TUGAS AKHIR - KI141502

FORENSIK DIGITAL DETEKSI COPY-MOVE PADA CITRA MENGGUNAKAN MODIFIKASI EXPANDING BLOCK ALGORITHM

HANIF SUDIRA
NRP 5113100184

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

Dosen Pembimbing II
Hudan Studiawan, S.Kom., M.Kom.

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

TUGAS AKHIR – KI141502

FORENSIK DIGITAL DETEKSI COPY-MOVE PADA CITRA MENGGUNAKAN MODIFIKASI EXPANDING BLOCK ALGORITHM

HANIF SUDIRA
NRP 5113100184

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

Dosen Pembimbing II
Hudan Studiawan, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI141502

DIGITAL FORENSICS DETECTION OF IMAGE COPY-MOVE USING MODIFIED EXPANDING BLOCK ALGORITHM

HANIF SUDIRA
NRP 5112100184

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

Second Advisor
Hudan Studiawan, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

FORENSIK DIGITAL DETEKSI COPY-MOVE PADA CITRA MENGGUNAKAN MODIFIKASI EXPANDING BLOCK ALGORITHM

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:
HANIF SUDIRA
NRP: 5113100184

Disetujui oleh Pembimbing Tugas Akhir:

1. Tohari Ahmad, S.Kom., MIT, Ph.D. ..... (Pembimbing 1)
   (NIP. 197505252003121003)
2. Hudan Studiawan, S.Kom., M.Kom. ..... (Pembimbing 2)
   (NIP. 198705112012121003)

SURABAYA
JANUARI, 2017

[Halaman ini sengaja dikosongkan]

FORENSIK DIGITAL DETEKSI COPY-MOVE PADA CITRA MENGGUNAKAN MODIFIKASI EXPANDING BLOCK ALGORITHM

Nama Mahasiswa : HANIF SUDIRA
NRP : 5113100184
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Tohari Ahmad, S.Kom., MIT,. Ph.D.
Dosen Pembimbing 2 : Hudan Studiawan, S.Kom., M.Kom.

Abstrak

Copy-move adalah suatu bentuk tindakan kriminal pada perangkat digital dengan cara pemalsuan citra, dimana pengguna menyalin suatu bagian citra lalu menempelkannya di bagian lain pada citra yang sama. Biasanya teknik copy-move digunakan untuk menutupi objek dari suatu citra dengan cara menempelkan objek yang serupa di sekitarnya dengan maksud dan tujuan tertentu yang dapat merugikan. Oleh karena itu diperlukan metode yang dapat mendeteksi serangan copy-move secara cepat dan efisien, terlebih citra sering menjadi barang bukti dalam pernyelidikan tindakan kriminal.

Tugas Akhir ini mengimplementasikan dua cara untuk mencari dominant feature pada citra yaitu image gray dominant feature dan image color dominant feature. Tugas Akhir ini menggunakan metode Averaging Filter dan Median Filter sebagai penghilang noise pada citra yang akan dideteksi menggunakan metode expanding block algorithm. Tugas Akhir ini juga menggunakan data masukan citra yang diganggu oleh noise uniform distribution sebesar 0.5%.

Hasil uji coba parameter yang dilakukan didapatkan bahwa nilai parameter terbaik yang digunakan pada Tugas Akhir ini adalah numBuckets 12000, *pVal* 0.50, *blockSize* 16 dan *minArea* 90. *Untuk hasil uji coba performa didapatkan bahwa nilai MSE terendah 1223.25, nilai similarity terbesar 99.27% dan nilai*

akurasi terbesar 93.06%. Penggunaan filter pada metode expanding block algorithm akan menurunkan performa serta metode expanding block algorithm dapat digunakan untuk mendeteksi citra gangguan noise uniform distribution 0.5% dengan menggunakan image color dominant feature.

Kata kunci: Copy-move, filter, paramater, dominant feature, expanding block algorithm.

DIGITAL FORENSICS DETECTION OF IMAGE COPY-MOVE USING MODIFIED EXPANDING BLOCK ALGORITHM

Student's Name	**: HANIF SUDIRA**
Student's ID	**: 5113100184**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Tohari Ahmad, S.Kom., MIT,. Ph.D.**
Second Advisor	**: Hudan Studiawan, S.Kom., M.Kom.**

Abstract

Copy-move forgery is a form of criminal acts digital device specific of image tampering where the part of the image is copied and pasted in another part of the same image. This technique is usually used to cover up an object in the image by attaching similar object around it. Therefore developing a method to verify and detect manipulated image became very important, especially because of the image as evidence in the investigation of criminal acts.

This undergraduate thesis implements two ways to search for a dominant feature in the image that is image gray dominant feature and image color dominant feature. This undergraduate thesis project using Averaging Filter and Median Filter as relieving noise at the image to be detected using the method of expanding the block algorithm. This undergraduate thesis also uses input data image plagued by noise uniform distribution of 0.5%

The trial results showed that the best parameters values used in this undergraduate thesis are numBuckets 12000, pVal 0.50, blocksize 16 and minArea 90. For the test results showed that the performance of the lowest MSE value 1223.25, the largest similarity value of 99.27% and value the accuracy of 93.06%. The use of filters on expanding block algorithm method will decrease performance. expanding block algorithm can be used to detect

image noise with a uniform distribution of 0.5% by using image color dominant feature.

Keywords: Copy-move, filter, parameters, dominant feature, expanding block algorithm.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "**FORENSIK DIGITAL DETEKSI COPY-MOVE MENGGUNAKAN MODIFIKASI EXPANDING BLOCK ALGORITHM**". 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 penulisa 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. Orang tua penulis Bapak Syamsul Bahar dan Ibu Roslinda serta keluarga penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Tohari Ahmad, S.Kom., MIT,. Ph.D. dan Bapak Hudan Studiawan, S.Kom., M.Kom selaku dosen pembimbing penulis yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Royyana M Ijtihadie, S.Kom., M.Kom., Ph.D. selaku dosen wali penulis yang telah memberikan arahan, masukan dan motivasi kepada penulis.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.

6. Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator Tugas Akhir.
7. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
8. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
9. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
10. Teman-teman administrator NCC/KBJ, yang telah menemani dan menyemangati penulis selama penulis menjadi administrator, menjadi rumah kedua penulis selama penulis berkuliah.
11. Teman-teman angkatan 2013, yang sudah mendukung penulis selama perkuliahan.
12. 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, Januari 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi perangkat digital pengolahan gambar, video dan kemampuan komputasi suatu komputer yang sangat pesat membuat media digital menjadi lebih mudah untuk di modifikasi termasuk manipulasi citra. Ditambah dengan adanya perangkat lunak pengolahan gambar digital yang dapat mempermudah sesorang untuk mengolah dan memanipulasi informasi yang didapat sesuai dengan kebutuhan yang diinginkan salah satunya *copy-move*.

Copy-move atau duplikasi sebagian daerah pada sebuah citra merupakan salah satu bentuk pemalsuan citra dari banyak pemalsuan citra. *Copy-move* adalah suatu cara dimana pengguna menyalin suatu bagian dari sebuah citra dan meletakan bagian salinan citra tersebut ke bagian lain pada citra yang sama guna untuk menutupi objek yang tidak ingin ditampilkan [1, 2, 3].

Dalam beberapa tahun terakhir sudah terdapat banyak metode penelitian terkait deteksi pemalsuan citra untuk berbagai kasus diantaranya *detecting duplicated image regions* [2], *color filter interpolation* [4]. Terdapat juga cara untuk mendeteksi *copy-move* langsung pada seluruh citra [5].

Pada penelitian kali ini akan dilakukan cara untuk medeteksi serangan *copy-move* dengan menggunakan metode modifikasi *expanding block algorithm* [1]. Dengan melakukan implementasi modifikasi *expanding block algorithm* maka diharapkan bisa menghasilkan sebuah algoritma untuk melakukan analisis deteksi *copy-move* pada citra yang lebih cepat dan efisien.

Dikemudian hari serangan *copy-move* pada citra dapat dideteksi dengan cepat, sehingga pemalsuan pada citra dapat diatasi untuk menghindari kejadian yang tidak diinginkan.

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana melakukan deteksi terhadap serangan *copy-move*?
2. Bagaimana melakukan implementasi metode *expanding block algorithm* untuk mendeteksi serangan *copy-move*?
3. Bagaimana melakukan modifikasi terhadap metode *expanding block algorithm*?

1.3 Batasan Permasalahan

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

1. Bahasa pemrograman yang digunakan untuk menganalisa serangan *copy-move* ini adalah Python dengan distribusi *package* Anaconda.
2. Dataset yang digunakan adalah citra digital yang memiliki kualitas rendah.
3. Tidak dapat mendeteksi *copy-move* pada duplikat yang telah di *resize* dan *rotate*.
4. Keluaran dari implementasi adalah *file* citra daerah *copy-move* yang ditandai dengan warna.
5. Metode yang digunakan untuk mendeteksi *copy-move* adalah *expanding block algorithm*.
6. Perangkat lunak yang digunakan adalah PyCharm Professional 2016.2.3 sebagai IDE.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Melakukan deteksi pemalsuan citra terhadap serangan *copy-move*.
2. Melakukan implementasi modifikasi metode *expanding block algorithm*.

3. Membuat aplikasi yang mudah diakses dan bisa melakukan deteksi *copy-move* sesuai dengan metode yang diusulkan.

1.5 Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Memberikan manfaat di bidang *image processing* terutama pada bidang forensik digital untuk dapat mendeteksi *copy-move* pada citra.
2. Menerapkan ilmu yang dipelajari selama kuliah di Teknik Informatika ITS agar dapat berguna bagi masyarakat.

1.6 Metodologi

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

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal Tugas Akhir ini adalah menyusun proposal Tugas Akhir. Pada proposal, berisi tentang deskripsi pendahuluan dari Tugas Akhir yang akan dibuat. Prosposal juga berisi tentang garis besar yang akan dikerjakan sehingga memberikan gambaran untuk dapat mengerjakan Tugas Akhir sesuai dengan *timeline* yang dibuat. Gagasan untuk menganalisis dan mengidentifikasi pemalsuan citra terhadap serangan *copy-move* menggunakan metode *expanding block algorithm* [1].

1.6.2 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-metode statistika seperti mean dan variance serta 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].

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, implementasi dilakukan dengan menggunakan suatu perangkat lunak yaitu Python versi 2.7 sebagai bahasa pemrograman utama dengan distribusi *package* Anaconda [7] dan PyCharm Professional 2016.2.3 sebagai IDE [8].

1.6.4 Pengujian dan Evaluasi

Pada tahap ini algoritma yang telah diimplementasikan akan diuji coba dengan menggunakan data uji coba yang ada. Uji coba dilakukan dengan menggunakan suatu perangkat lunak dengan tujuan mengetahui kemampuan metode yang digunakan dan mengevaluasi hasil Tugas Akhir dengan jurnal pendukung yang digunakan. Hasil evaluasi juga mencakup kompleksitas, parameter dan performa dari metode yang telah diimplementasikan dalam mendeteksi serangan *copy-move* pada citra.

1.6.5 Penyusunan Buku

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

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang forensik digital, forensik citra digital, *copy-move*, *Variance*, *Mean Squared Error* dan Qt.
3. Bab III. Perancangan Perangkat Lunak
   Bab ini berisi pembahasan mengenai perancangan dari metode *expanding block algorithm* yang digunakan untuk mendeteksi pemalsuan citra pada serangan *copy-move* dan perancangan *GUI* atau antarmuka menggunakan Qt.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk *Pseudocode* yang berupa *Pseudocode* dari metode *expanding block algorithm*.
5. Bab V. Pengujian dan Evaluasi
   Bab ini berisikan hasil uji coba dari metode *expanding block algorithm* yang digunakan untuk mendeteksi pemalsuan citra pada serangan *copy-move* yang sudah diimplementasikan. Uji coba dilakukan dengan menggunakan dataset citra yang memiliki kualitas rendah. Hasil evaluasi mencakup perbandingan kompleksitas, parameter dan performa dari masing-masing data masukan.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya adalah *expanding block 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 perangkat lunak.

2.1 Citra Digital

Citra digital adalah gambar dua dimensi yang bisa ditampilkan pada perangkat digital. Citra digital dihasilkan dari gambar *analog* dua dimensi yang *continue* menjadi gambar diskrit melalui proses *sampling*. Gambar *analog* dibagi menjadi *N* baris dan *M* kolom sehingga menjadi gambar diskrit. Citra digital juga bisa disebut suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra yang disebut sebagai *pixel*. *Pixel* menyatakan tingkat keabuan pada titik tersebut disuatu citra [9].

2.2 Forensik Digital

Forensik digital adalah salah satu cabang ilmu forensik yang berakitan dengan bukti legal yang ditemui pada komputer dan media penyimpanan digital, atau bisa diartikan sebagai cabang ilmu dari forensik yang membutuhkan metode ilmiah atau teknik khusus untuk mengenali, mengklasifikasikan, mengumpulkan, mengevalusi, menganalisis dokumen dan presentasi bukti digital dari perangkat elektronik untuk membuktikan sebuah tindak kejahatan dan dapat dipertanggungjawabkan. Dalam forensik digital terdapat bukti fisik yaitu komputer, harddisk, perangkat penyimpanan komputer dan lain-lain, sedangkan bukti berupa perangkat lunak atau dokumen-dokumen yang tersimpan dalam

komputer seperti gambar, pdf, video, lalu lintas data dalam jaringan dan lain sebagainya. Karena cakupan forensik digital yang luas forensik digital dapat dibagi menjadi beberapa bagian kecil seperti forensik komputer, forensik media penyimpanan komputer, forensik lalu-lintas data komputer, forensik jaringan komputer, forensik digital dan lain sebagainya [2].

2.3 Forensik Citra Digital

Forensik citra digital merupakan bagian dari forensik digital yang bertujuan untuk memvalidasi keaslian suatu gambar atau citra dengan cara mengetahui informasi riwayat gambar aslinya [5]. Terdapat 2 cara untuk memvalidasi sebuah citra yaitu metode aktif dan metode pasif. Contoh metode aktif adalah *watermarking* sedangkan metode pasif adalah kasus dimana informasi di dalam citra sulit untuk dipecahkan, contahnya adalah kasus *copy-move* [1]. Beberapa kasus yang banyak terjadi dalam pemalsuan citra digital adalah *Copy-Splicing* dan *Copy-Move* dimana *Copy-Splicing* melibatkan dua citra atau lebih sedangkan *Copy-Move* hanya melibatkan satu citra [2]. *Copy-Splicing* adalah memindahkan sebagian citra kepada citra lain sedangkan *Copy-Move* memindahkan sebagian citra pada bagian lainnya pada citra yang sama.

2.4 *Copy Move*

Copy-move adalah salah satu bentuk pemalsuan citra dengan cara pengguna menyalin sebagian citra lalu menempelkannya di bagian lain pada citra yang sama [1]. Teknik ini biasanya digunakan untuk menutupi objek dari sebagian citra dengan cara menempelkan objek yang serupa di sekitarnya. Umumnya objek yang disalin adalah objek yang memiliki pola tidak beraturan dengan tujuannya agar objek yang menutupi bagian lain di dalam gambar yang sama untuk tujuan tertentu dimana sulit untuk membedakan dengan mata. Sehingga dibutuhkan sebuah bantuan perangkat komputer untuk melakukan pengecekan, yaitu dengan

cara mengolah citra yang sudah terdeteksi *copy-move*. Gambar 2.1 merupakan contoh citra yang terkena serangan *copy-move*, yaitu terdapat pada logo biru yang ada di dinding gedung. Aslinya hanya terdapat satu logo biru, setelah serangan *copy-move* terjadi terdapat 4 logo biru yang sama yang ada di dinding gedung pada Gambar 2.1.

Gambar 2.1 Contoh Copy-Move Citra

2.5 Python

Python adalah salah satu bahas pemrograman tingkat tinggi yang bersifat interpreter, interaktif, *object oriented* dan dapat beroperasi di hampir semua *platform* seperti UNIX, Mac, Windows ataupun yang lain. Sebagai bahasa pemrograman tingkat tinggi Python termasuk salah satu bahasa pemrograman yang mudah dipelajari karena sintaks yang jelas . Python menyediakan *module-module* yang siap dipakai dan juga Python memiliki

struktur data tingkat tinggi yang efisien. Python menyediakan 2 versi yaitu versi 2 dan versi 3 dimana disetiap versi memiliki kekurangan dan kelebihan masing-masing [10]. Gambar 2.2 merupakan logo Python

Gambar 2.2 Logo Python

2.6 Anaconda

Anaconda adalah *open data science platform* yang dibangun menggunakan bahasa pemrograman Python. Versi Anaconda *open-source* menyediakan distribusi dengan performa tingkat tinggi dari bahasa pemrograman Python dan R yang berisikan lebih dari 100 paket untuk *data science.* Anaconda menyediakan CLI (*command line interface*) berupa perintah "conda" sebagai manajemen lingkungan untuk bahasa pemrograman Python. Perintah "conda" bisa untuk *create, export, list, remove* dan *update* lingkungan yang bisa digunakan untuk versi bahasa pemrograman Python atau *package* yang dipasang didalamnya. Anaconda dapat digunakan di berbagai *platform* seperti Windows, Linux dan Mac [7]. Gambar 2.3 merupakan logo Anaconda.

Gambar 2.3 Logo Anaconda

2.7 OpenCV

OpenCV (*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*, seperti GNU/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 [11]. Gambar 2.4 merupakan logo OpenCV.

Gambar 2.4 Logo OpenCV

2.8 Matplotlib

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 grafik. Matplotlib membuat semua jenis grafik, histogram, chart untuk mudah dibuat [12]. Gambar 2.5 merupakan logo matplotlib.

Gambar 2.5 Logo Matplotlib

2.9 NumPy

NumPy atau *numeric Python* merupakan pustaka perangkat lunak bahasa pemrograman Python bersifat *open source* yang digunakan untuk pengolahan ilmiah matematika. Selain digunakan untuk hal ilmiah, NumPy juga bisa digunakan untuk *container* multidimensi yang efisien untuk data generik [13]. Tipe data *arbitrary* dapat didefinisikan, ini memungkinkan NumPy secara lancar dan cepat mengintegrasikan dengan banyak tipe basis data. Gambar 2.6 merupakan logo NumPy.

Gambar 2.6 Logo NumPy

2.10 SciPy

SciPy merupakan singkatan dari *Scientific Python*. SciPy adalah pustaka perangkat lunak bahasa pemrograman Python bersifat *Open Source*. SciPy sering 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 [13].

Gambar 2.7 Logo SciPy

2.11 PyQt

Qt merupakan sebuah IDE (*Integrated Development Environment*) yang dibuat pada tahun 1996. Qt adalah toolkit untuk membantu pengembangan aplikasi yang memiliki tampilan grafis yang dapat berjalan di berbagai *platform*. Sedangkan PyQt merukapan pustaka perangkat lunak Qt yang dapat membantu mengembangkan aplikasi yang memiliki tampilan grafis dengan menggunakan bahasa pemrograman Python [14]. Gambar 2.8 merupakan logo Qt.

Gambar 2.8 Logo Qt

2.12 JSON

JSON meruapakan singkatan dari *JavaScript Object Notation* adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemrograman Javascript. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apaun sehinggan membuat JSON menjadi ideal sebagai bahasa pertukaran data [15]. Gambar 2.9 merupakan logo JSON.

Gambar 2.9 Logo JSON

2.13 *Variance*

Variance berhubungan erat dengan standar deviasi, yakni variabel yang digunakan untuk mengukur dan mengetahui seberapa jauh penyebaran data dalam distribusi data. Dalam artian lain, *variance* digunakan untuk mengukur variabilitas data, atau tingkat keragaman dalam data. Semakin tinggi nilai *variance*, maka semakin bervariasi dan beragam suatu data [1]. *Variance* dapat dihitung menggunakan Persamaan 2.1.

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

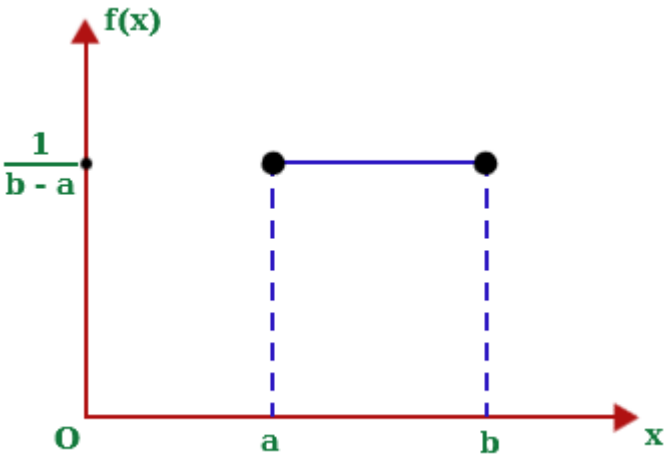
2.14 *Mean Squared Error*

Mean Squared Error adalah metode untuk mengevaluasi dua buah matriks atau *array*. Masing-masing kesalahan atau sisa dikuadratkan. Kemudian dijumlahkan dan ditambahkan dengan jumlah observasi. Pendekatan ini mengatur kesalahan peramalan yang besar karena kesalahan-kesalahan itu dikuadratkan. Metode ini menghasilkan kesalahan-kesalahan sedang yang kemungkinan lebih baik untuk kesalahan kecil, tetapi kadang menghasilkan perbedaan yang besar. *Mean Squared Error* dapat dihitung menggunakan Persamaan 2.2.

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

2.15 Distribusi *Uniform*

Distibusi *uniform* adalah probabilitas distribusi yang sederhana yang semua peubah acaknya mempunyai probabilitas yang sama [16]. Gambar 2.10 merupakan kurva distribusi *uniform*.



Gambar 2.10 Kurva Distribusi *Uniform*

2.16 *Similarity*

Similarity adalah metode untuk mengevaluasi dua buah citra dengan ukuran sama. *Similarity* bertujuan untuk mencari kemiripin dari dua buah citra yaitu dengan cara membandingkan piksel antara dua buah citra. *Similarity* dapat dihitung menggunakan Persamaan 2.3.

$$similarity = \frac{kesamaan\ piksel}{total\ piksel} \quad (2.3)$$

2.17 *Convolution*

Convolution adalah cara untuk menkombinasikan dua buah deret angka untuk menghasilkan deret angka yang ketiga. *Convolution* dapat dihitung dengan Persamaan 2.4.

$$C[k] = \sum_{n=0}^{n+k} a[k]b[k-n] \quad (2.4)$$

Pada citra *convolution* digunakan untuk *filter* atau *image processing*, yaitu proses dimana citra dimanipulasi dengan menggunakan eksternal *mask* atau *kernel* untuk menghasilkan citra baru.

2.18 *Canny Edge Detector*

Canny edge merupakan salah satu dari teknik *edge detection* yang cukup populer penggunaanya dalam pengolahan citra. *Canny Edge Detector* menggunakan algoritma *multi-stage* untuk mendeteksi berbagai bagian tepi dalam citra [17].

2.19 *Confusion Matrix*

Confusion matrix adalah suatu metode yang biasa digunakan untuk melakukan penghitungan performa suatu algoritma pada konsep data mining. *Confusion matrix* memiliki informasi hasil prediksi dan aktual pada data yang telah diklasifikasi.

Tabel 2.1 *Confusion Matrix* Dua Kelas

Prediksi / Aktual	1	0
1	TP (True Positive)	FN (False Negative)
0	FP (False Positive)	TN (True Negative)

Penjelasn *confusion matrix* pada *copy-move* adalah sebagai berikut:

1. TP adalah ketika citra ada pemalsuan *copy-move* dan program mendeteksi adanya *copy-move*.
2. FP adalah ketika citra tidak ada pemalsuan *copy-move* dan program mendeteksi adanya *copy-move.*
3. FN adalah ketika citra ada pemalsuan *copy-move* dan program tidak mendeteksi adanya *copy-move.*
4. TN adalah ketika citra tidak ada pemalsuan *copy-move* dan program tidak mendeteksi adanya *copy-move.*

Nilai performa yang bisa dihitung menggunakan *confusion matrix* antara lain: akurasi. Akurasi adalah hasil bagi dari jumlah prediksi yang terklasifikasi secara benar dibagi total data yang diklasifikasi seperti pada Persamaan 2.5.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.5)$$

[Halaman ini sengaja dikosongkan]

BAB III
PERANCANGAN PERANGKAT LUNAK

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

3.1 Data

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

3.1.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan awal dari program pendeteksian *copy-move*. Data yang digunakan dalam perangkat lunak pendeteksi serangan *copy-move* pada citra digital terdiri dari tiga jenis data masukan yaitu citra yang terkena serangan *copy-move*, kunci jawaban dari citra yang terkena serangan *copy-move* dan *threshold config* yang akan dijelaskan berikutnya.

Data citra masukan yang digunakan berjumlah 20 buah citra yang mengalami serangan *copy-move* yang didapat dari Universitas Friedich-Alexander [18] dan *CoMoFod* [19].Selain itu, dari 20 data citra yang terkena serangan *copy-move* akan diedit dengan menambahkan *noise* dengan menggunakan perangkat lunak pembantu untuk pengolahan citra atau membuat efek yaitu *Photoshop*. Dengan menggunakan fitur *add noise uniform distribution* sebesar 0.5% yang ada pada perangkat lunak *Photoshop*, data citra akan dengan mudah untuk ditambahkan *noise*. Sehingga terdapat 20 data masukan citra *copy-move* dan 20 data masukan citra *copy-move* dengan *noise*.

Adapun penjelasan data masukan dari Tugas Akhir ini adalah sebagai berikut:

1. Citra *Copy-Move*

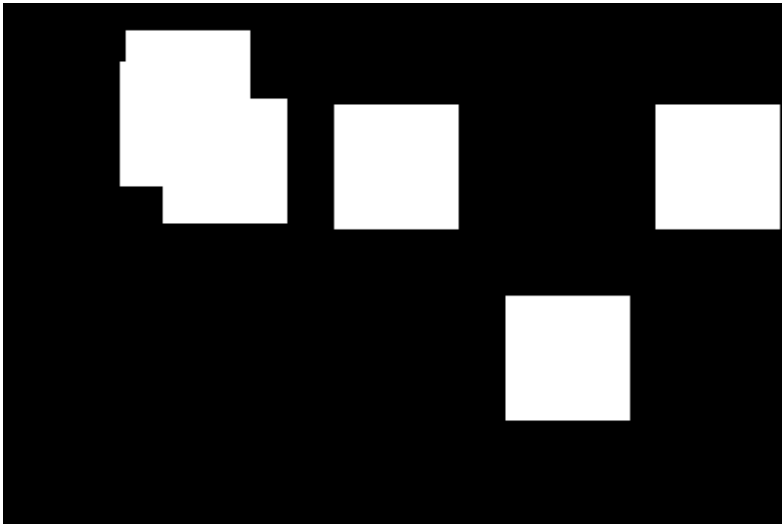
Citra *copy-move* merupakan citra yang sudah terkena serangan *copy-move*. Pada Gambar 3.1 terdapat 5 bagian daerah *copy-move* yang disalin dari bagian leher hewan dan ditempelkan di bagian lain.

Gambar 3.1 Contoh Data Masukan Citra yang Terkena Serangan *Copy-Move*

2. 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 dimana bagian yang terkena *copy-move* digambarkan dengan warna putih dan bagian yang tidak terkena *copy-move* digambarkan dengan warna hitam . Gambar 3.2

merupakan contohkunci jawaban citra yang sudah terkena serangan *copy-move*



Gambar 3.2 Contoh Data Masukan Citra Kunci Jawaban dari Citra yang Terkena Serangan *Copy-Move*

3. 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,
```

```
    "numBuckets" : 12000,
    "minArea" : 100,
  },

  "pixelNumber": {
    "pixel4" : [215, 40, 40, 0.9],
    "pixel3" : [0, 255, 0]
  }
}
```

Pseudocode 3.1 Contoh Threshold Config

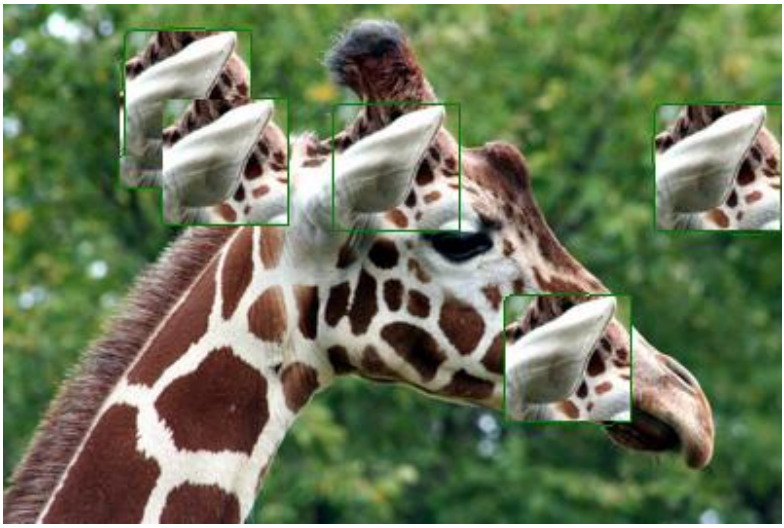
Penjelasan dari *threshold config* adalah sebagai berikut:

a. *pVal* adalah nilai antara 0 dan 1 yang digunakan sebagai batas untuk membandingkan *block*.
b. *numBuckets* adalah nilai yang digunakan untuk membuat *groups* dan *buckets* atau jumlah *buckets* yang digunakan untuk membanding *block*.
c. *blockSize* adalah ukuran dari *block* yang akan dibuat sehingga ukuran sebuah *block* adalah *blockSize x blockSize*.
d. *minArea* adalah nilai yang menunjukan daerah minimum dari bagian yang diduplikat.

Sedangkan *pixelNumber* adalah warna yang akan digunakan untuk menjadi penanda garis tepi bagian dalam citra yang terkena serangan *copy-move*.

3.1.2 Data Keluaran

Setelah 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. Gambar 3.3 merupakan contoh keluaran citra yang terdeteksi *copy-move* dengan *edge* dan Gambar 3.4 merupakan contoh citra yang terdeteksi serangan *copy-move* dengan *mask*.



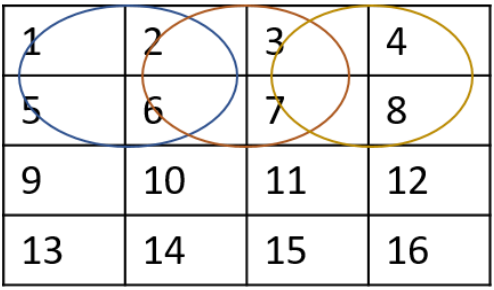
Gambar 3.3 Contoh Keluaran Citra yang Terdeteksi *Copy-Move* dengan *Edge*

Gambar 3.4 Contoh Keluaran Citra yang Terdedeteksi Copy-Move dengan *Mask*

3.2 Dekripsi Umum Sistem

Rancangan perangkat lunak pendeteksian serangan *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 3.5.



Gambar 3.5 Overlapping Block 2 x 2

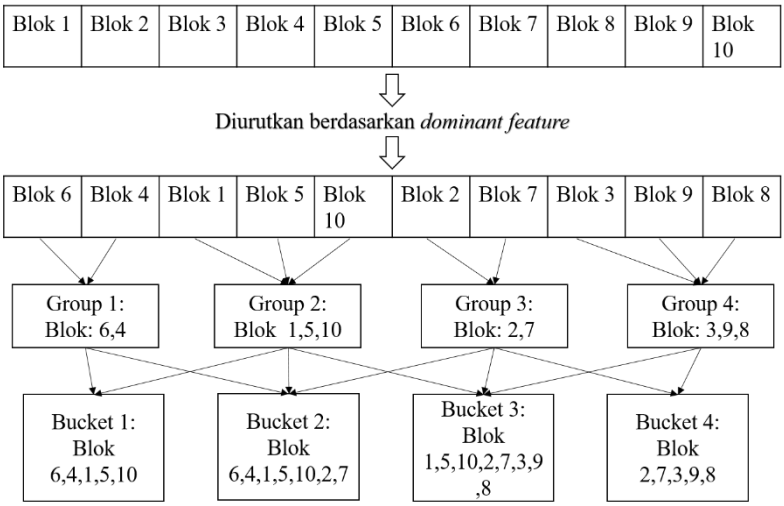
Tiap kotak pada Gambar 3.5 diasumsikan adalah piksel pada citra. Proses *overlapping block* pertama digambarkan pada area yang dibatasi dengan warna biru, *overlapping* kedua berada pada area yang dibatasi dengan warna merah. Banyaknya *overlapping block* sebesar $B \; x \; B$ pada sebuah citra berukuran $M \; x \; N$ dapa dihitung menggunakan Persamaan 3.1.

$$overlappingBlockSize = (M - B + 1) \; x \; (N - B + 1) \quad (3.1)$$

Setelah pembuatan *overlapping block* selesai, untuk setiap *block* akan dihitung *variance* sebagai *dominant feature*. Block akan diurutkan berdasarkan *dominant feature* secara *ascending*. Proses pengurutan ini menyebabkan *block* yang mirip akan berdekatan.

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

Setelah proses penempatan *block* kedalam *groups* selesai tahap selanjutnya adalah membuat *buckets* sebanyak *numBuckets*, untuk penempatan *groups* kedalam *buckets* adalah dengan cara menempatkan *groups* ke i-1, i, dan i+1 kedalam *buckets* i. Adapun proses pengurutan *block*, penempatan *block* ke *groups* dan penempatan group ke dalam *buckets* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Proses Pengurutan *Block* dan Penempatan ke *Buckets*

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

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

Jika *S* < *blockSize* ulangi langkah-langkah yang ada diatas dimana nilai *S = min(S*^{2}*, blockSize)*. Dari sisa *block* yang ada di dalam *bucket* hitung total area, jika total area kecil dari *minArea* maka lewatkan pengecekan dari *block* yang tersisa. Sisa *block* diasumsikan sebagai bagian dari *copy-move*.

Persamaan 3.2, Persamaan 3.3, dan Persamaan 3.4 adalah langkah – langkah untuk menghitung *statistic* antara *block Px* dan *block Py* dimana setiap *block* memiliki *b* piksel.

$$differencePixel = (Px - Py)^2 \quad (3.2)$$

$$sigmaSquare = \frac{variance(Px) - variance(Py)}{2} \quad (3.3)$$

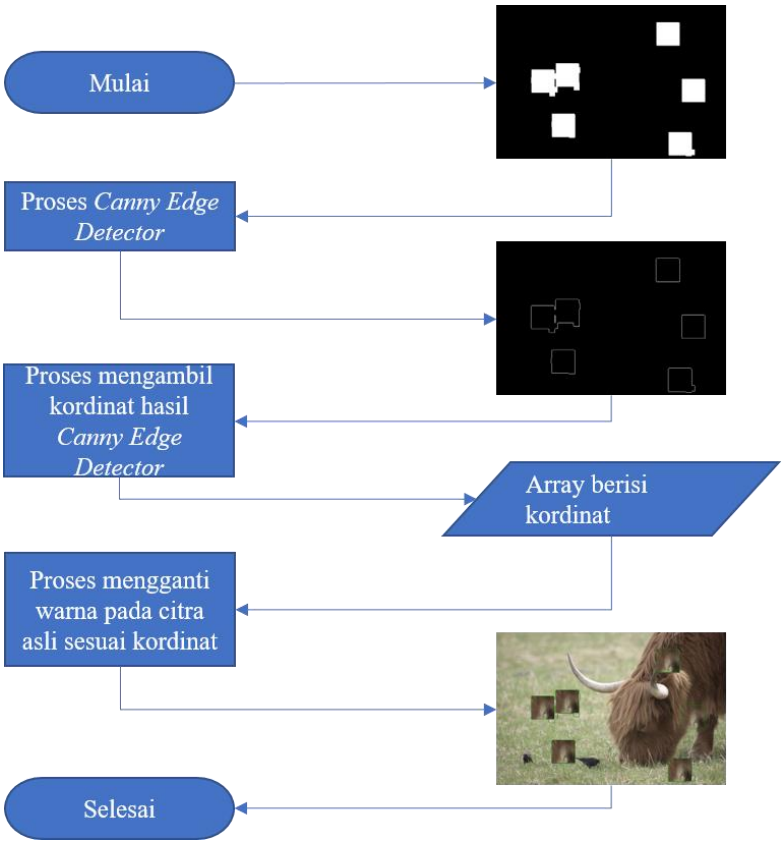
$$statistic = \frac{differencePixel}{sigmaSquare\ x\ b} \quad (3.4)$$

Persamaan 3.2 adalah proses untuk mencari perbedan antara dua *block* atau *differencePixel* dengan cara mengurangkan nilai piksel *block Px* dengan *block Py* dan selanjutnya mengkuadrakatan hasil pengurangan piksel antara dua *block* tersebut. Persamaan 3.3 adalah proses untuk menghitung *sigmaSquare* dengan cara mengurangi *variance block Px* dengan *variance block Py* dan hasilnya dibagi dua. Persamaan 3.4 merupakan proses mendapatkan *statistic* pada metode *expanding block algorithm* yaitu dengan cara membagi *differencePixel* dengan *sigmaSquare* yang telak dikalikan dengan *b* piksel. Diagram alir deteksi serangan *copy-move* pada citra menggunakan metode *expanding block algorithm* ditunjukan pada Gambar 3.9.

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 serangan *copy-move*. Sisa *block* hasil proses deteksi, dianggap sebagai daerah yang mengalami serangan *copy-move*. Setiap *block* yang tersisa akan digambar kembali ke dalam *mask* dengan menggunakan warna putih. Setelah proses penggambaran bagian *copy-move* selesai, citra akan disimpan sehingga menghasilkan citra hasil deteksi *copy-move* yang menggambarkan bagian mana saja yang terjadi serangan *copy-move*.

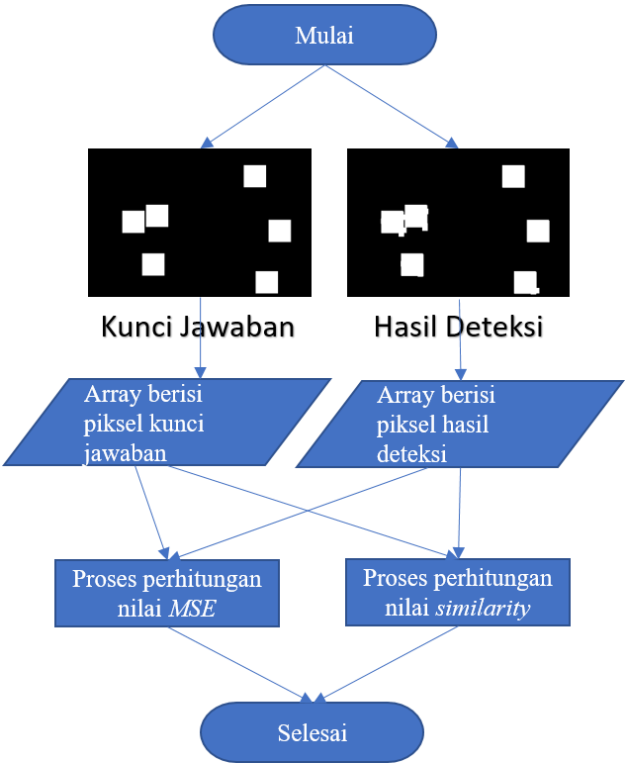
Tahap selanjutnya adalah pembuatan pola pinggiran. *Canny Edge Detector* berfungsi untuk mendapatkan pola pinggiran objek pada citra hasil deteksi serangan *copy-move*. Koordinat piksel dari pola pinggiran yang didapat menggunakan *Canny Edge Detector* akan disimpan. Proses selanjutnya adalah menggambar pola

pinggiran pada citra yang terkena serangan *copy-move* dengan menggunakan koordinat piksel yang telah didapat. Pada citra yang terkena serangan *copy-move* akan diganti warna sesuai dengan koordinat dengan warna hijau. Proses ini menghasilkan citra dengan pola pinggiran dengan warna hijau pada bagian citra yang terkena serangan *copy-move*. Diagram alir pembuatan garis tepi pada citra yang terkena serangan *copy-move* ditunjukan pada Gambar 3.7.

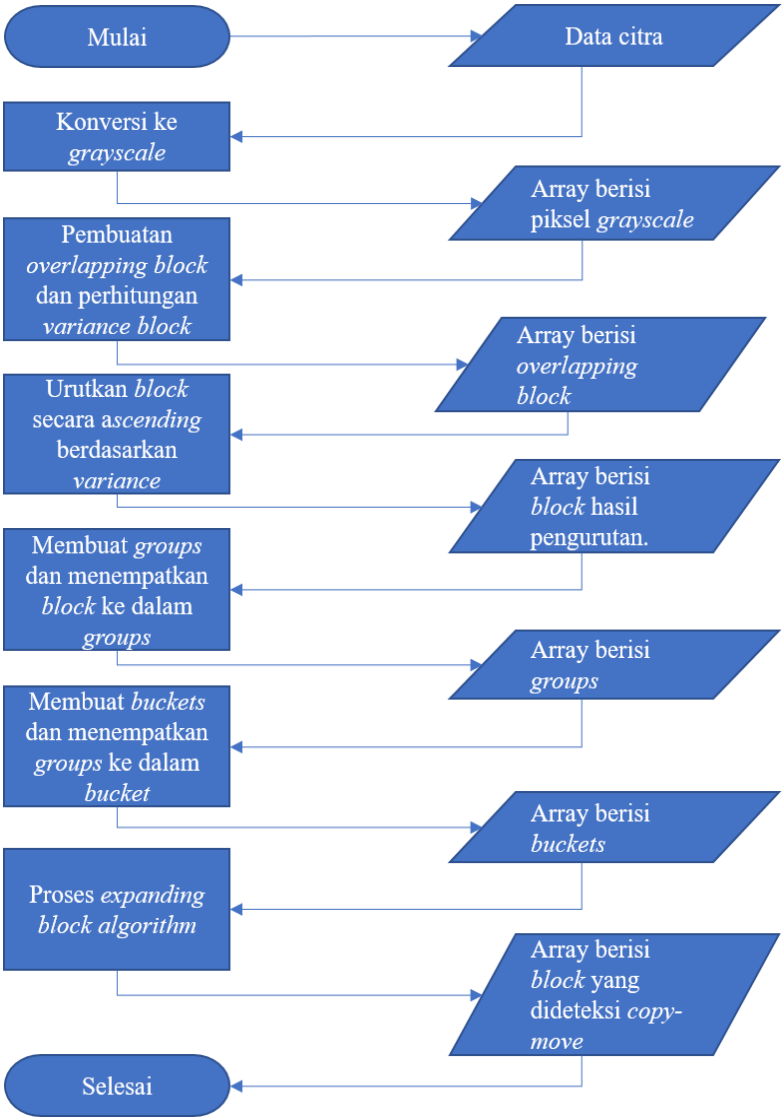


Gambar 3.7 Proses Pembuatan Garis Tepi pada Citra *Copy-Move*

Setelah semua proses pendeteksian *copy-move* pada citra selesai, proses selanjutnya adalah proses perhitungan nisal *MSE* dan *similarity*. Proses perhitungan nilai *MSE* dan *similarity* merupakan proses diluar program pendeteksian *copy-move* pada citra, melainkan program lain untuk proses untuk melakukan analisis terhadap hasil pendeteksian pada citra yang terkena serangan *copy-move*. Dengan mendapatkan nilai *MSE* dan *similarity* memudahkan untuk mencari perbedaan antara citra kunci jawaban dan citra hasil deteksi *copy-move*. Diagram alir proses perhitungan nilai *MSE* dan *similarity* ditunjukan pada Gambar 3.8.



Gambar 3.8 Proses Perhitunga Nilai *MSE* dan *Similarity*



Gambar 3.9 Proses Deteksi *Copy-Move* pada Citra Menggunakan Metode *Expanding Block Algorithm*

3.3 Modifikasi *Expanding Block Algorithm*

Modifikasi pertama yang dilakukan pada metode *expanding block algorithm* adalah dengan menambahkan *filter* sebelum proses deteksi *copy-move* dilakukan. *Filter* bertujuan untuk mengurangi *noise* yang terjadi pada citra yang terkena serangan *copy-move*. Terdapat dua Metode *filter* yang digunakan yaitu *averaging filter* dan *median filter*.

Modifikasi kedua yang dilakukan adalah menggunakan *dominant feature* dari citra yang memiliki tiga warna dasar atau tiga *channel* (*red*,*gren*,*blue*) pada metode *expanding block algorithm*, dimana pada dasarnya untuk mencari *dominant feature* pada metode *expanding block algorithm* adalah dari citra yang memiliki satu warna dasar atau satu *channel* (*grayscale image*). Adapun pilihan terbaik untuk *dominant feature* dari citra bewarna adalah dengan mencari rata-rata intensitas ((*red* + *green* + *blue*) / 3) dari sebuah piksel yang ada di dalam *block*.

Modifikasi ketiga yang dilakukan adalah dengan membuat *threshold config* sebagai masukan parameter pada *expanding block algorithm*. Tujuan pembuatan *threshold config* adalah untuk melakukan pendeteksian *copy-move* secara efisien dimana pengguna dapat melakukan pengaturan parameter sesuai dengan kondisi citra yang terkena serangan *copy-move*. *Threshold config* yang dibuat bertipe data JSON.

Modifikasi keempat yang dilakukan adalah pembuatan *edge* atau garis tepi dengan warna pada citra hasil keluaran pendeteksian serangan *copy-move*. Pembuatan garis tepi menggunakan *Canny Edge Detector*. Tujuan pembuatan garis tepi ini adalah untuk menandakan bagian mana saja yang telah disalin.

3.4 Metode *Filter*

Filter merupakan 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.

3.4.1 *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. Gambar 3.10 merupakan contoh proses *averaging filter* pada matriks dengan ukuran *3 x 3*, dimana bagian A merupakan matriks sebelum proses *averaging filter* dan bagian B merupakan matriks setelah proses *averaging filter* dengan cara mencari rata-rata.

5	3	6
2	1	9
8	4	7

A

*	*	*
*	5	*
*	*	*

B

Gambar 3.10 Proses *Averaging Filter*

3.4.2 *Median Filter*

Median *filter* merupakan *filter order-statistic* yang paling banyak diketahui, yang mana sesuai namanya, *filter* ini mengganti nilai pixel dengan nilai *median* dari 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 *filter*s secara khusus efektif dalam keadaan bipolar dan unipolar *impulse noise*. Median *filter* memberikan hasil yang sangat bagus untuk image yang dirusak oleh *noise*. Gambar 3.11 merupakan contoh proses *median filter* pada matriks *3 x 3* dimana bagian A merupakan matriks sebelum proses *median filter* dan bagian B merupakan matriks setelah proses *median filter* dengan cara mencari nilai tengah.

6	2	0
3	97	4
19	3	10

A

*	*	*
*	4	*
*	*	*

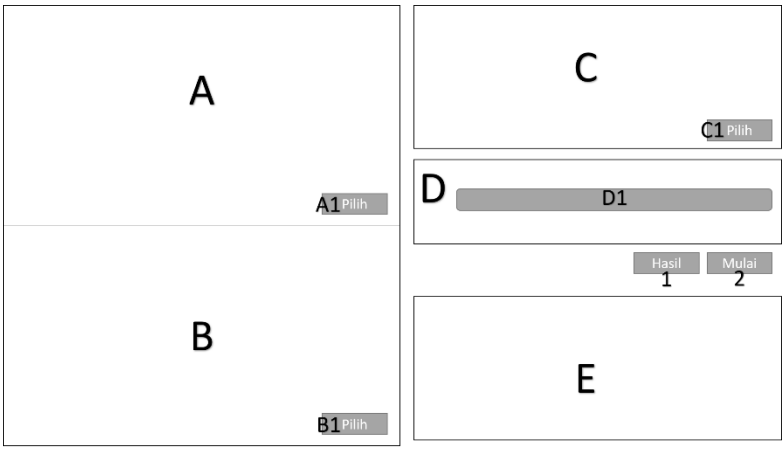
B

Gambar 3.11 Proses *Median Filter*

3.5 Perancangan Antarmuka

Aplikasi antarmuka digunakan untuk mempermudah pengguna dalam melakukan pendeteksian serangan *copy-move* pada citra mengguanakan metode *expanding block algorithm*. Aplikasi antarmuka yang dibuat pada Tugas Akhir ini menggunakan pustaka Qt pada bahasa pemrograman Python dengan ditribusi pustaka PyQt5. Aplikasi antamuka yang dibangun dengan menggunakan pustaka Qt merupakan aplikasi *multiplatform* yang dapat berjalan di berbagai sistem operasi seperti Windows, Linux, dan lain-lain. Untuk mendesain aplikasi antamuka yaitu menggunakan aplikasi Qt-Creator, sehingga memudahkan untuk menempatkan *layout, button, form, dropdown, textearea, textviews, menubar, toolbar* dan lain sebagainya.

Gambar 3.12 merupakan desain tampilan antarmuka yang dibuat pada Tugas Akhir ini.



Gambar 3.12 Desain Tampilan Antarmuka

Adapun komponen yang terdapat dalam Gambar 3.12 antara lain:

A. 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.
B. 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. Form ini bersifat *optional*, tidak harus diisi.
C. 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.

D. 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.
E. 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.

4.1 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 4.1 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

4.2 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.

4.2.1 Kelas *Block*

Kelas *block* digunakan sebagai objek atau *container* yang digunakan untuk menampung *block* dari proses membangun *overlapping block*. Kelas *block* nantinya digunakan untuk melakukan proses pengelompokan *block* yang mirip dengan mencari kemiripan fitur *variance*. Untuk detail dari kelas *block* dapat dilihat pada Pseudocode 4.1.

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

Pseudocode 4.1 Kode Program Kelas *Block Container*

4.2.1.1 Detail Atribut Kelas *Block*

Untuk detail dari setiap atribut yang ada di kelas *block* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Atribut Kelas *Block Container*

Nama Atribut	Keterangan
row	Merupakan indeks baris dari *block overlap*
col	Merupakan indeks kolom dari *block overlap*

blockImage	Merupakan *block* overlap dengan ukuran *blockSize* x *blockSize*
variance	Merupakan nilai *variance* dari *block overlap*

4.2.2 Inisialisasi *File*

Inisialiasi *file* merupakan proses awal dari program, pada inisialisasi *file* program akan mendefenisikan *path* citra *copy-move*, path kunci jawaban citra *copy-move*, *path threhsold config* dan *filter* yang dipilih. Setelah itu program membaca citra masukan, konversi ke *grayscale* ketika memilih *image gray dominant feature*, mencari dimensi citra, melakukan *filter* terhadap citr. Proses inisialisasi *file* dapat dilihat pada Pseudocode 4.2.

```
1   CLASS ExpandingBlock:
        FUNCTION __init__(self, fileName,
2   keyFileName, fileConfig, statusFilter):
3           fileName <- fileName
4           fileConfig <- fileConfig
5           statusFilter <- statusFilter
6           keyFileName <- keyFileName
7           initFile()
8           initThreshold()
9       ENDFUNCTION
10
11      FUNCTION initFile(self):
12          try:
13              image <- io.imread( fileName)
14              IF  statusFilter = 1:
15                   image <- cv2.blur( image,(5,
16  5))
17              ENDIF
18              IF  statusFilter = 2:
                     image <- cv2.medianBlur( image,
19  5)
20              ENDIF
21              height,  width, _ <- np.shape(
    image)
```

```
                lenpixel <- len( image[0][0])
22              #using grayscale image dominant
23  feature
                 grayImage <- color.rgb2gray( image)
24              #using rbg intensity dominant
25  feature
                 grayImage <-  RGBFeature()
27               size <-  height *  width
28          except:
29              OUTPUT 'File Not Found'
30      ENDFUNCTION
```

Pseudocode 4.2 Kode Program Inisialisasi *File*

Proses inisialisasi *file* dimulai dari pustaka *skiimage* dengan modul *io* membaca *file* dan ditampung ke variabel image, setelah itu citra dikonversi menjadi *grayscale image* untuk mencari *image gray dominant feature* tetapi ketika memilih *image color dominant feature* citra tidak akan dikonversi menjadi *grayscale*. Proses inisialisasi *file* juga menyimpan dimensi citra yaitu dengan menggunakan fitur *shape* yang mengembalikan nilai *height, width* dan *channel*, kemudian untuk mencari dimensi citra dilakukan perkalian *height* dengan *width*. Proses inisialisasi *file* juga mendefenisikan apakah sebuah citra itu *RGB* atau *RGBA*. Selain itu proses penggunaan *filter* juga dilakukan oleh fungsi ini. Ketika *file* tidak ditemukan maka program akan menampilkan pesan *error*.

4.2.3 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   FUNCTION initThreshold(self):
2           with open( fileConfig,'rb') as file:
4                   data <- json.load(file)
3                   minArea <-
    data['threshold']['minArea']
```

```
6                 blockSize <-
    data['threshold']['blockSize']
7                 numBuckets <-
    data['threshold']['numBuckets']
8                 pVal <- data['threshold']['pVal']
9                 blockDistance <-
    data['threshold']['blockDistance']
10                varianceThreshold <-
    data['threshold']['varianceThreshold']
11                pixel3 <-
    data['threshold']['pixel3']
12                pixel4 <-
    data['threshold']['pixel4']
13  ENDFUNCTION
```

Pseudocode 4.3 Kode Program Inisialisasi *Threshold*

4.2.4 Perhitungan RGB *Feature*

Fungsi ini bertujuan untuk melakukan perhitungan terhadapa intensitas RGB pada citra. Proses perhitungan intensitas RGB yaitu dengan cara menjumlah nilai R, nilai G, dan nilai B lalu dibagi dengan tiga atau ((R+G+B)/3). Fungs ini akan mengembalikan sebuah *array* yang berisi intensitas dari RGB pada citra. Proses perhitungan RGB *feature* ditunjukan oleh Pseudocode 4.4.

```
1   FUNCTION RGBFeature(self):
2         divison <- 4.0 IF  lenpixel = 4 else 3.0
    ENDIF
3         temp <- np.ones(( height,  width))
4         for i in range( height):
5               for j in range( width):
6                     sum <- np.sum( image[i, j])/
    divison
7                     temp[i,j] <- sum
8               ENDFOR
9         ENDFOR
10        RETURN temp
11  ENDFUNCTION
```

Pseudocode 4.4 Kode Program Perhitungan RGB *Feature*

4.2.5 Pembuatan *Overlapping Block*

Overlapping block merupakan proses membuat *block* yang *overlap* dari gambar asli. Block yang dibuat merupakan *block* dinamis sesuai dengan *config* yang telah ditetapkan. Proses selanjutnya *block* yang telah dibuat akan dibagi ke dalam group secara merata. Proses pembuatan *overlapping block* dapat dilihat pada Pseudocode 4.5.

```
1   FUNCTION overlappBlock(self):
2         overlappingBlockLen <- ( height -
    blockSize + 1) * ( width -  blockSize + 1)
3         overlappingBlock <- []
4         for i in range( height -  blockSize + 1):
5               for j in range( width -  blockSize
    + 1):

6   overlappingBlock.append(Container( grayImage, i,
    j,  blockSize))
7               ENDFOR
8         ENDFOR
9         overlappingBlock.sort(key=lambda val:
    val.variance)
10  ENDFUNCTION
```

Pseudocode 4.5 Kode Program *Overlapping Block*

Setelah pembuatan *overlapping block* selesai proses selanjutnya adalah mengurutkan *block* berdasarkan *dominant feature*. Fitur *block* yang digunakan adalah variance, sehingga *block* akan diurutkan berdasarkan variance secara *ascending*. Proses pengurutan *block* dapat dilihat pada Pseudocode 4.6.

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

Pseudocode 4.6 Kode Program Pengurutan *Block*

4.2.6 Pembuatan *Groups*

Proses pembuatan *groups* merupakan proses membagi *block* secara merata ke dalam *groups*. Banyak *groups* yang dibuat adalah sebanyak *numBuckets* yang telah ditetapkan pada saat proses inisialisasi *threshold*. Proses pembuatan *groups* dapat dilihat pada Pseudocode 4.7.

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

Pseudocode 4.7 Kode Program Pembuatan *Groups*

Proses awal pembuatan *groups* adalah dengan mendeklarasikan sebuah *array* yang berisi *array* kosong sepanjang *numBuckets* yang telah ditentukan. Setelah itu program akan mendefenisikan banyak *block* yang ada di dalam satu *groups* yaitu dengan cara membagi total panjang *overalapping block* dengan *numBuckets*. Setelah mendapatkan banyak *block* yang ada di dalam *groups*, program akan membagi secara merata *block* ke dalam *groups* yaitu dengan cara memasukan *block* sebanyak *blockpergroup* yang telah ditentukan ke dalam satu *groups*. Proses pembagian *block* secara merata ke dalam groups ditunjukan oleh Pseudocode 4.7 pada baris 6 – 14.

4.2.7 Pembuatan *Buckets*

Proses pembuatan *buckets* merupakan proses membagi *groups* yang telah dibuat ke dalam *buckets*. Banyak *buckets* yang dibuat adalah sebanyak *numBuckets*. Proses pembuatan *buckets* dapat dilihat pada Pseudocode 4.8.

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

Pseudocode 4.8 Kode Program Pembuatan Bucket

Proses awal pembuatan *buckets* adalah dengan mendeklarasikan *array* kosong sepanjang *numBuckets* yang telah ditentukan. Setelah pembuatan *buckets* kosong selesai, selanjutnya proses meletakan *groups* ke dalam *buckets*. Proses meletakan *groups* kedalam *bucket*s mempunyai alur yaitu dengan cara meletakan group *i – 1, i , i + 1* ke dalam *buckets i*.

4.2.8 Perhitungan Nilai *Statistic*

Proses ini yaitu untuk menghitung nilai *statistic* untuk setiap pasangan perbandingan sub *block* di setiap *block* yang ada di dalam *bucket*. Proses untuk mengitung nilai *statistic* dapat dilihat pada Pseudocode 4.9.

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

Pseudocode 4.9 Kode Program Perhitungan *Statistic*

4.2.9 Pengecekan *Overlap Block*

Proses ini adalah proses pengecekan *overlap* setiap *block* yang ada di dalam *bucket*. Tahapanya adalah mengambil *row* dan *col* dari setiap *block* yang ada di dalam *bucket*. Selanjutnya yaitu mencari nilai mutlak dari *row* dan *col*. Sebuah *block* dinyatakan *overlap* ketika nilai mutlak dari *row* atau *col* kurang dari *blockSize* yang telah ditentukan. Adapun proses pengecekan *overlap block* dapa dilihat pada Pseudocode 4.10.

```
I   Bucket
O   Status Overlap
1   FUNCTION findOverLap(self, bucket):
2           row, col <- [],[]
3           for block in bucket:
4                   row.append(block.row)
```

```
5               col.append(block.col)
6       ENDFOR
7       rowDistance <- np.abs(row -
np.reshape(row, (-1, 1)))
8       colDistance <- np.abs(col -
np.reshape(col, (-1, 1)))
9       rowOverlap <- rowDistance <  blockSize
10      colOverlap <- colDistance <  blockSize
11      RETURN np.logical_or(rowOverlap,
colOverlap)
12  ENDFUNCTION
```

Pseudocode 4.10 Kode Program Pengecekan *Overlap*

4.2.10 Pengecekan *Connection*

Proses ini merupakan proses untuk menandakan apakah setiap *block* yang ada di dalam *buckets* memiliki kesamaan satu sama lain. Proses ini akan mengembalikan sebuah *array* yang berisi status dari *bucket* tersebut. Proses ini melakukan pengecekan dengan menggunakan nilai *statistic* serta status *overlap* yang telah didapatkan. Proses pengecekan ini dapat dilihat pada Pseudocode 4.11.

```
I   Statistic dan Status Overlap
O   Status Connection
1   FUNCTION findConnection(self, statistic,
    overLap):
2       tooSimilar <- statistic <  pVal
3       connection <- np.ones_like(tooSimilar)
        connection <- np.logical_xor(connection,
4   np.logical_or(overLap,
5   np.logical_not(tooSimilar)))
6       RETURN connection
7   ENDFUNCTION
```

Pseudocode 4.11 Kode Program Pengekan *Connection*

4.2.11 Pengecekan *Buckets*

Proses ini merupakan proses pengecekan setiap *bucket*s. Proses ini akan memanggil beberapa proses yaitu perhitungan nilai *statistic*, pengecekan *overlap block* serta pengecekan connection setiap *block* yang ada di dalam *buckets*. Adapun proses pengecekan setiap *buckets* ditunjukan oleh Pseudocode 4.12.

I	Bucket
O	Bukcet dengan status *copy-move*

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

Pseudocode 4.12 Kode Program Memproses Setiap Bucket

4.2.12 Penyimpanan Hasil Deteksi

Fungsi ini merupakan fungsi untuk menyimpan citra hasil deteksi *copy-move*. Fungsi ini berjalan dengan mengecek sisa *block* yang ada dari hasil pendeteksian *copy-move*. Ketika sisa *block* hasil deteksi *copy-move* adalah nol maka citra tidak terdeteksi adanya serangan *copy-move*. Ketika terdapat *block* 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 4.13.

```
1   FUNCTION buildCopyMoveImage(self):
2           IF len( blocks) = 0:
3                   OUTPUT "Fail to build copy move
    because no copy move detected"
4           ENDIF
5           mask <-  createMask()
6           imageOut <- np.uint8( writeMask(mask))
7           imageWedge <-  buildEdgeImage( imageOut)
8           temp <-  fileName.split("/")[1].split(".")
9           realName <- temp[0]
10          type <- temp[1]
11          maskName <- "Hasil-TA/parameter/14000/"+
    realName+"-Mask."+ type
12          edgeName <- "Hasil-TA/parameter/14000/"+
    realName+"-Edge."+ type
13          io.imsave( maskName, imageOut)
14          io.imsave( edgeName,imageWedge)
15  ENDFUNCTION
```

Pseudocode 4.13 Kode Program Membangun Gambar *Copy-Move*

4.2.13 Pembuatan *Mask*

Pembuatan *mask* adalah pembuatan citra hasil deteksi *copy-move*. Proses pembuatan *mask* dimulai membuat sebuah *array* dua dimensi yang berusukuran sama seperti citra yang terkena serangan *copy-move*. Dari setiap *block* yang tersisa dari pendeteksian *copy-move*, diambil koordinat dari *block* tersebut

sepanjang *blockSize* yang telah ditentukan untuk diisi sebuah *channel* yang menandakan daerah *copy-move*. Adapun proses pembuatan *mask* dapat dilihat pada Pseudocode 4.14.

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

Pseudocode 4.14 Kode Program Membuat Gambar *Mask*

4.2.14 Penulisan *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 menandakan daerah 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* ditunjukan oleh Pseudocode 4.15.

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

Pseudocode 4.15 Kode Program Menempelkan *Mask* Ke Gambar

4.2.15 Pembuatan *Edge* Citra *Copy-Move*

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 4.16.

```
I   Gambar dengan mask
O   Gambar dengan edge
1   FUNCTION buildEdgeImage(self, imageOut):
2           edges <- cv2.Canny(imageOut,  height,
    width)
3           coord <- np.column_stack(np.where(edges =
    255))
4           imageWedge <-  image
5           pixel <-  pixel4 IF  lenpixel = 4 else
    pixel3
6           for index in coord:
7                   x <- index[0]
```

```
8                   y <- index[1]
9                   imageWedge[x, y] <- pixel
10          ENDFOR
11          RETURN imageWedge
12  ENDFUNCTION
```

Pseudocode 4.16 Kode Program Membuat *Edge* Pada Citra

4.2.16 Perhitungan Nilai *MSE*

Mean squared error adalah metode untuk mengevaluasi dua 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 4.17.

```
I   Citra kunci jawaban dan citra hasil deteksi
O   MSE
1   FUNCTION accuracyMSE(self,imgKey,imgPredict):
2           mse <- np.sum((imageA.astype("float") -
    imageB.astype("float")) ** 2)
3             mse /= float(imageA.shape[0] *
    imageA.shape[1])
4           RETURN mse
5   ENDFUNCTION
```

Pseudocode 4.17 Kode Program Menghitung *Mean Squared Error*

4.2.17 Perhitungan Nilai *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 pada citra kunci jawaban dengan citra hasil deteksi *copy-move*. Hasil kemiripan dapat dihitung dengan cara membagi total piksel yang sama dengan total semua piksel. Adapun proses menghitung kemiripan dapat dilihat pada Pseudocode 4.18.

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

Pseudocode 4.18 Kode Program Menghitung Kemiripan

4.2.18 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 menjalankan deteksi dapat dilihat pada Pseudocode 4.19.

```
1   FUNCTION getFilter(self,i):
2           filterChoice <- str(i)
3   ENDFUNCTION
4
5   FUNCTION append(self, text):
6           cursor <- logOutput.textCursor()
7           cursor.movePosition(cursor.End)
8           cursor.insertText(text)
9   ENDFUNCTION
10
11  FUNCTION stdoutReady(self):
12          text <- str(
    process.readAllStandardOutput())
13          OUTPUT text.strip()
```

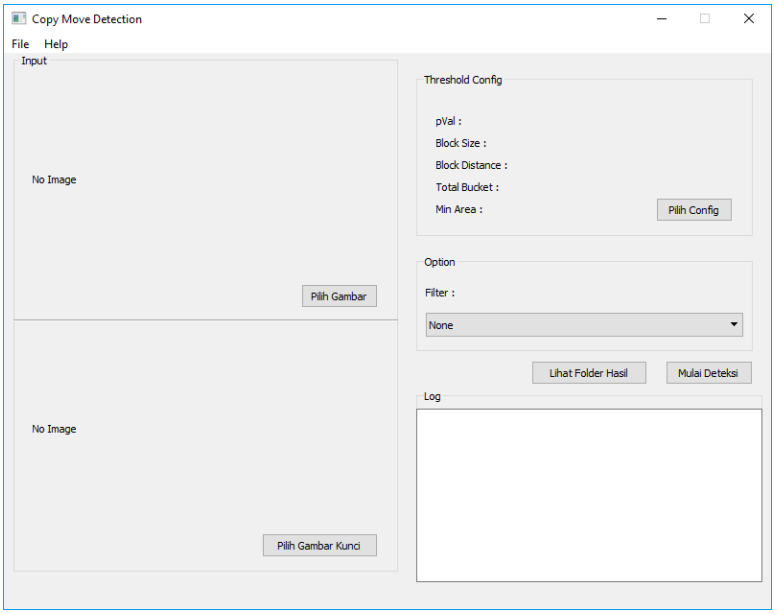
```
14          append(text)
15  ENDFUNCTION
16
17  FUNCTION stderrReady(self):
18          text <- str(
    process.readAllStandardError())
19          OUTPUT text.strip()
20          append(text)
21  ENDFUNCTION
22
23  FUNCTION startDetection(self):
24          argv <-
    ["/media/sudirahanif/Kuliah/TA/Program/TugasAkhi
    r.py", imageFile, thresholdFile, filterChoice]
25          process.start('/home/sudirahanif/anaconda2
    /bin/Python',argv)
26  ENDFUNCTION
27
28  FUNCTION initUI(self):
29          buttonStart.clicked.connect(startDetection
    )
30          process <- QProcess(self)
31          process.readyReadStandardOutput.connect(st
    doutReady)
32          process.readyReadStandardError.connect(std
    errReady)
33          process.started.connect(lambda:
    buttonStart.setEnabled(False))
34          process.finished.connect(lambda:
    buttonStart.setEnabled(True))
35  ENDFUNCTION
```

Pseudocode 4.19 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*. Tampilan anatamuka juga akan menampilkan *log* saat program dijalankan, tampilan *log* bertujuan untuk melihat proses pendeteksian *copy-*

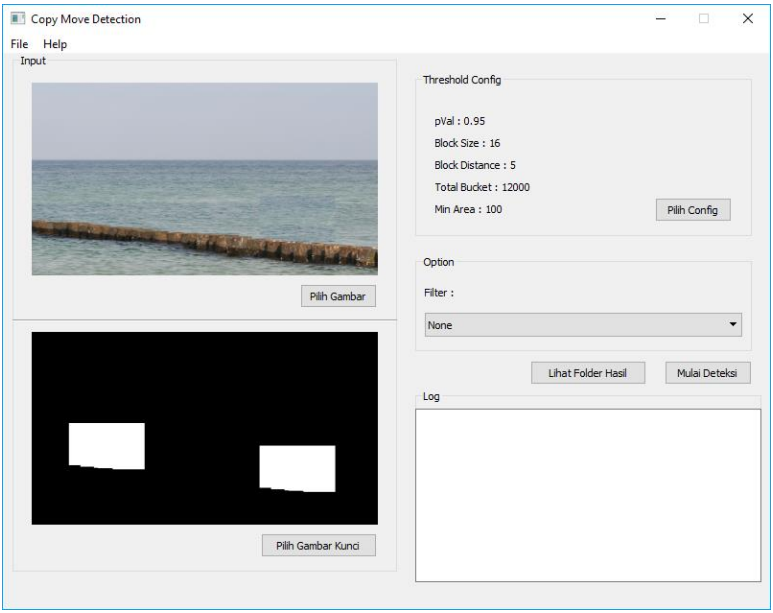
move. Untuk implementasi antarmuka utama dapat dilihat pada Gambar 4.1 dan Gambar 4.2.

Pada Gambar 4.1 merupakan tampilan awal antarmuka saat program dijalankan. Pada saat program pertama dijalankan, semua *form* yang ada akan kosong.



Gambar 4.1 Antarmuka Awal Dijalankan

Pada Gambar 4.2 merupakan tampilan saat semua *form* telah diisi. Tampilan antarmuka akan berubah sesuai dengan masukan *form*, seperti akan menampilkan data citra masukan, data citra kunci jawaban dan menampilkan *threshold* yang digunakan.



Gambar 4.2 Antarmuka Masukan Data

[Halaman ini sengaja dikosongkan]

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.

5.1 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. Sistem operasi *Windows* digunakan untuk membangun program *expanding block algorithm* sedangkan sistem operasi *Linux Mint Cinnamon* digunakan untuk membangun antarmuka program yang dibangun menggunakan Qt.

Tabel 5.1 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

5.2 Data Pengujian

Sub bab ini mejelaskan mengenai data yang digunakan untuk uji coba Adapun data pengujian dalam Tugas Akhir ini adalah sebagai berikut:

5.2.1 Citra *Copy-Move*

Data citra *copy-move* terdiri dari 20 buah citra yang mengalami serangan *copy-move* yang akan disebut sebagai citra *copy-move* asli dan 20 buah citra sama yang mengalami serangan *copy-move* tetapi sudah dilakukan perubahan dengan menambah *noise unifrom distribution* sebesar 0.5% menggunakan program aplikasi pengolah citra *Photoshop* yang akan disebut sebagai citra *copy-move noise*. Terdapat juga 20 buah citra kunci jawaban yang menunjukan bagian-bagian yang terkena serangan *copy-move.* Citra yang digunakan adalah citra yang beresolusi kecil.

Berikut merupakan citra *copy-move* sebagai data uji beserta kunci jawaban.

1. 01_giraffe.png

Pada Gambar 5.1 bagian yang disalin adalah bagian telinga hewan jerapah dan ditempelkan sebanyak empat kali di bagian lain citra ini.

Gambar 5.1 01_giraffe.png dan Kunci Jawaban

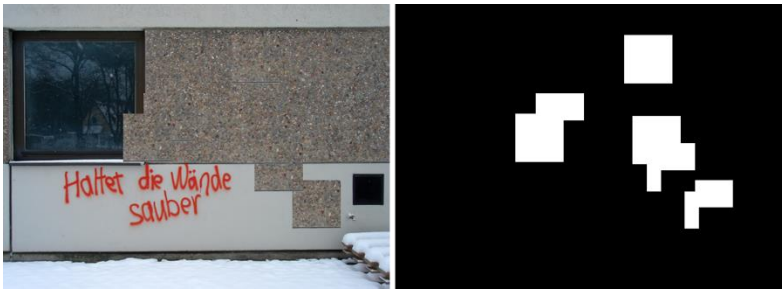
2. 02_view.png

Pada Gambar 5.2 bagian yang disalin adalah bagian rumput yang ada di pemandangan, dan ditempelkan sebanyak lima kali di bagian lain citra ini.

Gambar 5.2 02_view dan Kunci Jawaban

3. 03_clean_wall.png

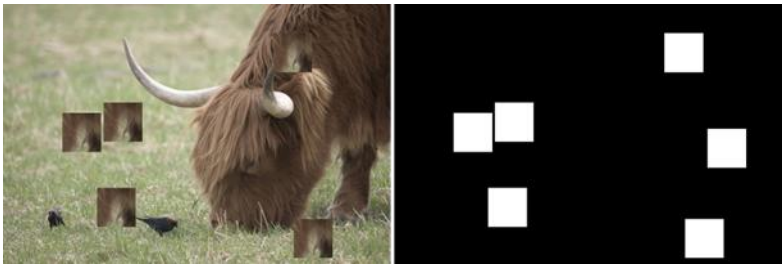
Pada Gambar 5.3 terdapat tiga objek pada bagian yang disalin, objek pertama berbentuk persegi, objek kedua berbentuk persegi panjang besar dan objek ketiga berbentuk persegi panjang kecil. Objek pertama ditempel sebanyak dua kali, objek kedua ditempel sebanyak satu kali dan objek ketiga ditempel sebanyak satu kali. Pada citra ini terdapat bagian disalin yang berimpitan.



Gambar 5.3 03_clean_wall dan Kunci Jawaban

4. 04_cattle.png

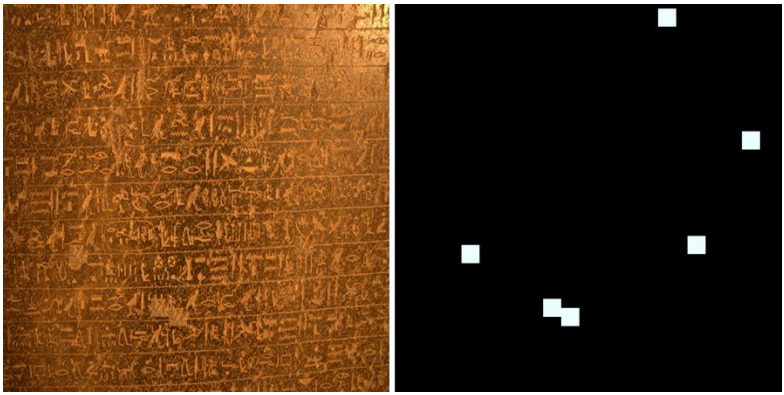
Pada Gambar 5.4 bagian yang disalin adalah bagian leher hewan yang ada digambar. Bagian yang disalin ditempelkan sebanyak 5 kali di bagian lain citra ini.



Gambar 5.4 04_cattle dan Kunci Jawaban

5. 05_history.png

Pada Gambar 5.5 bagian yang disalin adalah bagian paling atas sebelah kanan yang ada di citra ini, kemudian ditempelkan sebanyak lima kali di bagian lain pada citra ini.



Gambar 5.5 05_history dan Kunci Jawaban

6. 06_three_hundred.png

Pada Gambar 5.6 bagian yang disalin adalah bagian atas sebelah kanan yang ada pada citra ini, kemudian ditemplekan sebanyak lima kali di bagian lain pada citra ini.

Gambar 5.6 06_three_hundred dan Kunci Jawaban

7. 07_tree.png

Pada Gambar 5.7 bagian yang disalin adalah bagian ranting di pohon dan ditempelkan sebanyak satu kali pada citra ini.

Gambar 5.7 07_three dan Kunci Jawaban

8. 08_coin.png

Pada Gambar 5.8 bagian yang disalin adalah logo yang ada di tengah koin pada citra ini lalu ditempelkan sebanyak dua kali ini tengah koin citra lain.

Gambar 5.8 08_coin dan Kunci Jawaban

9. 09_fountain.png

Pada Gambar 5.9 bagian yang disalin adalah langit yang ada di citra ini lalu ditempelkan sebanyak satu kali pada citra ini.

Gambar 5.9 09_fountain dan Kunci Jawaban

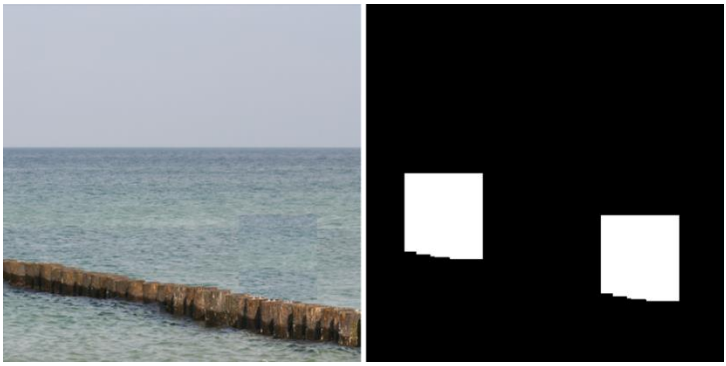
10. 10_park.png

Pada Gambar 5.10 bagian yang disalin adalah objek mobil yang ada di parkir, lalu ditempelkan sebanyak satu kali di bagian lain tempat parkir pada citra ini.

Gambar 5.10 10_park dan Kunci Jawaban

11. 11_barrier.png

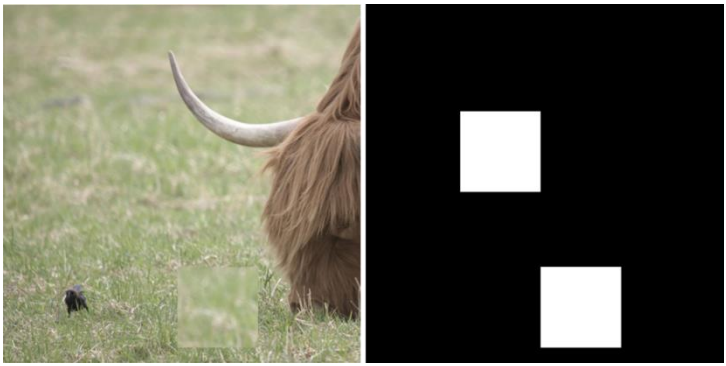
Pada Gambar 5.11 bagian yang disalin adalah bagian air laut dan ditempelkan sebanyak satu kali pada citra ini.



Gambar 5.11 11_barrier dan Kunci Jawaban

12. 12_cattle_remake.png

Pada Gambar 5.12 bagian yang disalin adalah objek rumput dan ditempelkan sebanyak satu kali di bagian rumput lain pada citra ini.



Gambar 5.12 12_cattle_remake dan Kunci Jawaban

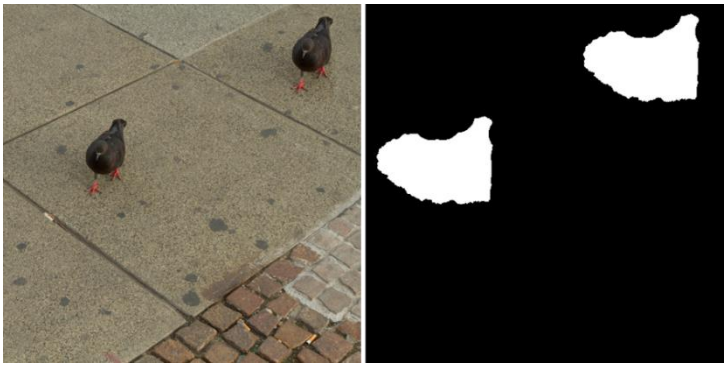
13. 13_clean_wall_remake.png

Pada Gambar 5.13 bagian yang disalin adalah objek dinding dan ditempelkan sebanyak satu kali pada citra ini.

Gambar 5.13 13_clean_wall_remake dan Kunci Jawaban

14. 14_bird.png

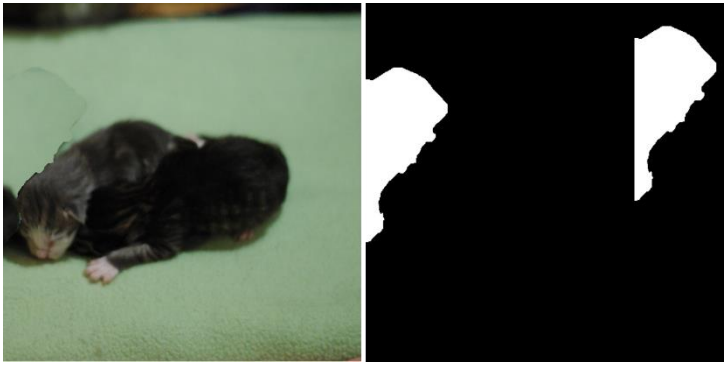
Pada Gambar 5.14 bagian yang disalin adalah objek burung yang ada pada citra ini lalu ditempelkan sebanyak satu kali di bagian lain pada citra ini.



Gambar 5.14 14_bird dan Kunci Jawaban

15. 15_four_babies.png

Pada Gambar 5.15 bagian yang disalin adalah bagian diatas kepala hewan lalu ditempelkan sebanyak satu kali pada citra ini.



Gambar 5.15 15_four_babies dan Kunci Jawaban

16. 16_ship.png

Pada Gambar 5.16 bagian yang disalin adalah objek nomor dan objek pintu lalu ditempelkan sebanyak satu kali masing-masing obej di bagian lain pada citra ini.

Gambar 5.16 15_ship dan Kunci Jawaban

17. 17_tree_flower.png

Pada Gambar 5.17 bagian yang disalin adalah bagian objek bunga lalu ditempelkan sebanyak dua kali pada citra ini.

Gambar 5.17 17_tree_flower dan Kunci Jawaban

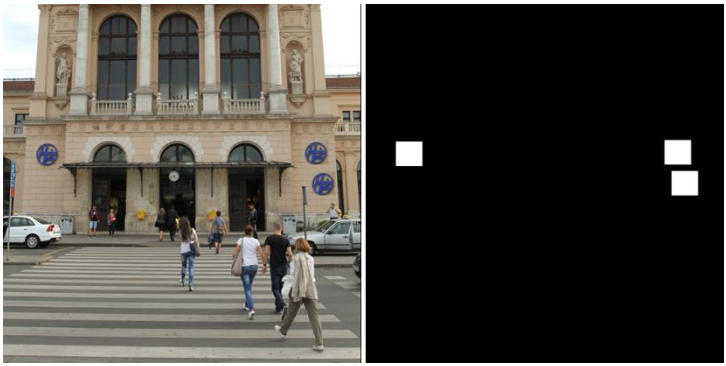
18. 18_flower.png

Pada Gambar 5.18 bagian yang disalin adalah objek bunga lalu ditempelkan sebanyak satu kali di bagian lain pada citra ini.

Gambar 5.18 18_flower dan Kunci Jawaban

19. 19_stree_building.png

Pada Gambar 5.19 bagian yang disalin adalah objek logo yang ada di dinding dan disalin sebanyak dua kali di bagian dinding lain pada citra ini.



Gambar 5.19 19_street_building dan Kunci Jawaban

20. 20_book.png

Pada Gambar 5.20 bagian yang disalin adalah objek buku pada rak lalu ditempelkan sebanyak satu kali di bagian lain pada citra ini.

Gambar 5.20 20_book dan Kunci Jawaban

Adapun detail dari 20 buah citra tersebut ditunjukan pada Tabel 5.2.

Tabel 5.2 Detail Data Citra

No	Nama	Dimensi Citra	Ukuran Citra
1	01_giraffe.png	267 x 400	0.228 MB
2	02_view.png	342 x 512	0.294 MB
3	03_clean_wall.png	377 x 512	0.358 MB
4	04_cattle.png	427 x 640	0.523 MB
5	05_history.png	512 x 512	0.788 MB
6	06_three_hundred.png	512 x 512	1.049 MB
7	07_tree.png	512 x 512	0.576 MB
8	08_Coin.png	512 x 512	0.425 MB
9	09_fountain.png	512 x 512	0.42 MB

No	Nama	Dimensi Citra	Ukuran Citra
10	10_park.png	512 x 512	0.556 MB
11	11_barrier.png	512 x 512	0.395 MB
12	12_cattle_remake.png	512 x 512	0.415 MB
13	13_clean_walls_remake.png	512 x 512	0.525 MB
14	14_bird.png	512 x 512	0.558 MB
15	15_four_babies.png	512 x 512	0.457 MB
16	16_ship.png	512 x 512	0.79 MB
17	17_tree_flower.png	512 x 512	0.625 MB
18	18_flower.png	512 x 512	0.551 MB
19	19_Street_building.png	512 x 512	0.416 MB
20	20_Book.png	512 x 512	0.467 MB

5.2.2 *Threshold Config*

Threshold config merupakan data masukan berupa parameter yang dibutuhkan oleh program sebagai nilai batasan untuk menjalankan metode *expanding block algorihtm* seperti *pVal*, *numBuckets*, *blockSize*, dan *minArea*. Tabel 5.3 Contoh dari *threshold config*

Tabel 5.3 Contoh *Threshold Config*

Parameter	Nilai
pVal	0.5
numBuckets	2048
blockSize	16
minArea	100

5.3 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 filter* dan *median filter* yang telah dijelaskan pada bab sebelumnya.

5.4 Skenario Uji Coba

Skenario uji coba diperlukan untuk menguji kebenaran yang diusulkan. Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, program akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario dan membandingkan skenario manakah yang memiliki hasil yang lebih baik.

5.4.1 Skenario Uji Coba Paramater

Skenario Uji coba parameter dilakukan untuk mencari nilai paramter yang terbaik yang akan digunakan. Terdapat empat parameter yang akan diteliti yaitu *numBuckets*, *pVal*, *blockSize* dan *minArea*. Adapun citra yang digunakan yaitu citra nomor 1 – 10 yang dapat dilihat pada Tabel 5.2.

5.4.1.1 Skenario Uji Coba Parameter *numBuckets*

Skenario uji coba parameter *numBuckets* yaitu menghitung nilai *MSE* dan *Similarity* dengan mencoba nilai *numBuckets* 1024, 2048, 4096, 8192, dan 12000 dengan menetapkan nilai *pVal* 0.99, *blockSize* 16 dan *minArea* 100 bertujuan untuk menetapkan parameter *numBuckets* yang terbaik.

Tabel 5.4 merupakan tabel rata-rata *MSE* dari hasil uji coba parameter *numBuckets*. Dari tabel ini nilai *MSE* terbesar yaitu ketika menggunakan nilai 1024 untuk *numBuckets* sedangkan *MSE* terkecil ketika menggunakan nilai 12000 untuk *numbuckets*.

Tabel 5.4 Rata-Rata *MSE* Uji Coba Parameter *numBuckets*

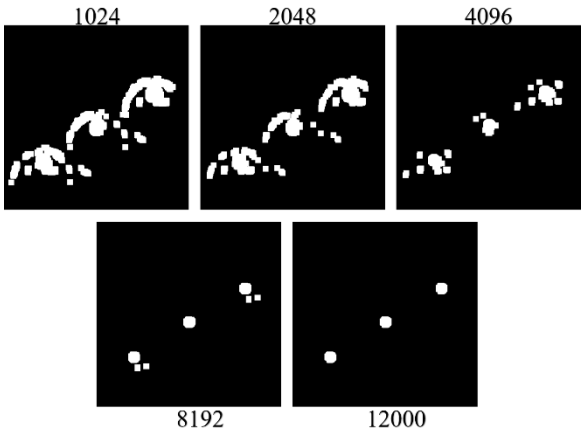
numBuckets	1024	2048	4096	8192	12000
MSE	3769.1	3136.9	2262.7	1742.6	1652.9

Tabel 5.5 merupakan tabel rata-rata *similarity* dari hasil uji coba parameter *numbuckets*, dari tabel ini nilai *similarity* terbesar yaitu ketika menggunakan nilai 12000 untuk *numBuckets* sedangkan similarity terkecil ketika menggunakan nilai 1024 untuk *numbuckets*.

Tabel 5.5 Rata-Rata *Similarity* Uji Coba Parameter *numBuckets*

numBuckets	1024	2048	4096	8192	12000
Similarity	88.99%	89.25%	89.70%	89.96%	90.01%

Gambar 5.21 merupakan salah satu contoh hasil keluaran hasil uji coba parameter *numBuckets*. Hasil yang paling sesuai dengan kunci jawaban yaitu bagian citra keluaran dengan nilai *numBuckets* 12000.



Gambar 5.21 Contoh Hasil Keluaran Uji Coba Parameter *numBuckets*

5.4.1.2 Skenario Uji Coba Parameter *pVal*

Skenario uji coba parameter *pVal* yaitu menghitung nilai *MSE* dan *Similarity* dengan mencoba nilai *pVal* 0.50, 0.75, 0.90,

0.95 dan 0.99 dengan menetapkan nilai *numBuckets* 12000, *blockSize* 16 dan *minArea* 100 bertujuan untuk menetapkan parameter *pVal* yang terbaik.

Tabel 5.6 merupakan tabel rata-rata nilai *MSE* dari hasil uji coba parameter *pVal*. Dari tabel ini perbedaan nilai *MSE* untuk masing-masing nilai *pVal* tidak terlalu signifikan.

Tabel 5.6 Rata-Rata *MSE* Uji Coba Parameter *pVal*

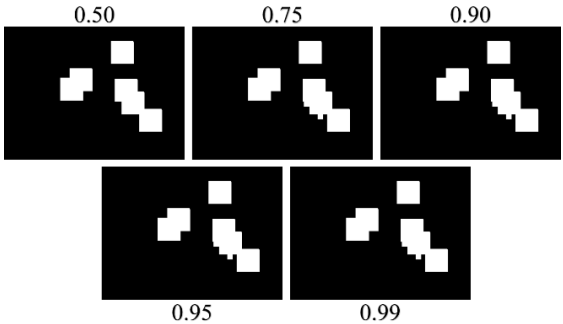
pVal	0.50	0.75	0.90	0.95	0.99
MSE	1539.89	1592.35	1610.44	1652.48	1652.48

Tabel 5.7 merupakan tabel rata-rata *similarity* dari hasil uji coba parameter *pVal*. Sama halnya seperti rata-rata *MSE* pada uji coba parameter *pVal*, perbedaan *similarity* untuk masing-masing nilai *pVal* tidak terlalu signifikan.

Tabel 5.7 Rata-Rata *Similarity* Uji Coba Parameter *pVal*

pVal	0.50	0.75	0.90	0.95	0.99
Similiarity	99.17%	99.14%	99.13%	99.11%	99.11%

Gambar 5.22 merupakan salah satu contoh hasil keluaran hasil uji coba parameter *pVal*. Perbedaan dari contoh hasil keluaran uji coba parameter *pVal* tidak terlalu signifikan.



Gambar 5.22 Contoh Hasil Keluaran Uji Coba Parameter *pVal*

5.4.1.3 Skenario Uji Coba Parameter *blockSize*

Skenario uji coba parameter *numBuckets* yaitu menghitung *MSE* dan *Similarity* dengan mencoba nilai *blockSize* 4, 8, 16, 32 dan 64 dengan menetapkan nilai *numBuckets* 12000, *pVal* 0.50 dan *minArea* 100 bertujuan untuk menetapkan parameter *blockSize* yang terbaik.

Tabel 5.8 merupakan tabel rata-rata nilai *MSE* dari hasil uji coba parameter *blockSize*. Dari tabel ini nilai *MSE* yang terkecil yaitu ketika menggunakan nilai 16 untuk *blockSize*. Ketika menggunkan nilai yang kecil atau yang besar dari 16 untuk parameter *blockSize*, maka akan mendapatkan nilai *MSE* yang besar.

Tabel 5.8 Rata-Rata *MSE* Uji Coba Parameter *blockSize*

blockSize	4	8	16	32	64
MSE	14626.25	2080.71	1539.89	2667.98	7545.69

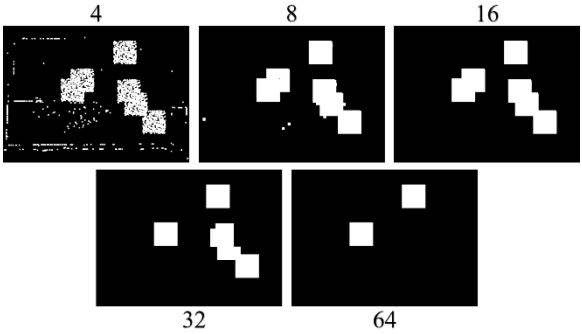
Tabel 5.9 merupakan tabel rata-rata *similarity* dari hasil uji coba parameter *blockSize*. Sama halnya seperti *MSE* pada uji coba parameter *blockSize*, nilai *similarity* terbaik yaitu ketika menggunakan nilai 16 untuk *blockSize*.

Tabel 5.9 Rata-Rata *Similarity* Uji Coba Parameter *blockSize*

blockSize	4	8	16	32	64
Similiarity	92.48%	98.89%	99.17%	98.59	96.13%

Gambar 5.23 merupakan salah satu contoh hasil keluaran uji coba parameter *blockSize*. Dapat dilihat dari gambar, ketika menggunakan nilai 4 untuk *blockSize* hasil keluaran akan mendeteksi bagian yang seharusnya tidak terjadi *copy-move*, sedangkan ketika menggunakan nila 64 untuk *blockSize* hasil keluaran tidak mendeteksi bagian yang seharusnya terjadi *copy-move*. Nilai 8 dan 32 menghasilkan hasil deteksi yang hampir sesuai dengan kunci jawaban, tetapi masih ada bagian yang tidak

seharusnya *copy-move* terdeteksi. Nilai 16 untuk *blockSize* menghasilkan hasil deteksi yang paling sesuai dengan kunci jawaban.



Gambar 5.23 Contoh Hasil Keluaran Uji Coba Parameter *blockSize*

5.4.1.4 Skenario Uji Coba Parameter *minArea*

Skenario uji coba parameter *minArea* yaitu menghitung *MSE* dan *Similarity* dengan mencoba nilai *minArea* 60, 70, 80, 90 dan 100 dengan menetapkan nilai *numBuckets* 12000, *pVal* 0.5 dan *blockSize* 16 bertujuan untuk menetapkan parameter *minArea* yang terbaik.

Tabel 5.10 merupakan tabel rata-rata nilai *MSE* dari hasil uji coba parameter *minArea*. Dari tabel ini perbedaan nilai *MSE* untuk masing-masing nilai *minArea* tidak terlalu signifikan.

Tabel 5.10 Rata-Rata *MSE* Uji Coba Parameter *minArea*

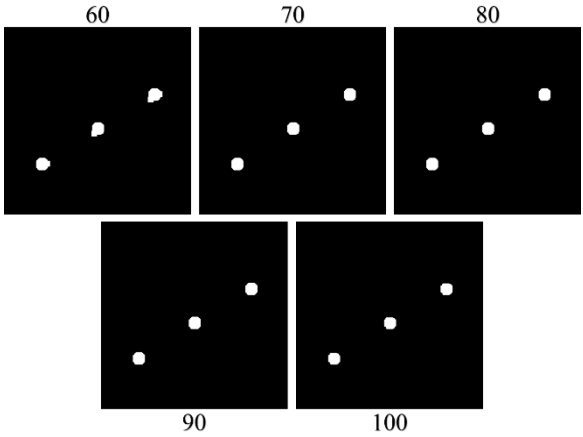
minArea	60	70	80	90	100
MSE	1518.04	1480.59	1480.59	1480.20	1539.89

Tabel 5.11 merupakan tabel rata-rata *similarity* dari hasil uji coba parameter *minArea*. Sama halnya seperti rata-rata *MSE* pada uji coba parameter *minArea*, perbedaan *similarity* untuk masing-masing nilai *pVal* tidak terlalu signifikan.

Tabel 5.11 Rata-Rata *Similarity* Uji Coba parameter *minArea*

minArea	60	70	80	90	100
Similarity	99.18%	99.20%	99.20%	99.20%	99.17%

Gambar 5.24 merupakan salah satu contoh hasil keluaran hasil uji coba parameter *minArea*. Perbedaan dari contoh hasil keluaran uji coba parameter *minArea* tidak terlalu signifikan.

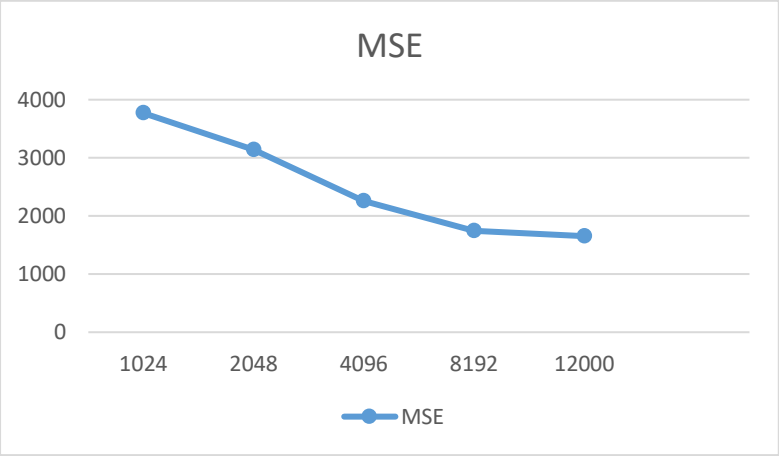


Gambar 5.24 Contoh Hasil Keluaran Uji Coba Parameter *minArea*

5.4.2 Evaluasi Uji Coba Paramater

Evaluasi dari hasil uji coba parameter adalah nilai yang paling berpengaruh adalah *numBuckets*. Hal ini dikarenakan *numBuckets* merupakan nilai yang digunakan untuk membangun banyak *groups* dan banyak *buckets* dimana *groups* dan *buckets* ini berisi *block-block* yang telah diurutkan berdasarkan *dominant feature*. Dengan semakin banyak *buckets* maka *block* yang memiliki nilai *dominant feature* yang mirip akan berdekatan. Melainkan ketika *buckets* sedikit, maka *buckets* akan berisi *block* dengan nilai *dominant feature* yang berdeda jauh. Gambar 5.25

merupakan grafik rata-rata *MSE* hasil uji coba parameter *numBuckets*, dari grafik dapat dilihat bahwa semakin besar nilai *numBuckets* maka akan semakin kecil nilai *MSE*. Sehingga nilai *numBuckets* yang dipilih adalah 12000.

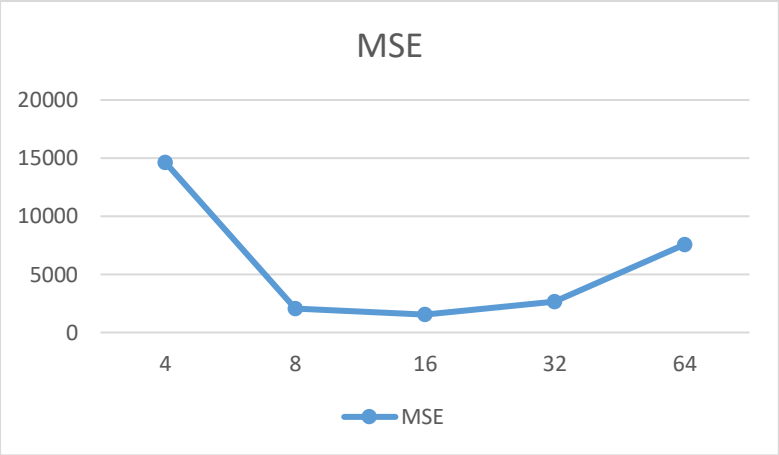


Gambar 5.25 Grafik Rata-Rata *MSE* Uji Coba Parameter *numBuckets*

Hasil uji coba untuk parameter *pVal* yang dilakukan tidak terlalu berpengaruh, dapat dilihat pada Tabel 5.6 dan Tabel 5.7. nilai *MSE* dan *Similarity* untuk masing-masing percobaan tidak terlalu berbeda. Hal ini dikarenakan parameter *pVal* merupakan nilai anatara 0 dan 1 yang digunakan sebagain batasan probabilitas untuk perbandingan antara dua *block*. Sehingga nilai *pVal* yang dipilih adalah 0.50.

Hasil uji coba untuk parameter *blockSize* yang dilakukan adalah dengan memilih nilai 16 untuk parameter *blockSize*. Parameter *blockSize* adalah nilai yang digunakan untuk membangun ukuran *block*, sebuah *block* yang dibangun berukuran *blockSize x blockSize*. Gambar 5.26 merupakan grafik rata-rata *MSE* hasi uji coba parameter *blockSize*, dari grafik dapat dilihat bahwa nilai *MSE* terkecil yaitu ketika parameter untuk *blockSize*

16, ketika menggunkan nilai yang kecil atau yang besar dari 16 untuk parameter *blockSize*, maka akan mendapatkan nilai *MSE* yang lebih besar. Sehingga nilai *blockSize* yang dipilih adalah 16.



Gambar 5.26 Grafik Rata-Rata *MSE* Uji Coba Parameter *blockSize*

Sama halnya dengan hasil uji coba parameter *pVal*, hasil uji coba untuk parameter *minArea* yang dilakukan tidak terlalu berpengaruh, dapat dilihat pada Tabel 5.10 dan Tabel 5.11. nilai *MSE* dan *Similarity* untuk masing-masing percobaan tidak terlalu berbeda. Hal ini dikarenakan parameter *minArea* merupakan nilai yang menunjukan daerah minimum dari bagian yang di duplikat. Sehingga nilai *minArea* yang dipilih adalah 90.

5.4.3 Skenario Uji Coba Performa

Setelah mendapatkan nilai parameter terbaik untuk metode *expanding block algorithm* yang telah dilakukan melalui uji coba paramater yaitu *numBuckets* 12000, *pVal* 0.50, *blockSize* 16 dan *minArea* 90. Uji coba yang dilakukan selanjutnya yaitu uji coba performa. Sebelum melakukan uji coba performa perlu ditentukan

skenario yang akan digunakan dalam uji coba performa. Melalaui skenario ini, sistem akan diuji apakah sistem yang dibangun sudah berjalan dengan benar dan membandingkan skenario manakah yang memiliki hasil yang terbaik. Terdapat 12 macam skenario uji coba performa, yaitu:

1. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dan tanpa *filter*.
2. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dengan *averaging filter*.
3. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dengan *median filter*.
4. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dan tanpa *filter*.
5. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dengan *averaging filter*.
6. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dengan *median filter*.
7. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dan tanpa *filter*.
8. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dengan *averaging filter*.
9. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dengan *median filter*.
10. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dan tanpa *filter*.

11. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dengan *averaging filter*.
12. Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dengan *median filter*.

5.4.3.1 Skenario Uji Coba Performa 1

Skenario uji coba performa 1 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dan tanpa *filter*. Hasil *MSE*, *similarity* dan akurasi pada uji coba performa 1 dapat dilihat pada Tabel 5.12.

Tabel 5.12 *Confusion Matrix* Skenario Uji Coba Performa 1

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	1448.45	99.26%	100%
2	5	0	0	0	6503.84	96.67%	100%
3	5	0	0	0	3912.13	97.99%	100%
4	5	0	0	0	643.87	99.67%	100%
5	5	0	0	0	209.11	99.89%	100%
6	5	0	0	0	415.24	99.79%	100%
7	1	0	0	0	647.41	99.67%	100%
8	2	0	0	0	157.02	99.92%	100%
9	1	0	0	0	415.54	99.38%	100%
10	1	0	0	0	449.47	99.77%	100%
11	1	0	0	0	526.69	99.42%	100%
12	1	0	0	0	344.06	99.48%	100%
13	1	0	0	0	323.21	99.51%	100%
14	1	0	0	0	706.94	99.64%	100%
15	1	8	0	0	2921.46	98.14%	11.11%

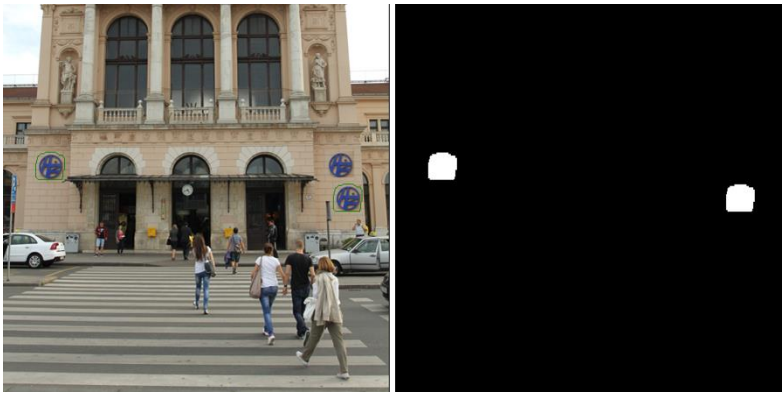
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
16	2	0	0	0	365.02	99.51%	100%
17	2	0	0	0	1963.07	98.99%	100%
18	1	0	0	0	1802.34	99.08%	100%
19	1	0	0	1	1263.57	99.35%	50%
20	1	0	0	0	400.35	99.79%	100%
	Rata-Rata MSE				1270.94		
	Rata-Rata Similarity					99.25%	
	Rata-Rata Akurasi						93.06%

Dari Tabel 5.12 didapatkan rata-rata *MSE* sebesar 1270.94, rata-rata *similarity* sebesar 99.25% dan rata-rata akurasi sebesar 93.06%. Dilihat dari nilai rata-rata *MSE*, *similarity* dan akurasi didapatkan bahwa uji coba performa 1 menghasilkan hasil pendeteksian yang optimal.

Gambar 5.27 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 1 ditandai dengan garis bewarna hijau pada objek yang terkena serangan *copy-move*. Program dapat mendeteksi dengan benar dua objek yang terjadi *copy-move* yaitu objek nomor yang disalin sebanyak satu kali serta jendela yang disalin sebanyak satu kali yang ada pada citra.

Gambar 5.28 merupakan contoh citra hasil deteksi dengan akurasi 50% pada uji coba performa 1. Pada citra ini terdapat objek yang disalin yaitu objek logo bewarna biru yang ada di dinding gedung. Objek logo tersebut aslinya hanya ada satu di bagian kiri gedung setelah terjadi serangan *copy-move* objek tersebut disalin sebanyak dua kali di bagian kanan gedung. Pada citra ini, program hanya dapat mendeteksi satu logo yaitu bagian bawah sebelah kanan dinding gedung, sedangkan logo bagian atas sebelah kanan dinding gedung program tidak dapat mendeteksi bahwa itu adalah bagian *copy-move*

Gambar 5.27 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 1



Gambar 5.28 Contoh Citra dengan Akurasi 50% Hasil Uji Coba performa 1

5.4.3.2 Skenario Uji Coba Performa 2

Skenario uji coba performa 2 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dengan *averaging*

filter. *MSE*, *similarity* dan akurasi pada uji coba performa 2 dapat dilihat pada Tabel 5.13.

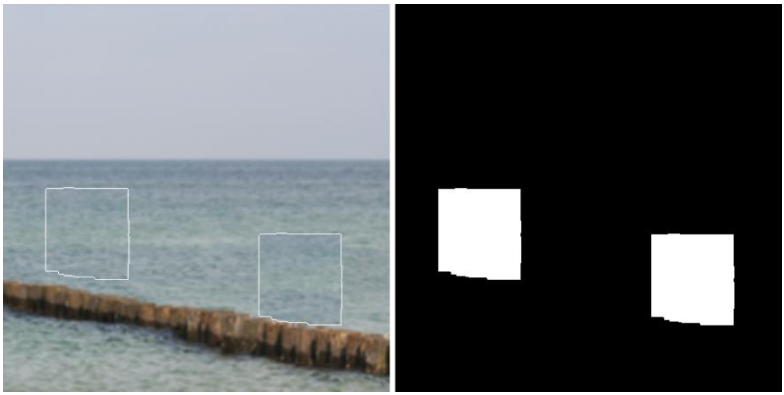
Tabel 5.13 *Confusion Matrix* Skenario Uji Coba Performa 2

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	3775.47	98.06%	100%
2	5	0	0	0	7429.61	96.19%	100%
3	5	0	0	0	5011.69	97.43%	100%
4	5	0	0	0	1950.89	99.00%	100%
5	5	0	0	0	602.76	99.69%	100%
6	5	0	0	0	1168.32	99.40%	100%
7	1	0	0	0	1564.21	99.20%	100%
8	2	4	0	0	1199.57	99.39%	33.33%
9	1	0	0	0	1161.53	99.39%	100%
10	1	2	0	0	1303.01	99.33%	33.33%
11	1	0	0	0	976.04	99.41%	100%
12	1	0	0	0	971.74	99.48%	100%
13	1	0	0	0	903.47	99.51%	100%
14	1	1	0	0	1907.26	99.02%	50%
15	1	15	0	0	9993.99	94.79%	6.67%
16	2	2	0	0	2181.77	98.53%	50%
17	2	0	0	0	2928.24	98.50%	100%
18	1	0	0	0	4015.44	97.94%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	1522.54	99.22%	100%
	Rata-Rata MSE				2676.84		
	Rata-Rata Similarity					98.60%	
	Rata-Rata Akurasi						78.67%

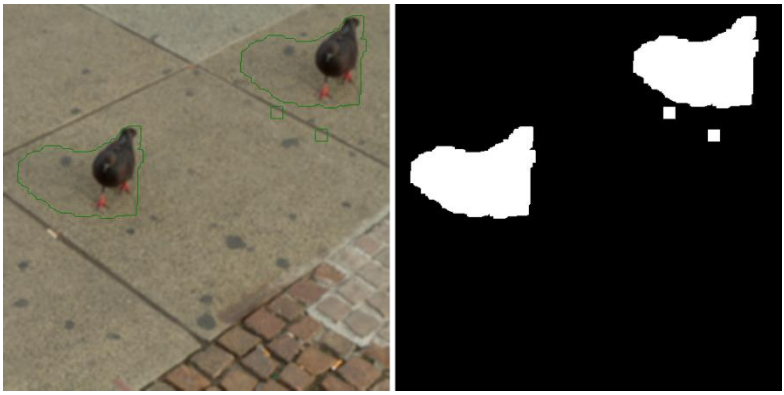
Dari Tabel 5.13 didapatkan rata-rata *MSE* sebesar 2676.84, rata-rata *similarity* sebesar 99.25% dan rata-rata akurasi sebesar 78.67%. Dilihat nilai rata-rata *MSE*, *similarity* dan akurasi hasil uji coba performa 2 kurang mendapatkan hasil yang optimal, yaitu dengan terdapat banyak hasil deteksi dari program yang tidak tepat.

Gambar 5.29 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 2 ditandai dengan garis putih pada objek yang terkena serangan *copy-move*. Program berhasil mendeteksi dengan benar bagian yang terjadi serangan *copy-move* yaitu bagian air laut yang disalin dibagian lain pada citra.

Gambar 5.30 merupakan contoh citra hasil deteksi dengan akurasi 50% pada uji coba performa 2. Program telah berhasil mendeteksi objek pada citra yang terkena serangan *copy-move* yaitu objek burung pada citra, tetapi terdapat objek yang tidak *copy-move* terdeteksi yaitu kotak kecil yang ada dibagian kanan atas.



Gambar 5.29 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 2



Gambar 5.30 Contoh Citra dengan Akurasi 50% Hasil Uji Coba Performa 2

5.4.3.3 Skenario Uji Coba Performa 3

Skenario uji coba performa 3 adalah Perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image gray dominant feature* dengan *median filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 3 dapat dilihat pada Tabel 5.14.

Tabel 5.14 *Confusion Matrix* Skenario Uji Coba Performa 3

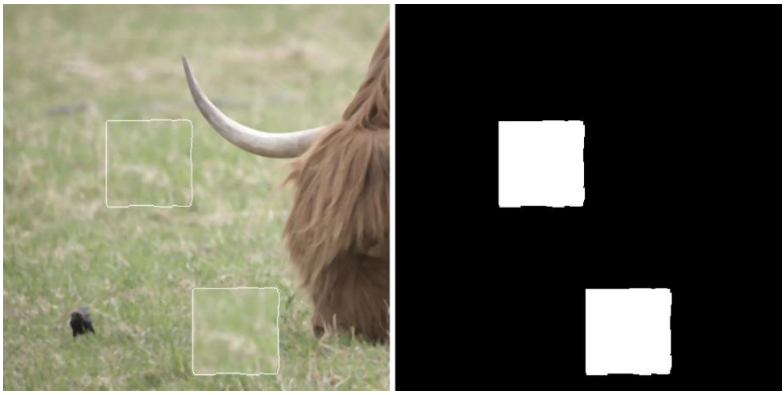
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	3035.72	98.44%	100%
2	5	0	0	0	7216.83	96.30%	100%
3	5	0	0	0	4947.01	97.46%	100%
4	5	0	0	0	944.39	99.52%	100%
5	5	0	0	0	602.76	99.69%	100%
6	5	0	0	0	755.31	99.61%	100%
7	1	0	0	0	1193.62	99.39%	100%
8	2	0	0	0	409.28	99.79%	100%

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
9	1	0	0	0	927.87	99.43%	100%
10	1	1	0	0	1319.38	99.32%	50%
11	1	0	0	0	988.94	99.41%	100%
12	1	0	0	0	820.32	99.46%	100%
13	1	0	0	0	926.29	99.51%	100%
14	1	0	0	0	1503.19	99.23%	100%
15	1	10	0	0	7232.11	96.20%	10%
16	2	0	0	0	1229.96	99.02%	100%
17	2	0	0	0	2175.9	98.88%	100%
18	1	0	0	0	2826.29	98.55%	100%
19	1	0	0	1	2091.07	98.93%	50%
20	1	0	0	0	1378.17	99.29%	100%
	Rata-Rata MSE				2126.22		
	Rata-Rata Similarity					98.87%	
	Rata-Rata Akurasi						90.50%

Dari Tabel 5.14 didapatkan rata-rata *MSE* sebesar 2126.22, rata-rata *similarity* sebesar 98.87% dan rata-rata akurasi sebesar 90.50%. Dilihat dari nilai rata-rata *MSE*, *similarity* dan akurasi hasil uji coba performa 3 mendapatkan hasil yang optimal tetapi masih ada hasil pengujian dengan akurasi yang rendah yaitu dengan akurasi 10% pada citra ke 15.

Gambar 5.31 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 3 ditandai dengan garis putih. Program berhasil mendeteksi dengan baik bagian yang terkena serangan *copy-move*

Gambar 5.32 merupakan contoh hasil deteksi dengan akurasi 50% pada uji coba performa 3, program berhasil mendeteksi bagian yang terjadi *copy-move* tetapi terdapat satu kesalahan deteksi yaitu bagian kecil yang ada dibawah citra.



Gambar 5.31 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 3

Gambar 5.32 Contoh Citra dengan Akurasi 50% Hasil Uji Coba Performa 3

5.4.3.4 Skenario Uji Coba Performa 4

Skenario uji coba performa 4 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dan tanpa *filter*. *MSE*,

similarity dan akurasi pada uji coba performa 4 dapat dilihat pada Tabel 5.15.

Tabel 5.15 *Confusion Matrix* Skenario Uji Coba Performa 4

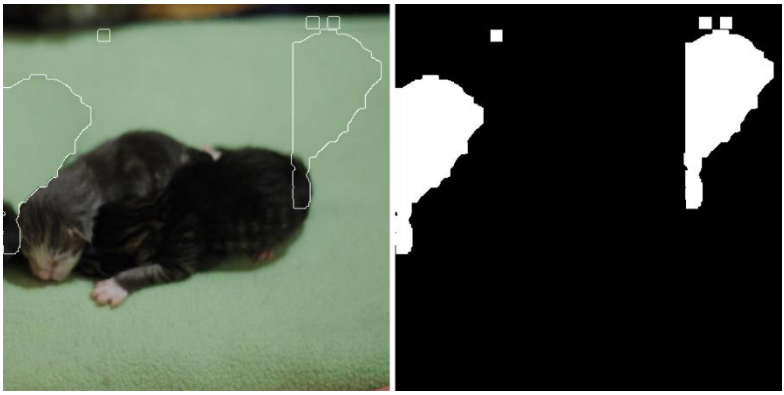
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	1428.36	99.27%	100%
2	5	0	0	0	6503.84	96.67%	100%
3	5	0	0	0	3902.03	98.00%	100%
4	5	0	0	0	575.35	99.71%	100%
5	5	0	0	0	209.11	99.89%	100%
6	5	0	0	0	415.24	99.79%	100%
7	1	0	0	0	684.62	99.65%	100%
8	2	0	0	0	192.74	99.90%	100%
9	1	0	0	0	415.54	99.38%	100%
10	1	0	0	0	404.82	99.79%	100%
11	1	0	0	0	531.18	99.41%	100%
12	1	0	0	0	342.57	99.48%	100%
13	1	0	0	0	323.21	99.51%	100%
14	1	0	0	0	747.13	99.62%	100%
15	1	2	0	0	1477.09	98.87%	33.33%
16	2	1	0	0	690.1	99.35%	66.67%
17	2	0	0	0	1873.03	99.04%	100%
18	1	0	0	0	1830.61	99.06%	100%
19	1	0	0	1	1507.65	99.23%	50%
20	1	0	0	0	410.77	99.79%	100%
	Rata-Rata MSE				1223.25		
	Rata-Rata Similarity					99.27%	
	Rata-Rata Akurasi						92.50%

Dari Tabel 5.15 didapatkan rata-rata *MSE* sebesar 1223.25, rata-rata *similarity* sebesar 99.27% dan rata-rata akurasi sebesar 92.50%. Dilihat dari nilai rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 4 mendapatkan hasil yang optimal. Uji coba performa 4 mempunyai nilai rata-rata *MSE* yang kecil dan nilai rata-rata *similarity* yang besar diantari uji coba performa yang lainya. Akurasinya yang terendah pada uji coba performa 4 yaitu dengan akurasi 33.33% pada citra ke 15.

Gambar 5.33 merupakan salah satu contoh citra *copy-move* yang behasil dideteksi dengan akurasi 100% pada uji coba performa 4 ditandai dengan garis putih pada bagian yang terkena serangan *copy-move*. Program berhasil mendeteksi dengan baik bagian yang terkena *copy-move* yaitu objek langit yang ada pada citra.

Gambar 5.34 merupakan contoh hasil deteksi dengan akurasi 33.33% pada uji coba performa 4. Program telah berhasil mendeteksi bagian yang terkena *copy-move* pada citra tetapi terdapat bagian yang seharusnya tidak *copy-move* terdeteksi, yaitu 3 kotak kecil yang ada pada bagian atas citra. Bagian

Gambar 5.33 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 4



Gambar 5.34 Contoh Citra dengan Akurasi 33.33% Hasil Uji Coba Performa 4

5.4.3.5 Skenario Uji Coba Performa 5

Skenario uji coba performa 5 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dengan *averaging filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 5 dapat dilihat pada Tabel 5.16.

Tabel 5.16 *Confusion Matrix* Skenario Uji Coba Performa 5

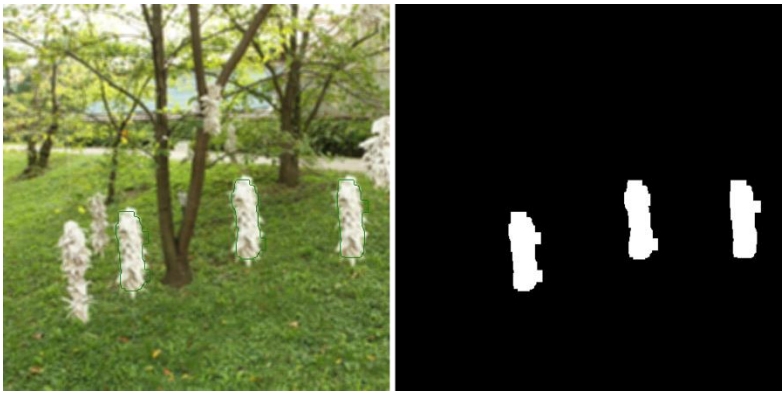
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	3391.89	98.26%	100%
2	5	0	0	0	7419.59	96.20%	100%
3	5	0	0	0	5029.88	97.42%	100%
4	5	0	0	0	1549.01	99.21%	100%
5	5	0	0	0	602.76	99.69%	100%
6	5	0	0	0	1168.32	99.40%	100%
7	1	0	0	0	1537.42	99.21%	100%
8	2	1	0	0	634.76	99.67%	66.67%

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
9	1	0	0	0	1158.67	99.39%	100%
10	1	1	0	0	1052.23	99.46%	50%
11	1	0	0	0	983.01	99.42%	100%
12	1	0	0	0	968.83	99.48%	100%
13	1	0	0	0	881.09	99.52%	100%
14	1	1	0	0	1910.98	99.02%	50%
15	1	16	0	0	10102.26	94.74%	5.88%
16	2	1	0	0	1755.73	98.74%	66.67%
17	2	0	0	0	2922.29	98.50%	100%
18	1	0	0	0	3970.8	97.96%	100%
19	1	0	0	1	2336.64	98.80%	50%
20	1	0	0	0	1452.58	99.26%	100%
	Rata-Rata MSE				2541.44		
	Rata-Rata Similarity					98.67%	
	Rata-Rata Akurasi						84.46%

Dari Tabel 5.16 didapatkan rata-rata *MSE* sebesar 2541.44, rata-rata *similarity* sebesar 98.67% dan rata-rata akurasi sebesar 84.46%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 5 kurang mendapatkan hasil yang optimal yaitu dengan banyak hasil deteksi yang tidak tepat.

Gambar 5.35 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 5, objek yang disalin yaitu objek bunga yang ada pada citra yang ditandai dengan garis bewarna hijau.

Gambar 5.36 merupakan contoh hasil deteksi dengan akurasi 66.67% pada uji coba performa 5. Program telah berhasil mendeteksi bagian yang terkena *copy-move* pada citra tetaoi terdapat bagian yang seharusnya tidak *copy-move* terdeteksi yaitu objek kotak kecil yang ada pada citra.



Gambar 5.35 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 5

Gambar 5.36 Contoh Citra dengan Akurasi 66.67% Hasil Uji Coba Performa 5

5.4.3.6 Skenario Uji Coba Performa 6

Skenario uji coba performa 6 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move* asli menggunakan *image color dominant feature* dengan *median filter*.

MSE, similarity dan akurasi pada uji coba performa 6 dapat dilihat pada Tabel 5.17.

Tabel 5.17 *Consusion Matrix* Skenario Uji Coba Performa 6

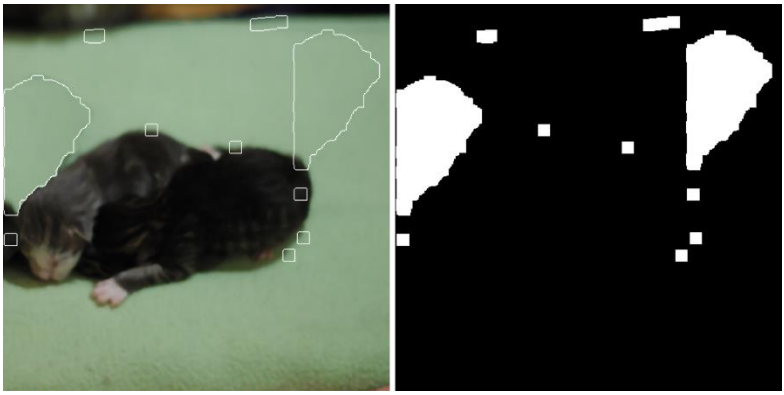
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	3850.36	98.03%	100%
2	5	0	0	0	7327.12	96.24%	100%
3	5	0	0	0	4892.44	97.49%	100%
4	5	0	0	0	1156.4	99.41%	100%
5	5	0	0	0	602.76	99.69%	100%
6	5	0	0	0	876.61	99.55%	100%
7	1	0	0	0	1320.13	99.32%	100%
8	2	0	0	0	225.48	99.88%	100%
9	1	0	0	0	986.22	99.41%	100%
10	1	1	0	0	1678.81	99.14%	50%
11	1	0	0	0	959.17	99.43%	100%
12	1	0	0	0	907.57	99.44%	100%
13	1	0	0	0	876.74	99.52%	100%
14	1	0	0	0	1517.33	99.22%	100%
15	1	8	0	0	4416.88	97.64%	11.11%
16	2	0	0	0	1291.28	98.98%	100%
17	2	0	0	0	2364.92	98.79%	100%
18	1	0	0	0	2939.4	98.49%	100%
19	1	0	0	1	2424.45	98.76%	50%
20	1	0	0	0	1360.31	99.30%	100%
	Rata-rata MSE				2098.72		
	Rata-Rata Similarity					98.89%	
	Rata-Rata Akurasi						90.56%

Dari Tabel 5.17 didapatkan rata-rata *MSE* sebesar 1098.72, rata-rata *similarity* sebesar 98.89% dan rata-rata akurasi sebesar 90.56%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 6 sudah optimal mendapatkan hasil yang optimal, tetapi masih terdapat ketidaktepatan pendeteksian pada beberapa citra. Akurasi yang terendah pada uji coba performa 6 yaitu dengan akurasi 11.11% pada citra ke 15.

Gambar 5.37 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 6 ditandai dengan garis putih pada bagian yang terkena serangan *copy-move*. Program berhasi mendeteksi dengan baik bagian yang terkena *copy-move* yaitu bagian dinding yang ada pada citra.

Gambar 5.38 merupakan contoh hasil deteksi dengan akurasi 11.11% pada uji coba performa 6. Program telah berhasil mendeteksi bagian yang terkena *copy-move* pada citra tetapi terdapat banyak bagian yang seharusnya tidak *copy-move* terdeteksi, yaitu terdapat banyak kotak kecil pada citra.

Gambar 5.37 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 6



Gambar 5.38 Contoh Citra dengan Akurasi 11.11% Hasil Uji Coba Performa 6

5.4.3.7 Skenario Uji Coba Performa 7

Skenario uji coba performa 7 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dan tanpa *filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 7 dapat dilihat pada Tabel 5.18.

Tabel 5.18 *Confusion Matrix* Skenario Uji Coba Performa 7

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	2993.71	98.47%	100%
2	5	0	0	0	13213.77	93.23%	100%
3	5	0	0	0	11775.81	93.96%	100%
4	5	0	0	0	6228.87	96.81%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	5	0	0	0	6995.03	96.41%	100%
7	1	0	0	0	15845.97	91.88%	100%
8	2	0	0	0	685.36	99.65%	100%

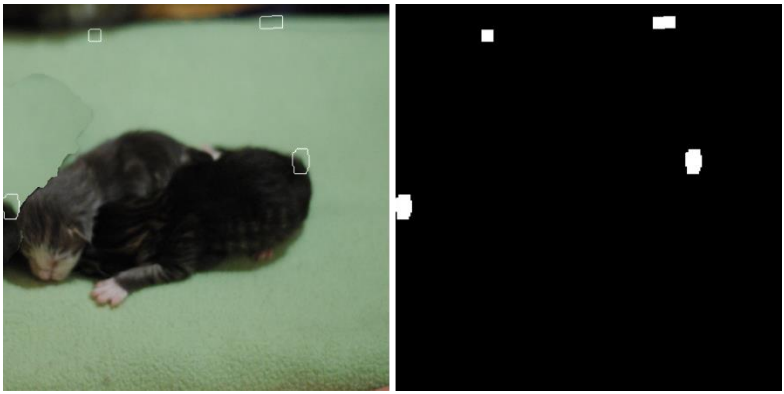
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
9	0	0	0	1	16369.02	91.59%	0%
10	1	0	0	0	2119.35	98.91%	100%
11	1	0	0	0	16143.99	91.69%	100%
12	0	0	0	1	19656.47	89.91%	0%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	15892.11	91.85%	100%
15	0	3	0	1	22614.24	88.36%	0%
16	1	0	0	1	8105.38	95.48%	50%
17	1	0	0	1	9077.17	95.35%	50%
18	1	0	0	0	15636.12	91.98%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	5825.22	97.01%	100%
	Rata-Rata MSE				10548.42		
	Rata-Rata Similarity					94.57%	
	Rata-Rata Akurasi						65.00%

Dari Tabel 5.18 didapatkan nilai rata-rata *MSE* sebesar 10548.42, rata-rata *similarity* sebesar 94.57% dan rata-rata akusari sebesar 65%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 7 sangat tidak optimal. Banyak terdapat hasil deteksi dengan akurasi 0%

Gambar 5.39 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 7 ditandai dengan garis hijau pada bagian yang terkena serangan *copy-move*. Program berhasil mendeteksi bagian yang terkena serangan *copy-move* yaitu objek telinga hewan pada citra.

Gambar 5.40 merupakan contoh hasil deteksi dengan akurasi 0%. Program tidak dapat mendeteksi bagian yang merupakan *copy-move* tetapi program mendeteksi bagian lain yang bukan bagian *copy-move*.

Gambar 5.39 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 7



Gambar 5.40 Contoh Citra dengan Akurasi 0% Hasil uji Coba Performa 7

5.4.3.8 Skenario Uji Coba Performa 8

Skenario uji coba performa 8 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dengan *averaging filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 8 dapat dilihat pada Tabel 5.19.

Tabel 5.19 *Consusion Matrix* Skenario Uji Coba Performa 8

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	6533.55	96.65%	100%
2	5	0	0	0	15718.16	91.94%	100%
3	5	0	0	0	17792.06	90.88%	100%
4	5	0	0	0	6236	96.80%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	4	1	0	0	8327.06	95.73%	80%
7	0	0	0	1	17670.64	90.94%	0%
8	2	1	0	0	1268.78	99.35%	66.67%
9	0	0	0	1	16369.02	91.59%	0%
10	1	1	0	0	2186.32	98.88%	50%
11	1	0	0	0	16234.28	91.66%	100%
12	0	0	0	1	19656.47	89.91%	0%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	13339.67	93.16%	100%
15	0	16	0	0	28156.34	85.52%	0%
16	1	0	0	1	7220.56	95.94%	50%
17	0	0	0	2	9688.86	95.03%	0%
18	1	0	0	0	15116.71	92.25%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	8980.43	95.40%	100%
	Rata-Rata MSE				11614.29		
	Rata-Rata Similarity					94.02%	
	Rata-Rata Akurasi						52.33%

Dari Tabel 5.19 didapatkan rata-rata *MSE* sebesar 116114.29, rata-rata *similarity* sebesar 94.02% dan rata-rata akurasi sebesar 52.33%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 8 sangat tidak optimal.

Gambar 5.41 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 8, tetapi hasil deteksi bentuknya tidak sempurna.

Gambar 5.42 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 50%. Program hanya dapat mendeteksi satu diantara dua objek yang terjadi *copy-move.*

Gambar 5.41 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 8

Gambar 5.42 Contoh Citra dengan Akurasi 50% Hasil Uji Coba Performa 8

5.4.3.9 Skenario Uji Coba Performa 9

Skenario uji coba performa 9 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image gray dominant feature* dengan *median filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 9 dapat dilihat pada Tabel 5.20.

Tabel 5.20 *Consusion Matrix* Skenario Uji Coba Performa 9

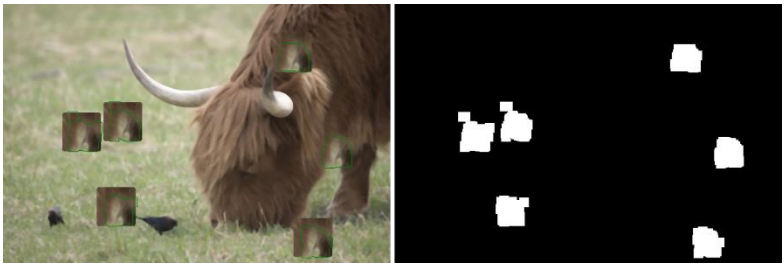
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	7304.35	96.26%	100%
2	0	0	0	5	18905.47	90.31%	0%
3	0	0	0	5	18657.16	90.44%	0%
4	5	0	0	0	7791.44	96.01%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	0	0	0	5	9429.9	95.17%	0%
7	0	0	0	1	17670.64	90.94%	0%
8	2	0	0	1	1008.33	99.48%	66.67%
9	0	0	0	2	16369.02	91.59%	0%
10	1	0	0	0	3210.27	98.35%	100%
11	0	0	0	1	19856.81	89.81%	0%
12	0	0	0	1	19656.47	89.91%	0%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	17385.62	91.09%	100%
15	0	16	0	0	26503.08	86.38%	0%
16	1	0	0	1	10289.45	94.36%	50%
17	2	0	0	0	9045.91	95.36%	100%
18	1	0	0	0	21321.45	89.07%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	35570.47	81.77%	100%
	Rata-Rata MSE				14088.33		

	Rata-Rata Similarity		92.76%	
	Rata-Rata Akurasi			40.83%

Dari Tabel 5.20 didapatkan rata-rata *MSE* sebesar 14088.33, rata-rata *similarity* sebesar 92.76% dan rata-rata akurasi sebesar 40.83%. Dilihat dari rata-rata *MSE, similarity* dan akurasi, uji coba performa 9 sangat tidak optimal. Banyak terdapat citra yang tidak dapat dideteksi atau banyak hasil deteksi yang memiliki akurasi 0%. Karena banyaknya citra yang tidak dapat dideteksi atau banyak hasil deteksi yang memiliki akurasi 0% menyebabkan rata-rata akurasi dari uji coba performa 9 menjadi kecil.

Gambar 5.43 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 9, tetapi banya hasil deteksi bentuknya tidak sempurna pada uji coba performa 9. Program berhasil mendeteksi bagian mana yang terjadi *copy-move* tetapi dibagian hasil deteksi tersebut terdapat bagian yang tidak sempurna atau bagian yang keluar dari bagian yang seharusnya *copy-move*

Gambar 5.44 merupakan salah satu contoh citra dengan akurasi 0%. Dapat dilihat dari gambar bahwa tidak ada satu bagian yang bisa terdeteksi yang seharusnya terdapat 5 bagian *copy-move* pada citra.



Gambar 5.43 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 9

Gambar 5.44 Contoh Citra dengan Akurasi 0% Hasil Uji Coba Performa 9

5.4.3.10 Skenario Uji Coba Performa 10

Skenario uji coba performa 10 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dan tanpa *filter*. *MSE*, *similarity* dan akurasi pada uji coba performa 10 dapat dilihat pada Tabel 5.21.

Tabel 5.21 *Consusion Matrix* Skenario Uji Coba Performa 10

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	2215.6	98.86%	100%
2	5	0	0	0	10761.74	94.48%	100%
3	5	0	0	0	7768.68	96.02%	100%
4	5	0	0	0	3467.78	98.22%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	5	0	0	0	4770.76	97.55%	100%
7	1	0	0	0	14728.26	92.45%	100%
8	2	0	0	0	450.21	99.77%	100%

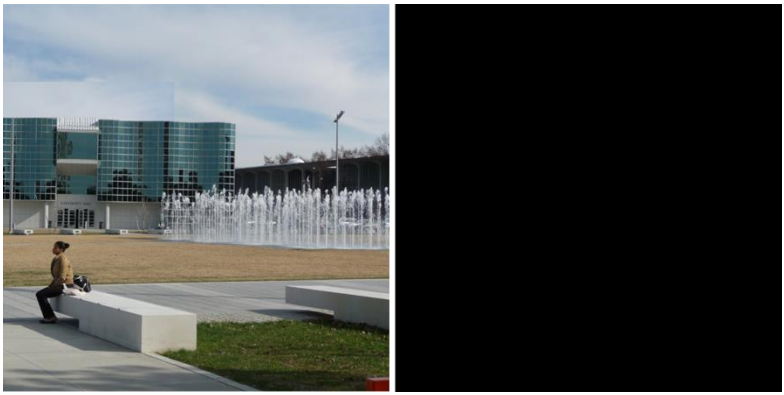
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
9	0	0	0	1	16369.02	91.59%	0%
10	1	0	0	0	3159.67	98.38%	100%
11	1	0	0	0	16205.91	91.66%	100%
12	1	0	0	0	18513.45	90.50%	100%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	15411.39	92.10%	100%
15	0	1	0	0	22258.25	88.55%	0%
16	2	0	0	0	5866.06	96.63%	100%
17	2	0	0	0	8100.1	95.85%	100%
18	1	0	0	0	12236.84	93.73%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	4797.55	97.54%	100%
	Rata-Rata MSE				9443.61		
	Rata-Rata Similarity					95.14%	
	Rata-Rata Akurasi						75.00%

Dari Tabel 5.21 didapatkan rata-rata *MSE* sebesar 9443.61, rata-rata *similarity* sebesar 95.14% dan rata-rata akurasi sebesar 75%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 10 kurang optimal dikarenakan masih terdapatnya hasil deteksi dengan akurasi 0%.

Gambar 5.45 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 10. Program dapat mendeteksi dengan baik bagian yang terjadi *copy-move* yang ditandai dengan garis bewarna hijau.

Gambar 5.46 merupakan salah satu contoh citra copy move dengan akurasi 0% pada uji coba performa 10. Program tidak dapat mendeteksi bagian yang terjadi *copy-move* yang seharusnya terdapat satu bagian *copy-move* yaitu di langit pada citra.

Gambar 5.45 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 10



Gambar 5.46 Contoh Citra dengan Akurasi 0% Hasil Uji Coba Performa 10

5.4.3.11 Skenario Uji Coba Performa 11

Skenario uji coba performa 11 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dengan *averaging*

filter. *MSE*, *similarity* dan akurasi pada uji coba performa 11 dapat dilihat pada Tabel 5.22.

Tabel 5.22 *Consusion Matrix* Skenario Uji Coba Performa 11

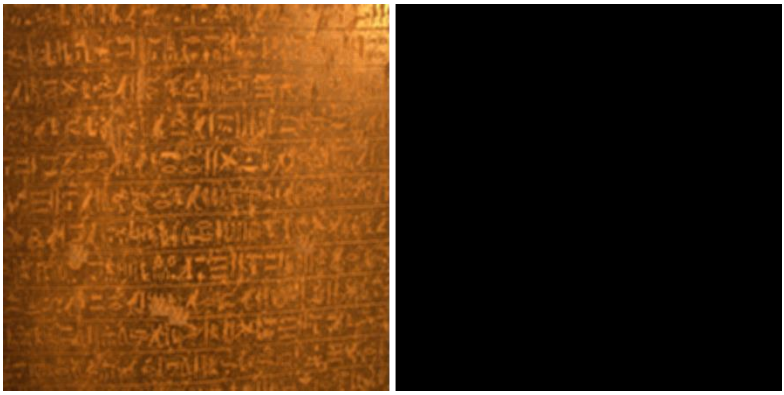
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	5388.31	97.24%	100%
2	5	0	0	0	16254.02	91.67%	100%
3	5	0	0	0	17113.93	91.23%	100%
4	5	0	0	0	5671.37	97.09%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	0	0	0	5	9429.9	95.17%	0%
7	1	0	0	0	15896.58	91.85%	100%
8	3	8	0	0	2499.61	98.72%	27.28%
9	0	0	0	1	16369.02	91.59%	0%
10	1	0	0	0	2687.88	98.62%	100%
11	1	0	0	0	16368.23	91.60%	100%
12	0	0	0	1	19656.47	89.91%	0%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	12003.92	93.85%	100%
15	0	16	0	0	28910.62	85.14%	0%
16	2	0	0	0	5145.12	97.00%	100%
17	2	0	0	0	6919.13	96.45%	100%
18	1	0	0	0	16530.59	91.53%	100%
19	0	0	0	1	2969.17	98.48%	0%
20	1	0	0	0	6455.52	96.69%	100%
	Rata-Rata MSE				11254.55		
	Rata-Rata Similarity					94.21%	
	Rata-Rata Akurasi						61.36%

Dari Tabel 5.22 didapatkan rata-rata *MSE* sebesar 11254.55, rata-rata *similarity* sebesar 94.21% dan rata-rata akurasi sebesar 61.36%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 11 kurang mendapatkan hasil yang optimal dikarenakan masih terdapatnya hasil deteksi dengan akurasi 0%. Dikarenakan banyak hasil deteksi dengan akurasi 0% pada uji coba performa 11 menyebabkan rata-rata akurasi uji coba performa 11 menjadi kecil.

Gambar 5.47 merupakan salah satu contoh citra *copy-move* yang berhasil dideteksi dengan akurasi 100% pada uji coba performa 11. Program dapat mendeteksi bagian yang terjadi *copy-move* yaitu objek mobil bewana hitam yang ada di citra yang ditandai dengan garis bewarna hijau.

Gambar 5.48 merupakan salah satu contoh citra *copy-move* dengan akurasi 0% pada uji coba performa 11. Program tidak dapat mendeteksi bagian yang terjadi *copy-move* yang seharusnya terdapat 5 bagian yang terjadi *copy-move* pada citra.

Gambar 5.47 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 11



Gambar 5.48 Contoh Citra dengan Akurasi 0% Hasil Uji Coba Performa 11

5.4.3.12 Skenario Uji Coba Performa 12

Skenario uji coba performa 12 adalah perhitungan nilai *MSE*, *similarity* dan akurasi pada citra *copy-move noise* menggunakan *image color dominant feature* dengan *median filter*. *MSE, similarity* dan akurasi pada uji coba performa 12 dapat dilihat pada Tabel 5.23.

Tabel 5.23 *Consusion Matrix* Skenario Uji Coba Performa 12

Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
1	4	0	0	0	6661.41	96.59%	100%
2	0	0	0	5	18905.47	90.31%	0%
3	5	0	0	0	17495.95	91.03%	100%
4	5	0	0	0	7594.42	96.11%	100%
5	0	0	0	5	2571.79	98.68%	0%
6	0	0	0	5	9429.9	95.17%	0%
7	1	0	0	0	16527.62	91.53%	100%
8	2	0	0	0	1176.5	99.40%	100%

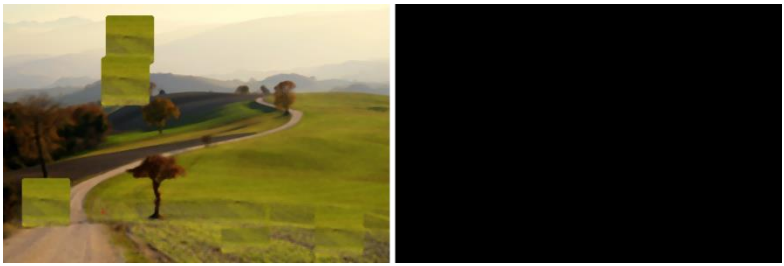
Citra -Ke	TP	FP	TN	FN	MSE	Similarity	Akurasi
9	0	0	0	1	16369.02	91.59%	0%
10	1	0	0	0	2778.66	98.58%	100%
11	0	0	0	1	19856.81	89.81%	0%
12	1	0	0	0	18513.45	90.50%	100%
13	0	0	0	1	16249.87	91.66%	0%
14	1	0	0	0	16650.4	91.46%	100%
15	0	8	0	0	23806.33	87.76%	0%
16	1	0	0	1	10118.29	94.45%	50%
17	0	0	0	2	9688.86	95.03%	0%
18	1	0	0	0	20050.43	89.72%	100%
19	0	0	0	2	2969.17	98.48%	0%
20	1	0	0	0	27148.16	86.08%	100%
	Rata-Rata MSE				13228.13		
	Rata-Rata Similarity					93.20%	
	Rata-Rata Akurasi						52.50%

Dari Tabel 5.23 didapatkan rata-rata *MSE* sebesar 13228.13, rata-rata *similarity* sebesar 93.20% dan rata-rata akurasi sebesar 52.50%. Dilihat dari rata-rata *MSE*, *similarity* dan akurasi, uji coba performa 12 kurang sanga tidak optimal.

Gambar 5.49 merupakan salah satu contoh citra *copy-move* dengan akurasi 100% pada uji coba performa 12. Program dapat mendeteksi bagian yang terjadi *copy-move* yaitu objek burung yang ada di citra.

Gambar 5.50 merupakan salah satu contoh citra *copy-move* dengan akurasi 0%. Program tidak dapat mendeteksi bagian yang terjadi *copy-move* yang seharusnya terdapat 5 bagian yang terjadi *copy-move* di citra.

Gambar 5.49 Contoh Citra dengan Akurasi 100% Hasil Uji Coba Performa 12



Gambar 5.50 Contoh Citra dengan Akurasi 0% Hasil Uji Coba Performa 12

5.4.4 Evaluasi Uji Coba Performa

Terdapat dua pembahasan untuk evaluasi uji coba performa.

1. Citra *Copy-Move* Asli

Tabel 5.24 merupakan rata-rata hasil *MSE, Similarity* dan Akurasi dari uji coba perfroma menggunakan *image gray dominant feature* dan Tabel 5.25 merupakan rata-rata hasil *MSE, Similarity* dan *Akurasi* dari uji coba performa menggunakan *image color*

dominant feature pada citra *copy-move* asli. Dari kedua tabel dapat diketahui bahwa penggunaan *image gray dominant feature* dan *image color dominant feature* perbedaanya tidak terlalu signifikan karena sama-sama menghasilkan *MSE* yang rendah, *similarity* yang tinggi dan akurasi yang tinggi. Dari tabel juga dapat dilihat bawhwa penggunaan *median filter* lebih baik daripada *averaging filter* untuk citra *copy-move* asli tetapi penggunaan *filter* akan mengurangi performa dari pendeteksian *copy-move*.

Tabel 5.24 Rata-Rata Hasil Uji Coba Performa Menggunakan *Image Gray Dominant Feature* pada Citra *Copy-Move* Asli

Uji Coba Performa	Rata-Rata MSE	Rata-Rata Similarity	Rata-Rata Akurasi	Filter
1	1270.94	99.25%	93.06%	Tanpa Filter
2	2676.84	98.60%	78.67%	Averaging
3	2126.22	98.87%	90.50%	Median

Tabel 5.25 Rata-Rata Hasil Uji Coba Performa Menggunakan *Image Color Dominant Feature* pada Citra *Copy-Move* Asli

Uji Coba Performa	Rata-Rata MSE	Rata-Rata Similarity	Rata-Rata Akurasi	Filter
4	1223.25	99.27%	92.50%	Tanpa Filter
5	2541.44	98.67%	84.46%	Averaging
6	2098.72	98.89%	90.56%	Median

2. Citra *Copy-Move Noise*

Tabel 5.26 merupakan rata-rata hasil *MSE, Similarity* dan Akurasi dari uji coba perfroma menggunakan *image gray dominant feature* dan Tabel 5.27 merupakan rata-rata hasil *MSE, Similarity* dan *Akurasi* dari uji cobar performa menggunakan *image color*

dominant feature pada citra *copy-move noise.* Dari kedua tabel dapat dikethaui bahwa pengguna *image color dominant feature* akan menghasilkan hasil yang lebih optimal dari pada menggunakan *image gray dominant feature* untuk pendeteksian *copy-move* citra *noise* dengan *uniform distribution*. Dari tabel juga dapat dilihat bahwa penggunaan median filter lebih baik dari pada *averaging filter* untuk citra *copy-move noise* tetapi penggunaan *filter* akan mengurangi performa dai pendeteksian *copy-move*.

Tabel 5.26 Rata-Rata Hasil Uji Coba Performa Menggunakan *Image Gray Dominant Feature* pada Citra *Copy-Move Noise*

Uji Coba Performa	Rata-Rata MSE	Rata-Rata Similarity	Rata-Rata Akurasi	Filter
7	10548.42	94.57%	65.00%	Tanpa Filter
8	11614.29	94.02%	52.33%	Averaging
9	14088.33	92.76%	40.83%	Median

Tabel 5.27 Rata-Rata Hasil Uji Coba Performa Menggunakan *Image Color Dominant Feature* pada Citra *Copy-Move Noise*

Uji Coba Performa	Rata-Rata MSE	Rata-Rata Similarity	Rata-Rata Akurasi	Filter
10	9443.61	95.14%	75.00%	Tanpa Filter
11	11254.55	94.21%	61.36%	Averaging
12	13228.13	93.20%	52.50%	Median

5.5 Evaluasi Umum Skenario Uji Coba

Dari uji coba parameter didapatkan bahwa parameter yang paling berpengaruh adalah *numBuckets* dikarenakan *numBuckets* adalah nilai yang digunakan untuk pembuatan *bucket*. Untuk nilai parameter yang telah ditetapkan adalah *numBuckets* 12000, *pVal*

0.5, *blockSize* 16 dan *minArea* 90 dapat menghasilkan hasil deteksi yang optimal. Pada citra *copy-move* asli penggunaa *image gray dominant feature* dan penggunaan *image color dominant feature* sama-sama menghasilkan hasil yang optimal sedangkan pada citra *copy-move noise uniform distribution* penggunaan *image color dominant feature* akan lebih baik dibandingkan menggunakan *image gray dominant feature*. Penggunaan *filter* akan mengurangi performa pendeteksian *copy-move.*

Dari hasil uji coba performa akurasi tertinggi yang didapat adalah 93.06%. sedangkan *MSE* terendah yang didapat adalah 1223.25 dan *Similarity* tertinggi yang didapat adalah 99.27%.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

6.1 Kesimpulan

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

1. Penerapan metode *expanding block algorithm* dapat digunakan sebagai salah satu metode untuk melakukan pendeteksian serangan *copy-move* pada citra.
2. Penerapan *image color dominant feature* dapat digunakan untuk mencari *dominant feature* pada citra yang terkena serangan *copy-move*.
3. Metode *expanding block algorithm* baik digunakan untuk mendeteksi *copy-move* di dalam citra pada objek yang besar maupun yang kecil dan objek yang memiliki banyak bentuk.
4. Akurasi tertinggi yang didapat dari skenario uji coba performa adalah 93.06% dengan nilai parameter *numBuckets* 12000, *pVal* 0.50, *blockSize* 16 dan *minArea* 90.
5. *MSE* terendah yang didapat dari skenario uji coba performa adalah 1223.25 denga nilai parameter *numBuckets* 12000, *pVal* 0.50, *blockSize* 16 dan *minArea* 90.
6. *Similarity* tertinggi yang didapat dari skenario uji coba performa adalah 99.27% dengan nilai parameter *numBuckets* 12000, *pVal* 0.50, *blockSize* 16 dan *minArea* 90
7. Metode *expanding block algorithm* dapat digunakan untuk mendeteksi *copy-move* pada citra yang terkena *noise uniform*

distribution sebesar 0.5% dengan menggunakan *image color dominant feature*.

6.2 Saran

Berikut meruapakan beberapa saran untuk pengembangan perangkat lunak di masa yang akan datang, saran diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Meningkatkan ketepatan pendeteksian serangan *copy-move* pada citra yang memiliki banyak kemiripan warna piksel.
2. Dapat mencoba suatu metode untuk dapat mendeteksi bagian *copy-move* pada citra yang mengalami *rotasi*, *resize dan rescale*.

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] "Continuum," *Continuum*. [Online]. Available: https://www.continuum.io/. [Accessed: 15-Dec-2016].

[8] "PyCharm," *JetBrains*. [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed: 14-Dec-2016].

[9] "Pengolahan Citra Digital - Chapter 5." [Online]. Available: http://viplab.if.its.ac.id/pcd_online/Chapter%205.html. [Accessed: 15-Dec-2016].

[10] "Welcome to Python.org," *Python.org*. [Online]. Available: https://www.python.org/. [Accessed: 14-Dec-2016].

[11] "OpenCV | OpenCV." [Online]. Available: http://opencv.org/. [Accessed: 01-Jan-2017].

[12] "matplotlib: python plotting — Matplotlib 1.5.3 documentation." [Online]. Available: http://matplotlib.org/. [Accessed: 01-Jan-2017].
[13] "Numpy and Scipy Documentation — Numpy and Scipy documentation." [Online]. Available: https://docs.scipy.org/doc/. [Accessed: 15-Dec-2016].
[14] "PyQt/Tutorials - Python Wiki." [Online]. Available: https://wiki.python.org/moin/PyQt/Tutorials. [Accessed: 15-Dec-2016].
[15] "JSON." [Online]. Available: http://www.json.org/. [Accessed: 01-Jan-2017].
[16] E. W. Weisstein, "Uniform Distribution." [Online]. Available: http://mathworld.wolfram.com/UniformDistribution.html. [Accessed: 16-Dec-2016].
[17] "OpenCV: Canny Edge Detection." [Online]. Available: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html. [Accessed: 03-Jan-2017].
[18] "Image Manipulation Dataset." [Online]. Available: https://www5.cs.fau.de/research/data/image-manipulation/. [Accessed: 16-Jan-2017].
[19] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic, "CoMoFoD #x2014; New database for copy-move forgery detection," in *Proceedings ELMAR-2013*, 2013, pp. 49–54.
[20] "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].
[21] "Overview — Python 2.7.2 documentation." [Online]. Available: http://python.readthedocs.io/en/v2.7.2/. [Accessed: 15-Dec-2016].

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 Timur (2001-2007), SMPN 12 Padang (2007-2010), SMAN 10 Padang (2010-2013) dan S1 Teknik Informatika ITS (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi 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 juga terpilih sebagai mahasiswa yang mengikut pelatihan Huawei Certified Network Associate (HCNA) pada tahun 2016. Penulis dapat dihubungi melalui email: sudirahanif@gmail.com.