

**Laporan Tugas Multimedia Forensik
Deteksi Duplicate Region Menggunakan Robust-Duplication Detection
IF-42-GAB**



Disusun Oleh Kelompok 3:

1301180379	Adabi Raihan Muhammad
1301198497	Muhammad Faisal Amir
1301174524	Muhammad Irfan Aldi
1301184219	Sya Raihan Heggi
1301184005	Gia Nusantara

**S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2022**

DAFTAR ISI

BAB 1	2
1.1 Latar Belakang	2
1.2 Rumusan Masalah	3
1.3 Tujuan	3
1.4 Struktur Penulisan	3
BAB 2	4
2.1 Desain Sistem	4
2.2 Diagram Alir Pengerjaan	6
2.3 Dataset	10
2.3.1 Original Image	11
2.3.2 Added Image	12
2.3.2 Removed Image	13
2.3.4 Groundtruth Image	14
2.4 Pengerjaan Kode	15
BAB 3	17
3.1 Pengujian terhadap dataset	17
3.2 Skenario menambahkan noise	26
3.3 Skenario dilakukan kompresi	28
BAB 4	31
4.1 Kesimpulan	31
4.2 Saran	32
4.3 Tautan Terkait	32
DAFTAR PUSTAKA	33
LAMPIRAN	34
Lampiran Gambar	34
Lampiran Tabel	35

BAB 1

Pendahuluan

1.1 Latar Belakang

Kemajuan teknologi saat ini menyebabkan bermunculan perangkat-perangkat lunak yang berkaitan dengan citra, perangkat-perangkat lunak ini sangat mudah digunakan untuk melakukan manipulasi dari citra digital [1], selain dengan perangkat lunak yang berkembang begitu pesat, penggunaan kamera digital dan komputer juga meningkatkan angka dalam pemalsuan gambar [1], salah satu pemalsuan yang sering terjadi adalah pemalsuan duplikasi piksel terhadap suatu objek yang ada pada gambar [2], pemalsuan ini memiliki karakteristik memiliki nilai yang identik dari asal dan wilayah pemalsuan [2].

Namun peneliti sudah menemukan beberapa teknik untuk menyelesaikan atau mendeteksi gambar yang mengalami pemalsuan, diantaranya Farid dan Popescu mengajukan metode statistik dengan menggunakan PCA kemudian melakukan analisis terhadap wilayah yang terduplikasi, *color filter interpolation*, dan *resampling* [1], selanjutnya Fridrich yang mengajukan metode untuk mengajukan metode *copy-move* [1], selanjutnya metode yang dapat digunakan cukup beragam, untuk tugas besar ini akan merujuk pada hasil kerja Nazali et al dimana dikembangkan metode yang menggabungkan penelitian yang dilakukan oleh Popescu et al dan Luo et al.

Metode yang digunakan adalah melakukan perubahan gambar menjadi blok-blok tertentu, pembangunan blok ini sendiri akan menerapkan PCA untuk menghasilkan *small fixed block*, kemudian selagi membagi menjadi blok-blok tersebut didapatkan pula karakteristik dari wilayah tersebut, selanjutnya blok yang sudah terkumpul tersebut akan dilakukan pengurutan menggunakan metode *lexicographical sort*, selanjutnya akan dilakukan pengecekan terhadap blok yang identik, dan terakhir melakukan rekonstruksi wilayah blok mana yang terdeteksi, dengan menggunakan metode ini sendiri dikatakan dapat meningkatkan efisiensi dari kinerja sistem deteksi, kemudian meningkatkan *robustness* dan *sensitivity* jika terdapat noise dan kasus lossy JPEG.

1.2 Rumusan Masalah

Terdapat kejahatan dengan memanfaatkan pemalsuan terhadap sebuah gambar, pemalsuan ini sendiri dapat berupa melakukan penambahan atau mengurangi sebuah

objek pada citra, dengan menggunakan prinsip *duplicate region* atau melakukan duplikasi wilayah tertentu untuk dipindahkan ke lokasi lainnya, untuk menyelesaikan ini maka diajukan metode deteksi *robust detection*.

1.3 Tujuan

Tujuan dari tugas besar ini adalah melakukan uji coba algoritma yang diajukan, kemudian membuat sistem yang berbasis GUI untuk melakukan deteksi terhadap gambar, sistem yang dibangun harus melibatkan *lexicographical sort* dan *PCA*.

1.4 Struktur Penulisan

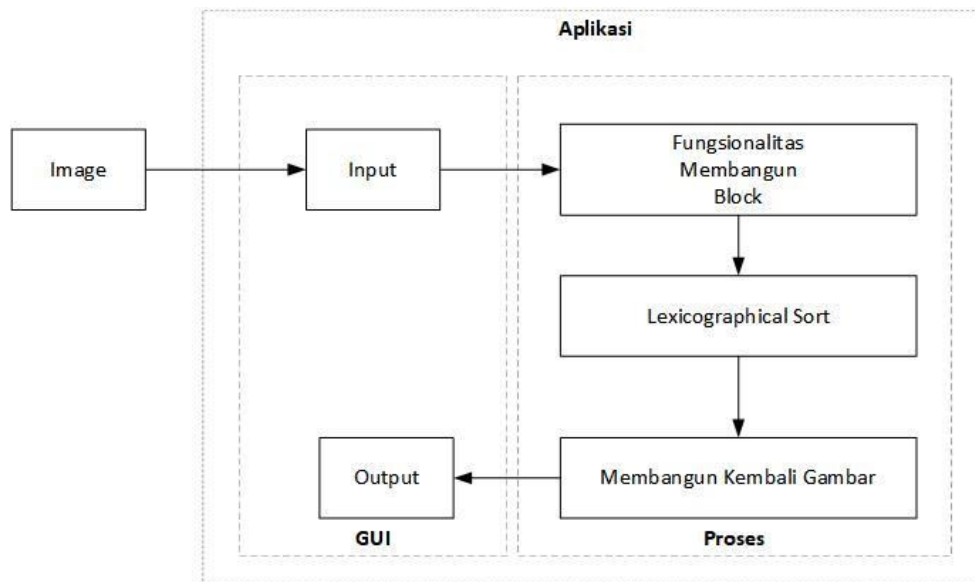
Dalam tugas besar ini memiliki struktur pengerjaan seperti berikut ini.

1. **BAB 1 PENDAHULUAN** Pada bagian ini akan dijelaskan latar belakang, rumusan masalah, dan tujuan dari tugas besar.
2. **BAB 2 KAJIAN TEORI DAN PERANCANGAN SISTEM** pada bagian ini akan dijelaskan secara rinci mengenai sistem yang dibangun, baik itu menggunakan diagram alir maupun diagram kerja sistem, kemudian dijelaskan juga metode-metode yang digunakan pada setiap tahapannya.
3. **BAB 3 HASIL UJI COBA** pada bagian ini akan dilampirkan hasil uji coba dan hasil perhitungan MSE, Similarity, dsb untuk menunjukkan kinerja dari sistem yang dibangun
4. **BAB 4 KESIMPULAN** pada bagian ini akan dijelaskan kesimpulan yang didapatkan dan menjelaskan kelemahan dan kelebihan dari sistem yang dibangun.

BAB 2

Kajian Teori dan Perencanaan Sistem

2.1 Desain Sistem



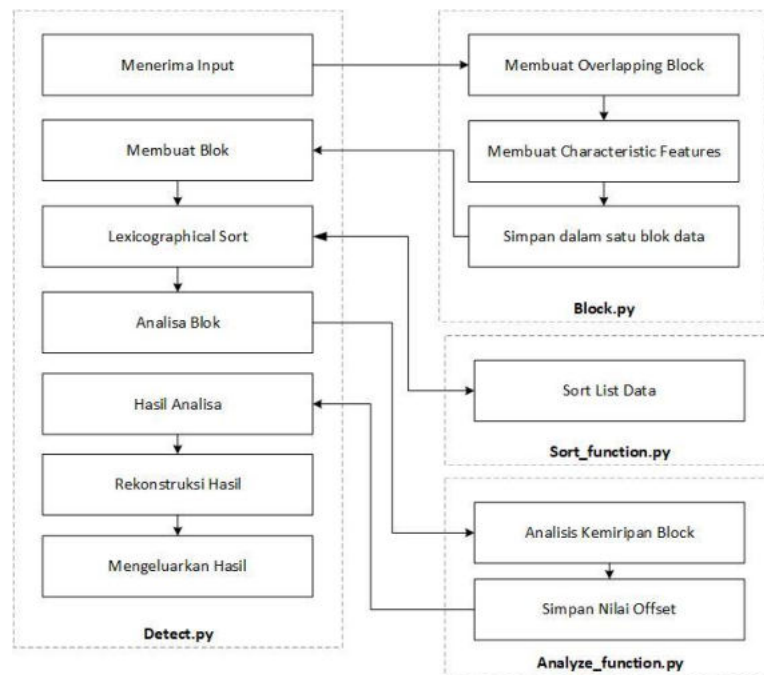
Gambar 1. Desain Sistem yang dibangun

Untuk sistem yang dibangun akan merujuk seperti pada Gambar 1, pada gambar tersebut terdapat beberapa objek yang dapat dijelaskan, **kotak garis tidak putus** merupakan sebuah proses atau objek yang dimasukan ke sistem, **kotak garis putus-putus** merupakan sebuah pengelompokkan dari suatu proses, dan terakhir adalah **panah** yang menggambarkan aliran pengerjaan dari satu dengan ke proses lainnya.

Untuk alur kerja akan diawali dengan sistem diberi masukan berupa gambar ke GUI yang sudah disiapkan, GUI akan mendapatkan *path* dari gambar dan kemudian akan diteruskan ke proses pembuatan blok, selanjutnya akan dilakukan *lexicographical sort* berdasarkan nilai blok dan *features characteristics*, selanjutnya akan dilakukan pengecekan terhadap blok yang memiliki kemiripan dengan menggunakan metode yang dijelaskan, dan kemudian akan dibangun kembali gambar hasil yang menunjukan lokasi yang terindikasi.

Dalam pengerjaan tugas ini akan digunakan bahasa pemrograman python sebagai sarana untuk mengembangkan sistem deteksi ini, akan terbagi menjadi beberapa modul diantaranya **main_ui.py** yang akan bertindak sebagai GUI dari sistem, **detect.py** yang

merupakan fungsionalitas utama untuk melakukan deteksi, **block.py** merupakan modul yang menangani dalam pembangun blok dalam hal ini melakukan pembuatan blok dan PCA, **sort.py** merupakan modul yang melakukan pengurutan terhadap data, **analyze.py** yang akan berfungsi untuk melakukan analisa terhadap blok yang didapatkan, untuk lebih jelasnya dapat dilihat pada Gambar 2.

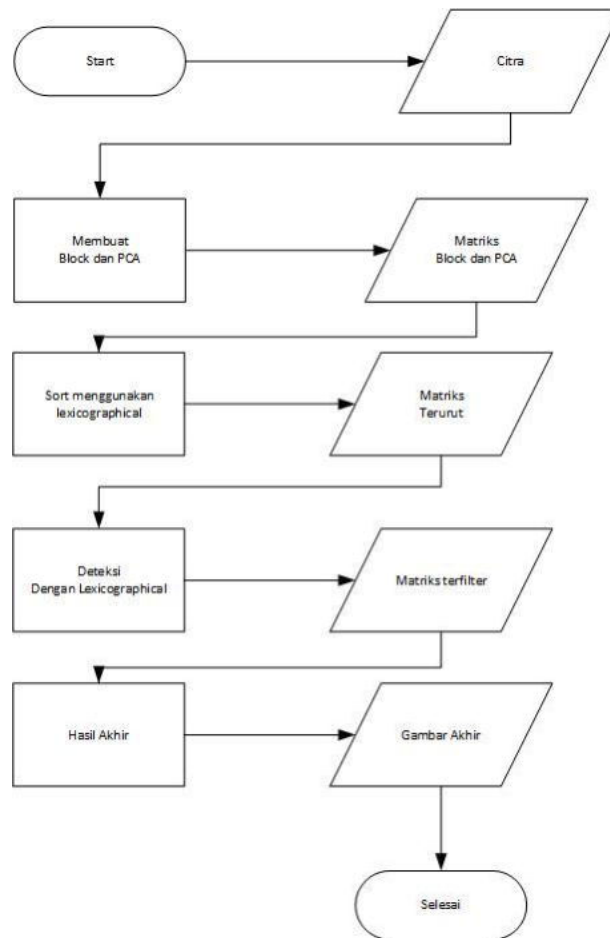


Gambar 2. Alur Pengerjaan dari Sistem

Untuk lebih rinci dan jelasnya bagaimana metode yang digunakan dapat dicek pada sub bab Diagram Alir Pengerjaan, dimana pada subbab tersebut akan dijelaskan metode dan cara kerja dari sistem yang dibangun sehingga dapat mengeluarkan hasil.

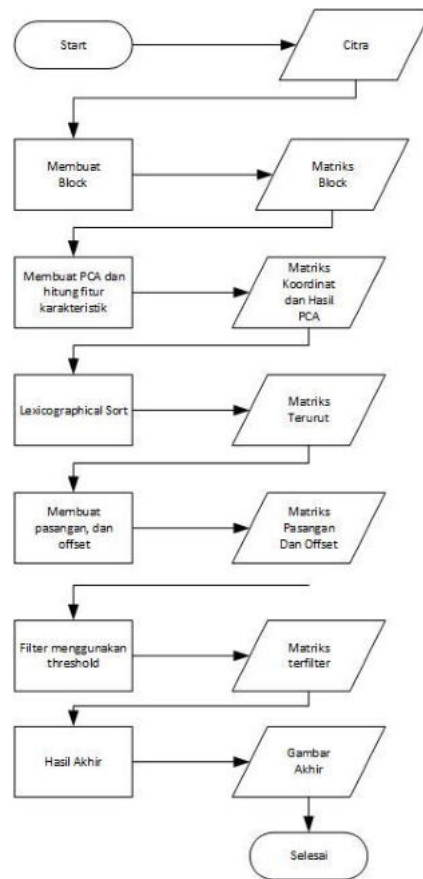
2.2 Diagram Alir Pengerjaan

Untuk Pembangunan Pengerjaan dari sistem ini sendiri secara sederhana dapat direpresentasikan dengan diagram pada Gambar 3.



Gambar 3. Diagram Alir Sederhana

Pada Gambar 3 metode yang menjadi inti dari penyelesaian permasalahan yang adalah dapat diurutkan menjadi *image*->**membuat blok dan penerapan PCA** -> *block matrix* -> *Lexicographical sort* -> *sorted matrix*-> **deteksi menggunakan fitur yang ada** -> *matrix offset/matrix terfilter* -> **Selesai**, sehingga dapat disimpulkan tahapan intinya di antara lain adalah membuat blok dengan penerapan PCA, kemudian melakukan *Lexicographical Sort*, dan terakhir mengecek kesamaan dari wilayah yang dideteksi, namun jika ingin melihat diagram lebih lengkap dapat melihat pada Gambar 4.



Gambar 4. Diagram Alir Lengkap

Metode yang akan digunakan dapat dijelaskan pada poin-poin berikut ini.

1. Membuat Blok dan PCA

```

# algorithm's parameters from the first paper
self.N = self.image_width * self.image_height
self.b = self.block_dimension * self.block_dimension

#popescu paper optimal reference
# self.b = 64

#luo et al Nb calculation formulas
# self.Nb = (self.image_width - self.block_dimension + 1) * (
#     self.image_height - self.block_dimension + 1
# )

#popescu et al Nb calculation formulas
self.Nb = (math.sqrt(self.N) - math.sqrt(self.b) + 1) ** 2
  
```

Gambar 5. Parameter Implementasi *Overlapping Block* dan PCA

Untuk membuat sistem lebih efisien maka perlu dibangun blok-blok dengan ukuran tertentu menggunakan PCA, PCA sendiri digunakan untuk melakukan pengurangan dimensi dari data, dalam hal ini sistem yang digunakan pada PCA adalah melakukan analisis statistik multivariat, dan kemudian nantinya akan mengurangi set variable data yang kurang berkorelasi, sehingga data yang digunakan lebih berkorelasi

Kemudian data blok ini akan dikelompokkan menjadi satu wilayah dengan ukuran tertentu untuk menentukan banyaknya jumlah *overlapping block* yang dapat digunakan dapat mengacu pada rumus yang dijelaskan oleh Luo et al (2.1) atau Popescu et al (2.2)..

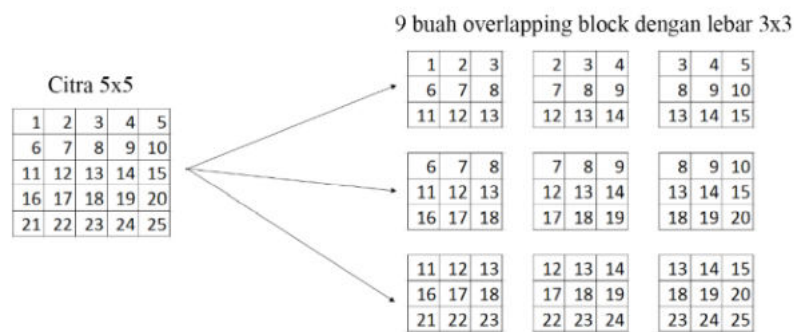
$$Nb = (Height - Dimension + 1) \times (Width - Dimension + 1) \quad (2.1)$$

$$Nb = (\sqrt{N \text{ pixels}} - \sqrt{Block} + 1)^2 \quad (2.2)$$

pada penugasan dituliskan keduanya dapat dipilih untuk menggunakan rumusan 2.1 atau rumus 2.2 hal ini dapat dilihat pada Gambar 5, kemudian untuk parameter block sendiri dapat dicapai seperti pada rumus (2.3), yang dapat dicapai dengan melakukan perkalian dengan dimensi block yang diinginkan.

$$Block = Block \text{ Dimension} \times Block \text{ Dimension} \quad (2.3)$$

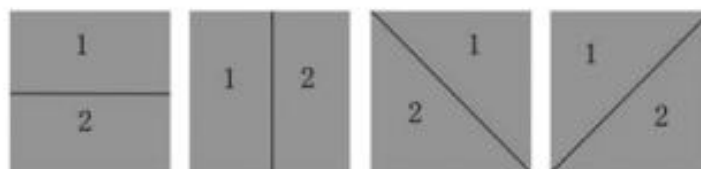
dan diakhir nantinya akan menghasilkan hasil seperti yang ditunjukkan pada Gambar 6



Gambar 7. Overlapping Block [3]

2. Ekstraksi Fitur

Ekstraksi fitur ini sendiri tidak dijelaskan pada rujukan penugasan namun metode ini didapatkan dari hasil percobaan yang dilakukan oleh Luo et al, pada hasilnya disarankan untuk mengeluarkan c1,c2,c3 dengan nilai dari rata-rata dari komponen R,G,B kemudian untuk c4,c5,c6,c7 dapat dicapai dengan mencari karakteristik dengan arah seperti pada Gambar 7.



Gambar 7. Arah untuk C4-C7 [1]

Bagaimana bila gambar awalnya berasal dari gambar berwarna hitam-putih maka cukup memberikan nilai 0 pada c1,c2,c3, setelah nilai ini didapatkan maka akan

dikembalikan dalam satu penyimpanan data dengan *overlapping block* untuk dilanjutkan ke proses berikutnya.

3. Lexicographical Sort

Pada tahapan ini cukup mudah, karena yang dilakukan adalah melakukan pengurutan terhadap data, untuk *lexicographical sort* sendiri berarti mengurutkan berdasarkan abjad oleh karena itu untuk implementasinya akan mengurutkan nilai a terlebih dahulu sebelum aa, dan x sebelum y [3], pada implementasi tugas besar ini akan diurutkan nilai blok dan karakternya yang dapat dijalankan seperti pada Gambar 8.

```
def sort_features(self):
    # melakukan sort list berdasarkan nilai characteristic dan pixel
    self.feature_list = sorted(self.feature_list, key=lambda x: (x[1], x[2]))
    return self.feature_list
```

Gambar 8. Melakukan Sort dengan parameter karakteristik fitur dan *overlapping block*

4. Analisis Wilayah

Untuk melakukan analisis wilayah ini sendiri menggunakan metode perhitungan offset, sebelum menghitung nilai offsetnya untuk setiap block akan dicek menggunakan persamaan 2.4,2.5,2.6 persamaan ini dituliskan oleh Luo et al pada model robust detection.

$$Diff(k) < P(k) \quad (2.4)$$

$$Diff(1) + Diff(2) + Diff(3) < t1 \quad (2.5)$$

$$Diff(4) + Diff(5) + Diff(6) < t1 \quad (2.6)$$

jika blok yang dimasukan memenuhi pengecekan pada rumus 2.4-2.6 maka dapat dilanjutkan untuk menghitung nilai offset sendiri dengan rumus.

$$Dx = x_i - x_j \quad (2.7)$$

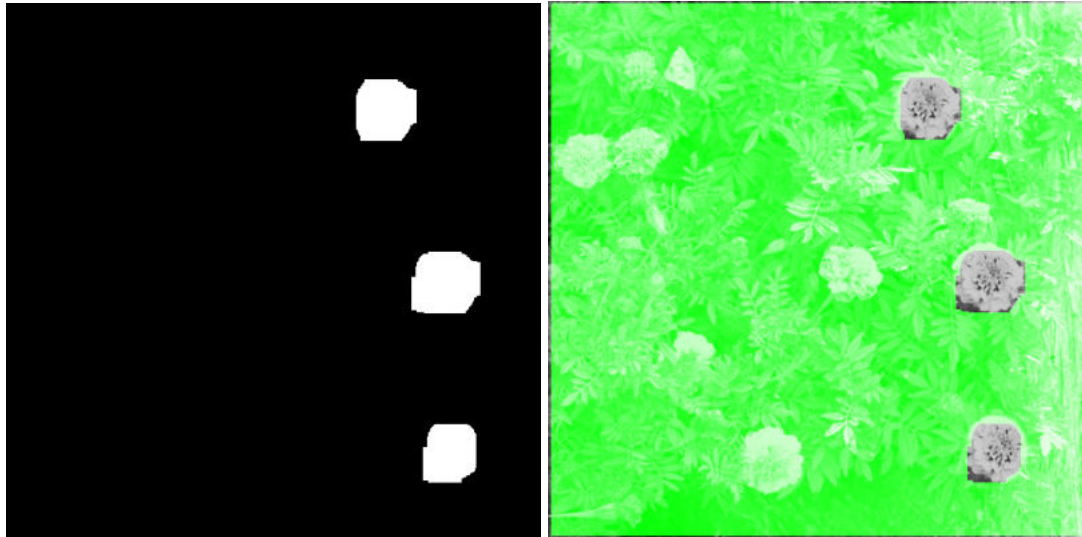
$$Dy = y_i - y_j \quad (2.8)$$

$$magnitude = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2.9)$$

Nilai offset ini sendiri didapatkan dengan menggunakan perhitungan membagi nilai blok satu dengan blok dua, kemudian pada metode karya Popescu et al dikatakan bila nilai offset kurang dari Nf (threshold frekuensi) dan nilai magnitude kurang dari Nd (threshold magnitude) maka akan dibuang, untuk nilai magnitude sendiri dapat dicari seperti pada rumus 2.9, jika memenuhi maka kedua nilai ini akan disimpan sebagai wilayah yang terduga sama.

5. Rekonstruksi

Kemudian setelah pasangan ditemukan maka dari hasil percobaan tadi akan direkonstruksi kembali menjadi gambar, untuk proses ini kelompok 3 mencontoh kode yang ada pada <https://github.com/tomopy/tomopy> dan juga melihat dari pekerjaan Nazali et al, sehingga hasilnya kurang lebih akan seperti pada Gambar 9.



Gambar 9. Hasil gambar dari rekonstruksi, kanan merupakan gambar groundtruth, kiri lokasi yang diterapkan pada gambar asal.

Selain dari metode-metode di atas sendiri pada tugas besar ini terdapat dua kode tambahan yang berfungsi untuk analisis dan melakukan penambahan noise untuk pengujian, untuk penambahan noise sendiri menggunakan *gaussian noise*, dan untuk perhitungan akan menghitung nilai TP, TN, Similarity, dan MSE.

2.3 Dataset


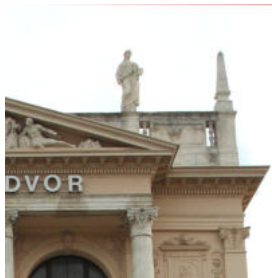



Dataset yang dibangun didasarkan dari gambar yang didapatkan pada CoMoFoD 512x512 dataset yang kemudian dilakukan modifikasi menggunakan Photoshop untuk menciptakan lokasi terduplikasi, kemudian untuk melakukan penambahan noise akan menggunakan fungsi yang sudah disiapkan menggunakan basis kode python, untuk kompresi akan menggunakan *online tools* yang terkait dengan kompresi.

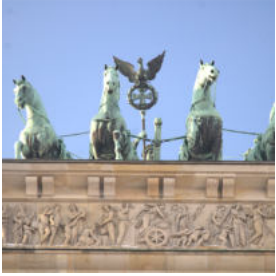
Selain dari dataset CoMoFoD ini sendiri disiapkan dari foto yang dimiliki oleh kelompok, namun kenapa lebih banyak CoMoFoD dikarenakan ukurannya lebih terstandar, sehingga gambarnya lebih terkontrol, kemudian untuk dataset ini sendiri disimpan pada tautan yang dapat diakses, dan di dalam tautan tersebut ada dataset baru dan lama, yang masing-masing berjumlah lima gambar dengan masing-masing dua manipulasi satu gambar.

Untuk dataset sendiri terdiri dari data-data yang dapat dijelaskan sebagai berikut ini.

2.3.1 Original Image





Untuk dataset ini merupakan gambar asal yang tidak dilakukan manipulasi apapun, untuk data gambar yang termasuk kategori ini adalah berikut ini.


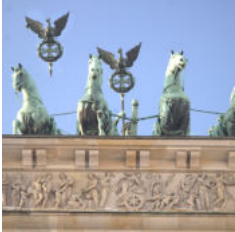
Nama Gambar	Jenis	Gambar
001_O.png	PNG	
012_O.png	PNG	
195_O.png	PNG	
200_O.png	PNG	
real_heggi.jpg	JPG	

kuda_dataset_original.png	PNG	
---------------------------	-----	-------------------------------------------------------------------------------------

2.3.2 Added Image




Untuk dataset ini merupakan gambar yang dilakukan manipulasi dengan melakukan duplikasi wilayah untuk menambahkan objek di dalamnya, dataset terdiri dari data berikut ini.



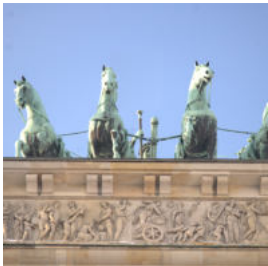
Nama Gambar	Jenis	Gambar
001_O_added.png	PNG	
012_O_added.png	PNG	
195_O_added.png	PNG	
200_O_added.png	PNG	

heggi_added.png	PNG	
kuda_dataset_added.png	PNG	

2.3.2 Removed Image

Untuk dataset ini merupakan gambar yang dilakukan manipulasi dengan melakukan duplikasi wilayah untuk menyembunyikan objek di dalamnya, dataset terdiri dari data berikut ini. .

Nama Gambar	Jenis	Gambar
001_O_added.png	PNG	
012_O_added.png	PNG	
195_O_added.png	PNG	

200_O_added.png	PNG	
heggi_removed.png	PNG	
kuda_dataset_added.png	PNG	

2.3.4 Groundtruth Image

Untuk dataset ini merupakan sebuah gambar dasar untuk melakukan perbandingan dengan hasil klasifikasi yang dilakukan oleh kode, kurang lebih contohnya akan seperti pada Gambar 10.



Gambar 10. Contoh Salah Satu Groundtruth Image

untuk membuat Groundtruth ini dilakukan dengan Photoshop CC 2019, dilakukan dengan merubah latar menjadi warna yang solid dan kemudian, layer yang dimanipulasi dilakukan layer via cut, sehingga layer tersebut nantinya dapat diberi *color overlay*.

Dataset dibuat menggunakan bantuan photoshop untuk melakukan duplicate region, namun untuk skenario pengkompresian menggunakan aplikasi lain yang akan dijelaskan pada bab 3, dan untuk dataset dengan noise akan dibangun dengan menggunakan kode yang sudah disiapkan, dataset dapat diakses pada :

- https://drive.google.com/drive/folders/1_6cqqawr75kLWE3LIjiAh3qV14emGTkc?usp=sharing,

untuk dataset yang dijelaskan terdapat dalam subfolder Dataset Baru.

2.4 Pengerjaan Kode

Untuk kode dari program yang dibangun dapat diakses pada tautan berikut ini https://github.com/RaihanHeggi/tubes_duplicate_region.git , untuk branchnya sendiri **master** berisi pengerjaan terbaru, sementara untuk yang sudah di integrasikan dengan GUI disimpan pada **branch graphic-ui**, kemudian untuk versi CMD dapat diakses pada branch non-GUI, untuk penjelasan lebih lanjut mengenai kode yang dibangun dan fungsionalitasnya akan dijelaskan di tabel dibawah ini.

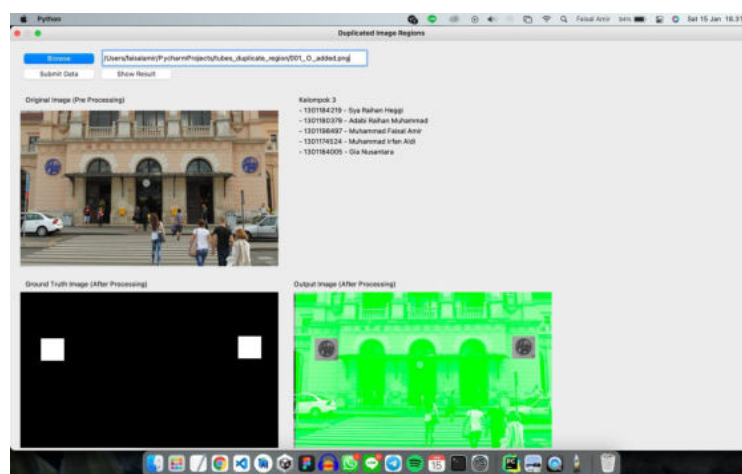
Nama Kode	Fungsi
add_noise.py	Fungsi yang dibangun untuk menambahkan, gaussian noise pada gambar yang diinginkan.
evaluation.py	Fungsi yang dibangun untuk menghasilkan evaluasi dari hasil klasifikasi, dengan memasukan gambar ground truth dari dataset dan hasil klasifikasi, kemudian akan menghasilkan perhitungan MSE, TPR, TNR, FPR, Accuracy.
Sistem Deteksi Duplicate Region	
main_ui.py	Merupakan kode yang berisi komponen dari GUI, sehingga aplikasi dapat dijalankan menggunakan GUI
detect.py	Merupakan kode yang berisi komponen untuk melakukan deteksi, dan kemudian melakukan rekonstruksi hasil dari pendeteksian
block.py	Merupakan kode yang berisi fungsionalitas dalam mengeluarkan <i>overlapping block</i> dan <i>characteristic features</i> , yang didapatkan dengan metode yang sudah dijelaskan sebelumnya.
sort_function.py	Merupakan kode yang berisi fungsionalitas, untuk melakukan lexicographical sort,

	dengan menggunakan parameter dari data characteristic dan block
analyze_function.py	Merupakan kode yang berisi fungsionalitas untuk melakukan analisis terhadap block data, sehingga dapat disimpulkan ada kemiripan atau tidak, kemudian bila ada maka akan dihitung offsetnya dan di kumpulkan untuk dibangun hasil akhir.

Untuk dapat menjalankan kode tersebut dapat menggunakan fungsi non GUI melalui CMD dengan memasukan pathnya pada kolom yang sudah disediakan, atau dapat menggunakan GUI dengan mengunduh versi GUInya pada branch **graphic-ui** , selain itu kode ini dibangun dengan beberapa komponen tambahan seperti yang berikut ini.

1. imageio==2.13.5 (**Digunakan untuk membangun gambar hasil**)
2. matplotlib==3.1.3 (**Digunakan untuk menampung plot jika dibutuhkan**)
3. numpy==1.18.1 (**Digunakan dalam pemrosesan data array/piksel dari gambar**)
4. opencv_contrib_python_headless==4.4.0.40 (**OpenCV yang digunakan untuk melakukan pembacaan pada fungsi evaluasi pada *evaluation.py***)
5. Pillow==9.0.0 (**Mirip seperti openCV untuk membaca gambar, namun library ini digunakan di fungsi deteksi**)
6. scikit_image==0.19.1 (**Terdapat fitur alternatif untuk membuat noise pada library ini**)
7. scikit_learn==1.0.2 (**Terdapat fitur yang digunakan untuk membuat noise pada library ini**)
8. skimage==0.0
9. tqdm==4.48.2 (**Untuk CMD base sebagai loading counter**)
10. PyQt5==5.15.6 (**Basis GUI yang digunakan**)

Untuk memudahkan sudah dibuat requirement.txt yang dapat digunakan untuk menginstal menggunakan pip.



Gambar 11. GUI Aplikasi

BAB 3

Hasil Pengujian

Untuk pengujian sendiri kelompok 3 melakukan pengujian sesuai dengan penugasan, dengan menggunakan deteksi pada dataset, dan juga pengujian jika data dimodifikasi dengan kompresi dan noise, untuk lebih jelasnya akan dijelaskan pada subbab-subbab berikutnya.

3.1 Pengujian terhadap dataset

Seperti yang sudah dijelaskan pada bab sebelumnya, sudah diciptakan dataset yang terdiri dari gambar yang dimanipulasi menambahkan dan menghilangkan objek, maka dilakukan pengujian dengan menggunakan parameter standar yang dijelaskan pada paper atau lebih jelasnya seperti pada Gambar 13.

```
# algorithm's parameters from the first paper
self.N = self.image_width * self.image_height
self.b = self.block_dimension * self.block_dimension

# popescu paper optimal reference
# self.b = 64

# luo et al Nb calculation formulas
# self.Nb = (self.image_width - self.block_dimension + 1) * (
#     self.image_height - self.block_dimension + 1
# )

# popescu et al Nb calculation formulas
self.Nb = (math.sqrt(self.N) - math.sqrt(self.b) + 1) ** 2


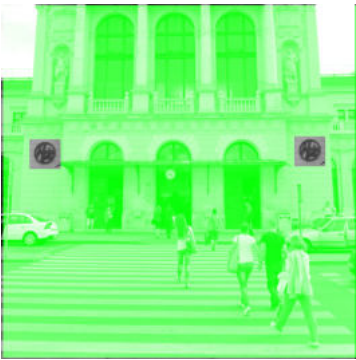
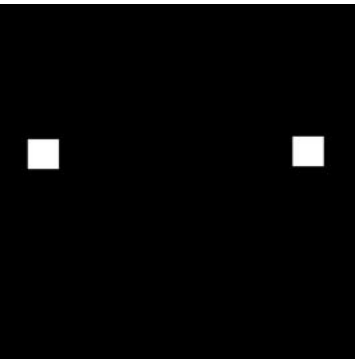

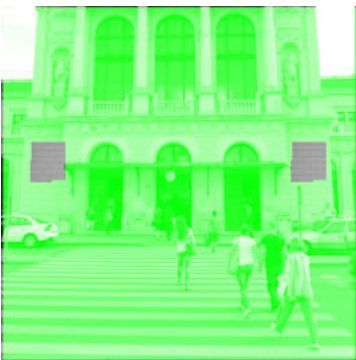


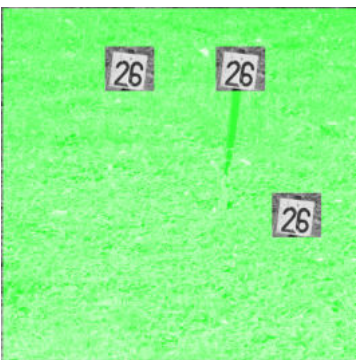
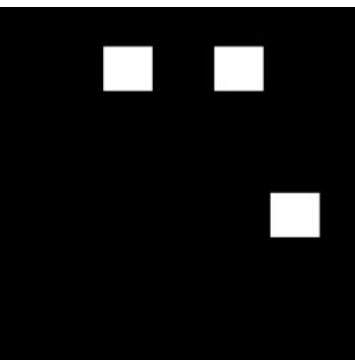

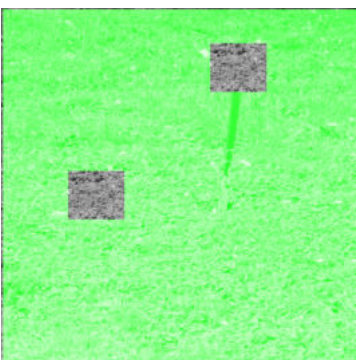
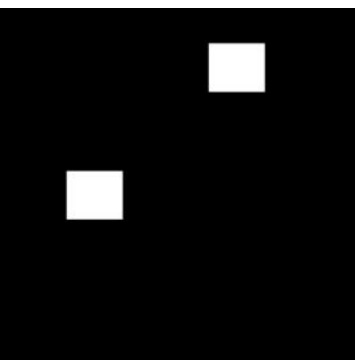
self.Nn = 100 # amount of neighboring block to be evaluated
self.Nf = 128 # minimum threshold of the offset's frequency
self.Nd = 16 # minimum threshold of the offset's magnitude

# algorithm's parameters from the second paper
self.P = (1.80, 1.80, 1.80, 0.0125, 0.0125, 0.0125, 0.0125)
self.t1 = 2.80
self.t2 = 0.02

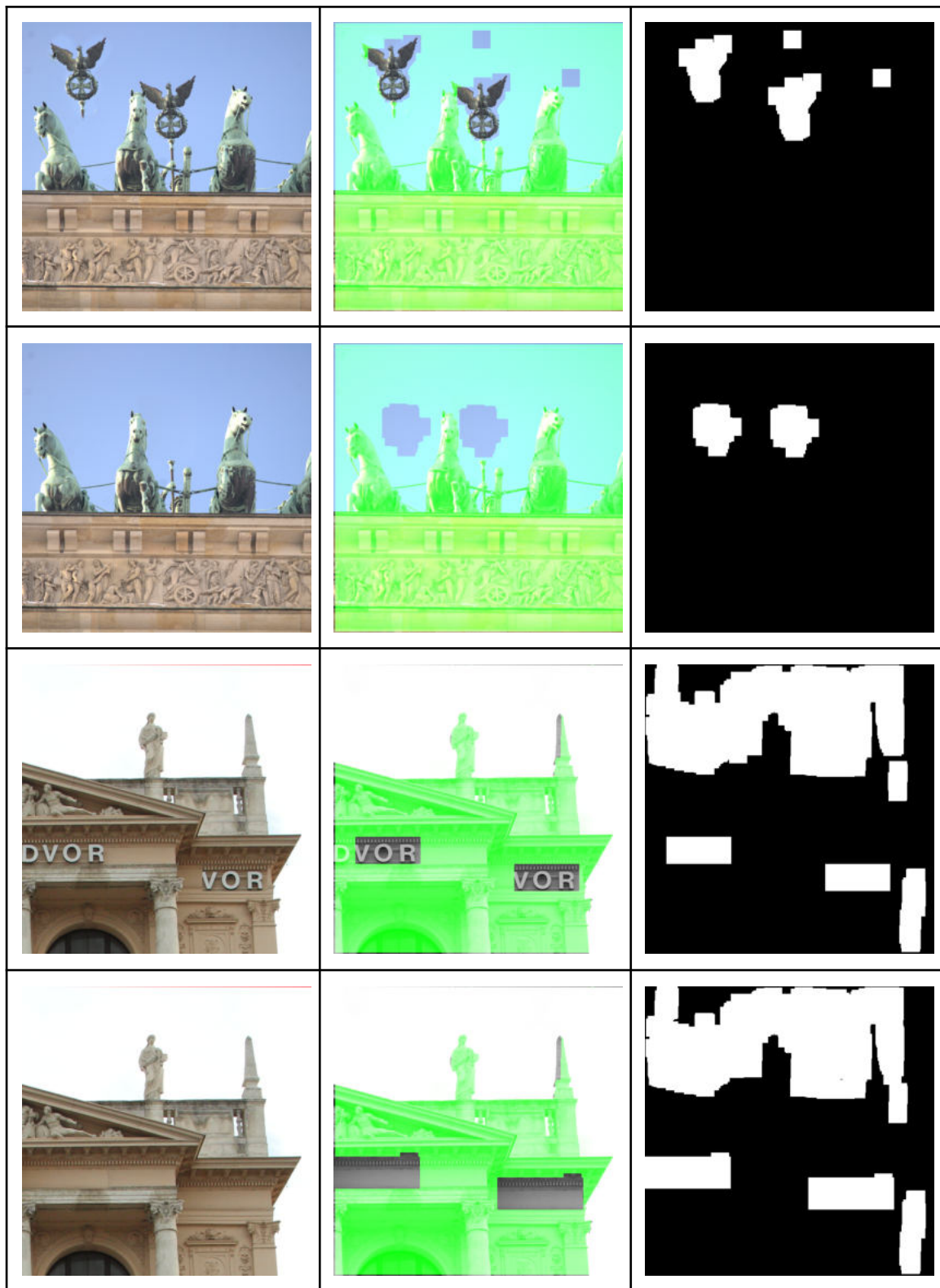
self.feature_list = list()
self.offset_dictionary = dict()
```

Gambar 13. Parameter Sistem yang digunakan

Dengan menggunakan parameter tersebut berhasil dilakukan pengujian terhadap serangan *duplicate region* ini, kemudian untuk uji coba yang dilakukan adalah memasukan dataset yang sudah dimodifikasi dan kemudian juga memasukan dataset yang aslinya, dan hasil yang didapatkan seperti tabel hasil di bawah ini.

Gambar Masukan	Gambar Keluaran	Groundtruth
		
		
		
		



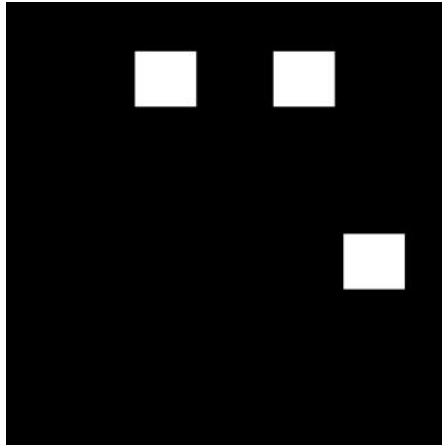


Kemudian dari percobaan sendiri sistem mampu untuk melakukan deteksi terhadap *duplicate region*, namun ada kekurangan yang dapat dilihat pada dataset yang memiliki warna putih di background yang cukup banyak, akan mengganggu hasil akhir dari deteksi seperti percobaan deteksi pada 012_O_added.png, dimana bisa dilihat lokasi duplikasi terdeteksi namun ada wilayah putih lain diatas yang merupakan background putih, oleh karena itu ada kecurigaan untuk mengatur parameter dari sistem yang

dibangun. Kemudian dilakukan percobaan untuk mengatur parameter, parameter yang kami gunakan berasal dari penelitian yang dilakukan oleh Nazali et al selain itu kami juga menggunakan parameter dari paper rujukan, kemudian dengan parameter yang digunakan akan didapatkan hasil seperti berikut ini.

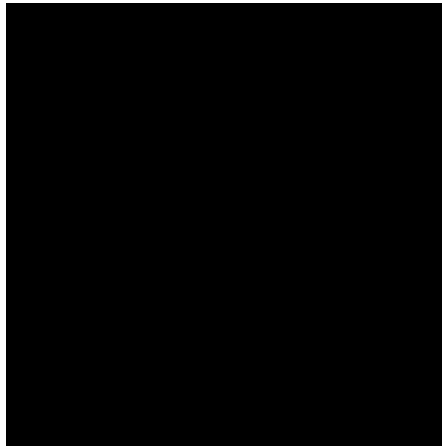
Nb	Nn	Nf	Nd	P	T1	T2	MSE
Rumus 2.1	100	128	16	(1.80, 1.80, 1.80, 0.0125, 0.0125, 0.0125, 0.0125)	2.80	0.0.2	6608.07 0373535 156
Rumus 2.2	2	50	750	(1.80, 1.80, 1.80, 0.0125, 0.0125, 0.0125, 0.0125)	2.80	0.0.2	9001.26 3427734 375
Rumus 2.1	2	50	750	(1.80, 1.80, 1.80, 0.0125, 0.0125, 0.0125, 0.0125)	2.80	0.0.2	9001.26 3427734 375
Rumus 2.1	50	128	16	(1.80, 1.80, 1.80, 0.0125, 0.0125, 0.0125, 0.0125)	2.80	0.0.2	95.2514 6484375

Untuk hasil yang dengan memodifikasi parameter dapat digunakan, didapatkan hasil-hasil berikut ini.



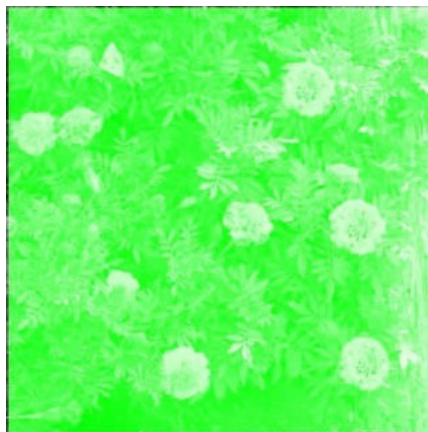
Gambar 14. Hasil Parameter Pertama

Dengan menggunakan parameter pertama sistem dapat menghasilkan hasil deteksi, namun dengan menggunakan parameter kedua sistem tidak dapat menemukan hasil yang dikeluarkan, wilayah yang terduga lokasi yang diaplikasikan

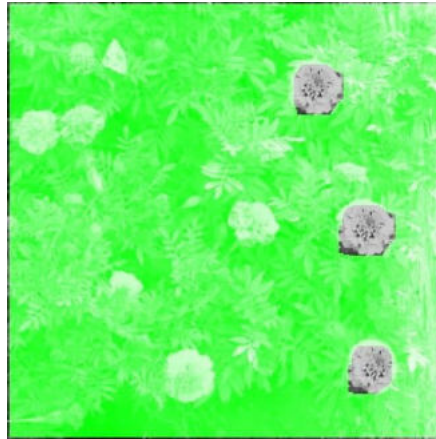


Gambar 15. Hasil Parameter Kedua

Sehingga disimpulkan bahwa parameter sendiri dapat mempengaruhi hasil akhir dari deteksi, dan permasalahan berikutnya yang ditemukan karena parameter ini adalah belum ada yang cocok untuk memproses data JPEG, karena dari percobaan yang dilakukan melakukan deteksi objek yang sama namun berbeda tipe hasilnya hanya PNG terdeteksi untuk contoh hasilnya ada di Gambar 16 dan Gambar 17.

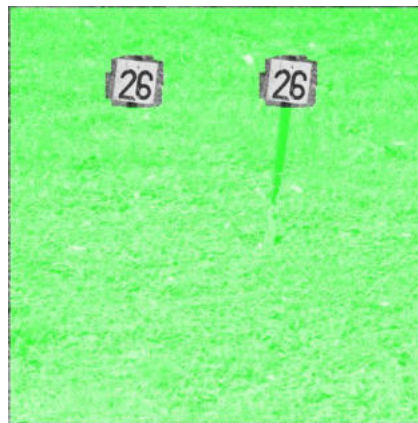


Gambar 16. Objek JPEG



Gambar 17. Objek PNG

Namun dengan mengganti metode manipulasinya dengan menggunakan *layer copy*, bukan menggunakan metode *clone stamp*, sistem dapat melakukan deteksi seperti pada Gambar 18, sehingga bisa disimpulkan parameter dan metode manipulasi berpengaruh terhadap deteksi sistem.



Gambar 18. Objek JPEG Terdeteksi

Sehingga pada akhirnya diputuskan untuk menggunakan parameter yang digunakan pada paper karena hasilnya masih dapat dilakukan deteksi, dan hasilnya memiliki nilai MSE yang cukup baik.

Selanjutnya dari hasil yang sudah didapatkan dilakukan pengecekan evaluasi yang dapat yang dapat ditampilkan sebagai berikut ini.

Gambar	(TN,TP,F P,FN)	TPR	TNR	FPR	Akurasi	MSE
ground_tr uth_real_ heggi_add ed.png	(197248, 62272,0, 2624)	23.99	100	0	98.99	1952.66

ground_truth_real_hoggi_removed.png	(253058, 7038, 0, 2048)	2.71	100	0	99.22	1524.02
groundtruth_001_O_added.png	(258274, 3870, 0, 0)	1.48	100	0	100	0
groundtruth_001_O_removed.png	(256446, 5390, 0, 308)	2.06	100	0	99.88	229.20
groundtruth_195_O_added.png	(244840, 8424, 3672, 5208)	3.33	98.52	1.48	96.61	6608.07
groundtruth_195_O_removed.png	(254104, 8040, 0, 0)	3.07	100	0	100	0
groundtruth_200_added_O.png	(0, 16200, 0, 0)	-	-	-	100	137.91
groundtruth_200_removed_O.png	(254336, 7808, 0, 0)	2.97	100	0	100	0
groundtruth_remove_012_O.png	(165177, 17328, 0, 79639)	9.49	100	0	69.62	59263.53

Selanjutnya untuk parameter evaluasi sendiri mengapa dipilih untuk menggunakan **True Positive Rate, True Negative Rate, False Positive Rate, MSE, dan Similarity** dapat dijelaskan pada poin-poin berikut ini.

- **True Positive Rate** merupakan persentase nilai benar-benar positif dari hasil deteksi yang dilakukan, digunakan sehingga mengetahui performansi sistem dalam mendeteksi data dalam kategori positif.

- **True Negative Rate** merupakan persentase nilai benar-benar salah dari hasil deteksi yang dilakukan, digunakan untuk mengetahui performansi sistem dalam mendeteksi data dalam kategori negatif, dan mencegah terjadinya misinformasi.
- **False Negative Rate** merupakan persentase nilai dari hasil deteksi yang salah klasifikasi sebagai positif, mengapa perlu diketahui agar dapat meningkatkan lagi sistem dan parameter yang digunakan, sehingga tidak ada salah deteksi menjadi benar
- **Mean Squared Error (MSE)** merupakan sebuah parameter yang menampilkan kemiripan gambar, pada tugas besar ini digunakan untuk membandingkan ground truth yang sudah disiapkan dengan ground truth yang dihasilkan oleh sistem, untuk nilai MSE sendiri semakin kecil maka semakin mirip kedua gambar tersebut..

Untuk mendapatkan hasil evaluasi ini maka akan digunakan kode yang sudah di siapkan di **evaluation.py** dimana akan menerima dua buah masukan *img_a* yang merupakan *ground truth* dari dataset, sementara *img_b* merupakan *ground truth* yang dihasilkan dari klasifikasi, selanjutnya diproses untuk mendapatkan nilai TN, TP, FP, FN (Confusion Matrix), untuk mendapatkan nilai ini maka dilakukan asumsi, piksel yang berwarna putih merupakan hasil klasifikasi bernilai 1 (wilayah duplikasi), sementara piksel berwarna hitam hasil klasifikasi bernilai 0 (tidak ada duplikasi), lebih jelasnya di jelaskan pada Gambar 19.

```
# [255, 255, 255] white region (equal to 1)
# [0,0,0] black region (equal to 0)
def calculateValue(img_a, img_b):
    x, y = img_a.shape[0:2]
    truePositivePx, trueNegativePx, falseNegativePx, falsePositivePx, = 0, 0, 0, 0
    for i in range(x):
        for j in range(y):
            if sum(img_a[i, j]) == 0 and sum(img_b[i, j]) == 0:
                trueNegativePx += 1
            elif sum(img_a[i, j]) == 765 and sum(img_b[i, j]) == 765:
                truePositivePx += 1
            elif sum(img_a[i, j]) == 0 and sum(img_b[i, j]) == 765:
                falseNegativePx += 1
            elif sum(img_a[i, j]) == 765 and sum(img_b[i, j]) == 0:
                falsePositivePx += 1
    return truePositivePx, trueNegativePx, falsePositivePx, falseNegativePx
```

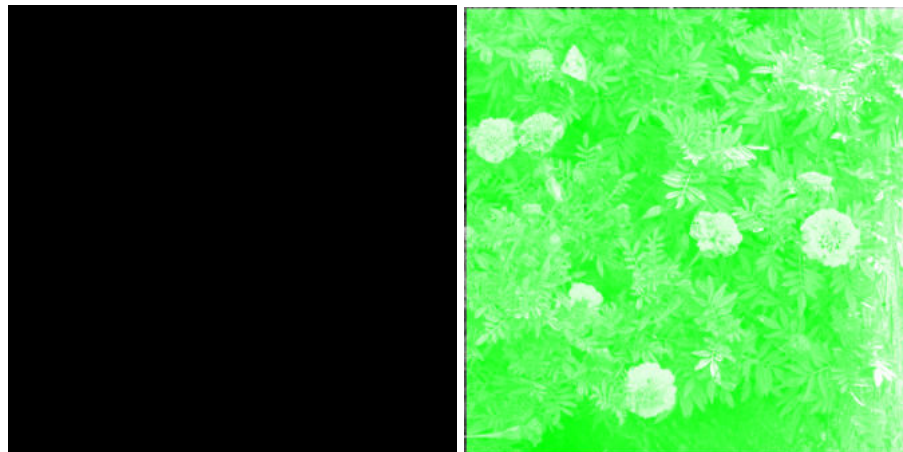
Gambar 19. Mendapatkan nilai *Confusion Matrix*

Pada gambar 19 sendiri terdapat sebuah fungsi penjumlahan, untuk nilai 765 merupakan piksel berwarna putih, mengapa didapatkan angka tersebut sebagai warna putih, karena didalam pikselnya terdapat R,G,B maka untuk memudahkan maka untuk menentukan bahwa piksel itu putih atau hitam dari jumlahnya, dan dari hasil observasi yang dilakukan didapatkan nilai 0 (hitam) dan 765 (putih).

Kemudian ketika dilakukan pengujian terhadap dataset original, dalam hal ini datasetnya tidak dimanipulasi ketika dijalankan maka hasilnya seperti pada Gambar 20 dan 21.



Gambar 20. Original Image Detection



Gambar 21. Original Image Detection

Dari Gambar 20 dan 21 bisa dilihat bahwa gambar tanpa manipulasi terdeteksi sebagai gambar yang tidak termanipulasi, dengan tidak ditemukan wilayah yang terduplikasi, selain itu bila merujuk pada hasil parameter evaluasi bisa dikatakan parameter dan sistem bekerja dengan baik untuk dataset yang disiapkan, dari percobaan pengujian yang dilakukan sendiri akurasi terendah dari sistem adalah 69% dan tertinggi 100%, dengan nilai FPR di setiap deteksi hampir 0, selain itu ada beberapa hasil yang menghasilkan gambar dengan nilai MSE 0 atau dalam kata lain gambar asal dan hasil klasifikasi sama.


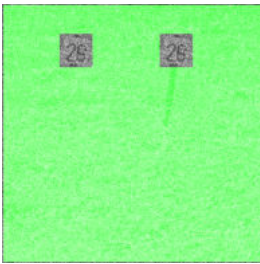

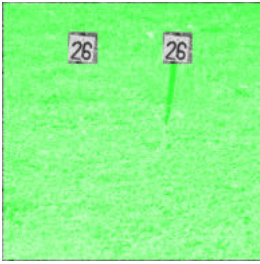

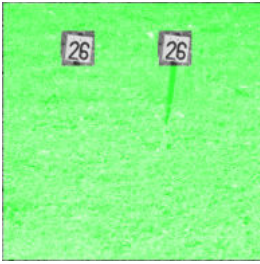

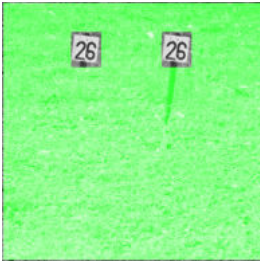
3.2 Skenario menambahkan noise

Skenario ini akan melakukan deteksi pada gambar yang memiliki noise, untuk noise sendiri menggunakan SNR dengan gaussian noise, untuk pembuatan noise sendiri menggunakan kode yang sudah disiapkan pada *add_noise.py*, untuk memberikan noise sendiri kurang lebih menggunakan variabel SNR, SNR sendiri dapat didapatkan menggunakan rumus 3.1 dan 3.2

$$SNR = \frac{P_{signal}}{P_{noise}} \quad (3.1)$$

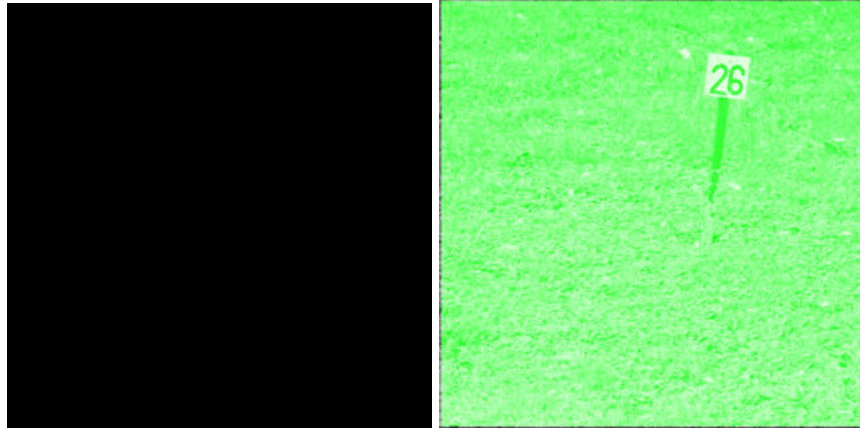
$$SNR_{conversion} = 10x\left(\frac{SNR_{dB}}{10}\right) \quad (3.2)$$

Kemudian dengan landasan tersebut maka dibuat baris kode dengan basis pengetahuan tersebut, untuk melakukan pengujian maka dipilih satu gambar dengan nama 195_O.png yang kemudian akan di generate noise (10dB,20dB,30dB,40dB) dan dilakukan manipulasi. Sehingga dari pengujian didapatkan hasil yang dapat dijelaskan menggunakan tabel berikut ini.

dB	Asal	Hasil	MSE	Akurasi
10			0	100
20			47.63	99.98
30			110.14	99.94
40			47.63	99.98

Kemudian bila tidak terjadi manipulasi namun gambar memiliki noise apakah sistem dapat melakukan deteksi, hasil ini dibuktikan dengan Gambar 22, pada gambar tersebut

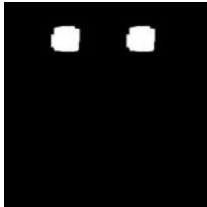
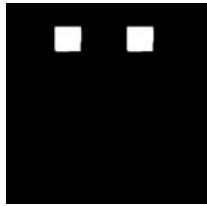
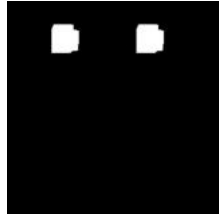
tidak ditemukan lokasi yang terdeteksi, dalam hal ini berarti sistem dengan parameter yang digunakan tidak melakukan *False Positive Detection*, selain itu juga dapat disimpulkan dengan adanya gangguan noise sendiri sistem masih dapat untuk melakukan deteksi.

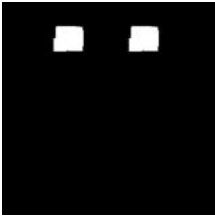


Gambar 22. Noise tanpa manipulasi

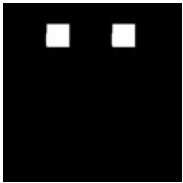

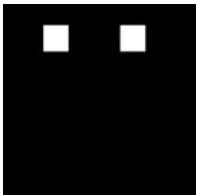
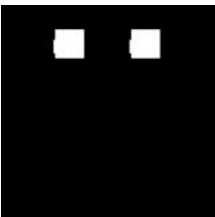
3.3 Skenario dilakukan kompresi

Untuk kompresi sendiri dilakukan menggunakan aplikasi online dan salah satu *tools* https://github.com/evgenyneu/image_compressor_python, namun seperti yang diketahui sistem parameternya belum tepat untuk melakukan deteksi file .jpg jadi alternatif kompresinya akan melakukan kompresi pada file .png.

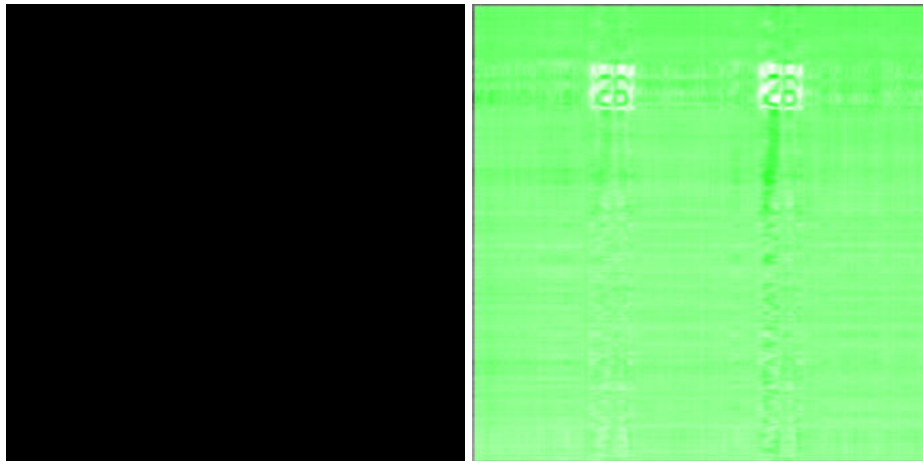
Kompresi	Hasil	MSE	Akurasi
50		981.94	93.29
60		337.50	99.89
70		703.13	99.74

80		1403.47	99.41
----	-----------------------------------------------------------------------------------	---------	-------

dari hasil pengujian sendiri dapat dilihat dengan melakukan kompresi sendiri dapat mempengaruhi kualitas pendeteksian, dikarenakan sistem tidak dapat mendeteksi 100% benar, namun hasilnya masih cukup memuaskan karena MSE dan akurasi masih cukup baik. Kemudian jika kompresi dilakukan pada file png, dan hasilnya dapat dirangkum dengan tabel berikut ini.

Kompresi	Hasil	MSE	Akurasi
50		123.49	-
60		142.88	99.93
70		0	100
80		190.50	99.91

kemudian dilakukan dengan menggunakan kompresi singular terhadap yang sudah disiapkan dari repository dan hasil yang didapatkan seperti Gambar 21.



Gambar 21. Hasil Deteksi Kompresi dengan 5 term

Dari hasil yang didapatkan bahwa dengan melakukan kompresi sendiri terhadap gambar, akan mempengaruhi hasil akhir, pada percobaan JPEG kompresi semakin tinggi maka nilai MSE juga meningkat, begitu juga pada PNG sama hasil yang didapatkan. Namun sistem yang dibangun masih dapat melakukan deteksi terhadap skenario tersebut, walaupun akurasi tidak mencapai 100%, kemudian untuk kompresi dengan metode dengan tools yang didapatkan dari github menyebabkan tidak dapat ditemukan hasil akhirnya.

BAB 4

Kesimpulan

Pada bab ini akan berisi kesimpulan dari hasil percobaan yang dilakukan dan selain itu akan dijelaskan juga evaluasi yang dapat dilakukan untuk kedepannya.

4.1 Kesimpulan

Kesimpulan yang didapatkan dari percobaan yang dilakukan dengan variasi skenario, dan dataset dengan menggunakan *robust-duplication detection*, adalah sebagai berikut:

1. Metode *Robust-Duplication Detection* dapat melakukan deteksi pada kasus pemalsuan Wilayah Duplikasi, namun sistem ini masih perlu dilakukan *tuning* agar hasil deteksi menjadi lebih baik, hal ini dikarenakan sistem yang dibangun terkadang tidak dapat mendeteksi gambar berformat PNG dan JPG pada parameter tertentu.
2. Untuk percobaan terhadap dataset yang sudah dibuat memiliki nilai akurasi tertinggi 100% dan nilai terendah 69%, dalam kata lain sistem ini sudah cukup baik, dikarenakan sudah mampu mendeteksi diatas 50% namun masih dapat ditingkatkan.
3. Bila dilihat gambar yang memiliki akurasi rendah dikarenakan ada objek yang memiliki warna yang monoton seperti gambar 012_O.png, sehingga ini masih menjadi kelemahan dari sistem.
4. Bila merujuk pada nilai-nilai evaluasinya bisa dilihat, tingkat FPR yang dihasilkan rendah dan nilai TNR tinggi dalam hal ini berarti sistem sudah mampu melakukan deteksi dengan baik, dan tidak melakukan kesalahan untuk mengklasifikasikan nilai yang salah menjadi benar.
5. Kemudian untuk nilai MSE sendiri nilai paling besarnya adalah 59263.53, yang merupakan nilai deteksi terburuk, namun untuk rata-rata nilainya sendiri cukup rendah, dalam kata lain sudah optimal kinerjanya.
6. Dengan menambahkan noise membuat sistem juga sedikit terganggu kinerjanya, namun dari percobaan yang dilakukan sudah cukup baik untuk percobaan dengan noise (10 dB - 40 dB), dan sistem masih bisa membedakan gambar yang dimanipulasi serta tidak.
7. Kemudian skenario kompresi, juga berpengaruh pada sistem namun sistem masih dapat mendeteksi, namun dengan metode kompresi yang didapatkan pada github, sistem tidak dapat melakukan deteksi lebih lanjut.
8. Proses manipulasi juga berperan karena pada percobaan dengan menggunakan *clone stamp* sistem hanya dapat mendeteksi gambar berformat PNG, sementara dengan *layer copy* sistem dapat mendeteksi keduanya.
9. Algoritma yang digunakan bisa lebih baik lagi, karena dengan ukuran gambar yang terstandar di 512x512 piksel memakan 10-20 menit dalam melakukan deteksi, pada device pengujian.
10. Kemudian jika ukuran block diperbesar maka waktu operasi yang digunakan semakin lama, namun untuk wilayah yang dideteksi terkadang menjadi lebih baik.

11. Untuk Evaluasi dan Penambahan Noise mungkin kedepannya dapat diintegrasikan ke dalam satu sistem yang sama sehingga memudahkan untuk pengerjaan.

Sehingga dari penjelasan yang diuraikan dapat dikatakan sistem yang dibangun bekerja dengan baik, hal ini juga di dukung dengan hasil evaluasi yang dihasilkan cukup bagus, namun masih ada celah yang dapat diperbaiki kedepannya.

4.2 Saran

Kedepannya sistem dapat dibuat lebih efektif lagi dari segi runtime, maupun hasil deteksi dikarenakan hasil saat ini masih belum maksimal, parameter masih dapat digali lebih dalam lagi, kemudian untuk metode pengecekannya bisa dibuat dengan lebih baik dengan semisal mengecek *distance* karena pada sistem bergantung kepada threshold dan probabilitas yang dijelaskan pada paper (tidak dinamis), selain itu untuk GUI bisa di perbaiki lagi tampilannya.

4.3 Tautan Terkait

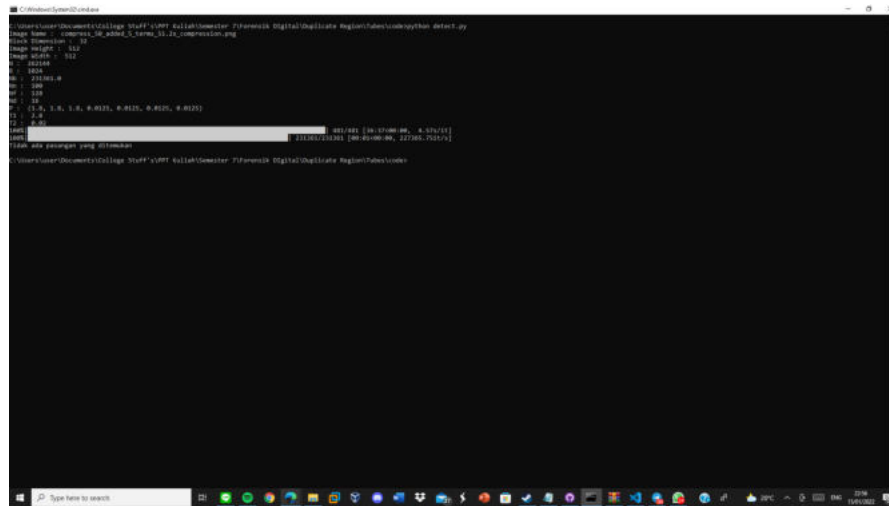
1. Presentasi
https://docs.google.com/presentation/d/1yBh25Efaw2J0XFEIOXw7j0HJAt7SuJc7NQ1F_1E7lrl/edit?usp=sharing
2. Drive Data
<https://drive.google.com/drive/folders/16MsemAoYtVwBHoat1uonHRO2qNhCftgl?usp=sharing>
3. Repository Kode
https://github.com/RaihanHeggi/tubes_duplicate_region.git (Dimohon membaca readme karena ada beberapa branch dan versi dari sistem yang dibangun)

DAFTAR PUSTAKA

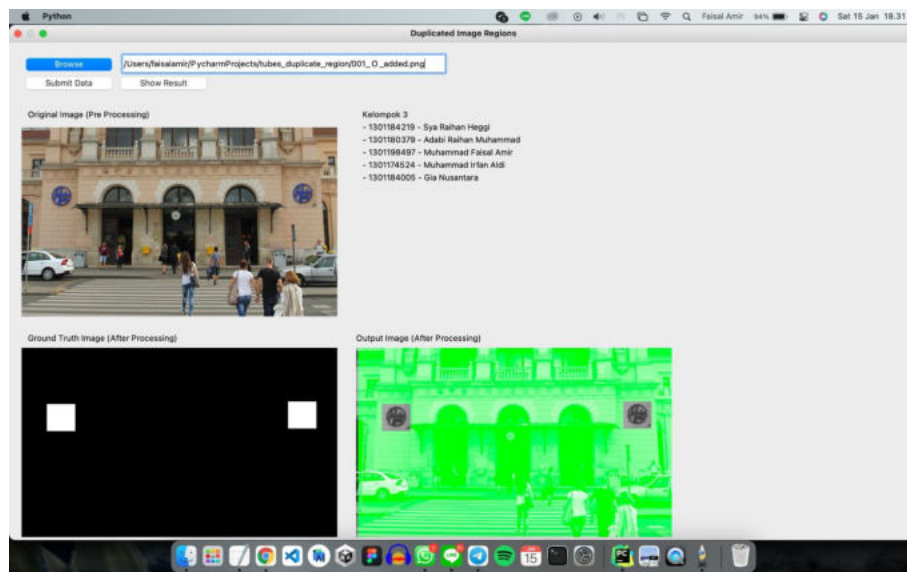
- [1] Luo, Weiqi, Jiwu Huang, and Guoping Qiu. "Robust detection of region-duplication forgery in digital image." *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 4. IEEE, 2006.
- [2] Popescu, Alin C., and Hany Farid. "Exposing digital forgeries by detecting duplicated image regions." (2004).
- [3] Studiawan, Hudan, Rahmat Nazali Salimi, and Tohari Ahmad. "Forensic Analysis of Copy-Move Attack with Robust Duplication Detection." *International Conference on Soft Computing and Pattern Recognition*. Springer, Cham, 2020.

LAMPIRAN

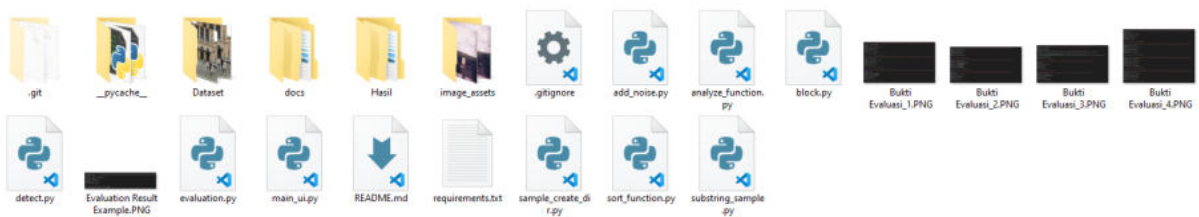
Lampiran Gambar



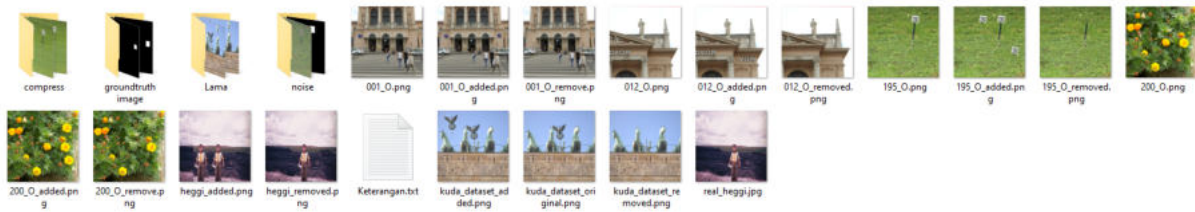
Gambar Sistem Basis CMD



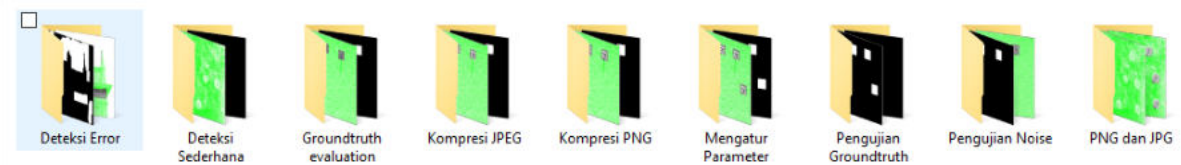
Gambar Sistem Basis GUI



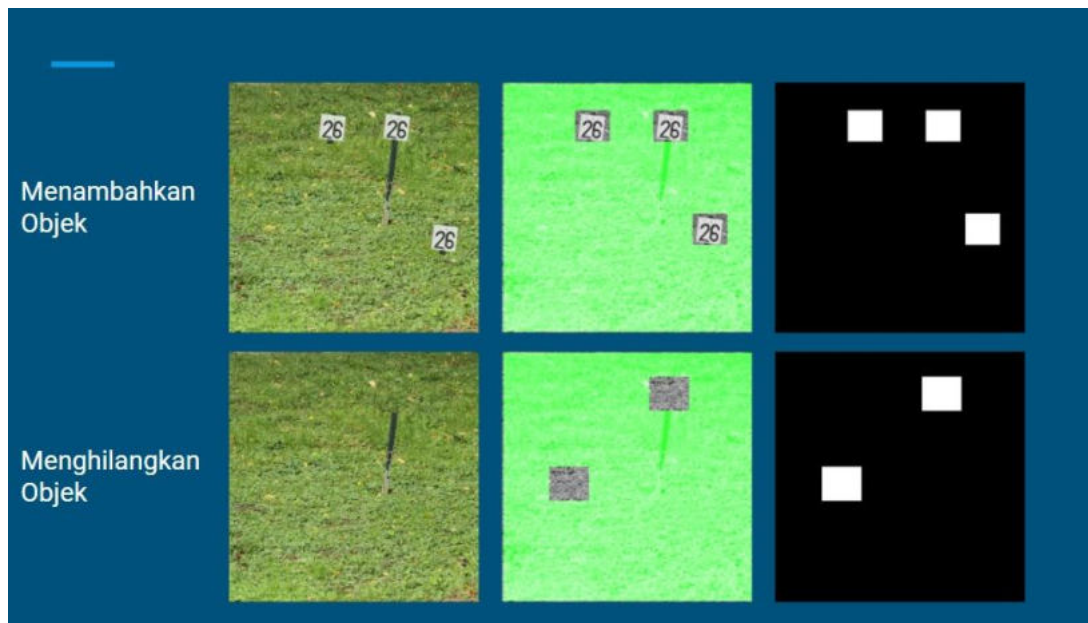
Gambar Listing Objek Yang Ada



Gambar Dataset Disiapkan



Gambar Hasil Deteksi dan Skenario Pengujian



Gambar Input dan Output Sistem

Lampiran Tabel

Daftar file dan objek yang digunakan	
Nama Objek	Tipe Data
__pycache__ (python cache file)	.pyc
DATASET	
Compress (Dataset yang digunakan untuk skenario kompresi)	.JPG, .PNG

Groundtruth Image	PNG
Lama (folder dataset lama tidak digunakan)	.JPG, .PNG
Noise (Dataset yang digunakan untuk skenario noise)	.PNG
File Gambar Dataset	.PNG, .JPG
HASIL	
Deteksi Error (Deteksi dan terdapat kesalahan)	.PNG
Deteksi Sederhana (Hasil Deteksi Dataset tanpa skenario)	.PNG
Groundtruth Evaluation (Folder hasil untuk evaluasi)	.PNG
Kompresi JPEG	.JPG
Kompresi PNG	.JPG
Mengatur Parameter (Contoh gambar ketika parameter diatur)	.PNG
Pengujian Groundtruth (Folder Groundtruth Image untuk Pengujian)	.PNG
Pengujian Noise	..PNG
PNG dan JPG (Hasil Perbandingan klasifikasi pada jenis gambar)	.PNG, .JPG
SISTEM	
add_noise.py (Modul Penambahan Noise)	py
analyze_function.py (Modul Analisis Gambar)	py
detect.py (Modul Deteksi Gambar)	py
evaluation.py (Modul Evaluasi Hasil)	py
main_ui.py (Modul GUI)	py
sample_create_dir.py (Kode contoh untuk menyimpan data hasil)	py
sort_function.py (Modul Lexicographical Sort)	py

substring_sample.py (Kode contoh untuk penamaan)	py
requirement.txt (berisi library python yang digunakan)	.txt
LAINNYA	
Bukti_Evaluasi_1	.png
Bukti_Evaluasi_2	.png
Bukti_Evaluasi_3	.png
Bukti_Evaluasi_4	.png
Evaluation Result Example	.png
image_assets (berisi beberapa gambar yang sudah tidak digunakan)	.jpg, .png
docs (berisi dokumentasi GUI)	.jpg