UTS

PENGOLAHAN CITRA

NAMA : Muh Qhofi Rafiatul

NIM : 202331127

KELAS : C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC :

ASISTEN : 1. Cynthia Maharani

2. Renata Yasmine Selomita

3. Erin Katholica Sakraly Putri Marrison

4. Yasa Hilda An Faza

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Apa saja jenis-jenis filter yang dapat digunakan untuk meningkatkan kualitas gambar dalam pemrosesan citra digital.
2. Bagaimana cara menghitung dan memvisualisasikan histogram dari citra digital untuk analisis lebih lanjut?
3. Apa pengaruh teknik filtering terhadap histogram gambar dalam hal distribusi intensitas piksel?

1.2 Tujuan Masalah

1. Mengetahui bagaimana ketetanggaan piksel dalam citra dapat memengaruhi pemrosesan gambar secara keseluruhan.
2. Mengeksplorasi berbagai metode filtering yang dapat diterapkan untuk meningkatkan kualitas gambar, seperti filter rata-rata, median, dan gaussian.
3. Mengukur pengaruh teknik filtering terhadap histogram gambar untuk memahami perubahan yang terjadi pada distribusi intensitas piksel setelah pemrosesan.

1.3 Manfaat Masalah

1. Memberikan wawasan tentang pentingnya ketetanggaan piksel dalam analisis gambar dan bagaimana hal itu memengaruhi teknik pemrosesan citra digital.
2. Memperkenalkan cara praktis menghitung dan menganalisis histogram gambar untuk mempermudah analisis citra dan membantu dalam aplikasi seperti segmentasi citra dan deteksi tepi.
3. Meningkatkan pemahaman tentang berbagai metode filtering dalam meningkatkan kualitas gambar, yang bermanfaat dalam aplikasi.

BAB II

LANDASAN TEORI

Pemrosesan citra digital adalah proses manipulasi dan analisis gambar menggunakan algoritma komputer untuk menghasilkan informasi yang berguna. Salah satu konsep dasar dalam pemrosesan citra digital adalah ketetanggaan piksel (pixel adjacency), yang mengacu pada hubungan antara piksel-piksel dalam citra. Dalam konteks citra dua dimensi, piksel yang berdekatan secara spatial memiliki hubungan ketetanggaan yang memengaruhi teknik pemrosesan seperti deteksi tepi, pemfilteran, dan segmentasi citra. Ketetanggaan piksel ini memengaruhi cara algoritma memproses gambar dan memberikan hasil yang lebih tajam atau lebih halus sesuai dengan konteks aplikasi.

Filter dalam pemrosesan citra digunakan untuk meningkatkan kualitas gambar atau mengekstraksi informasi penting dengan memodifikasi intensitas piksel. Salah satu bentuk filtering yang paling umum adalah filter konvolusi, yang melibatkan operasi di mana sebuah matriks filter atau kernel diterapkan ke setiap piksel dalam gambar untuk menghasilkan nilai piksel yang baru. Metode ini dapat digunakan untuk berbagai tujuan, seperti mengurangi noise, mempertajam citra, atau mendeteksi tepi. Filtering berbasis ketetanggaan piksel ini memungkinkan analisis lebih dalam terhadap hubungan spasial antar elemen dalam gambar dan meningkatkan kualitas visual citra, yang sangat berguna dalam aplikasi pengolahan citra medis dan industri.

Seiring dengan perkembangan teknologi, histogram gambar menjadi alat penting dalam analisis citra digital. Histogram adalah representasi grafis dari distribusi intensitas piksel dalam citra. Setiap titik dalam histogram mewakili jumlah piksel yang memiliki nilai intensitas tertentu. Histogram memberikan gambaran menyeluruh tentang kontras, pencahayaan, dan distribusi warna dalam gambar. Proses analisis histogram sering kali digunakan dalam berbagai teknik pemrosesan gambar, seperti pemilihan thresholding, deteksi objek, dan koreksi pencahayaan. Dalam konteks ini, histogram menjadi dasar dalam evaluasi kualitas gambar dan dalam mengatur pengolahan citra melalui teknik-teknik seperti pemfilteran atau penyesuaian kontras.

Proses filtering berfungsi untuk memodifikasi gambar dengan mengubah intensitas piksel berdasarkan informasi yang ada di sekitarnya. Salah satu contoh teknik filtering yang banyak digunakan adalah filter rata-rata (mean filter), yang menghitung rata-rata intensitas piksel di sekitarnya dan menggantikan nilai piksel tersebut dengan rata-rata tersebut. Meskipun efektif untuk mengurangi noise, filter rata-rata dapat mengurangi ketajaman detail citra. Oleh karena itu, filter lainnya, seperti filter median, sering digunakan untuk mengatasi masalah tersebut. Filter median menggantikan setiap piksel dengan nilai median dari intensitas piksel yang ada di sekitarnya, yang lebih efektif dalam mengurangi noise impulsif atau "salt and pepper".

Selain filter rata-rata dan median, filter Gaussian adalah jenis filter yang sering digunakan dalam pemrosesan citra untuk menghaluskan citra. Filter ini menggunakan distribusi Gaussian untuk menghitung bobot piksel di sekitar piksel target. Konvolusi dengan filter Gaussian memberikan efek blur yang efektif untuk mengurangi noise dan menghaluskan

detail gambar, yang berguna dalam pengurangan noise latar belakang atau pemrosesan citra sebelum dilakukan analisis lebih lanjut, seperti deteksi tepi. Meskipun efek blur ini mengurangi ketajaman citra, ia memberikan kontribusi penting dalam aplikasi seperti deteksi fitur dan segmentasi.

Histogram gambar sangat berguna dalam menentukan kontras dan kualitas visual citra. Dalam gambar dengan kontras rendah, intensitas piksel tersebar di rentang yang sangat sempit, sehingga mengurangi detail visual. Sebaliknya, gambar dengan kontras tinggi memiliki distribusi intensitas piksel yang lebih merata. Dalam proses pemrosesan gambar, teknik seperti histogram equalization dapat digunakan untuk meningkatkan kontras gambar. Teknik ini meratakan distribusi intensitas piksel di seluruh rentang yang memungkinkan gambar menjadi lebih terang dan lebih kontras, sehingga meningkatkan visibilitas detail penting dalam gambar.

Pemrosesan citra dengan menggunakan ketetanggaan piksel dan filtering juga memiliki peran besar dalam teknik deteksi tepi, yang merupakan bagian integral dalam analisis gambar. Deteksi tepi bertujuan untuk mengidentifikasi perubahan signifikan dalam intensitas piksel yang menunjukkan adanya batas antara objek yang berbeda. Algoritma seperti operator Sobel dan Canny memanfaatkan ketetanggaan piksel dan teknik konvolusi untuk mendeteksi perbedaan intensitas yang tajam, mengidentifikasi kontur atau tepi objek dalam citra. Filter yang diterapkan dalam proses ini, seperti filter Gaussian, membantu menghaluskan citra sebelum deteksi tepi untuk mengurangi noise yang dapat mengganggu hasil deteksi.

Selain itu, ketetanggaan piksel juga digunakan dalam segmentasi citra, yang merupakan teknik untuk membagi citra menjadi bagian-bagian yang lebih kecil berdasarkan kriteria tertentu. Segmentasi ini bergantung pada hubungan spasial antar piksel untuk mengelompokkan piksel-piksel yang memiliki kesamaan dalam hal warna atau intensitas. Teknik seperti watershed dan k-means clustering mengandalkan ketetanggaan piksel untuk memisahkan citra menjadi objek atau area yang berbeda, yang bermanfaat dalam berbagai aplikasi seperti pengenalan objek, pemantauan lingkungan, dan pengolahan citra medis.

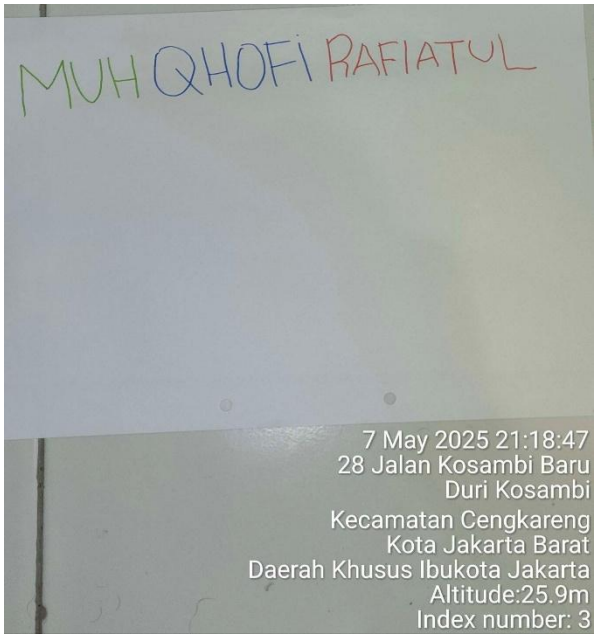
Pengolahan gambar dengan histogram dan filtering memiliki dampak besar dalam peningkatan kualitas citra yang diperoleh dari berbagai sumber, termasuk citra satelit, gambar medis, atau citra hasil pemindaian. Teknik-teknik ini memungkinkan peningkatan kualitas visual yang lebih baik, yang berguna dalam berbagai aplikasi profesional. Misalnya, dalam bidang pengolahan citra medis, filtering digunakan untuk mengurangi noise dalam gambar hasil pemindaian medis, sementara histogram digunakan untuk menganalisis distribusi intensitas dan meningkatkan kontras untuk memastikan deteksi yang lebih akurat terhadap fitur-fitur penting dalam gambar.

BAB III

HASIL

1. Deteksi warna pada citra

Pertama sebelum kita memulai kita membuat gambar dulu yang isinya dengan nama kita



Setelah itu kita masukkan ke dalam file anaconda kita. Jikau sudah masuk kita dapat memulai kodingnya dengan mengimport library yaitu

“import cv2 = pustaka yang sangat populer untuk pemrosesan citra dan visi komputer

import numpy as np = pustaka untuk komputasi numerik di Python, yang memungkinkan kita bekerja dengan array multidimensi, matriks, dan operasi matematika.

import matplotlib.pyplot as plt” = modul yang memungkinkan kita membuat berbagai jenis grafik, termasuk histogram, garis, dan citra.

IMPORT LIBRARY

[126]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Setelah itu kita akan mengubah gambar ke rgb dan membuat gambarnya dengan kode

“rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)” mengubah gambar bgr ke rgb
image = cv2.imread('kertas1.jpg')” = memasukan gambar yang bernana kertas1

MENGUBAH GAMBAR KE RGB

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

MEMBUAT GAMBAR

```
image = cv2.imread('kertas1.jpg')
```

Setelah itu kita akan mendifinisikan warna biru, hijau merah, dengan menulis kode

“red_channel = rgb_image[:, :, 0]

green_channel = rgb_image[:, :, 1]

blue_channel = rgb_image[:, :, 2]” yang berarti Gambar RGB terdiri dari tiga saluran warna: merah (Red), hijau (Green), dan biru (Blue). Gambar ini disusun dalam array tiga dimensi (height, width, 3), di mana dimensi ketiga mewakili tiga saluran warna. red_channel, menggunakan indeks 0, green_chanel, menggunakan indeks 1, blue_channel, menggunakan indeks 2.

Mendefinisikan warna untuk biru, hijau, dan merah

```
red_channel = rgb_image[:, :, 0]
green_channel = rgb_image[:, :, 1]
blue_channel = rgb_image[:, :, 2]
```

Setelah itu kita menulis untuk menampilkan gambar tersebut dengan menulis kode

“fig, axes = plt.subplots(2, 2, figsize=(10, 10))” = Membuat sebuah grid subplots dengan 2 baris dan 2 kolom, dan ukuran figure 10x10 inci. axes adalah array 2x2 yang memegang referensi ke setiap subplot, memungkinkan kita untuk mengakses dan menyesuaikan setiap subplot individual.

“axes[0, 0].imshow(rgb_image)

axes[0, 0].set_title('CITRA KONTRAS')

axes[0, 0].axis('off')” ini berarti Menampilkan gambar RGB asli pada subplot di baris 0, kolom 0. imshow() digunakan untuk menampilkan gambar. set_title() menetapkan judul subplot. axis('off') menyembunyikan sumbu dari subplot pertama agar gambar terlihat lebih jelas.

```
axes[0, 0].imshow(rgb_image)
axes[0, 0].set_title('CITRA KONTRAS')
axes[0, 0].axis('off')
```

“axes[1, 0].imshow(red_channel, cmap='gray')

axes[1, 0].set_title('MERAH')

axes[1, 0].axis('on')” ini berarti imshow(red_channel, cmap='gray') menampilkan saluran merah dalam skala abu-abu (cmap='gray') karena saluran merah memiliki intensitas nilai piksel yang bervariasi dalam kisaran 0 hingga 255. axis('on') menampilkan sumbu pada subplot ini agar kita bisa melihat detail koordinat pada gambar.

```
axes[1, 0].imshow(red_channel, cmap='gray')
axes[1, 0].set_title('MERAH')
axes[1, 0].axis('on')
```

Kita akan melakukan kode yang sama dengan citra hijau dan biru

“axes[1, 1].imshow(green_channel, cmap='gray')

axes[1, 1].set_title('HIJAU')

axes[1, 1].axis('on')

axes[0, 1].imshow(blue_channel, cmap='gray')

axes[0, 1].set_title('BIRU')

axes[0, 1].axis('on')” artinya sama dengan yang diatas menampilkan saluran hijau dan biru dalam skala abu-abu (cmap='gray') karena saluran merah memiliki intensitas nilai piksel yang bervariasi dalam kisaran 0 hingga 255. Dan setelah itu kita menampilkan gambar yang sudah di buat tadi dengan kode “plt.tight_layout()

plt.show()”. Menampilkan gambar dan subplot di layar.

```
axes[1, 1].imshow(green_channel, cmap='gray')
axes[1, 1].set_title('HIJAU')
axes[1, 1].axis('on')

axes[0, 1].imshow(blue_channel, cmap='gray')
axes[0, 1].set_title('BIRU')
axes[0, 1].axis('on')

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

OUTPUT

Setelah membuat gambar diatas kita akan membuat histogramnya, kita dapat menulis kode

fig_hist, axes_hist = plt.subplots(2, 2, figsize=(15, 10)) Membuat grid subplot dengan dua baris dan dua kolom, serta ukuran figure yang lebih besar (15x10 inci) agar histogramnya lebih jelas dan tidak terlalu rapat. axes_hist adalah array 2x2 yang menyimpan referensi ke setiap subplot

```
fig_hist, axes_hist = plt.subplots(2, 2, figsize=(15, 10))
```

HISTOGRAM CITRA ASLI

axes_hist[0, 0].hist(rgb_image.ravel(), bins=256, alpha=0.7)

axes_hist[0, 0].set_title('CITRA KONTRAS') rgb_image.ravel() Mengubah gambar RGB (yang biasanya berupa array 3D) menjadi array 1D dengan menggunakan fungsi ravel(). Ini menggabungkan semua intensitas piksel gambar RGB ke dalam satu dimensi untuk membuat histogram gambar keseluruhan. bins=256: Menentukan jumlah bin yang digunakan untuk menghitung histogram. Karena nilai intensitas piksel berada dalam rentang 0-255 (untuk citra 8-bit). alpha=0.7: Mengatur transparansi histogram. set_title('CITRA KONTRAS'), Memberikan judul pada subplot pertama yang menunjukkan histogram untuk gambar asli.

```
# HISTOGRAM CITRA ASLI
axes_hist[0, 0].hist(rgb_image.ravel(), bins=256, alpha=0.7)
axes_hist[0, 0].set_title('CITRA KONTRAS')
```

HISTOGRAM MERAH

axes_hist[1, 0].hist(rgb_image[:, :, 0].ravel(), bins=256, color='red', alpha=0.7)

axes_hist[1, 0].set_title('MERAH')

HISTOGRAM HIJAU

axes_hist[1, 1].hist(rgb_image[:, :, 1].ravel(), bins=256, color='green', alpha=0.7)

axes_hist[1, 1].set_title('HIJAU')

HISTOGRAM BIRU

axes_hist[0, 1].hist(rgb_image[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7)

axes_hist[0, 1].set_title('BIRU')

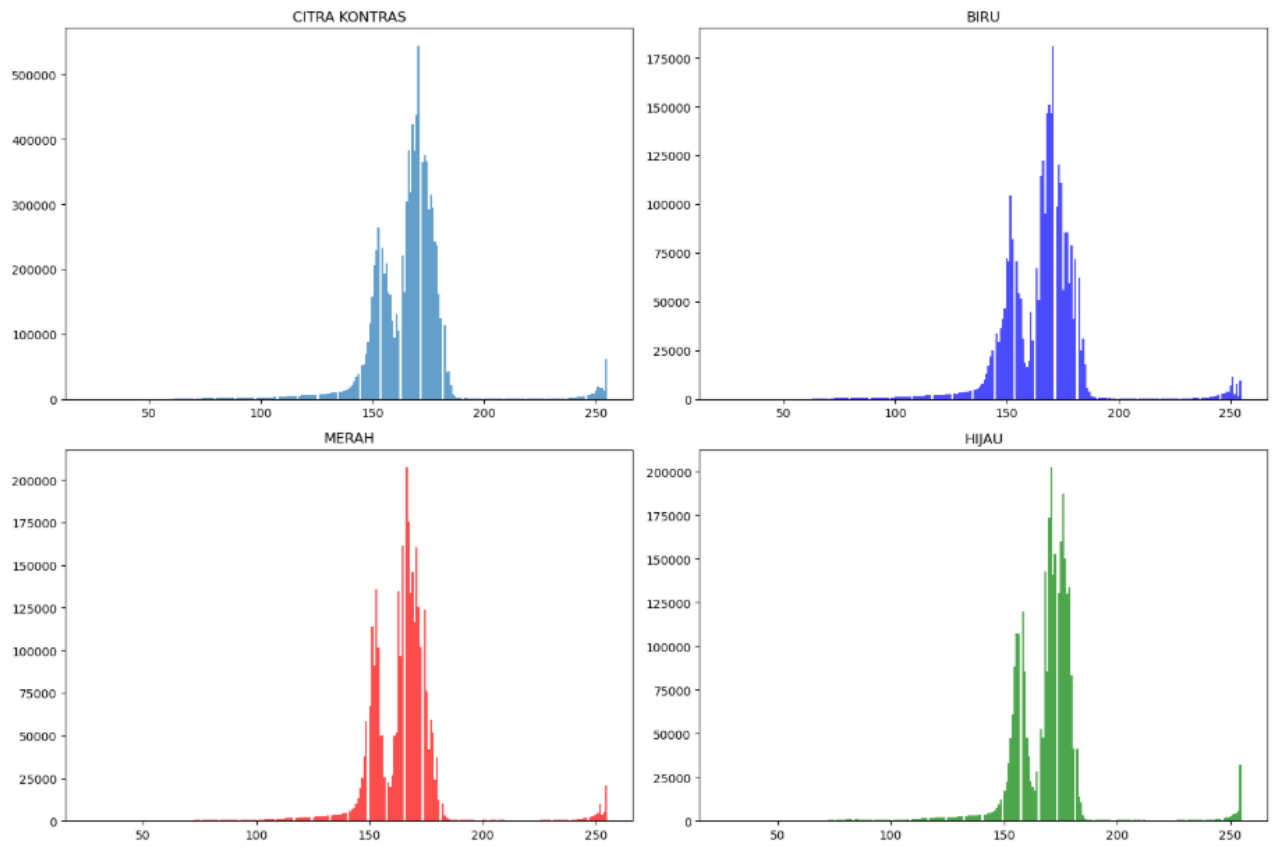
plt.tight_layout()

plt.show()

Ini berarti histogram bewarna merah, hijau, dan biru mempunyai kode yang sama. kode diatas ini bertujuan untuk menampilkan distribusi intensitas piksel pada gambar RGB serta setiap saluran warna individual (merah, hijau, biru).

```
# HISTOGRAM CITRA ASLI
axes_hist[0, 0].hist(rgb_image.ravel(), bins=256, alpha=0.7)
axes_hist[0, 0].set_title('CITRA KONTRAS')

# HISTOGRAM MERAH
axes_hist[1, 0].hist(rgb_image[:, :, 0].ravel(), bins=256, color='red', alpha=0.7)
axes_hist[1, 0].set_title('MERAH')

# HISTOGRAM HIJAU
axes_hist[1, 1].hist(rgb_image[:, :, 1].ravel(), bins=256, color='green', alpha=0.7)
axes_hist[1, 1].set_title('HIJAU')

# HISTOGRAM BIRU
axes_hist[0, 1].hist(rgb_image[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7)
axes_hist[0, 1].set_title('BIRU')
```

OUTPUTNYA



2. Mencari dan urutan ambang batas terkecil sampai dengan terbesar

Pertama kita mengimport library dan membaca gambar tersebut
img = cv2.imread('kertas1.jpg') = membaca gambar yang di simpan dalam file

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) = Mengonversi gambar dari format BGR (format standar OpenCV) menjadi format HSV (Hue, Saturation, Value).

IMPORT LIBRARY

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

MEMBUAT GAMBAR

```
img = cv2.imread('kertas1.jpg')
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Setelah itu kita mengatur rentang warnanya dengan color range. Jika color range biru rentang hue 100-140, dengan saturation dan value dari 50-255. Merah diwakili dengan dua rentang hue karena berada di dua sisi ruang warna HSV, rentang pertama untuk merah adalah 0-10, dan yang kedua

adalah 170-180. Kedua rentang ini akan digabungkan. Hijau rentang hue 40-80, dengan saturation dan value 50-255. color_ranges yang menyimpan batas bawah dan atas untuk setiap warna yang ingin disaring: biru, merah, dan hijau.

MENGATUR WARNA RANGE DARI GAMBAR

```
color_ranges = {
    'blue': {
        'lower': np.array([100, 50, 50]),
        'upper': np.array([140, 255, 255])
    },
    'red': {
        'lower1': np.array([0, 70, 50]),
        'upper1': np.array([10, 255, 255]),
        'lower2': np.array([170, 70, 50]),
        'upper2': np.array([180, 255, 255])
    },
    'green': {
        'lower': np.array([40, 50, 50]),
        'upper': np.array([80, 255, 255])
    }
}
```

Sekarang kita mengatur mask warna. Mask berguna untuk memisahkan warna terntentu dari gambar pertama kita membuat mask_none, ini berarti Membuat masker kosong yang seluruhnya berisi nilai 0. Ini akan menjadi gambar hitam tanpa fitur atau warna yang disaring. Setelah itu buat mask blue yang menggunakan fungsi cv2.inRange() untuk menghasilkan masker yang hanya menampilkan warna biru dari gambar, berdasarkan rentang yang telah didefinisikan dalam color_ranges.

MELAKUKAN PEMISAHAN DENGAN MENGGUNAKAN MASK

```
mask_none = np.zeros(hsv.shape[:2], dtype=np.uint8)

mask_blue = cv2.inRange(hsv, color_ranges['blue']['lower'], color_ranges['blue']['upper'])
```

Setelah itu membuat mask_red1 dan mask_red2 yang masing-masing mengekstrak warna merah dari dua rentang yang berbeda. mask_red menggabungkan kedua mask merah menggunakan operasi logika OR (bitwise_or), sehingga masker merah akan mencakup dua rentang tersebut.

```
mask_red1 = cv2.inRange(hsv, color_ranges['red']['lower1'], color_ranges['red']['upper1'])
mask_red2 = cv2.inRange(hsv, color_ranges['red']['lower2'], color_ranges['red']['upper2'])
mask_red = cv2.bitwise_or(mask_red1, mask_red2)
```

Setelah itu menulis mask_green dan menambahkan mask_red_blue, mask_rgb. mask_green menghasilkan masker untuk warna hijau berdasarkan rentang yang telah ditetapkan. mask_red_blue menggabungkan masker merah dan biru. mask_rgb menggabungkan masker merah, biru, dan hijau dalam satu masker, yang berarti gambar ini akan menampilkan ketiga warna tersebut.

```
mask_green = cv2.inRange(hsv, color_ranges['green']['lower'], color_ranges['green']['upper'])

mask_red_blue = cv2.bitwise_or(mask_red, mask_blue)
mask_rgb = cv2.bitwise_or(mask_red_blue, mask_green)
```

Yang hasilnya begini

MELAKUKAN PEMISAHAN DENGAN MENGGUNAKAN MASK

```
mask_none = np.zeros(hsv.shape[:2], dtype=np.uint8)

mask_blue = cv2.inRange(hsv, color_ranges['blue']['lower'], color_ranges['blue']['upper'])

mask_red1 = cv2.inRange(hsv, color_ranges['red']['lower1'], color_ranges['red']['upper1'])
mask_red2 = cv2.inRange(hsv, color_ranges['red']['lower2'], color_ranges['red']['upper2'])
mask_red = cv2.bitwise_or(mask_red1, mask_red2)

mask_green = cv2.inRange(hsv, color_ranges['green']['lower'], color_ranges['green']['upper'])

mask_red_blue = cv2.bitwise_or(mask_red, mask_blue)
mask_rgb = cv2.bitwise_or(mask_red_blue, mask_green)
```

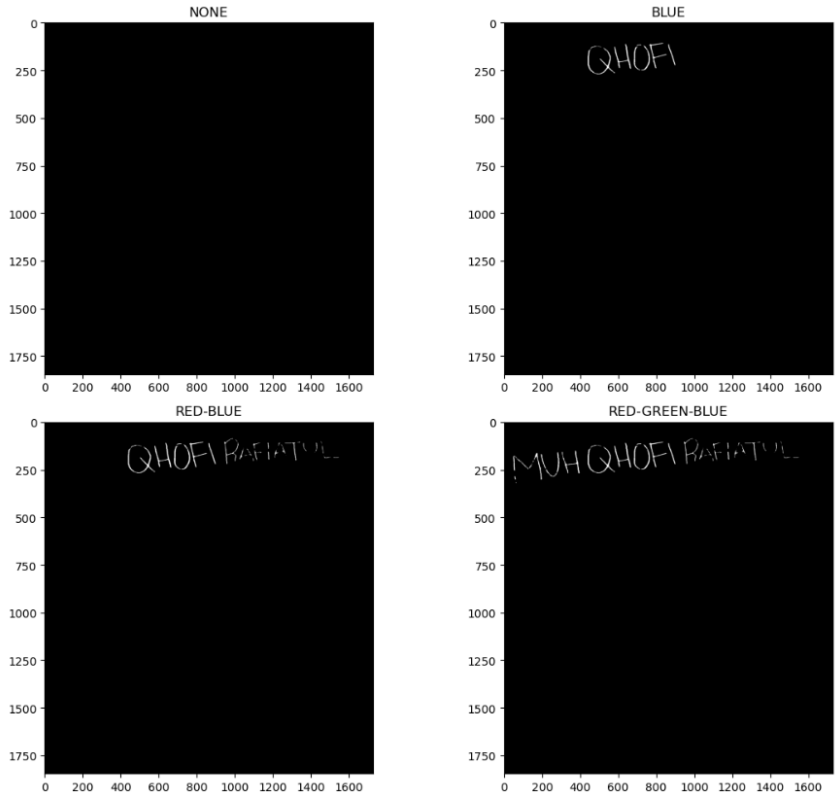
Setelah itu kita menampilkan gambar denga plt.imshow

MENAMPILKAN GAMBAR

```
titles = ['NONE', 'BLUE', 'RED-BLUE', 'RED-GREEN-BLUE']
masks = [mask_none, mask_blue, mask_red_blue, mask_rgb]

plt.figure(figsize=(12, 10))
for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.imshow(masks[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('on')

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

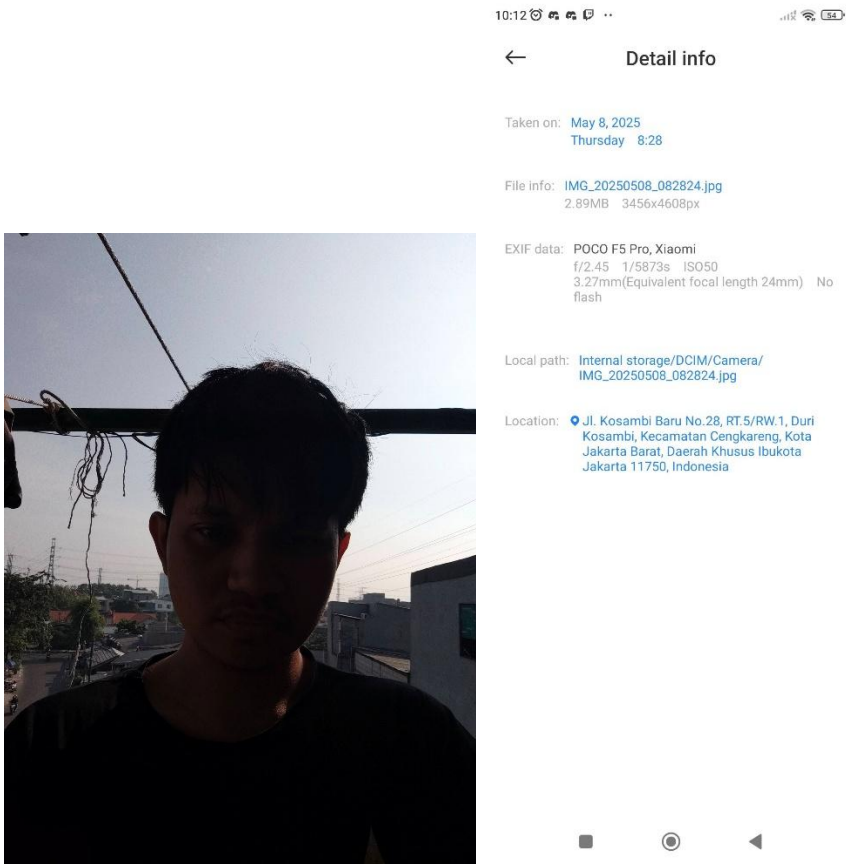
titles dan masks membuat dua list yang berisi judul untuk setiap gambar dan daftar masker yang sudah dibuat. plt.figure(figsize=(12, 10)) menentukan ukuran gambar untuk tampilan subplot. for loop membuat subplots dan menampilkan masker yang telah dibuat di masing-masing subplot (mask_none, mask_blue, mask_red_blue, mask_rgb). Dan plt.tight_layout(), plt.show() mengatur layout subplot agar tidak tumpang tindih dan menampilkan gambar.

Outputnya



3. Memperbaiki Gambar Backlight

kita menggunakan gambar ini



Pertama kita akan mengimport library dan membuat gambar tersebut

IMPORT LIBRARY

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

MEMBUAT GAMBAR

```
image = cv2.imread('selfie1.jpg')
```

Setelah itu kita akan mengubah gambarnya dengan menggunakan greyscale dari BGR ke GRAY

GRAYSCALE

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Setelah itu kita akan mengubah kecerahan dan kontras gambar. Fungsi ini bekerja pada gambar dalam format grayscale (gray_image), dengan dua parameter penting yaitu alpha=1.5, faktor pengali yang akan meningkatkan kontras gambar. beta=50, penambahan nilai ke setiap pixel, yang berfungsi untuk mengubah kecerahan gambar.

DIPERCERAH / BRIGHTENESS

```
brightened_image = cv2.convertScaleAbs(gray_image, alpha=1.5, beta=50)
```

Setelah itu melakukan histogram equalization pada gambar yang sudah dipercepat kecerahannya (brightened_image). Proses ini bertujuan untuk meningkatkan kontras gambar dengan mendistribusikan ulang nilai intensitas pixel agar lebih merata di seluruh spektrum.

DIPERKONTRAS

```
contrast_image = cv2.equalizeHist(brightened_image)
```

Kita menggabungkan dipercerah dan diperkontras ke dalam satu kode dengan gambar gray_image terlebih dahulu diproses dengan meningkatkan kecerahan (beta=70) dan kontras (alpha=3), kemudian histogram equalization diterapkan untuk lebih meningkatkan kontras.

GABUNGAN DIPERCERAH DAN DIPERKONTRAS

```
brightened_and_contrasted_image = cv2.convertScaleAbs(gray_image, alpha=3, beta=70)
brightened_and_contrasted_image = cv2.equalizeHist(brightened_and_contrasted_image)
```

Setelah itu kita Membuat layout dengan 3 baris dan 2 kolom untuk menampilkan gambar-gambar yang telah diproses. Dan memanggil gambar asli, Gambar Grayscale, Gambar yang Dipercerah,

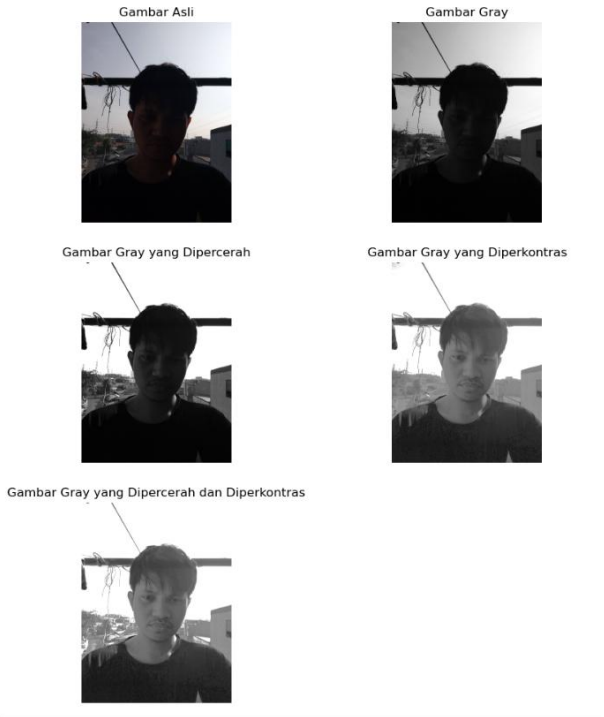
Gambar yang Diperkontras, dan Gambar yang Dipercerah dan Diperkontras. Dan menulis plt.imshow untuk menampilkan hasil

MEMPERLIHATKAN GAMBAR

```
fig, ax = plt.subplots(3, 2, figsize=(10, 12))

ax[0, 0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
ax[0, 0].set_title('Gambar Asli')
ax[0, 0].axis('off')

ax[0, 1].imshow(gray_image, cmap='gray')
ax[0, 1].set_title('Gambar Gray')
ax[0, 1].axis('off')

ax[1, 0].imshow(brightened_image, cmap='gray')
ax[1, 0].set_title('Gambar Gray yang Dipercerah')
ax[1, 0].axis('off')

ax[1, 1].imshow(contrast_image, cmap='gray')
ax[1, 1].set_title('Gambar Gray yang Diperkontras')
ax[1, 1].axis('off')

ax[2, 0].imshow(brightened_and_contrasted_image, cmap='gray')
ax[2, 0].set_title('Gambar Gray yang Dipercerah dan Diperkontras')
ax[2, 0].axis('off')

ax[2, 1].axis('off')

plt.show()
```

Pada subplot ax[2, 1].axis('off') di tulis agar axis tidak nyala pada layout agar rapi karena jika tanpa kode itu akan ada axis nyala sendiri.

Outputnya



BAB IV

PENUTUP

pemrosesan citra digital yang melibatkan ketetanggaan piksel, filtering, dan histogram merupakan elemen-elemen penting dalam analisis dan peningkatan kualitas gambar. Dengan memanfaatkan teknik-teknik seperti filter Gaussian, median, dan rata-rata, serta teknik histogram equalization, kita dapat memodifikasi gambar untuk mencapai kualitas yang lebih baik, mengurangi noise, dan meningkatkan kontras visual. Selain itu, teknik deteksi tepi dan segmentasi citra, yang bergantung pada ketetanggaan piksel, memainkan peran kunci dalam berbagai aplikasi, termasuk pengenalan objek, pemantauan lingkungan, dan pengolahan citra medis. Dengan demikian, pemrosesan citra digital memiliki dampak signifikan dalam meningkatkan kualitas gambar dan memberikan informasi yang lebih berguna dalam berbagai bidang profesional, termasuk kedokteran, industri, dan teknologi penginderaan jarak jauh.

DAFTAR PUSTAKA

Nasution, I. F. (2021). Peningkatan kualitas citra dengan metode filter Gaussian, mean, dan median untuk reduksi noise pada citra ultrasonography. Universitas Islam Negeri Sumatera Utara, Medan, 15(4), 87-95.

Hasibuan, A. M., Rifansyah, M. R., & Rambe, R. P. (2023). Penerapan citra median filter dan histogram equalization pada gambar bangunan tua. Universitas Islam Negeri Sumatera Utara, 23(3), 121-130.

Winarno, G., Irsal, M., Karenina, C. A., Sari, G., & Hidayati, R. N. (2022). Metode histogram equalization untuk peningkatan kualitas citra dengan menggunakan studi phantom lumbosacral. Journal of Radiology and Health Technology, 18(2), 45-53.