

**UTS**

**PENGOLAHAN CITRA**



**INTELLIGENT COMPUTING**

**NAMA : M. Rafif Soniansyah**

**NIM : 202331018**

**KELAS : D**

**DOSEN : Ir. Darma Rusjdi, M.Kom**

**NO.PC :**

**ASISTEN : 1. Davina Najwa Ermawan**

**2. Fakhrol Fauzi Nugraha Tarigan**

**3. Viana Salsabila Fairuz Syahla**

**4. Muhammad Hanief Febriansyah**

**INSTITUT TEKNOLOGI PLN**

**TEKNIK INFORMATIKA**

**2024/2025**

## DAFTAR ISI

DAFTAR ISI.....	2
BAB I PENDAHULUAN.....	3
1.1 Rumusan Masalah.....	3
1.2 Tujuan Masalah.....	3
1.3 Manfaat Masalah.....	3
BAB II LANDASAN TEORI.....	4
2.1. Pengolahan Citra Digital.....	4
2.2. Deteksi Warna.....	7
2.3. Histogram Citra.....	9
2.4. Ambang Batas (Thresholding).....	10
2.5. Backlight dan Peningkatan Kualitas Citra.....	11
BAB III HASIL.....	13
3.1. Deteksi Warna Pada Citra.....	13
3.2. Ambang Batas Terkecil sampai dengan Terbesar.....	18
3.3. Memperbaiki Gambar Backlight.....	19
BAB IV PENUTUP.....	23
4.1. Kesimpulan.....	23
DAFTAR PUSTAKA.....	24

**BAB I PENDAHULUAN****1.1 Rumusan Masalah**

1. Bagaimana cara mendeteksi warna merah, hijau, dan biru dalam sebuah citra digital?
2. Bagaimana menganalisis hasil deteksi warna melalui histogram warna?
3. Bagaimana menentukan dan mengurutkan nilai ambang batas dari yang terkecil hingga terbesar?
4. Bagaimana memperbaiki citra digital yang mengalami pencahayaan backlight agar objek utama terlihat jelas?.

**1.2 Tujuan Masalah**

1. Mendeteksi komponen warna utama (merah, hijau, biru) pada gambar menggunakan teknik pengolahan citra digital.
2. Menampilkan dan menganalisis histogram warna dari setiap kanal warna.
3. Menentukan nilai ambang batas optimal untuk menampilkan kategori warna pada citra.
4. Menerapkan teknik peningkatan kualitas citra pada gambar dengan kondisi backlight agar informasi objek lebih jelas.

**1.3 Manfaat Masalah**

1. Mahasiswa memahami konsep dasar dan implementasi pengolahan citra digital secara praktis.
2. Menambah kemampuan dalam menganalisis dan memanipulasi gambar digital berdasarkan fitur warna.
3. Meningkatkan kemampuan dalam penggunaan Python dan pustaka OpenCV untuk pemrosesan citra.
4. Mengembangkan solusi untuk permasalahan umum dalam pengambilan gambar seperti pencahayaan buruk (backlight).

## BAB II LANDASAN TEORI

### 2.1. Pengolahan Citra Digital

Pengolahan citra digital adalah cabang dari ilmu komputer dan teknik elektro yang berkaitan dengan manipulasi dan analisis gambar dalam bentuk digital menggunakan algoritma komputer. Citra digital sendiri merupakan representasi visual dua dimensi yang disimpan dalam bentuk diskret (berbasis piksel). Pengolahan citra digital bertujuan untuk meningkatkan kualitas visual gambar, mengekstraksi informasi yang bermanfaat, serta mendukung proses pengambilan keputusan secara otomatis dalam berbagai sistem cerdas (Tirumani et al., 2021).

Pengolahan citra memiliki beberapa tujuan utama, antara lain:

- **Peningkatan visual (image enhancement)**

Peningkatan visual (image enhancement) adalah proses dalam pengolahan citra digital yang bertujuan untuk memperbaiki kualitas tampilan gambar sehingga lebih jelas dan mudah dipahami baik oleh manusia maupun sistem komputer. Proses ini fokus pada peningkatan kontras, pengurangan noise, dan penajaman tepi gambar. Teknik peningkatan kontras digunakan untuk memperjelas perbedaan antara bagian terang dan gelap dalam citra, seperti melalui histogram equalization atau contrast stretching. Pengurangan noise bertujuan menghilangkan gangguan acak yang muncul pada citra, menggunakan filter seperti Gaussian filter atau median filter. Penajaman tepi dilakukan untuk menonjolkan batas-batas objek dalam citra, menggunakan metode seperti operator Sobel atau unsharp masking, sehingga detail gambar lebih terlihat jelas. Selain itu, penyesuaian kecerahan dilakukan untuk mengatur tingkat terang atau gelap pada citra yang diambil dalam kondisi pencahayaan buruk. Peningkatan warna juga sering diterapkan untuk memperbaiki atau mempertegas warna dalam citra berwarna, agar lebih mudah membedakan objek satu dengan lainnya. Secara keseluruhan, teknik-teknik ini digunakan untuk mempermudah analisis citra lebih lanjut, seperti dalam aplikasi medis, pemrosesan citra satelit, atau foto digital, dengan memastikan citra yang dihasilkan memiliki kualitas yang optimal.

- **Analisis citra (image analysis)**

Analisis citra (image analysis) adalah proses yang bertujuan untuk mengekstraksi informasi yang relevan dari citra digital untuk digunakan dalam aplikasi atau proses lebih lanjut, seperti pengenalan pola, klasifikasi objek, atau segmentasi bagian penting dari citra. Proses ini melibatkan teknik-teknik untuk memahami dan menginterpretasi struktur serta elemen-elemen dalam gambar yang dapat memberikan informasi berguna. Misalnya, dalam pengenalan pola, algoritma digunakan untuk mengidentifikasi kesamaan atau fitur tertentu dalam citra yang dapat dikaitkan dengan kategori atau kelas objek tertentu. Sedangkan dalam klasifikasi objek, analisis citra memungkinkan untuk

membedakan dan mengategorikan objek-objek yang ada dalam citra berdasarkan fitur yang telah diekstraksi, seperti bentuk, tekstur, atau warna. Segmentasi, sebagai salah satu langkah penting dalam analisis citra, bertujuan untuk memisahkan objek atau wilayah penting dalam citra dari latar belakang atau elemen lainnya, sehingga memudahkan dalam proses pengolahan lebih lanjut. Analisis citra banyak digunakan dalam berbagai bidang, seperti medis untuk mendiagnosis penyakit, keamanan untuk pengenalan wajah, serta dalam industri untuk inspeksi otomatis produk.

- **Restorasi citra (image restoration)**

Restorasi citra (image restoration) adalah proses untuk mengembalikan citra yang rusak atau terdistorsi ke kondisi aslinya, atau setidaknya mendekati citra asli sebelum terdegradasi. Degradasi citra dapat terjadi karena berbagai faktor, seperti noise, blur, atau gangguan lainnya yang menyebabkan kualitas gambar menurun. Restorasi citra bertujuan untuk memperbaiki kerusakan ini dengan menggunakan model degradasi tertentu dan teknik pemrosesan matematis. Teknik yang umum digunakan dalam restorasi citra meliputi pemfilteran (seperti filter Wiener atau filter Kalman), deblurring (untuk mengatasi citra buram), dan penggunaan metode inverse filtering yang mengembalikan citra ke bentuk yang lebih jelas. Proses ini sangat penting dalam bidang medis untuk memperbaiki citra medis yang terdegradasi, dalam bidang forensik untuk memperbaiki gambar dari rekaman yang rusak, atau dalam fotografi untuk memulihkan foto yang blur atau terkena noise. Restorasi citra memanfaatkan pengetahuan tentang jenis kerusakan citra yang terjadi dan menerapkan algoritma pemrosesan untuk meminimalkan dampak degradasi tersebut.

- **Kompresi citra (image compression)**

Kompresi citra (image compression) adalah teknik yang digunakan untuk mengurangi ukuran file gambar, sehingga memungkinkan efisiensi dalam penyimpanan dan transmisi data. Kompresi citra dapat dibagi menjadi dua jenis utama: kompresi tanpa kehilangan data (*lossless*) dan kompresi dengan kehilangan data (*lossy*). Pada kompresi *lossless*, proses ini mengurangi ukuran file tanpa menghilangkan informasi apa pun dari citra, sehingga citra yang didekompresi akan identik dengan citra asli. Contoh algoritma kompresi *lossless* adalah PNG dan TIFF. Sebaliknya, pada kompresi *lossy*, sebagian informasi citra akan dibuang untuk mencapai tingkat kompresi yang lebih tinggi, yang dapat menyebabkan penurunan kualitas gambar, tetapi dengan pengurangan ukuran file yang signifikan. Format kompresi *lossy* yang paling umum adalah JPEG, yang digunakan secara luas di web dan media digital. Kompresi citra sangat penting dalam berbagai aplikasi, seperti pengiriman gambar melalui internet, penyimpanan gambar dalam perangkat dengan kapasitas terbatas, atau pemrosesan citra dalam sistem yang

membutuhkan pengurangan waktu transmisi data. Pemilihan jenis kompresi yang digunakan tergantung pada kebutuhan aplikasi, apakah lebih mengutamakan kualitas gambar atau efisiensi ruang penyimpanan..

#### • **Pengenalan objek dan pola**

Pengenalan objek dan pola adalah proses dalam pengolahan citra digital yang bertujuan untuk mengidentifikasi dan mengklasifikasikan objek-objek dalam gambar secara otomatis menggunakan teknik pembelajaran mesin (machine learning) dan kecerdasan buatan (artificial intelligence). Dalam proses ini, algoritma dilatih untuk mengenali berbagai pola, fitur, dan karakteristik dalam citra yang memungkinkan sistem untuk membedakan antara objek yang satu dengan yang lainnya. Misalnya, dalam pengenalan wajah, algoritma dapat dilatih untuk mengenali fitur khas wajah manusia seperti mata, hidung, dan mulut, dan mengklasifikasikannya berdasarkan ciri-ciri tersebut. Teknik yang digunakan dalam pengenalan objek dan pola sering kali melibatkan metode seperti jaringan syaraf tiruan (neural networks), terutama *deep learning* dengan Convolutional Neural Networks (CNN), yang telah terbukti sangat efektif dalam tugas pengenalan citra. Pengenalan objek dan pola banyak diterapkan dalam berbagai bidang, seperti sistem keamanan untuk pengenalan wajah, kendaraan otonom untuk deteksi objek di jalan, dan dalam bidang medis untuk mendeteksi kelainan atau penyakit berdasarkan citra medis. Keberhasilan pengenalan objek dan pola sangat bergantung pada kualitas data latih yang digunakan dan kompleksitas model yang diterapkan.

### **Tahapan dalam Pengolahan Citra Digital**

1. **Akuisisi Citra (Image Acquisition)**  
Proses pertama ini mencakup penangkapan gambar melalui perangkat input seperti kamera digital, scanner, atau sensor khusus. Citra kemudian dikonversi ke format digital berupa piksel yang memiliki nilai intensitas tertentu.
2. **Prapemrosesan (Preprocessing)**  
Tahap ini bertujuan meningkatkan kualitas citra sebelum dianalisis lebih lanjut. Teknik yang sering digunakan meliputi normalisasi intensitas, penghapusan noise menggunakan filter (median, Gaussian, bilateral), dan koreksi pencahayaan.
3. **Segmentasi (Segmentation)**  
Proses pemisahan objek atau wilayah penting dalam citra dari latar belakang atau bagian lainnya. Teknik segmentasi umum meliputi thresholding, edge detection, region growing, dan metode berbasis deep learning seperti U-Net.
4. **Ekstraksi Fitur (Feature Extraction)**  
Setelah segmentasi, dilakukan pengambilan fitur atau ciri khas objek yang akan digunakan untuk proses analisis atau pengenalan. Contohnya adalah fitur bentuk (kontur), tekstur (GLCM), warna (histogram), serta fitur berbasis transformasi (Fourier, Wavelet).
5. **Klasifikasi dan Interpretasi (Classification and Interpretation)**  
Berdasarkan fitur yang telah diekstrak, objek dalam citra diklasifikasikan menggunakan model seperti K-Nearest Neighbor, Support Vector Machine, atau model deep learning seperti Convolutional Neural Network (CNN). Hasil klasifikasi digunakan untuk pengambilan keputusan atau visualisasi.
6. **Pasca-Pemrosesan (Postprocessing)**  
Tahap akhir ini bertujuan memperbaiki hasil pemrosesan atau menambahkan

informasi tambahan, seperti memberi label, membuat bounding box, atau overlay hasil deteksi pada citra asli.

### Aplikasi Pengolahan Citra Digital

Pengolahan citra digital memiliki aplikasi yang sangat luas di berbagai bidang:

- **Bidang Medis:** Deteksi dan klasifikasi penyakit melalui citra X-ray, CT-scan, MRI, dan USG. Contoh: deteksi tumor otak, segmentasi organ, klasifikasi sel kanker.
- **Industri Manufaktur:** Kontrol kualitas produk secara otomatis melalui citra kamera pada lini produksi. Misalnya, mendeteksi cacat pada permukaan logam atau keretakan pada bahan bangunan.
- **Keamanan dan Forensik:** Sistem pengenalan wajah, deteksi plat nomor kendaraan (ANPR), hingga analisis forensik digital pada gambar dan video.
- **Pertanian dan Lingkungan:** Pemantauan kesehatan tanaman dengan citra satelit atau drone, deteksi area kekeringan, atau klasifikasi tutupan lahan.
- **Transportasi dan Navigasi:** Sistem bantuan pengemudi (ADAS), pengenalan rambu lalu lintas, deteksi jalur, dan kendaraan otonom.
- **Astronomi:** Meningkatkan citra dari teleskop luar angkasa dan mengidentifikasi benda langit secara otomatis.

### Software dan Bahasa Pemrograman Umum

Beberapa software dan pustaka yang sering digunakan dalam pengolahan citra digital antara lain:

- **OpenCV (Open Source Computer Vision Library)** – pustaka C++/Python populer untuk pemrosesan citra dan visi computer (Guillen, 2019).
- **MATLAB** – banyak digunakan dalam bidang akademik dan penelitian dengan dukungan toolbox khusus citra digital.
- **Python + Pustaka Tambahan** seperti Pillow, Scikit-image, TensorFlow/Keras (untuk deep learning).
- **ImageJ** – aplikasi open-source yang banyak digunakan dalam bidang bioinformatika dan analisis mikroskop.

## 2.2. Deteksi Warna

Deteksi warna merupakan teknik untuk mengidentifikasi dan memisahkan bagian tertentu dari citra berdasarkan informasi warna. Warna pada citra digital biasanya direpresentasikan dalam ruang warna seperti RGB (Red, Green, Blue) atau HSV (Hue, Saturation, Value). Pada ruang HSV, pemisahan warna lebih mudah dilakukan karena tiap warna memiliki rentang hue yang spesifik.

Penggunaan ruang warna HSV sangat umum dalam deteksi warna karena memisahkan informasi warna (hue) dari pencahayaan (value) dan kejenuhan (saturation). Hal ini memungkinkan sistem pengolahan citra untuk mendeteksi warna dengan lebih konsisten meskipun terjadi perubahan pencahayaan atau bayangan. Misalnya, warna merah dapat dideteksi dengan menentukan rentang nilai hue tertentu, seperti 0–10 derajat dan 160–180 derajat pada lingkaran warna HSV (Goenawan et al., 2022).

Dalam implementasinya, **deteksi warna** biasanya dilakukan melalui beberapa tahapan berikut:

1. **Konversi warna citra dari ruang RGB ke HSV**  
Citra digital biasanya disimpan dalam format RGB, namun untuk deteksi warna, ruang HSV lebih disukai karena memisahkan informasi warna (Hue) dari intensitas cahaya (Value). Konversi ini memungkinkan sistem untuk mengenali warna secara lebih akurat meskipun terdapat variasi pencahayaan. Misalnya, objek biru dalam kondisi terang dan redup masih bisa dikenali karena nilai Hue-nya tetap stabil (Marso & El Merouani, 2020).
2. **Penentuan rentang warna (thresholding) berdasarkan nilai Hue, Saturation, dan Value**  
Setelah citra berada dalam ruang HSV, langkah selanjutnya adalah menentukan rentang nilai HSV yang mewakili warna target. Misalnya, untuk mendeteksi warna merah, ditentukan batas bawah dan batas atas nilai Hue, Saturation, dan Value. Proses ini disebut *thresholding*, dan hasilnya adalah sebuah range warna yang akan digunakan untuk menyaring piksel.
3. **Penerapan masker biner untuk menyorot piksel yang sesuai dengan kriteria warna**  
Masker biner adalah citra hitam-putih (0 dan 1) yang menunjukkan area di mana warna target terdeteksi. Piksel yang berada dalam rentang HSV akan ditandai sebagai putih (1), sedangkan piksel lainnya sebagai hitam (0). Ini menghasilkan citra baru yang hanya berisi bagian-bagian dari objek berwarna yang diinginkan, sehingga bagian tersebut dapat diisolasi dari latar belakang.
4. **Ekstraksi objek yang telah tersegmentasi berdasarkan warna**  
Setelah piksel target dipisahkan melalui masker, langkah berikutnya adalah mengekstrak objek secara menyeluruh. Biasanya digunakan operasi morfologi seperti:
  - **Erosi**, untuk menghilangkan noise atau piksel kecil yang tidak diinginkan.
  - **Dilasi**, untuk memperbesar area objek yang relevan.
  - Kombinasi **opening dan closing**, untuk membersihkan hasil segmentasi dan memperhalus bentuk objek.

Tahapan ini penting untuk meningkatkan kualitas segmentasi dan memastikan objek yang terdeteksi memiliki bentuk yang utuh dan jelas.

5. **Analisis atau pelacakan objek berdasarkan hasil deteksi warna**  
Setelah objek berhasil dideteksi dan diekstraksi, hasilnya dapat digunakan untuk berbagai tujuan seperti pelacakan gerak, penghitungan jumlah objek, atau sebagai masukan untuk sistem kecerdasan buatan. Misalnya, dalam sistem robotika, robot dapat mengikuti garis berwarna tertentu atau mengambil objek berdasarkan warna yang terdeteksi.

Deteksi warna banyak digunakan dalam berbagai aplikasi seperti pelacakan objek, pengenalan isyarat tangan, sistem penglihatan robotik, dan pemrosesan gambar medis. Teknik ini sederhana namun sangat efektif ketika objek memiliki warna yang kontras terhadap latar belakang.



### 2.3. Histogram Citra

Histogram citra adalah representasi grafik dari distribusi intensitas piksel pada citra. Setiap sumbu horizontal menunjukkan nilai intensitas (biasanya dari 0 hingga 255), sedangkan sumbu vertikal menunjukkan jumlah piksel yang memiliki nilai intensitas tersebut. Histogram digunakan untuk mengetahui sebaran warna dan kontras gambar (Fakultas et al., 2020).

Histogram dapat diterapkan pada citra grayscale maupun citra berwarna:

- **Pada citra grayscale**, histogram menunjukkan distribusi tingkat kecerahan (dari hitam ke putih).
- **Pada citra berwarna**, histogram biasanya dibagi menjadi tiga saluran warna (Red, Green, Blue) yang masing-masing memiliki histogram sendiri.

Dengan menganalisis histogram, kita dapat mengetahui karakteristik pencahayaan dan kontras suatu citra:

- Jika histogram terkonsentrasi di sisi kiri, citra cenderung gelap.
- Jika terkonsentrasi di sisi kanan, citra cenderung terang.
- Jika tersebar merata, citra memiliki kontras yang baik.

Beberapa **penggunaan histogram dalam pengolahan citra** antara lain:

#### 1. **Peningkatan kontras (contrast enhancement)**

Histogram digunakan dalam teknik seperti **histogram equalization** untuk menyebarkan nilai intensitas secara lebih merata di seluruh rentang (0–255). Jika suatu citra memiliki kontras rendah (histogramnya sempit dan terpusat pada area tertentu), equalization akan menyebarkan nilai intensitas tersebut agar mencakup rentang yang lebih luas. Hasilnya adalah citra dengan perbedaan terang-gelap yang lebih jelas, sehingga detail menjadi lebih mudah terlihat.

#### 2. **Segmentasi citra**

Histogram sangat membantu dalam proses **thresholding**, yaitu menentukan batas nilai intensitas yang memisahkan objek dari latar belakang. Misalnya, pada citra dengan latar belakang gelap dan objek terang, histogram akan menunjukkan dua puncak (bimodal). Dengan melihat jarak antar puncak tersebut, dapat ditentukan ambang batas optimal untuk memisahkan kedua area. Ini banyak digunakan dalam pengenalan objek, OCR (Optical Character Recognition), dan analisis medis.

#### 3. **Deteksi tepi atau fitur (feature detection)**

Meskipun bukan metode langsung, histogram bisa digunakan untuk menganalisis adanya perubahan tajam dalam distribusi intensitas yang mengindikasikan **tepi objek** atau **perubahan tekstur**. Sebagai contoh, perbedaan drastis antara area gelap dan terang bisa menjadi indikasi adanya batas objek. Ini dapat dikombinasikan dengan metode lain seperti Sobel atau Canny edge detection.

#### 4. **Normalisasi pencahayaan**

Dalam banyak aplikasi, seperti visi komputer atau pengenalan wajah, gambar bisa diambil dalam kondisi pencahayaan yang berbeda. Dengan **membandingkan histogram** dari dua gambar, sistem dapat menyesuaikan intensitas (melalui teknik normalisasi histogram) agar pencahayaan menjadi seragam. Hal ini penting agar

sistem dapat mendeteksi objek secara konsisten, meskipun dalam kondisi cahaya yang berubah-ubah.

Dengan memahami histogram citra, kita dapat melakukan berbagai manipulasi dan analisis gambar secara lebih efektif karena memiliki informasi statistik dasar dari intensitas piksel dalam citra tersebut.

#### 2.4. Ambang Batas (Thresholding)

Ambang batas (*thresholding*) adalah metode dalam pengolahan citra yang digunakan untuk memisahkan objek dari latar belakang berdasarkan tingkat intensitas piksel. Teknik ini mengubah citra grayscale menjadi citra biner, yaitu hanya terdiri dari dua nilai: putih (1) untuk objek dan hitam (0) untuk latar belakang (Zahra Salsa Azizah et al., 2024).

Proses thresholding dilakukan dengan menentukan nilai ambang (threshold) tertentu, lalu setiap piksel dalam citra dibandingkan dengan nilai tersebut:

- Jika nilai piksel lebih besar atau sama dengan threshold, maka piksel dianggap bagian dari objek.
- Jika nilai piksel lebih kecil dari threshold, maka piksel dianggap sebagai latar belakang.

Thresholding memiliki beberapa variasi, di antaranya:

##### 1. Thresholding

##### Global

Thresholding global menggunakan **satu nilai ambang tetap** yang diterapkan ke seluruh piksel dalam citra. Metode ini sederhana dan cepat, serta sangat efektif apabila kondisi pencahayaan citra merata dan objek memiliki kontras tinggi terhadap latar belakang.

Salah satu metode populer dalam thresholding global adalah **Otsu's method**, yang secara otomatis menghitung nilai ambang optimal dengan meminimalkan *intra-class variance* (persebaran dalam kelas) dan memaksimalkan *inter-class variance* (perbedaan antara objek dan latar). Namun, metode ini menjadi kurang efektif pada citra dengan pencahayaan tidak merata, karena satu nilai threshold tidak dapat memisahkan objek dan latar belakang secara konsisten di seluruh area.

##### 2. Thresholding

##### Lokal

##### (Adaptif)

Pada thresholding lokal atau adaptif, nilai ambang **dihitung secara dinamis berdasarkan wilayah kecil** (blok atau jendela) dalam citra. Setiap area lokal memiliki ambang tersendiri, biasanya berdasarkan rata-rata atau median intensitas piksel di wilayah tersebut.

Teknik ini sangat cocok untuk citra yang memiliki **pencahayaan tidak seragam**, seperti dokumen dengan bayangan, pantulan cahaya, atau foto dengan gradasi gelap-terang.

Beberapa metode adaptif antara lain:

- *Mean adaptive thresholding*: threshold dihitung berdasarkan rata-rata intensitas lokal.
- *Gaussian adaptive thresholding*: threshold dihitung berdasarkan rata-rata tertimbang (dengan bobot Gaussian).

### 3. **Thresholding Multilevel (Multithresholding)**

Multilevel thresholding digunakan untuk citra yang mengandung lebih dari dua kelas piksel. Artinya, bukan hanya membedakan objek dan latar belakang, tapi juga mungkin terdapat kelas tambahan seperti **noise**, **tepi objek**, atau **area transisi**. Metode ini menetapkan **lebih dari satu nilai ambang**, sehingga citra dibagi menjadi beberapa rentang intensitas, masing-masing mewakili kelas yang berbeda. Misalnya, intensitas 0–85 untuk latar belakang, 86–170 untuk objek, dan 171–255 untuk bagian terang atau noise. Teknik ini sering digunakan dalam analisis citra medis, segmentasi citra satelit, atau situasi di mana objek dan latar tidak memiliki batas intensitas yang tegas.

Masing-masing metode thresholding memiliki keunggulan dan keterbatasan, tergantung pada kondisi citra yang diolah. Oleh karena itu, pemilihan jenis thresholding yang tepat sangat penting untuk menghasilkan segmentasi yang akurat.

#### 2.5. Backlight dan Peningkatan Kualitas Citra

**Backlight** adalah kondisi pencahayaan di mana **sumber cahaya berada di belakang objek utama**, menyebabkan objek terlihat gelap atau bahkan menjadi siluet. Situasi ini umum terjadi dalam pengambilan gambar di luar ruangan, seperti saat objek difoto dengan latar belakang langit terang, jendela, atau lampu. Karena sensor kamera menangkap cahaya latar lebih dominan, detail objek di bagian depan seringkali hilang atau kurang terlihat (*MDP STUDENT CONFERENCE (MSC) 2022*, n.d.).

Untuk mengatasi masalah ini, digunakan berbagai **teknik peningkatan kualitas citra** agar informasi visual dari objek tetap dapat ditampilkan dengan jelas. Beberapa teknik yang umum digunakan antara lain:

##### 1. **Peningkatan Kontras (Contrast Enhancement)**

Tujuannya adalah untuk memperbesar perbedaan antara area gelap dan terang dalam citra, sehingga kontur dan detail objek menjadi lebih jelas. Teknik seperti *Contrast Stretching* dan *Histogram Equalization* sering digunakan untuk menyebarkan distribusi intensitas piksel agar tampak lebih seimbang.

##### 2. **Penyesuaian Kecerahan (Brightness Adjustment)**

Dengan menaikkan tingkat kecerahan, piksel-piksel gelap di area objek dapat dibuat lebih terang, sehingga informasi visual dapat diungkap tanpa kehilangan terlalu banyak detail di area terang. Namun, penyesuaian ini perlu hati-hati agar tidak menyebabkan *overexposure* (terlalu terang) pada latar belakang (Das et al., 2015).

##### 3. **CLAHE (Contrast Limited Adaptive Histogram Equalization)**

Teknik ini merupakan bentuk adaptif dari histogram equalization yang

membagi citra menjadi beberapa wilayah kecil dan meningkatkan kontras lokal pada masing-masing wilayah. CLAHE sangat efektif untuk menangani kondisi pencahayaan tidak merata seperti backlight, karena tidak menyebabkan noise berlebihan di area terang atau gelap .

#### 4. **Peningkatan lokal (local enhancement)**

Teknik ini memperkuat area tertentu dalam citra yang dianggap penting (misalnya wajah seseorang dalam bayangan), tanpa mempengaruhi area lain secara keseluruhan. Biasanya menggunakan *region of interest* (ROI) dan metode pemrosesan berbasis filter atau segmentasi.

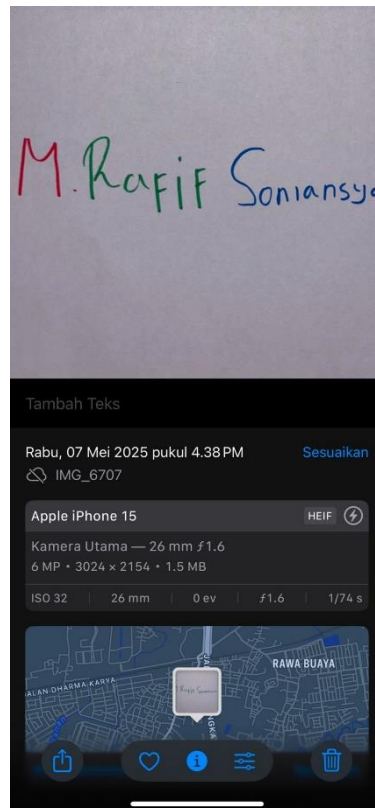
#### 5. **Penggunaan Filter Adaptif dan Masking**

Filter seperti *unsharp masking*, *bilateral filter*, atau *guided filter* dapat digunakan untuk mempertajam dan memperjelas kontur objek sambil tetap mempertahankan detail. Beberapa metode juga menggunakan segmentasi warna atau deteksi tepi untuk mengisolasi dan memperbaiki bagian objek yang terkena bayangan backlight.

Teknik-teknik ini sering dikombinasikan dalam sistem pengolahan citra otomatis, seperti pada kamera pengawas, pengenalan wajah, sistem parkir otomatis, dan pengolahan citra medis, agar citra tetap informatif meskipun berada dalam kondisi pencahayaan yang menantang.

## BAB III HASIL

### 3.1. Deteksi Warna Pada Citra



Penjelasan :

Langkah pertama dalam proses pengolahan citra adalah **membaca gambar dari file**

```

jupyter Deteksi Warna Last Checkpoint: 21 hours ago
File Edit View Run Kernel Settings Help
JupyterLab Python [conda env:base]

[1]: import cv2
import numpy as np
import matplotlib.pyplot as plt

[2]: # Baca gambar
img = cv2.imread('Nama.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Konversi ke HSV
hsv_image = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# Salin gambar untuk setiap warna
blue_faded = img_rgb.copy()
red_faded = img_rgb.copy()
green_faded = img_rgb.copy()

```

menggunakan fungsi bawaan pustaka seperti OpenCV. Setelah itu, karena OpenCV secara default menyimpan gambar dalam format **BGR (Blue, Green, Red)**, gambar perlu dikonversi ke format **RGB (Red, Green, Blue)** agar dapat ditampilkan dengan benar menggunakan pustaka visualisasi seperti *matplotlib*. Selanjutnya, gambar dikonversi dari ruang warna RGB ke **HSV (Hue, Saturation, Value)**, karena HSV lebih cocok digunakan dalam proses **deteksi warna**. Ruang warna HSV memisahkan informasi warna (hue) dari intensitas (saturation dan value), sehingga mempermudah dalam mengekstraksi warna tertentu dari citra. Lalu setiap salinan akan dimodifikasi sesuai dengan warna target (biru, merah, hijau).

```

blue_faded = img_rgb.copy()
red_faded = img_rgb.copy()
green_faded = img_rgb.copy()

[7]: # === Deteksi Warna Biru ===
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([140, 255, 255])
blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)
blue_faded[blue_mask > 0] = blue_faded[blue_mask > 0] // 2 # Efek pudar

# === Deteksi Warna Merah ===
lower_red1 = np.array([0, 120, 70])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 120, 70])
upper_red2 = np.array([180, 255, 255])
red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
red_mask = cv2.bitwise_or(red_mask1, red_mask2)
red_faded[red_mask > 0] = red_faded[red_mask > 0] // 2 # Efek pudar

# === Deteksi Warna Hijau ===
lower_green = np.array([100, 70, 70])
upper_green = np.array([140, 255, 255])
green_mask = cv2.inRange(hsv_image, lower_green, upper_green)
green_faded[green_mask > 0] = green_faded[green_mask > 0] // 2 # Efek pudar

# === Tampilkan Hasil ===
fig, axes = plt.subplots(2, 2, figsize=(10, 8))

axes[0, 0].imshow(img_rgb)
axes[0, 0].set_title("Citra Kontras (Asli)")
axes[0, 0].axis('off')

axes[0, 1].imshow(cv2.cvtColor(blue_faded, cv2.COLOR_RGB2GRAY), cmap='gray', vmin=0, vmax=255)
axes[0, 1].set_title("BIRU (Soniansyah)")
axes[0, 1].axis('off')

axes[1, 0].imshow(cv2.cvtColor(red_faded, cv2.COLOR_RGB2GRAY), cmap='gray', vmin=0, vmax=255)
axes[1, 0].set_title("MERAH (H.)")
axes[1, 0].axis('off')

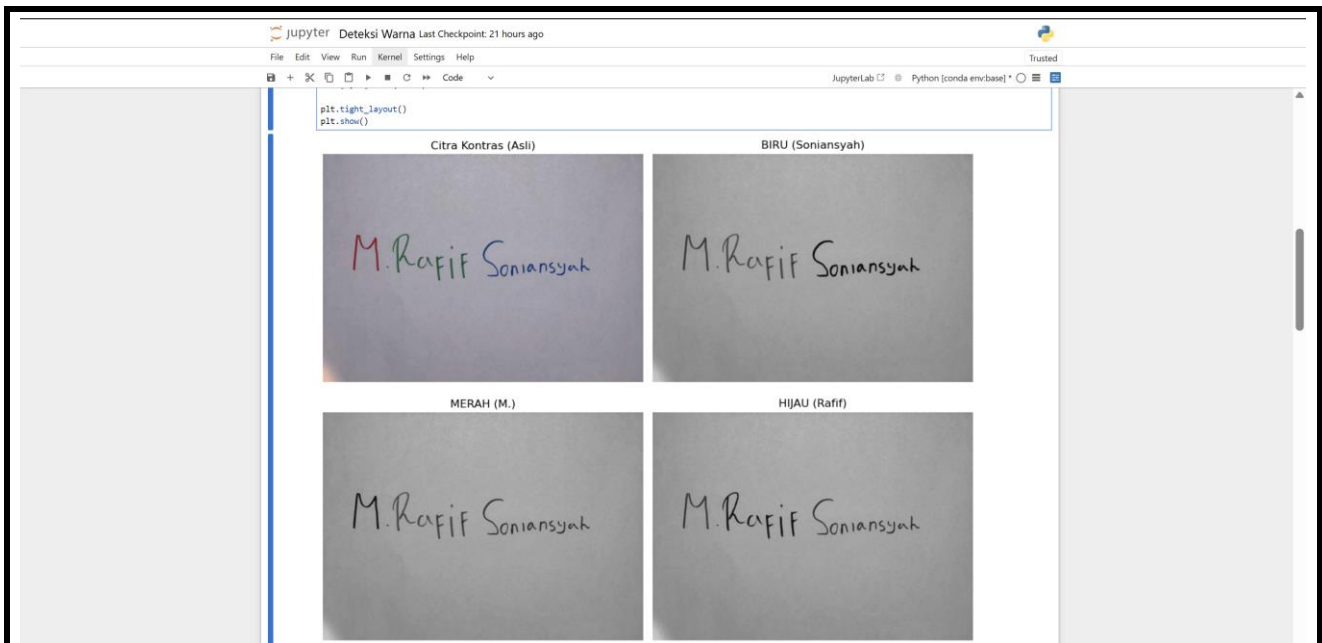
axes[1, 1].imshow(cv2.cvtColor(green_faded, cv2.COLOR_RGB2GRAY), cmap='gray', vmin=0, vmax=255)
axes[1, 1].set_title("HIJAU (Rafif)")
axes[1, 1].axis('off')

plt.tight_layout()
plt.show()

```

Penjelasan :

Langkah selanjutnya dalam proses deteksi warna adalah **menentukan rentang HSV untuk masing-masing warna**, dimulai dari warna biru. Dengan menggunakan fungsi `cv2.inRange`, dibuat sebuah **masker biner** yang menyoroti piksel-piksel yang termasuk dalam rentang warna biru tersebut. Setelah itu, piksel-piksel biru yang terdeteksi **dipudarkan** dengan cara membagi nilai RGB-nya agar lebih gelap dan kontras terhadap latar belakang. Untuk warna merah, karena merah berada di dua ujung rentang hue HSV (yaitu di sekitar  $0^\circ$  dan  $180^\circ$ ), maka perlu dibuat **dua masker terpisah**, kemudian hasil keduanya digabung untuk mendapatkan deteksi warna merah secara utuh. Piksel merah yang terdeteksi juga diproses dengan teknik pemudaran yang sama. Proses serupa diterapkan pula untuk warna hijau, dengan menentukan rentang hue khusus untuk hijau dan membuat masker biner yang sesuai. Setelah semua warna berhasil dideteksi, hasilnya disusun dalam sebuah **grid 2x2** untuk ditampilkan: posisi [0,0] berisi gambar asli, [0,1] menampilkan hasil deteksi warna biru, [1,0] untuk deteksi merah, dan [1,1] untuk hijau. Seluruh hasil deteksi warna ditampilkan dalam **skala abu-abu (grayscale)** agar lebih mudah dianalisis secara visual, karena intensitas piksel putih pada citra menunjukkan area di mana warna yang dideteksi paling dominan.



Output :

### Penjelasan

#### 1. Citra Kontras (Asli)

Citra kontras yang ditampilkan pada gambar paling kiri atas merupakan citra asli yang menampilkan tulisan tangan dengan tiga warna yang berbeda. Dalam gambar tersebut, huruf "M." ditulis menggunakan warna merah, sedangkan kata "Rafif" ditulis dengan warna hijau. Sementara itu, "Soniansyah" dituliskan dengan warna biru. Perbedaan warna ini memberikan kontras visual yang jelas antar elemen tulisan.

#### 2. Biru (Soniansyah)

Gambar kanan atas merupakan hasil ekstraksi atau pendeteksian warna biru dari citra asli. Dalam citra ini, hanya bagian teks "Soniansyah" yang ditampilkan berwarna hitam gelap dan sangat tebal karena ditulis menggunakan warna biru, sesuai dengan rentang warna yang dideteksi. Sementara itu, bagian lain dari gambar menjadi abu – abu, karena tidak termasuk dalam spektrum warna biru yang dicari. Proses ini menyoroti elemen tertentu dalam citra berdasarkan warna spesifik, dalam hal ini warna biru.

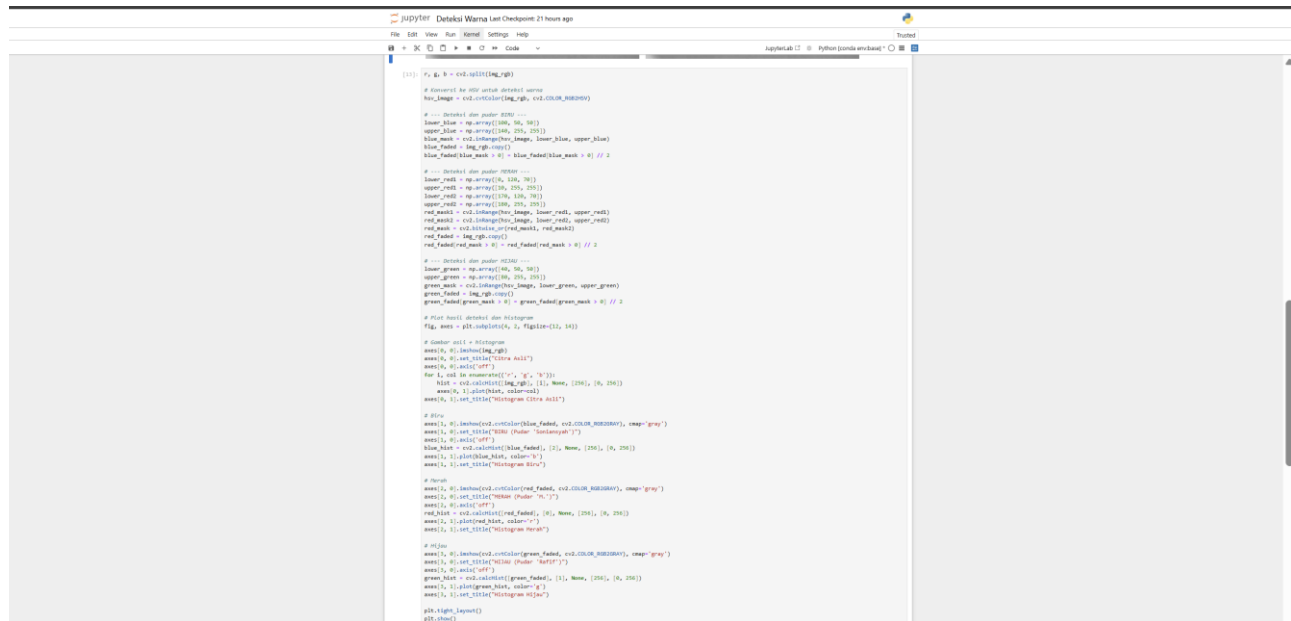
#### 3. Merah (M.)

Gambar kiri bawah merupakan hasil ekstraksi atau pendeteksian warna merah dari citra asli. Dalam citra ini, hanya bagian teks "M." yang ditampilkan berwarna hitam gelap dan sangat tebal karena ditulis menggunakan warna merah, sesuai dengan rentang warna yang dideteksi. Sementara itu, bagian lain dari gambar menjadi abu – abu, karena tidak termasuk dalam spektrum warna merah yang dicari. Proses ini menyoroti elemen tertentu dalam citra berdasarkan warna spesifik, dalam hal ini warna merah.

#### 4. Hijau (Rafif)

Gambar kanan bawah merupakan hasil ekstraksi atau pendeteksian warna hijau dari citra asli. Dalam citra ini, hanya bagian teks "Rafif" yang ditampilkan berwarna hitam gelap dan sangat tebal karena ditulis menggunakan warna hijau, sesuai dengan rentang warna yang dideteksi. Sementara itu, bagian lain dari gambar menjadi abu – abu, karena tidak termasuk dalam spektrum warna hijau yang dicari. Proses ini menyoroti elemen tertentu dalam citra berdasarkan warna spesifik, dalam hal ini warna hijau.

## MENAMPILKAN HISTOGRAM



```
[1]: g, b = cv2.split(img_rgb)

# Konversi ke HSV untuk deteksi warna
img_hsv = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2HSV)

# --- Deteksi dan masker biru ---
lower_blue = np.array([110, 50, 50])
upper_blue = np.array([130, 255, 255])
blue_mask = cv2.inRange(img_hsv, lower_blue, upper_blue)
blue_faded = img_rgb * (1 - blue_mask)

# --- Deteksi dan masker merah ---
lower_red = np.array([0, 150, 150])
upper_red = np.array([10, 255, 255])
lower_red2 = np.array([170, 150, 150])
upper_red2 = np.array([190, 255, 255])
red_mask1 = cv2.inRange(img_hsv, lower_red, upper_red)
red_mask2 = cv2.inRange(img_hsv, lower_red2, upper_red2)
red_mask = red_mask1 | red_mask2
red_faded = img_rgb * (1 - red_mask)

# --- Deteksi dan masker hijau ---
lower_green = np.array([40, 50, 50])
upper_green = np.array([60, 255, 255])
green_mask = cv2.inRange(img_hsv, lower_green, upper_green)
green_faded = img_rgb * (1 - green_mask)

# Plot hasil deteksi dan histogram
fig, axes = plt.subplots(3, 1, figsize=(10, 10))

# Gambar asli
axes[0].imshow(img_rgb)
axes[0].set_title('Gambar Asli')

# Masker biru
axes[1].imshow(blue_mask)
axes[1].set_title('Masker Biru')

# Masker merah
axes[2].imshow(red_mask)
axes[2].set_title('Masker Merah')

# Histogram
# Biru
axes[3].imshow(cv2.cvtColor(blue_faded, cv2.COLOR_BGR2GRAY), cmap='gray')
axes[3].set_title('Histogram Biru')
hist, bins = np.histogram(blue_faded.ravel(), bins=256, range=(0, 256))
axes[3].plot(bins, hist, color='b')
axes[3].set_xlabel('Intensitas Biru')

# Merah
axes[4].imshow(cv2.cvtColor(red_faded, cv2.COLOR_BGR2GRAY), cmap='gray')
axes[4].set_title('Histogram Merah')
hist, bins = np.histogram(red_faded.ravel(), bins=256, range=(0, 256))
axes[4].plot(bins, hist, color='r')
axes[4].set_xlabel('Intensitas Merah')

# Hijau
axes[5].imshow(cv2.cvtColor(green_faded, cv2.COLOR_BGR2GRAY), cmap='gray')
axes[5].set_title('Histogram Hijau')
hist, bins = np.histogram(green_faded.ravel(), bins=256, range=(0, 256))
axes[5].plot(bins, hist, color='g')
axes[5].set_xlabel('Intensitas Hijau')

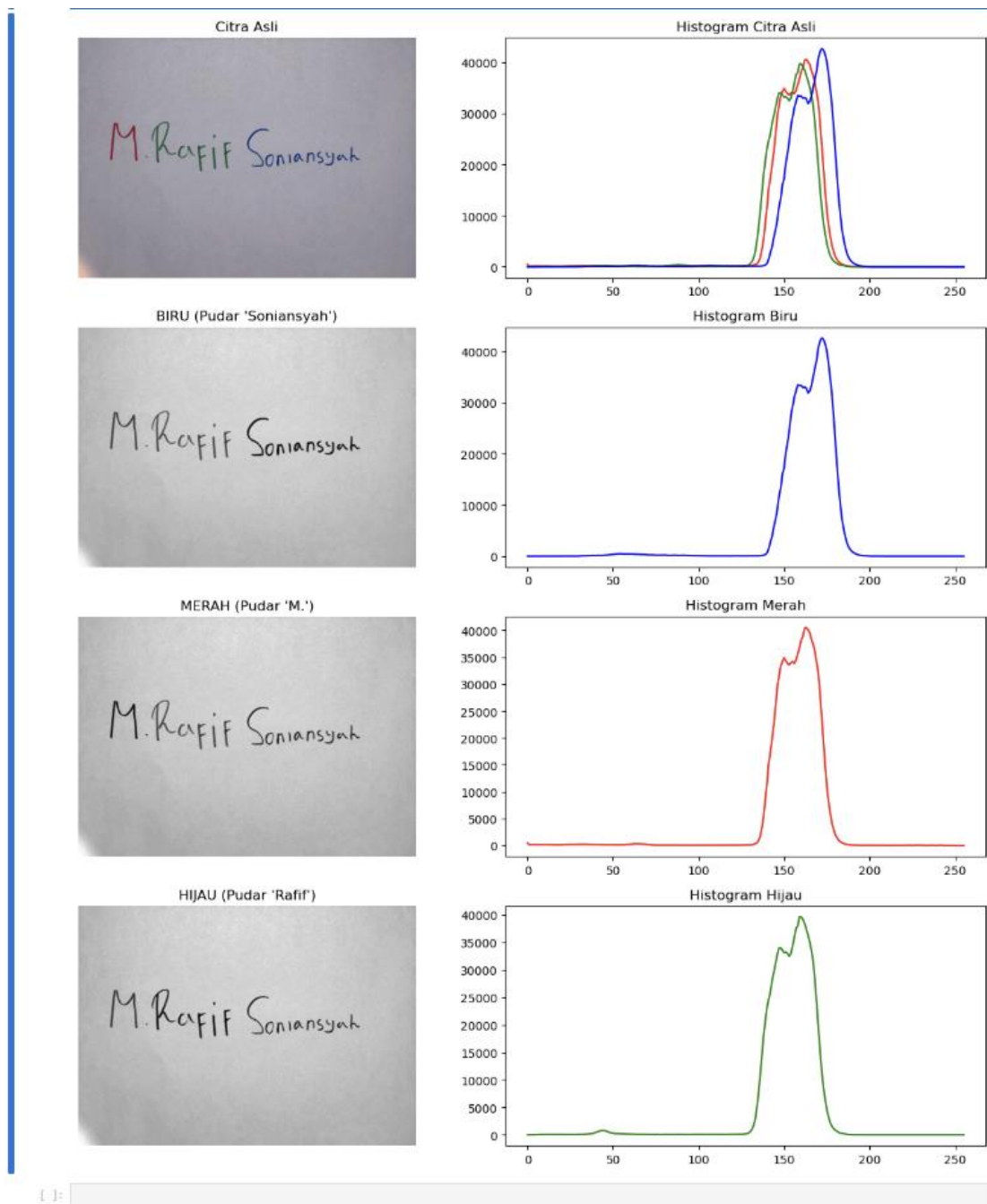
plt.tight_layout()
plt.show()
```

### Penjelasan:

Kode di atas merupakan proses deteksi warna dalam citra digital menggunakan OpenCV dan Matplotlib. Pertama, citra RGB dipisahkan menjadi tiga saluran warna (merah, hijau, biru), lalu dikonversi ke format HSV yang lebih efektif untuk deteksi warna. Selanjutnya, dilakukan deteksi warna biru, merah, dan hijau berdasarkan rentang HSV masing-masing. Area yang terdeteksi dari setiap warna dipudarkan (diperlemah intensitas warnanya) dengan membaginya dua, untuk menekankan keberadaannya. Hasil deteksi ini kemudian ditampilkan dalam bentuk gambar grayscale beserta histogram intensitas warna untuk setiap saluran yang dideteksi. Proses ini memungkinkan analisis visual dan statistik terhadap distribusi warna dalam sebuah citra.



Output :



Penjelasan :

Gambar output tersebut menampilkan hasil deteksi warna dari sebuah citra tulisan tangan berwarna merah, hijau, dan biru. Pada baris pertama ditampilkan citra asli beserta histogram RGB-nya, yang menunjukkan distribusi intensitas ketiga warna utama dalam gambar. Baris kedua hingga keempat masing-masing menunjukkan hasil pendeteksian warna biru, merah, dan hijau. Setiap deteksi warna menghasilkan gambar dengan bagian tulisan yang sesuai warna tersebut tampak lebih terang, sementara bagian lainnya tampak gelap atau pudar. Di samping masing-masing gambar, ditampilkan pula histogram warna yang menunjukkan sebaran intensitas dari warna yang terdeteksi. Secara keseluruhan, output ini menggambarkan proses identifikasi dan analisis visual terhadap komponen warna tertentu dalam citra, baik melalui tampilan visual maupun representasi grafik histogram.

### 3.2. Ambang Batas Terkecil sampai dengan Terbesar

```

Jupyter Notebook: Nilai Ambang Batas last checkpoint 22 hours ago
File Edit View Help Python 3 (conda env)

[1]: import cv2
import numpy as np
import matplotlib.pyplot as plt

[1]: image = cv2.imread('maw.jpg')
blue = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Buat background hitam
background = np.zeros_like(image)

# Fungsi untuk membuat mask
def get_hsv_mask(hsv, lower, upper):
    mask = cv2.inRange(hsv, np.array(lower), np.array(upper))
    return mask

# Mask biru (hue sekitar 100-140)
blue_mask = get_hsv_mask(blue, [100, 100, 100], [140, 255, 255])

# Mask merah (hue sekitar 0-10 dan 160-180)
red_mask = get_hsv_mask(blue, [0, 10, 10], [10, 255, 255])
red_mask = get_hsv_mask(blue, [160, 10, 10], [180, 255, 255])
red_mask = cv2.bitwise_or(red_mask, red_mask)

# Mask hijau (hue sekitar 35-85)
green_mask = get_hsv_mask(blue, [35, 10, 10], [85, 255, 255])

# Dilatasi mask hijau
kernel = np.ones((3, 3), np.uint8)
green_mask_dilated = cv2.dilate(green_mask, kernel, iterations=1)

# Gabungkan semua mask
blue_image = background.copy()
blue_image[blue_mask > 0] = [255, 255, 255]

# Mask biru
red_blue_mask = cv2.bitwise_or(red_mask, blue_mask)
red_blue_image = background.copy()
red_blue_image[red_blue_mask > 0] = [255, 255, 255]

# Mask merah + biru
red_green_blue_mask = cv2.bitwise_or(red_mask, green_mask_dilated)
red_green_blue_image = background.copy()
red_green_blue_image[red_green_blue_mask > 0] = [255, 255, 255]

# Tampilkan hasil
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(background, cv2.COLOR_BGR2RGB))
plt.title('NONE')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(cv2.cvtColor(blue_image, cv2.COLOR_BGR2RGB))
plt.title('BLUE')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(cv2.cvtColor(red_blue_image, cv2.COLOR_BGR2RGB))
plt.title('RED-BLUE')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(cv2.cvtColor(red_green_blue_image, cv2.COLOR_BGR2RGB))
plt.title('RED-GREEN-BLUE')
plt.axis('off')

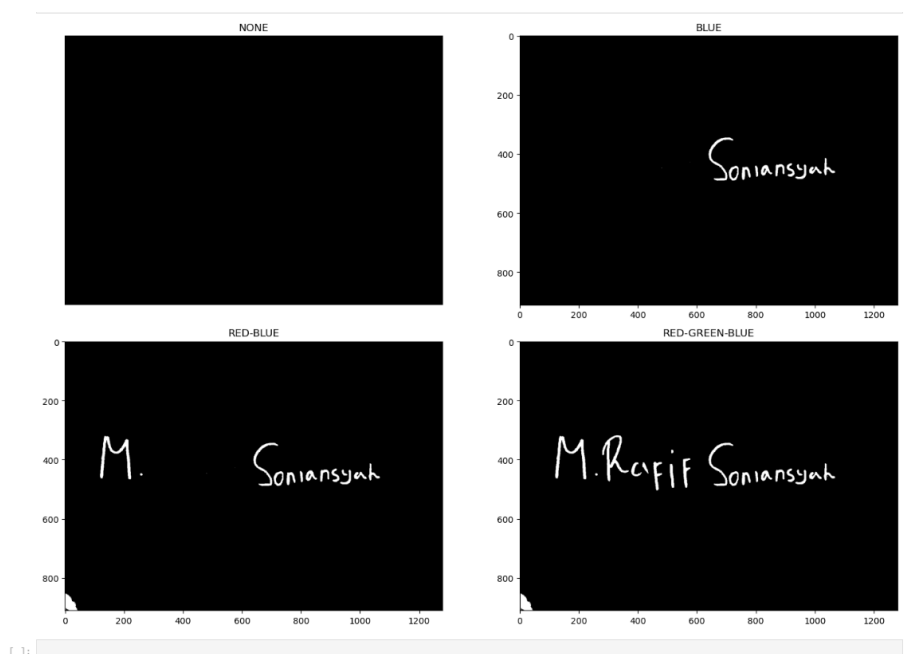
plt.tight_layout()
plt.show()

```

Penjelasan :

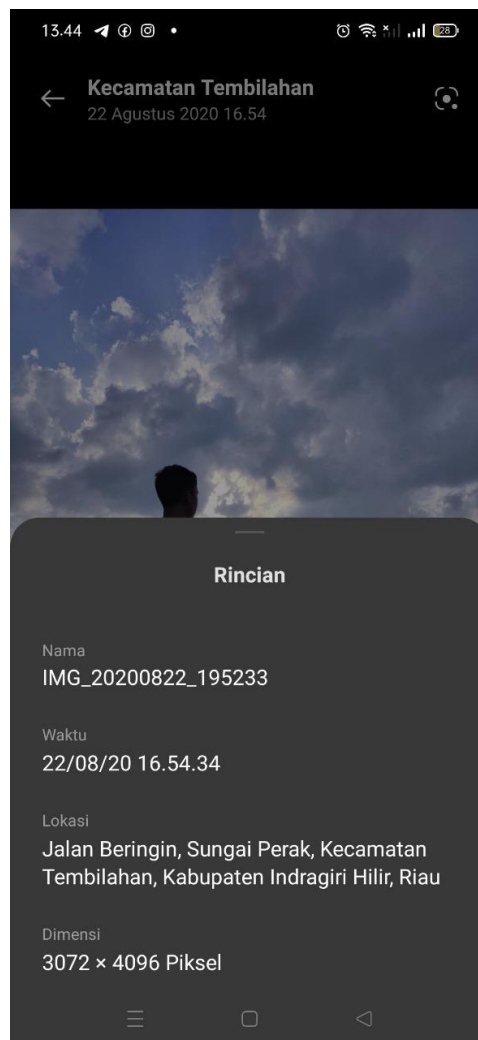
Kode di atas merupakan program Python menggunakan OpenCV dan Matplotlib untuk melakukan deteksi warna pada gambar, khususnya warna merah, hijau, dan biru. Proses dimulai dengan membaca citra dan mengubahnya ke dalam format HSV agar lebih mudah dalam mendeteksi warna berdasarkan hue (warna dasar). Selanjutnya, dibuat tiga jenis masker untuk masing-masing warna berdasarkan rentang hue-nya: biru (100–140), merah (terbagi dua rentang: 0–10 dan 160–180), dan hijau (dengan rentang agak lebar yaitu 35–85). Masker hijau kemudian mengalami proses dilasi agar hasil deteksi teks hijau menjadi lebih tegas. Kemudian dibuat tiga jenis gambar hasil: pertama hanya menampilkan warna biru, kedua menggabungkan warna merah dan biru, dan ketiga menggabungkan ketiganya (merah, hijau, biru). Setiap hasil ditampilkan sebagai gambar hitam-putih dengan latar belakang hitam dan warna yang terdeteksi berubah menjadi putih. Proses ini berguna untuk memisahkan elemen gambar berdasarkan warna dominan tertentu.

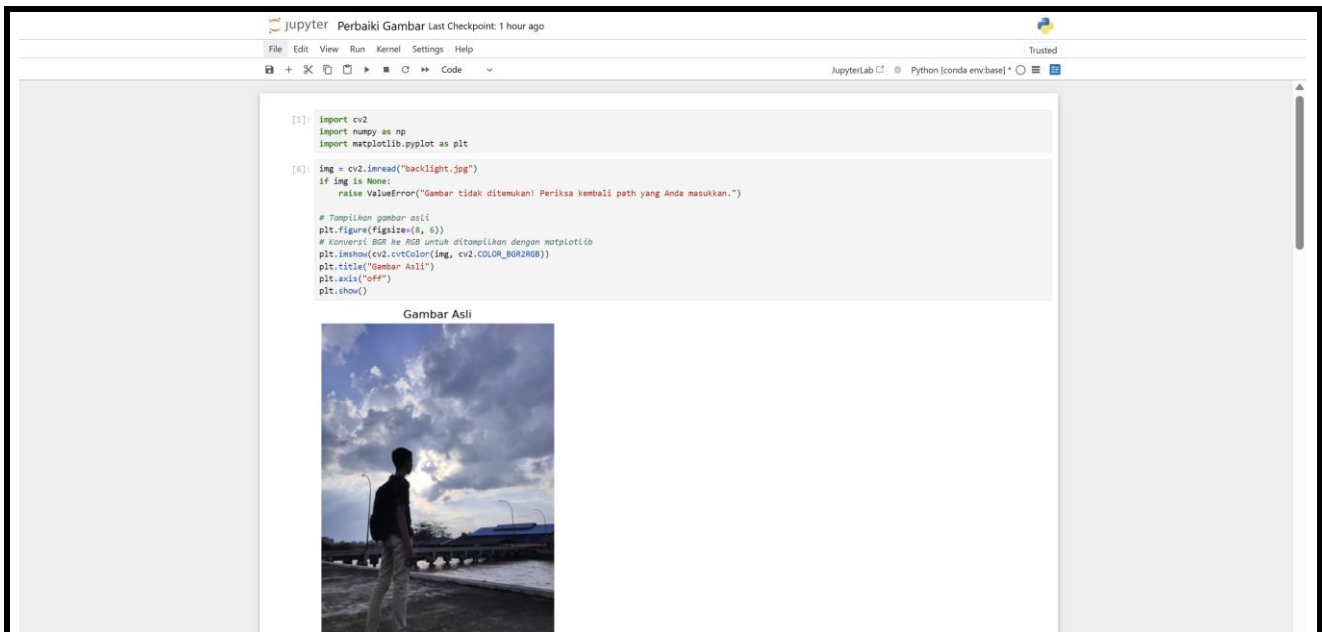
Output :



**Penjelasan :**

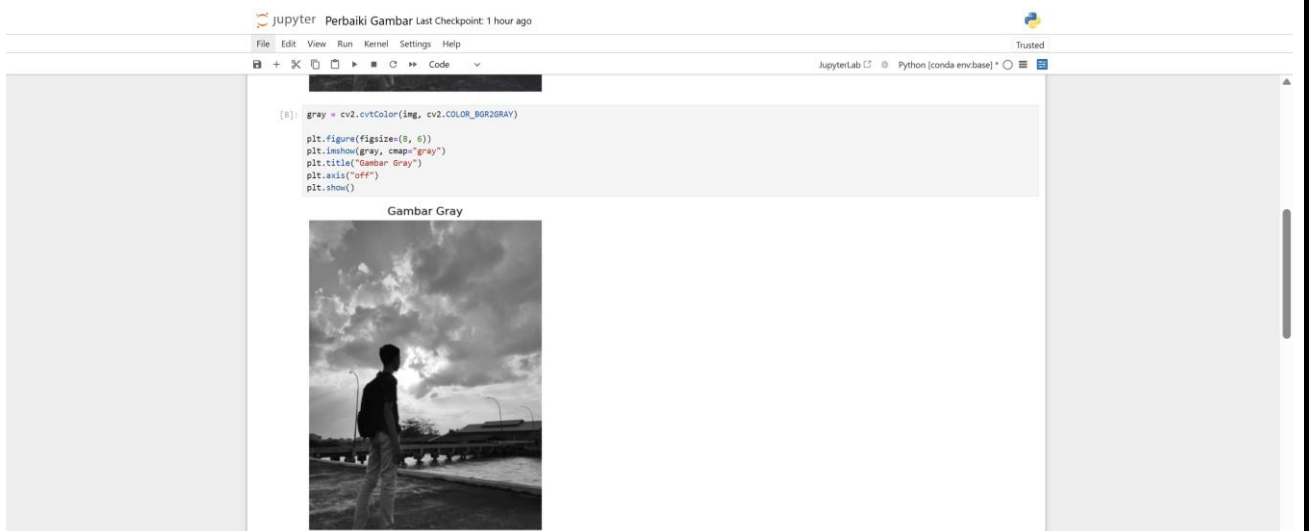
Output pada gambar tersebut menampilkan hasil dari proses deteksi warna tulisan dalam sebuah gambar berdasarkan warna merah, hijau, dan biru. Gambar pertama berjudul "NONE" menunjukkan latar belakang hitam tanpa bagian tulisan yang terdeteksi, karena belum ada warna yang disorot. Gambar kedua berjudul "BLUE" hanya menampilkan bagian tulisan berwarna biru (teks "Soniansyah") yang terdeteksi dan ditampilkan sebagai warna putih di atas latar hitam. Pada gambar ketiga, "RED-BLUE", terlihat dua bagian tulisan yang ditampilkan: "M." (merah) dan "Soniansyah" (biru), menunjukkan bahwa sistem berhasil menggabungkan deteksi warna merah dan biru. Gambar keempat, "RED-GREEN-BLUE", menampilkan keseluruhan tulisan "M. Rafif Soniansyah", karena semua bagian tulisan dari ketiga warna (merah, hijau, biru) berhasil terdeteksi. Hasil ini menunjukkan keberhasilan program dalam memisahkan dan mengekstraksi informasi berdasarkan warna tertentu dalam citra.

**3.3. Memperbaiki Gambar Backlight**



Penjelasan :

Menggunakan pustaka OpenCV (cv2) dan matplotlib. Gambar ini berjudul “Gambar Asli” dan menampilkan sosok seseorang yang berdiri membelakangi cahaya, mengarah ke langit dan bangunan di latar belakang. Dalam konteks pengolahan citra, ini adalah gambar dengan kondisi backlight, di mana objek utama tampak gelap karena cahaya yang kuat dari belakangnya. Langkah

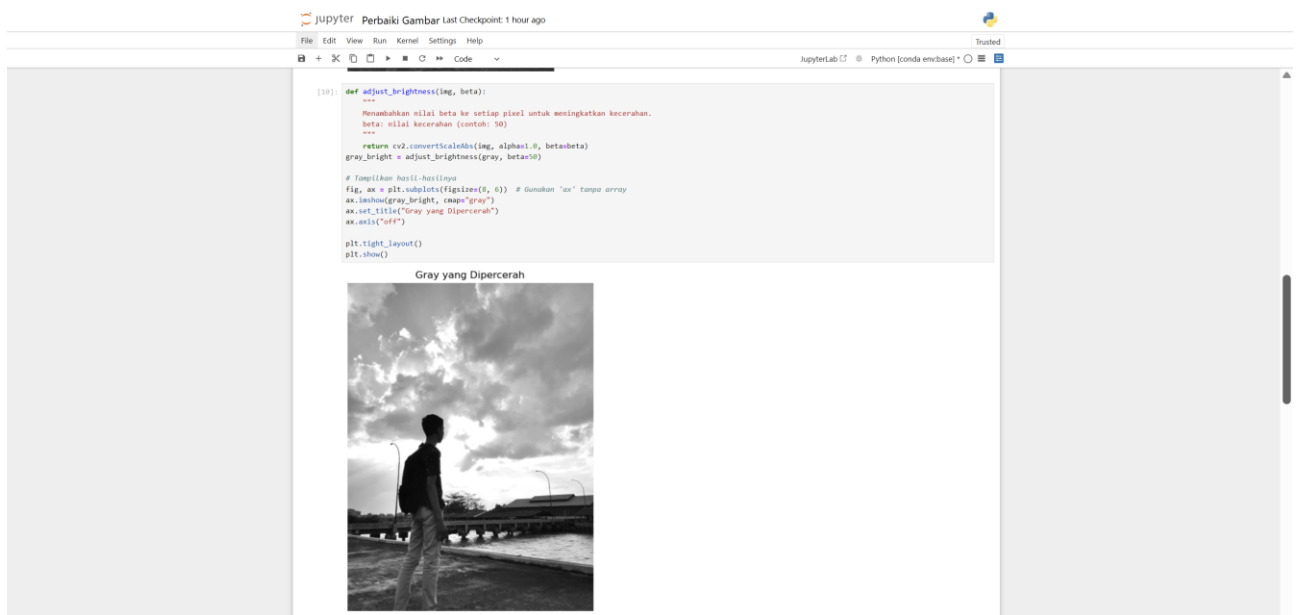


ini merupakan tahap awal sebelum dilakukan perbaikan atau pemrosesan lebih lanjut untuk memperjelas objek utama atau mengatasi efek backlight.

Penjelasan :

Gambar tersebut menunjukkan proses konversi gambar berwarna menjadi gambar grayscale (abu-abu) menggunakan OpenCV. Pada bagian kode, fungsi `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)` digunakan untuk mengubah citra dari format BGR (Blue-Green-Red) menjadi grayscale, yang hanya menyimpan intensitas cahaya tanpa informasi warna.

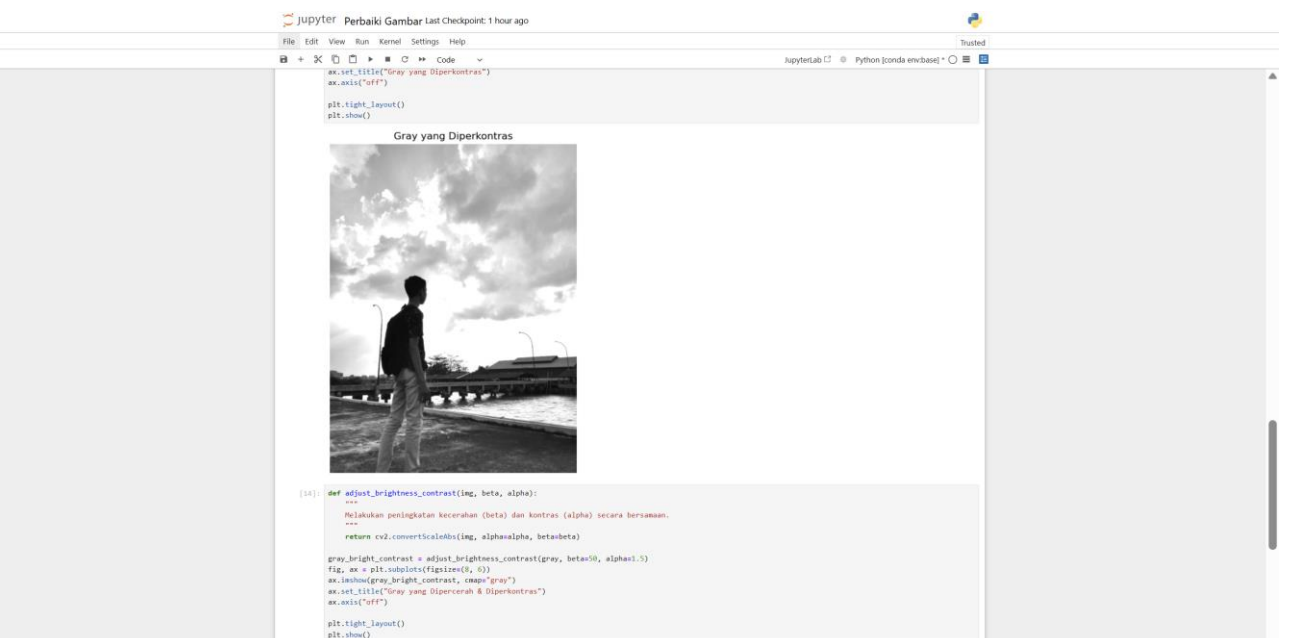
Hasil konversinya ditampilkan menggunakan matplotlib dengan judul “Gambar Gray”. Dalam gambar grayscale ini, objek manusia dan latar belakang seperti awan dan bangunan tetap terlihat,



tetapi tanpa warna. Konversi ini penting dalam pemrosesan citra karena banyak algoritma komputer vision bekerja lebih efisien dengan gambar hitam putih, terutama untuk deteksi tepi, segmentasi, dan pengenalan objek.

Penjelasan :

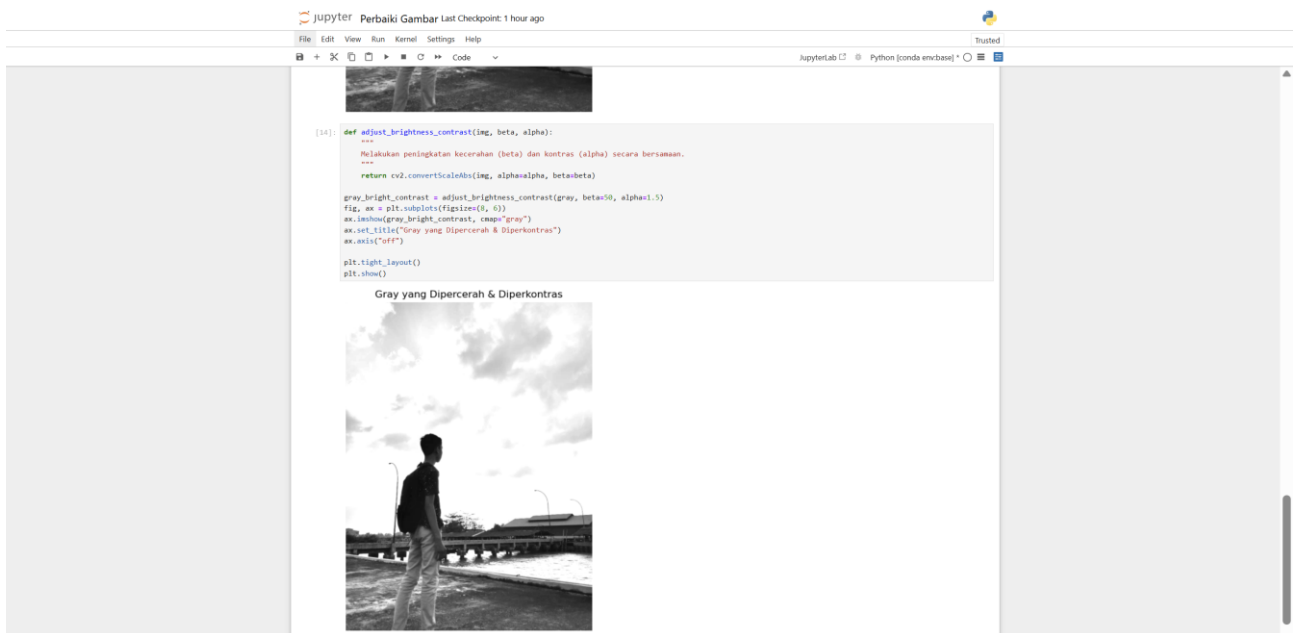
Gambar di atas menunjukkan proses peningkatan kecerahan dari gambar grayscale. Secara garis besar, gambar asli yang berwarna terlebih dahulu dikonversi ke format hitam-putih (grayscale), lalu dilakukan penyesuaian kecerahan menggunakan fungsi khusus. Fungsi ini menambahkan nilai



tertentu ke setiap piksel agar gambar terlihat lebih terang. Hasil akhirnya adalah gambar grayscale yang lebih cerah dan detailnya menjadi lebih terlihat, terutama di area gelap. Proses ini bermanfaat untuk meningkatkan kualitas visual sebelum dilakukan analisis gambar lebih lanjut.

Penjelasan :

Gambar di atas merupakan hasil dari pemrosesan citra menggunakan Python dalam lingkungan Jupyter Notebook. Secara garis besar, kode tersebut bertujuan untuk meningkatkan kualitas gambar grayscale dengan menyesuaikan kecerahan (brightness) dan kontras (contrast) secara



bersamaan. Fungsi `adjust_brightness_contrast` menggunakan fungsi `cv2.convertScaleAbs` dari OpenCV, di mana parameter `beta` digunakan untuk menambah kecerahan dan `alpha` untuk meningkatkan kontras. Hasil akhir ditampilkan dengan matplotlib, menunjukkan gambar yang telah diperbaiki pencahayaannya di bawah judul "Gray yang Dipercerah & Diperkontras".

Penjelasan :

Gambar dan kode pada bagian ini menunjukkan proses peningkatan kualitas citra grayscale melalui penyesuaian **kecerahan (brightness)** dan **kontras (contrast)** secara bersamaan. Fungsi `adjust_brightness_contrast` digunakan lagi, kali ini dengan parameter `beta=50` untuk menaikkan kecerahan dan `alpha=1.5` untuk meningkatkan kontras. Hasilnya adalah gambar yang terlihat lebih terang dan lebih kontras dibanding sebelumnya, membuat detail pada langit dan objek manusia lebih menonjol. Gambar ini ditampilkan dengan judul "Gray yang Dipercerah & Diperkontras".

## BAB IV PENUTUP

### 4.1. Kesimpulan

Secara keseluruhan, laporan ini membahas beberapa aspek penting dalam pengolahan citra digital menggunakan Python dan pustaka OpenCV, dengan fokus pada deteksi warna serta peningkatan kualitas gambar. tugas ini mencakup proses identifikasi warna primer—merah, hijau, dan biru—dalam citra RGB, yang berguna dalam banyak aplikasi seperti pelacakan objek, klasifikasi warna, dan pengenalan pola.

Selain itu, dilakukan analisis histogram warna yang memberikan gambaran distribusi intensitas piksel pada masing-masing kanal warna. Teknik ini penting untuk memahami pencahayaan dan kontras gambar, serta menjadi dasar dalam langkah-langkah pengolahan lanjutan.

Teknik thresholding digunakan untuk melakukan segmentasi objek berdasarkan intensitas piksel tertentu, sehingga objek-objek yang memiliki karakteristik warna atau kecerahan spesifik dapat dipisahkan dari latar belakang. Ini sangat bermanfaat dalam aplikasi seperti deteksi tepi, pengenalan objek, dan pemrosesan gambar medis.

Bagian penting lainnya adalah peningkatan kualitas citra yang mengalami pencahayaan backlight—yakni kondisi di mana objek tampak gelap karena cahaya datang dari belakang. Dengan menggunakan fungsi `cv2.convertScaleAbs`, gambar mengalami peningkatan pada nilai brightness dan contrast, sehingga detail objek menjadi lebih terlihat. Teknik ini menunjukkan bagaimana pemrosesan citra dapat memperbaiki hasil visual yang sebelumnya kurang optimal.

Secara keseluruhan, tugas ini mencerminkan pemahaman mendalam dan penerapan praktis dari teori-teori dasar pengolahan citra digital, terutama dalam hal peningkatan kualitas visual dan segmentasi berbasis warna. Pendekatan ini tidak hanya meningkatkan keterbacaan gambar tetapi juga mendukung analisis objek yang lebih akurat dalam berbagai aplikasi nyata.

### DAFTAR PUSTAKA

- Das, S., Gulati, T., & Mittal, V. (2015). Histogram Equalization Techniques for Contrast Enhancement: A Review. *International Journal of Computer Applications*, 114(10), 32–36. <https://doi.org/10.5120/20017-2027>
- Fakultas, S. R., Informasi, T., Islam, U., Muhammad, K., & Al Banjari, A. (2020). PENGOLAHAN CITRA DIGITAL DAN HISTOGRAM DENGAN PHYTON DAN TEXT EDITOR PHYCHARM. In *Technologia* (Vol. 11, Issue 3).
- Goenawan, A. D., Bakhara, M., Rachman, A., Pulungan, M. P., Komputer, I., & Esq, S. (2022). Identifikasi Warna Pada Objek Citra Digital Secara Real Time Menggunakan Pengolahan Model Warna HSV. In *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA) Jakarta-Indonesia* (Vol. 1).
- Guillen, G. (2019). *Digital Image Processing with Python and OpenCV* (pp. 97–140). [https://doi.org/10.1007/978-1-4842-5299-4\\_5](https://doi.org/10.1007/978-1-4842-5299-4_5)
- Marso, S., & El Merouani, M. (2020). Predicting financial distress using hybrid feedforward neural network with cuckoo search algorithm. *Procedia Computer Science*, 170, 1134–1140. <https://doi.org/10.1016/j.procs.2020.03.054>
- MDP STUDENT CONFERENCE (MSC) 2022*. (n.d.).
- Tirumani, V. H. L., Tenneti, M., Ch. Srikavya, K., & Kotamraju, S. K. (2021). Image resolution and contrast enhancement with optimal brightness compensation using wavelet transforms and particle swarm optimization. *IET Image Processing*, 15(12), 2833–2840. <https://doi.org/10.1049/ipr2.12268>
- Zahra Salsa Azizah, A., Nazar To Yalis, F., Shiva Narayana, I., Azelia Syalom, K., & Rosyani, P. (2024). Pengolahan Citra Digital Dengan Penerapan Teknik Ambang Batas: Studi Kasus Menggunakan Opencv. In *Jurnal Artificial Inteligent dan Sistem Penunjang Keputusan* (Vol. 1, Issue 4). <https://jurnalmahasiswa.com/index.php/aidanspk>