

Tugas Analisis Multimedia: Image (Citra Digital)

Mata Kuliah: Sistem & Teknologi Multimedia

Nama: Joshua Fernandes Sectio Purba

NIM: 122140170

SOAL 1

```
In [1]: import matplotlib.pyplot as plt
import cv2
import numpy as np
import mediapipe as mp
```

```
In [2]: # Memuat gambar
image_path = 'Gambar Wajah.jpeg'
image = cv2.imread(image_path)

# Menampilkan gambar asli
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)) # Mengonversi warna BGR ke RGB
plt.title('Gambar Asli')
plt.axis('off')
plt.show()
```

Gambar Asli



```
In [3]: face_crop = cv2.imread('Crop Wajah.jpeg')
background_crop = cv2.imread('Crop Latar Belakang.jpeg')
```

```
# Resize hasil crop menjadi 920x920 piksel
face_crop_resized = cv2.resize(face_crop, (920, 920))
background_crop_resized = cv2.resize(background_crop, (920, 920))

# Menampilkan hasil crop dan resize
plt.figure(figsize=(10, 5))

# Menampilkan gambar wajah hasil crop dan resize
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(face_crop_resized, cv2.COLOR_BGR2RGB))
plt.title('Wajah Setelah Crop dan Resize')
plt.axis('off')

# Menampilkan gambar latar belakang hasil crop dan resize
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(background_crop_resized, cv2.COLOR_BGR2RGB))
plt.title('Latar Belakang Setelah Crop dan Resize')
plt.axis('off')

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

Wajah Setelah Crop dan Resize



Latar Belakang Setelah Crop dan Resize



In [4]:

```
# Mengonversi gambar ke grayscale dan HSV untuk wajah
face_gray = cv2.cvtColor(face_crop_resized, cv2.COLOR_BGR2GRAY)
face_hsv = cv2.cvtColor(face_crop_resized, cv2.COLOR_BGR2HSV)

# Mengonversi gambar ke grayscale dan HSV untuk latar belakang
background_gray = cv2.cvtColor(background_crop_resized, cv2.COLOR_BGR2GRAY)
background_hsv = cv2.cvtColor(background_crop_resized, cv2.COLOR_BGR2HSV)

# Menampilkan 6 gambar (Asli, Grayscale, dan HSV untuk wajah dan latar belakang)
plt.figure(figsize=(15, 10))

# Menampilkan gambar asli (Wajah)
plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(face_crop_resized, cv2.COLOR_BGR2RGB))
plt.title('Wajah (Asli)')
plt.axis('off')

# Menampilkan gambar grayscale (Wajah)
```

```

plt.subplot(2, 3, 2)
plt.imshow(face_gray, cmap='gray')
plt.title('Wajah (Grayscale)')
plt.axis('off')

# Menampilkan gambar HSV (Wajah)
plt.subplot(2, 3, 3)
plt.imshow(cv2.cvtColor(face_hsv, cv2.COLOR_BGR2RGB))
plt.title('Wajah (HSV)')
plt.axis('off')

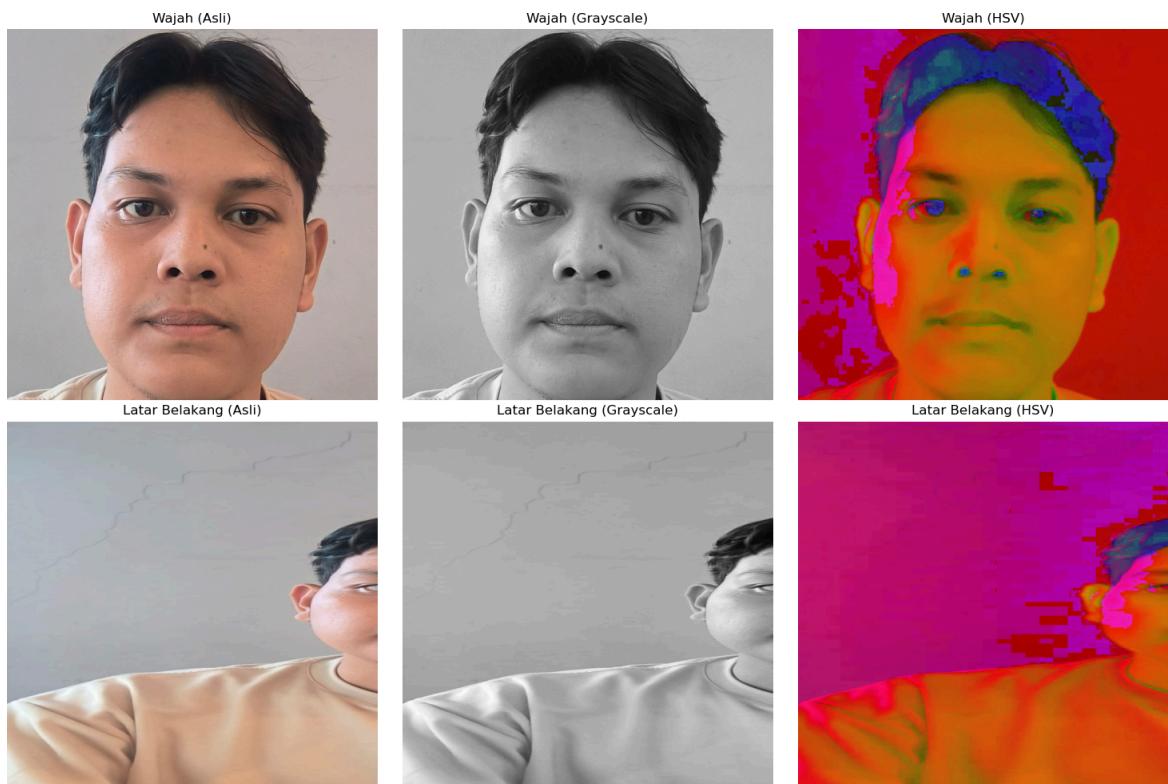
# Menampilkan gambar asli (Latar Belakang)
plt.subplot(2, 3, 4)
plt.imshow(cv2.cvtColor(background_crop_resized, cv2.COLOR_BGR2RGB))
plt.title('Latar Belakang (Asli)')
plt.axis('off')

# Menampilkan gambar grayscale (Latar Belakang)
plt.subplot(2, 3, 5)
plt.imshow(background_gray, cmap='gray')
plt.title('Latar Belakang (Grayscale)')
plt.axis('off')

# Menampilkan gambar HSV (Latar Belakang)
plt.subplot(2, 3, 6)
plt.imshow(cv2.cvtColor(background_hsv, cv2.COLOR_BGR2RGB))
plt.title('Latar Belakang (HSV)')
plt.axis('off')

plt.tight_layout()
plt.show()

```



In [5]:

```

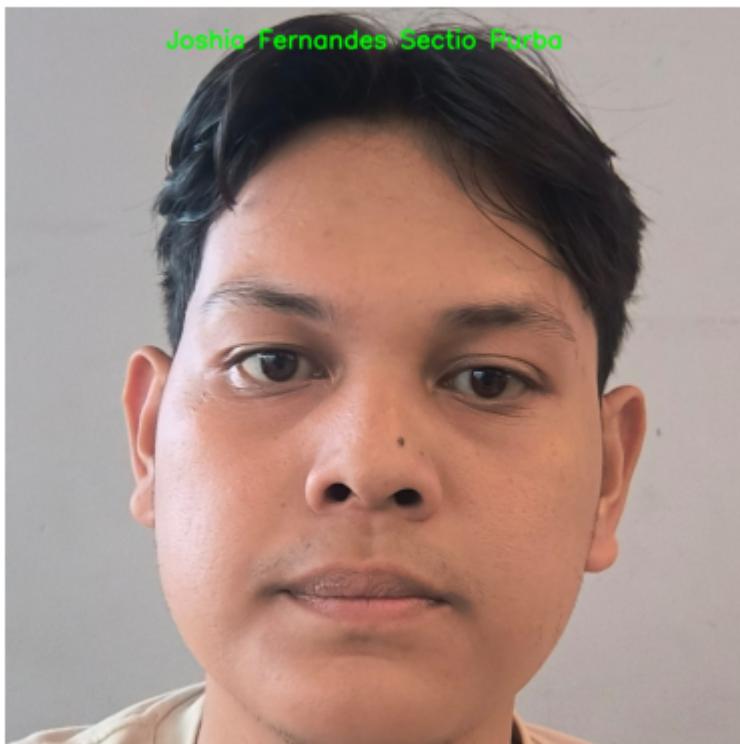
# Menambahkan teks pada gambar wajah nama saya(Joshia)
font = cv2.FONT_HERSHEY_SIMPLEX
text = 'Joshia Fernandes Sectio Purba'
position = (200, 50) # posisi teks yang saya inginkan

```

```
# Menambahkan teks pada gambar hasil crop wajah
face_with_text = face_crop_resized.copy()
cv2.putText(face_with_text, text, position, font, 1, (0, 255, 0), 2, cv2.LINE_AA)

# Menampilkan gambar dengan teks
plt.imshow(cv2.cvtColor(face_with_text, cv2.COLOR_BGR2RGB))
plt.title('Wajah dengan Anotasi Teks')
plt.axis('off')
plt.show()
```

Wajah dengan Anotasi Teks



Penjelasan

1. Efek Cropping:

- **Fungsi:** Memotong bagian gambar menjadi dua area: wajah dan latar belakang.
- **Dampak:**
 - **Wajah:** Gambar wajah saya dipotong menjadi kotak persegi yang berfokus pada wajah saya, membuat objek utama lebih jelas dan terisolasi.
 - **Latar Belakang:** Bagian latar belakang dipotong dengan ukuran persegi panjang, memisahkan objek dari konteks lingkungan dominan latar belakang daripada wajah saya.
 - **Resize:** Setelah dipotong, gambar di-resize menjadi 920x920 piksel, membuat ukuran gambar seragam.

2. Perubahan Warna (Grayscale dan HSV):

- **Grayscale:**

- **Fungsi:** Mengubah gambar menjadi hitam-putih, menghilangkan informasi warna.
- **Dampak:** Memfokuskan pada intensitas cahaya, membuat gambar lebih sederhana dan lebih mudah diproses, tetapi kehilangan informasi warna.
- **HSV:**
 - **Fungsi:** Mengubah gambar ke ruang warna HSV (Hue, Saturation, Value).
 - **Dampak:** Memberikan gambaran lebih jelas tentang warna dan kecerahan gambar, memisahkan informasi warna dan intensitas, menghasilkan tampilan yang berbeda dari RGB atau grayscale.

Kesimpulan:

- **Cropping** memfokuskan gambar pada bagian tertentu, membuat objek utama lebih jelas.
- **Perubahan warna ke grayscale** mengurangi kompleksitas warna, sementara **HSV** memberikan lebih banyak informasi tentang warna dan kecerahan gambar.

SOAL 2

```
In [6]: # Konversi gambar dari BGR ke RGB
face_rgb = cv2.cvtColor(face_crop_resized, cv2.COLOR_BGR2RGB)
background_rgb = cv2.cvtColor(background_crop_resized, cv2.COLOR_BGR2RGB)

# Memisahkan channel warna (R, G, B) untuk wajah
R_face, G_face, B_face = cv2.split(face_rgb)

# Memisahkan channel warna (R, G, B) untuk Latar belakang
R_bg, G_bg, B_bg = cv2.split(background_rgb)

# Manipulasi channel warna:
# 1. Naikkan intensitas channel merah sebanyak 50 poin (maksimum 255)
R_face = cv2.add(R_face, 50)
R_bg = cv2.add(R_bg, 50)
# 2. Turunkan intensitas channel biru sebanyak 30 poin (minimum 0)
B_face = cv2.subtract(B_face, 30)
B_bg = cv2.subtract(B_bg, 30)
```

```
In [7]: # Gabungkan kembali channel yang telah dimodifikasi untuk wajah
modified_face = cv2.merge([R_face, G_face, B_face])

# Gabungkan kembali channel yang telah dimodifikasi untuk Latar belakang
modified_background = cv2.merge([R_bg, G_bg, B_bg])

# Menampilkan gambar asli dan hasil modifikasi untuk wajah dan Latar belakang
plt.figure(figsize=(15, 10))

# Menampilkan gambar asli (Wajah)
plt.subplot(2, 3, 1)
plt.imshow(face_rgb)
plt.title('Wajah (Asli)')
plt.axis('off')

# Menampilkan gambar hasil modifikasi (Wajah)
```

```
plt.subplot(2, 3, 2)
plt.imshow(modified_face)
plt.title('Wajah (Hasil Modifikasi)')
plt.axis('off')

# Menampilkan gambar asli (Latar Belakang)
plt.subplot(2, 3, 4)
plt.imshow(background_rgb)
plt.title('Latar Belakang (Asli)')
plt.axis('off')

# Menampilkan gambar hasil modifikasi (Latar Belakang)
plt.subplot(2, 3, 5)
plt.imshow(modified_background)
plt.title('Latar Belakang (Hasil Modifikasi)')
plt.axis('off')

plt.tight_layout()
plt.show()

# Simpan gambar hasil modifikasi dalam format .png secara otomatis
cv2.imwrite('Gambar Modifikasi Wajah.png', cv2.cvtColor(modified_face, cv2.COLOR_BGR2RGB))
cv2.imwrite('Gambar Modifikasi Latar Belakang.png', cv2.cvtColor(modified_background, cv2.COLOR_BGR2RGB))

# Menampilkan histogram per channel untuk gambar asli dan hasil modifikasi
plt.figure(figsize=(15, 10))

# Histogram untuk gambar asli wajah
plt.subplot(2, 3, 1)
plt.hist(face_rgb[:, :, 0].ravel(), bins=256, color='red', alpha=0.7, label='Red')
plt.hist(face_rgb[:, :, 1].ravel(), bins=256, color='green', alpha=0.7, label='Green')
plt.hist(face_rgb[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7, label='Blue')
plt.title('Histogram Wajah (Asli)')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()

# Histogram untuk gambar hasil modifikasi wajah
plt.subplot(2, 3, 2)
plt.hist(modified_face[:, :, 0].ravel(), bins=256, color='red', alpha=0.7, label='Red')
plt.hist(modified_face[:, :, 1].ravel(), bins=256, color='green', alpha=0.7, label='Green')
plt.hist(modified_face[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7, label='Blue')
plt.title('Histogram Wajah (Hasil Modifikasi)')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()

# Histogram untuk gambar asli latar belakang
plt.subplot(2, 3, 4)
plt.hist(background_rgb[:, :, 0].ravel(), bins=256, color='red', alpha=0.7, label='Red')
plt.hist(background_rgb[:, :, 1].ravel(), bins=256, color='green', alpha=0.7, label='Green')
plt.hist(background_rgb[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7, label='Blue')
plt.title('Histogram Latar Belakang (Asli)')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()

# Histogram untuk gambar hasil modifikasi latar belakang
plt.subplot(2, 3, 5)
plt.hist(modified_background[:, :, 0].ravel(), bins=256, color='red', alpha=0.7, label='Red')
plt.hist(modified_background[:, :, 1].ravel(), bins=256, color='green', alpha=0.7, label='Green')
```

```
plt.hist(modified_background[:, :, 2].ravel(), bins=256, color='blue', alpha=0.7
plt.title('Histogram Latar Belakang (Hasil Modifikasi)')
plt.xlabel('Intensity')
plt.ylabel('Frequency')
plt.legend()

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

Wajah (Asli)



Wajah (Hasil Modifikasi)

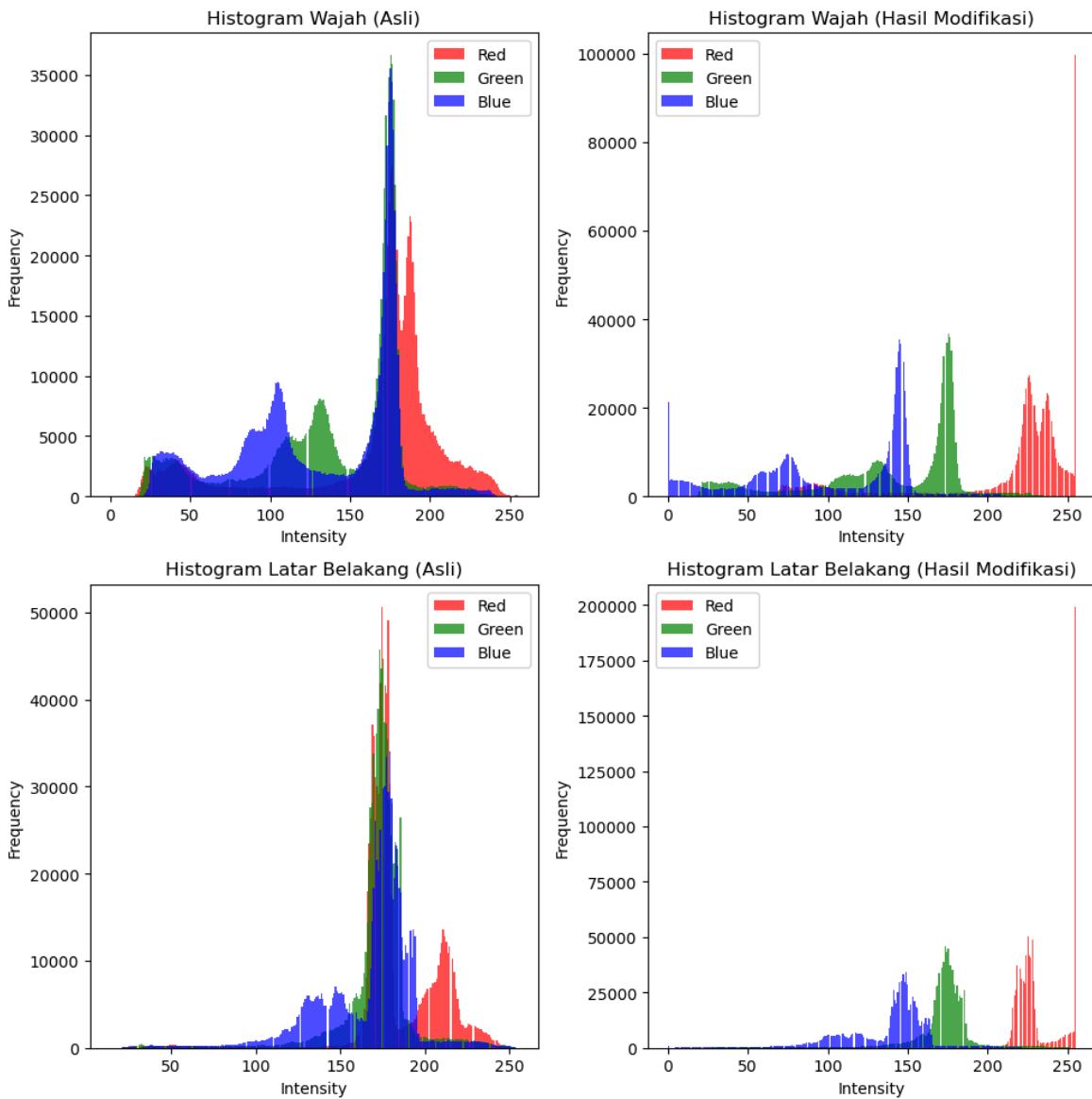


Latar Belakang (Asli)



Latar Belakang (Hasil Modifikasi)





Penjelasan

1. Naikkan Intensitas Channel Merah

- **Fungsi:** Menambah intensitas merah sebanyak 50 poin.
- **Dampak:** Warna gambar jadi lebih merah, terutama pada wajah dan objek di gambar saya.

2. Turunkan Intensitas Channel Biru

- **Fungsi:** Mengurangi intensitas biru sebanyak 30 poin.
- **Dampak:** Warna biru di gambar berkurang, sehingga gambar lebih terlihat kekuningan atau oranye bisa diliat digambar.

3. Pengaruh terhadap Gambar

- Gambar jadi lebih dominan warna merah dan kekuningan karena perubahan pada channel merah dan biru. Gambar jadi lebih hangat dan cerah.

Kesimpulan:

Mengubah intensitas RGB membuat gambar lebih merah dan kekuningan, dengan mengurangi warna biru. Ini memberikan efek warna yang lebih hangat pada gambar.

SOAL 3

```
In [8]: # Memuat gambar objek dengan latar belakang bertekstur
image_path = 'Gambar Bertekstur.jpeg'
image = cv2.imread(image_path)

# Konversi gambar ke grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 1. Terapkan edge detection (Canny)
edges = cv2.Canny(gray_image, threshold1=100, threshold2=150)

# 2. Lakukan thresholding untuk segmentasi objek
_, thresholded = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

# 3. Temukan kontur (bounding box otomatis) dari objek yang tersegmentasi
contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Buat bounding box di sekitar objek yang terdeteksi
image_with_bbox = image.copy()
for contour in contours:
    # Menggambar bounding box dengan koordinat yang ditemukan dari kontur
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(image_with_bbox, (x, y), (x+w, y+h), (0, 255, 0), 2) # Menggaris-bawahi kontur

# 4. Terapkan filter blur (Gaussian)
blurred_image = cv2.GaussianBlur(image, (5, 5), 0)

# 5. Terapkan filter sharpening
kernel_sharpening = np.array([[-1, -1, -1], [-1, 9,-1], [-1, -1, -1]])
sharpened_image = cv2.filter2D(image, -1, kernel_sharpening)

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

# Gambar Asli
plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Gambar Asli')
plt.axis('off')

# Edge Detection (Canny)
plt.subplot(2, 3, 2)
plt.imshow(edges, cmap='gray')
plt.title('Deteksi Tepi (Canny)')
plt.axis('off')

# Gambar dengan Bounding Box
plt.subplot(2, 3, 3)
plt.imshow(cv2.cvtColor(image_with_bbox, cv2.COLOR_BGR2RGB))
plt.title('Bounding Box di Sekitar Objek')
plt.axis('off')
```

```

# Gambar dengan Thresholding
plt.subplot(2, 3, 4)
plt.imshow(thresholded, cmap='gray')
plt.title('Hasil Thresholding')
plt.axis('off')

# Gambar dengan Filter Blur
plt.subplot(2, 3, 5)
plt.imshow(cv2.cvtColor(blurred_image, cv2.COLOR_BGR2RGB))
plt.title('Filter Blur (Gaussian)')
plt.axis('off')

# Gambar dengan Filter Sharpening
plt.subplot(2, 3, 6)
plt.imshow(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2RGB))
plt.title('Filter Sharpening')
plt.axis('off')

plt.tight_layout()
plt.show()

```



Penjelasan

1. Gambar Asli

- Ini adalah gambar awal yang menunjukkan objek (Gambar saya di depan tulisan "Aloha") dengan pencahayaan dan latar belakang yang cukup jelas.

2. Deteksi Tepi (Canny)

- Deteksi tepi (Canny) menunjukkan garis-garis yang jelas pada objek, seperti tepi tulisan "Aloha" dan sekitar wajah saya. Ini membuat batas objek terlihat lebih tajam, namun bagian lain dari gambar menjadi lebih kabur. Fungsi utama Canny adalah untuk menemukan batas objek dalam gambar.

3. Bounding Box di Sekitar Objek

- Kotak hijau ditambahkan di sekitar objek yang terdeteksi (wajah atau objek utama). Ini membantu menandai lokasi objek dalam gambar agar lebih jelas terlihat.

4. Hasil Thresholding

- Setelah thresholding diterapkan, objek utama menjadi lebih jelas, sementara latar belakang yang bertekstur hilang atau menjadi sangat gelap. Teknik ini digunakan untuk memisahkan objek utama dari latar belakang dengan memberi kontras yang jelas.

5. Filter Blur (Gaussian)

- Filter blur membuat gambar menjadi lebih halus dan kabur, mengurangi detail halus pada objek dan latar belakang. Ini menghasilkan efek yang lebih lembut dan mengurangi ketajaman detail gambar.

6. Filter Sharpening

- Filter sharpening meningkatkan ketajaman gambar, membuat tepi objek dan detail lainnya lebih jelas. Hasilnya adalah gambar yang lebih tajam dengan detail yang lebih menonjol.

Kesimpulan:

- **Deteksi Tepi (Canny)** menonjolkan tepi objek, tapi menghilangkan detail latar belakang.
- **Thresholding** memisahkan objek utama dari latar belakang dan membuat objek lebih terlihat jelas.
- **Filter Blur** membuat gambar lebih halus dengan mengurangi detail, sedangkan **Filter Sharpening** membuat gambar lebih tajam dan memperjelas detail.

SOAL 4

```
In [9]: # Inisialisasi MediaPipe Face Mesh
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence=0.5, min_tracking_confidence=0.5)

# Memuat gambar wajah
face_image_path = 'Gambar Wajah Netral.jpeg'
face_image = cv2.imread(face_image_path)
```

```
face_image_rgb = cv2.cvtColor(face_image, cv2.COLOR_BGR2RGB)

# Deteksi Landmark wajah
results = face_mesh.process(face_image_rgb)

# Mengambil dimensi gambar wajah
face_h, face_w = face_image_rgb.shape[:2]

# Menggambar semua Landmark pada gambar wajah
if results.multi_face_landmarks:
    landmarks = results.multi_face_landmarks[0]
    for landmark in landmarks.landmark:
        x = int(landmark.x * face_w)
        y = int(landmark.y * face_h)
        # Menggambar titik pada Landmark
        cv2.circle(face_image, (x, y), 1, (0, 255, 0), -1)

# Menampilkan gambar dengan Landmark
plt.figure(figsize=(10, 10))
plt.imshow(cv2.cvtColor(face_image, cv2.COLOR_BGR2RGB))
plt.title('Titik Landmark Wajah', fontsize=14, fontweight='bold')
plt.axis('off')
plt.show()
```

Titik Landmark Wajah



```
In [10]: # Memuat gambar topi (PNG dengan transparansi)
hat_image_path = 'Topi.jpg'
hat_image = cv2.imread(hat_image_path, cv2.IMREAD_UNCHANGED)

# Menghapus latar belakang putih dari gambar topi dan membuat channel alpha
if hat_image.shape[2] == 3:
    hat_hsv = cv2.cvtColor(hat_image, cv2.COLOR_BGR2HSV)
    lower_white = np.array([0, 0, 200])
    upper_white = np.array([180, 50, 255])
    white_mask = cv2.inRange(hat_hsv, lower_white, upper_white)
    alpha_channel = cv2.bitwise_not(white_mask)
    hat_image_bgra = cv2.cvtColor(hat_image, cv2.COLOR_BGR2BGRA)
```

```

hat_image_bgra[:, :, 3] = alpha_channel
hat_image = hat_image_bgra
print("Latar belakang putih dihapus dari topi - Channel alpha dibuat")
elif hat_image.shape[2] == 4:
    print("Gambar topi sudah memiliki channel alpha")
else:
    print("Format gambar topi tidak dikenali")

# Deteksi Landmark wajah
results = face_mesh.process(face_image_rgb)

# Mengambil dimensi wajah
face_h, face_w = face_image_rgb.shape[:2]

if results.multi_face_landmarks:
    landmarks = results.multi_face_landmarks[0]

    # Mengambil Landmark yang relevan
    center_landmark = landmarks.landmark[2]
    left_eye_landmark = landmarks.landmark[234]
    right_eye_landmark = landmarks.landmark[454]
    top_head_landmark = landmarks.landmark[10] # Puncak kepala

    # Mengonversi koordinat Landmark ke piksel
    center_x = int(center_landmark.x * face_w)
    center_y = int(center_landmark.y * face_h)
    left_eye_x = int(left_eye_landmark.x * face_w)
    right_eye_x = int(right_eye_landmark.x * face_w)
    top_head_y = int(top_head_landmark.y * face_h)

    # Menghitung lebar kepala
    head_width = right_eye_x - left_eye_x

    # Menyesuaikan ukuran topi - skala hingga 2x lebar kepala
    hat_scale = (head_width * 1.2) / hat_image.shape[1]

    # Mengubah ukuran topi
    new_hat_w = int(hat_image.shape[1] * hat_scale)
    new_hat_h = int(hat_image.shape[0] * hat_scale)

    hat_resized = cv2.resize(hat_image[:, :, :3], (new_hat_w, new_hat_h))
    hat_alpha_resized = cv2.resize(hat_image[:, :, 3].astype(float) / 255.0, (ne

    # Posisi topi di atas kepala
    hat_x = int(center_x - new_hat_w // 2)
    hat_y = int(top_head_y - new_hat_h * 1.5) # Posisi lebih tinggi

    # Pastikan posisi topi tetap dalam batas gambar
    hat_x = max(0, min(hat_x, face_w - new_hat_w))
    hat_y = max(0, hat_y)

    # Membuat salinan gambar hasil
    result_image = face_image_rgb.copy()

    # Alpha blending untuk topi dengan transparansi
    hat_end_x = min(hat_x + new_hat_w, face_w)
    hat_end_y = min(hat_y + new_hat_h, face_h)

    hat_crop_w = hat_end_x - hat_x
    hat_crop_h = hat_end_y - hat_y

```

```

for c in range(3):
    result_image[hat_y:hat_end_y, hat_x:hat_end_x, c] = (
        hat_resized[:hat_crop_h, :hat_crop_w, c] * hat_alpha_resized[:hat_cr
        result_image[hat_y:hat_end_y, hat_x:hat_end_x, c] * (1 - hat_alpha_r
    ).astype(np.uint8)

# Menampilkan perbandingan hasil akhir
plt.figure(figsize=(15, 6))

plt.subplot(1, 2, 1)
plt.imshow(face_image_rgb)
plt.title('Wajah Asli', fontsize=14, fontweight='bold')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(result_image)
plt.title('Wajah dengan Filter Topi', fontsize=14, fontweight='bold')
plt.axis('off')

plt.tight_layout()
plt.show()

# Menyimpan hasil
result_bgr = cv2.cvtColor(result_image, cv2.COLOR_RGB2BGR)
cv2.imwrite('Wajah dengan filter Topi.png', result_bgr)
print(f"Posisi Y puncak kepala: {top_head_y}")
print(f"Posisi Y topi: {hat_y}")
print(f"Tinggi topi: {new_hat_h}")

```

Latar belakang putih dihapus dari topi - Channel alpha dibuat

Wajah Asli



Wajah dengan Filter Topi



Posisi Y puncak kepala: 307
 Posisi Y topi: 0
 Tinggi topi: 500

Penjelasan

Menghitung Posisi Overlay:

- **Posisi Overlay** dihitung dengan mendeteksi landmark wajah, seperti titik pusat wajah (landmark 2) dan puncak kepala (landmark 10).
- Menggunakan perhitungan lebar kepala (jarak antara mata kiri dan kanan) untuk menyesuaikan ukuran overlay (topi).
- **Posisi X dan Y** topi ditentukan berdasarkan landmark pusat dan puncak kepala, kemudian disesuaikan agar topi berada di atas kepala.

Tantangan yang Dihadapi:

- Menentukan **koordinat yang tepat** untuk setiap landmark agar overlay tepat pada posisi yang diinginkan karena dari tadi kesulitan menentukan titik nya supaya topi bisa akurat sesuai yang diinginkan.
- Mengatasi **perbedaan ukuran wajah** pada setiap gambar, sehingga ukuran overlay perlu disesuaikan secara dinamis.

Kesimpulan:

Proses ini melibatkan deteksi wajah yang akurat dan penyesuaian posisi overlay, meskipun tantangan seperti perbedaan ukuran wajah dan akurasi landmark perlu diatasi agar hasilnya terlihat natural.

SOAL 5

```
In [11]: # Memuat gambar input
image_path = 'Gambar Minim Pencahayaan.jpeg'
image = cv2.imread(image_path)

# 1. Konversi ke grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# 2. Koreksi Perspektif (Homography)
# Menentukan 4 titik manual dari gambar asli
# Format: [(x1, y1), (x2, y2), (x3, y3), (x4, y4)]
# Sesuaikan titik-titik ini dengan posisi yang sesuai pada gambar Anda
pts1 = np.float32([[100, 200], [400, 200], [100, 400], [400, 400]]) # Titik asli
pts2 = np.float32([[0, 0], [500, 0], [0, 500], [500, 500]]) # Titik tujuan (set

# Matriks homografi
matrix = cv2.getPerspectiveTransform(pts1, pts2)

# Mengaplikasikan transformasi perspektif untuk mendapatkan gambar yang lebih se
warped_image = cv2.warpPerspective(image, matrix, (500, 500))

# 3. Thresholding (Otsu atau Adaptive Thresholding)
# Menggunakan Thresholding Otsu
_, otsu_threshold = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Atau menggunakan Adaptive Thresholding
adaptive_threshold = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)

# 4. Menampilkan Semua Tahap Pemrosesan dalam Grid
# Menyiapkan subplot
```

```
plt.figure(figsize=(12, 8))

# Menampilkan gambar asli
plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Gambar Asli')
plt.axis('off')

# Menampilkan gambar grayscale
plt.subplot(2, 3, 2)
plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale')
plt.axis('off')

# Menampilkan gambar dengan koreksi perspektif
plt.subplot(2, 3, 3)
plt.imshow(cv2.cvtColor(warped_image, cv2.COLOR_BGR2RGB))
plt.title('Koreksi Perspektif')
plt.axis('off')

# Menampilkan gambar dengan Otsu Thresholding
plt.subplot(2, 3, 4)
plt.imshow(otsu_threshold, cmap='gray')
plt.title('Otsu Thresholding')
plt.axis('off')

# Menampilkan gambar dengan Adaptive Thresholding
plt.subplot(2, 3, 5)
plt.imshow(adaptive_threshold, cmap='gray')
plt.title('Adaptive Thresholding')
plt.axis('off')

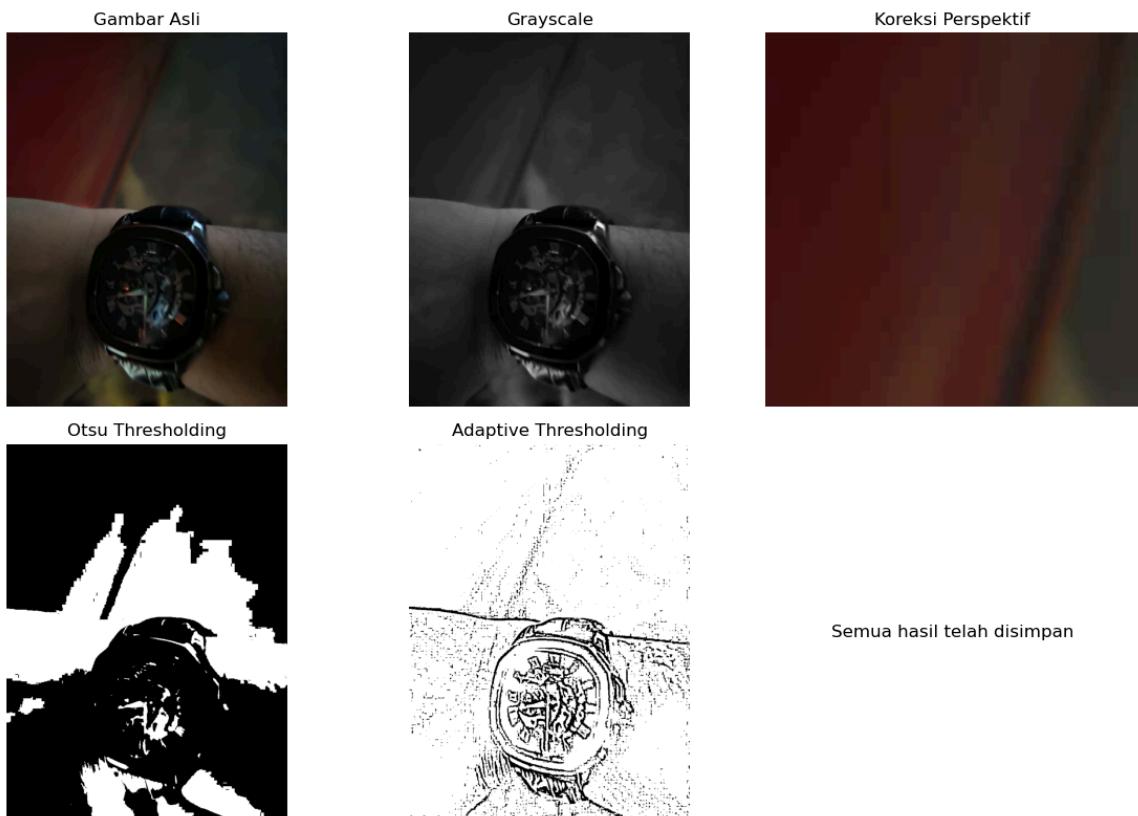
# Menyimpan semua hasil gambar
cv2.imwrite('1_Gambar Asli.png', image)
cv2.imwrite('2_Grayscale.png', gray_image)
cv2.imwrite('3_Koreksi Perspektif.png', warped_image)
cv2.imwrite('4_Otsu Thresholding.png', otsu_threshold)
cv2.imwrite('5_Adaptive Thresholding.png', adaptive_threshold)
print("Semua hasil gambar telah disimpan:")
print("1. 1_Gambar Asli.png")
print("2. 2_Grayscale.png")
print("3. 3_Koreksi Perspektif.png")
print("4. 4_Otsu Thresholding.png")
print("5. 5_Adaptive Thresholding.png")

# Subplot ke-6 untuk menampilkan ringkasan
plt.subplot(2, 3, 6)
plt.text(0.5, 0.5, 'Semua hasil telah disimpan',
         horizontalalignment='center', verticalalignment='center',
         fontsize=12, transform=plt.gca().transAxes)
plt.axis('off')

# Menampilkan plot
plt.tight_layout()
plt.show()
```

Semua hasil gambar telah disimpan:

1. 1_Gambar Asli.png
2. 2_Grayscale.png
3. 3_Koreksi Perspektif.png
4. 4_Otsu Thresholding.png
5. 5_Adaptive Thresholding.png



Penjelasan

1. Konversi ke Grayscale:

- **Fungsi:** Mengubah gambar jadi hitam-putih.
- **Peningkatan:** Membuat citra lebih sederhana dan memudahkan proses selanjutnya.

2. Koreksi Perspektif:

- **Fungsi:** Memperbaiki distorsi pada gambar yang diambil dari sudut miring.
- **Peningkatan:** Membuat objek terlihat lebih sejajar dan rapi.

3. Thresholding (Otsu atau Adaptive):

- **Fungsi:** Memisahkan objek dari latar belakang dengan batas kecerahan.
- **Peningkatan:** Memperjelas objek dengan meningkatkan kontras.

Kesimpulan:

Proses ini membuat citra lebih jelas dengan menghilangkan gangguan warna, memperbaiki distorsi, dan menonjolkan objek utama.

Link AI : <https://chatgpt.com/share/e/6915fdfd-838c-800d-a38f-20ae49c3fdf7>

Juga beberapa bantuan dari AI tools pada VS Code