tabel 4.5 Detail Metode Kelas *embed***

| **Nama Metode** | **Keterangan** |
|---|---|
| getCover(path_cover, split) | Berfungsi untuk membuka berkas *coveraudio* dan mengubahnya ke *array numpy* |
| reshapeCover(shape) | Berfungsi untuk mengubah bentuk *array* dari *coveraudio* |
| getPesan(path_message) | Berfungsi untuk membuka berkas pesan tersembunyi dan mengubahnya menjadi *list* yang berisi *bit-bit* pesan tersebut |
| writeStego(path_stego) | Berfungsi untuk menulis berkas *stegoaudio* setelah proses penyisipan selesai |
| expandVector(m, vtemp, x) | Berfungsi untuk mengubah nilai sampel dari *coveraudio* |
| reduce(v) | Berfungsi untuk mengubah nilai *v* menjadi *vr* |
| cekLM(vr, v) | Berfungsi untuk mengeluarkan nilai *location map* |
| cekTeks(j) | Berfungsi untuk memeriksa *bit* pesan tersembunyi |

#### 4.2.2.2 Modul *embed_de*

Pada kelas *embed_de* penyisipan pesan tersembunyi dilakukan dengan metode *difference expansion*. Sampel-sampel audio dibentuk menjadi *array* dua dimensi berbentuk (n, 2). *Array* tersebut bernama *vectorArray*. Tiap *vector* dalam array berisi dua sampel audio. Sampel-sampel tersebut dihitung beda dan rata-ratanya. Masing-masing disimpan dalam variabel *v*, dan *m*. *Bit* pesan tersembunyi disimpan dalam variabel *b*. Setelah itu dilakukan penyisipan dengan memanggil fungsi *expandable*. *Location map* dibuat dalam fungsi *expandable*. Detail *location map* dapat dilihat pada Tabel 4.6 *Location Map DE*. Hasil fungsi dari *expandable* adalah *vtemp*, yaitu hasil penyisipan *b* dalam *v*. Setelah itu dilakukan pengubahan *vector* dengan memanggil fungsi *expandVector*.

**Tabel 4.6 *Location Map DE***

| **Nilai *Location Map*** | **Jenis *Location Map*** | **Keterangan** |
|---|---|---|
| 0 | *Expandable* | Dapat disisipi |
| 1 | *Changeable* | Dapat disisipi |
| 2 | *Unchangeable* | Tidak dapat disisipi |

```
1   CLASS embed_de(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3               getCover(path_cover, 2)
4               reshapeCover((-1, 2))
5               getPesan(path_message)
6               expand <- de.de()
7               embed()
8               writeStego(path_stego)
9               expand.writelm(path_lm)
10      ENDFUNCTION
11
12      FUNCTION embed(self):
13          x <- 0
14          j <- 0
```

```
15          FOR vector IN  vectorArray:
16              i <- 0
17              IF j <  panjangteks:
18                  WHILE i < 1:
19                      v <- int(vector[i]) -
    int(vector[i + 1])
20                      m <-
    np.floor((int(vector[i]) + int(vector[i + 1])) /
    2)
21                      b <- int( teks[j])
22                      vtemp <-
    expand.expandable(v, m, b)
23                      IF vtemp != "aduh":
24                           expandVector(m, vtemp,
    x)
25                          j += 1
26                      ENDIF
27                      i += 1
28                      x += 1
29                  ENDWHILE
30              ELSE:
31                  BREAK
32              ENDIF
33          ENDFOR
34      ENDFUNCTION
35
36  ENDCLASS
```

**Pseudocode 4.9 Kelas *embed_de***

#### 4.2.2.3 Modul *embed_rde*

Kelas *embed_de* digunakan untuk melakukan penyisipan pesan tersembunyi dengan metode *reduced difference expansion*. Proses *embed* di kelas *embed_rde* hampir sama dengan kelas *embed_de*. Sebelum dilakukan penyisipan pesan tersembunyi, dilakukan perubahan nilai *v*. Tujuan dari perubahan nilai *v* adalah untuk mengurangi besar perubahan sampel setelah penyisipan pesan tersembunyi. Perubahan nilai *v* disimpan dalam variabel *vr*. *Location map* metode ini berbeda dengan *location map* metode sebelumnya (*DE*). Detail *location map* dapat dilihat pada Tabel 4.7. Detail kelas *embed_de* dapat dilihat pada Pseudocode 4.9.

**Tabel 4.7 *Location Map RDE***

| **Nilai *Location Map*** | **Jenis *Location Map*** | **Keterangan** |
|---|---|---|
| 0 | *Reduced Expandable* | *Reduce Map*: *True*; Dapat disisipi |
| 1 | *Reduced Expandable* | *Reduce Map*: *False*; Dapat disisipi |
| 2 | *Expandable* | Dapat disisipi |
| 3 | *Changeable* | Dapat disisipi |
| 4 | *Unchangeable* | Tidak dapat disisipi |

```
1   CLASS embed_rde(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3           getCover(path_cover, 2)
4           reshapeCover((-1, 2))
5           getPesan(path_message)
6           expand <- rde.rde()
7           embed()
8           writeStego(path_stego)
9           expand.writelm(path_lm)
10      ENDFUNCTION
11
12      FUNCTION embed(self):
13          x <- 0
14          j <- 0
15          FOR vector IN  vectorArray:
16              location_map <- False
17              i <- 0
18              IF j <  panjangteks:
19                  WHILE i < 1:
20                      v <- int(vector[i]) -
    int(vector[i + 1])
21                      vr, rde <-  reduce(v)
22                      IF rde AND  cekLM(vr, v):
23                          location_map <- True
24                      ENDIF
25                      m <-
    np.floor((int(vector[i]) + int(vector[i + 1])) /
    2)
26                      b <- int( teks[j])
```

```
27                          vtemp <-
    expand.expandable(vr, m, b, rde, location_map)
28                          IF vtemp != "aduh":
29                               expandVector(m, vtemp,
    x)
30                              j += 1
31                          ENDIF
32                          i += 1
33                          x += 1
34                      ENDWHILE
35                  ELSE:
36                      BREAK
37                  ENDIF
38              ENDFOR
39          ENDFUNCTION
40
41  ENDCLASS
```

**Pseudocode 4.10 Kelas *embed_rde***

#### 4.2.2.4 Modul *embed_grde*

Modul ini digunakan untuk menyisipkan pesan tersembunyi dalam metode *generalized reduced difference expansion*. Beda metode ini dengan metode *RDE* adalah bentuk *vectorArray* yang berbentuk (n, 4). Penghitungan *v* dilakukan dengan menghitung selisih *vector* indeks pertama dengan tiga indeks setelahnya. Perubahan sampel disimpan dalam variabel *uaksen*. Setelah semua sampel dalam *vector* diproses maka *vectorArray* dengan indeks yang sesuai akan diganti dengan *uaksen*. Dalam proses *embed* dipanggil fungsi *cekTeks*_untuk mengecek jumlah *bit* pesan. Hal ini dilakukan agar jumlah *bit* pesan sesuai dengan jumlah *vector*. Detail kelas *embed_grde* dapat dilihat pada Pseudocode 4.11.

```
1   CLASS embed_grde(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3           getCover(path_cover, 4)
```

```
4           reshapeCover((-1, 4))
5           getPesan(path_message)
6           expand <- grde.grde()
7           embed()
8           writeStego(path_stego)
9           expand.writelm(path_lm)
10      ENDFUNCTION
11
12      FUNCTION embed(self):
13          x <- 0
14          j <- 0
15          FOR vector IN  vectorArray:
16              i <- 0
17              uaksen <- [int(vector[0])]
18              IF j <  panjangteks:
19                  WHILE i < 3:
20                      location_map <- False
21                      v <- int(vector[i + 1]) -
    int(vector[0])
22                      vr, rde <-  reduce(v)
23                      IF rde AND  cekLM(vr, v):
24                          location_map <- True
25                      ENDIF
26                      b <-  cekTeks(j)
27                      vtemp <-
    expand.expandable(vr, b, int(vector[0]), rde,
    location_map)
28                      IF vtemp != "aduh":
29                          uaksen.append(vtemp +
    int(vector[0]))
30                          j += 1
31                      ELSE:
32                          uaksen.append(0)
33                          j += 1
34                      ENDIF
35                      i += 1
36                  ENDWHILE
37                  IF 4 IN  expand.lm[-3:]:
38                      j -= i
39                      WHILE i > 0:
40                           expand.lm[-i] <- 4
41                          i -= 1
42                      ENDWHILE
43                  ELSE:
```

```
44                          vectorArray[x] <-
    [uaksen[0], uaksen[1], uaksen[2], uaksen[3]]
45                      ENDIF
46                      x += 1
47                  ELSE:
48                      BREAK
49                  ENDIF
50              ENDFOR
51          ENDFUNCTION
52
53  ENDCLASS
```

**Pseudocode 4.11 Kelas *embed_grde***

#### 4.2.2.5 Modul *embed_quad*

Kelas e*mbed_quad* digunakan untuk menyisipkan pesan tersembunyi dengan metode *quad-based reduced difference expansion*. Beda metode ini dengan metode *GRDE* adalah dalam penghitungan *v*. Pengitungan *v* dilakukan dengan menghitung *vector* indeks ke dua, tiga, dan empat dengan *vector* indeks sebelumnya. Detail kelas *embed_quad* dapat dilihat pada Pseudocode 4.12

```
1   CLASS embed_quad(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3               getCover(path_cover, 4)
4               reshapeCover((-1, 4))
5               getPesan(path_message)
6               expand <- grde.grde()
7               embed()
8               writeStego(path_stego)
9               expand.writelm(path_lm)
10      ENDFUNCTION
11      FUNCTION embed(self):
12          x <- 0
13          j <- 0
14          FOR vector IN  vectorArray:
15              i <- 0
16              uaksen <- [int(vector[0])]
17              IF j <  panjangteks:
```

```
18                  WHILE i < 3:
19                      location_map <- False
20                      v <- int(vector[i + 1]) -
21  int(vector[i])
                        vr, rde <-  reduce(v)
22                      IF rde AND  cekLM(vr, v):
23                          location_map <- True
24                      ENDIF
25                      b <-  cekTeks(j)
26                      vtemp <-
27  expand.expandable(vr, b, int(vector[0]), rde,
    location_map)
                        IF vtemp != "aduh":
28                          uaksen.append(vtemp +
29  uaksen[-1])
                            j += 1
30                      ELSE:
31                          uaksen.append(0)
32                          j += 1
33                      ENDIF
34                      i += 1
35                  ENDWHILE
36                  IF 4 IN  expand.lm[-3:]:
37                      j -= i
38                      WHILE i > 0:
39                           expand.lm[-i] <- 4
40                          i -= 1
41                      ENDWHILE
42                  ELSE:
43                       vectorArray[x] <-
44  [uaksen[0], uaksen[1], uaksen[2], uaksen[3]]
                    ENDIF
45                  x += 1
46              ELSE:
47                  BREAK
48              ENDIF
49          ENDFOR
50      ENDFUNCTION
51  ENDCLASS
```

**Pseudocode 4.12 Kelas *embed_quad***

#### 4.2.2.6 Modul *embed_overlap*

Modul *embed_overlap* digunakan untuk menyisipkan pesan tersembunyi dengan metode *block overlap reduced difference expansion*. Variabel *vectorArray* pada kelas ini memiliki bentuk (n, 1). Penyipan pesan dilakukan secara urutan dilakukan secara urut dari indeks awal sampai akhir. Variabel *avg* adalah nilai indeks pertama dalam *vectorArray*. Variabel *avg* berganti setelah tiga iterasi penyisipan. Variabel *avg* digunakan untuk menghitung *v*. Nilai *v* dihitung dari selisih *vectorArray* setelah indeks *avg* dengan *avg*. Detail kelas *embed_overlap* dapat dilihat pada Pseudocode 4.13.

```
1   CLASS embed_overlap(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3            getCover(path_cover,4)
4            getPesan(path_message)
5            expand <- grde.grde()
6            embed()
7            writeStego(path_stego)
8            expand.writelm(path_lm)
9       ENDFUNCTION
10
11      FUNCTION embed(self):
12          avg <-  vectorArray[0]
13          i, j, k <- 0, 0, 0
14          WHILE j <  panjangteks:
15              location_map <- False
16              IF k = 2:
17                  i += 1
18                  avg <-  vectorArray[i]
19                  k <- 0
20              ENDIF
21              v <- int( vectorArray[i + 1]) - avg
22              vr, rde <-  reduce(v)
23              IF rde AND  cekLM(vr, v):
24                  location_map <- True
25              ENDIF
26              b <- int( teks[j])
```

```
27              vtemp <-  expand.expandable(vr, b,
    avg, rde, location_map)
28               vectorArray[i + 1] <- vtemp + avg
29              i += 1
30              j += 1
31              k += 1
32          ENDWHILE
33      ENDFUNCTION
34
35  ENDCLASS
```

**Pseudocode 4.13 Kelas *embed_overlap***

### 4.2.2.7 Modul *embed_smooth*

Kelas *embed_smooth* digunakan untuk menyisipkan pesan dengan metode *quad smoothness reduced difference expansion*. Beda metode ini dengan metode *quad-RDE* adalah urutan penyisipan pesan tersembunyi. Urutan didapat dari perhitungan *smoothness*. s*moothness* didapatkan dari perhitungan *variance* dari *quad of quad* yang diurutkan dari besar ke kecil. Setelah *smoothness* dihitung maka dilakukan penulisan berkas urutan. Bentuk *vectorArray* dari kelas *embed_smooth* adalah (n, 4, 4). Detail kelas *embed_smooth* dapat dilihat pada Pseudocode 4.14.

```
1   CLASS embed_smooth(em.embed):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_urutan, path_stego):
3           getCover(path_cover, 16)
4           reshapeCover((-1, 4, 4))
5           getPesan(path_message)
6           expand <- smooth.smooth()
7           writeUrutan(path_urutan)
8           embed()
9           writeStego(path_stego)
10          expand.writelm(path_lm)
11      ENDFUNCTION
12
13      FUNCTION writeUrutan(self, path_urutan):
14          expand.getquadavg( cvr.head)
15          smoothlist <-  expand.getsmoothness()
```

```
16          expand.saveurutan(path_urutan)
17      ENDFUNCTION
18
19      FUNCTION embed(self):
20          y <- 0
21          j <- 0
22          WHILE j <  panjangteks:
23              x <- int( smoothlist[y][0])
24              i <- 0
25              WHILE i < 3:
26                  k <- 0
27                  uaksen <- [int(
    vectorArray[x][i][0])]
28                  WHILE k < 3:
29                      location_map <- False
30                      v <- int(
    vectorArray[x][i][k + 1]) - int(
    vectorArray[x][i][k])
31                      vr, rde <-  reduce(v)
32                      IF rde AND  cekLM(vr, v):
33                          location_map <- True
34                      ENDIF
35                      b <-  cekTeks(j)
36                      vtemp <-
    expand.expandable(vr, b, int(
    vectorArray[x][i][0]), rde, location_map)
37                      IF vtemp != "aduh":
38                          uaksen.append(vtemp +
    uaksen[-1])
39                          j += 1
40                      ELSE:
41                          uaksen.append(0)
42                          j += 1
43                      ENDIF
44                      k += 1
45                  ENDWHILE
46                  IF 4 IN  expand.lm[-3:]:
47                      j -= k
48                      WHILE k > 0:
49                           expand.lm[-k] <- 4
50                          k -= 1
51                      ENDWHILE
52                  ELSE:
```

```
53                          vectorArray[x][i] <-
    [uaksen[0], uaksen[1], uaksen[2], uaksen[3]]
54                  ENDIF
55                  i += 1
56              ENDWHILE
57              y += 1
58          ENDWHILE
59      ENDFUNCTION
60
61  ENDCLASS
```

**Pseudocode 4.14 Kelas *embed_smooth***

### 4.2.2.8 Modul *extract*

Modul *extract* merupakan *base class* untuk semua kelas metode *extraction* pesan tersembunyi. Kelas *extract* berisi metode-metode yang berfungsi untuk melakukan penyembunyian pesan tersembunyi. Detail kelas *extract* dapat dilihat pada Pseudocode 4.15.

```
1   CLASS extract:
2       FUNCTION __init__(self):
3           message <- []
4       ENDFUNCTION
5
6       FUNCTION getStego(self, path_stego, split):
7           cvr <- audio.audio(path_stego)
8           cvr.buka(split)
9           vectorArray <-  cvr.head
10      ENDFUNCTION
11
12      FUNCTION reshapeStego(self, shape):
13          vectorArray <-
    vectorArray.reshape(shape)
14      ENDFUNCTION
15
16      FUNCTION getLM(self, path_lm):
17          WITH open(path_lm, mode='rb') AS file:
18              fileContent <- file.read()
19          fileContent <- list(fileContent)
20          ENDWITH
```

```
21      ENDFUNCTION
22
23      FUNCTION cekLM(self, j, v, b):
24          IF  fileContent[j] = 0:
25              v <- np.floor(v / 2)
26          ELSE:
27              IF 0 <= v <= 1:
28                  v <- 1
29              ELSEIF -2 <= v <= -1:
30                  v <- -2
31              ELSE:
32                  v <- 2 * np.floor(v / 2) + b
33          ENDIF
34              ENDIF
35          RETURN v
36      ENDFUNCTION
37
38      FUNCTION cekLMR(self, j, vr, v, b):
39          IF  fileContent[j] = 0:
40              IF vr > 0:
41                  v <- vr + 2 **
    (np.floor(np.log2(np.absolute(vr))) - 1)
42              ELSEIF vr < 0:
43                  v <- vr - 2 **
    (np.floor(np.log2(np.absolute(vr))) - 1)
44              ENDIF
45          ELSEIF  fileContent[j] = 1:
46              IF vr > 0:
47                  v <- vr + 2 **
    np.floor(np.log2(np.absolute(vr)))
48              ELSEIF vr < 0:
49                  v <- vr - 2 **
    np.floor(np.log2(np.absolute(vr)))
50              ENDIF
51          ELSEIF  fileContent[j] = 2:
52              v <- vr
53          ELSE:
54              IF 0 <= v <= 1:
55                  v <- 1
56              ELSEIF -2 <= v <= -1:
57                  v <- -2
58              ELSE:
59                  v <- 2 * np.floor(v / 2) + b
60          ENDIF
```

```
61              ENDIF
62          RETURN v
63      ENDFUNCTION
64
65      FUNCTION returnAudio(self, path_cover):
66           cvr.head <-  vectorArray
67           cvr.writeAudio(path_cover)
68      ENDFUNCTION
69
70      FUNCTION returnMessage(self, path_message):
71          psn <- pesan.Pesan(path_message)
72          message <- np.array( message,
    dtype=bool)
73          message <- message.tolist()
74          psn.writeReturn(message)
75      ENDFUNCTION
76
77  ENDCLASS
```

**Pseudocode 4.15 Kelas *extract***

Detail metode-metode kelas *extract* dapat dilihat pada Tabel 4.8.

**Tabel 4.8 Detail Metode Kelas *extract***

| Nama Metode | Keterangan |
|---|---|
| __init__() | Berfungsi untuk meninisialisasi *array message* |
| getStego(path_stego, split) | Berfungsi untuk membuka berkas *stegoaudio* dan mengubahnya ke *array numpy* |
| reshapeStego(shape) | Berfungsi untuk mengubah bentuk *array* dari *stegoaudio* |
| getLM(path_lm) | Berfungsi untuk membuka berkas *location map* dan mengubahnya menjadi *list* yang *location map* |
| cekLM(j, v, b) | Berfungsi untuk memeriksa sampel pesan dan |

| | |
|---|---|
| | mengembalikannya sesuai dengan *location map* |
| cekLMR(j, vr, v, b) | Berfungsi untuk memeriksa sampel pesan dan mengembalikannya sesuai dengan *location map* dengan metode *RDE* |
| returnAudio(path_cover) | Berfungsi untuk menulis berkas *stegoaudio* setelah proses penyisipan selesai |
| returnMessage(path_message) | Berfungsi untuk menulis berkas pesan tersembunyi setelah proses *extraction* selesai |

#### 4.2.2.9 Modul *extract_de*

Modul *extract_de* digunakan untuk mengambil pesan pada metode *DE*. Kelas extract_*de* merupakan *derived class* dari kelas *extract*. Berkas *stegoaudio* dibuka dan disimpan dalam *array numpy*. *Array* tersebut dibentuk menjadi (n, 2). Setelah itu berkas *location map* dibuka dan disimpan dalam *list*. Dalam proses *extract* dilakukan penghitungan *v*, *m*, dan *b*. Variabel *v* dihitung dari selisih sampel yang ada dalam variabel *vector*. Varibel *m* dihitung dari rata-rata sampel yang ada dalam variabel *vector*. Variabel *b* dihitung dari *bit* terakhir *v*. Nilai *v* dikembalikan ke nilai semula pada fungsi *cekLM*. Nilai kembalian *v* sesuai dengan nilai *location map*. Setelah itu dilakukan proses penulisan *vectorArray*. Setelah proses *extraction* selesai maka dilakukan proses penulisan berkas audio kembalian dan berkas pesan tersembunyi. Detail kelas *extract_de* dapat dilihat pada Pseudocode 4.16.

```
1  CLASS extract_de(ex.extract):
2      FUNCTION __init__(self, path_cover, path_lm,
   path_message, path_stego):
```

```
3           super().__init__()
4            getStego(path_stego, 2)
5            reshapeStego((-1, 2))
6            getLM(path_lm)
7            extract()
8            returnAudio(path_cover)
9            returnMessage(path_message)
10      ENDFUNCTION
11
12      FUNCTION extract(self):
13           message <- []
14          j <- 0
15          FOR vector IN  vectorArray:
16              i <- 0
17              IF j < len( fileContent):
18                  WHILE i < 1:
19                      v <- int(vector[i]) -
    int(vector[i + 1])
20                      m <-
    np.floor((int(vector[i]) + int(vector[i + 1])) /
    2)
21                      b <- v & 1
22                      v <-  cekLM(j, v, b)
23                      IF  fileContent[j] != 2:
24                           vectorArray[j] <- [m +
    np.floor((v + 1) / 2), m - np.floor(v / 2)]
25                           message.append(b)
26                      ENDIF
27                      j += 1
28                      i += 1
29                  ENDWHILE
30              ELSE:
31                  BREAK
32              ENDIF
33          ENDFOR
34      ENDFUNCTION
35
36  ENDCLASS
```

**Pseudocode 4.16 Kelas *extract_de***

#### 4.2.2.10 Modul *extract_rde*

Kelas *extract_rde* digunakan untuk mengambil pesan pada metode dengan metode *reduced difference expansion*. Proses *extract* hampir sama dengan metode *difference expansion*. Pengecekan *location map* pada metode ini berbeda dengan metode sebelumnya. Metode ini menggunakan *cekLMR*. Hal ini dikarenakan metode ini memiliki *location map* yang berbeda. Pengecekan *location map* mengembalikan nilai *v* ke nilai asal sebelum di-*reduce* jika *location map* menunjukkan nilai 0 atau 1. Detail kelas *extract-rde* dapat dilihat pada Pseudocode 4.17.

```
1   CLASS extract_rde(ex.extract):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3           super().__init__()
4            getStego(path_stego, 2)
5            reshapeStego((-1, 2))
6            getLM(path_lm)
7            extract()
8            returnAudio(path_cover)
9            returnMessage(path_message)
10      ENDFUNCTION
11
12      FUNCTION extract(self):
13           message <- []
14          j <- 0
15          FOR vector IN  vectorArray:
16              i <- 0
17              IF j < len( fileContent):
18                  WHILE i < 1:
19                      v <- int(vector[i]) -
    int(vector[i + 1])
20                      m <-
    np.floor((int(vector[i]) + int(vector[i + 1])) /
    2)
21                      b <- v & 1
22                      vr <- np.floor(v / 2)
23                      v <-  cekLMR(j, vr, v, b)
24                      IF  fileContent[j] != 4:
```

```
25                              vectorArray[j] <- [m +
    np.floor((v + 1) / 2), m - np.floor(v / 2)]
26                              message.append(b)
27                          ENDIF
28                          j += 1
29                          i += 1
30                      ENDWHILE
31                  ELSE:
32                      BREAK
33                  ENDIF
34              ENDFOR
35          ENDFUNCTION
36
37  ENDCLASS
```

**Pseudocode 4.17 Kelas *extract_rde***

#### 4.2.2.11 Modul *extract_grde*

Modul *extract_grde* ini digunakan untuk mengambil pesan dengan metode *generalized reduced difference expansion*. Beda metode ini dengan metode *RDE* adalah bentuk *vectorArray* kelas *extract_grde* yang berbentuk (n, 4). Detail kelas *extract_grde* dapat dilihat pada Pseudocode 4.18.

```
1   CLASS extract_grde(ex.extract):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3           super().__init__()
4            getStego(path_stego, 4)
5            reshapeStego((-1, 4))
6            getLM(path_lm)
7            extract()
8            returnAudio(path_cover)
9            returnMessage(path_message)
10      ENDFUNCTION
11
12      FUNCTION extract(self):
13           message <- []
14          x <- 0
15          j <- 0
16          FOR vector IN  vectorArray:
```

```
17              i <- 0
18              uaksen <- [int(vector[0])]
19              IF j < len( fileContent):
20                  WHILE i < 3:
21                      v <- int(vector[i + 1]) -
    int(vector[0])
22                      b <- v & 1
23                      vr <- np.floor(v / 2)
24                      v <-  cekLMR(j, vr, v, b)
25                      uaksen.append(v +
    int(vector[0]))
26                      IF  fileContent[j] != 4:
27                           message.append(b)
28                      ENDIF
29                      j += 1
30                      i += 1
31                  ENDWHILE
32                   vectorArray[x] <- [uaksen[0],
    uaksen[1], uaksen[2], uaksen[3]]
33                  x += 1
34              ELSE:
35                  BREAK
36              ENDIF
37          ENDFOR
38      ENDFUNCTION
39
40  ENDCLASS
```

**Pseudocode 4.18 Kelas *extract_grde***

#### 4.2.2.12 Modul *extract_quad*

Modul *extract_quad* digunakan untuk mengambil pesan dari metode *quad-based reduced difference expansion*. Beda metode ini dengan metode GRDE adalah pada proses penulisan kembali sampel audio. Penulisan sampel dilakukan dengan menggunakan rumus *quad-RDE*. Detail kelas *extract_quad* dapat dilihat pada Pseudocode 4.19.

```
1   CLASS extract_quad(ex.extract):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
```

```
3           super().__init__()
4            getStego(path_stego, 4)
5            reshapeStego((-1, 4))
6            getLM(path_lm)
7            extract()
8            returnAudio(path_cover)
9            returnMessage(path_message)
10      ENDFUNCTION
11
12      FUNCTION extract(self):
13           message <- []
14          x <- 0
15          j <- 0
16          FOR vector IN  vectorArray:
17              i <- 0
18              uaksen <- [int(vector[0])]
19              IF j < len( fileContent):
20                  WHILE i < 3:
21                      v <- int(vector[i + 1]) -
    int(vector[i])
22                      b <- v & 1
23                      vr <- np.floor(v / 2)
24                      v <-  cekLMR(j, vr, v, b)
25                      uaksen.append(v + uaksen[-
    1])
26                      IF  fileContent[j] != 4:
27                           message.append(b)
28                      ENDIF
29                      j += 1
30                      i += 1
31                  ENDWHILE
32                   vectorArray[x] <- [uaksen[0],
    uaksen[1], uaksen[2], uaksen[3]]
33                  x += 1
34              ELSE:
35                  BREAK
36              ENDIF
37          ENDFOR
38      ENDFUNCTION
39
40  ENDCLASS
```

**Pseudocode 4.19 Kelas *extract_quad***

#### 4.2.2.13 Modul *extract_overlap*

Metode pengembalian pesan dengan metode *block overlap rduced difference expansion* dapat dilakukan dengan kelas *extract_overlap*. Pengembalian pesan dilakukan secara urut mulai dari indeks awal *vectorArray* sampai akhir indeks. Variabel *vectorArray* berbentuk (n, 1). Detail kelas *extract_overlap* dapat dilihat pada Pseudocode 4.20.

```
1   CLASS extract_overlap(ex.extract):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_stego):
3           super().__init__()
4            getStego(path_stego, 4)
5            getLM(path_lm)
6            extract()
7            returnAudio(path_cover)
8            returnMessage(path_message)
9       ENDFUNCTION
10
11      FUNCTION extract(self):
12           message <- []
13          avg <-  vectorArray[0]
14          i, j, k <- 0, 0, 0
15          WHILE j < len( fileContent):
16              IF k = 2:
17                  i += 1
18                  avg <-  vectorArray[i]
19                  k <- 0
20              ENDIF
21              v <- int( vectorArray[i + 1]) - avg
22              b <- v & 1
23              vr <- np.floor(v / 2)
24              v <-  cekLMR(j, vr, v, b)
25               vectorArray[i + 1] <- v + avg
26              IF  fileContent[j] != 4:
27                   message.append(b)
28              ENDIF
29              i += 1
30              j += 1
31              k += 1
32          ENDWHILE
```

```
33      ENDFUNCTION
34
35  ENDCLASS
```

**Pseudocode 4.20 Kelas *extract_overlap***

#### 4.2.2.14 Modul *extract_smooth*

Modul *extract_smooth* digunakan untuk mengembalikan pesan dalam metode *quad smoothness reduced difference expansion*. Bentuk *vectorArray* diubah menjadi bentuk (n, 4, 4). Berkas *urutan* diperlukan untuk mengembalikan pesan dan sampel audio dengan benar. Hal ini dilakukan karena penyisipan pesan dilakukan sesuai dengan urutan *smoothness*. Detail kelas *extract_smooth* dapat dilihat pada Pseudocode 4.21.

```
1   CLASS extract_smooth(ex.extract):
2       FUNCTION __init__(self, path_cover, path_lm,
    path_message, path_urutan, path_stego):
3           super().__init__()
4            getStego(path_stego, 16)
5            reshapeStego((-1, 4, 4))
6            getSmooth(path_urutan)
7            getLM(path_lm)
8            extract()
9            returnAudio(path_cover)
10           returnMessage(path_message)
11      ENDFUNCTION
12
13      FUNCTION getSmooth(self, path_urutan):
14           smoothlist <- np.loadtxt(path_urutan,
    delimiter='\n', dtype=int)
15      ENDFUNCTION
16
17      FUNCTION extract(self):
18           message <- []
19          y <- 0
20          j <- 0
21          WHILE j < len( fileContent):
22              x <-  smoothlist[y]
23              i <- 0
24              WHILE i < 3:
```

```
25                  k <- 0
26                  uaksen <- [int(
    vectorArray[x][i][0])]
27                  WHILE k < 3:
28                      v <- int(
    vectorArray[x][i][k + 1]) - int(
    vectorArray[x][i][k])
29                      b <- v & 1
30                      vr <- np.floor(v / 2)
31                      v <-  cekLMR(j, vr, v, b)
32                      uaksen.append(v + uaksen[-
    1])
33                      IF  fileContent[j] != 4:
34                           message.append(b)
35                      ENDIF
36                      j += 1
37                      k += 1
38                  ENDWHILE
39                   vectorArray[x][i] <-
    [uaksen[0], uaksen[1], uaksen[2], uaksen[3]]
40                  i += 1
41              ENDWHILE
42              y += 1
43          ENDWHILE
44      ENDFUNCTION
45
46  ENDCLASS
```

**Pseudocode 4.21 Kelas *extract_smooth***

### 4.2.3 Pembuatan Antarmuka

Antarmuka diimplementasikan menggunakan pustaka pyQt. Implementasi dilakukan dengan menggunakan perangkat lunak qtdesigner yang merupakan perangka lunak bawaan dari Anaconda3.

Antarmuka dibuat dengan cara *drag and drop* objek terkait yang ada dalam *Widget Box* pada qtdesigner. Setelah itu antarmuka disimpan dalam format .ui. Untuk mengubah berkas .ui ke berkas Python maka dilakukan perintah pengubahan pada perangkat lunak *Windows Powershell*. Perintah pengubahan dapat dilihat pada Pseudocode 4.22.

```
1  pyuic5 -x ui.ui -o ui.py
2  pyuic5 -x finishdialog.ui -o finishdialog.py
3  pyuic5 -x hash.ui -o hashform.py
```

**Pseudocode 4.22 Perintah Pengubahan Antarmuka**

#### 4.2.3.1 Kelas *Stegano*

Kelas *stegano* merupakan kelas *main* pada GUI. Semua berkas kelas GUI dipanggil pada kelas ini. Modul-modul *embedding* dan *extraction* dipanggil pada kelas ini. Terdapat juga modul-modul pendukung seperti modul *hash_form*, *psnr*, dan *graph_data*. Detail kelas *Stegano* dapat dilihat pada Pseudocode 4.23.

```
1  CLASS Stegano(Ui_MainWindow):
2      FUNCTION hashCheck(self):
3          audio_1 <-
   hash_window.path_1.toPlainText()
4          audio_2 <-
   hash_window.path_2.toPlainText()
5          log <- hash_check(audio_1, audio_2)
6          IF log[0] = log[1]:
7              result <- "Cocok"
8          ELSE:
9              result <- "Tidak Cocok"
10         ENDIF
11          hash_window.log.setPlainText(log[0] +
   "\n" + log[1] + "\n" + result)
       ENDFUNCTION
12
13     FUNCTION embedding(self):
14         path_cover <-  path_cover.toPlainText()
15         path_lm <-  path_lm.toPlainText()
16         path_message <-  message.toPlainText()
17         path_urutan <-
18 path_smoothness.toPlainText()
           path_stego <-
19 path_stegoaudio.toPlainText()
20         TRY:
21             IF  method.currentIndex() = 0:
```

```
22                  embed_de(path_cover, path_lm,
    path_message, path_stego)
23              ELSEIF  method.currentIndex() = 1:
24                  embed_rde(path_cover, path_lm,
    path_message, path_stego)
25              ELSEIF  method.currentIndex() = 2:
26                  embed_grde(path_cover, path_lm,
    path_message, path_stego)
27              ELSEIF  method.currentIndex() = 3:
28                  embed_quad(path_cover, path_lm,
    path_message, path_stego)
29              ELSEIF  method.currentIndex() = 4:
30                  embed_overlap(path_cover,
    path_lm, path_message, path_stego)
31              ELSEIF  method.currentIndex() = 5:
32                  embed_smooth(path_cover,
    path_lm, path_message, path_urutan, path_stego)
33              ENDIF
34               progress_bar.setProperty("value",
    100)
35               showFinish(path_cover, path_stego)
36          EXCEPT IOError:
37              OUTPUT "Error: can\'t find file or
    read data"
38          ELSE:
39              OUTPUT "whoops"
40          ENDTRY
41      ENDFUNCTION
42
43      FUNCTION extracting(self):
44          path_cover <-
    path_cover_2.toPlainText()
45          path_lm <-  path_lm_2.toPlainText()
46          path_message <-  message_2.toPlainText()
47          path_urutan <-
    path_smoothness_2.toPlainText()
48          path_stego <-
    path_stegoaudio_2.toPlainText()
49          TRY:
50              IF  method_2.currentIndex() = 0:
51                  extract_de(path_cover, path_lm,
    path_message, path_stego)
52              ELSEIF  method_2.currentIndex() = 1:
53
```

```
                    extract_rde(path_cover, path_lm,
54  path_message, path_stego)
55              ELSEIF  method_2.currentIndex() = 2:
                    extract_grde(path_cover,
56  path_lm, path_message, path_stego)
57              ELSEIF  method_2.currentIndex() = 3:
                    extract_quad(path_cover,
58  path_lm, path_message, path_stego)
59              ELSEIF  method_2.currentIndex() = 4:
                    extract_overlap(path_cover,
60  path_lm, path_message, path_stego)
61              ELSEIF  method_2.currentIndex() = 5:
                    extract_smooth(path_cover,
62  path_lm, path_message, path_urutan, path_stego)
63              ENDIF
                 progress_bar_2.setProperty("value",
64  100)
65               showHash()
66          EXCEPT IOError:
                OUTPUT "Error: can\'t find file or
67  read data"
68          ELSE:
69              OUTPUT "whoops"
70          ENDEXCEPT
71      ENDFUNCTION
72
        FUNCTION showFinish(self, path_cover,
73  path_stego):
            finish_window.setPath(path_cover,
74  path_stego)
            finish_window.setPsnr(path_cover,
    path_stego)
75          finish_dialog.show()
76      ENDFUNCTION
77
78      FUNCTION showHash(self):
79          hash_form.show()
80      ENDFUNCTION
81
82  ENDCLASS
```

**Pseudocode 4.23 Kelas *Stegano***

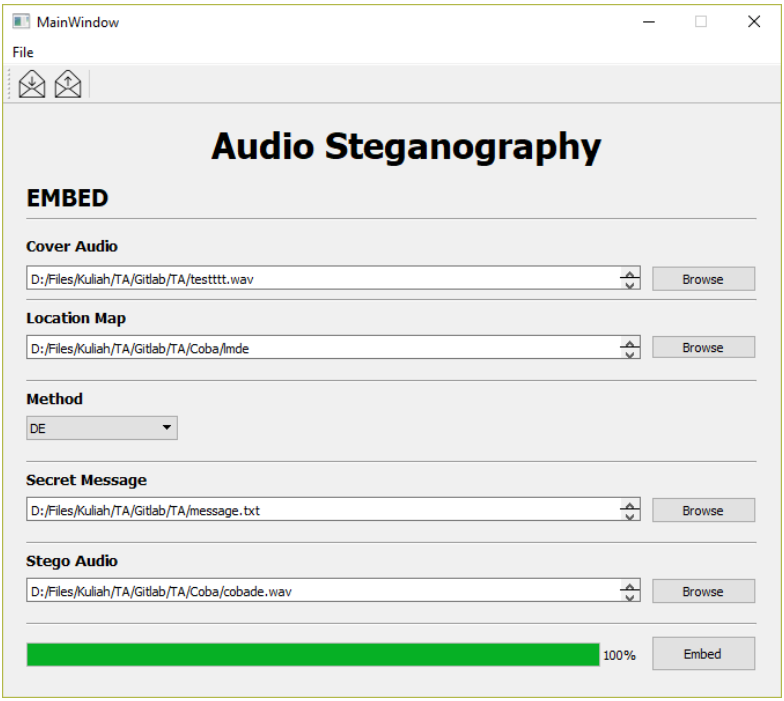
Detail metode-metode pada kelas *Stegano* dapat dilihat pada Tabel 4.2.

| **Nama Metode** | **Keterangan** |
|---|---|
| hash_check() | Berfungsi untuk menghitung *psnr* antara dua berkas audio |
| embedding() | Berfungsi untuk memulai proses *embedding* |
| extraction() | Berfungsi untuk memulai proses *extraction* |
| showFinish() | Berfungsi untuk menampilkan *dialog finishdialog* |
| showHash() | Berfungsi untuk menampilkan *widget hashform* |

**Gambar 4.1 Detail Metode Kelas *Stegano***

#### 4.2.3.2 Kelas *ui*

Kelas *ui* berisi antarmuka pada *window ui*. Antarmuka ini berfungsi sebagai antarmuka utama perangkat lunak. Pada kelas ini ditambahkan metode untuk menghubungkan objek dengan proses-proses terkait. Terdapat tombol-tombol "*Browse*" untuk membuka dialog berkas, *menubar* dengan tombol "*Hash Checker*", *toolbar* dengan gambar yang melambangkan proses *embedding* dan *extraction*, berbagai form-form, tombol "*Embed*", dan tombol "*Extract*". Antarmuka dapat dilihat pada Gambar 4.2.



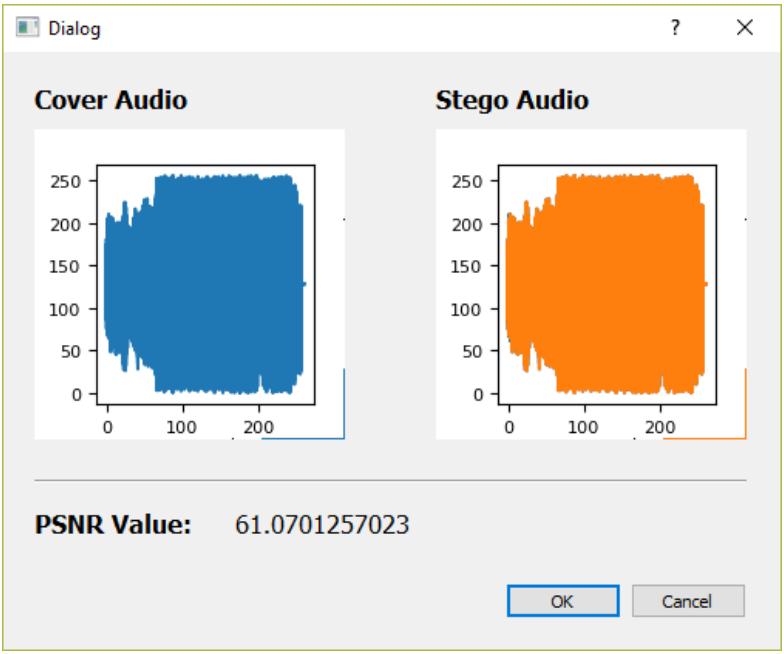

**Gambar 4.2 Antarmuka *ui***

### 4.2.3.3 Kelas *finishdialog*

Kelas *finishdialog* berisi antarmuka dalam *dialog finishdialog*. Dalam antarmuka terdapat dua grafik yang dibuat dengan menggunakan pustaka *matplotlib.pyplot*. Pada kelas ini terdapat metode *setPath* yang digunakan untuk menginisialisasi grafik, metode *plot* yang digunakan untuk membuat grafik, dan metode *psnr* yang digunakan untuk menghitung dan menampilkan nilai *psnr* antara *coveraudio* dan *stegoaudio*. Detail kelas *finishdialog* dapat dilihat pada Pseudocode 4.24. Antarmuka dapat dilihat pada Gambar 4.3.

```
1   CLASS Ui_Dialog(object):
2       FUNCTION setPath(self, path_cover,
    path_stego):
3           plt.rcParams['agg.path.chunksize'] <-
    10000
4           plt.rcParams.update({'font.size': 8})
5            graph_cover <- plt.figure()
6            graph_stego <- plt.figure()
7            canvas_cover <- FigureCanvas(
    graph_cover)
8            canvas_stego <- FigureCanvas(
    graph_stego)
9            cover.addWidget( canvas_cover)
10           stego.addWidget( canvas_stego)
11           plot(path_cover, path_stego)
12      ENDFUNCTION
13      FUNCTION plot(self, path_cover, path_stego):
14          cover <- Graph_Data(path_cover)
15          stego <- Graph_Data(path_stego)
16          data_cover <- cover.getData()
17          data_stego <- stego.getData()
18           graph_cover.clear()
19           graph_stego.clear()
20          # create an axis
21          ax <-  graph_cover.add_subplot(111)
22          bx <-  graph_stego.add_subplot(111)
23          # plot data
24          ax.plot(data_cover[0], data_cover[1])
25          bx.plot(data_stego[0], data_stego[1])
26          bx.plot(data_cover[0], data_cover[1])
27          # refresh canvas
28           graph_cover.subplots_adjust(left=0.2)
29           graph_stego.subplots_adjust(left=0.2)
30           canvas_cover.draw()
31           canvas_stego.draw()
32      ENDFUNCTION
33      FUNCTION setPsnr(self, path_cover,
34  path_stego):
35          set <- psnr(path_cover, path_stego)
             label_4.setText(set.getPsnr())
36      ENDFUNCTION
37  ENDCLASS
```

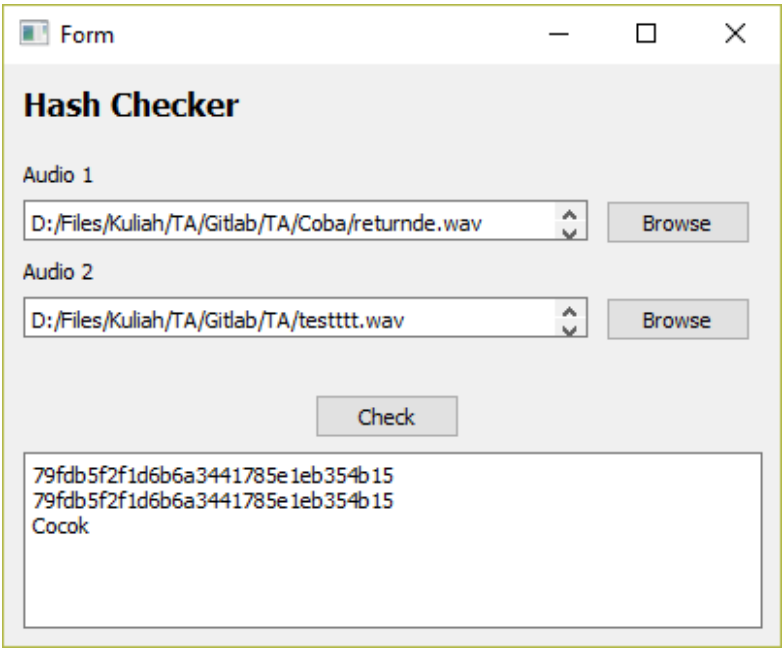
**Pseudocode 4.24 Kelas *finishdialog***




**Gambar 4.3 Antarmuka *finishdialog***

#### 4.2.3.4 Kelas *hashform*

Kelas *hashform* berisi antarmuka dari *widget hashform*. Pada kelas ini terdapat form-form *path* audio, tombol-tombol "*Browse*", sebuah tombol "*Check*", dan sebuah *QPlainTextEdit* yang akan menampilkan hasil fungsi *hash* kedua audio masukan. Antarmuka dapat dilihat pada Gambar 4.4.




**Gambar 4.4 Antarmuka *hashform***

### 4.2.4 Kelas *graphdata*

Kelas *graphdata* digunakan untuk mendapatkan data yang digunakan untuk membuat grafik. Data yang dikeluarkan adalah *Time* dan *signal*. *Time* adalah waktu dalam detik. *signal* adalah data sampel audio. Detail kelas *graphdata* dapat dilihat pada Pseudocode 4.25

| | |
|---|---|
| I | String *path* berkas audio |
| O | *Array Time* dan *array signal* |
| 1 | **CLASS** Graph_Data: |
| 2 | **FUNCTION** __init__(self, path): |
| 3 | spf <- wave.open(path, 'r') |
| 4 | signal <- spf.readframes(-1) |
| 5 | signal <- np.fromstring(signal, 'uint8') |
| 6 | fs <- spf.getframerate() |

```
7           IF spf.getnchannels() = 2:
8               OUTPUT 'Just mono files'
9               sys.exit(0)
10          ENDIF
11           Time=np.linspace(0, len(signal)/fs,
num=len(signal))
12           signal <- signal
13      ENDFUNCTION
14
15      FUNCTION getData(self):
16          RETURN  Time,  signal
17      ENDFUNCTION
```

**Pseudocode 4.25 Kelas *graphdata***

### 4.2.5 Kelas *psnsr*

Kelas *psnr* digunakan untuk mendapatkan nilai *psnr* dari dua berkas audio. Kelas ini menggunakan pustaka *numpy* untuk membantu proses penghitungan.

```
I   String path coveraudio, path stegoaudio
O   Nilai psnr
1   CLASS psnr:
2       FUNCTION __init__(self, cover, stego):
3           spf <- wave.open(cover, 'r')
4           read <- spf.readframes(-1)
5           A <- np.fromstring(read, 'uint8')
6           spf <- wave.open(stego, 'r')
7           read <- spf.readframes(-1)
8           B <- np.fromstring(read, 'uint8')
9            mse <- ((A - B) ** 2).mean(axis=None)
10           psnr <- 20 * np.log10(255 / np.sqrt(
    mse))
11      ENDFUNCTION
12
13      FUNCTION getPsnr(self):
14          RETURN psnr
15      ENDFUNCTION
```

**Pseudocode 4.26 Kelas *psnr***

### 4.2.6 Fungsi *hash_check*

Fungsi *hash_check* mengembalikan nilai fungsi *hash* dari dua berkas audio.

| | |
|---|---|
| I | String *path* dua berkas audio |
| O | Nilai *hash* dua berkas audio |

```
1   FUNCTION hash_check(audio_1, audio_2):
2       before <- hashlib.md5()
3       WITH open(audio_1, 'rb') AS afile:
4           buf <- afile.read()
5           before.update(buf)
6       ENDWITH
7       after <- hashlib.md5()
8       WITH open(audio_2, 'rb') AS afile:
9           buf <- afile.read()
10          after.update(buf)
11      ENDWITH
12      before <- before.hexdigest()
13      after <- after.hexdigest()
14      RETURN before, after
15  ENDFUNCTION
```

**Pseudocode 4.27 Fungsi *hash_check***

# 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 5.1.

**Tabel 5.1 Lingkungan Pengujian**

| Perangkat | Jenis Perangkat | Spesifikasi |
|---|---|---|
| **Perangkat Keras** | Prosesor | Intel® Core™ i5 3230M 2.60 GHZ |
| | Memori | 8 GB 1333 MHz DDR3 |
| **Perangkat Lunak** | Sistem Operasi | Windows 10 Education |
| | Perangkat Pengembang | PyCharm Professional 2017.1.3 |

## 5.2 Data Pengujian

Data pengujian yang digunakan untuk uji coba perangkat lunak penyisipan pesan tersembunyi adalah berkas audio dengan format .wav dan mempunyai kedalaman 8-*bit*, berkas pesan tersembunyi dengan format .txt. Detail tentang data pengujian dapat dilihat pada Tabel 5.2 dan Tabel 5.3.

**Tabel 5.2 Data Berkas Audio**

| No. | Nama Berkas | Ukuran Berkas | Durasi Berkas |
|---|---|---|---|
| 1 | Careless Whisper.wav | 4052 KB | 94 detik |
| 2 | Darah Muda.wav | 4049 KB | 94 detik |
| 3 | Forever in Love.wav | 4049 KB | 94 detik |
| 4 | Wind of Change.wav | 4049 KB | 94 detik |

**Tabel 5.3 Data Berkas Pesan Tersembunyi**

| No. | Nama Berkas | Ukuran Berkas | Panjang Berkas |
|---|---|---|---|
| 1 | 25.txt | 25 kb | 3141 karakter |
| 2 | 50.txt | 50 kb | 6286 karakter |
| 3 | 100.txt | 100 kb | 12580 karakter |
| 4 | 200.txt | 200 kb | 25164 karakter |

## 5.3 Skenario Uji Coba

Metode-metode penyembunyian pesan tersembunyi pada berkas audio diuji dengan memperhatikan kualitas *stegoaudio* dan kapasitas *coveraudio*. Beberapa skenario uji coba akan dilakukan dengan membandingkan PSNR pada berkas-berkas *coveraudio* dengan ukuran berkas tersembunyi yang berbeda-beda. Skenario-skenario uji coba dapat dilihat pada Tabel 5.4.

**Tabel 5.4 Skenario-skenario Uji Coba**

| **Skenario** | **Ukuran Berkas** |
|---|---|
| Uji Coba 1 | 25 kb |
| Uji Coba 2 | 50 kb |
| Uji Coba 3 | 100 kb |
| Uji Coba 4 | 200 kb |
| Uji Coba 5 | Kapasitas maksimal *coveraudio* |

### 5.3.1 Skenario Uji Coba 1

Skenario uji coba 1 adalah perhitungan nilai *PSNR* dari semua *stegoaudio* yang telah disisipi pesan tersembunyi sebesar 25 kb. Hasil penghitungan PSNR dapat dilihat pada Tabel 5.5.

**Tabel 5.5 Hasil Uji Coba Skenario 1**

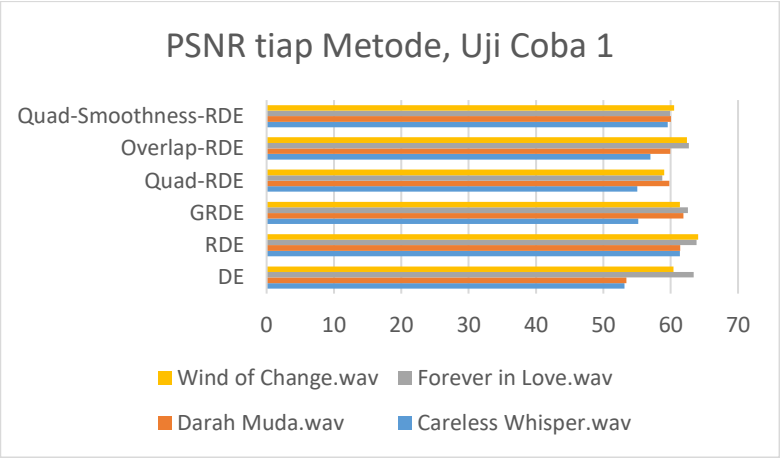
| **Coveraudio** | **Metode** | **PSNR** |
|---|---|---|
| Careless Whisper.wav | DE | 53.127 |
| | RDE | 61.356 |
| | GRDE | 55.179 |
| | Quad-RDE | 55.032 |
| | Overlap-RDE | 56.983 |
| | Quad-Smoothness-RDE | 59.572 |
| Rata-rata | | 56.874 |
| Darah Muda.wav | DE | 53.427 |
| | RDE | 61.397 |
| | GRDE | 61.905 |
| | Quad-RDE | 59.786 |
| | Overlap-RDE | 59.953 |
| | Quad-Smoothness-RDE | 60.050 |
| Rata-rata | | 59.419 |
| Forever in Love.wav | DE | 63.411 |
| | RDE | 63.840 |
| | GRDE | 62.575 |

| | | |
|---|---|---|
| | Quad-RDE | 58.744 |
| | Overlap-RDE | 62.715 |
| | Quad-Smoothness-RDE | 59.897 |
| Rata-rata | | 61.863 |
| Wind of Change.wav | DE | 60.394 |
| | RDE | 64.063 |
| | GRDE | 61.384 |
| | Quad-RDE | 59.040 |
| | Overlap-RDE | 62.437 |
| | Quad-Smoothness-RDE | 60.508 |
| Rata-rata | | 61.463 |
| Rata-rata total | | 59.905 |

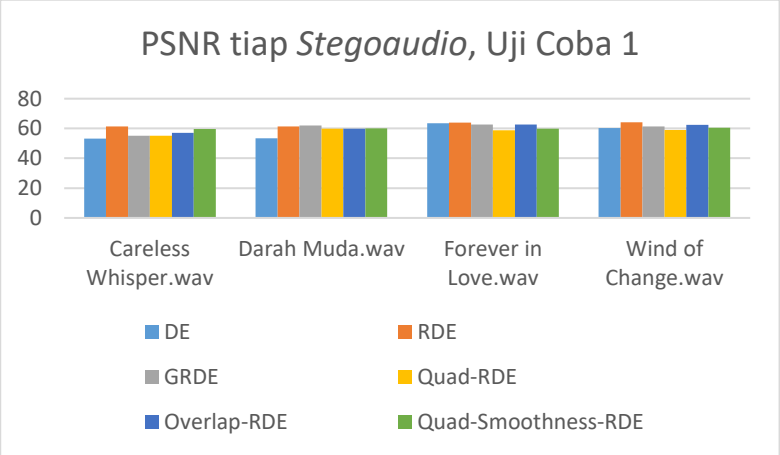
Dari Tabel 5.5 didapatkan rata-rata PSNR sebesar 59.90546849. Pada Gambar 5.1 dan Gambar 5.2 dapat dilihat perbandingan PSNR antar metode dan *stegoaudio*. *Stegoaudio* Forever In Love.wav dan Wind of Change.wav memiliki rata-rata PSNR terbaik.

Gambar 5.3 merupakan grafik dari Wind of Change.wav yang disisipi pesan tersembunyi menggunakan metode RDE. Perbedaan antara *coveraudio* dan *stegoaudio* tidak terlihat jelas dalam grafik.

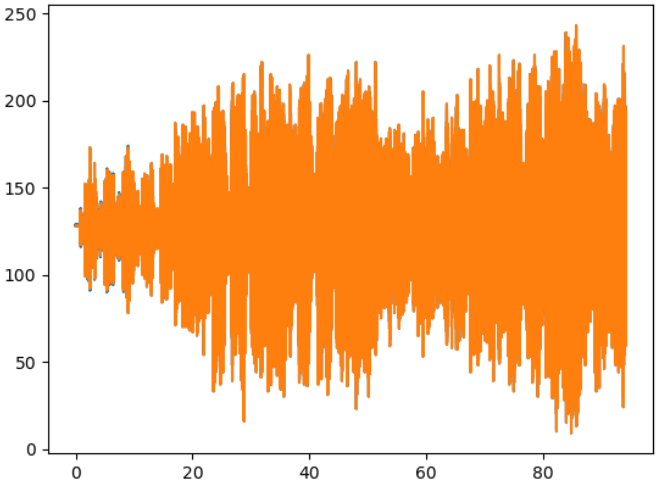
Gambar 5.4 merupakan grafik dari Careless Whisper.wav yang disisipi pesan tersembunyi menggunakan metode DE. Perbedaan antara *coveraudio* dan *stegoaudio* dapat dilihat jelas pada grafik. Grafik biru merupakan beda antara *coveraudio* dan *stegoaudio*.



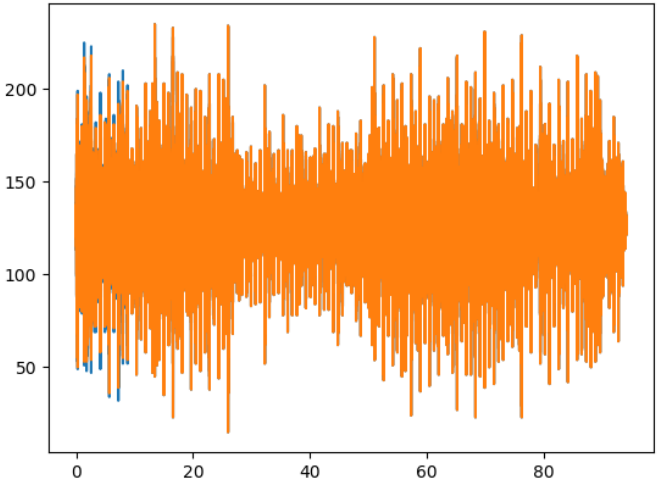

**Gambar 5.1 Grafik PSNR tiap Metode, Uji Coba 1**




**Gambar 5.2 Grafik PSNR tiap *Stegoaudio*, Uji Coba 1**




**Gambar 5.3 Grafik Sampel Audio dengan *PSNR* Tertinggi pada Uji Coba 1**




**Gambar 5.4 Grafik Sampel Audio dengan *PSNR* Terendah pada Uji Coba 1**

### 5.3.2 Skenario Uji Coba 2

Skenario uji coba 2 adalah perhitungan nilai *PSNR* dari semua *stegoaudio* yang telah disisipi pesan tersembunyi sebesar 50 kb. Hasil penghitungan PSNR dapat dilihat pada Tabel 5.6.

**Tabel 5.6 Hasil Uji Coba Skenario 2**

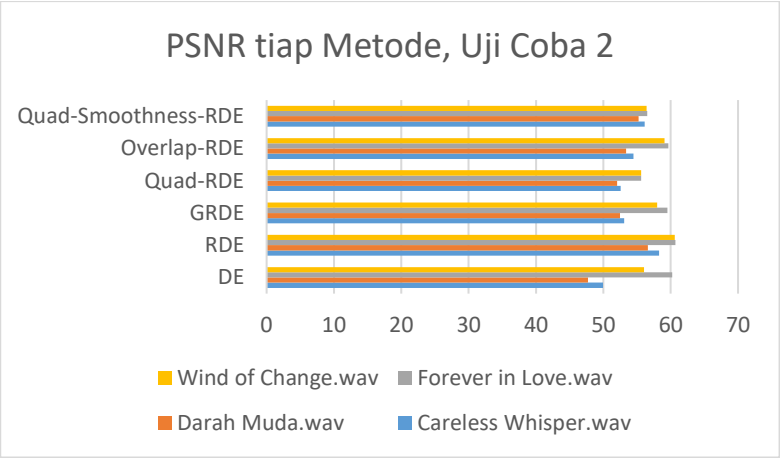
| **Coveraudio** | **Metode** | **PSNR** |
|---|---|---|
| Careless Whisper.wav | DE | 49.970 |
| | RDE | 58.254 |
| | GRDE | 53.096 |
| | Quad-RDE | 52.594 |
| | Overlap-RDE | 54.487 |
| | Quad-Smoothness-RDE | 56.129 |
| Rata-rata | | 54.088 |
| Darah Muda.wav | DE | 47.699 |
| | RDE | 56.596 |
| | GRDE | 52.489 |
| | Quad-RDE | 52.048 |
| | Overlap-RDE | 53.401 |
| | Quad-Smoothness-RDE | 55.241 |
| Rata-rata | | 52.912 |
| Forever in Love.wav | DE | 60.239 |
| | RDE | 60.712 |
| | GRDE | 59.494 |
| | Quad-RDE | 55.630 |
| | Overlap-RDE | 59.633 |
| | Quad-Smoothness-RDE | 56.501 |
| Rata-rata | | 58.701 |
| Wind of Change.wav | DE | 56.051 |
| | RDE | 60.630 |
| | GRDE | 58.005 |
| | Quad-RDE | 55.619 |
| | Overlap-RDE | 59.102 |

| | | |
|---|---|---|
| | Quad-Smoothness-RDE | 56.405 |
| Rata-rata | | 57.635 |
| Rata-rata total | | 55.834 |

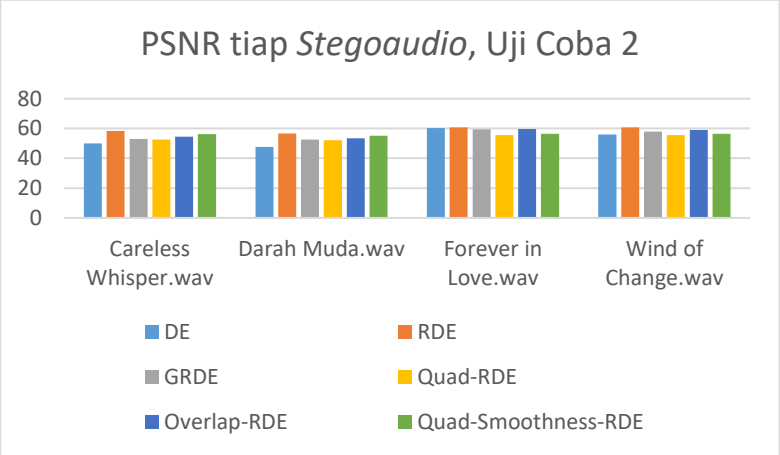
Dari Tabel 5.6 didapatkan rata-rata PSNR sebesar 55.83429681. Pada Gambar 5.5 dan Gambar 5.6 dapat dilihat perbandingan PSNR antar metode dan *stegoaudio*. *Stegoaudio* Forever In Love.wav memiliki rata-rata PSNR terbaik.

Gambar 5.7 merupakan grafik dari Forever In Love.wav yang disisipi pesan tersembunyi menggunakan metode RDE. Perbedaan antara *coveraudio* dan *stegoaudio* tidak terlihat jelas dalam grafik. Beda antar sampel pada audio ini tidak besar sehingga perubahan *stegoaudio* tidak terlalu besar.

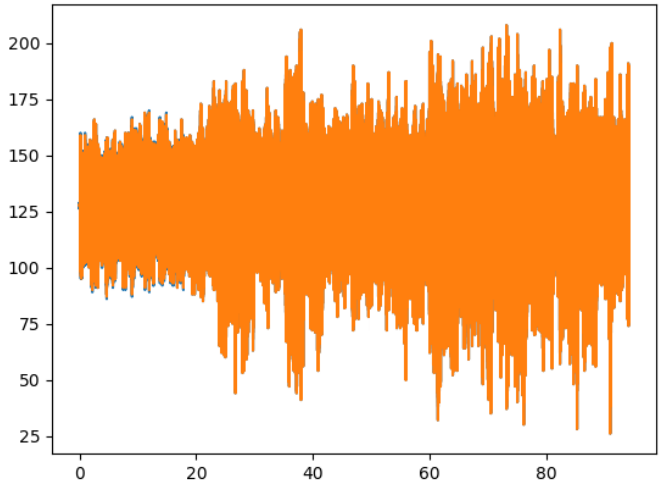
Gambar 5.8 merupakan grafik dari Darah Muda.wav yang disisipi pesan tersembunyi menggunakan metode DE. Perbedaan antara *coveraudio* dan *stegoaudio* dapat dilihat jelas pada grafik. Grafik biru merupakan beda antara *coveraudio* dan *stegoaudio*. Sampel-sampel pada grafik ini memiliki rata-rata yang besar sehingga grafik terlihat penuh. Terdapat juga perbedaan antar sampel yang besar yang terlihat pada grafik yang terlihat seperti bergerigi.



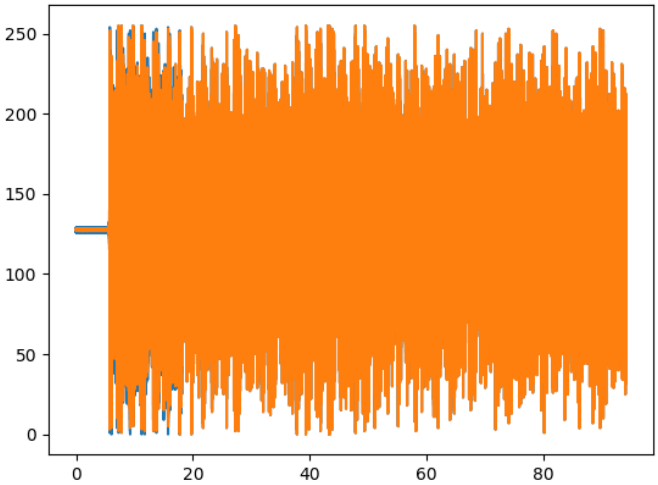

**Gambar 5.5 Grafik PSNR tiap Metode, Uji Coba 2**




**Gambar 5.6 Grafik PSNR tiap Stegoaudio, Uji Coba 2**




**Gambar 5.7 Grafik Sampel Audio dengan PSNR Tertinggi pada Uji Coba 2**




**Gambar 5.8 Grafik Sampel Audio dengan PSNR Terendah pada Uji Coba 2**

### 5.3.3 Skenario Uji Coba 3

Skenario uji coba 3 adalah perhitungan nilai *PSNR* dari semua *stegoaudio* yang telah disisipi pesan tersembunyi sebesar 100 kb. Hasil penghitungan PSNR dapat dilihat pada Tabel 5.7.

**Tabel 5.7 Hasil Uji Coba Skenario 3**

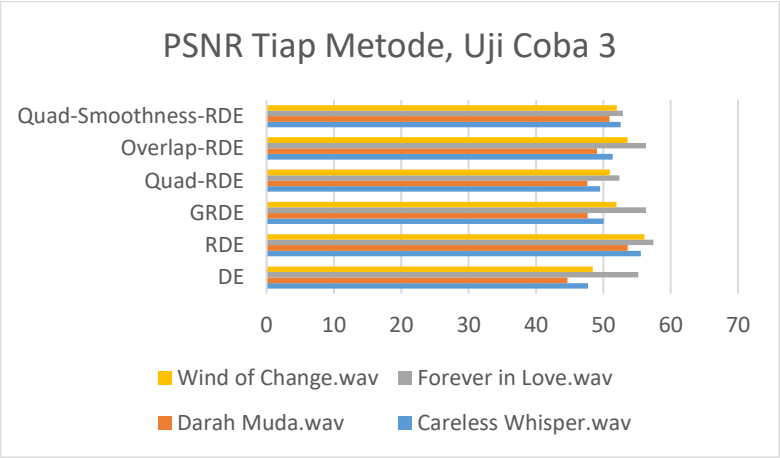
| **Coveraudio** | **Metode** | **PSNR** |
|---|---|---|
| Careless Whisper.wav | DE | 47.731 |
| | RDE | 55.561 |
| | GRDE | 50.028 |
| | Quad-RDE | 49.529 |
| | Overlap-RDE | 51.404 |
| | Quad-Smoothness-RDE | 52.572 |
| Rata-rata | | 51.137 |
| Darah Muda.wav | DE | 44.659 |
| | RDE | 53.617 |
| | GRDE | 47.651 |
| | Quad-RDE | 47.617 |
| | Overlap-RDE | 49.075 |
| | Quad-Smoothness-RDE | 50.909 |
| Rata-rata | | 48.921 |
| Forever in Love.wav | DE | 55.170 |
| | RDE | 57.435 |
| | GRDE | 56.339 |
| | Quad-RDE | 52.370 |
| | Overlap-RDE | 56.305 |
| | Quad-Smoothness-RDE | 52.880 |
| Rata-rata | | 55.083 |
| Wind of Change.wav | DE | 48.436 |
| | RDE | 56.095 |
| | GRDE | 51.962 |
| | Quad-RDE | 50.972 |
| | Overlap-RDE | 53.632 |

| | | |
|---|---|---|
| | Quad-Smoothness-RDE | 51.999 |
| Rata-rata | | 52.182 |
| Rata-rata total | | 51.831 |

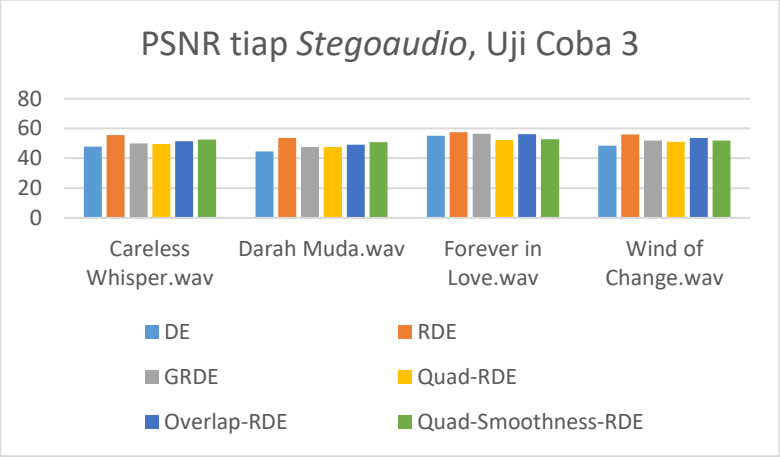
Dari Tabel 5.7 didapatkan rata-rata PSNR sebesar 51.8312636. Pada Gambar 5.9 dan Gambar 5.10 dapat dilihat perbandingan PSNR antar metode dan *stegoaudio*. *Stegoaudio* Forever In Love.wav memiliki rata-rata PSNR terbaik.

Gambar 5.11 merupakan grafik dari Forever In Love.wav yang disisipi pesan tersembunyi menggunakan metode RDE. Perbedaan antara *coveraudio* dan *stegoaudio* tidak terlihat jelas dalam grafik. Beda antar sampel pada audio ini tidak besar sehingga perubahan *stegoaudio* tidak terlalu besar.

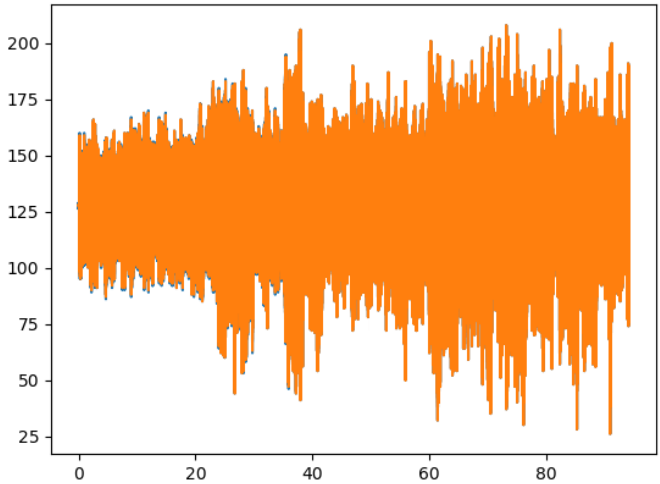
Gambar 5.12 merupakan grafik dari Darah Muda.wav yang disisipi pesan tersembunyi menggunakan metode DE. Perbedaan antara *coveraudio* dan *stegoaudio* dapat dilihat jelas pada grafik. Grafik biru merupakan beda antara *coveraudio* dan *stegoaudio*. Terlihat perbedaan antara *coveraudio* dan *stegoaudio* yang besar pada 20 sampai 40.



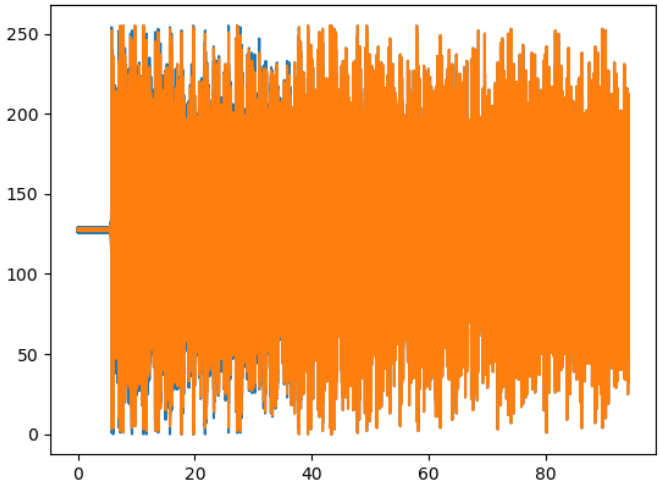

**Gambar 5.9 Grafik PSNR tiap Metode, Uji Coba 3**




**Gambar 5.10 Grafik PSNR tiap Stegoaudio, Uji Coba 3**




**Gambar 5.11 Grafik Sampel Audio dengan PSNR Tertinggi pada Uji Coba 3**




**Gambar 5.12 Grafik Sampel Audio dengan PSNR Terendah pada Uji Coba 3**

### 5.3.4 Skenario Uji Coba 4

Skenario uji coba 4 adalah perhitungan nilai *PSNR* dari semua *stegoaudio* yang telah disisipi pesan tersembunyi sebesar 200 kb. Hasil penghitungan PSNR dapat dilihat pada Tabel 5.8

**Tabel 5.8 Hasil Uji Coba Skenario 4**

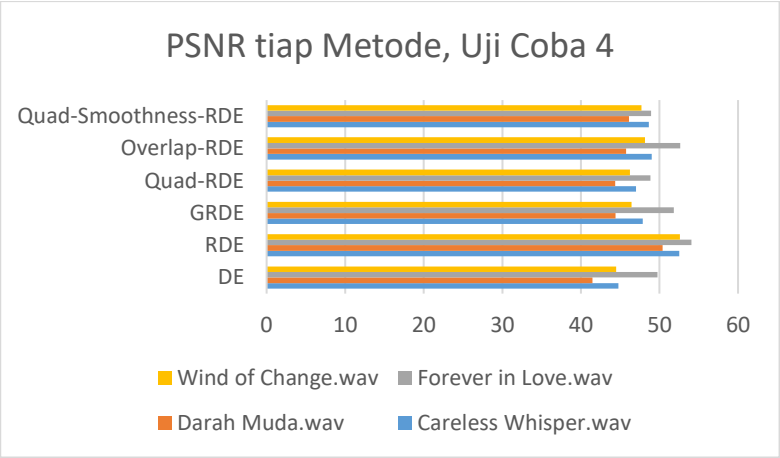
| **Coveraudio** | **Metode** | PSNR |
|---|---|---|
| Careless Whisper.wav | DE | 44.782 |
| | RDE | 52.528 |
| | GRDE | 47.859 |
| | Quad-RDE | 47.021 |
| | Overlap-RDE | 49.011 |
| | Quad-Smoothness-RDE | 48.645 |
| Rata-rata | | 48.307 |
| Darah Muda.wav | DE | 41.460 |
| | RDE | 50.419 |
| | GRDE | 44.402 |
| | Quad-RDE | 44.374 |
| | Overlap-RDE | 45.769 |
| | Quad-Smoothness-RDE | 46.133 |
| Rata-rata | | 45.426 |
| Forever in Love.wav | DE | 49.751 |
| | RDE | 54.061 |
| | GRDE | 51.818 |
| | Quad-RDE | 48.853 |
| | Overlap-RDE | 52.643 |
| | Quad-Smoothness-RDE | 48.929 |
| Rata-rata | | 51.009 |
| Wind of Change.wav | DE | 44.475 |
| | RDE | 52.586 |
| | GRDE | 46.438 |
| | Quad-RDE | 46.237 |
| | Overlap-RDE | 48.146 |

| | | |
|---|---|---|
| | Quad-Smoothness-RDE | 47.699 |
| Rata-rata | | 47.596 |
| Rata-rata total | | 48.085 |

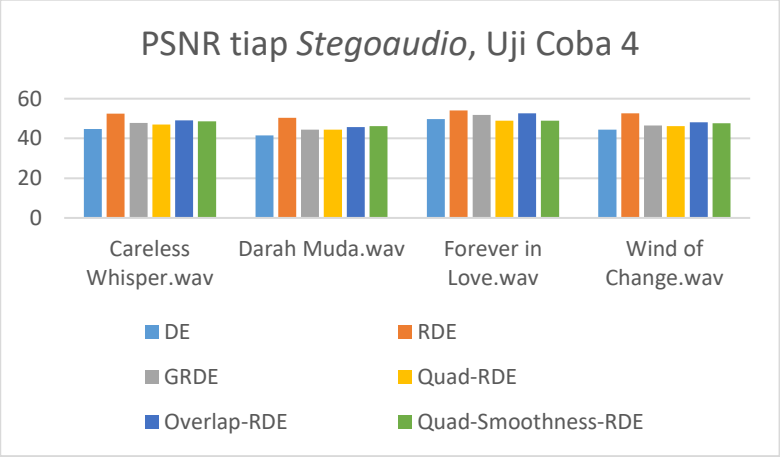
Dari Tabel **5.8** didapatkan rata-rata PSNR sebesar 51.8312636. Pada Gambar 5.13 dan Gambar 5.14 dapat dilihat perbandingan PSNR antar metode dan *stegoaudio*. *Stegoaudio* Forever In Love.wav memiliki rata-rata PSNR terbaik.

Gambar 5.15 merupakan grafik dari Forever In Love.wav yang disisipi pesan tersembunyi menggunakan metode RDE. Perbedaan antara *coveraudio* dan *stegoaudio* tidak terlihat jelas dalam grafik. Perbedaan hanya terlihat pada sampel yang memililki beda yang besar saja.

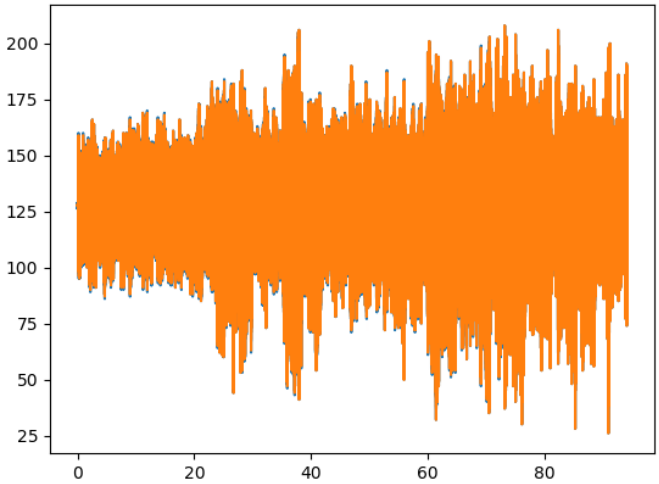
Gambar 5.16 merupakan grafik dari Darah Muda.wav yang disisipi pesan tersembunyi menggunakan metode DE. Perbedaan antara *coveraudio* dan *stegoaudio* dapat dilihat jelas pada grafik. Grafik biru merupakan beda antara *coveraudio* dan *stegoaudio*. Pesan tersembunyi mengisi sekitar tiga per empat *stegoaudio*.



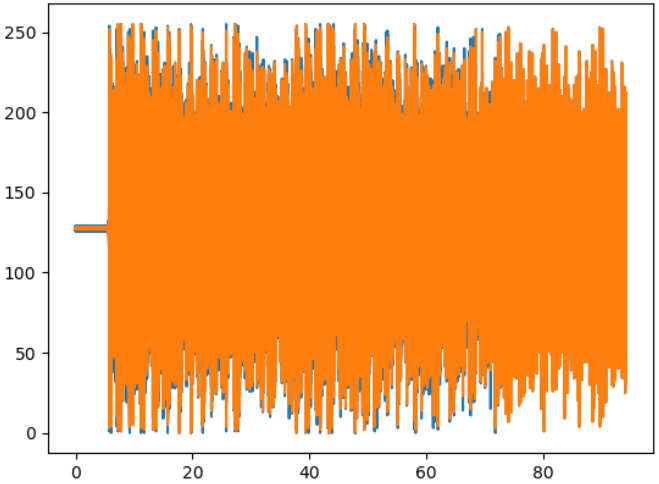

**Gambar 5.13 Grafik PSNR tiap Metode, Uji Coba 4**




**Gambar 5.14 Grafik PSNR tiap Stegoaudio, Uji Coba 4**




**Gambar 5.15 Grafik Sampel Audio dengan PSNR Tertinggi pada Uji Coba 4**




**Gambar 5.16 Grafik Sampel Audio dengan PSNR Terendah pada Uji Coba 4**

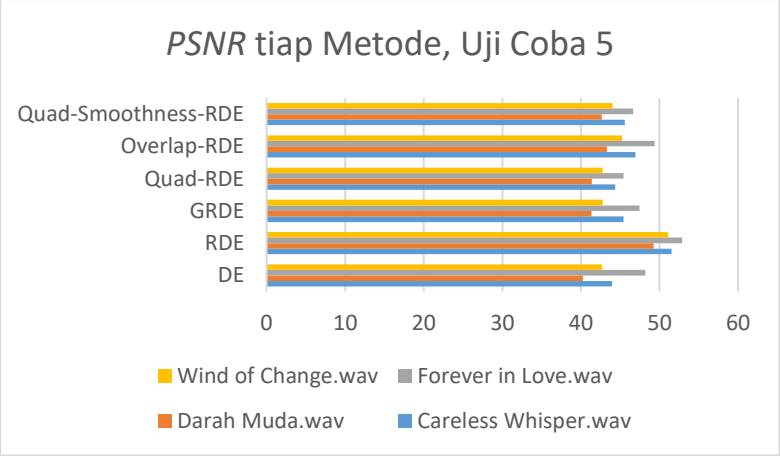
### 5.3.5 Skenario Uji Coba 5

Skenario ini dilakukan untuk mengetahui *PSNR stegoaudio* bila disisipi pesan tersembunyi secara penuh. Hasil dari skenario dapat dilihat pada Tabel 5.9.

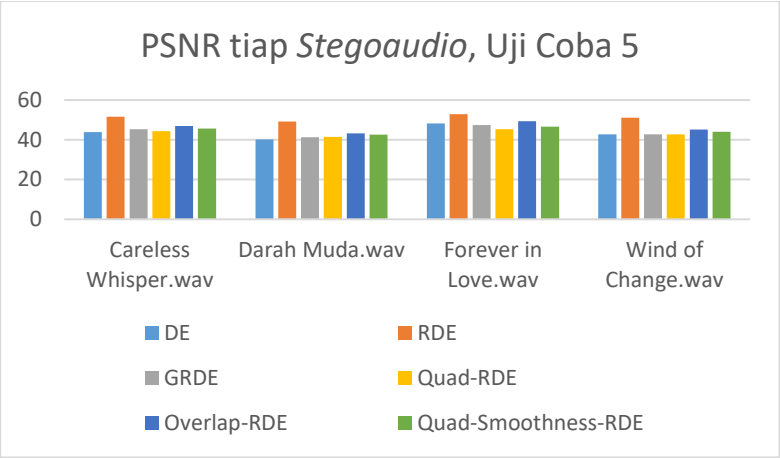
**Tabel 5.9 Hasil Uji Coba Skenario 5**

| Coveraudio | Metode | Kapasitas | PSNR |
|---|---|---|---|
| Careless Whisper.wav | DE | 2074176 | 43.959 |
| | RDE | 2074176 | 51.541 |
| | GRDE | 3111264 | 45.408 |
| | Quad-RDE | 3111261 | 44.349 |
| | Overlap-RDE | 2765568 | 46.921 |
| | Quad-Smoothness-RDE | 2333445 | 45.607 |
| Darah Muda.wav | DE | 2072699 | 40.254 |
| | RDE | 2072700 | 49.267 |
| | GRDE | 3109047 | 41.365 |
| | Quad-RDE | 3109047 | 41.389 |
| | Overlap-RDE | 2763599 | 43.298 |
| | Quad-Smoothness-RDE | 2331783 | 42.640 |
| Forever in Love.wav | DE | 2072700 | 48.204 |
| | RDE | 2072700 | 52.886 |
| | GRDE | 3109050 | 47.475 |
| | Quad-RDE | 3109050 | 45.407 |
| | Overlap-RDE | 2763600 | 49.379 |
| | Quad-Smoothness-RDE | 2331783 | 46.662 |
| Wind of Change.wav | DE | 2072700 | 42.678 |
| | RDE | 2072700 | 51.101 |
| | GRDE | 3109050 | 42.770 |
| | Quad-RDE | 3109050 | 42.785 |
| | Overlap-RDE | 2763600 | 45.232 |
| | Quad-Smoothness-RDE | 2331783 | 44.037 |

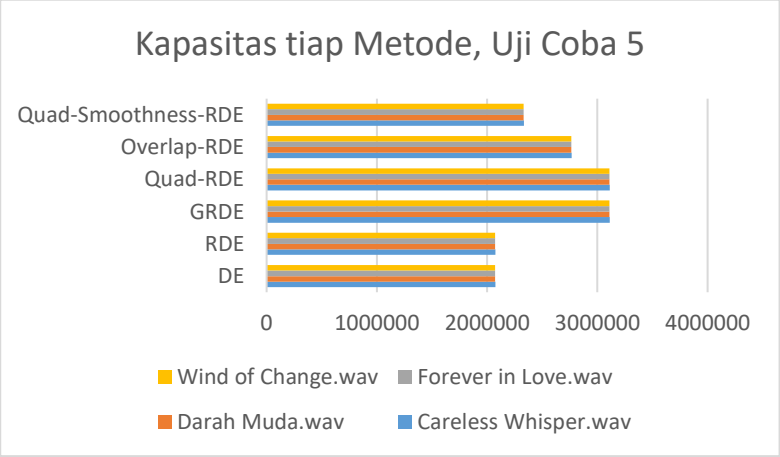
Dari Tabel 5.9 dapat dilihat kapasitas penyisipan pesan pada tiap *coveraudio* dengan metode-metode yang diimplementasikan dan nilai *PSNR* dari *stegoaudio* dari tiap *coveraudio*. Pada Gambar 5.17 dan Gambar 5.18 dapat dilihat perbandingan PSNR antar metode dan *stegoaudio*. Perbandingan kapasitas penyisipan antar metode dan *stegoaudio* dapat dilihat pada Gambar 5.19 dan Gambar 5.20.



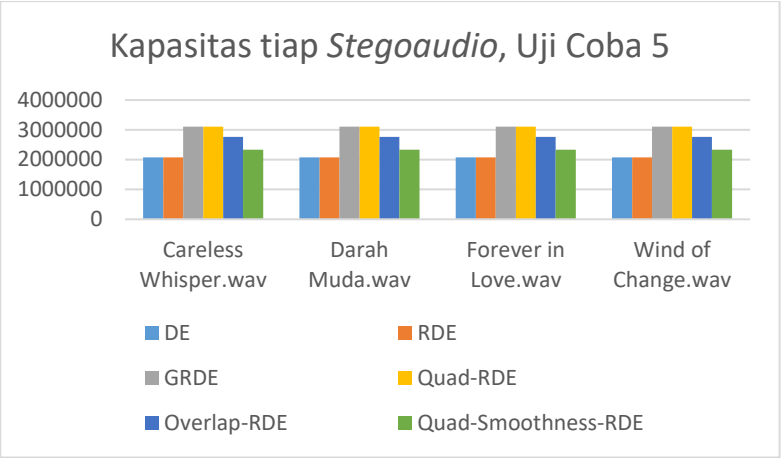

**Gambar 5.17 Grafik PSNR tiap Metode, Uji Coba 5**




**Gambar 5.18 Grafik PSNR tiap Stegoaudio, Uji Coba 5**




**Gambar 5.19 Grafik Kapasitas tiap Metode, Uji Coba 5**




**Gambar 5.20 Grafik Kapasitas tiap *Stegoaudio*, Uji Coba 5**

## 5.4 Evaluasi Skenario Uji Coba

Dari skenario-skenario yang telah dilakukan didapatkan bahwa metode-metode yang diajukan dapat diimplementasikan dengan berhasil. Semua metode dapat diimplementasikan ke semua *coveraudio.*

Pemilihan *coveraudio* berpengaruh pada nilai *PSNR stegoaudio.* C*overaudio* yang memiliki grafik sampel yang halus dan perbedaan antar sampelnya sedikit dapat memiliki nilai *PSNR* yang besar.

Panjang *coveraudio* tidak berpengaruh pada kapasitas penyisipan pesan tersembunyi. Hal tersebut dapat dilihat pada Tabel 5.9. Kapasitas penyimpanan dipengaruhi oleh metode penyisipan pesan tersembunyi.

GRDE dan Quad-RDE memiliki kapasitas penyimpanan lebih besar dari metode lainnya. Hal ini disebabkan oleh pengolahan sampel menjadi *vectorArray* berbentuk (n, 4) sehingga *vector* dapat disisipi lebih banyak pesan tersembunyi.

Perbandingan kapasitas penyisipan pesan tersembunyi dalam *vector* dapat dilihat pada Tabel 5.10

**Tabel 5.10 Perbandingan Kapasitas *vector***

| **Metode** | **Bentuk Array** | **Kapasitas *bit*** |
|---|---|---|
| *DE* | (n, 2) | 1 *bit* |
| *RDE* | (n, 2) | 1 *bit* |
| *GRDE* | (n, 4) | 3 *bit* |
| *Quad-RDE* | (n, 4) | 3 *bit* |
| *Overlap-RDE* | (n, 1) | 1 *bit* tiap iterasi |
| *Quad-Smoothness-RDE* | (n, 4, 4) | 3 *bit* tiap *quad* |

Metode RDE adalah metode yang memiliki hasil *PSNR* yang lebih besar dari metode lain. Nilai *PSNR* metode ini memiliki rata-rata paling besar dibanding metode lain. Metode ini dapat menghasilkan nilai *PSNR* paling besar pada penyisipan pesan tersembunyi secara penuh.

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

# 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. Metode *difference expansion, reduced difference expansion, generalized reduced difference expansion, quad-based reduced difference expansion, block-overlap reduced difference expansion,* dan *quad smoothness reduced difference expansion* dapat digunakan sebagai metode untuk penyembunyian pesan tersembunyi pada berkas audio.
2. Metode *general reduced difference expansion* diimplementasikan pada berkas audio dengan cara membuat *vectorArray* dengan bentuk (n, 4) sebelum penyisipan pesan dilakukan.
3. Metode *quad-based general rde* diimplementasikan pada berkas audio dengan cara membuat *vectorArray* dengan bentuk (n, 4) ) sebelum penyisipan pesan dilakukan.
4. Metode *block overlap general rde* diimplementasikan pada berkas audio dengan cara membuat *vectorArray* dengan bentuk (n, 1) dan melakukan iterasi penyisipan pesan tersembunyi dari awal hingga akhir *vectorArray*.
5. Metode *quad smoothness general rde* diimplementasikan pada berkas audio dengan cara membuat *vectorArray* dengan

bentuk (n, 4, 4) serta melakukan penyisipan berdasarkan urutan *variance quad of quad*.

6. Nilai *PSNR* tertinggi dihasilkan oleh metode *reduced difference expansion* dengan rata-rata nilai *PSNR* 51,19874 pada penyisipan pesan tersembunyi secara penuh.
7. Kapasistas penyisipan pesan terbesar dihasilkan oleh metode *generalized reduced difference expansion* dengan rata-rata 310.963 *bit* yang berhasil disisipkan

## 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 Tugas Akhir ini adalah:

1. Pengembangan skema penyisipan pesan tersembunyi sehingga tidak diperlukan berkas *location map* pada proses *extraction*.
2. Pengembangan skema penyisipan pesan tersembunyi sehingga perangkat lunak dapat menerima berkas pesan tersembunyi masukan selain teks.
3. Pengembangan teknik pengurangan nilai beda pada proses *embedding* sehingga kualitas *stegoaudio* dapat ditingkatkan.
4. Pengembangan teknik pemilihan sampel sehingga kualitas *stegoaudio* dapat ditingkatkan.
5. Optimasi perhitungan dan pendistribusian komputasi pada proses *embedding* dan *extraction* untuk mempercepat proses penyisipan dan pengembalian.

# DAFTAR PUSTAKA


[1] F. A. P. Petitcolas, R. J. Anderson, dan M. G. Kuhn, "Information hiding-a survey," *Proc. IEEE*, vol. 87, no. 7, hal. 1062–1078, Jul 1999.

[2] M. M. Amin, M. Salleh, S. Ibrahim, M. R. Katmin, dan M. Z. I. Shamsuddin, "Information hiding using steganography," in *4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings.*, 2003, hal. 21–25.

[3] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, hal. 890–896, Agu 2003.

[4] T. Ahmad dan M. Holil, "Increasing the Performance of Difference Expansion-based Steganography when Securing Medical Data," *Smart Comput. Rev.*, vol. 4, no. 4, hal. 397–322, Agu 2014.

[5] A. Izvorski, *video-quality: Video quality metrics, reference implementation in python: VIF, SSIM, PSNR, ..* 2017.

[6] "python - How to plot a wav file - Stack Overflow." [Daring]. Tersedia pada: http://stackoverflow.com/questions/18625085/how-to-plot-a-wav-file. [Diakses: 18-Mei-2017].

[7] "python - Convert string to list of bits and viceversa - Stack Overflow." [Daring]. Tersedia pada: http://stackoverflow.com/questions/10237926/convert-string-to-list-of-bits-and-viceversa. [Diakses: 18-Mei-2017].

[8] J. Watkinson, *Art of Digital Audio*. Taylor & Francis, 2013.

[9] S. Katzenbeisser dan F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[10] C. L. Liu, D. C. Lou, dan C. C. Lee, "Reversible Data Embedding Using Reduced Difference Expansion," in *Third International Conference on Intelligent Information Hiding*

*and Multimedia Signal Processing (IIH-MSP 2007)*, 2007, vol. 1, hal. 433–436.

[11] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, hal. 1147–1156, Agu 2004.

[12] "Welcome to Python.org," *Python.org*. [Daring]. Tersedia pada: https://www.python.org/about/. [Diakses: 18-Mei-2017].

[13] "Anaconda," *Continuum*. [Daring]. Tersedia pada: https://www.continuum.io/Anaconda-Overview. [Diakses: 18-Mei-2017].

[14] "Quickstart tutorial — NumPy v1.13.dev0 Manual." [Daring]. Tersedia pada: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html. [Diakses: 18-Mei-2017].

[15] "PyQt - Python Wiki." [Daring]. Tersedia pada: https://wiki.python.org/moin/PyQt. [Diakses: 18-Mei-2017].

[16] "Peak Signal-to-Noise Ratio as an Image Quality Metric - National Instruments." [Daring]. Tersedia pada: http://www.ni.com/white-paper/13306/en/. [Diakses: 18-Mei-2017].

[17] "Your first GUI app with Python and PyQt," *Python For Engineers*, 17-Mar-2015. .

[18] "How to Install PyQt5 and Build Your First GUI in Python 3.4," *SkyLogic Projects*, 01-Jun-2015. .

## BIODATA PENULIS

Danang Adi Nugroho merupakan anak kedua dari pasangan Bapak Nasikhun dan Ibu Endang Nurmiyati. Lahir di Pasuruan pada 13 Juni 1995. Penulis menempuh pendidikan formal pada SMA Negeri 1 Pasuruan (2010-2013) dan menjadi mahasiswa S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (2013-2017). Bidang studi yang dipilih penulis pada saat berkuliah adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (2014-2015). Penulis juga aktif dalam kegiatan SCHEMATICS 2014 sebagai panitia REEVA 2014. Penulis memenuhi kuliah kerja praktiknya di PT. Telkom Indonesia pada divisi SSO pada pertengahan 2016. Penulis dapat dihubungi melalui surel: danang.adi.n@outlook.com.