

**TUGAS AKHIR – KI141502**

**IMPLEMENTASI ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA) UNTUK MENGATASI BLACKHOLE DAN WORMHOLE ATTACK PADA KOMUNIKASI V2V DI LINGKUNGAN VANETS**

**TITIES NOVANINDA OVARI**

**NRP 5113100083**

Dosen Pembimbing I

Dr. Radityo Anggoro, S.Kom., M.Sc.

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]***



**TUGAS AKHIR – KI141502**

**IMPLEMENTASI ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA) UNTUK MENGATASI BLACKHOLE DAN WORMHOLE ATTACK PADA KOMUNIKASI V2V DI LINGKUNGAN VANETS**

**TITIES NOVANINDA OVARI**

**NRP 511310003**

**Dosen Pembimbing I**

**Dr. Radityo Anggoro, S.Kom., M.Sc.**

**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 THESES – KI141502**

**ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA) IMPLEMENTATION TO OVERCOME BLACKHOLE AND WORMHOLE ATTACK IN V2V COMMUNICATION OF VANETS**

**TITIES NOVANINDA OVARI**

**NRP 5113100083**

**First Advisor**

**Dr. Radityo Anggoro, S.Kom., M.Sc.**

**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

**IMPLEMENTASI ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM UNTUK MENGATASI BLACKHOLE DAN WORMHOLE ATTACK PADA KOMUNIKASI V2V DI LINGKUNGAN VANETS**

**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:

**TITIES NOVANINDA OVARI**

**NRP: 5113100083**

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Radityo Anggoro, S.Kom., M.Sc. .....................

(NIP. 198410162008121002) (Pembimbing 1)

1. Henning Titi Ciptaningtyas, S.Kom., M.Kom. .....................

(NIP. 198407082010122004) (Pembimbing 2)

**SURABAYA**

**MARET, 2017 *[Halaman ini sengaja dikosongkan]***

**IMPLEMENTASI ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA) UNTUK MENGATASI BLACKHOLE DAN WORMHOLE ATTACK PADA KOMUNIKASI V2V DI LINGKUNGAN VANETS**

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa** | **:** | **TITIES NOVANINDA OVARI** |
| **NRP** | **:** | **5113100083** |
| **Jurusan** | **:** | **Teknik Informatika FTIF-ITS** |
| **Dosen Pembimbing 1** | **:** | **Dr. Radityo Anggoro, S.Kom., M.Sc.** |
| **Dosen Pembimbing 2** | **:** | **Henning Titi Ciptaningtyas, S.Kom., M.Kom.** |

# Abstrak

*Vehicular Ad-hoc Network (VANET) merupakan teknologi yang mendukung pertukaran data antar kendaraan di jalan raya. VANET menggunakan jaringan wireless berbasis ad-hoc yang memungkinkan terjadi serangan, seperti Blackhole dan Wormhole attack. Blackhole attack adalah serangan yang dilakukan suatu malicious node dengan menjatuhkan paket data yang diterima, sedangkan Wormhole attack dilakukan dengan mengubah isi paket data. Kedua serangan ini dapat mempengaruhi performa routing protocol yang digunakan dalam komunikasi antar kendaraan.*

*Kedua serangan tersebut dapat diatasi jika tiap node dapat mendeteksi jika ada malicious node di sekitarnya, sehingga tiap node tersebut tidak meneruskan paket data ke malicious node. Blackhole attack dapat dideteksi dengan melakukan pemantauan terhadap arus pertukaran data pada tiap node, sedangkan Wormhole attack dapat dideteksi dengan menggunakan digital signature. Tugas akhir ini mengimplementasikan Elliptic Curve Digital Signature Algorithm (ECDSA) sebagai algoritma digital signature yang dilampirkan pada tiap paket data.*

*Pada tugas akhir ini didapatkan routing protocol yang menerapkan pemantauan arus pertukaran data dan digital signature pada tiap paket data yang dikirimkan mempunyai performa yang lebih baik jika dihadapkan dengan Blackhole dan Wormhole attack dibandingkan dengan routing protocol yang tidak menerapkannya.*

***Kata kunci: ECDSA, Blackhole attack, Wormhole attack, VANET.***

**ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM IMPLEMENTATION TO OVERCOME BLACKHOLE AND WORMHOLE ATTACK IN V2V COMMUNICATION OF VANETS**

|  |  |  |
| --- | --- | --- |
| **Student’s Name** | **:** | **TITIES NOVANINDA OVARI** |
| **Student’s ID** | **:** | **5113100083** |
| **Department** | **:** | **Teknik Informatika FTIF-ITS** |
| **First Advisor** | **:** | **Dr. Radityo Anggoro, S.Kom., M.Sc.** |
| **Second Advisor** | **:** | **Henning Titi Ciptaningtyas, S.Kom., M.Kom.** |

# *Abstract*

*Vehicular Ad-hoc Network (VANET) is a technology enabling data exchange between vehicular on roads, in order to support Intelligent Transportation System. VANET use an ad-hoc base wireless network that can be attacked by several attacks, such as Blackhole attack and Wormhole attack. Blackhole attack is an attack conducted by dropping data packet that has been received, meanwhile Wormhole attack conducted by tampering the data packet. Both attacks can affect the performance of the routeing protocol used in vehicular communication.*

*Both attacks can be overcome if each node can detect the malicious node around them, so they do not forward the data packet to malicious node. Blackhole attack can be detected by recording the data packet exchange traffic in each node, meanwhile Wormhole attack can be detected by attaching digital signature on each data packets that will be sent. This thesis implements Elliptic Curve Digital Signature Algorithm (ECDSA) as the digital signature algorithm attached to each data packets that will be sent.*

*In this thesis, it is obtained that the routeing protocol which applies data packet exchange traffic recording and the digital signature on each packet that will be sent has better performance than the routeing protocol which does not.*

***Keywords: ECDSA, Blackhole attack, Wormhole attack, VANET.***

**KATA PENGANTAR**



Alhamdulillahirabbil’alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul “**IMPLEMENTASI ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA) UNTUK MENGATASI BLACKHOLE DAN WORMHOLE ATTACK PADA KOMUNIKASI V2V DI LINGKUNGAN VANETS**”. 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 perkuliahaan 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. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Radityo Anggoro, S.Kom., M.Sc. dan Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom.selaku dosen pembimbing penulis yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan tugas akhir ini.
4. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
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. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
8. Teman-teman administrator NCC/KBJ, yang telah menemani dan menyemangati penulis selama penulis menjadi administrator, menjadi rumah kedua penulis selama penulis berkuliah.
9. Teman-teman angkatan 2013, yang sudah mendukung penulis selama perkuliahan.
10. Sahabat penulis 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, Maret 2017

# DAFTAR ISI

LEMBAR PENGESAHAN v

Abstrak vii

*Abstract* ix

DAFTAR ISI xiii

DAFTAR GAMBAR xvii

DAFTAR TABEL xix

DAFTAR KODE SUMBER xxi

BAB 1 BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Batasan Permasalahan 2

1.4 Tujuan 2

1.5 Manfaat 3

1.6 Metodologi 3

1.6.1 Penyusunan Proposal Tugas Akhir 3

1.6.2 Studi Literatur 3

1.6.3 Implementasi Perangkat Lunak 4

1.6.4 Pengujian dan Evaluasi 4

1.6.5 Penyusunan Buku 4

1.7 Sistematika Penulisan Laporan 5

BAB 2 BAB II TINJAUAN PUSTAKA 7

2.1 Citra Digital 7

2.2 Forensik Digital 7

2.3 Forensik Citra Digital 8

2.4 *Copy Move* 8

2.5 Python 9

2.6 Anaconda 10

2.7 OpenCV 11

2.8 Matplotlib 11

2.9 NumPy 12

2.10 SciPy 12

2.11 PyQt 13

2.12 JSON 13

2.13 Variance 14

2.14 Mean Squared Error 14

2.15 Convolution 14

2.16 Canny Edge Detection 15

BAB 3 BAB III PERANCANGAN PERANGKAT LUNAK 17

3.1 Data 17

3.1.1 Data Masukan 17

3.1.2 Data Keluaran 19

3.2 Dekripsi Umum Sistem 20

3.3 Modifikasi *Expanding Block Algorithm* 23

3.4 Metode Filter 23

3.4.1 *Averaging Filter* 23

3.4.2 *Median Filter* 23

3.5 Perancangan Antarmuka 24

BAB 4 BAB IV IMPLEMENTASI 27

4.1 Lingkungan Implementasi 27

4.2 Implementasi 27

4.2.1 Kelas Block Container 28

4.2.1.1 Detail Atribut Kelas Blok Container 28

4.2.2 Inisialisasi File 29

4.2.3 Inisialisasi Threshold 30

4.2.4 Pembuatan *Overlapping Block* 31

4.2.5 Pembuatan Grup 32

4.2.6 Pembuatan Bucket 33

4.2.7 Calculate Statistic 33

4.2.8 Find Overlap Block 34

4.2.9 Find Connection 35

4.2.10 Bucket Process 35

4.2.11 Build Copy-Move Image 37

4.2.12 Create Mask 38

4.2.13 Write Mask 39

4.2.14 Build Edge Copy-Move Image 39

4.2.15 Mean Squared Error 40

4.2.16 Similarity 41

4.2.17 Pembuatan Antarmuka 42

BAB 5 BAB V HASIL UJI COBA DAN EVALUASI 45

5.1 Lingkungan Pengujian 45

5.2 Data Pengujian 45

5.3 Preprocessing citra 48

5.4 Skenario Uji Coba 48

5.4.1 Skenario Uji Coba 1 51

5.4.2 Skenario Uji Coba 2 52

5.4.3 Skenario Uji Coba 3 53

5.4.4 Skenario Uji Coba 4 54

5.4.5 Skenario Uji Coba 5 55

5.4.6 Skenario Uji Coba 6 56

5.4.7 Skenario Uji Coba 7 57

5.4.8 Skenario Uji Coba 8 58

5.4.9 Skenario Uji Coba 9 59

5.4.10 Skenario Uji Coba 10 60

5.4.11 Skenario Uji Coba 11 61

5.4.12 Skenario Uji Coba 12 62

5.4.13 Skenario Uji Coba 13 63

5.4.14 Skenario Uji Coba 14 64

5.4.15 Skenario Uji Coba 15 65

5.5 Evaluasi Umum Skenario Uji Coba 66

BAB 6 BAB VI KESIMPULAN DAN SARAN 68

6.1 Kesimpulan 68

6.2 Saran 68

BAB 7 DAFTAR PUSTAKA 69

BAB 8 LAMPIRAN 70

8.1 Kumpulan Kode Sumber 70

BAB 9 BIODATA PENULIS 79

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

# DAFTAR GAMBAR

Gambar 2.1 Contoh Copy-Move Citra 9

Gambar 2.2 Logo Python 10

Gambar 2.3 Logo Anaconda 10

Gambar 2.4 Logo OpenCV 11

Gambar 2.5 Logo Matplotlib 11

Gambar 2.6 Logo NumPy 12

Gambar 2.7 Logo SciPy 12

Gambar 2.8 Logo Qt 13

Gambar 2.9 Logo JSON 13

Gambar 3.1 Contoh citra *copy-move* 17

Gambar 3.2 Contoh kunci jawaban citra *copy-move* 18

Gambar 3.3 Contoh keluaran citra terdeteksi *copy-move* 19

Gambar 3.4 Overlapping Block 2 x 2 20

Gambar 3.5 Proses Pengurutan Blok dan Penempatan ke Buckets 21

Gambar 3.6 Desain Tampilan Antarmuka 24

Gambar 4.1 Antarmuka Awal Dijalankan 44

Gambar 4.2 Antarmuka Masukan Data 44

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

# DAFTAR TABEL

[Tabel 4.1 Lingkungan Implementasi Perangkat Lunak 27](#_Toc470245388)

[Tabel 4.2 Atribut Kelas Blok Container 28](#_Toc470245389)

[Tabel 5.1 Spesifikasi Lingkungan Pengujian 45](#_Toc470245390)

[Tabel 5.2 Threshold a 48](#_Toc470245391)

[Tabel 5.4 Confusion matrix uji coba 1 51](#_Toc470245392)

[Tabel 5.5 Confusion matrix uji coba 2 52](#_Toc470245393)

[Tabel 5.6 Confusion matrix uji coba 3 53](#_Toc470245394)

[Tabel 5.7 Confusion matrix uji coba 4 54](#_Toc470245395)

[Tabel 5.8 Confusion matrix uji coba 5 55](#_Toc470245396)

[Tabel 5.9 Confusion matrix uji coba 6 56](#_Toc470245397)

[Tabel 5.10 Confusion matrix uji coba 7 57](#_Toc470245398)

[Tabel 5.11 Confusion matrix uji coba 8 58](#_Toc470245399)

[Tabel 5.12 Confusion matrix uji coba 9 59](#_Toc470245400)

[Tabel 5.13 Confusion matrix uji coba 10 60](#_Toc470245401)

[Tabel 5.14 Confusion matrix uji coba 11 61](#_Toc470245402)

[Tabel 5.15 Confusion matrix uji coba 12 62](#_Toc470245403)

[Tabel 5.16 Confusion matrix uji coba 13 63](#_Toc470245404)

[Tabel 5.17 Confusion matrix uji coba 14 64](#_Toc470245405)

[Tabel 5.18 Confusion matrix uji coba 15 65](#_Toc470245406)

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

# DAFTAR KODE SUMBER

[Pseudocode 3.1 Contoh Threshold Config 19](#_Toc470245407)

[Pseudocode 4.1 Kode Program Kelas Blok Container 28](#_Toc470245408)

[Pseudocode 4.2 Kode Program Inisialisasi File 30](#_Toc470245409)

[Pseudocode 4.3 Kode Program Inisialisasi Threshold 31](#_Toc470245410)

[Pseudocode 4.4 Kode Program Overlapping block 31](#_Toc470245411)

[Pseudocode 4.5 Kode Program Pengurutan Blok 31](#_Toc470245412)

[Pseudocode 4.6 Kode Program Pembuatan Grup 32](#_Toc470245413)

[Pseudocode 4.7 Kode Program Pembuatan Bucket 33](#_Toc470245414)

[Pseudocode 4.8 Kode Program Perhitungan Statistik 34](#_Toc470245415)

[Pseudocode 4.9 Kode Program Pengecekan Overlap 35](#_Toc470245416)

[Pseudocode 4.10 Kode Program Pengekan Connection 35](#_Toc470245417)

[Pseudocode 4.11 Kode Program Memproses Setiap Bucket 37](#_Toc470245418)

[Pseudocode 4.12 Kode Program Membangun Gambar Copy-Move 38](#_Toc470245419)

[Pseudocode 4.13 Kode Program Membuat Gambar Mask 39](#_Toc470245420)

[Pseudocode 4.14 Kode Program Menempelkan Mask Ke Gambar 39](#_Toc470245421)

[Pseudocode 4.15 Kode Program Membuat Edge Pada Gambar 40](#_Toc470245422)

[Pseudocode 4.16 Kode Program Menghitung Mean Squared Error 41](#_Toc470245423)

[Pseudocode 4.17 Kode Program Menghitung Kemiripan 41](#_Toc470245424)

[Pseudocode 4.18 Kode Program Menjalankan Deteksi 43](#_Toc470245425)

# BAB I PENDAHULUAN

## Latar Belakang

Vehicular Ad-hoc Network (VANET) merupakan salah satu teknologi yang saat ini banyak dikembangkan di berbagai negara. Terdapat beberapa area riset dalam teknologi VANET, diantaranya adalah *routing protocol* dan sekuritas. Fokus pada *routing protocol* adalah untuk memenuhi kebutuhan mengenai penentuan rute dari *source node* ke *destination node* dengan jangka waktu sesingkat mungkin dan akurasi rute yang optimal, sedangkan sekuritas berfokus pada ketercapaian dan autentikasi paket dalam pertukaran paket. Skenario serangan yang memungkinkan terjadi pada jaringan VANET adalah *Blackhole* dan *Wormhole attack.*

*Blackhole attack* merupakan serangan dimana *malicious node* selalu melakukan *drop* pada paket yang diterimanya dan membalas pada *source node* seolah paket sudah diterima oleh *destination node*. Sedangkan *Wormhole attack* memiliki banyak variasi serangan, salah satunya dengan mengubah isi dari paket yang diterimanya.

Dalam beberapa tahun terakhir terdapat penelitian terkait metode untuk mengatasi *Blackhole* maupun *Wormhole Attack* pada jaringan VANET, salah satunya adalah penerapan *evaluation system* dan *digital signature* pada *routing protocol* GPSR.

Pada penelitian kali ini akan dilakukan cara untuk mengatasi *Blackhole* dan *Wormhole attack* dengan menggunakan *evaluation value* yang didapatkan melalui *forward-backward* evaluation dan implementasi *Elliptic Curve Digital Signature Algorithm* (ECDSA) sebagai algoritma digital signature pada *routing protocol* AODV.

Dikemudian hari dapat dihasilkan routing protocol yang cukup efektif dalam mengatasi serangan dalam jaringan VANET, sehingga didapatkan *routing protocol* yang aman dan *reliable* untuk komunikasi antar kendaraan.

## Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara mendeteksi *malicious node* yang melakukan *Blackhole* dan *Wormhole attack*?
2. Bagaimana cara mengimplementasikan ECDSA untuk mengatasi *Blackhole* dan *Wormhole attack*?
3. Bagaimana perbandingan performa *routing protocol* AODV ketika terdapat *malicious node* yang melakukan *Blackhole* dan *Wormhole attack* sebelum dan sesudah ECDSAditerapkan?

## Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Simulator yang digunakan adalah simulator NS2.
2. *Routing protocol* yang digunakan adalah *routing protocol* AODV.
3. Algoritma *digital signature* yang digunakan adalah algoritma ECDSA.

## Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Dapat mendeteksi *malicious node* yang melakukan *blackhole* dan *wormhole attack* pada lingkungan VANET dengan *routing protocol* AODV.
2. Dapat mengimplementasikan ECDSA untuk mengatasi *blackhole* dan *wormhole attack* yang terjadi di lingkungan VANET dengan *routing protocol* AODV.
3. Menganalisa perbandingan performa *routing protocol* AODV ketika terdapat *malicious node* yang melakukan *blackhole* dan *wormhole attack* sebelum dan sesudah ECDSA diterapkan.

## Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah sebagai berikut:

1. Memberikan manfaat di bidang VANET terutama pada area riset sekuritas dengan mendapatkan performa *routing protocol* yang relatif stabil meskipun terdapat *malicious node* yang melakukan *Blackhole* dan *Wormhole attack* pada lingkungan VANET.
2. Menerapkan ilmu yang dipelajari selama kuliah di Teknik Informatika ITS agar dapat berguna bagi orang banyak.

## Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagi berikut:

### 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. Prosposal juga berisi tentang garis besa tugas akhir yang akan dikerjakan sehingga memberikan gambaran untuk dapat mengerjakan tugas akhir sesuai dengan *timeline* yang dibuat. Gagasan untuk menganalisa dan mengidentifikasi pemasuan citra terhadap serangan *copy-move* menggunakan metode *expanding block algorithm*[1].

### Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi 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 *expanding block algorithm*, dan metode-metodestatistika seperti mean dan variance seta metode-metode untuk melakukan pengolahan gambar atau *Image Processing* seperti *Blur, Sharpen, Edge Detection*[6]*.* Tugas akhir ini juga mengacu pada literatur jurnal dengan judul “*An efficient expanding block algorithm for image copy-move forgery detection”* yang diterbitkan pada tahun 2013[1].

### 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 2.7 sebagai bahasa pemrograman utama dengan distribusi package Anaconda dan PyCharm Professional 2016.2.3 sebagai IDE.

### Pengujian dan Evaluasi

Pada tahap ini algoritma yang telah disusun diuji coba dengan menggunakan data uji coba yang ada. Data uji coba tersebut diuji coba dengan menggunakan suatu perangkat lunak dengan tujuan mengetahui kemampuan metode yang digunakan dan mengevaluasi hasil tugas akhir dengan jurnal pendukung yang ada. Hasil evaluasi juga mencakup kompleksitas dari kemampuan masing-masing metode tersebut dalam mendeteksi serangan *copy-move*.

### Penyusunan Buku

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

## 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.

* + 1. 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 forensik digital, forensik citra digital, *copy-move*, Mean, Variance, Mean Squared Error dan Qt*.*

* + 1. Bab III. Perancangan Perangkat Lunak

Bab ini berisi pembahasan mengenai perancangan dari metode *expanding block*  yang digunakan untuk menedeteksi pemalsuan citra pada serangan *copy-move* dan perancangan GUI menggunakan Qt.

* + 1. Bab IV. Implementasi

Bab ini menjelaskan implementasi yang berbentuk *Pseudocode* yang berupa *Pseudocode* dari metode *expanding block algorithm.*

* + 1. Bab V. Pengujian dan Evaluasi

Bab ini berisikan hasil uji coba dari metode *expanding block algorithm* yang digunakan untuk menedeteksi pemalsuan citra pada serangan *copy-move* yang sudah diimplementasikan pada *Pseudocode*. Uji coba dilakukan dengan menggunakan dataset citra yang memiliki kualitas rendah dengan ukuran 512 x 512 dengan memiliki.Hasil evaluasi mencakup perbanding masing-masing data masukan dan filter atau *image processig*.

* + 1. 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.

* + 1. Daftar Pustaka

Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

* + 1. 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 diantaranya adalah *Elliptic Curve Digital Signature Algorithm*, dan beberapa teori lain yang mendukung pembuatan tugas akhir. Penjelesan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

## Vanet

Vanet

## AODV (Ad hoc On-Demand Distance Vector)

AODV merupakan *routing protocol* yang umum digunakan di MANET dan telah diaplikasikan juga di VANET. Pada AODV, rute dibuat hanya ketika diminta oleh *node* asal (*source node*). Metode penemuan rute AODV berdasarkan pada *routing table* yang menyimpan rute terhadap beberapa *node*. *Source node* akan menyebarkan paket *route request* (RREQ) ke tiap *neighbor*-nya yang akan mengirimkan paket RREQ juga ke tiap *neighbor*-nya, begitu seterusnya hingga mencapai *node* tujuan (*destination node*). Ketika paket RREQ mencapai *destination node* atau *node* yang mengetahui rute ke *destination node*, paket *route replay* (RREP) akan dikirimkan kembali ke *source node* melalui *reverse route*. AODV menggunakan sequence number untuk mendapatkan informasi rute yang paling update dan mencegah *routing loop*.

## ECDSA (Elliptic Curve Digital Signature Algorithm)

*Elliptic Curve Digital Signature Algorithm* (ECDSA) merupakan varian dari *Digital Signature Algorithm* (DSA) yang beroperasi pada grup *elliptic curve*. Untuk mengirim pesan yang tertandatangani dari node asal ke node tujuan, keduanya sepakat pada parameter domain *elliptic curve*. Pengirim memiliki *key pair* yang terdiri dari *private key* (sebuah *integer* kurang dari n yang dipilih secara acak, dimana n merupakan urutan dari kurva, sebuah parameter domain *elliptic curve*) dan *public key* (G adalah *generator point*, sebuah parameter domain *elliptic curve*).

## *Blackhole Attack*

*Blackhole attack* terjadi saat terdapat *malicious node* yang menunggu mendapatkan paket RREQ dari *neighbor*-nya. Namun ketika sudah mendapatkan paket RREQ, *malicious node* tersebut membalas seolah-olah dia memiliki rute terpendek menuju *destination node*. Sehingga *source node* akan meneruskan dengan mengirim paket data ke *destination node* melalui *malicious node* tersebut. Ketika mendapatkan paket data dari *source node*, *malicious node* tersebut tidak meneruskan paket data ke *destination node*, tetapi melakukan *drop* terhadap paket data yang diterima. Sehingga paket data tersebut tidak mencapai *destination source*-nya dan *malicious node* tidak memberi tahu *source node* jika paket datanya tidak mencapai *destination node.*

## *Wormhole Attack*

*Wormhole attack* terdiri dari 2 fase yang dilancarkan oleh satu atau lebih *malicious node*. Pada fase pertama, *malicious node* mencoba mengumpan *node neighbor*-nya agar mengirimkan paket melalui dirinya. Pada fase kedua, *malicious node* tersebut memanfaatkan paket data yang diterimanya sesuai keinginannya, salah satunya dengan mengubah isi paket data.

## NS-2 (Network Simulator-2)

NS-2 merupakan sebuah discrete event simulator yang didesain untuk membantu penelitian pada bidang jaringan komputer. Pengembangan NS dimulai pada tahun 1989 sebagai sebuah varian dari REAL network simulator, pada tahun 1995 pengembangan NS didukung oleh DARPA melalui VINT project di LBL, Xerox PARC, UCB dan USC/ISI. NS kemudian memasuki versi dua pada tanggal 31 Juli 1995. Saat ini pengembangan NS didukung oleh DARPA melalui SAMAN dan NSF melalui CONSER beserta peneliti lainnya termasuk ACIRI.

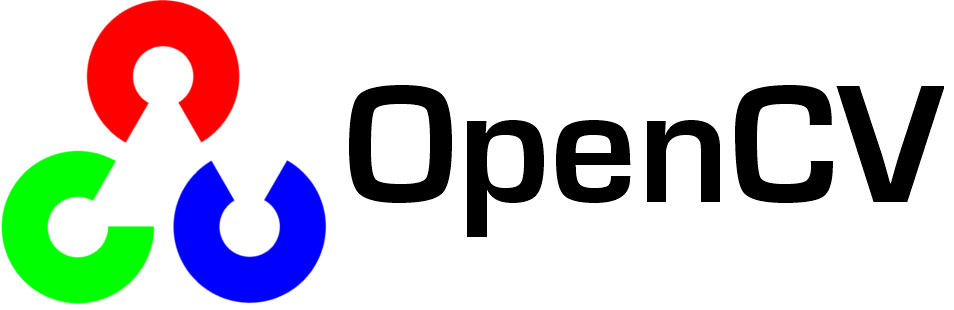
### Instalasi

NS-2 membutuhkan beberapa package yang harus sudah terinstall sebelum memulai instalasi NS-2.

### *Trace File*

## OpenSSL

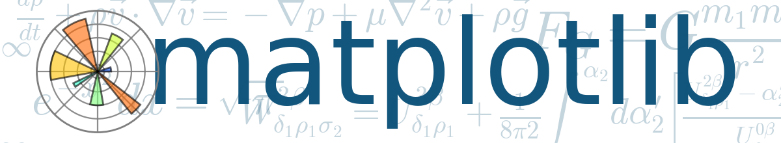
*Open*CV (*Open Source Computer Vision*) adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis secara *real-time* yang dibuat oleh Intel. *Open*CV adalah pustaka perangkat lunak gratis yang dapat digunakan di berbagai *platform*, sepertiGNU/Linux maupun Windows, bahkan OpenCV dapat berjalan pada *Smartphone* Android maupun iOS. *Open*CV mulanya ditulis dalam basaha pemrograman C++. *Open*CV dapat digunakan pada berbagai bahasa seperti Python, Java, atau MATLAB. Gambar 2.4 merupakan logo OpenCV.



Gambar . Logo OpenCV

## Sumo

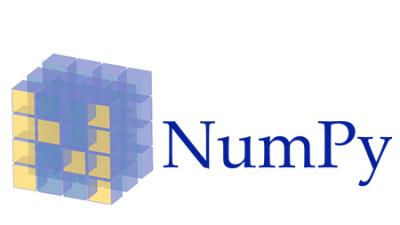
Matplotlib merupakan sebuah library python yang mendukung untuk menghasil plot 2d dari berbagai bentuk format dan jenis data. Matplotlib bersifat *open source* yang banyak digunakan untuk pengolahan ilmiah matematika dan ditampilkan dalam bentuk plot ata grafik. Matplotlib membuat semua jenis plot, histogram, chart untuk mudah dibuat. Gambar 2.7 merupakan logo matplotlib.



Gambar . Logo Matplotlib

## OpenStreetMap

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[7]. Tipe data *arbitrary* dapat didefinisikan, ini memungkinkan NumPy secara lancar dan cepat mengintegrasikan dengan banyak tipe basis data. Gambar 2.5 merupakan logo NumPy.



Gambar . Logo NumPy

## AWK

SciPy merupakan singkatan dari *Scientific Python*. SciPy adalah pustaka perangkat lunak bahasa pemrograman python bersifat *Open Source*. SciPysering digunakan bersama dengan modul NumPy karena SciPy merupakan bagian besar dari NumPy dimana banyak fitur yang memiliki fungsi sama tetapi SciPy dianggap lebih lengkap dari pada NumPy.



Gambar . Logo SciPy

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

# BAB III PERANCANGAN

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak. perangkat lunak yang dibuat pada tugas akhir ini adalah mendeteksi serangan *copy-move* pada citra dengan metode *expanding block algorithm.*

## Data

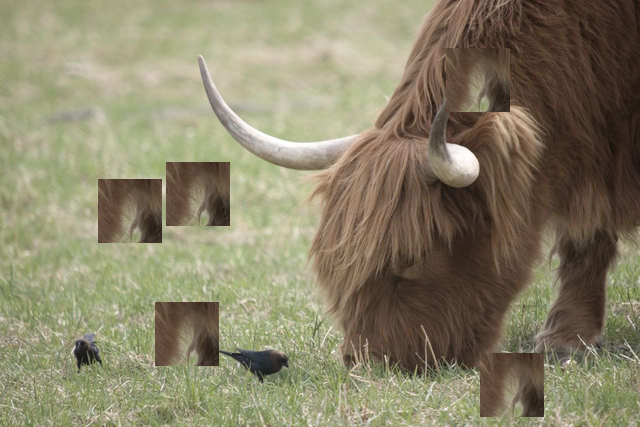
Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian.

### Data Masukan

Data masukan dari tugas akhir ini adalah sebagai berikut:

1. Citra *copy-move*

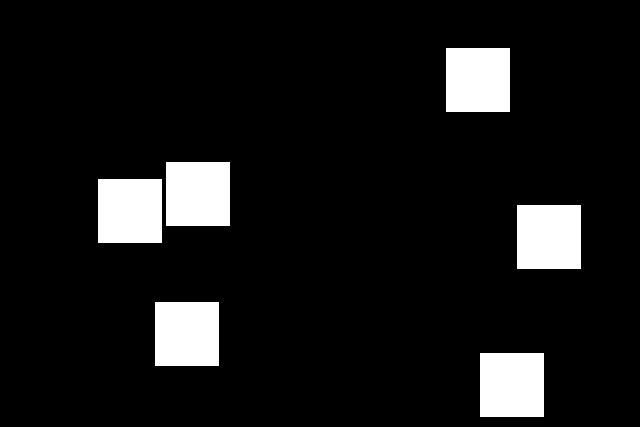
Citra *copy-move* merupaka citra yang sudah terkena serangan *copy-move*. Gambar 3.1 merupakan contoh citra yang terkena serangan *copy-move.*



Gambar . Contoh citra *copy-move*

1. Kunci jawaban citra *copy-move*

Kunci jawaban citra *copy-move* merupakan citra yang menggambarkan bagian mana saja yang terkena serangan *copy-move* dari citra sesungguhnya. Gambar 3.2 merupakan contohkunci jawaban citra yang sudah terkena serangan *copy-move*



Gambar . Contoh kunci jawaban citra *copy-move*

1. Threshold Config

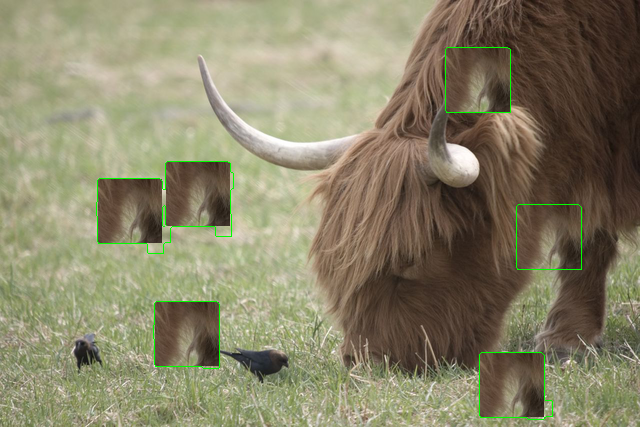
*Threshold config* merupakan data yang bertipe json yang berisi *config* dari *threshold* pada metode *expanding block algorithm* yang digunakan. Adapun parameter utama dari *threshold* yaitu *pVal*, *blockSize*, *numBuckets*, dan *minArea*. Membuat *threhsold config* terpisah dari program utama adalah untuk memudahkan pergantian percobaan nilai paramater yang digunakan. Pseudocode 3.1 merupakan contoh *threshold* *config* yang digunakan sebagai data masukan.

|  |
| --- |
| {  **"threshold"** :{  **"pVal"** : 0.95,  **"blockSize"** : 16,  **"blockDistance"** : 5,  **"numBuckets"** : 12000,  **"minArea"** : 100,  **"varianceThreshold"** : 1024  },  **"pixelNumber"**: {  **"pixel4"** : [215, 40, 40, 0.9],  **"pixel3"** : [0, 255, 0]  } } |

Pseudocode . Contoh Threshold Config

### Data Keluaran

Seteleah data masukan diproses menggunakan metode *expanding block algorithm*, program mengeluarkan 2 buah data yaitu citra *mask* yang mirip dengan citra kunci jawaban dan citra dengan *edge* bewarna sebagai penanda bagian yang terkena *copy- move* pada citra. *Edge* Gambar 3.3 merupakan contoh keluaran citra yang terdeteksi serangan *copy-move*.

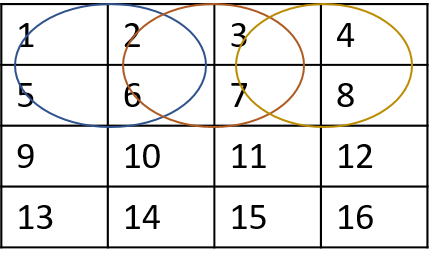


Gambar . Contoh keluaran citra terdeteksi *copy-move*

## Dekripsi Umum Sistem

Rancangan perangkat lunak pendeteksian *copy-move* pada citra menggunakan metode *expanding block algorithm* dimulai dengan pengolahan citra atau yang disebut *image processing* yaitu dengan cara tanpa *filter*, menggunkan *averaging filter* atau menggunakan *median filter*. *Image processing* berguna untuk membantu mengurangi *noise* yang ada pada suatu citra.

Setelah melakukan *image processing* piksel piksel yang ada pada suatu citra akan mengalami perubahan. Proses selanjutnya adalah mengkonversi citra RGB ke dalam bentuk *grayscale* untuk proses pembuatan *overlapping block* dengan ukuran *blockSize x blockSize*. Contoh *overlapping block* pada citra digambarkan pada Gambar 3.4.



Gambar . Overlapping Block 2 x 2

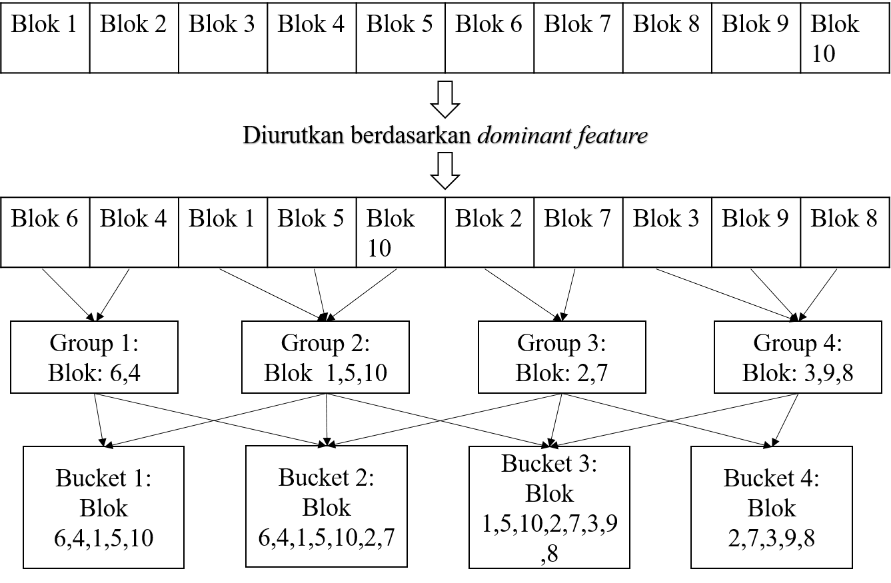
Tiap kotak pada Gambar 3.4 diasumsikan adalah piksel pada citra. Proses *overlapping* pertama digambarkan pada area yang dibatasi dengan warna biru, *overlapping* kedua berada pada area yang dibatasi dengan warna merah. Banyaknya *overlapping block* sebesar *B x B* pada sebuah citra berukuran *M x N* dapa dihitung menggunakan Persamaan 3.1.

(3.1)

Setelah pembuatan *overlapping block* selesai, untuk setiap blok dicari *dominant feature*. Blok akan diurutkan berdasarkan *dominant feature* secara *ascending*. Proses pengurutan ini menyebabkan blok yang mirip akan berdekatan.

Setelah blok diurutkan berdasarkan *dominanant feature*, tahap selanjutnya adalah membuat *groups* sebanyak *numBuckets*, blok akan ditempakan ke dalam *numBuckets groups* secara merata.

Setelah proses penempatan blok kedalam *groups* selesai tahap selanjutnya adalah membuat *buckets* sebanyak *numBuckets*, untuk penempatan *groups* kedalam *buckets* adalah tempatkan *groups* ke i-1, i, dan i+1 kedalam *buckets* i. Adapun proses pengurutan blok penempatan ke *groups* dan penempatan ke *buckets* dapat dilihat pada Gambar 3.5.



Gambar . Proses Pengurutan Blok dan Penempatan ke Buckets

Proses selanjutnya adalah proses pendeteksian *copy-move*. Setiap blok yang ada didalam buckets akan diproses. Proses yang terjadi di dalam bucketdimulai dengan mendefenisikan *S = 2*, dimana *S* merepresentasikan ukuran dari perbandingan sub blok yaitu dimulai dengan *S x S* piksel. Proses untuk setiap *bucket* adalah sebagai berikut:

1. Misalkan ada N blokdi dalam bucket. Defenisikan sebuah *N x N* matriks yang disebut *connection matrix*, matriks ini mendakan kesamaan bloksatu sama lain. Defenisikan semua nilai *connection matrix* dengan nilai satu, yang pada awalnya semua blok dianggap sama satu sama lain.
2. Jika ukuran 2 blok kurang dari *blockSize* , maka 2 blok tersebut *overlap*, defenisikan *connection matrix* menjadi nol untuk blok tersebut.
3. Hitung statistik untuk setiap pasangan perbandingan sub blok *S x S* di setiap blok yang ada di dalam bucket, jika hasil statistik perbandingan sub blok *S x S* kecil dari *pVal*, defenisikan nilai *connection matrix* menjadi nol untuk blok tersebut.
4. Untuk setiap bucket, jika *connection matrix* memiliki deretan angka nol, maka blok yang sesuai dengan baris tersebut tidak terhubung ke setiap blok yang ada di dalam bucket, keluarkan blok tersebut dari butket.

Jika *S < blockSize* ulangi langkah-langkah yang ada diatas dimana nilai *S = min(S2, blockSize)*. Dari sisa blok yang ada di dalam bucket hitung total area, jika total area kecil dari *minArea* maka lewatkan pengecekan dari blok yang tersisa. Sisa blok diasumsikan sebagai bagian dari *copy-move*.

Tahap selanjutnya adalah pemubuatan *mask* hasil deteksi *copy-move*. Awal pembuatanya adalah dengan membuat *mask* bewarna dasar hitam dengan ukuran sesuai dengan citra yang terkena serang *copy-move*. Dari sisa blok, ambil posisi titik tepi kiri atas, dari titik tersebut lalu sepanjang *blockSize* dan selebar *blockSize* ubah warna menjadi warna putih. Hasil dari pembuatan *mask* ini adalah citra hasil deteksi *copy-move*.

## Modifikasi *Expanding Block Algorithm*

Modifikasi yang dilakukan adalah menambahkan *filter* sebelum proses deteksi *copy-move* dilakukan. *Filter* bertujuan untuk mengurangi *noise* yang terjadi pada citra yang terkena serangan *copy-move*. Selanjutnya adalah pembuatan *edge* pada citra yang terkena seranga *copy-move* untuk mendapatkan keluaran citra *copy-move* bergaris tepi bewarna menggunakan *canny-edge*. Selanjutnya ada pembuatan *threshold* *config* ini bertujuan untuk melakukan pendeteksian *copy-move* secara efisien.

## Metode Filter

Filtermerupakan teknik untuk memodifikasi atau meningkatkan sebuah gambar. Filter juga dapat untuk menekan fitur tertentu atau mengapus fitur dalam sebuah gambar. Pengolahan gambar dengan filter antara lain adalah *smoothing*, *sharpening*, *blurring* ,*edge enhancement* dan lain-lain. Filter juga dapat diartikan sebagai operasi mengganti nilai dari sebuah piksel.

### *Averaging Filter*

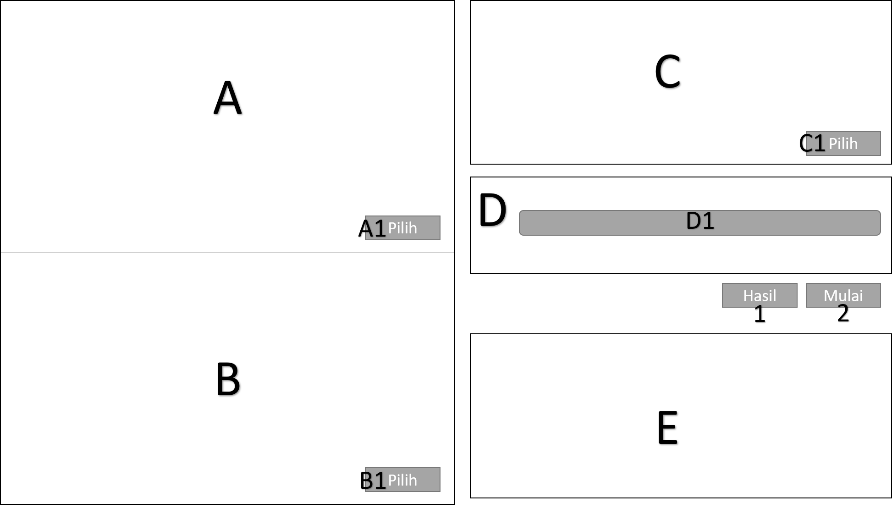
Averaging filter atau yang disebut juga box blur. Adalah metode filter yang sederhana, dan mudah untuk menerapkanya. Yaitu dengan mengganti setiap nilai piksel dalam sebuah gambar dengan rata-rata nilai piksel tetangganya dan termasuk nilai sendiri. Filter ini mempunyai efek menghilangkan nilai-nilai piksel yang representatif.

### *Median Filter*

Median filter merupakan order-statistic yang paling banyak diketahui, yang mana sesuai namanya, mengganti nilai pixel dengan median dari gray level dalam neighbourhood dari pixel tersebut.Nilai asli dari pixel diikutkan dalam komputasi median. Median filter sangat popular karena, untuk tipe – tipe tertentu *random noise*, median filter memberikan kemampuan *noise-reduction* yang sangat baik, dengan *blurring* yang lebih sedikit daripada *linear smoothing* filter dari ukuran yang sama. Median filters secara khusus efektif dalam keadaan bipolar dan unipolar *impulse noise*. Median filter memberikan hasil yang sangat bagus untuk image yang dirusak oleh *noise*.

## Perancangan Antarmuka

Aplikas antarmuka digunakan untuk mempermudah pengguna dalam melakukan pendeteksian serangan *copy-move* pada citra mengguanakan *expanding block algorithm*. Aplikasi antarmuka yang dibuat pada tugas akhir ini menggunakan pustaka Qt pada bahasa pemrograman python dengan ditribusi pustaka PyQt. Aplikasi yang dibangun dengan menggunakan pusataka Qt merupakan aplikasi *multiplatform* yang dapat berjalan di berbagai sistem operasi. Gambar 3.10 merupakan desain tampilan antarmuka yang dibuat pada tugas akhir ini.



Gambar . Desain Tampilan Antarmuka

Adapun komponen yang terdapat dalam Gambar 3.10 antara lain:

1. Bagian A merupakan form untuk memasukan data *copy-move* citra. Pada saat awal aplikasi dijalankan tidak ada gambar di bagian A, setelah gambar dimasukan dengan menekan tombol A1 yang akan menampilkan *file explorer* maka bagian A akan terdapat gambar *copy-move* citra yang telah terpilih.
2. Bagian B merupakan form untuk memasukan data kunci jawaban *copy-move* citra. Pada saat awal aplikasi dijalankan tidak ada gambar di bagian B, setelah gambar dimasukan dengan menekan tombol B1 yang akan menampilkan *file explorer* maka bagian B akan terdapat gambar kunci jawaban *copy-move* citra yang telah terpilih.
3. Bagian C merupakan form untuk memasukan *threshold config.* Pada saat awal aplikasi dijalankan tidak ada informasik *config* di bagian C, setelah threshold config dimasukan dengan menekan tombol C1 yang akan menampilkan file explorer maka bagian C akan terdapat informasi *threshold config* yang telah terpilih.
4. Bagian D merupakan form untuk memilih filter, pada saat awal aplikasi dijalankan tidak ada filter yang terpilih pada bagian D, dengan memilih *combo box* pada bagian D1 maka akan menampilkan *list* filter yang dapat dipilih.
5. Bagian E merupakan form untuk melihat *log* dari aplikasi, setelah aplikasi dijalankan dengan menekan tombol 2 maka pendeteksian *copy-move* akan berjalan dan menampilkan log pada bagian E, sedangkan tombol 1 merupkan tombol untuk menampilkan folder hasil pendeteksian *copy-move.*

***[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 program.

## Lingkungan Implementasi

Implementasi pendeteksian serangan *copy-move* pada citra dengan metode *expanding block algorithm* menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1.

Tabel . Lingkungan Implementasi Perangkat Lunak

|  |  |  |
| --- | --- | --- |
| Perangkat | Jenis Perangkat | Spesifikasi |
| Perangkat Keras | Prosesor | Intel(R) Core(TM) i7-4720 HQ @ 2.6 GHz (8 CPUs) |
| Memori | 8 GB 1600 MHz DDR3 |
| Perangkat Lunak | Sistem Operasi | Windows 10 dan Mint Cinnamon 64 bit |
| Perangkat Pengembang | PyCharm Professional 2016.2.3 |

## Implementasi

Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan *Pseudocode* yang digunakan mulai tahap *preprocessing* hingga pendeteksian citra menggunakan metode *expanding block algorithm*. Pada tugas akhir ini, implementasi menggunakan bahasa pemrograman python. Program yang dibangun menggunakan konsep *Object-Oriented Programming* pada bahasa pemrograman python.

### Kelas Block Container

Kelas blok digunakan sebagai objek atau *container* yang digunakan untuk menampung blokdari proses membangun *overlapping block*.

Kelas blok nantinya digunakan untuk melakukan proses pengelompokan blok yang mirip dengan mencari kemiripan fitur variance. Untuk detail dari kelas blok *container* dapat dilihat pada Pseudocode 4.1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **CLASS** Block**:**  **FUNCTION** \_\_init\_\_(self, image, row, col, blockSize):  rowEnd <- row + (blockSize-1)  colEnd <- col + (blockSize-1)  row <- row  col <- col  overlap <- image [row:rowEnd,col:colEnd]  variance <- np.var(image)  **ENDFUNCTION** |

Pseudocode . Kode Program Kelas Blok Container

#### Detail Atribut Kelas Blok Container

Untuk detail dari setiap atribut yang ada di kelas blok container dapat dilihat pada Tabel 4.2

Tabel . Atribut Kelas Blok Container

|  |  |
| --- | --- |
| Nama Atribut | Keterangan |
| row | Merupakan indeks baris dari blok *overlap* |
| col | Merupakan indeks kolom dari blok *overlap* |
| blockImage | Merupakan blok overlap dengan ukuran blockSize x blockSize |
| variance | Merupakan nilai variance dari block *overlap* |

### Inisialisasi File

Inisialiasi file merupakan proses awal dari program, pada inisialisasi file program akan mendefenisikan *path* citra masukan, *path* *threhsold config* dan filter yang dipilih. Setelah itu program membaca citra masukan, konversi ke grayscale, mencari ukuran citra, melakukan filter terhadap citra dan panjang satu piksel. Proses inisialisasi file dapat dilihat pada Pseudocode 4.2.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | **CLASS** ExpandingBlock:  **FUNCTION** \_\_init\_\_(self, fileName, fileConfig, statusFilter):  fileName <- fileName  fileConfig <- fileConfig  statusFilter <- statusFilter  initFile()  initThreshold()  **ENDFUNCTION**  **FUNCTION** initFile(self):  **try**:  image <- cv2.imread( fileName)  **IF** statusFilter = 1:  averagingFilter()  **ENDIF**  **IF** statusFilter = 2:  medianFilter()  **ENDIF**  **IF** statusFilter = 3:  bilateralFilter()  **ENDIF**  grayImage <- cv2.cvtColor( image, cv2.COLOR\_BGR2RGB)  height, width, \_ <- np.shape( image)  lenpixel <- len( image[0][0])  except:  **OUTPUT** 'File Not Found'  **ENDFUNCTION** |

Pseudocode . Kode Program Inisialisasi File

Proses inisialisasi file dimulai dari pustaka *openCV* dengan modul *io* membaca file dan ditampung ke variabel image, setelah itu gambar dikonversi menjadi gambar hitam karena untuk proses pendeteksian *copy-move* dibutuhkan gambar hitam putih, konversi inimenggunakan module *openCV BGR2GRAY*. Proses inisialisasi file juga menyimpan ukuran gambar serta mencaru panjang satu pixel dari gambar. Ketika file tidak ditemukan maka program akan menampilkan pesan *error*.

### Inisialisasi Threshold

Insialisasi *threshold* merupakan proses membaca file *threshold* yang bertipe *json*, dimana nantinya di proses ini nilai *threshold* pada program akan di deklarasikan. Proses inisialiasi *threshold* dapat dilihat pada Pseudocode 4.3.

|  |  |
| --- | --- |
| 1  2  4  3  6  7  8  9  10  11  12  13 | **FUNCTION** initThreshold(self):  **with** open( fileConfig,'rb') as file:  data <- json.load(file)  minArea <- data['threshold']['minArea']  blockSize <- data['threshold']['blockSize']  numBuckets <- data['threshold']['numBuckets']  pVal <- data['threshold']['pVal']  blockDistance <- data['threshold']['blockDistance']  varianceThreshold <- data['threshold']['varianceThreshold']  pixel3 <- data['threshold']['pixel3']  pixel4 <- data['threshold']['pixel4']  **ENDFUNCTION** |

Pseudocode . Kode Program Inisialisasi Threshold

### Pembuatan *Overlapping Block*

*Overlapping block* merupakan proses membuat blok yang *overlap* dari gambar asli. Block yang dibuat merupakan blok dinamis sesuai dengan config yang telah ditetapkan. Proses selanjutnya blok yang telah dibuat akan dibagi ke dalam grup secara merata. Proses pembuatan overlapping block dapat dilihat pada Pseudocode 4.4.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **FUNCTION** overlappBlock(self):  overlappingBlockLen <- ( height - blockSize + 1) \* ( width - blockSize + 1)  overlappingBlock <- []  **for** i in range(overlappingBlockLen):  row <- i / (height - blockSize+1)  col <- i % (height - blockSize+1)  overLappingBlock.append(Container(block,row,col, blockSize))  **ENDFOR**  **ENDFUNCTION** |

Pseudocode . Kode Program Overlapping block

Setelah pembuatan overlapping block selesai proses selanjutnya adalah mengurutkan blok berdasarkan dominan fitur. Fitur blok yang digunakan adalah variance, sehingga blok akan diurutkan berdasarkan variance secara *ascending*. Proses pengurutan blok dapat dilihat pada Pseudocode 4.5.

|  |  |
| --- | --- |
| 1  2  3 | **FUNCTION** overlappBlockSort(self):  overlappingBlock.sort(key=lambda value: value.variance)  **ENDFUNCTION** |

Pseudocode . Kode Program Pengurutan Blok

### Pembuatan Grup

Proses pembuatan grup merupakan proses membagi blok secara merata ke dalam grup. Banyak grup yang dibuat adalah sebanyak *numBuckets* yang telah ditetapkan pada saat proses inisialisasi *threshold*. Proses pembuatan grup dapat dilihat pada Pseudocode 4.6.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **FUNCTION** createGroup(self):  groups <- [[] for i in range(numBuckets) **ENDFOR**]  blocksPerGroup <- int(overlappingBlockLen / numBuckets)  group <- 0  count <- 0  **for** block in overlappingBlock:  groups[group].append(block)  count += 1  **IF** count > blocksPerGroup:  group += 1  count <- 0  **ENDIF**  **ENDFOR**  **ENDFUNCTION** |

Pseudocode . Kode Program Pembuatan Grup

Prose awal pembuatan grup adalah dengan mendeklarasikan sebuah array dengan isi array kosong sepanjang *numBuckets* yang telah ditentukan. Setelah itu program akan mendefenisikan banyak blok yang ada di dalam satu grup yaitu dengan cara total panjang *overalapping block* dibagi dengan *numBuckets*. Setelah mendapatkan banyak blok yang ada di dalam grup, program akan membagi secara merata blok ke dalam grup yaitu dengan cara memasukan blok sebanyak *blockpergroup* yang telah ditentukan ke dalam satu grup.

### Pembuatan Bucket

Proses pembuatan *bucket* merupakan proses membagi grup yang telah dibuat ke dalam *bucket*. Banyak *bucket* yang dibuat adalah sebanyak *numBuckets*. Proses pembuatan *bucket* dapat dilihat pada Pseudocode 4.7.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | **FUNCTION** createBucket(self):  buckets <- [None]\* numBuckets  **for** i in range(numBuckets):  **try**:  buckets[i] <- groups[i-1] + groups[i] + groups[i+1]  **except** IndexError:  **IF** i = numBuckets - 1:  buckets[i] <- groups[i-1]+ groups[i]  **ELSE**:  raise IndexError  **ENDIF**  **ENDFOR**  **ENDFUNCTION** |

Pseudocode . Kode Program Pembuatan Bucket

Proses awal pembuatan *bucket* adalah dengan mendeklarasikan array sepanjang *numBuckets* yang telah ditentukan. Setelah itu proses meletkan grup ke dalam *bucket*. Proses meletakan grup kedalam bucket mempunyai alur yaitu meletkan grup i – 1, i, i + 1 ke dalam *bucket* i.

### Calculate Statistic

Proses ini yaitu untuk menghitung statistik untuk setiap pasangan perbandingan sub blok *S x S* di setiap blok yang ada di dalam bucket. Adapun proses untuk mengitung statistik dapat dilihat pada Pseudocode 4.8.

|  |  |
| --- | --- |
| I | Sublock |
| O | Sublock Statistic |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **FUNCTION** calculateStatistic(self, subBlocks, subSize):  lenSubBlocks <- len(subBlocks)  statistic <- np.zeros((lenSubBlocks,lenSubBlocks))  variance <- [np.var(subBlock) for subBlock in subBlocks]  for i, subBlock in enumerate(subBlocks):  differencePixel <- np.sum((subBlock - subBlocks)\*\*2, axis=(1,2))  sigmaSq <- (variance[i] + variance) / 2  **try**:  statistic[i] <- differencePixel / (sigmaSq\*subSize)  **except** ZeroDivisionError:  statistic[i] <- 0  **ENDFOR**  **RETURN** statistic  **ENDFUNCTION** |

Pseudocode . Kode Program Perhitungan Statistik

### Find Overlap Block

Proses ini adalah proses pencarian overlap setiap blok yang ada di dalam bucket. Tahapanya adalah mengambil *row* dan *col* dari setiap blok yang ada di dalam bucket. Selanjutnya yaitu mencari nilai mutlak dari *row* dan *col*. Sebua blok dinyatakan *overlap* ketika nilai mutlak dari *row* atau *col* kurang dari *blockSize* yang telah ditentukan. Adapun proses pengecekan *overlap* blok dapa dilihat pada Pseudocode 4.9.

|  |  |
| --- | --- |
| I | Bucket |
| O | Status Overlap |
| 1  2  3  4  5  6  7  8  9 | **FUNCTION** findOverLap(self, bucket):  row <- [block.row for block in bucket]  col <- [block.col for block in bucket]  rowDistance <- np.abs(row - np.reshape(row, (-1, 1)))  colDistance <- np.abs(col - np.reshape(col, (-1, 1)))  rowOverlap <- rowDistance < blockSize  colOverlap <- colDistance < blockSize  **RETURN** np.logical\_or(rowOverlap, colOverlap)  **ENDFUNCTION** |

Pseudocode . Kode Program Pengecekan Overlap

### Find Connection

Proses ini merupakan proses untuk menandakan apakah setiap blok yang ada di dalam bucket memiliki kesamaan satu sama lain. Proses ini akan mengebalikan sebuah array yang berisi status dari bucket tersebut. Proses pengecekan ini dapat dilihat pada Pseudocode 4.10.

|  |  |
| --- | --- |
| I | Statistic dan Status Overlap |
| O | Status Connection |
| 1  2  3  4  5  6 | **FUNCTION** findConnection(self, statistic, overLap):  tooSimilar <- statistic < pVal  connection <- np.ones\_like(tooSimilar)  connection <- np.logical\_xor(connection, np.logical\_or(overLap, np.logical\_not(tooSimilar)))  **RETURN** connection  **ENDFUNCTION** |

Pseudocode . Kode Program Pengekan Connection

### Bucket Process

Proses ini merupakan proses pengecekan setiap bucket, proses ini. Proses yang terjadi di dalam bucket dimulai dengan mendefenisikan S = 2, dimana S merepresentasikan ukuran dari perbandingan sub blok yaitu dimulai dengan S x S piksel.

Proses untuk setiap bucket dimulai dengan memisalkan ada N blok di dalam bucket. Defenisikan sebuah N x N matriks yang disebut connection matrix, matriks ini mendakan kesamaan blok satu sama lain. Defenisikan semua nilai connection matrix dengan nilai satu, yang pada awalnya semua blok dianggap sama satu sama lain. Jika ukuran 2 blok kurang dari blockSize , maka 2 blok tersebut overlap, defenisikan connection matrix menjadi nol untuk blok tersebut. Selanjutnya yaitu menghitung statistik untuk setiap pasangan perbandingan sub blok S x S di setiap blok yang ada di dalam bucket, jika hasil statistik perbandingan sub blok S x S kecil dari pVal, defenisikan nilai connection matrix menjadi nol untuk blok tersebut. Untuk setiap bucket, jika connection matrix memiliki deretan angka nol, maka blok yang sesuai dengan baris tersebut tidak terhubung ke setiap blok yang ada di dalam bucket, keluarkan blok tersebut dari butket. Jika *S < blockSize* proses ini akan mengulangi langkah-langkah yang ada diatas dimana nilai *S = min(S2, blockSize)*. Dari sisa blok yang ada di dalam bucket hitung total area, jika total area kecil dari minArea maka lewatkan pengecekan dari blok yang tersisa. Prose ini akan mengembalikan bucket berisi blok yang merupakan bagian yang terkena serangan *copy-move*. Pemrosesan bucket ini dapat dilihat pada Pseudocode 4.11.

|  |  |
| --- | --- |
| I | Bucket |
| O | Bukcet dengan status *copy-move* |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | **FUNCTION** bucketProcess(self, bucket):  subSize <- 1  **while** subSize < blockSize:  **IF** len(bucket)\* blockSize < minArea:  RETURN []  **ENDIF**  subSize <- min(subSize << 1, blockSize)  subBlocks <- [block.blockImage[0:subSize,0:subSize] for block in bucket]  statistic <- calculateStatistic(subBlocks, subSize)  overLap <- findOverLap(bucket)  connection <- findConnection(statistic, overLap)  connection <- np.any(connection, axis=0)  bucket <- [bucket[i] for i in range(len(bucket)) IF connection[i]]  **IF** len(bucket) \* blockSize < minArea:  **RETURN** []  **ENDIF**  **ENDWHILE**  **RETURN** bucket  **ENDFUNCTION** |

Pseudocode . Kode Program Memproses Setiap Bucket

### Build Copy-Move Image

Fungsi ini merupakan fungsi untuk menyimpan citra hasil deteksi *copy-move*. Fungsi ini berjalan dengan mengecek sisa blok yang ada dari hasil pendeteksian *copy-move*. Ketika sisa blok hasil deteksi *copy-move* adalah nol maka citra tidak terdeteksi adanya serangan *copy-move*. Ketika terdapat blok setelah proses pendeteksian *copy-move* maka fungsi ini akan memanggil fungsi *createMask*, *writeMask* dan Proses menyimpan citra hasil deteksi copy move dapat dilihat pada Pseudocode 4.12.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **FUNCTION** buildCopyMoveImage(self):  **IF** len( blocks) = 0:  **OUTPUT** "Fail to build copy move because no copy move detected"  **RETURN**  **ENDIF**  mask <- createMask()  imageOut <- np.uint8( writeMask(mask))  imageWedge <- buildEdgeImage(imageOut)  fileName <- fileName.split("/")[2]  imgKey <- io.imread("../Dataset-Fix/001\_B.png")  io.imsave("../Hasil/Mask-"+ fileName,imageOut)  io.imsave("../Hasil/Edge-"+ fileName,imageWedge)  accuracyMSE(imgKey,imageOut)  **ENDFUNCTION** |

Pseudocode . Kode Program Membangun Gambar *Copy-Move*

### Create Mask

Pembuatan *mask* adalah pembuatan citra hasil deteksi *copy-move*. Proses pembuatan *mask* dimulai membuat sebuah array 2 dimensi yang berusukuran sama seperti citra yang terkena serangan *copy-move*. Dari setiap blok yang tersisa dari pendeteksian *copy-move*, diambil koordinat dari blok tersebut sepanjang *blockSize* yang telah ditentukan untuk diisi sebuah *channel* yang menandakan daerah *copy-move*. Adapun proses pembuatan *mask* dapat dilihat pada Pseudocode 4.13.

|  |  |
| --- | --- |
| I | **-** |
| O | Mask |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | **FUNCTION** createMask(self):  RED <- 0  GREEN <- 1  BLUE <- 2  FILL\_CHANNEL <- 4  mask <- np.zeros(( height, width,3), dtype=np.uint8)  **for** block in blocks:  row <- block.row  col <- block.col  endRow <- row + blockSize  endCol <- col + blockSize  mask[row:endRow, col:endCol, RED] <- FILL\_CHANNEL  mask[row:endRow, col:endCol, BLUE] <- FILL\_CHANNEL  mask[row:endRow, col:endCol, GREEN] <- FILL\_CHANNEL  **ENDFOR**  **RETURN** mask  **ENDFUNCTION** |

Pseudocode . Kode Program Membuat Gambar Mask

### Write Mask

Proses penulisan *mask* adalah proses dimana mengganti dengan warna hitam ketika tidak terdapat serangan *copy-move* dan mengganti dengan warna putih jika daerah tersebut terdapat serangan *copy-move*. Proses ini dilakukan setelah proses pembuatan untuk menandakandaerah mana saja yang terserang *copy-move*. Keluaran dari proses ini adalah citra yang mirip dengan citra kunci jawaban yang menandakan daerah *copy-move* Adapun proses penulisan *mask*.

|  |  |
| --- | --- |
| I | Mask |
| O | Gambar dengan mask |
| 1  2  3  4  5  6  7  8  9 | **FUNCTION** writeMask(self,mask):  FILL\_CHANNEL <- 4  REMOVE\_CHANNEL <- 8  imgMasked <- np.zeros(( height, width, 3), dtype=np.uint8)  imgMasked[:] <- 0  imgMasked[mask = FILL\_CHANNEL] <- 255  imgMasked[mask = REMOVE\_CHANNEL] <- 0  **RETURN** imgMasked  **ENDFUNCTION** |

Pseudocode . Kode Program Menempelkan Mask Ke Gambar

### Build Edge Copy-Move Image

Proses pembuatan garis tepi pada citra *copy-move* adalah dengan menggunakan Canny-edge pada pustaka OpenCV. Dari citra hasil deteksi *copy-move* di prosess menggunakan cv2.Canny sehingga mendapatkan garis tepi di setiap bagian yang terdeteksi *copy-move*. Dari garis tepi tersebut diambil kordinatnya. Setelah itu dari kordinat tersebut pada citra *copy-move* asli diganti diganti dengan warna hijau untuk menandakan garis tepi di bagian citra yang terkena serangan *copy-move*. Proses pembuatan *edge* dapat dilihat pada Pseudocode 4.15.

|  |  |
| --- | --- |
| I | Gambar dengan mask |
| O | Gambar dengan edge |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **FUNCTION** buildEdgeImage(self, imageOut):  edges <- cv2.Canny(imageOut, height, width)  coord <- np.column\_stack(np.where(edges = 255))  imageWedge <- image  pixel <- pixel4 IF lenpixel = 4 else pixel3  **for** index in coord:  x <- index[0]  y <- index[1]  imageWedge[x, y] <- pixel  **ENDFOR**  **RETURN** imageWedge  **ENDFUNCTION** |

Pseudocode . Kode Program Membuat Edge Pada Gambar

### Mean Squared Error

*Mean squared error* adalah metode untuk mengevaluasi 2 buah citra yaitu kunci jawaban dengan citra hasil deteksi. Metode ini bekerja dengan cara masing masing kesalahan atau perbedaan piksel akan dikuadrakatkan. Metode ini mengatur kesalahan dalam pengukuran yang besar karena tiap kesalahan-kesalahn tersebut dikuadratkan. Proses perhitungan MSE dapat dilihat pada Pseudocode 4.16.

|  |  |
| --- | --- |
| I | Citra kunci jawaban dan citra hasil deteksi |
| O | MSE |
| 1  2  3  4 | **FUNCTION** accuracyMSE(self,imgKey,imgPredict):  mse <- mean\_squared\_error(imgKey.flatten(),imgPredict.flatten())  **RETURN** mse  **ENDFUNCTION** |

Pseudocode . Kode Program Menghitung Mean Squared Error

### Similarity

*Similarity* adalah proses pencarian kemiripan antar citra kunci jawaban dengan citra hasil deteksi *copy-move*. Proses pencarian kemiripan ini yaitu dengan membandingkan kemiripan setiap piksel citra kunci jawaban dengan citra hasil deteksi *copy-move*. Hasil kemiripan yaitu tota piksel yang sama dibagi total semua piksel. Adapun proses menghitung kemiripan dapat dilihat pada Pseudocode 4.17.

|  |  |
| --- | --- |
| I | Citra kunci jawaban dan citra hasil deteksi |
| O | Similarity |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **FUNCTION** findSimmiliarity(self, imageA, imageB):  same <- 0  **for** i in range( height):  **for** j in range( width):  **IF** all(imageA[i,j]==imageB[i,j]):  same += 1  **ENDIF**  **ENDFOR**  **ENDFOR**  simmiliarity <- (float(same)/ size) \* 100  **RETURN** simmiliarity  **ENDFUNCTION** |

Pseudocode . Kode Program Menghitung Kemiripan

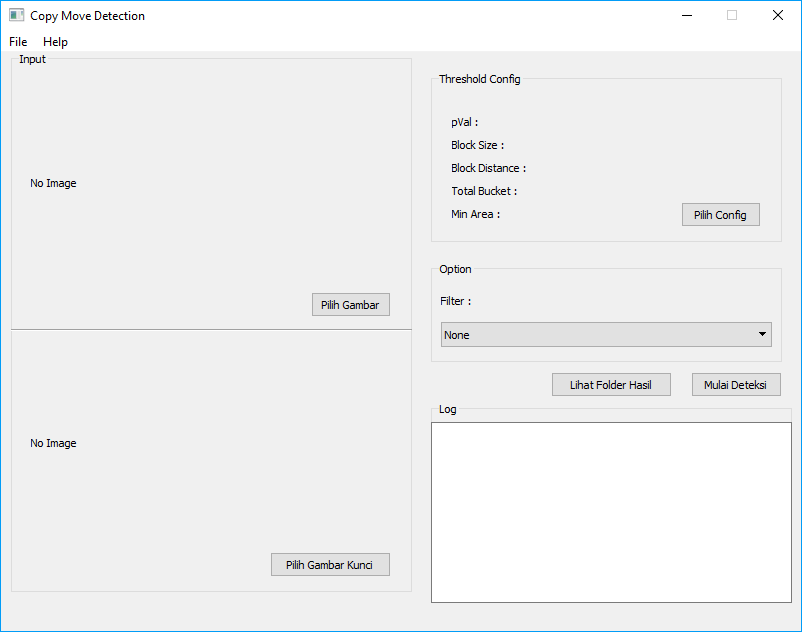
### Pembuatan Antarmuka

Antarmuka diimplementasikan menggunakan pustaka Qt dengan bahasa pemrograman python memakai modul PyQt. Degan menggunakan Qt memungkinkan aplikasi jalan di berbagai *platform*. Antarmuka merupakan perantara untuk menjalankan program utama yaitu deteksi *copy-move* pada citra. Adapun proses antarmuka menjalan deteksi dapat dilihat pada Pseudocode 4.8.

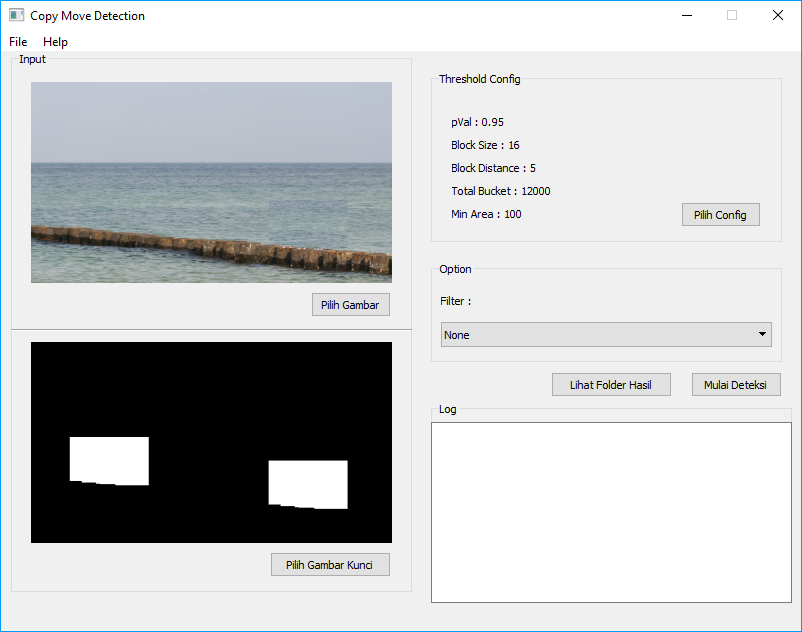
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | **FUNCTION** getFilter(self,i):  filterChoice <- str(i)  **ENDFUNCTION**  **FUNCTION** append(self, text):  cursor <- logOutput.textCursor()  cursor.movePosition(cursor.End)  cursor.insertText(text)  **ENDFUNCTION**  **FUNCTION** stdoutReady(self):  text <- str( process.readAllStandardOutput())  OUTPUT text.strip()  append(text)  **ENDFUNCTION**  **FUNCTION** stderrReady(self):  text <- str( process.readAllStandardError())  OUTPUT text.strip()  append(text)  **ENDFUNCTION**  **FUNCTION** startDetection(self):  argv <- ["/media/sudirahanif/Kuliah/TA/Program/TugasAkhir.py", imageFile, thresholdFile, filterChoice]  process.start('/home/sudirahanif/anaconda2/bin/python',argv)  **ENDFUNCTION**  **FUNCTION** initUI(self):  buttonStart.clicked.connect(startDetection)  process <- QProcess(self)  process.readyReadStandardOutput.connect(stdoutReady)  process.readyReadStandardError.connect(stderrReady)  process.started.connect(lambda: buttonStart.setEnabled(False))  process.finished.connect(lambda: buttonStart.setEnabled(True))  **ENDFUNCTION** |

Pseudocode . Kode Program Menjalankan Deteksi

Ketika tombol *start* ditekan maka program deteksi copy-move akan dijalankan yaitu dengan menggunakan *thread*, antarmuka akan menjalankan proses memanggil program deteksi *copy-move* dengan masukan argumen yaitu *path* file citra *copy-move*, *path* file citra kunci jawaban, serta filter. Untuk implementasi antarmuka utama dapat dilihat pada



Gambar . Antarmuka Awal Dijalankan



Gambar . Antarmuka Masukan Data

# BAB V HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pada deteksi serangan *copy-move* pada *dataset* citra yang terkena serang *copy-move*. Hasil uji coba didapatkan dari implementasi pada bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

## Lingkungan Pengujian

Lingkungan pengujian pada uji coba permasalahan pendeteksian serangan *copy-move* pada citra dengan metode *expanding block algorithm* dengan menggunakan spesifikasi keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1.

Tabel . Spesifikasi Lingkungan Pengujian

|  |  |  |
| --- | --- | --- |
| Perangkat | Jenis Perangkat | Spesifikasi |
| Perangkat Keras | Prosesor | Intel(R) Core(TM) i7-4720 HQ @ 2.6 GHz (8 CPUs) |
| Memori | 8 GB 1600 MHz DDR3 |
| Perangkat Lunak | Sistem Operasi | Windows 10 dan Mint Cinnamon 64 bit |
| Perangkat Pengembang | PyCharm Professional 2016.2.3 |

## Data Pengujian

Adapun data pengujian dalam tugas akhir ini adalah sebagai berikut:

1. Citra *Copy-move*

Data terdiri dari 20 buah citra yang mengalami serangan *copy-move,* Juga terdapat 20 buah citra kunci jawaban yang menunjukan bagian-bagian yang dideteksi terkena serangan *copy-move.* Citrayang digunakan adalah citra yang beresolusi kecil. Adapun penjelasan 22 buah citra tersebut adalah sebagai berikut:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Nama | Ukuran File | Ukuran Gambar | Sumber |
| 1 | 01\_giraffe.png | 267 x 400 | 0.228 MB | Internet |
| 2 | 02\_tree.png | 342 x 512 | 0.294 MB | Internet |
| 3 | 03\_clean\_wall.png | 377 x 512 | 0.358 MB | Internet |
| 4 | 04\_cattle.png | 427 x 640 | 0.523 MB | Internet |
| 5 | 05\_knight\_moves.png | 720 x 720 | 0.888 MB | Internet |
| 6 | 06\_three\_hundred.png | 791 x 1181 | 1.795 MB | Internet |
| 7 | 07\_lone\_cat.png | 512 x 512 | 0.587 MB | Internet |
| 8 | 08\_Coin.png | 512 x 512 | 0.425 MB | Internet |
| 9 | 09\_fountain.png | 512 x 512 | 0.42 MB | Internet |
| 10 | 10\_park.png | 512 x 512 | 0.556 MB | Internet |
| 11 | 11\_barrier.png | 512 x 512 | 0.395 MB | Membuat Sendiri |
| 12 | 12\_cattle\_remake.png | 512 x 512 | 0.415 MB | Membuat Sendiri |
| 13 | 13\_clean\_walls\_remake.png | 512 x 512 | 0.525 MB | Membuat Sendiri |
| 14 | 14\_horses.png | 512 x 512 | 0.481 MB | Membuat Sendiri |
| 15 | 15\_four\_babies.png | 512 x 512 | 0.457 MB | Membuat Sendiri |
| 16 | 16\_malawi.png | 512 x 512 | 0.357 MB | Membuat Sendiri |
| 17 | 17\_mykene.png | 512 x 512 | 0.522 MB | Membuat Sendiri |
| 18 | 18\_knight\_moves\_remake.png | 512 x 512 | 0.385 MB | Membuat Sendiri |
| 19 | 19\_Street\_building.png | 512 x 512 | 0.416 MB | Membuat Sendiri |
| 20 | 20\_Book.png | 512 x 512 | 0.467 MB | Membuat Sendiri |

1. *Threshold Config*

Data terdiri dari 5 *threshold* dengan 4 nilai parameter yaitu *pVal, numBuckets, blockSize, minArea*. Untuk *threshold* pertama yaitu *threshold* yang telah ditetapkan, kedua yaitu mengganti nilai *pVal*, ketiga yaitu mengganti nilai *numBukets*, keempat yaitu mengganti nilai *blockSize* dan kelima yaitu mengganti nilai *minArea*. *Threshold* akan dibuat dengan format json sehingga memudahkan penggunaan.

Tabel . Threshold a

|  |  |
| --- | --- |
| Parameter | Nilai |
| *pVal* | 0.99 |
| *numBuckets* | 12000 |
| *blockSize* | 16 |
| *minArea* | 100 |

Pada Tabel 4.1 merupakan tabel threshold-a, yang merupakan threshold yang telah ditetapkan. Untuk threshold-b yaitu mengganti nilai *pVal* menjadi 0.89, Untuk threshold-c yaitu mengurangi nilai numBucket menjadi 6000, untuk threshold-d yaitu mengganti nilai *numBuckets* menjadi 8 sedangkan untuk threshold-e yaitu mengganti nilai minArea jadi 50. Pada tahap menggunakan threshold ini yaitu untuk mencoba apa pengaruh masing-masing threhsold terhadap jalanya program pendeteksian *copy-move*.

## Preprocessing citra

Preprocessing citra yang digunakan dalam skenario uji coba adalah pada tahap *filtering* yaitu tahap untuk meminimalisasikan *noise* pada suatu citra. Metode *filtering* yang dibandingkan dalam skenario uji coba ini adalah metode *Averaging* dan *Median* yang telah dijelaskan pada bab sebelumnya.

## Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik. Terdapat 20 macam skenario uji coba, yaitu:

1. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-a dan tanpa metode *Filter.*
2. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-a dan metode *Averaging* *Filter.*
3. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-a dan metode *Median* *Filter.*
4. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-b dan tanpa metode *Filter.*
5. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-b dan metode *Averaging* *Filter.*
6. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-b dan metode *Median* *Filter..*
7. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-c dan tanpa metode *Filter.*
8. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-c dan metode *Averaging* *Filter.*
9. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-c dan metode *Median* *Filter.*
10. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-d dan tanpa metode *Filter.*
11. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-d dan metode *Averaging* *Filter.*
12. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-d dan metode *Median* *Filter.*
13. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-e dan tanpa metode *Filter.*
14. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-e dan metode *Averaging* *Filter.*
15. Penghitungan akurasi, MSE, *similarity* pendeteksian *copy-move* citra dengan menggunakan threshold-e dan metode *Median* *Filter.*

### Skenario Uji Coba 1

Skenario uji coba 1 adalah penghitungan akurasi, MSE, similarity pendeteksian *copy-move* citra dengan menggunakan threshold-a dan tanpa metode *filter*.

Tabel . *Confusion matrix* uji coba 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 | 4 | 0 | 0 | 0 | 1583.61 | 99.19% | 100% |
| 2 | 5 | 0 | 0 | 0 | 6503.84 | 96.67% | 100% |
| 3 | 5 | 0 | 0 | 0 | 4354.79 | 97.77% | 100% |
| 4 | 5 | 0 | 0 | 0 | 1105.72 | 99.43% | 100% |
| 5 | 5 | 28 | 0 | 0 | 3237.33 | 98.34% | 15.16% |
| 6 | 5 | 0 | 0 | 0 | 157.87 | 99.92% | 100% |
| 7 | 0 | 0 | 0 | 1 | 22511.45 | 0.00% | 0% |
| 8 | 2 | 0 | 0 | 0 | 244.08 | 99.87% | 100% |
| 9 | 1 | 0 | 0 | 0 | 419.71 | 8.41% | 100% |
| 10 | 1 | 0 | 0 | 0 | 913.82 | 99.53% | 100% |
| 11 | 1 | 0 | 0 | 0 | 532.2 | 10.15% | 100% |
| 12 | 1 | 0 | 0 | 0 | 349.19 | 10.09% | 100% |
| 13 | 1 | 0 | 0 | 0 | 328.89 | 8.34% | 100% |
| 14 | 0 | 0 | 0 | 1 | 15493.88 | 0.00% | 0% |
| 15 | 1 | 30 | 0 | 0 | 10488.6 | 10.69% | 3.22% |
| 16 | 0 | 0 | 0 | 1 | 56471.81 | 0.00% | 0% |
| 17 | 0 | 0 | 0 | 1 | 12739.56 | 0.00% | 0% |
| 18 | 0 | 0 | 0 | 1 | 60198.29 | 0.00% | 0% |
| 19 | 1 | 0 | 0 | 1 | 2244.36 | 98.85% | 50% |
| 20 | 1 | 0 | 0 | 0 | 400.35 | 99.79% | 100% |

### Skenario Uji Coba 2

Tabel . *Confusion matrix* uji coba 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 | 4 | 0 | 0 | 0 | 3981.87 | 97.96% | 100% |
| 2 | 5 | 0 | 0 | 0 | 7400.65 | 96.21% | 100% |
| 3 | 5 | 0 | 0 | 0 | 5047.06 | 97.41% | 100% |
| 4 | 5 | 0 | 0 | 0 | 2771.8 | 98.58% | 100% |
| 5 | 5 | 131 | 0 | 0 | 12865.39 | 93.40% | 3.67% |
| 6 | 5 | 14 | 0 | 0 | 1268.59 | 99.35% | 26.31% |
| 7 | 0 | 0 | 0 | 1 | 22511.45 | 0.00% | 0% |
| 8 | 2 | 3 | 0 | 0 | 4131.53 | 97.88% | 40% |
| 9 | 1 | 3 | 0 | 0 | 1932.35 | 7.81% | 25% |
| 10 | 1 | 12 | 0 | 0 | 3887.45 | 98.01% | 7.69% |
| 11 | 1 | 0 | 0 | 0 | 1024.31 | 9.72% | 100% |
| 12 | 1 | 0 | 0 | 0 | 1009.44 | 9.58% | 100% |
| 13 | 1 | 0 | 0 | 0 | 938.21 | 7.87% | 100% |
| 14 | 0 | 0 | 0 | 1 | 15493.88 | 0.00% | 0% |
| 15 | 1 | 30 | 0 | 0 | 18591.99 | 10.22% | 3.22% |
| 16 | 0 | 0 | 0 | 1 | 56471.81 | 0.00% | 0% |
| 17 | 0 | 7 | 0 | 0 | 13904.72 | 0.01% | 0% |
| 18 | 0 | 0 | 0 | 1 | 60198.29 | 0.00% | 0% |
| 19 | 0 | 0 | 0 | 2 | 2969.17 | 98.48% | 0% |
| 20 | 1 | 0 | 0 | 0 | 1553.79 | 99.20% | 0% |

### Skenario Uji Coba 3

Tabel . *Confusion matrix* uji coba 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 | 4 | 0 | 0 | 0 | 3214.72 | 98.35% | 100% |
| 2 | 5 | 0 | 0 | 0 | 7178.95 | 96.32% | 100% |
| 3 | 5 | 0 | 0 | 0 | 5010.68 | 97.43% | 100% |
| 4 | 5 | 0 | 0 | 0 | 1026.49 | 99.47% | 100% |
| 5 | 5 | 28 | 0 | 0 | 4716.95 | 97.58% | 15.16% |
| 6 | 5 | 0 | 0 | 0 | 393.42 | 99.80% | 100% |
| 7 | 0 | 0 | 0 | 1 | 22511.45 | 0.00% | 100% |
| 8 | 2 | 10 | 0 | 0 | 3353.15 | 98.28% | 16.67% |
| 9 | 1 | 1 | 0 | 0 | 1323.6 | 7.98% | 50% |
| 10 | 1 | 12 | 0 | 0 | 3371.75 | 98.27% | 7.6% |
| 11 | 1 | 0 | 0 | 0 | 1026.12 | 9.71% | 100% |
| 12 | 1 | 0 | 0 | 0 | 838.66 | 9.68% | 100% |
| 13 | 1 | 0 | 0 | 0 | 914.4 | 7.88% | 100% |
| 14 | 0 | 0 | 0 | 1 | 15493.88 | 0.00% | 0% |
| 15 | 1 | 30 | 0 | 0 | 14522.1 | 10.30% | 3.22% |
| 16 | 0 | 0 | 0 | 1 | 56471.81 | 0.00% | 0% |
| 17 | 0 | 0 | 0 | 1 | 12739.56 | 0.00% | 0% |
| 18 | 0 | 0 | 0 | 1 | 60198.29 | 0.00% | 0% |
| 19 | 0 | 0 | 0 | 2 | 2969.17 | 98.48% | 0% |
| 20 | 1 | 0 | 0 | 0 | 1381.15 | 99.29% | 100% |

### Skenario Uji Coba 4

Tabel . *Confusion matrix* uji coba 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 | 4 | 0 | 0 | 0 | 1583.61 | 99.19% | 100% |
| 2 | 5 | 0 | 0 | 0 | 6503.84 | 96.67% | 100% |
| 3 | 5 | 0 | 0 | 0 | 4354.79 | 97.77% | 100% |
| 4 | 5 | 0 | 0 | 0 | 685.28 | 99.65% | 100% |
| 5 | 5 | 26 | 0 | 0 | 2800.44 | 98.56% | 16.12% |
| 6 | 5 | 0 | 0 | 0 | 157.87 | 99.92% | 100% |
| 7 | 0 | 0 | 0 | 1 | 22511.45 | 0.00% | 0% |
| 8 |  |  |  |  | 244.08 | 99.87% |  |
| 9 |  |  |  |  | 419.71 | 8.41% |  |
| 10 |  |  |  |  | 913.82 | 99.53% |  |
| 11 |  |  |  |  | 532.2 | 10.15% |  |
| 12 |  |  |  |  | 349.19 | 10.09% |  |
| 13 |  |  |  |  | 328.89 | 8.34% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 9575.94 | 10.69% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60198.29 | 0.00% |  |
| 19 |  |  |  |  | 2244.36 | 98.85% |  |
| 20 |  |  |  |  | 400.35 | 99.79% |  |

### Skenario Uji Coba 5

Tabel . *Confusion matrix* uji coba 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 3981.87 | 97.96% |  |
| 2 |  |  |  |  | 7400.65 | 96.21% |  |
| 3 |  |  |  |  | 5027.86 | 97.42% |  |
| 4 |  |  |  |  | 2345.64 | 98.80% |  |
| 5 |  |  |  |  | 10116.51 | 94.81% |  |
| 6 |  |  |  |  | 938.03 | 99.52% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 3118.74 | 98.40% |  |
| 9 |  |  |  |  | 1956.16 | 7.79% |  |
| 10 |  |  |  |  | 3127.67 | 98.40% |  |
| 11 |  |  |  |  | 1012.75 | 9.72% |  |
| 12 |  |  |  |  | 1009.44 | 9.58% |  |
| 13 |  |  |  |  | 938.21 | 7.87% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 16950.47 | 10.18% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60198.29 | 0.00% |  |
| 19 |  |  |  |  | 2969.17 | 98.48% |  |
| 20 |  |  |  |  | 1553.79 | 99.20% |  |

### Skenario Uji Coba 6

Tabel . *Confusion matrix* uji coba 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 3216.55 | 98.35% |  |
| 2 |  |  |  |  | 7178.95 | 96.32% |  |
| 3 |  |  |  |  | 5010.68 | 97.43% |  |
| 4 |  |  |  |  | 1026.49 | 99.47% |  |
| 5 |  |  |  |  | 3180.51 | 98.37% |  |
| 6 |  |  |  |  | 393.42 | 99.80% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 3016.05 | 98.45% |  |
| 9 |  |  |  |  | 1315.55 | 7.98% |  |
| 10 |  |  |  |  | 1805.31 | 99.07% |  |
| 11 |  |  |  |  | 1026.12 | 9.71% |  |
| 12 |  |  |  |  | 854.28 | 9.67% |  |
| 13 |  |  |  |  | 914.4 | 7.88% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 12982.09 | 10.29% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60198.29 | 0.00% |  |
| 19 |  |  |  |  | 2969.17 | 98.48% |  |
| 20 |  |  |  |  | 1381.15 | 99.29% |  |

### Skenario Uji Coba 7

Tabel . *Confusion matrix* uji coba 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 1901.43 | 99.03% |  |
| 2 |  |  |  |  | 6508.29 | 96.66% |  |
| 3 |  |  |  |  | 4376.01 | 97.76% |  |
| 4 |  |  |  |  | 1279.89 | 99.34% |  |
| 5 |  |  |  |  | 5913.59 | 96.97% |  |
| 6 |  |  |  |  | 294.86 | 99.85% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 1608.86 | 99.18% |  |
| 9 |  |  |  |  | 419.71 | 8.41% |  |
| 10 |  |  |  |  | 473.28 | 99.76% |  |
| 11 |  |  |  |  | 535.12 | 10.15% |  |
| 12 |  |  |  |  | 349.19 | 10.09% |  |
| 13 |  |  |  |  | 328.89 | 8.34% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 15015.07 | 11.48% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60182.84 | 0.12% |  |
| 19 |  |  |  |  | 1373.7 | 99.30% |  |
| 20 |  |  |  |  | 494.12 | 99.75% |  |

### Skenario Uji Coba 8

Tabel . *Confusion matrix* uji coba 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 3559.94 | 98.18% |  |
| 2 |  |  |  |  | 7344.95 | 96.23% |  |
| 3 |  |  |  |  | 6062.74 | 96.89% |  |
| 4 |  |  |  |  | 3322.87 | 98.30% |  |
| 5 |  |  |  |  | 23396.21 | 88.01% |  |
| 6 |  |  |  |  | 3082.62 | 98.42% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 17771.84 | 90.89% |  |
| 9 |  |  |  |  | 3689.07 | 7.82% |  |
| 10 |  |  |  |  | 5947.26 | 96.95% |  |
| 11 |  |  |  |  | 2006.86 | 9.78% |  |
| 12 |  |  |  |  | 901.91 | 9.65% |  |
| 13 |  |  |  |  | 1430.06 | 7.88% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 21958.04 | 10.60% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 17182.2 | 0.07% |  |
| 18 |  |  |  |  | 59531.55 | 0.70% |  |
| 19 |  |  |  |  | 3933.59 | 97.98% |  |
| 20 |  |  |  |  | 2094.79 | 98.93% |  |

### Skenario Uji Coba 9

Tabel . *Confusion matrix* uji coba 9

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 3211.07 | 98.35% |  |
| 2 |  |  |  |  | 7046.38 | 96.39% |  |
| 3 |  |  |  |  | 4928.82 | 97.47% |  |
| 4 |  |  |  |  | 1720.33 | 99.12% |  |
| 5 |  |  |  |  | 10622.63 | 94.55% |  |
| 6 |  |  |  |  | 323.46 | 99.83% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 8170.79 | 95.81% |  |
| 9 |  |  |  |  | 2711.13 | 7.99% |  |
| 10 |  |  |  |  | 4266.97 | 97.81% |  |
| 11 |  |  |  |  | 964.84 | 9.75% |  |
| 12 |  |  |  |  | 817.09 | 9.76% |  |
| 13 |  |  |  |  | 914.4 | 7.88% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 19910.93 | 10.51% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60198.29 | 0.00% |  |
| 19 |  |  |  |  | 2969.17 | 98.48% |  |
| 20 |  |  |  |  | 1358.82 | 99.30% |  |

### Skenario Uji Coba 10

Tabel . *Confusion matrix* uji coba 10

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 1947.1 | 99.00% |  |
| 2 |  |  |  |  | 6517.21 | 96.66% |  |
| 3 |  |  |  |  | 4866.16 | 97.51% |  |
| 4 |  |  |  |  | 1134.99 | 99.42% |  |
| 5 |  |  |  |  | 24712.89 | 87.33% |  |
| 6 |  |  |  |  | 3950.07 | 97.98% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 1985.4 | 98.98% |  |
| 9 |  |  |  |  | 409.62 | 8.40% |  |
| 10 |  |  |  |  | 3783.27 | 98.06% |  |
| 11 |  |  |  |  | 831.77 | 10.12% |  |
| 12 |  |  |  |  | 1295.12 | 10.06% |  |
| 13 |  |  |  |  | 746.52 | 8.34% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 16704.75 | 10.73% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 15958.03 | 30.47% |  |
| 19 |  |  |  |  | 3781.78 | 98.06% |  |
| 20 |  |  |  |  | 16837.93 | 91.37% |  |

### Skenario Uji Coba 11

Tabel . *Confusion matrix* uji coba 11

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 9256.93 | 95.25% |  |
| 2 |  |  |  |  | 9838.2 | 94.96% |  |
| 3 |  |  |  |  | 6943 | 96.44% |  |
| 4 |  |  |  |  | 13098.75 | 93.29% |  |
| 5 |  |  |  |  | 65557.84 | 66.39% |  |
| 6 |  |  |  |  | 24976.72 | 87.20% |  |
| 7 |  |  |  |  | 23041.03 | 0.00% |  |
| 8 |  |  |  |  | 24481.86 | 87.45% |  |
| 9 |  |  |  |  | 10885.41 | 7.76% |  |
| 10 |  |  |  |  | 16070.71 | 91.76% |  |
| 11 |  |  |  |  | 8543.29 | 9.67% |  |
| 12 |  |  |  |  | 8655.17 | 9.50% |  |
| 13 |  |  |  |  | 9688.32 | 7.82% |  |
| 14 |  |  |  |  | 16970.55 | 0.00% |  |
| 15 |  |  |  |  | 24394.99 | 9.95% |  |
| 16 |  |  |  |  | 61875.48 | 0.04% |  |
| 17 |  |  |  |  | 22356.68 | 0.35% |  |
| 18 |  |  |  |  | 40029.81 | 30.15% |  |
| 19 |  |  |  |  | 44981.02 | 76.94% |  |
| 20 |  |  |  |  | 35522.1 | 81.79% |  |

### Skenario Uji Coba 12

Tabel . *Confusion matrix* uji coba 12

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 5134.42 | 97.37% |  |
| 2 |  |  |  |  | 7573.33 | 96.12% |  |
| 3 |  |  |  |  | 5045.04 | 97.41% |  |
| 4 |  |  |  |  | 2988.8 | 98.47% |  |
| 5 |  |  |  |  | 30583.2 | 84.32% |  |
| 6 |  |  |  |  | 9591.59 | 95.08% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 5665.97 | 97.10% |  |
| 9 |  |  |  |  | 1773.65 | 7.73% |  |
| 10 |  |  |  |  | 6631.88 | 96.60% |  |
| 11 |  |  |  |  | 2905.43 | 9.65% |  |
| 12 |  |  |  |  | 3698.22 | 9.39% |  |
| 13 |  |  |  |  | 1051.32 | 7.81% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 19709.61 | 10.06% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 13175.67 | 0.01% |  |
| 18 |  |  |  |  | 23228.99 | 29.96% |  |
| 19 |  |  |  |  | 3070.37 | 98.43% |  |
| 20 |  |  |  |  | 13958.8 | 92.84% |  |

### Skenario Uji Coba 13

Tabel . *Confusion matrix* uji coba 13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 1631.1 | 99.16% |  |
| 2 |  |  |  |  | 6508.29 | 96.66% |  |
| 3 |  |  |  |  | 4364.89 | 97.76% |  |
| 4 |  |  |  |  | 1147.84 | 99.41% |  |
| 5 |  |  |  |  | 3527.83 | 98.19% |  |
| 6 |  |  |  |  | 278.57 | 99.86% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 2331.43 | 98.80% |  |
| 9 |  |  |  |  | 419.71 | 8.41% |  |
| 10 |  |  |  |  | 386.96 | 99.80% |  |
| 11 |  |  |  |  | 533.67 | 10.15% |  |
| 12 |  |  |  |  | 349.19 | 10.09% |  |
| 13 |  |  |  |  | 328.89 | 8.34% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 11784.86 | 11.64% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 12739.56 | 0.00% |  |
| 18 |  |  |  |  | 60230.46 | 0.10% |  |
| 19 |  |  |  |  | 1095.39 | 99.44% |  |
| 20 |  |  |  |  | 398.87 | 99.80% |  |

### Skenario Uji Coba 14

Tabel . *Confusion matrix* uji coba 14

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 3790.08 | 98.06% |  |
| 2 |  |  |  |  | 7399.53 | 96.21% |  |
| 3 |  |  |  |  | 6056.68 | 96.90% |  |
| 4 |  |  |  |  | 3011.64 | 98.46% |  |
| 5 |  |  |  |  | 14796.2 | 92.42% |  |
| 6 |  |  |  |  | 2569.13 | 98.68% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 14560.82 | 92.54% |  |
| 9 |  |  |  |  | 3652.52 | 7.82% |  |
| 10 |  |  |  |  | 5277.53 | 97.29% |  |
| 11 |  |  |  |  | 1019.84 | 9.72% |  |
| 12 |  |  |  |  | 934.65 | 9.63% |  |
| 13 |  |  |  |  | 1359.75 | 7.88% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 19568.33 | 10.86% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 16505.9 | 0.03% |  |
| 18 |  |  |  |  | 58145.87 | 1.36% |  |
| 19 |  |  |  |  | 4197.76 | 97.85% |  |
| 20 |  |  |  |  | 2062.79 | 98.94% |  |

### Skenario Uji Coba 15

Tabel . *Confusion matrix* uji coba 15

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Citra -Ke | TP | FP | TN | FN | MSE | Similarity | Akurasi |
| 1 |  |  |  |  | 2955.35 | 98.49% |  |
| 2 |  |  |  |  | 7175.61 | 96.32% |  |
| 3 |  |  |  |  | 4949.03 | 97.46% |  |
| 4 |  |  |  |  | 1233.5 | 99.37% |  |
| 5 |  |  |  |  | 6110.39 | 96.87% |  |
| 6 |  |  |  |  | 393.42 | 99.80% |  |
| 7 |  |  |  |  | 22511.45 | 0.00% |  |
| 8 |  |  |  |  | 7838.9 | 95.98% |  |
| 9 |  |  |  |  | 1969.53 | 7.98% |  |
| 10 |  |  |  |  | 3811.55 | 98.05% |  |
| 11 |  |  |  |  | 1018.52 | 9.72% |  |
| 12 |  |  |  |  | 824.61 | 9.73% |  |
| 13 |  |  |  |  | 914.4 | 7.88% |  |
| 14 |  |  |  |  | 15493.88 | 0.00% |  |
| 15 |  |  |  |  | 15973.67 | 10.88% |  |
| 16 |  |  |  |  | 56471.81 | 0.00% |  |
| 17 |  |  |  |  | 13489.62 | 0.00% |  |
| 18 |  |  |  |  | 60198.29 | 0.00% |  |
| 19 |  |  |  |  | 1509.14 | 99.23% |  |
| 20 |  |  |  |  | 1373.7 | 99.30% |  |

## Evaluasi Umum Skenario Uji Coba

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

# BAB VI KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

## Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba pendeteksian *copy-move* citra menggunakan metode *expanding block algorithm* adalah sebagai berikut:

## Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

# DAFTAR PUSTAKA

[1] G. Lynch, F. Y. Shih, and H.-Y. M. Liao, “An efficient expanding block algorithm for image copy-move forgery detection,” *Inf. Sci.*, vol. 239, pp. 253–265, Aug. 2013.

[2] F. J, S. D, and L. J, “Detection of copy-move forgery in digital images,” *Proc Digit. Forensic Res. Workshop*, Aug. 2003.

[3] C. S. Lin, C. C. Chen, and Y. C. Chang, “An Efficiency Enhanced Cluster Expanding Block Algorithm for Copy-Move Forgery Detection,” in *2015 International Conference on Intelligent Networking and Collaborative Systems*, 2015, pp. 228–231.

[4] A. Langille and M. Gong, “An Efficient Match-based Duplication Detection Algorithm,” in *The 3rd Canadian Conference on Computer and Robot Vision (CRV’06)*, 2006, pp. 64–64.

[5] J. A. Redi, W. Taktak, and J.-L. Dugelay, “Digital image forensics: a booklet for beginners,” *Multimed. Tools Appl.*, vol. 51, no. 1, pp. 133–162, Jan. 2011.

[6] “OpenCV: Morphological Transformations.” [Online]. Available: http://docs.opencv.org/trunk/d9/d61/tutorial\_py\_morphological\_ops.html. [Accessed: 14-Dec-2016].

[7] “Numpy and Scipy Documentation — Numpy and Scipy documentation.” [Online]. Available: https://docs.scipy.org/doc/. [Accessed: 15-Dec-2016].

[8] “PyQt/Tutorials - Python Wiki.” [Online]. Available: https://wiki.python.org/moin/PyQt/Tutorials. [Accessed: 15-Dec-2016].

[9] “4.3. Preprocessing data — scikit-learn 0.18.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing. [Accessed: 15-Dec-2016].

# LAMPIRAN

## Kumpulan Kode Sumber

|  |
| --- |
| **from** PyQt5.QtGui **import** \* **from** PyQt5.QtCore **import** \* **from** PyQt5.QtWidgets **import** \* **import** json,subprocess  **class** CopyMoveGUI(QMainWindow):  **def** \_\_init\_\_(self):  super(CopyMoveGUI, self).\_\_init\_\_()  self.initUI()  self.centerWindow()   **def** initUI(self):  self.setWindowTitle(**"Copy Move Detection"**)  self.setFixedSize(800, 600)  self.createAction()  self.createMenuBar()   self.centralWidget = QWidget()   *#group input* self.groupInput = QGroupBox(self.centralWidget)  self.groupInput.setGeometry(QRect(10, 0, 401, 541))  self.groupInput.setTitle(**"Input"**)   self.inputGambar = QLabel(self.groupInput)  self.inputGambar.setGeometry(QRect(20, 30, 361, 201))  self.inputGambar.setText(**"No Image"**)   self.pilihInputGambar = QPushButton(self.groupInput)  self.pilihInputGambar.setGeometry(QRect(300, 240, 80, 25))  self.pilihInputGambar.setText(**"Pilih Gambar"**)  self.pilihInputGambar.clicked.connect(self.openImage)   self.line = QFrame(self.groupInput)  self.line.setGeometry(QRect(0, 270, 401, 16))  self.line.setFrameShape(QFrame.HLine)  self.line.setFrameShadow(QFrame.Sunken)   self.inputGambarKunci = QLabel(self.groupInput)  self.inputGambarKunci.setGeometry(QRect(20, 290, 361, 201))  self.inputGambarKunci.setText(**"No Image"**)   self.pilihInputGambarKunci = QPushButton(self.groupInput)  self.pilihInputGambarKunci.setGeometry(QRect(259, 500, 121, 25))  self.pilihInputGambarKunci.setText(**"Pilih Gambar Kunci"**)  self.pilihInputGambarKunci.clicked.connect(self.openKeyImage)   *#group threshold config* self.groupThreshold = QGroupBox(self.centralWidget)  self.groupThreshold.setGeometry(430, 20, 351, 171)  self.groupThreshold.setTitle(**"Threshold Config"**)   self.vlWidThresold = QWidget(self.groupThreshold)  self.vlWidThresold.setGeometry(QRect(10, 30, 331, 131))  self.vltThreshold = QVBoxLayout(self.vlWidThresold)  self.vltThreshold.setContentsMargins(11,11,11,11)  self.vltThreshold.setSpacing(6)   self.pval = QLabel(self.vlWidThresold)  self.pval.setText(**"pVal : "**)  self.vltThreshold.addWidget(self.pval)  self.blocksize = QLabel(self.vlWidThresold)  self.blocksize.setText(**"Block Size : "**)  self.vltThreshold.addWidget(self.blocksize)  self.blockdistance = QLabel(self.vlWidThresold)  self.blockdistance.setText(**"Block Distance : "**)  self.vltThreshold.addWidget(self.blockdistance)  self.totalbucket = QLabel(self.vlWidThresold)  self.totalbucket.setText(**"Total Bucket : "**)  self.vltThreshold.addWidget(self.totalbucket)  self.minarea = QLabel(self.vlWidThresold)  self.minarea.setText(**"Min Area : "**)  self.vltThreshold.addWidget(self.minarea)   self.pilihInputThreshold = QPushButton(self.groupThreshold)  self.pilihInputThreshold.setGeometry(QRect(250, 130, 80, 25))  self.pilihInputThreshold.setText(**"Pilih Config"**)  self.pilihInputThreshold.clicked.connect(self.openThrehsold)   *#group option* self.groupOption = QGroupBox(self.centralWidget)  self.groupOption.setGeometry(QRect(430, 210, 351, 101))  self.groupOption.setTitle(**"Option"**)   self.filter = QLabel(self.groupOption)  self.filter.setGeometry(QRect(10, 30, 54, 17))  self.filter.setText(**"Filter : "**)   self.filterCombo = QComboBox(self.groupOption)  self.filterCombo.setGeometry(QRect(10, 60, 331, 25))  listFilter = [**"None"**,**"Averaging"**,**"Median"**,**"Bilateral"**]  self.filterCombo.addItems(listFilter)  self.filterCombo.currentIndexChanged.connect(self.getFilter)   *#start and result folder* self.buttonStart = QPushButton(self.centralWidget)  self.buttonStart.setGeometry(QRect(690, 320, 91, 25))  self.buttonStart.setText(**"Mulai Deteksi"**)  self.buttonStart.clicked.connect(self.startDetection)   self.buttonResult = QPushButton(self.centralWidget)  self.buttonResult.setGeometry(QRect(550, 320, 121, 25))  self.buttonResult.setText(**"Lihat Folder Hasil"**)  self.buttonResult.clicked.connect(self.openResultDir)   *#group log* self.groupLog = QGroupBox(self.centralWidget)  self.groupLog.setGeometry(QRect(430, 350, 361, 201))  self.groupLog.setTitle(**"Log"**)   self.logOutput = QTextEdit(self.groupLog)  self.logOutput.setGeometry(QRect(0, 20, 361, 181))   self.setCentralWidget(self.centralWidget)   *#Process Detection* self.process = QProcess(self)  self.process.readyReadStandardOutput.connect(self.stdoutReady)  self.process.readyReadStandardError.connect(self.stderrReady)  self.process.started.connect(**lambda**: self.buttonStart.setEnabled(False))  self.process.finished.connect(**lambda**: self.buttonStart.setEnabled(True))   **def** getFilter(self,i):  **print** i  self.filterChoice = str(i)   **def** append(self, text):  cursor = self.logOutput.textCursor()  cursor.movePosition(cursor.End)  cursor.insertText(text)  *#self.logOutput.ensureCursorVisible()* **def** stdoutReady(self):  text = str(self.process.readAllStandardOutput())  **print** text.strip()  self.append(text)   **def** stderrReady(self):  text = str(self.process.readAllStandardError())  **print** text.strip()  self.append(text)   **def** startDetection(self):  self.argv = [**"/media/sudirahanif/Kuliah/TA/Program/TugasAkhir.py"**,self.imageFile,self.thresholdFile,self.filterChoice]  self.process.start(**'/home/sudirahanif/anaconda2/bin/python'**,self.argv)    **def** openResultDir(self):  *#subprocess.Popen(["xdg-open", "/media/sudirahanif/Kuliah/TA/Hasil"])* subprocess.Popen([**"explorer"**, **"."**])   **def** openThrehsold(self):  fileName, \_ = QFileDialog.getOpenFileName(self, **"Open File"**, QDir.currentPath(), **"JSON (\*.json)"**)  self.thresholdFile = fileName  **if** fileName:  **with** open(fileName, **'rb'**) **as** file:  data = json.load(file)  minArea = data[**'threshold'**][**'minArea'**]  blockSize = data[**'threshold'**][**'blockSize'**]  numBuckets = data[**'threshold'**][**'numBuckets'**]  pVal = data[**'threshold'**][**'pVal'**]  blockDistance = data[**'threshold'**][**'blockDistance'**]  self.pval.setText(**"pVal : "**+str(pVal))  self.blocksize.setText(**"Block Size : "**+str(blockSize))  self.blockdistance.setText(**"Block Distance : "**+str(blockDistance))  self.totalbucket.setText(**"Total Bucket : "**+str(numBuckets))  self.minarea.setText(**"Min Area : "**+str(minArea))   **def** openImage(self):  fileName, \_ = QFileDialog.getOpenFileName(self, **"Open File"**, QDir.currentPath(), **"Images (\*.png \*.jpg)"**)  self.imageFile = fileName  **if** fileName:  image = QImage(fileName)  **if** image.isNull():  QMessageBox.information(self, **"Image Viewer"**, **"Cannot load %s."** % fileName)  **return  else**:  self.inputGambar.setPixmap(QPixmap.fromImage(image))  self.inputGambar.setScaledContents(True)   **def** openKeyImage(self):  fileName, \_ = QFileDialog.getOpenFileName(self, **"Open File"**, QDir.currentPath(), **"Images (\*.png \*.jpg)"**)  self.imageFileKey = fileName  **if** fileName:  image = QImage(fileName)  **if** image.isNull():  QMessageBox.information(self, **"Image Viewer"**, **"Cannot load %s."** % fileName)  **return  else**:  self.inputGambarKunci.setPixmap(QPixmap.fromImage(image))  self.inputGambarKunci.setScaledContents(True)    **def** centerWindow(self):  frameGm = self.frameGeometry()  screen = QApplication.desktop().screenNumber(QApplication.desktop().cursor().pos())  centerPoint = QApplication.desktop().screenGeometry(screen).center()  frameGm.moveCenter(centerPoint)  self.move(frameGm.topLeft())   **def** createMenuBar(self):  self.fileMenu = QMenu(**"&File"**, self)  self.fileMenu.addAction(self.exitAct)   self.helpMenu = QMenu(**'&Help'**, self)  self.helpMenu.addAction(self.aboutAct)  self.helpMenu.addAction(self.aboutActQt)   self.menuBar().addMenu(self.fileMenu)  self.menuBar().addMenu(self.helpMenu)   **def** createAction(self):  self.exitAct = QAction(**"&Keluar"**, self, shortcut=**"Ctrl+Q"**, triggered=self.close)  self.aboutAct = QAction(**"Tentang &Penulis"**, self, triggered=self.about)  self.aboutActQt = QAction(**"Tentang &Qt"**, self,triggered=QApplication.instance().aboutQt)   **def** about(self):  QMessageBox.about(self, **"Tentang Tugas Akhir"**,**"<p>Hanif Sudira - 5113100184</p>""<p>Teknik Informatika ITS 2013</p>"**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  **import** sys  app = QApplication(sys.argv)  CopyMove = CopyMoveGUI()  CopyMove.show()  sys.exit(app.exec\_()) |

Kode Sumber . Antarmuka Aplikasi

# BIODATA PENULIS

******Hanif Sudira merupakan anak dari pasangan Bapak Syamsul Bahar dan Ibu Roslinda. Lahir di Padang pada tanggal 22 April 1995. Penulis menempuh pendidikan formal dimulai dari TK Aisyah (2000-2001), SDN 29 Ganting Utara Padang Timura (2001-2007), SMPN 12 Padang (2007-2010), SMAN 10 Padanga (2010-2013) dan S1 Teknik Informatika ITS (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (2014-2016) dan KMI (2014-2015). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2014 divisi NPC dan SCHEMATICS 2014 NPC. Penulis juga pernah kerja praktik di Tokopedia periode Juli – Agustus 2016.. Penulis dapat dihubungi melalui email: [sudirahanif@gmail.com](mailto:sudirahanif@gmail.com).