

UTS
PENGOLAHAN CITRA



INTELLIGENT COMPUTING

NAMA : Felix Ewaldo Panggabean.

NIM : 202331084.

KELAS : C.

DOSEN : Ir. Darma Rusjdi, M.Kom.

NO.PC : 11.

ASISTEN : 1. Abdur Rasyid Ridho.

2. Rizqy Amanda.

3. Kashrina Masyid Azka.

4. Izzat Islami Kagapi.

INSTITUT TEKNOLOGI PLN
TEKNIK INFORMATIKA
2024/2025

DAFTAR ISI

Contents

DAFTAR ISI.....	2
BAB I.....	4
PENDAHULUAN	4
Rumusan Masalah.....	4
1.1 Tujuan Masalah.....	4
1.2 Manfaat Masalah.....	6
BAB II.....	10
LANDASAN TEORI	10
2.1 PENGANTAR PENGOLAHAN CITRA DIGITAL.....	10
2.2 REPRESENTASI CITRA DIGITAL.....	10
2.3 OPERASI-OPERASI DASAR PADA PENGOLAHAN CITRA DIGITAL.....	10
2.4 KONSEP DASAR TEORI WARNA DAN ARAS KEABUAN.....	11
2.5 PENGOLAHAN CITRA BERBASIS PIKSEL.....	12
2.6 MORFOLOGI CITRA.....	12
2.7 KONSEP HISTOGRAM CITRA.....	12
2.8 PROSES MEMPERBAIKI KUALITAS CITRA.....	13
2.9 CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE).....	13
2.10 RUANG WARNA LAB UNTUK MANIPULASI LUMINANSI.....	14
2.11 NON-LOCAL MEANS DENOISING.....	14
2.12 GAMMA CORRECTION UNTUK PENYESUAIAN KECERAHAN.....	14
2.13 BILATERAL FILTER UNTUK PENGHALUSAN ADAPTIF.....	14
2.14 DETEksi WAJAH BERBASIS HAAR CASCADE CLASSIFIER.....	15
2.15 METRIK EVALUASI: PSNR DAN SSIM.....	15
BAB III.....	17
HASIL.....	17
3.1 DETEksi WARNA PADA CITRA.....	17
3.2 MEMPERBAIKI GAMBAR BACKLIGHT.....	26
3.3 MEMPERBAIKI GAMBAR BACKLIGHT.....	47
BAB IV.....	53
PENUTUP.....	53
4.2. KESIMPULAN.....	53

4.2. SARAN.....	53
DAFTAR PUSTAKA.....	55

BAB I

PENDAHULUAN

Rumusan Masalah.

Pengolahan citra digital merupakan salah satu cabang ilmu komputer yang berkembang pesat seiring dengan kemajuan teknologi perangkat keras dan perangkat lunak. Citra digital memiliki peran penting dalam berbagai bidang, mulai dari medis, penginderaan jauh, hingga industri hiburan dan keamanan. Dengan kemampuan untuk merepresentasikan dunia nyata dalam bentuk numerik, pengolahan citra digital memungkinkan analisis yang lebih mendalam dan manipulasi citra untuk berbagai tujuan. Seiring dengan itu, muncul berbagai tantangan dalam pengolahan citra yang melibatkan pemrosesan, manipulasi, dan analisis citra agar sesuai dengan kebutuhan aplikasi tertentu.

Beberapa masalah utama yang sering dihadapi dalam pengolahan citra digital antara lain adalah kualitas citra yang rendah akibat noise, distorsi, atau pencahayaan yang buruk, serta kesulitan dalam mengidentifikasi objek atau pola dalam citra yang kompleks. Selain itu, dalam pengolahan citra berwarna, pemahaman tentang teori warna dan aras keabuan juga sangat penting agar citra dapat diproses secara tepat sesuai dengan persepsi visual manusia. Tantangan lainnya melibatkan pengolahan citra berbasis piksel, yang menjadi dasar dari hampir semua teknik pengolahan citra, serta pengolahan citra morfologi untuk menganalisis bentuk dan struktur objek dalam citra.

Dalam hal ini, penting untuk memahami berbagai teknik dan algoritma yang digunakan dalam pengolahan citra digital, seperti histogram citra, pengolahan citra berbasis piksel, filter bilateral, serta deteksi wajah menggunakan algoritma Haar Cascade. Oleh karena itu, rumusan masalah dalam penelitian ini adalah :

1. Bagaimana teknik-teknik dasar pengolahan citra digital dapat diterapkan untuk meningkatkan kualitas citra, terutama dalam hal kontras, ketajaman, dan penghilangan noise ?.
2. Bagaimana konsep teori warna dan aras keabuan mempengaruhi pengolahan citra dalam aplikasi praktis ?.
3. Apa saja metode yang dapat digunakan untuk analisis bentuk dan struktur objek dalam citra menggunakan morfologi citra ?.
4. Bagaimana algoritma deteksi wajah berbasis Haar Cascade dapat diimplementasikan dalam pengolahan citra digital ?.

1.1 Tujuan Masalah.

Tujuan dari penelitian ini adalah untuk mengidentifikasi dan menganalisis berbagai metode dan teknik dalam pengolahan citra digital serta aplikasinya dalam meningkatkan kualitas citra dan ekstraksi informasi visual. Secara rinci, tujuan penelitian ini adalah :

1. Meningkatkan kualitas citra : Menganalisis berbagai teknik peningkatan kualitas citra, seperti penyesuaian kecerahan, kontras, dan pengurangan noise, untuk menghasilkan citra yang lebih jelas dan mudah diinterpretasikan.

2. Memahami konsep teori warna dan aras keabuan : Mempelajari cara-cara pengolahan citra yang melibatkan representasi warna dan grayscale, serta pengaruhnya terhadap hasil akhir citra yang diproses.
3. Mengaplikasikan morfologi citra : Menggunakan operasi morfologi untuk menganalisis bentuk dan struktur objek dalam citra, serta mendeteksi fitur-fitur penting dalam citra untuk aplikasi seperti segmentasi dan klasifikasi.
4. Implementasi deteksi wajah : Mengimplementasikan algoritma Haar Cascade untuk mendeteksi wajah dalam citra, serta memahami keunggulan dan kelemahan metode ini dalam berbagai kondisi pencitraan.

Dengan mencakup berbagai aspek pengolahan citra digital, penelitian ini bertujuan untuk memberikan solusi terhadap tantangan yang ada dalam pengolahan citra, serta memberikan pemahaman yang lebih dalam tentang cara-cara mengoptimalkan kualitas citra melalui penggunaan teknologi terbaru.

Adapun tujuan lagi yang saya temui yang berhubungan di dalam kehidupan sehari-hari seiring maraknya berkembangnya teknologi, seperti :

1. Meningkatkan kualitas citra digital.

Tujuan utama dari penelitian ini adalah untuk mengembangkan teknik yang dapat meningkatkan kualitas citra digital yang terpengaruh oleh noise, pencahayaan buruk, atau distorsi. Teknik seperti perbaikan kontras, ketajaman, serta penghilangan noise akan diterapkan untuk menghasilkan citra yang lebih bersih, jelas, dan mudah dianalisis.

2. Memahami teori warna dan aras keabuan.

Penelitian ini bertujuan untuk menggali lebih dalam tentang teori warna dan aras keabuan dalam citra digital. Pengertian ini penting untuk memastikan pengolahan citra sesuai dengan persepsi visual manusia, sehingga citra yang dihasilkan lebih realistik dan memadai untuk aplikasi tertentu seperti pemrosesan citra berwarna atau grayscale.

3. Menerapkan pengolahan citra morfologi.

Salah satu tujuan penting adalah menerapkan operasi morfologi dalam pengolahan citra. Teknik-teknik seperti dilasi, erosi, dan transformasi morfologi lainnya akan digunakan untuk menganalisis bentuk objek dalam citra. Hal ini bermanfaat untuk aplikasi seperti segmentasi citra dan analisis struktural objek.

4. Menerapkan algoritma Haar Cascade untuk deteksi wajah.

Penelitian ini bertujuan untuk mengimplementasikan algoritma Haar Cascade dalam mendeteksi wajah pada citra. Algoritma ini akan diuji untuk mengukur efektivitasnya dalam kondisi pencitraan yang berbeda, baik pada citra dengan kualitas tinggi maupun yang terpengaruh noise atau pencahayaan buruk.

5. Menganalisis peran histogram dalam pengolahan citra.

Salah satu tujuan penelitian ini adalah untuk menganalisis peran histogram citra dalam meningkatkan kontras dan distribusi warna. Teknik-teknik seperti equalization atau perbaikan histogram akan digunakan untuk mengoptimalkan visualisasi citra agar lebih mudah dianalisis.

6. Mengembangkan teknik segmentasi citra.

Segmentasi citra adalah langkah penting dalam analisis citra. Tujuan penelitian ini adalah untuk mengembangkan teknik segmentasi yang dapat memisahkan objek dengan latar

belakang secara lebih efektif. Ini sangat penting dalam aplikasi seperti deteksi objek atau analisis citra medis.

7. Menilai efektivitas teknik pengurangan noise.

Penelitian ini bertujuan untuk menilai berbagai teknik pengurangan noise, seperti filter median atau filter Gaussian. Teknik-teknik ini akan diuji pada citra yang terpengaruh oleh berbagai jenis noise, termasuk Gaussian, salt-and-pepper, dan speckle noise.

8. Meningkatkan akurasi deteksi fitur citra.

Salah satu tujuan penelitian ini adalah untuk meningkatkan akurasi dalam mendeteksi fitur penting dalam citra, seperti tepi, sudut, atau titik-titik yang signifikan. Teknik-teknik seperti deteksi tepi dengan Canny Edge Detection atau penggunaan algoritma SIFT akan diterapkan untuk memaksimalkan akurasi fitur tersebut.

9. Mengoptimalkan pemrosesan citra berwarna.

Tujuan lainnya adalah untuk mengoptimalkan pemrosesan citra berwarna, khususnya dalam bidang pengolahan citra medis atau pemrosesan citra satelit. Teknik seperti pengubahan ruang warna (misalnya RGB ke HSV) akan diterapkan untuk mendapatkan representasi citra yang lebih akurat dan berguna dalam aplikasi-aplikasi tersebut.

10. Mengembangkan sistem pengolahan citra otomatis.

Penelitian ini bertujuan untuk mengembangkan sistem otomatis yang dapat memproses citra dengan minimal intervensi manusia. Sistem ini akan menggunakan algoritma berbasis pembelajaran mesin untuk mendeteksi dan mengklasifikasikan objek dalam citra tanpa membutuhkan pengaturan manual yang rumit.

1.2 Manfaat Masalah.

Penelitian ini diharapkan dapat memberikan manfaat yang signifikan dalam pengembangan dan aplikasi pengolahan citra digital, terutama di berbagai bidang yang membutuhkan analisis citra yang akurat dan efisien. Beberapa manfaat yang dapat diperoleh antara lain :

1. Peningkatan kualitas citra : Dengan memanfaatkan berbagai teknik pengolahan citra digital yang ada, penelitian ini dapat menghasilkan citra yang lebih berkualitas, baik dari segi ketajaman, kontras, maupun penghilangan noise. Ini sangat berguna dalam aplikasi-aplikasi seperti pencitraan medis (MRI, CT scan, dan X-ray), penginderaan jauh, serta keamanan (misalnya dalam pengenalan wajah).
2. Aplikasi pada bidang medis : Dalam dunia medis, pengolahan citra digital dapat membantu dalam meningkatkan kualitas citra diagnostik, seperti MRI atau CT scan. Peningkatan kualitas citra ini dapat mempermudah diagnosis penyakit, mempercepat proses deteksi, dan mengurangi kemungkinan kesalahan dalam pengobatan.
3. Pengembangan sistem pengenalan pola : Penelitian ini juga memberikan kontribusi pada pengembangan sistem pengenalan pola, khususnya dalam deteksi objek dan segmentasi citra. Hal ini dapat dimanfaatkan dalam berbagai aplikasi, mulai dari pemantauan lingkungan, pemrosesan citra satelit, hingga aplikasi dalam bidang keamanan seperti pengenalan wajah dan plat nomor kendaraan.
4. Pemahaman teori warna dan aras keabuan : Dengan memahami konsep-konsep dasar dalam pengolahan citra berwarna dan grayscale, penelitian ini dapat memberikan kontribusi dalam meningkatkan kemampuan pemrosesan citra sesuai dengan persepsi manusia terhadap warna dan kecerahan. Ini dapat meningkatkan akurasi dalam aplikasi seperti pengolahan citra

berbasis persepsi visual (seperti dalam editing gambar, efek visual, dan pengolahan citra medis).

5. Inovasi dalam deteksi wajah : Implementasi algoritma Haar Cascade dalam deteksi wajah diharapkan dapat meningkatkan akurasi dan efisiensi dalam pengenalan wajah dalam citra, yang merupakan teknologi yang penting dalam bidang keamanan dan pengawasan. Penerapan metode ini juga dapat membuka peluang untuk pengembangan sistem yang lebih kompleks dan canggih, seperti pengenalan wajah real-time.
6. Pengembangan teknologi adaptif : Metode seperti CLAHE dan bilateral filter yang digunakan untuk meningkatkan kualitas citra dan menghilangkan noise memiliki potensi untuk diterapkan pada perangkat-perangkat yang memerlukan pengolahan citra dalam kondisi yang sangat variatif, seperti dalam pengolahan citra medis atau pencitraan dalam kondisi pencahayaan rendah.

Secara keseluruhan, penelitian ini berpotensi memberikan kontribusi yang besar dalam pengembangan teknologi pengolahan citra digital yang lebih efisien, akurat, dan dapat diterapkan dalam berbagai bidang, serta membuka peluang untuk pengembangan metode baru dalam analisis dan manipulasi citra digital.

Adapun manfaat lain dalam kehidupan sehari-hari seiring maraknya berkembangnya teknologi, seperti :

1. Peningkatan kualitas citra dalam aplikasi medis.
Salah satu manfaat utama dari penelitian ini adalah kemampuan untuk meningkatkan kualitas citra medis seperti CT scan, MRI, dan X-ray. Dengan meningkatkan kualitas citra medis, diagnosis penyakit menjadi lebih akurat dan cepat. Citra yang lebih jernih membantu dokter dalam mengidentifikasi masalah kesehatan dengan lebih tepat.
2. Mempermudah pengolahan citra dalam sistem keamanan.
Penelitian ini memiliki manfaat besar dalam pengembangan sistem keamanan yang menggunakan pengolahan citra, seperti pengawasan CCTV atau sistem pengenalan wajah. Pengolahan citra dapat membantu mendeteksi wajah atau objek dalam kondisi pencahayaan yang buruk atau citra yang kabur, sehingga meningkatkan efektivitas sistem pengawasan otomatis.
3. Penerapan teknologi untuk analisis citra satelit.
Manfaat lainnya adalah dalam pengolahan citra satelit untuk aplikasi seperti pemantauan lingkungan, pertanian, dan bencana alam. Dengan menggunakan teknik pemrosesan citra yang tepat, kualitas citra satelit dapat ditingkatkan, sehingga informasi yang diperoleh lebih akurat untuk analisis dan pengambilan keputusan.
4. Peningkatan performa sistem pengolahan citra berwarna.
Dalam aplikasi yang memerlukan citra berwarna yang akurat, seperti desain grafis atau pengeditan gambar, teknik yang dikembangkan dalam penelitian ini dapat menghasilkan citra dengan warna yang lebih tepat dan realistik, meningkatkan kualitas visual yang dihasilkan.
5. Pengolahan citra untuk keperluan industry.
Penelitian ini akan memberikan kontribusi pada pengolahan citra dalam industri, seperti inspeksi visual otomatis dalam produksi manufaktur atau deteksi cacat produk. Dengan meningkatkan kualitas citra, deteksi cacat pada produk menjadi lebih akurat, mengurangi risiko kesalahan produksi dan meningkatkan kualitas kontrol.

6. Peningkatan akurasi dalam sistem pengenalan objek.

Dalam aplikasi pengenalan objek, seperti pengenalan tulisan tangan atau kendaraan, penelitian ini dapat membantu meningkatkan akurasi deteksi objek meskipun citra yang digunakan terdistorsi atau terpengaruh noise. Ini akan bermanfaat dalam sistem pengolahan citra yang digunakan untuk analisis data visual dalam waktu nyata.

7. Inovasi dalam aplikasi pengolahan citra untuk hiburan.

Dalam industri hiburan, terutama dalam efek visual (VFX) dan pembuatan film, penelitian ini dapat digunakan untuk meningkatkan kualitas citra dalam pembuatan animasi atau efek visual yang realistik. Teknik pengolahan citra yang lebih canggih dapat menghasilkan gambar dengan kualitas tinggi yang sesuai dengan standar industri.

8. Penerapan dalam sistem pemandu kendaraan otomatis.

Manfaat lain adalah dalam penerapan pengolahan citra untuk sistem pemandu kendaraan otomatis (self-driving cars). Citra dari sensor seperti kamera dan LiDAR akan diproses untuk mengidentifikasi objek dan jalur jalan, meningkatkan keselamatan dan efisiensi kendaraan otonom.

9. Efisiensi dalam pemrosesan citra besar.

Penelitian ini juga memberikan manfaat bagi pengolahan citra dalam aplikasi yang melibatkan jumlah data besar, seperti pengolahan citra dalam penelitian ilmiah atau penyimpanan gambar digital. Teknik yang dikembangkan akan meningkatkan kecepatan dan efisiensi dalam memproses data citra berukuran besar.

10. Pengembangan teknologi pengolahan citra yang lebih adaptif.

Teknologi yang dikembangkan dalam penelitian ini akan membuka peluang untuk pengembangan teknologi pengolahan citra yang lebih adaptif, yang dapat menangani citra dengan kualitas rendah atau citra yang terpengaruh gangguan seperti noise, distorsi, atau kekaburan. Ini akan memperluas aplikasi pengolahan citra dalam berbagai sektor, termasuk penelitian ilmiah dan pengolahan citra industri.

11. Pemanfaatan dalam aplikasi pengolahan citra di bidang kejahatan digital.

Salah satu manfaat tambahan dari penelitian ini adalah untuk memperkenalkan teknik pengolahan citra dalam konteks forensik digital. Misalnya, dalam rekonstruksi citra yang rusak atau dalam analisis gambar yang digunakan dalam penyelidikan kejahatan dunia maya.

12. Inovasi dalam teknologi pemetaan digital.

Pengolahan citra ini juga dapat bermanfaat dalam pemetaan digital, seperti pembuatan peta 3D atau peta topografi dari citra satelit. Teknik pengolahan yang dihasilkan dapat digunakan untuk meningkatkan akurasi pemetaan dan analisis wilayah geografis secara lebih detail.

13. Dampak pada pengembangan teknologi augmented reality (AR).

Penelitian ini juga dapat memberikan manfaat pada pengembangan teknologi AR dengan cara memperbaiki kualitas citra yang digunakan dalam pengalaman AR. Hal ini akan meningkatkan pengalaman pengguna dengan gambar dan objek yang lebih tajam serta realistik.

14. Meningkatkan pemahaman tentang analisis citra dalam riset ilmiah.

Dalam dunia akademik dan riset ilmiah, penelitian ini bermanfaat dalam meningkatkan pemahaman tentang bagaimana pengolahan citra dapat digunakan untuk menganalisis data visual dalam eksperimen dan observasi ilmiah. Hal ini sangat penting dalam bidang seperti mikroskopi digital atau studi astronomi.

15. Memfasilitasi perkembangan aplikasi pengolahan citra berbasis cloud.

Dengan meningkatkan teknik pemrosesan citra, penelitian ini dapat memfasilitasi pengembangan aplikasi berbasis cloud yang memungkinkan pemrosesan citra secara efisien dan lebih hemat biaya. Aplikasi-aplikasi ini dapat digunakan dalam berbagai sektor, mulai dari e-commerce hingga analisis citra medis secara online.

BAB II

LANDASAN TEORI

2.1 PENGANTAR PENGOLAHAN CITRA DIGITAL.

Pengolahan citra digital adalah bidang yang terus berkembang dan memanfaatkan algoritma komputer untuk memanipulasi serta menganalisis citra digital demi berbagai tujuan, seperti peningkatan kualitas visual, ekstraksi informasi, dan pemahaman visual oleh manusia maupun mesin. Citra digital merupakan representasi visual dunia nyata dalam format numerik yang dapat diproses komputer, tersusun atas piksel-piksel yang merekam tingkat kecerahan atau warna pada posisi tertentu. Peralihan dari metode analog ke digital memungkinkan manipulasi dan analisis citra secara lebih luas dan kompleks. Teknologi ini kini digunakan di berbagai bidang, seperti kedokteran untuk analisis MRI, CT scan, dan X-ray dalam mendeteksi penyakit; penginderaan jauh untuk pemantauan lingkungan dan perencanaan wilayah; keamanan untuk pengenalan wajah dan plat nomor; industri hiburan untuk efek visual dan media sosial; serta sektor industri untuk kontrol kualitas dan pemantauan produksi. Kemampuannya dalam merepresentasikan data visual secara numerik menjadikan pengolahan citra digital sangat fleksibel dan aplikatif dalam berbagai domain.

2.2 REPRESENTASI CITRA DIGITAL.

Representasi citra digital adalah cara komputer menyimpan dan menginterpretasikan gambar melalui susunan piksel dalam bentuk matriks dua dimensi, di mana setiap piksel memiliki nilai numerik yang merepresentasikan intensitas cahaya atau warna. Dalam citra grayscale, tiap piksel diwakili oleh satu nilai antara 0 (hitam) hingga 255 (putih), yang menyederhanakan analisis visual dan mengurangi beban komputasi. Sebaliknya, citra berwarna menggunakan model RGB yang merepresentasikan tiap piksel dengan tiga komponen warna: merah, hijau, dan biru. Model warna lain seperti HSV, LAB, dan CMY digunakan dalam konteks yang lebih spesifik karena keunggulannya dalam merepresentasikan warna sesuai persepsi manusia atau kebutuhan algoritmik tertentu. Selain itu, citra biner hanya memiliki dua nilai (0 dan 1) dan sering digunakan dalam tugas-tugas segmentasi dan deteksi objek karena kesederhanaannya dalam merepresentasikan fitur secara eksplisit. Representasi citra sebagai matriks memungkinkan penerapan operasi matematis seperti konvolusi, transformasi, dan pemfilteran secara efisien. Seiring kemajuan teknologi, format baru seperti HEIF dikembangkan untuk kompresi lebih efisien tanpa mengorbankan kualitas, serta representasi berbasis vektor, multimodal, dan 3D (seperti point cloud dan volumetric video) diperkenalkan untuk mendukung aplikasi seperti grafis presisi tinggi dan video imersif. Inovasi ini bertujuan meningkatkan efisiensi penyimpanan, akurasi analisis, dan memperluas cakupan aplikasi dalam pengolahan citra digital.

2.3 OPERASI-OPERASI DASAR PADA PENGOLAHAN CITRA DIGITAL.

Operasi dasar dalam pengolahan citra digital merupakan serangkaian teknik yang diterapkan setelah citra direpresentasikan secara numerik, dengan tujuan memodifikasi atau mengekstrak informasi

dari gambar. Operasi ini dibedakan menjadi tiga kategori utama: operasi tingkat piksel, lokal, dan global. Operasi tingkat piksel bekerja langsung pada nilai piksel individual tanpa mempertimbangkan tetangganya, seperti penyesuaian kecerahan, kontras, pembuatan citra negatif, dan thresholding. Transformasi ini sederhana namun penting sebagai dasar untuk pengolahan yang lebih kompleks. Operasi tingkat lokal melibatkan hubungan antar piksel dalam wilayah tertentu, menggunakan kernel atau mask untuk melakukan penghalusan (seperti mean dan Gaussian filter) dan deteksi tepi melalui konvolusi, memungkinkan identifikasi fitur seperti batas objek. Operasi tingkat global mempertimbangkan seluruh citra, misalnya penyetaraan histogram yang mendistribusikan ulang intensitas piksel untuk meningkatkan kontras. Selain itu, operasi aritmetika memungkinkan manipulasi antar citra seperti penjumlahan dan pengurangan untuk reduksi noise atau deteksi perubahan, sementara operasi logika (AND, OR, NOT) berguna dalam citra biner untuk masking dan segmentasi. Operasi geometri, seperti translasi, rotasi, penskalaan, dan pencerminan, memodifikasi susunan spasial piksel untuk penyelarasan dan transformasi bentuk citra. Perkembangan terbaru dalam pengolahan citra mengintegrasikan kecerdasan buatan, khususnya deep learning, untuk tugas-tugas lanjutan seperti restorasi citra, peningkatan resolusi, penghilangan artefak, dan fusi citra dari berbagai sumber, menghasilkan analisis visual yang lebih akurat, efisien, dan adaptif terhadap kompleksitas data modern.

2.4 KONSEP DASAR TEORI WARNA DAN ARAS KEABUAN.

Pemahaman teori warna dan aras keabuan memainkan peran sentral dalam pengolahan citra digital, khususnya dalam menangani citra berwarna dan aplikasi yang memerlukan pemrosesan perceptual. Warna, sebagai respons fisiologis terhadap panjang gelombang cahaya, direpresentasikan dalam pengolahan digital melalui berbagai model warna seperti RGB, CMYK, HSV, dan LAB, masing-masing dengan aplikasi spesifik. RGB dominan di tampilan digital, CMYK di pencetakan, sedangkan HSV dan LAB berguna untuk manipulasi berbasis persepsi karena memisahkan elemen warna dari kecerahan. Teori warna juga mencakup pemahaman tentang pencampuran warna (aditif dan subtraktif), harmoni warna, dan pengaruh warna terhadap persepsi visual serta keterlibatan pengguna. Di sisi lain, aras keabuan merepresentasikan intensitas cahaya tanpa informasi warna, biasanya dalam skala 0–255. Citra grayscale penting karena menyederhanakan pemrosesan data, mengurangi kebutuhan komputasi, dan tetap mempertahankan informasi esensial—seperti dalam deteksi tepi, analisis tekstur, atau pencitraan medis. Konversi warna ke grayscale sering dilakukan dengan metode luminositas yang mempertimbangkan sensitivitas mata terhadap warna ($Gray = 0.299R + 0.587G + 0.114B$). Perkembangan terbaru memanfaatkan AI dan deep learning untuk tugas-tugas lanjutan seperti colorization citra grayscale, koreksi warna otomatis (color constancy), serta integrasi data warna dan keabuan untuk segmentasi dan klasifikasi. Model seperti GAN mampu menambahkan warna secara realistik ke citra hitam-putih, sementara penelitian di bidang color constancy berupaya menghadirkan citra digital yang tetap konsisten secara warna dalam berbagai kondisi pencahayaan. Dengan demikian, pemahaman teoritis dan praktis tentang warna dan keabuan sangat penting dalam menciptakan dan memproses citra digital secara efektif dan adaptif.

2.5 PENGOLAHAN CITRA BERBASIS PIKSEL.

Pengolahan citra berbasis piksel merupakan pendekatan fundamental dalam pengolahan citra digital yang bekerja langsung pada data diskrit berupa piksel, baik secara individual maupun dalam kelompok kecil. Operasi ini mencakup teknik dasar seperti peningkatan kecerahan, penyesuaian kontras, dan thresholding, serta konvolusi yang melibatkan nilai piksel tetangga namun tetap beroperasi pada tingkat piksel. Pendekatan ini memberikan kendali presisi tinggi dalam manipulasi citra dan memungkinkan penerapan algoritma dari modifikasi sederhana hingga ekstraksi fitur kompleks. Perkembangan terbaru didorong oleh kemajuan deep learning dan teknologi pencitraan inovatif, seperti super-resolution yang memprediksi nilai piksel hilang, single-pixel imaging (SPI) yang efektif dalam kondisi pencahayaan rendah, serta watermarking berbasis piksel menggunakan teknik Least Significant Bit (LSB) untuk keamanan data. Selain itu, pixel grouping juga dimanfaatkan dalam rendering citra 3D. Keseluruhan kemajuan ini menunjukkan bahwa pengolahan berbasis piksel tetap relevan dan berkembang dalam meningkatkan resolusi, keamanan, dan efisiensi pencitraan.

2.6 MORFOLOGI CITRA.

Morfologi citra adalah cabang pengolahan citra digital yang berfokus pada analisis dan manipulasi bentuk serta struktur objek dalam citra, umumnya diterapkan pada citra biner namun juga dapat diperluas ke citra grayscale dan berwarna. Operasi dasar seperti dilasi dan erosi berfungsi memperbesar atau mengecilkan objek, dengan dilasi memperluas batas objek untuk mengisi celah dan menghubungkan bagian terpisah, sedangkan erosi mengurangi ukuran objek untuk menghilangkan noise dan memisahkan objek yang berdekatan. Kombinasi kedua operasi ini membentuk operasi komposit seperti opening, yang menghilangkan detail kecil sambil mempertahankan bentuk objek utama, serta closing, yang menutup celah kecil dan memperhalus kontur. Selain itu, operasi lanjutan seperti Hit-or-Miss, Top-Hat, Gradient, Skeletonization, dan Region Filling menyediakan alat penting untuk deteksi pola, ekstraksi fitur, dan segmentasi objek secara lebih kompleks. Perkembangan terkini dalam morfologi citra mencakup penerapannya pada citra non-binari, integrasi dengan deep learning melalui pengembangan deep morphological neural networks (DMNN), serta pemanfaatannya dalam bidang bioinformatika dan penginderaan jauh. Morfologi grayscale digunakan untuk pemrosesan citra dan video, sementara attribute profiles memungkinkan ekstraksi fitur morfologis dalam analisis citra satelit. Dalam konteks medis, operasi morfologi berperan dalam prapemrosesan citra untuk meningkatkan kualitas visual dan mendukung analisis lanjutan. Integrasi dengan AI membuka peluang baru untuk pengembangan model yang lebih efisien dan adaptif dalam memahami struktur visual kompleks.

2.7 KONSEP HISTOGRAM CITRA.

Histogram citra merupakan representasi grafis dari distribusi intensitas piksel dalam citra digital yang memberikan informasi penting tentang karakteristik tonal, seperti kecerahan, kontras, dan sebaran nilai intensitas. Untuk citra grayscale, histogram memetakan frekuensi setiap tingkat keabuan dari hitam (0) hingga putih (255), sementara untuk citra berwarna RGB, histogram dibuat secara terpisah untuk kanal merah, hijau, dan biru guna menganalisis distribusi warna masing-masing. Histogram yang dinormalisasi menyajikan distribusi relatif intensitas piksel,

memungkinkan evaluasi cepat terhadap kualitas visual citra. Teknik ini digunakan secara luas dalam peningkatan citra, seperti dalam pelebaran dan ekualisasi histogram untuk memperbaiki kontras dan kecerahan. Ekualisasi histogram menyebarluaskan ulang intensitas agar lebih merata menggunakan fungsi distribusi kumulatif (CDF), sedangkan metode adaptif seperti AHE dan CLAHE digunakan untuk peningkatan kontras lokal, mengatasi keterbatasan ekualisasi global. Selain peningkatan kualitas visual, histogram juga digunakan dalam segmentasi citra, pencocokan histogram, dan analisis morfologi seperti pengukuran ukuran butiran dalam gambar teknik dan material. Pengembangan terbaru mencakup histogram orde kedua untuk menangani citra dengan noise tinggi, serta integrasi histogram dalam sistem pencarian citra berbasis konten sebagai fitur global untuk mengukur kesamaan antar citra. Secara keseluruhan, histogram citra adalah alat analisis yang fundamental dan serbaguna, dengan aplikasi luas dari peningkatan kualitas visual hingga pengenalan pola dan pengambilan informasi citra berbasis kecerdasan buatan.

2.8 PROSES MEMPERBAIKI KUALITAS CITRA.

Memperbaiki kualitas citra merupakan proses penting dalam pengolahan citra digital yang bertujuan untuk meningkatkan aspek visual gambar agar lebih mudah diinterpretasikan oleh manusia atau dianalisis secara otomatis. Proses ini mencakup dua pendekatan utama, yaitu peningkatan (enhancement) dan pemulihan (restoration). Peningkatan kualitas citra berfokus pada manipulasi atribut visual seperti kecerahan, kontras, ketajaman, dan warna dengan teknik seperti contrast stretching, histogram equalization, image sharpening, dan noise reduction. Tujuannya adalah untuk membuat citra lebih menarik secara visual dengan memperjelas detail dan mengurangi gangguan seperti noise atau blur. Di sisi lain, pemulihan citra bertujuan mengembalikan citra yang telah mengalami degradasi akibat faktor eksternal seperti blur atau noise, dengan menggunakan model matematis dari proses degradasi tersebut. Teknik seperti inverse filtering, Wiener filtering, dan deconvolution digunakan untuk memperkirakan kembali citra asli. Perkembangan terbaru dalam peningkatan dan pemulihan citra sangat dipengaruhi oleh kemajuan deep learning dan kecerdasan buatan, dengan arsitektur seperti CNN dan GAN yang mampu mempelajari hubungan kompleks antara citra rusak dan citra berkualitas tinggi. Model seperti Zero-DCE dan MIRNet telah dikembangkan untuk tugas-tugas khusus seperti peningkatan citra dalam kondisi cahaya rendah dan restorasi gambar nyata. Penerapan AI juga berkembang pesat dalam pencitraan medis, membantu peningkatan kualitas gambar diagnostik dan memungkinkan prosedur dengan dosis radiasi yang lebih rendah. Secara keseluruhan, integrasi metode konvensional dengan teknologi AI telah merevolusi proses perbaikan kualitas citra, menghasilkan performa yang lebih unggul dalam berbagai konteks aplikasi.

2.9 CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE).

CLAHE adalah varian dari Adaptive Histogram Equalization (AHE) yang dirancang untuk mengatasi masalah amplifikasi noise di area homogen. AHE melakukan ekualisasi histogram secara lokal pada jendela kecil yang bergerak di seluruh citra, yang dapat memperkuat noise di wilayah dengan sedikit variasi intensitas. CLAHE membatasi (mengklip) histogram di setiap jendela sebelum menghitung fungsi distribusi kumulatif (CDF). Klipping ini mencegah peningkatan kontras yang berlebihan di area noise. Setelah klipping, bagian yang terpotong didistribusikan kembali secara

UTS.

merata ke seluruh bin histogram. Transformasi pemetaan kemudian diterapkan pada piksel di setiap jendela. Untuk menghindari artefak batas antar jendela, interpolasi bilinear biasanya digunakan untuk menghaluskan transisi [67, 68]. CLAHE sangat efektif dalam meningkatkan kontras lokal tanpa memperkenalkan noise yang signifikan, menjadikannya populer dalam pencitraan medis dan aplikasi lain di mana detail lokal penting [67].

2.10 RUANG WARNA LAB UNTUK MANIPULASI LUMINANSI.

Ruang warna LAB (juga dikenal sebagai CIELAB) dirancang agar seragam secara perceptual, artinya perbedaan numerik yang sama dalam nilai LAB sesuai dengan perbedaan visual yang sama [16]. Ruang warna ini memisahkan komponen luminansi (L) dari komponen kromatik (A dan B). Kanal L merepresentasikan kecerahan, sedangkan kanal A dan B merepresentasikan warna (dari hijau ke merah dan dari biru ke kuning) [16]. Keunggulan utama ruang warna LAB untuk pengolahan citra adalah kemampuannya untuk memanipulasi kecerahan citra secara independen dari warnanya dengan hanya memproses kanal L. Ini memungkinkan penyesuaian kontras atau peningkatan kecerahan tanpa mengubah rona atau saturasi citra secara signifikan [16].

2.11 NON-LOCAL MEANS DENOISING.

Non-Local Means (NLM) adalah algoritma denoising yang efektif yang memanfaatkan redundansi dalam citra alami. Berbeda dengan filter lokal yang hanya mempertimbangkan piksel tetangga terdekat, NLM mengambil rata-rata nilai piksel di seluruh citra berdasarkan kesamaan patch di sekitar piksel [74]. Untuk menghitung nilai piksel yang di-denoise pada lokasi tertentu, algoritma mencari patch yang serupa di seluruh citra. Piksel dari patch yang lebih mirip akan diberi bobot yang lebih tinggi dalam perhitungan rata-rata. Pendekatan non-lokal ini memungkinkan NLM untuk mempertahankan detail dan tekstur halus yang mungkin hilang oleh filter lokal [74].

2.12 GAMMA CORRECTION UNTUK PENYESUAIAN KECERAHAN.

Gamma correction adalah operasi non-linear yang digunakan untuk menyesuaikan kecerahan keseluruhan citra dan meningkatkan detail di area gelap atau terang. Hubungan antara nilai piksel input $f(x,y)$ dan nilai piksel output $g(x,y)$ didefinisikan oleh fungsi pangkat :

$$g(x,y) = c \cdot f(x,y)^\gamma$$

di mana c adalah konstanta (seringkali 1) dan γ (gamma) adalah eksponen [79]. Jika $\gamma < 1$, transformasi akan mencerahkan citra, meningkatkan detail di area gelap. Jika $\gamma > 1$, transformasi akan menggelapkan citra, meningkatkan detail di area terang [79]. Gamma correction penting untuk memastikan bahwa citra ditampilkan dengan benar pada berbagai perangkat dengan karakteristik tampilan yang berbeda.

2.13 BILATERAL FILTER UNTUK PENGHALUSAN ADAPTIF.

Bilateral filter adalah filter non-linear yang efektif untuk menghaluskan citra sambil mempertahankan tepi. Filter ini mencapai hal ini dengan mempertimbangkan dua faktor saat

menghitung rata-rata tertimbang untuk setiap piksel: kedekatan spasial dan kesamaan intensitas [72]. Bobot spasial menurun seiring dengan jarak dari piksel pusat, mirip dengan filter Gaussian. Bobot intensitas menurun seiring dengan perbedaan intensitas antara piksel pusat dan piksel tetangga. Dengan menggabungkan kedua bobot ini, bilateral filter menghaluskan area homogen (di mana perbedaan intensitas kecil) tetapi tidak menghaluskan di sepanjang tepi (di mana perbedaan intensitas besar), sehingga mempertahankan tepi [72].

2.14 DETEKSI WAJAH BERBASIS HAAR CASCADE CLASSIFIER.

Deteksi wajah adalah aplikasi penting dari pengolahan citra digital. Salah satu metode klasik dan masih relevan adalah menggunakan Haar Cascade Classifier [85]. Algoritma ini menggunakan fitur Haar-like, yang merupakan pola persegi panjang yang membandingkan jumlah intensitas piksel di wilayah yang berdekatan. Fitur-fitur ini dilatih menggunakan algoritma machine learning (biasanya Adaboost) untuk mengidentifikasi pola yang khas dari wajah (seperti area mata yang lebih gelap dibandingkan dengan dahi). Classifier diatur dalam bentuk "cascade" (tingkatan), di mana setiap tingkatan terdiri dari serangkaian fitur. Jendela pencarian digeser ke seluruh citra, dan pada setiap posisi dan skala, fitur-fitur dievaluasi secara berurutan melalui tingkatan cascade. Jika suatu wilayah gagal pada tingkatan awal, ia segera ditolak sebagai non-wajah, yang secara signifikan meningkatkan efisiensi. Hanya wilayah yang melewati semua tingkatan yang diklasifikasikan sebagai wajah [85].

2.15 METRIK EVALUASI: PSNR DAN SSIM.

Untuk mengevaluasi kinerja algoritma pengolahan citra secara kuantitatif, terutama dalam tugas-tugas seperti denoising atau kompresi, metrik objektif digunakan. Dua metrik yang umum adalah Peak Signal-to-Noise Ratio (PSNR) dan Structural Similarity Index Measure (SSIM).

- Peak Signal-to-Noise Ratio (PSNR) : PSNR mengukur rasio antara daya maksimum sinyal (citra asli) dan daya noise yang mengganggu (perbedaan antara citra asli dan citra yang diproses). Nilai PSNR yang lebih tinggi menunjukkan kualitas citra yang lebih baik (noise lebih rendah). PSNR dihitung menggunakan Mean Squared Error (MSE) antara citra asli dan citra yang diproses.

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - K(i,j)]^2$$

di mana I adalah citra asli, K adalah citra yang diproses, M dan N adalah dimensi citra, dan (i,j) adalah koordinat piksel.

$$PSNR = 10 \log_{10} (MSE / MAX^2)$$

di mana MAX adalah nilai piksel maksimum yang mungkin (misalnya, 255 untuk citra 8-bit) [86].

- Structural Similarity Index Measure (SSIM) : SSIM adalah metrik yang dirancang untuk mengukur kemiripan antara dua citra dengan mempertimbangkan tiga faktor: luminansi, kontras, dan struktur. Berbeda dengan PSNR yang hanya mengukur perbedaan piksel, SSIM mencoba untuk menilai kualitas citra dengan cara yang lebih sesuai dengan persepsi

UTS.

manusia. Nilai SSIM berkisar antara -1 dan 1, di mana 1 menunjukkan kesamaan sempurna antara kedua citra [86].

UTS.

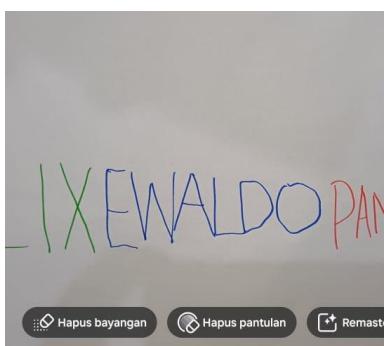
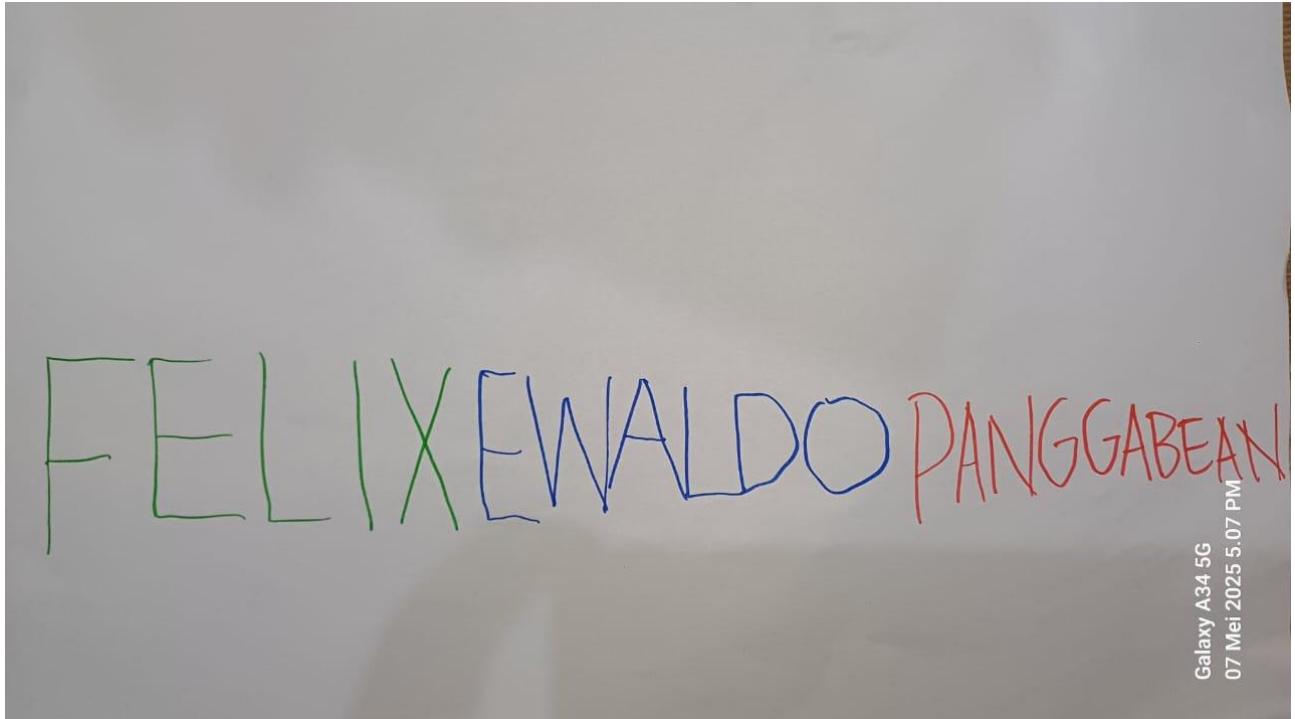
BAB III

HASIL

3.1 DETEKSI WARNA PADA CITRA.

Ketentuan :

- o Deteksi warna biru, merah, dan hijau pada gambar lalu tampilkan semua :



Rabu, 07 Mei 2025 · 17.07 Ubah

20250507_170729.jpg
/Penyimpanan internal/DCIM/Camera

Galaxy A34 5G

1,66 MB | 4000x2250 | 9MP
ISO 320 | 25mm | 0,0ev | F1,8 | 1/33 s



Jl Jl. Komp. Interkota Indah No.16, RT.12/RW.7,
Duri Kosambi, Kecamatan Cengkareng, Kota
Jakarta Barat, Daerah Khusus Ibukota Jakarta
11750, Indonesia

UTS.

program :

```
import cv2  
  
import numpy as np  
  
import matplotlib.pyplot as plt
```

Penjelasan :

cv2 : dari OpenCV, digunakan untuk membaca, memproses, dan menyimpan gambar atau video.

Numpy : digunakan untuk manipulasi array/matriks, sering digunakan untuk representasi gambar sebagai data numerik.

matplotlib.pyplot : digunakan untuk menampilkan gambar atau grafik secara visual di jendela plot.

```
img = cv2.imread('NAMA.jpg')
```

Penjelasan :

- Berarti membaca file gambar bernama 'NAMA.jpg' menggunakan OpenCV, lalu menyimpannya dalam variabel img sebagai array NumPy. Jika file tidak ditemukan, img akan bernilai None. Gambar dibaca dalam format BGR (Blue, Green, Red) secara default, bukan RGB.

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Penjelasan :

- `img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`. Mengubah gambar dari format BGR (default OpenCV) ke RGB, agar warnanya tampil benar saat ditampilkan dengan matplotlib.
- `hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)`. Mengubah gambar dari BGR ke HSV (Hue, Saturation, Value), yang sering dipakai untuk deteksi warna atau segmentasi karena lebih mudah memisahkan warna dari intensitas cahaya.

```
(baris, kolom) = img_rgb.shape[:2]
```

UTS.

Penjelasan :

berarti mengambil tinggi dan lebar gambar img_rgb :

- img_rgb.shape memberikan dimensi array gambar: (tinggi, lebar, saluran_warna).
- img_rgb.shape[:2] mengambil dua nilai pertama: tinggi (baris) dan lebar (kolom).

Jadi, variabel baris menyimpan jumlah piksel vertikal, dan kolom menyimpan jumlah piksel horizontal dari gambar tersebut.

```
# Memisahkan channel BGR
```

```
B, G, R = cv2.split(image)
```

```
# Konversi BGR ke RGB
```

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
faktor = 0.1 # Semakin kecil nilainya, semakin pudar
```

```
G_pudar = (G * faktor).astype(np.uint8)
```

```
# Membuat subplot untuk menampilkan gambar
```

```
fig, axs = plt.subplots(2, 2, figsize=(15, 8))
```

```
# Citra RGB
```

```
axs[0, 0].imshow(image_rgb)
```

```
axs[0, 0].set_title("CITRA KONTRAS")
```

```
axs[0, 0].axis('on')      # Menampilkan axis ticks
```

```
axs[0, 0].grid(False)    # Nonaktifkan grid
```

```
# Channel Biru
```

```
axs[0, 1].imshow(B, cmap='gray')
```

```
axs[0, 1].set_title("BIRU")
```

UTS.

```
axs[0, 1].axis('on')
axs[0, 1].grid(False)

# Channel Merah
axs[1, 0].imshow(R, cmap='gray')
axs[1, 0].set_title("MERAH")
axs[1, 0].axis('on')
axs[1, 0].grid(False)

# Channel Hijau
axs[1, 1].imshow(G, cmap='gray')
axs[1, 1].set_title("HIJAU")
axs[1, 1].axis('on')
axs[1, 1].grid(False)

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

Penjelasan :

- Memisahkan channel B, G, dan R.

B, G, R = cv2.split(image)

= Memisahkan gambar image (dalam format BGR) menjadi tiga channel: Biru, Hijau, dan Merah.

- Konversi ke RGB untuk ditampilkan.

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

= Agar warna gambar tampil benar di matplotlib.

- Membuat channel hijau lebih pudar.

G_pudar = (G * faktor).astype(np.uint8)

= Mengurangi intensitas hijau (meskipun hasil G_pudar tidak ditampilkan dalam plot ini).

UTS.

- Menampilkan gambar dalam 2x2 grid subplot.

```
fig, axs = plt.subplots(2, 2, figsize=(15, 8))
```

= Membuat area plot dengan 2 baris \times 2 kolom.

- Menampilkan :

Gambar asli (RGB).

Channel Biru (grayscale).

Channel Merah (grayscale).

Channel Hijau (grayscale).

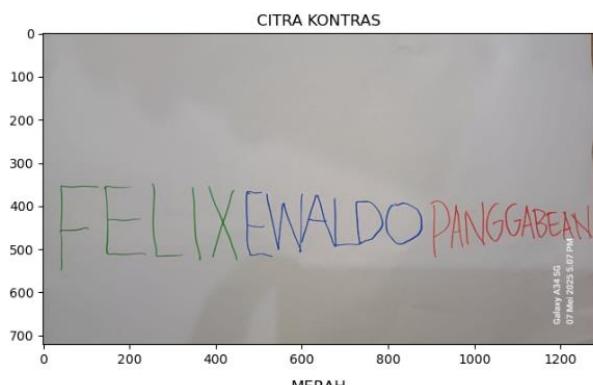
- Aksesori plot.

axis('on'): Menampilkan angka pada sumbu.

grid(False): Mematikan grid.

- G_pudar sudah dihitung tapi tidak digunakan di visualisasi.

- Hasil :



- Penjelasan :

1. CITRA KONTRAS :

Gambar output yang ditampilkan merupakan hasil dari proses pemrosesan citra digital yang bertujuan untuk memvisualisasikan komponen warna dalam gambar dengan memisahkan

channel BGR (Biru, Hijau, Merah). Citra asli terlebih dahulu dikonversi dari format BGR ke RGB menggunakan OpenCV agar warna yang ditampilkan sesuai dengan persepsi manusia saat divisualisasikan menggunakan Matplotlib. Hasil konversi ini ditampilkan pada subplot pertama (pojok kiri atas) dengan judul “CITRA KONTRAS”, di mana terlihat tulisan berwarna “FELIX” (hijau), “EWALDO” (biru), dan “PANGGABEAN” (merah) yang jelas menunjukkan kombinasi warna asli gambar.

2. BIRU :

Subplot kedua (pojok kanan atas) menampilkan channel biru (B) dalam skala abu-abu. Di sini, bagian tulisan “EWALDO” tampak paling terang karena mengandung komponen biru yang kuat. Warna putih atau terang pada gambar grayscale menunjukkan intensitas tinggi pada channel tersebut, sedangkan area yang lebih gelap menunjukkan sedikit atau tidak adanya kandungan warna biru.

3. MERAH :

Hal serupa berlaku pada subplot ketiga (bawah kiri), yang menampilkan channel merah (R). Tulisan “PANGGABEAN” tampak terang karena dominan merah, sementara tulisan lain seperti “FELIX” yang berwarna hijau tampak sangat gelap, menunjukkan tidak adanya komponen merah yang signifikan di bagian tersebut.

4. HIJAU :

Subplot keempat (bawah kanan) memperlihatkan channel hijau (G). Dalam visualisasi ini, tulisan “FELIX” tampak terang karena memiliki kandungan warna hijau yang dominan, sedangkan bagian tulisan lain yang mengandung sedikit hijau akan tampak gelap. Visualisasi channel-channel ini membantu kita memahami bagaimana masing-masing warna membentuk keseluruhan citra. Dengan memisahkan channel B, G, dan R, kita bisa mengamati seberapa besar kontribusi masing-masing warna terhadap tampilan akhir gambar.

- Proses ini sangat penting dalam bidang pengolahan citra digital, karena memungkinkan kita untuk menganalisis citra berdasarkan karakteristik warna, melakukan segmentasi objek berdasarkan warna dominan, atau melakukan manipulasi warna secara selektif. Meskipun dalam kode terdapat variabel G_pudar (hasil pemudaran channel hijau), nilai tersebut tidak digunakan dalam visualisasi akhir. Jika diinginkan, G_pudar bisa ditambahkan untuk menunjukkan bagaimana hasil manipulasi warna dapat mengubah persepsi visual citra. Keseluruhan proses ini mencerminkan pendekatan sistematis dalam eksplorasi dan analisis komponen warna dari suatu gambar digital.

```
# Ambil channel warna dari gambar RGB asli
```

```
r = img_rgb[:, :, 0]
```

```
g = img_rgb[:, :, 1]
```

```
b = img_rgb[:, :, 2]
```

```
# Tampilkan histogram
```

```
plt.figure(figsize=(20, 5))
```

UTS.

```
plt.subplot(1, 4, 1)
plt.title('Histogram Gabungan RGB')
plt.hist(img_rgb.flatten(), bins=256, range=[0,256], color='gray')

plt.subplot(1, 4, 2)
plt.title('Histogram Merah')
plt.hist(r.flatten(), bins=256, range=[0,256], color='r')

plt.subplot(1, 4, 3)
plt.title('Histogram Hijau')
plt.hist(g.flatten(), bins=256, range=[0,256], color='g')

plt.subplot(1, 4, 4)
plt.title('Histogram Biru')
plt.hist(b.flatten(), bins=256, range=[0,256], color='b')

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

Penjelasan : Kode yang diberikan bertujuan untuk menampilkan histogram warna dari sebuah citra digital dalam format RGB, yaitu dengan memvisualisasikan sebaran intensitas piksel untuk masing-masing channel warna (merah, hijau, biru) serta gabungannya.

Penjelasan Baris Per Baris :

- Ekstraksi channel warna :

```
r = img_rgb[:, :, 0]
```

```
g = img_rgb[:, :, 1]
```

```
b = img_rgb[:, :, 2]
```

Penjelasan :

UTS.

- Gambar RGB adalah array 3 dimensi (baris, kolom, channel).
- Baris ini mengekstrak masing-masing channel :
 - R : channel merah (index 0).
 - G : channel hijau (index 1).
 - B : channel biru (index 2).
- Inisialisasi area plotting.

```
plt.figure(figsize=(20, 5))
```

= Membuat kanvas plot berukuran besar agar semua histogram bisa ditampilkan berjajar dengan jelas.

- Histogram Gabungan RGB.

```
plt.subplot(1, 4, 1)
```

```
plt.title('Histogram Gabungan RGB')
```

```
plt.hist(img_rgb.flatten(), bins=256, range=[0,256], color='gray')
```

= Menggabungkan semua nilai piksel (dari ketiga channel) jadi satu array 1D menggunakan flatten(). Histogram ini menampilkan distribusi keseluruhan intensitas warna dari semua channel, jadi memberi gambaran umum tingkat kecerahan seluruh citra.

- Histogram Merah, Hijau, dan Biru.

```
plt.subplot(1, 4, 2) # Merah
```

```
plt.subplot(1, 4, 3) # Hijau
```

```
plt.subplot(1, 4, 4) # Biru
```

= Setiap subplot menunjukkan histogram untuk masing-masing channel secara terpisah :

1. plt.hist(r.flatten(), ...) menampilkan sebaran intensitas merah.
 2. plt.hist(g.flatten(), ...) untuk hijau.
 3. plt.hist(b.flatten(), ...) untuk biru.
 4. Warna histogram disesuaikan agar mudah dibedakan ('r', 'g', 'b').
- Tata letak.

```
plt.tight_layout()
```

= Mengatur tata letak otomatis agar semua subplot rapi dan tidak saling menimpa.

- Menampilkan plot.

```
plt.show()
```

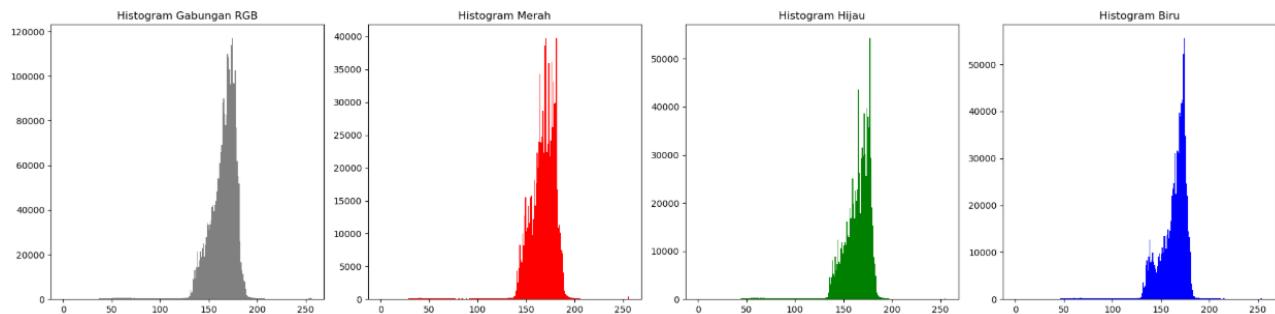
= Memunculkan semua histogram ke layar.

UTS.

Fungsi Histogram Citra.

Histogram warna sangat berguna dalam :

- Analisis kontras dan pencahayaan (apakah gambar terang, gelap, atau kontras rendah).
- Mengetahui distribusi warna dominan dalam gambar.
- Persiapan untuk pemrosesan lanjutan seperti equalization, thresholding, atau deteksi objek berbasis warna.



Penjelasan Histogram :

1. Gambar yang ditampilkan merupakan hasil visualisasi histogram warna dari sebuah citra digital dalam format RGB, yang dibagi menjadi empat grafik : histogram gabungan dan histogram masing-masing channel warna (merah, hijau, biru). Grafik pertama di sebelah kiri adalah histogram gabungan RGB yang memperlihatkan distribusi intensitas seluruh piksel dari ketiga channel secara bersamaan. Terlihat bahwa sebagian besar nilai intensitas berada di kisaran 150 hingga 200, yang menandakan bahwa gambar cenderung memiliki pencahayaan yang cukup terang dan tidak banyak piksel yang sangat gelap atau sangat terang (hampir putih). Warna abu-abu pada grafik ini menandakan bahwa ini adalah representasi gabungan dan bukan spesifik ke satu warna.
2. Grafik kedua adalah histogram channel merah, yang menunjukkan bahwa sebagian besar nilai intensitas piksel merah berkisar antara 160 hingga 200, dengan puncak yang tajam. Ini mengindikasikan bahwa warna merah cukup dominan di dalam citra, terutama pada elemen gambar yang mengandung unsur warna merah seperti teks "PANGGABEAN". Banyaknya piksel pada nilai intensitas tinggi menunjukkan bahwa komponen merah muncul dalam bentuk cerah, bukan merah gelap.
3. Grafik ketiga menunjukkan histogram channel hijau. Distribusinya relatif mirip dengan channel merah, namun puncaknya sedikit bergeser dan mungkin sedikit lebih tersebar. Ini sesuai dengan keberadaan teks "FELIX" yang ditulis dengan warna hijau. Sama seperti channel merah, nilai dominan intensitas berada di kisaran menengah hingga tinggi (sekitar 150–200), menunjukkan bahwa komponen hijau juga cukup kuat dalam gambar.
4. Histogram keempat adalah channel biru, dan menunjukkan bentuk distribusi yang hampir serupa namun dengan kerapatan yang sedikit lebih rendah dibanding hijau dan merah. Intensitas piksel biru juga terkonsentrasi di sekitar 160–190, mencerminkan keberadaan teks "EWALDO" yang ditulis dengan warna biru. Meskipun lebih rendah dari merah dan hijau, channel biru tetap menunjukkan bahwa warnanya jelas terlihat dan memiliki kontras yang baik terhadap latar belakang. Secara keseluruhan, histogram ini menunjukkan bahwa gambar memiliki pencahayaan yang baik, dominasi warna merah, hijau, dan biru yang cukup seimbang namun dengan intensitas yang tinggi, serta kontras warna yang jelas di elemen

UTS.

gambar. Histogram seperti ini sangat berguna dalam menganalisis ciri visual dan distribusi warna, yang dapat diterapkan dalam pengolahan citra lebih lanjut seperti peningkatan kontras, segmentasi berbasis warna, atau klasifikasi gambar.

3.2 MEMPERBAIKI GAMBAR BACKLIGHT.

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Baca gambar dan konversi ke HSV

img_bgr = cv2.imread('NAMA.jpg')

img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)

# Definisikan rentang warna biru dalam HSV

blue_range = {

    'lower': np.array([100, 50, 50]),

    'upper': np.array([130, 255, 255])

}

# Buat masker untuk area biru

blue_mask = cv2.inRange(img_hsv, blue_range['lower'], blue_range['upper'])

# Terapkan masking pada gambar asli

blue_segment = cv2.bitwise_and(img_bgr, img_bgr, mask=blue_mask)

# Tampilkan hasil menggunakan subplot

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

# Gambar asli dalam RGB
```

UTS.

```
ax1.imshow(cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB))

ax1.set_title('Citra Asli')

ax1.axis('off')

# Masker biru sebagai citra grayscale

ax2.imshow(blue_mask, cmap='gray')

ax2.set_title('Blue')

ax2.axis('off')

plt.tight_layout()

plt.show()
```

Penjelasan :

```
import cv2

import numpy as np

import matplotlib.pyplot as plt
```

- = Mengimpor pustaka yang dibutuhkan : cv2 (OpenCV) untuk pemrosesan gambar, numpy untuk manipulasi array, matplotlib.pyplot untuk menampilkan gambar.

Baca gambar dan konversi ke HSV

```
img_bgr = cv2.imread('NAMA.jpg')
```

- = Membaca gambar 'NAMA.jpg' dalam format warna BGR (standar OpenCV) dan menyimpannya ke variabel img_bgr.

```
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
```

- = Mengonversi gambar dari format warna BGR ke HSV (Hue, Saturation, Value) dan menyimpannya sebagai img_hsv — HSV lebih efektif untuk segmentasi warna.

Definisikan rentang warna biru dalam HSV

UTS.

```
blue_range = {  
    'lower': np.array([100, 50, 50]),  
    'upper': np.array([130, 255, 255])  
}
```

- = Mendefinisikan batas bawah (lower) dan atas (upper) warna biru dalam format HSV. Nilai ini digunakan untuk mengenali warna biru.

```
# Buat masker untuk area biru
```

```
blue_mask = cv2.inRange(img_hsv, blue_range['lower'], blue_range['upper'])
```

- = Membuat masker biner: piksel dalam rentang biru diberi nilai 255 (putih), sisanya 0 (hitam). Digunakan untuk menandai area biru.

```
# Terapkan masking pada gambar asli
```

```
blue_segment = cv2.bitwise_and(img_bgr, img_bgr, mask=blue_mask)
```

- = Mengambil bagian gambar asli (img_bgr) yang sesuai dengan area biru menggunakan operasi logika AND dengan masker. Hasil disimpan dalam blue_segment.

```
# Tampilkan hasil menggunakan subplot
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
```

- = Membuat dua subplot horizontal (1 baris, 2 kolom) untuk menampilkan dua gambar: gambar asli dan mask biru.

```
# Gambar asli dalam RGB
```

```
ax1.imshow(cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB))
```

```
ax1.set_title('Citra Asli')
```

```
ax1.axis('off')
```

- = Menampilkan gambar asli setelah dikonversi ke RGB (karena matplotlib menggunakan RGB). Axis disembunyikan dan diberi judul "Citra Asli".

```
# Masker biru sebagai citra grayscale
```

UTS.

```
ax2.imshow(blue_mask, cmap='gray')
```

```
ax2.set_title('Blue')
```

```
ax2.axis('off')
```

- = Menampilkan masker warna biru sebagai gambar grayscale. Hanya area biru akan tampak terang (putih).

```
plt.tight_layout()
```

```
plt.show()
```

- = Menata layout agar rapi dan menampilkan hasil akhir.



Penjelasan :

- Menampilkan area tulisan berwarna biru saja dari gambar yang mengandung tulisan warna-warna dengan bantuan pemrosesan citra berbasis warna HSV. Gambar Kiri (Citra Asli) : Ditampilkan dengan imshow() setelah dikonversi ke RGB, karena OpenCV membaca gambar dalam format BGR, bukan RGB. Gambar Kanan (Blue) : Menampilkan hasil masking warna biru (dengan cv2.inRange), yaitu hanya tulisan "EWALDO" yang berada dalam rentang warna biru yang telah ditentukan. Hanya tulisan "EWALDO" yang tampak pada masker biru karena hanya itu yang sesuai rentang HSV biru. Tulisan "FELIX" (hijau) dan "PANGGABEAN" (merah) tidak muncul di gambar kanan karena tidak termasuk dalam rentang warna biru.

```
import cv2
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
# Baca gambar dan ubah ke HSV
```

UTS.

```
image = cv2.imread('NAMA.jpg')

hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Fungsi pembuat masker dari rentang warna

def buat_masker(hsv, batas_bawah, batas_atas):

    return cv2.inRange(hsv, batas_bawah, batas_atas)

# Daftar rentang warna

warna_biru = ([100, 50, 50], [130, 255, 255])

merah_1 = ([0, 50, 50], [10, 255, 255])

merah_2 = ([170, 50, 50], [180, 255, 255])

# Buat masker masing-masing warna

mask_biru = buat_masker(hsv_image, np.array(warna_biru[0]), np.array(warna_biru[1]))

mask_merah1 = buat_masker(hsv_image, np.array(merah_1[0]), np.array(merah_1[1]))

mask_merah2 = buat_masker(hsv_image, np.array(merah_2[0]), np.array(merah_2[1]))

# Gabungkan kedua masker merah

mask_merah = cv2.bitwise_or(mask_merah1, mask_merah2)

# Gabungkan merah dan biru

mask_final = cv2.bitwise_or(mask_biru, mask_merah)

# Tampilkan hasil

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

axes[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

axes[0].set_title('Citra Asli')
```

UTS.

```
axes[0].axis('off')

axes[1].imshow(mask_final, cmap='gray')
axes[1].set_title('RED BLUE')
axes[1].axis('off')

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

Penjelasan :

- Import Library.

```
import cv2

import numpy as np

import matplotlib.pyplot as plt
```

- = cv2 : digunakan untuk membaca dan memproses gambar (OpenCV).
- Numpy : untuk operasi array, terutama dalam definisi rentang warna.
- matplotlib.pyplot : untuk menampilkan gambar hasil pemrosesan.

- Membaca dan Mengubah Gambar ke HSV.

```
image = cv2.imread('NAMA.jpg')

hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

- = cv2.imread('NAMA.jpg') : membaca gambar berformat BGR (default OpenCV).
- cv2.cvtColor(..., cv2.COLOR_BGR2HSV) : mengonversi gambar ke ruang warna HSV karena lebih cocok untuk deteksi warna.

- Fungsi Pembuat Masker.

```
def buat_masker(hsv, batas_bawah, batas_atas):

    return cv2.inRange(hsv, batas_bawah, batas_atas)

• = Fungsi untuk membuat masker biner :
```

UTS.

- Piksel yang berada dalam batas_bawah dan batas_atas akan menjadi putih (255).
- Sisanya menjadi hitam (0).

- Definisi Rentang Warna dalam HSV.

```
warna_biru = ([100, 50, 50], [130, 255, 255])
```

```
merah_1 = ([0, 50, 50], [10, 255, 255])
```

```
merah_2 = ([170, 50, 50], [180, 255, 255])
```

- = Biru : Hue sekitar 100–130.
- Merah : karena merah berada di ujung hue (0° dan 180°), dibagi jadi dua bagian: merah_1 dan merah_2.
- Membuat Masker untuk Masing-masing Warna.

```
mask_biru = buat_masker(hsv_image, np.array(warna_biru[0]), np.array(warna_biru[1]))
```

```
mask_merah1 = buat_masker(hsv_image, np.array(merah_1[0]), np.array(merah_1[1]))
```

```
mask_merah2 = buat_masker(hsv_image, np.array(merah_2[0]), np.array(merah_2[1]))
```

- = Menghasilkan 3 buah masker biner (hitam-putih) untuk warna biru, merah bagian awal (hue rendah), dan merah bagian akhir (hue tinggi).

- Gabungkan Masker Merah.

```
mask_merah = cv2.bitwise_or(mask_merah1, mask_merah2)
```

- = Karena warna merah dibagi dua rentang, kita gabungkan keduanya.

- Gabungkan Masker Merah dan Biru.

```
mask_final = cv2.bitwise_or(mask_biru, mask_merah)
```

- = Sekarang, mask_final menandai semua piksel yang termasuk biru atau merah.

- Tampilkan Gambar.

```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

- = Membuat dua subplot horizontal (1 baris, 2 kolom) dengan ukuran besar.

UTS.

```
axes[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

```
axes[0].set_title('Citra Asli')
```

```
axes[0].axis('off')
```

- = Gambar asli ditampilkan setelah dikonversi dari BGR ke RGB agar warnanya benar.

```
axes[1].imshow(mask_final, cmap='gray')
```

```
axes[1].set_title('RED BLUE')
```

```
axes[1].axis('off')
```

- = Menampilkan masker akhir (mask_final) dalam format grayscale.

```
plt.tight_layout()
```

```
plt.show()
```

- = Merapikan tata letak plot dan menampilkan hasilnya.



Penjelasan :

- Output gambar tersebut merupakan hasil segmentasi warna dari citra asli menggunakan ruang warna HSV berdasarkan kode Python yang kamu sertakan. Gambar kiri menunjukkan citra asli dengan tulisan "FELIX EWALDO PANGGABEAN", masing-masing ditulis dengan warna hijau, biru, dan merah. Gambar kanan berjudul "RED BLUE" adalah hasil dari proses deteksi warna, di mana hanya huruf berwarna merah ("PANGGABEAN") dan biru ("EWALDO") yang ditampilkan sebagai warna putih dengan latar belakang hitam. Dalam kode, gambar dibaca dan dikonversi ke ruang warna HSV agar lebih mudah dalam menentukan rentang warna tertentu. Kemudian : Fungsi buat_masker() digunakan untuk membuat masker biner berdasarkan batas bawah dan atas nilai HSV untuk warna tertentu. Tiga rentang warna didefinisikan: satu untuk biru ([100, 50, 50] hingga [130, 255, 255]) dan dua untuk merah, karena warna merah berada di dua ujung spektrum HSV ([0, 50, 50]–[10, 255, 255] dan [170, 50, 50]–[180, 255, 255]). Masker merah digabungkan menggunakan

cv2.bitwise_or() agar mencakup seluruh rentang merah. Kemudian masker merah dan biru digabungkan lagi untuk menghasilkan mask_final, yang menandai semua piksel dalam gambar yang termasuk dalam dua rentang warna tersebut. Terakhir, hasil mask_final ditampilkan sebagai gambar grayscale, di mana piksel putih menunjukkan area yang terdeteksi sebagai merah atau biru, sedangkan piksel hitam menunjukkan area lainnya. Dengan demikian, output menunjukkan bahwa hanya kata "EWALDO" dan "PANGGABEAN" yang terdeteksi karena hanya mereka yang memiliki warna dalam rentang merah dan biru, sedangkan "FELIX" berwarna hijau dan tidak termasuk dalam rentang yang didefinisikan, sehingga tidak muncul di hasil deteksi.

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Fungsi bantu untuk membuat masker dari rentang HSV

def buat_mask(hsv_img, batas):

    mask = np.zeros(hsv_img.shape[:2], dtype=np.uint8)

    for lower, upper in batas:

        mask = cv2.bitwise_or(mask, cv2.inRange(hsv_img, np.array(lower), np.array(upper)))

    return mask

# Baca citra dan konversi ke HSV

gambar = cv2.imread('NAMA.jpg')

gambar_hsv = cv2.cvtColor(gambar, cv2.COLOR_BGR2HSV)

# Definisikan rentang HSV untuk masing-masing warna

rentang_biru = [[(100, 50, 50), (130, 255, 255)]] 

rentang_merah = [[(0, 50, 50), (10, 255, 255)], [(170, 50, 50), (180, 255, 255)]] 

rentang_hijau = [[(30, 25, 25), (120, 255, 255)]] 

# Buat masker untuk masing-masing warna

mask_biru = buat_mask(gambar_hsv, rentang_biru)
```

UTS.

```
mask_merah = buat_mask(gambar_hsv, rentang_merah)
mask_hijau = buat_mask(gambar_hsv, rentang_hijau)

# Gabungkan semua masker warna
mask_final = cv2.bitwise_or(cv2.bitwise_or(mask_biru, mask_merah), mask_hijau)

# Terapkan masker ke citra asli
hasil = cv2.bitwise_and(gambar, gambar, mask=mask_final)

# Tampilkan citra asli dan hasil masker
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

axes[0].imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))
axes[0].set_title('Citra Asli')
axes[0].axis('off')

axes[1].imshow(mask_final, cmap='gray')
axes[1].set_title('RED-GREEN-BLUE')
axes[1].axis('off')

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

- Penjelasan :

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- = Mengimpor tiga library penting : cv2 untuk operasi pengolahan citra, numpy untuk manipulasi array, matplotlib.pyplot untuk menampilkan hasil dalam bentuk grafik.

```
# Fungsi bantu untuk membuat masker dari rentang HSV

def buat_mask(hsv_img, batas):

    mask = np.zeros(hsv_img.shape[:2], dtype=np.uint8)

    for lower, upper in batas:

        mask = cv2.bitwise_or(mask, cv2.inRange(hsv_img, np.array(lower), np.array(upper)))

    return mask
```

- = Membuat fungsi buat_mask() yang menerima gambar dalam format HSV dan daftar batas warna HSV. Fungsi ini : Menginisialisasi masker kosong (semua nol). Melakukan iterasi pada setiap pasangan lower dan upper HSV, Membuat masker warna menggunakan cv2.inRange() lalu digabung menggunakan bitwise_or.

```
# Baca citra dan konversi ke HSV

gambar = cv2.imread('NAMA.jpg')

gambar_hsv = cv2.cvtColor(gambar, cv2.COLOR_BGR2HSV)
```

- = Membaca gambar dari file "NAMA.jpg". Mengonversi gambar dari format BGR ke HSV untuk mempermudah segmentasi warna.

```
# Definisikan rentang HSV untuk masing-masing warna

rentang_biru = [[(100, 50, 50), [130, 255, 255]]]

rentang_merah = [[([0, 50, 50], [10, 255, 255]), ([170, 50, 50], [180, 255, 255])]

rentang_hijau = [[([30, 25, 25], [120, 255, 255])]
```

- = Menentukan rentang warna HSV untuk warna biru, merah (dua rentang karena merah ada di dua ujung spektrum), dan hijau.

```
# Buat masker untuk masing-masing warna

mask_biru = buat_mask(gambar_hsv, rentang_biru)

mask_merah = buat_mask(gambar_hsv, rentang_merah)

mask_hijau = buat_mask(gambar_hsv, rentang_hijau)
```

UTS.

- = Menghasilkan masker biner untuk masing-masing warna menggunakan fungsi buat_mask() yang telah didefinisikan sebelumnya.

```
# Gabungkan semua masker warna
```

```
mask_final = cv2.bitwise_or(cv2.bitwise_or(mask_biru, mask_merah), mask_hijau)
```

- = Menggabungkan ketiga masker (biru, merah, hijau) menjadi satu masker akhir (mask_final) menggunakan operasi logika OR.

```
# Terapkan masker ke citra asli
```

```
hasil = cv2.bitwise_and(gambar, gambar, mask=mask_final)
```

- = Menerapkan mask_final ke gambar asli. Hanya bagian yang sesuai dengan masker (area yang terdeteksi warnanya) yang ditampilkan; bagian lain menjadi hitam.

```
# Tampilkan citra asli dan hasil masker
```

```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

- = Membuat dua subplot berdampingan dengan ukuran 12x5 untuk menampilkan dua gambar.

```
axes[0].imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))
```

```
axes[0].set_title('Citra Asli')
```

```
axes[0].axis('off')
```

- = Menampilkan gambar asli dalam subplot pertama. Konversi BGR ke RGB dilakukan agar warna tampil dengan benar di matplotlib.

```
axes[1].imshow(mask_final, cmap='gray')
```

```
axes[1].set_title('RED-GREEN-BLUE')
```

```
axes[1].axis('off')
```

- = Menampilkan mask_final dalam grayscale pada subplot kedua. Putih berarti warna terdeteksi, hitam berarti tidak.

UTS.

```
plt.tight_layout()
```

```
plt.show()
```

- = Menata layout agar tidak saling tumpang tindih. Menampilkan kedua gambar.



Penjelasan : Output gambar yang ditampilkan di atas terdiri dari dua panel: Citra Asli di sebelah kiri dan RED-GREEN-BLUE di sebelah kanan. Panel Kiri – "Citra Asli" Merupakan gambar asli (NAMA.jpg) yang berisi teks : FELIX ditulis dengan warna hijau, EWALDO dengan warna biru dan PANGGABEAN dengan warna merah. Gambar ini ditampilkan pada bagian :

```
axes[0].imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))
```

```
axes[0].set_title('Citra Asli')
```

Panel Kanan – "RED-GREEN-BLUE"

Merupakan hasil segmentasi warna menggunakan model HSV (Hue, Saturation, Value). Di panel ini : Tulisan-tulisan berwarna hijau, biru, dan merah tampil dalam warna putih, Latar belakang tetap hitam. Ini adalah hasil dari proses deteksi warna spesifik, yang dijalankan dalam urutan berikut :

Konversi gambar ke HSV.

```
gambar_hsv = cv2.cvtColor(gambar, cv2.COLOR_BGR2HSV)
```

= Konversi ini penting karena HSV lebih efektif untuk segmentasi warna dibanding RGB.

Pendefinisian rentang HSV.

```
rentang_biru = [[(100, 50, 50), [130, 255, 255]]]
```

```
rentang_merah = [[(0, 50, 50), [10, 255, 255]], [(170, 50, 50), [180, 255, 255]]]
```

```
rentang_hijau = [[(30, 25, 25), [120, 255, 255]]]
```

= Setiap warna memiliki rentang hue-nya masing-masing. Untuk merah, ada dua rentang karena hue merah ada di awal dan akhir spektrum HSV.

Pembuatan masker warna.

```
mask_biru = buat_mask(gambar_hsv, rentang_biru)
```

UTS.

```
mask_merah = buat_mask(gambar_hsv, rentang_merah)
```

```
mask_hijau = buat_mask(gambar_hsv, rentang_hijau)
```

= Fungsi buat_mask menggunakan cv2.inRange() untuk membuat gambar biner (hitam-putih) berdasarkan rentang warna.

Penggabungan masker.

```
mask_final = cv2.bitwise_or(cv2.bitwise_or(mask_biru, mask_merah), mask_hijau)
```

= Semua masker digabung menjadi satu masker akhir yang mencakup semua teks berwarna.

Menampilkan hasil akhir.

```
axes[1].imshow(mask_final, cmap='gray')
```

```
axes[1].set_title('RED-GREEN-BLUE')
```

= Hasilnya adalah gambar biner di mana area yang terdeteksi sebagai merah, hijau, atau biru ditampilkan sebagai putih, dan area lainnya (latar) tetap hitam. Gambar ini menunjukkan keberhasilan segmentasi warna pada teks menggunakan model HSV. Teknik ini sangat bermanfaat untuk aplikasi seperti deteksi objek, OCR warna, atau preprocessing citra sebelum klasifikasi.

```
import cv2
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
def tampilan_citra(citra, judul="Citra"):
```

```
    rgb_image = cv2.cvtColor(citra, cv2.COLOR_BGR2RGB)
```

```
    plt.imshow(rgb_image)
```

```
    plt.title(judul)
```

```
    plt.axis('off')
```

```
    plt.show()
```

```
# Baca citra dan konversi ke HSV
```

```
citra_asli = cv2.imread('NAMA.jpg')
```

```
citra_hsv = cv2.cvtColor(citra_asli, cv2.COLOR_BGR2HSV)
```

UTS.

```
# Daftar rentang HSV untuk warna
```

```
rentang_hsv = {
```

```
    'biru': ([100, 50, 50], [130, 255, 255]),
```

```
    'merah1': ([0, 50, 50], [10, 255, 255]),
```

```
    'merah2': ([170, 50, 50], [180, 255, 255]),
```

```
    'hijau': ([50, 50, 50], [80, 255, 255])
```

```
}
```

```
# Buat masker untuk tiap warna
```

```
masker_biru = cv2.inRange(citra_hsv, np.array(rentang_hsv['biru'][0]),  
np.array(rentang_hsv['biru'][1]))
```

```
masker_merah1 = cv2.inRange(citra_hsv, np.array(rentang_hsv['merah1'][0]),  
np.array(rentang_hsv['merah1'][1]))
```

```
masker_merah2 = cv2.inRange(citra_hsv, np.array(rentang_hsv['merah2'][0]),  
np.array(rentang_hsv['merah2'][1]))
```

```
masker_hijau = cv2.inRange(citra_hsv, np.array(rentang_hsv['hijau'][0]),  
np.array(rentang_hsv['hijau'][1]))
```

```
# Gabungkan semua masker
```

```
masker_total = masker_biru | masker_merah1 | masker_merah2 | masker_hijau
```

```
# Buat citra hitam kosong (dalam format BGR)
```

```
citra_hitam = np.zeros_like(citra_asli)
```

```
# Aplikasikan masking pada citra hitam
```

```
hasil = cv2.bitwise_and(citra_hitam, citra_hitam, mask=masker_total)
```

```
# Tampilkan hasil
```

```
tampilkan_citra(hasil, "None")
```

UTS.

Penjelasan :

```
import cv2
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

- = Import pustaka OpenCV, NumPy, dan Matplotlib.

```
def tampilan_citra(citra, judul="Citra"):
```

```
    rgb_image = cv2.cvtColor(citra, cv2.COLOR_BGR2RGB)
```

```
    plt.imshow(rgb_image)
```

```
    plt.title(judul)
```

```
    plt.axis('off')
```

```
    plt.show()
```

- = Fungsi bantu untuk menampilkan citra dengan konversi dari BGR ke RGB agar warna tampil benar di Matplotlib.

```
citra_asli = cv2.imread('NAMA.jpg')
```

```
citra_hsv = cv2.cvtColor(citra_asli, cv2.COLOR_BGR2HSV)
```

- = Membaca citra dan mengubahnya ke format HSV untuk pemrosesan warna.

```
rentang_hsv = {
```

```
    'biru': ([100, 50, 50], [130, 255, 255]),
```

```
    'merah1': ([0, 50, 50], [10, 255, 255]),
```

```
    'merah2': ([170, 50, 50], [180, 255, 255]),
```

```
    'hijau': ([50, 50, 50], [80, 255, 255])
```

```
}
```

- = Menentukan rentang HSV untuk masing-masing warna (biru, merah, hijau).

```
masker_biru = cv2.inRange(citra_hsv, np.array(rentang_hsv['biru'][0]),  
                           np.array(rentang_hsv['biru'][1]))
```

```
masker_merah1 = cv2.inRange(citra_hsv, np.array(rentang_hsv['merah1'][0]),  
                           np.array(rentang_hsv['merah1'][1]))
```

```
masker_merah2 = cv2.inRange(citra_hsv, np.array(rentang_hsv['merah2'][0]),  
                           np.array(rentang_hsv['merah2'][1]))
```

UTS.

```
masker_hijau = cv2.inRange(citra_hsv, np.array(rentang_hsv['hijau'][0]),  
np.array(rentang_hsv['hijau'][1]))
```

- = Membuat masker biner (hitam-putih) untuk setiap warna berdasarkan rentang HSV-nya.

```
masker_total = masker_biru | masker_merah1 | masker_merah2 | masker_hijau
```

- = Menggabungkan semua masker menjadi satu masker total.

```
citra_hitam = np.zeros_like(citra_asli)
```

- = Membuat citra kosong (hitam) dengan ukuran sama seperti citra asli.

```
hasil = cv2.bitwise_and(citra_asli, citra_asli, mask=masker_total)
```

- = Melakukan operasi bitwise AND antara citra_asli dengan dirinya sendiri, hanya pada area yang dipilih oleh masker_total. citra_asli: gambar asli (warna penuh). mask=masker_total: hanya bagian putih (nilai 255) dari masker yang diproses, sisanya (nilai 0) diabaikan (jadi hitam). cv2.bitwise_and(): mempertahankan warna asli hanya di bagian yang dimasker.

```
tampilkan_citra(hasil, "None")
```

- = Menampilkan hasil akhir citra bertopeng. Gambar baru (hasil) akan menampilkan hanya bagian teks berwarna (hijau, biru, merah) dari gambar asli di atas latar belakang hitam. Seperti memotong huruf berwarna dari kertas lalu meletakkannya di atas papan hitam.

None



Penjelasan : Output berupa gambar teks "FELIX EWALDO PANGGABEAN" berwarna di atas latar belakang hitam merupakan hasil langsung dari source code yang diberikan, yang menggunakan teknik deteksi warna berbasis HSV (Hue, Saturation, Value). Kode ini membaca gambar NAMA.jpg, lalu mengonversinya ke ruang warna HSV karena HSV lebih efektif untuk segmentasi warna dibandingkan RGB. Kemudian, kode mendefinisikan rentang HSV untuk tiga warna utama—biru, merah, dan hijau—yang tampaknya digunakan dalam teks pada gambar. Dua rentang merah digunakan karena warna merah terletak di ujung bawah dan atas spektrum HSV (sekitar 0 dan 180 derajat), sehingga perlu dua masker untuk menjangkau seluruh variasinya.

UTS.

Dengan cv2.inRange, kode membuat masker biner untuk setiap warna: area yang sesuai warna akan menjadi putih (255), sisanya hitam (0). Semua masker ini lalu digabungkan menjadi satu masker_total. Terakhir, fungsi cv2.bitwise_and() digunakan untuk mengekstrak hanya bagian gambar asli (citra_asli) yang sesuai dengan masker_total. Hasil akhirnya adalah gambar yang hanya menampilkan teks berwarna (yang sesuai rentang HSV), sementara semua area lain menjadi hitam. Fungsi tampilkan_citra() hanya digunakan untuk menampilkan hasil ini dengan matplotlib.

```
import cv2

import numpy as np

import matplotlib.pyplot as plt


def tampilkan_subplots(data_gambar, judul_gambar, layout=(2, 2), ukuran=(12, 8)):

    fig, axes = plt.subplots(*layout, figsize=ukuran)

    axes = axes.flatten()

    for idx, (gambar, judul) in enumerate(zip(data_gambar, judul_gambar)):

        if len(gambar.shape) == 2: # Grayscale

            axes[idx].imshow(gambar, cmap='gray')

        else: # Warna (BGR ke RGB)

            axes[idx].imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))

        axes[idx].set_title(judul)

        axes[idx].axis('on')

    plt.tight_layout()

    plt.show()


# Buat daftar gambar yang ingin ditampilkan

none_output = cv2.bitwise_and(black_image, black_image, mask=combined_mask)

gambar_list = [none_output, blue_mask, red_blue_mask, combined_mask]
```

UTS.

```
judul_list = ['None', 'Blue', 'Red-Blue', 'Red-Green-Blue']
```

```
# Tampilkan semua dalam subplot
```

```
tampilkan_subplots(gambar_list, judul_list)
```

- Penjelasan :

```
import cv2
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

- Import library OpenCV, NumPy, dan Matplotlib.

```
def tampilkan_subplots(data_gambar, judul_gambar, layout=(2, 2), ukuran=(12, 8)):
```

- Definisikan fungsi untuk menampilkan beberapa gambar sekaligus dalam bentuk grid.

```
fig, axes = plt.subplots(*layout, figsize=ukuran)
```

```
axes = axes.flatten()
```

- Buat subplot sesuai layout (default 2x2) dan ubah array axes menjadi 1 dimensi.

```
for idx, (gambar, judul) in enumerate(zip(data_gambar, judul_gambar)):
```

- Loop untuk setiap gambar dan judul.

```
if len(gambar.shape) == 2: # Grayscale
```

```
    axes[idx].imshow(gambar, cmap='gray')
```

- Jika gambar grayscale, tampilkan dengan colormap abu-abu.

```
else: # Warna (BGR ke RGB)
```

```
    axes[idx].imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))
```

- Jika gambar berwarna, konversi dari BGR ke RGB untuk ditampilkan dengan benar.

```
axes[idx].set_title(judul)
```

```
    axes[idx].axis('on')
```

- Atur judul dan tampilkan sumbu (axis).

```
plt.tight_layout()
```

```
plt.show()
```

UTS.

- Rapikan tata letak dan tampilkan plot.

```
none_output = cv2.bitwise_and(black_image, black_image, mask=combined_mask)
```

- Buat gambar none_output dari black_image dengan masker combined_mask.

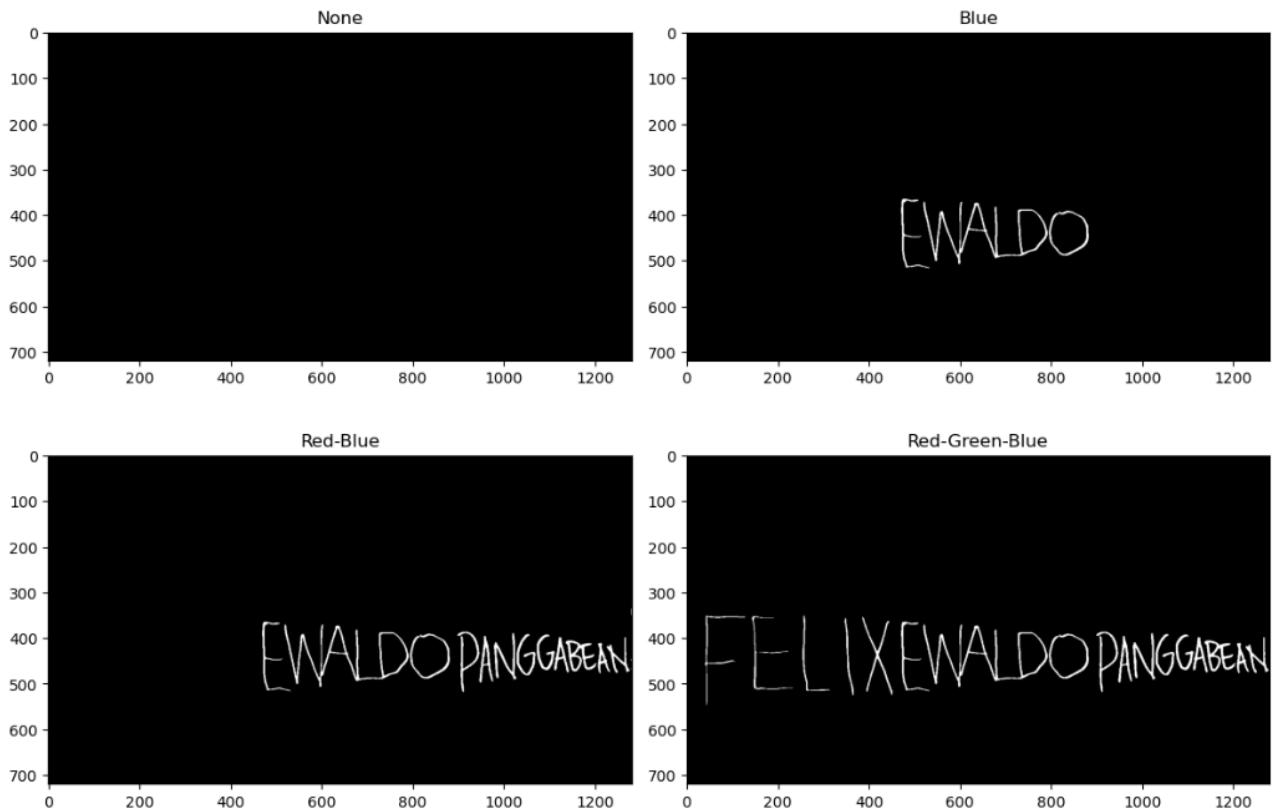
```
gambar_list = [none_output, blue_mask, red_blue_mask, combined_mask]
```

```
judul_list = ['None', 'Blue', 'Red-Blue', 'Red-Green-Blue']
```

- Daftar gambar dan judul untuk ditampilkan.

```
tampilkan_subplots(gambar_list, judul_list)
```

- Panggil fungsi untuk menampilkan semua gambar dalam subplot.



Penjelasan :

1. Output yang ditampilkan merupakan hasil visualisasi dari proses segmentasi warna terhadap gambar yang berisi teks nama berwarna, yaitu "FELIX EWALDO PANGGABEAN". Proses segmentasi ini dilakukan dengan menggunakan masker berbasis ruang warna HSV untuk mendeteksi tiga warna utama: biru, merah, dan hijau. Gambar tersebut ditampilkan dalam bentuk subplot berisi empat tampilan berbeda untuk memvisualisasikan bagaimana masing-masing masker bekerja dalam mengisolasi bagian-bagian tertentu dari teks berdasarkan warnanya.
2. Gambar pertama yang diberi label "None" menunjukkan hasil dari operasi bitwise_and antara citra hitam dengan citra hitam itu sendiri, menggunakan masker gabungan combined_mask. Karena citra dasarnya adalah gambar hitam polos (black_image),

meskipun masker sebenarnya sudah mencakup seluruh warna yang ada (merah, hijau, biru), hasil akhirnya tetap terlihat seperti gambar kosong atau hitam tanpa teks. Ini terjadi karena tidak ada informasi visual (warna) dalam black_image yang bisa dipertahankan oleh fungsi bitwise_and, sehingga teks tidak terlihat.

3. Gambar kedua yang berjudul "Blue" adalah hasil dari masker biru yang berhasil mengekstraksi bagian teks "EWALDO", karena pada gambar asli bagian ini ditulis menggunakan warna biru. Di sini, latar belakang tetap hitam sedangkan teks berwarna biru ditampilkan dalam bentuk putih (masker biner), menunjukkan keberhasilan segmentasi warna biru.
4. Selanjutnya, pada gambar ketiga yang berjudul "Red-Blue", tampak bagian teks "EWALDO PANGGABEAN" karena masker ini merupakan gabungan antara masker merah dan biru. Kata "EWALDO" muncul dari hasil segmentasi biru, sementara "PANGGABEAN" dari segmentasi merah. Namun, kata "FELIX" masih tidak terlihat karena bagian tersebut berwarna hijau dan belum termasuk dalam masker ini.
5. Gambar keempat yang berjudul "Red-Green-Blue" adalah hasil dari masker yang menggabungkan ketiga warna utama—merah, hijau, dan biru—yang digunakan dalam teks. Hasilnya menunjukkan keseluruhan teks "FELIX EWALDO PANGGABEAN" dengan sangat jelas dan lengkap, karena seluruh warna yang membentuk teks sudah berhasil dideteksi dan dimasukkan ke dalam masker. Dengan demikian, subplot ini menjadi visualisasi paling lengkap dari keseluruhan proses masking berdasarkan warna dalam gambar asli.

LAMPIRAN NILAI AMBANG BATAS HSV :

Channel	Rentang Hue (H)	Rentang Saturation (S)	Rentang Value (V)
Blue	100 - 130	100 - 255	50 - 255
Red	0 - 10 dan 160-180	100 - 255	50 - 255
Green	40 - 80	100 - 255	50 - 255
Red-Blue	Gabungan nilai Red dan Blue		-
Red-Green-Blue	Gabungan ketiga nilai HSV		-
None	Tidak diterapkan ambang HSV		-

ALASAN PEMILIHAN NILAI AMBANG :

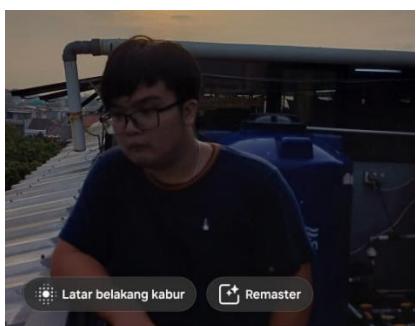
1. Rentang Hue (H).
 - Blue (100–130).
Dipilih karena warna biru dalam ruang HSV biasanya berada di kisaran hue 100 hingga 130. Nilai ini efektif untuk mengekstrak objek atau teks berwarna biru dari latar belakang gelap.
 - Red (0–10 dan 160–180).
Warna merah memiliki hue yang menyentuh batas bawah dan atas (wrap-around) dari spektrum HSV. Oleh karena itu, dua rentang digunakan: 0–10 dan 160–180 untuk menangkap keseluruhan warna merah secara akurat.

- Green (40–80).
Warna hijau terletak pada kisaran hue ini, sehingga rentang ini dipilih untuk mendeteksi objek atau teks hijau.
- 2. Rentang Saturation (S).
 - 100–255
Saturation rendah (0–100) cenderung menghasilkan warna keabu-abuan (kurang jenuh), yang bisa membuat teks sulit dibedakan dari latar belakang. Dengan memilih nilai jenuh tinggi, sistem lebih fokus pada warna-warna yang cerah dan tegas, seperti tinta atau cat pada objek nyata.
- 3. Rentang Value (V).
 - 50–255
Value terlalu rendah (0–50) sering kali terlalu gelap dan bisa mencampur objek dengan latar belakang hitam. Dengan batas bawah 50, deteksi bisa fokus pada objek yang cukup terang untuk dibedakan dari latar.
- 4. Kombinasi Channel (Red-Blue, Red-Green-Blue).
 - Digunakan untuk mendeteksi teks atau objek yang terdiri dari lebih dari satu warna. Misalnya, teks yang sebagian berwarna merah dan sebagian biru akan terdeteksi secara keseluruhan menggunakan kombinasi Red-Blue.
- 5. None.
 - Channel ini tidak menggunakan ambang HSV, berfungsi sebagai baseline (perbandingan awal tanpa filtering warna), untuk menunjukkan peran penting segmentasi warna.

3.3 MEMPERBAIKI GAMBAR BACKLIGHT.

- o Ambillah foto diri Anda dengan posisi menghadap langsung ke kamera dan membelakangi sumber cahaya matahari yang terang (backlight).
- o Gunakan logika serta teknik-teknik pengolahan citra yang telah Anda pelajari sebelumnya untuk mengolah gambar tersebut. Fokus utama adalah memperbaiki tampilan profil wajah atau tubuh Anda yang cenderung gelap akibat efek backlight.
- o Lakukan konversi gambar menjadi grayscale, kemudian tingkatkan kecerahan dan kontras khususnya pada area profil Anda sehingga lebih menonjol dibandingkan latar belakang yang terang.
- o Penilaian akan difokuskan pada seberapa efektif Anda membuat area profil menjadi pusat perhatian (fokus utama) dibandingkan dengan latar belakangnya.
- o Pengurangan nilai akan diberikan pada gambar yang mengalami efek color burn (terbakar warna) secara berlebihan. Namun, toleransi masih diberikan untuk efek color burn yang wajar dan tidak mengganggu kualitas visual citra.

UTS.



Rabu, 07 Mei 2025 · 16.47

Ubah

20250507_164717.jpg

/Penyimpanan internal/DCIM/Camera

Galaxy A34 5G Foto gerakan

5,62 MB | 2250x4000 | 9MP
ISO 50 | 25mm | -0,7ev | F1,8 | 1/3906 s



Jl. Komp. Interkota Indah No.D5, RT.12/RW.7,
Duri Kosambi, Kecamatan Cengkareng, Kota
Jakarta Barat, Daerah Khusus Ibukota Jakarta
11750, Indonesia

Tambah tag

```
import cv2
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

202331084.

Felix Ewaldo Panggabean.

UTS.

```
def ubah_kecerahan_kontras(image, alpha=1.0, beta=0):
    """Fungsi untuk mengubah kontras (alpha) dan kecerahan (beta)"""

    return cv2.convertScaleAbs(image, alpha=alpha, beta=beta)

def tampilan_semua(gambar_dict):
    plt.figure(figsize=(15, 10))

    for i, (judul, gambar) in enumerate(gambar_dict.items(), start=1):
        plt.subplot(2, 3, i)

        if len(gambar.shape) == 2:
            plt.imshow(gambar, cmap='gray')

        else:
            plt.imshow(cv2.cvtColor(gambar, cv2.COLOR_BGR2RGB))

        plt.title(judul)
        plt.axis("off")

    plt.tight_layout()
    plt.show()

# Baca gambar dan konversi
gambar_asli = cv2.imread("POTRET.jpg")
gambar_gray = cv2.cvtColor(gambar_asli, cv2.COLOR_BGR2GRAY)

# Proses
gray_terang = ubah_kecerahan_kontras(gambar_gray, alpha=1.0, beta=50)
gray_kontras = ubah_kecerahan_kontras(gambar_gray, alpha=1.5, beta=0)
gray_terang_kontras = ubah_kecerahan_kontras(gambar_gray, alpha=1.5, beta=50)

# Dictionary untuk ditampilkan
gambar_dict = {
```

UTS.

```
"Gambar Asli": gambar_asli,  
"Gambar Gray": gambar_gray,  
"Gray yang Dipencerah": gray_terang,  
"Gray yang Diperkontras": gray_kontras,  
"Gray Dipencerah & Kontras": gray_terang_kontras  
}  
  
# Tampilkan
```

```
tampilkan_semua(gambar_dict)
```

Penjelasan :

```
import cv2  
import numpy as np  
import matplotlib.pyplot as plt
```

- Mengimpor library: OpenCV, NumPy, dan Matplotlib.

```
def ubah_kecerahan_kontras(image, alpha=1.0, beta=0):
```

```
    return cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
```

- Fungsi untuk mengubah kontras (alpha) dan kecerahan (beta) pada gambar.

```
def tampilkan_semua(gambar_dict):
```

```
...
```

- Fungsi untuk menampilkan beberapa gambar dalam satu jendela menggunakan matplotlib.

```
gambar_asli = cv2.imread("POTRET.jpg")
```

- Membaca gambar berwarna dari file.

```
gambar_gray = cv2.cvtColor(gambar_asli, cv2.COLOR_BGR2GRAY)
```

- Mengubah gambar ke grayscale.

```
gray_terang = ubah_kecerahan_kontras(gambar_gray, alpha=1.0, beta=50)
```

- Mencerahkan gambar grayscale.

```
gray_kontras = ubah_kecerahan_kontras(gambar_gray, alpha=1.5, beta=0)
```

- Menambah kontras gambar grayscale.

UTS.

```
gray_terang_kontras = ubah_kecerahan_kontras(gambar_gray, alpha=1.5, beta=50)
```

- Meningkatkan kecerahan dan kontras sekaligus.

```
gambar_dict = { ... }
```

- Menyimpan semua gambar dalam dictionary untuk ditampilkan.

```
tampilkan_semua(gambar_dict)
```

- Menampilkan semua gambar dalam grid menggunakan subplot.



Penjelasan :

1. Gambar output di atas merupakan hasil visualisasi dari proses pengolahan citra menggunakan Python dan OpenCV, khususnya dalam hal manipulasi kecerahan dan kontras pada citra grayscale. Berdasarkan source code yang diberikan, program ini bertujuan untuk menunjukkan pengaruh perubahan nilai kontras dan kecerahan terhadap citra hitam putih (grayscale). Proses dimulai dengan membaca citra asli berwarna (dalam hal ini sebuah potret), lalu citra tersebut dikonversi ke mode grayscale menggunakan fungsi cv2.cvtColor dengan parameter cv2.COLOR_BGR2GRAY. Gambar grayscale inilah yang menjadi dasar untuk berbagai variasi transformasi.

2. Selanjutnya, dilakukan tiga proses utama menggunakan fungsi `ubah_kecerahan_kontras()` yang memanfaatkan metode `cv2.convertScaleAbs`. Fungsi ini mengatur nilai intensitas piksel berdasarkan dua parameter: alpha (untuk kontras) dan beta (untuk kecerahan). Gambar dengan judul “Gray yang Dipercerah” menunjukkan hasil dari peningkatan nilai beta (dari 0 menjadi 50) tanpa mengubah alpha, yang artinya gambar menjadi lebih terang secara keseluruhan namun tidak ada perubahan signifikan pada kontras antar objek. Hasilnya adalah gambar grayscale yang lebih cerah, namun kontras antar bayangan dan objek masih tergolong datar.
3. Pada gambar “Gray yang Diperkontras”, yang dimanipulasi adalah nilai alpha saja (dinaikkan dari 1.0 menjadi 1.5), sedangkan beta tetap 0. Hal ini menyebabkan perbedaan intensitas antar piksel menjadi lebih tajam—area gelap menjadi lebih gelap dan area terang menjadi lebih terang—sehingga tampilan gambar menjadi lebih dramatis dan detail objek terlihat lebih jelas. Perubahan ini sangat efektif untuk menonjolkan kontur atau bentuk dalam gambar monokrom.
4. Gambar terakhir, “Gray Dipercerah & Kontras”, merupakan kombinasi dari kedua modifikasi: alpha ditingkatkan menjadi 1.5 dan beta menjadi 50. Hasilnya adalah gambar grayscale yang tidak hanya lebih terang, tetapi juga lebih kontras. Perubahan ini memberikan efek visual yang signifikan karena detail objek terlihat lebih tajam dan gambar tampak menyala secara keseluruhan, sangat cocok untuk kebutuhan peningkatan visibilitas citra dalam kondisi pencahayaan rendah.
5. Akhirnya, semua gambar dikumpulkan ke dalam dictionary (`gambar_dict`) dan ditampilkan dalam grid 2x3 menggunakan `matplotlib`, sehingga pengguna dapat membandingkan efek setiap transformasi secara visual dan langsung. Proses ini sangat berguna dalam bidang pengolahan citra untuk analisis, pendekripsi fitur, atau sekadar meningkatkan kualitas gambar.

BAB IV

PENUTUP

4.2. KESIMPULAN.

- Berdasarkan hasil kajian dan pembahasan mengenai berbagai teknik dan konsep dalam pengolahan citra digital, dapat disimpulkan bahwa pengolahan citra merupakan bidang yang sangat penting dan aplikatif dalam berbagai aspek kehidupan modern. Teknik-teknik seperti peningkatan kualitas citra, pengolahan warna dan grayscale, morfologi citra, serta deteksi wajah menggunakan algoritma Haar Cascade memiliki peran besar dalam menghasilkan citra yang lebih informatif, bersih, dan siap digunakan untuk analisis lanjutan.
- Pemahaman tentang teori warna dan aras keabuan sangat membantu dalam menyesuaikan pengolahan citra dengan persepsi visual manusia, sehingga hasil yang diperoleh menjadi lebih akurat dan realistik. Penggunaan histogram, filter bilateral, dan teknik pengurangan noise juga terbukti efektif dalam meningkatkan kualitas visual dan memperjelas detail citra.
- Selain itu, teknik morfologi seperti dilasi dan erosi mempermudah dalam mengekstraksi bentuk serta struktur objek dalam citra. Di sisi lain, implementasi Haar Cascade menunjukkan efektivitas dalam deteksi wajah dengan berbagai tingkat pencahayaan dan kualitas citra, membuktikan potensinya dalam aplikasi keamanan maupun interaksi manusia-mesin.
- Secara keseluruhan, pengolahan citra digital memberikan manfaat signifikan dalam bidang medis, industri, keamanan, hiburan, dan teknologi informasi. Dengan terus berkembangnya teknologi perangkat keras dan algoritma, pengolahan citra digital diperkirakan akan menjadi bagian tak terpisahkan dari solusi berbasis visual di masa depan.

4.2. SARAN.

Untuk pengembangan lebih lanjut, beberapa saran dapat diberikan sebagai berikut :

- Pendalaman terhadap algoritma lanjutan : Mahasiswa dan peneliti disarankan untuk mempelajari metode pengolahan citra berbasis machine learning atau deep learning (seperti CNN), yang kini mulai menggantikan metode konvensional dalam tugas-tugas klasifikasi dan segmentasi citra.
- Eksplorasi dataset citra nyata : Penggunaan citra dari dataset dunia nyata (seperti citra medis, satelit, atau wajah dari kondisi lapangan) dapat memberikan pemahaman lebih praktis dan realistik terhadap tantangan dalam pengolahan citra.
- Optimasi performa sistem : Penting untuk memperhatikan efisiensi waktu dan sumber daya dalam pengolahan citra, terutama ketika diaplikasikan dalam sistem real-time seperti CCTV, kendaraan otonom, atau augmented reality.
- Integrasi dengan teknologi lain : Pengolahan citra digital sebaiknya mulai digabungkan dengan teknologi lain seperti Internet of Things (IoT), cloud computing, dan sistem embedded untuk memperluas aplikasinya dalam dunia nyata.

UTS.

- Pengujian dalam kondisi ekstrem : Pengolahan citra perlu diuji dalam kondisi citra ekstrem (misalnya dengan noise tinggi, pencahayaan buruk, atau objek tertutup sebagian) untuk meningkatkan robustnes dan reliabilitas sistem yang dikembangkan. Dengan mengikuti perkembangan teknologi dan memperdalam pemahaman terhadap teknik-teknik pengolahan citra, diharapkan penelitian dan implementasi dalam bidang ini dapat memberikan solusi yang lebih inovatif, efisien, dan berdampak luas bagi masyarakat.

DAFTAR PUSTAKA

1. *Yulina, S. (2021). Implementation of Haar Cascade Classifier for Face Detection and Grayscale Image Transformation Using OpenCV. Jurnal Komputer Terapan, 7(1), 100–109. <https://doi.org/10.35143/jkt.v7i1.3411>*
2. *Isnanto, R. R., Rochim, A. F., Eridani, D., & Cahyono, G. D. (2021). Multi-Object Face Recognition Using Local Binary Pattern Histogram and Haar Cascade Classifier on Low-Resolution Images. International Journal of Engineering and Technology Innovation, 11(1), 1–10. <https://doi.org/10.46604/ijeti.2021.6174>*
3. *Saraswati, N. M., Hariyono, R. C. S., & Chandra, D. (2021). Face Recognition Menggunakan Metode Haar Cascade Classifier dan Local Binary Pattern Histogram. Jurnal Media Elektrik, 20(3), 45–52. <https://doi.org/10.59562/metrik.v20i3.5546>*
4. *Kristianto, R. P. (2024). Pengembangan Face Recognition Menggunakan OpenCV dan Kombinasi Algoritma Haarcascade dan Local Binary Pattern Histogram (LBPH) untuk Aplikasi Presensi Mahasiswa. Smart Comp: Jurnalnya Orang Pintar Komputer, 13(4), 25–33. <https://ejournal.poltekharber.ac.id/index.php/smartcomp/article/view/7083>*
5. *Liu, W., Wang, L., & Cui, M. (2023). Quantum Image Segmentation Based on Grayscale Morphology. arXiv preprint arXiv:2311.11952.*
6. *Fahmi, M., Yudhana, A., & Sunardi, S. (2023). Image Processing Using Morphology on Support Vector Machine Classification Model for Waste Image. MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, 22(3), 553–566.*
7. *Yeh, J.-F., Lin, K.-M., Chang, C.-C., & Wang, T.-H. (2023). Expression Recognition of Multiple Faces Using a Convolution Neural Network Combining the Haar Cascade Classifier. Applied Sciences, 13(23), 12737. <https://doi.org/10.3390/app132312737>*
8. *Zeebaree, I. M., & Kareem, O. S. (2023). Face Mask Detection Using Haar Cascades Classifier To Reduce The Risk Of COVID-19. International Journal of Mathematics, Statistics, and Computer Science, 2, 19–27. <https://doi.org/10.59543/ijmscs.v2i.7845>*
9. *Damarsiwi, D. K., Pambudi, E. A., Fitriani, M. A., & Wibowo, F. (2024). Face Detection in Complex Background using Scale Invariant Feature Transform and Haar Cascade Classifier Methods. Sinkron: Jurnal dan Penelitian Teknik Informatika, 8(2), 852–860. <https://doi.org/10.33395/sinkron.v8i2.13556>*
10. *Musa, A. (2025). Face Detection Based on Haar Cascade and Convolution Neural Network (CNN). Journal of Advanced Research in Computing, 38(1). <https://doi.org/10.37934/arca.38.1.111>*