

Link Github: [Github](#)

Tugas Analisis Multimedia: Image (Citra Digital)

Mata Kuliah: Sistem & Teknologi Multimedia

Nama: A. Edwin Krisandika Putra

NIM: 122140003

Deskripsi Tugas

Tugas ini bertujuan untuk memahami **representasi dasar data citra digital (image)** melalui praktik langsung memuat data, visualisasi komponen warna, serta melakukan analisis spasial sederhana menggunakan berbagai teknik dasar pengolahan citra.

Anda akan bekerja dengan satu atau beberapa gambar (foto diri, objek, atau lingkungan sekitar) untuk:

- Mengamati struktur data piksel dan channel warna (RGB, Grayscale, HSV, dsb.)
- Menganalisis perbedaan hasil visualisasi antar representasi warna
- Melakukan eksplorasi sederhana terhadap transformasi citra (cropping, filtering, edge detection, dll.)
- Menyimpulkan pengaruh setiap tahap pemrosesan terhadap persepsi visual

Fokus tugas ini adalah pada **pemahaman konsep representasi spasial citra digital** dan **interpretasi hasil visualisasi**, **bukan** pada manipulasi kompleks atau penerapan model pembelajaran mesin.

Soal 1 — Cropping dan Konversi Warna

- Ambil sebuah gambar diri Anda (*selfie*) menggunakan kamera atau smartphone.
- Lakukan **cropping secara manual** untuk menghasilkan dua potongan:
 - Cropping **kotak persegi pada area wajah**.
 - Cropping **persegi panjang pada area latar belakang**.
- Resize hasil crop menjadi **920×920 piksel**.
- Konversi gambar menjadi **grayscale** dan **HSV**, lalu tampilkan ketiganya berdampingan.
- Tambahkan **anotasi teks** berisi nama Anda di atas kepala pada gambar hasil crop.
 - Gaya teks (font, warna, posisi, ukuran, ketebalan) **dibebaskan**.
- Jelaskan efek **cropping** dan **perubahan warna** menggunakan **Markdown**.

In [107...]

```
import os
import cv2
import matplotlib.pyplot as plt

selfie_path = os.path.join("assets_ws4", "selfie.jpg")
print(os.path.exists(selfie_path))

image = cv2.imread(selfie_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print(image.shape)
plt.imshow(image)
plt.axis("off")
plt.show()

output_dir = "results_ws4"
os.makedirs(output_dir, exist_ok=True)

#Crop bagian wajah dan background
crop_wajah = image[900:1700, 500:1300]
crop_bg = image[0:1000, 0:300]

#Resize gambar menjadi 920x920
wajah_resized = cv2.resize(crop_wajah, (920, 920))
cv2.imwrite(os.path.join(output_dir, "wajah_resized.jpg"), cv2.cvtColor(wajah_resized, cv2.COLOR_RGB2GRAY))
bg_resized = cv2.resize(crop_bg, (920, 920))
cv2.imwrite(os.path.join(output_dir, "bg_resized.jpg"), cv2.cvtColor(bg_resized, cv2.COLOR_RGB2GRAY))

#Konversi gambar menjadi grayscale dan HSV
wajah_gray = cv2.cvtColor(wajah_resized, cv2.COLOR_RGB2GRAY)
cv2.imwrite(os.path.join(output_dir, "wajah_gray.jpg"), wajah_gray)
wajah_hsv = cv2.cvtColor(wajah_resized, cv2.COLOR_RGB2HSV)
cv2.imwrite(os.path.join(output_dir, "wajah_hsv.jpg"), cv2.cvtColor(wajah_hsv, cv2.COLOR_RGB2GRAY))
bg_gray = cv2.cvtColor(bg_resized, cv2.COLOR_RGB2GRAY)
cv2.imwrite(os.path.join(output_dir, "bg_gray.jpg"), bg_gray)
bg_hsv = cv2.cvtColor(bg_resized, cv2.COLOR_RGB2HSV)
cv2.imwrite(os.path.join(output_dir, "bg_hsv.jpg"), cv2.cvtColor(bg_hsv, cv2.COLOR_RGB2GRAY))

#Tampilkan hasil resize, grayscale, dan HSV
```

```
#Wajah
anotasi = wajah_resized.copy()
cv2.putText(anotasi, "Edwin", (300, 100), cv2.FONT_HERSHEY_SIMPLEX, 3, (255, 0, 0))

plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(anotasi)
plt.title("Crop")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(wajah_gray, cmap="gray")
plt.title("Grayscale")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(wajah_hsv)
plt.title("HSV")
plt.axis("off")

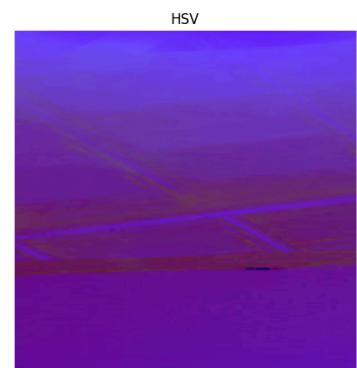
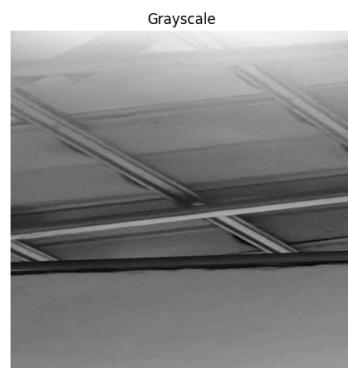
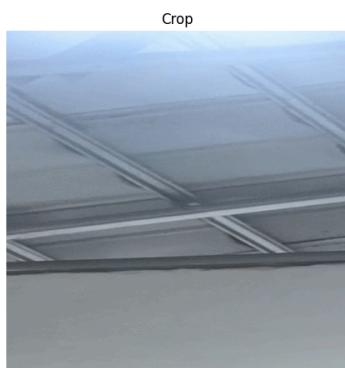
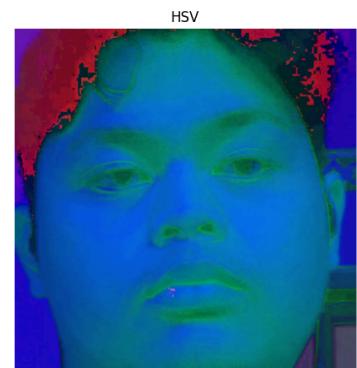
#background
plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(bg_resized)
plt.title("Crop")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(bg_gray, cmap="gray")
plt.title("Grayscale")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(bg_hsv)
plt.title("HSV")
plt.axis("off")

plt.show()
```

True
(2304, 1728, 3)



Penjelasan Proses

Crop dilakukan dua kali, crop pertama yaitu crop area wajah dengan bentuk kotak persegi dan crop kedua yaitu crop area latar belakang dengan bentuk persegi panjang. Cropping dilakukan secara manual dengan menuliskan pixel yang kira kira pas untuk setiap gambar hasilnya. Setelah gambar dicrop dan menghasilkan 2 gambar baru, kemudian gambar diresize menjadi 920x920 pixel (jadi persegi dua duanya). Lalu, kedua gambar barusan (yang sudah jadi 920) dikonversi warnanya, hasil konversian disimpan di folder results. Pada konversi grayscale, gambar berubah menjadi hitam putih dan pada konversi HSV, informasi gambar diubah yang awalnya dari intensitas warna merah, hijau,

biru (RGB) menjadi 3 komponen, Hue yaitu jenis warna, Saturation yaitu seberapa kuat warnanya, dan Value yaitu intensitas cahaya. Setelah itu, gambar hasil crop (yang masih RGB) diberi nama di atas kepala dengan fungsi cv2 (putText). Lalu gambar ditampilkan ke output.

Soal 2 — Manipulasi Channel Warna RGB

- Gunakan gambar hasil crop dari Soal 1.
- Konversikan gambar ke ruang warna **RGB**.
- Lakukan manipulasi channel warna dengan cara:
 - **Naikkan intensitas channel merah sebanyak 50 poin** (maksimum 255).
 - **Turunkan intensitas channel biru sebanyak 30 poin** (minimum 0).
- Teknik atau cara menaikkan/menurunkan intensitas **dibebaskan**, asalkan logis dan hasilnya terlimustache.
- Gabungkan kembali channel warna dan **simpan gambar hasil modifikasi dalam format .png**.
- **Tampilkan histogram per channel (R, G, B)** untuk gambar asli dan hasil modifikasi menggunakan `matplotlib.pyplot.hist`.
- Jelaskan dampak perubahan RGB pada warna gambar dalam sel **Markdown**.

In [109...]

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os

# Konversi ke RGB
crop_wajah_rgb = crop_wajah[:, :, :3].astype(np.int16)
crop_bg_rgb = crop_bg[:, :, :3].astype(np.int16)

# Manipulasi channel wajah
R_w = np.clip(crop_wajah_rgb[:, :, 0] + 50, 0, 255)
G_w = crop_wajah_rgb[:, :, 1]
B_w = np.clip(crop_wajah_rgb[:, :, 2] - 30, 0, 255)
crop_wajah_modified = np.dstack([R_w, G_w, B_w]).astype(np.uint8)

# Manipulasi channel background
R_b = np.clip(crop_bg_rgb[:, :, 0] + 50, 0, 255)
G_b = crop_bg_rgb[:, :, 1]
B_b = np.clip(crop_bg_rgb[:, :, 2] - 30, 0, 255)
crop_bg_modified = np.dstack([R_b, G_b, B_b]).astype(np.uint8)

# Simpan gambar hasil modifikasi
os.makedirs('results_ws4', exist_ok=True)
Image.fromarray(crop_wajah_modified).save('results_ws4/crop_wajah_modified.png')
Image.fromarray(crop_bg_modified).save('results_ws4/crop_bg_modified.png')

# Plot histogram sesuai instruksi: menggunakan plt.hist()
```

```
plt.figure(figsize=(18, 10))

# Wajah asli
plt.subplot(2, 4, 1)
plt.title("Wajah Asli")
plt.imshow(crop_wajah_rgb.astype(np.uint8))
plt.axis("off")

plt.subplot(2, 4, 2)
plt.title("Hist Wajah Asli (RGB)")
plt.hist(crop_wajah_rgb[:, :, 0].flatten(), bins=256, color='r', alpha=0.5)
plt.hist(crop_wajah_rgb[:, :, 1].flatten(), bins=256, color='g', alpha=0.5)
plt.hist(crop_wajah_rgb[:, :, 2].flatten(), bins=256, color='b', alpha=0.5)

# Wajah modifikasi
plt.subplot(2, 4, 3)
plt.title("Wajah Modifikasi")
plt.imshow(crop_wajah_modified)
plt.axis("off")

plt.subplot(2, 4, 4)
plt.title("Hist Wajah Modifikasi (RGB)")
plt.hist(crop_wajah_modified[:, :, 0].flatten(), bins=256, color='r', alpha=0.5)
plt.hist(crop_wajah_modified[:, :, 1].flatten(), bins=256, color='g', alpha=0.5)
plt.hist(crop_wajah_modified[:, :, 2].flatten(), bins=256, color='b', alpha=0.5)

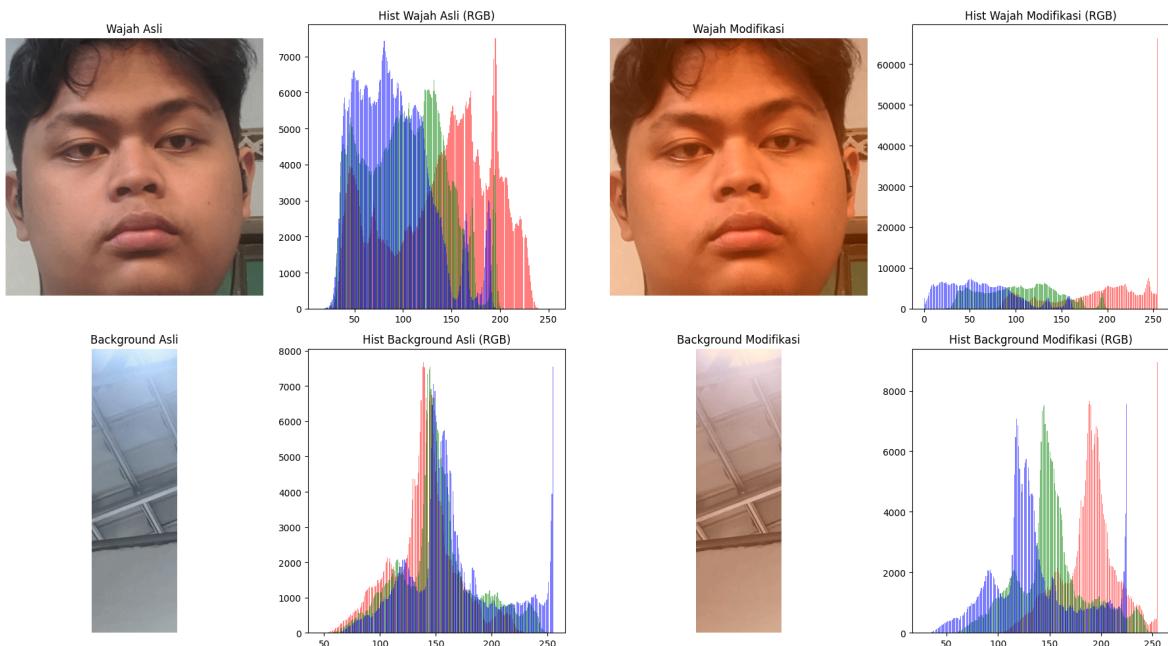
# Background asli
plt.subplot(2, 4, 5)
plt.title("Background Asli")
plt.imshow(crop_bg_rgb.astype(np.uint8))
plt.axis("off")

plt.subplot(2, 4, 6)
plt.title("Hist Background Asli (RGB)")
plt.hist(crop_bg_rgb[:, :, 0].flatten(), bins=256, color='r', alpha=0.5)
plt.hist(crop_bg_rgb[:, :, 1].flatten(), bins=256, color='g', alpha=0.5)
plt.hist(crop_bg_rgb[:, :, 2].flatten(), bins=256, color='b', alpha=0.5)

# Background modifikasi
plt.subplot(2, 4, 7)
plt.title("Background Modifikasi")
plt.imshow(crop_bg_modified)
plt.axis("off")

plt.subplot(2, 4, 8)
plt.title("Hist Background Modifikasi (RGB)")
plt.hist(crop_bg_modified[:, :, 0].flatten(), bins=256, color='r', alpha=0.5)
plt.hist(crop_bg_modified[:, :, 1].flatten(), bins=256, color='g', alpha=0.5)
plt.hist(crop_bg_modified[:, :, 2].flatten(), bins=256, color='b', alpha=0.5)

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



Pada kode, dilakukan dua manipulasi pada channel warna yaitu channel merah dinaikkan 50 poin (dengan maksimum 255) dan channel biru diturunkan -30 poin (dengan minimum 0). Dengan meningkatkan channel warna merah dan menurunkan channel biru ini, gambar terlihat menjadi lebih kemerah orenan, jadi terlihat lebih warm. Hal ini terjadi karena warnanya bertolak belakang dengan tone cool, biasanya biru dan biru di sini diturunkan. Pada histogram, terlihat kalau pixel biru bergeser ke kiri dan pixel merah bergeser ke kanan. Hal ini terjadi karena seluruh intensitas dikurang dan ditambah (geser ke kiri karena berkurang dan geser ke kanan karena bertambah). Untuk histogram atas kanan terlihat kecil, tetapi nilainya sama saja (dapat dilihat nilainya di sumbunya, rentangnya jadi lebih luas).

Soal 3 — Deteksi Tepi dan Filter Citra

- Ambil gambar **objek dengan background bertekstur** (misalnya kain bermotif, jerami, atau batu).
- Terapkan **edge detection (Canny)** dan tampilkan hasilnya.
- Lakukan **thresholding dengan nilai ambang tertentu** (bebas Anda tentukan) agar hanya objek utama yang tersisa.
- Buat **bounding box** di sekitar objek hasil segmentasi (boleh manual atau otomatis).
- Terapkan **filter blur** dan **filter sharpening**, lalu **bandingkan hasil keduanya**.
- Jelaskan bagaimana setiap filter memengaruhi detail gambar dalam format **Markdown**.

In [102...]

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

# Load gambar
ds4_path = os.path.join("assets_ws4", "ds4.jpg")
ds4 = cv2.cvtColor(cv2.imread(ds4_path), cv2.COLOR_BGR2RGB)
ds4_gray = cv2.cvtColor(ds4, cv2.COLOR_RGB2GRAY)
```

```
# Edge detection (Canny)
edges = cv2.Canny(ds4_gray, 150, 190)

# Thresholding
_, thresh = cv2.threshold(ds4_gray, 120, 255, cv2.THRESH_BINARY_INV)

# Bounding box dari threshold
contours_t, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
bbox_thresh = ds4.copy()
for c in contours_t:
    if cv2.contourArea(c) > 3000:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(bbox_thresh, (x, y), (x + w, y + h), (255, 0, 0), 3)

# Gaussian Blur
blurred = cv2.GaussianBlur(ds4, (15, 15), 0)
blurred_gray = cv2.cvtColor(blurred, cv2.COLOR_RGB2GRAY)

# Bounding box dari blur (pakai threshold yang sama)
_, thresh.blur = cv2.threshold(blurred_gray, 120, 255, cv2.THRESH_BINARY_INV)
contours_b, _ = cv2.findContours(thresh.blur, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
bbox.blur = blurred.copy()
for c in contours_b:
    if cv2.contourArea(c) > 1600:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(bbox.blur, (x, y), (x + w, y + h), (255, 0, 0), 3)

# Sharpening
sharpen_kernel = np.array([[0, -1, 0],
                           [-1, 5, -1],
                           [0, -1, 0]])
sharpened = cv2.filter2D(ds4, -1, sharpen_kernel)
sharpened_gray = cv2.cvtColor(sharpened, cv2.COLOR_RGB2GRAY)

# Bounding box dari sharpening
_, thresh.sharp = cv2.threshold(sharpened_gray, 120, 255, cv2.THRESH_BINARY_INV)
contours_s, _ = cv2.findContours(thresh.sharp, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
bbox.sharp = sharpened.copy()
for c in contours_s:
    if cv2.contourArea(c) > 5000:
        x, y, w, h = cv2.boundingRect(c)
        cv2.rectangle(bbox.sharp, (x, y), (x + w, y + h), (255, 0, 0), 3)

# Plot 8 gambar
plt.figure(figsize=(15, 20))

plt.subplot(4, 2, 1)
plt.imshow(ds4)
plt.title("1. Gambar Asli")
plt.axis("off")

plt.subplot(4, 2, 2)
plt.imshow(edges, cmap='gray')
plt.title("2. Canny Edge Detection")
plt.axis("off")

plt.subplot(4, 2, 3)
plt.imshow(thresh, cmap='gray')
plt.title("3. Thresholding")
plt.axis("off")
```

```
plt.subplot(4, 2, 4)
plt.imshow(bbox_thresh)
plt.title("4. Bounding Box (Threshold)")
plt.axis("off")

plt.subplot(4, 2, 5)
plt.imshow(blurred)
plt.title("5. Gaussian Blur")
plt.axis("off")

plt.subplot(4, 2, 6)
plt.imshow(bbox.blur)
plt.title("6. Bounding Box (Blur)")
plt.axis("off")

plt.subplot(4, 2, 7)
plt.imshow(sharpened)
plt.title("7. Sharpening")
plt.axis("off")

plt.subplot(4, 2, 8)
plt.imshow(bbox_sharp)
plt.title("8. Bounding Box (Sharpened)")
plt.axis("off")

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



Edge detection canny: Filter ini memakai dua parameter yaitu lower threshold dan upper threshold untuk menentukan detail yang masuk, pixel yang di bawah lower threshold otomatis tidak akan terbaca sebagai tepi, pixel yang di atas upper threshold sudah pasti terbaca sebagai tepi. Jika parameter terlalu tinggi, detail banyak yang hilang dan jika terlalu rendah, noise ikut terbaca sebagai tepi. Thresholding: contohnya
`(cv2.threshold(ds4_gray, 120, 255, cv2.THRESH_BINARY_INV))`, menggunakan nilai

parameter 120 untuk pisahkan pixel hitam/putih, nilai 255 untuk nilai maksimalnya. mode THRESH_BINARY_INV membalik hasilnya, jadi pixel yang lebih kecil dari parameter 120 akan menjadi putih dan sebaliknya. Hasilnya akan jadi gambar yang menggambarkan tepi dengan perpaduan pixel hitam dan putih. Gaussian blur: (cv2.GaussianBlur(ds4, (15, 15), 0)) memakai kernel 15x15 jadi gambar terlihat lebih pudar (semakin besar kernel semakin pudar). Sharpening: menggunakan kernel konvolusi, kalau di code [[0,-1,0], [-1,5,-1],[0,-1,0]], artinya memperkuat nilai pixel pusat (5) tapi juga mengurangi nilai pixel di sekitarnya (-1), jadinya tepi kelihatan lebih tajam. BoundingBox: mendeteksi semua kontur, pada contoh ini cv2.contourArea(c) > 3000, parameter akan dipakai untuk menyaring kontur kecil. Saat pakai parameter kecil, akan banyak kontur yang terdeteksi, jadi banyak yang dikotakin, cara saya adalah menaikkan parameter secara perlahan sehingga boundingbox tersisa pada objek yang diinginkan saja.

Soal 4 — Deteksi Wajah dan Filter Digital Kreatif

- Ambil gambar diri Anda dengan ekspresi wajah **netral**.
- Lakukan **deteksi wajah dan landmark** menggunakan salah satu dari:
 - **MediaPipe**, atau
 - **Dlib**, atau
 - **OpenCV**.
- Buat **overlay filter digital kreatif** karya Anda sendiri, misalnya:
 - topi, kumis, masker, helm, aksesoris, atau bentuk unik lainnya.
 - Filter boleh dibuat dari **gambar eksternal (PNG)** atau digambar langsung (misal bentuk lingkaran, garis, poligon, dll).
- Pastikan posisi overlay menyesuaikan **landmark wajah** dengan logis.
- **Gunakan alpha blending sebagai saran** agar hasil tampak lebih natural.
- Tampilkan perbandingan antara **gambar asli** dan **hasil dengan filter**.
- Jelaskan bagaimana Anda menghitung posisi overlay dan tantangan yang dihadapi selama implementasi (gunakan **Markdown**).

In [49]:

```
import cv2
import mediapipe as mp
import numpy as np
import matplotlib.pyplot as plt
import os

# Load image
image_path = "assets_ws4/selfie.jpg"
mustache_path = "assets_ws4/mustache.png"

image_bgr = cv2.imread(image_path)
image = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
h, w, _ = image.shape
original = image.copy()
```

```

# Run MediaPipe Face Mesh
mp_mesh = mp.solutions.face_mesh
with mp_mesh.FaceMesh(static_image_mode=True, max_num_faces=1) as mesh:
    result = mesh.process(image)

if not result.multi_face_landmarks:
    raise ValueError("Wajah tidak terdeteksi.")

# Convert Landmarks to pixel coordinates
face = result.multi_face_landmarks[0].landmark
landmarks = np.array([(int(p.x * w), int(p.y * h)) for p in face])

# Draw Landmark visualization
landmark_img = original.copy()
for (x, y) in landmarks:
    cv2.circle(landmark_img, (x, y), 3, (0, 255, 0), -1)

# Select landmarks for mustache placement
philtrum = landmarks[13]
mouth_left = landmarks[61]
mouth_right = landmarks[291]
mouth_width = np.linalg.norm(mouth_left - mouth_right)

# Load mustache PNG
mustache = cv2.imread(mustache_path, cv2.IMREAD_UNCHANGED)
if mustache is None:
    raise FileNotFoundError("PNG kumis tidak ditemukan.")

# Create alpha if missing
if mustache.shape[2] == 3:
    gray = cv2.cvtColor(mustache, cv2.COLOR_BGR2GRAY)
    _, alpha = cv2.threshold(gray, 240, 255, cv2.THRESH_BINARY_INV)
    mustache = np.dstack([mustache, alpha])

# Resize mustache according to mouth width
scale_factor = 1.5
mustache_width = int(mouth_width * scale_factor)
aspect_ratio = mustache.shape[0] / mustache.shape[1]
mustache_height = int(mustache_width * aspect_ratio)
mustache = cv2.resize(mustache, (mustache_width, mustache_height))

# Compute rotation angle based on mouth orientation
dx = mouth_right[0] - mouth_left[0]
dy = mouth_right[1] - mouth_left[1]
angle = -np.degrees(np.arctan2(dy, dx))

# Rotate mustache (RGBA)
M = cv2.getRotationMatrix2D((mustache_width//2, mustache_height//2), angle, 1.0)
rot_rgb = cv2.warpAffine(mustache[:, :, :3], M, (mustache_width, mustache_height))
rot_alpha = cv2.warpAffine(mustache[:, :, 3], M, (mustache_width, mustache_height))
rotated_mustache = np.dstack([rot_rgb, rot_alpha])

# Placement under philtrum
x_offset = philtrum[0] - mustache_width // 2
y_offset = philtrum[1] - 112

# Prepare overlay area
result_img = original.copy()
x1, x2 = max(0, x_offset), min(w, x_offset + mustache_width)
y1, y2 = max(0, y_offset), min(h, y_offset + mustache_height)

```

```

mx1 = x1 - x_offset
mx2 = mx1 + (x2 - x1)
my1 = y1 - y_offset
my2 = my1 + (y2 - y1)

mustache_crop = rotated_mustache[my1:my2, mx1:mx2]
roi = result_img[y1:y2, x1:x2]

# Alpha blending
alpha = mustache_crop[:, :, 3] / 255.0
for c in range(3):
    roi[:, :, c] = roi[:, :, c] * (1 - alpha) + mustache_crop[:, :, c] * alpha

result_img[y1:y2, x1:x2] = roi

# Show outputs
plt.figure(figsize=(18, 6))

plt.subplot(1, 3, 1)
plt.imshow(original)
plt.title("Gambar Asli")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(landmark_img)
plt.title("Gambar dengan Landmark")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(result_img)
plt.title("Gambar dengan Kumis + Rotasi")
plt.axis("off")

plt.tight_layout()
plt.show()

#simpan semua hasil gambar ke results_ws4
os.makedirs('results_ws4', exist_ok=True)
cv2.imwrite(os.path.join('results_ws4', 'landmark_img.jpg'), cv2.cvtColor(landmark_img, cv2.COLOR_BGR2RGB))
cv2.imwrite(os.path.join('results_ws4', 'filter_img.jpg'), cv2.cvtColor(result_img, cv2.COLOR_BGR2RGB))

```



Out[49]: True

Wajah pada gambar dideteksi pakai MediaPipe Face Mesh yang menghasilkan 468 titik landmark. Kemudian saya ambil beberapa titik yaitu philtrum (13), sudut kiri mulut (61),

dan sudut kanan mulut (291) untuk menentukan posisi overlay filter digital kumis. Jarak antara kedua sudut mulut dihitung untuk ukuran kumis sehingga skalanya jadi sesuai dengan lebar mulut. Sudut rotasi kumisnya didapat dari perbedaan koordinat mulut kiri dan kanan agar kumis ikut kemiringan wajah. Setelah itu, gambar kumis dicek untuk memastikan channel alphanya, lalu ditempatkan di titik philtrum dikurang 112 (dikurang supaya kumisnya pas). Lalu pakai alpha blending untuk membuat kumis lebih natural dengan tekstur kulit di bawahnya.

Tantangannya adalah saat menentukan rotasi gambarnya, rotasinya harus melakukan perhitungan antara dua titik untuk mendapatkan kemiringan mulut dan juga sempat kebalik balik untuk perputarannya.

Soal 5 — Perspektif dan Peningkatan Kualitas Citra

- Ambil gambar **objek datar** seperti karya tangan di kertas, tulisan di papan tulis, atau foto produk di meja dengan kondisi pencahayaan atau sudut yang tidak ideal.
- Lakukan **preprocessing** untuk memperbaiki tampilan agar lebih rapi dan jelas, dengan langkah-langkah:
 - Konversi ke **grayscale**.
 - **Koreksi perspektif (transformasi homografi)** menggunakan **4 titik manual** agar objek terlimustache sejajar dan tidak terdistorsi.
 - Terapkan **thresholding adaptif atau Otsu** (pilih salah satu, dan jelaskan alasan pilihan Anda).
- Tampilkan **setiap tahap pemrosesan dalam satu grid** agar mudah dibandingkan.
- Jelaskan fungsi masing-masing tahap dan bagaimana teknik ini meningkatkan kualitas visual citra (gunakan **Markdown**).

```
In [50]: kb_path = cv2.imread(os.path.join("assets_ws4", "keyboard.jpg"))
kb = cv2.cvtColor(kb_path, cv2.COLOR_BGR2RGB)
kb_gray = cv2.cvtColor(kb, cv2.COLOR_RGB2GRAY)

rows, cols = kb_gray.shape

pts1 = np.float32([
    [100, 885],
    [4700, 770],
    [100, 2965],
    [4780, 2850]
])

width = int(np.linalg.norm(pts1[1] - pts1[0]))
height = int(np.linalg.norm(pts1[2] - pts1[0]))

pts2 = np.float32([
    [0, 0],
    [width, 0],
    [0, height],
    [width, height]
])
```

```

M = cv2.getPerspectiveTransform(pts1, pts2)
kb_warp = cv2.warpPerspective(kb_gray, M, (width, height))

kb.blur = cv2.GaussianBlur(kb_warp, (5, 5), 0)
_, kb_otsu = cv2.threshold(kb.blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

plt.figure(figsize=(16, 10))

plt.subplot(2, 2, 1)
plt.imshow(kb_gray, cmap='gray')
plt.title("Gambar Keyboard Grayscale")
plt.axis("off")

plt.subplot(2, 2, 2)
plt.imshow(kb_warp, cmap='gray')
plt.title("Hasil Warp Perspective")
plt.axis("off")

plt.subplot(2, 2, 3)
plt.imshow(kb.blur, cmap='gray')
plt.title("Blur untuk Otsu (Gaussian 5x5)")
plt.axis("off")

plt.subplot(2, 2, 4)
plt.imshow(kb_otsu, cmap='gray')
plt.title("Otsu Thresholding")
plt.axis("off")

plt.tight_layout()
plt.show()

#simpan semua hasil gambar ke results_ws4
os.makedirs('results_ws4', exist_ok=True)
cv2.imwrite(os.path.join('results_ws4', 'kb_gray.jpg'), kb_gray)
cv2.imwrite(os.path.join('results_ws4', 'kb_warp.jpg'), kb_warp)
cv2.imwrite(os.path.join('results_ws4', 'kb_blur.jpg'), kb.blur)
cv2.imwrite(os.path.join('results_ws4', 'kb_otsu.jpg'), kb_otsu)

```

Gambar Keyboard Grayscale



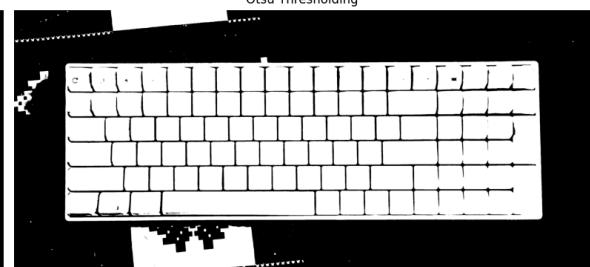
Hasil Warp Perspective



Blur untuk Otsu (Gaussian 5x5)



Otsu Thresholding



Out[50]: True

Gambar keyboard pertamanya dikonversi ke grayscale supaya informasi visual jadi intensitas pixel instead of warna. Lalu 4 titik acuan diambil untuk koreksi prespektif dengan transformasi homografi supaya permukaan yang miring dan terdistorsi dapat tampak lurus dan seperti dilihat dari atas. Setelah diratakan, gambar hasilnya diapplykan filter gaussian blur 5x5 untuk mengurangi noise. Lalu, diterapkan metode otsu yang otomatis menentukan threshold optimal sehingga background dan objek terpisah tanpa perlu memilih threshold manual.

Referensi

- [ChatGPT](#)
- [Github](#)
- [OpenCV Thresholding](#)

Aturan Umum Pengerjaan

- Kerjakan secara **mandiri**.
 - Bantuan AI (seperti CGPT, Copilot, dsb.) diperbolehkan **dengan bukti percakapan** (Screenshot / link / script percakapan).
 - Source code antar mahasiswa harus berbeda.
 - Jika mendapat bantuan teman, tuliskan nama dan NIM teman yang membantu.
 - Plagiarisme akan dikenakan sanksi sesuai aturan akademik ITERA.
 - Cantumkan seluruh **credit dan referensi** yang digunakan di bagian akhir notebook.
 - Penjelasan setiap soal ditulis dalam **Markdown**, bukan di dalam komentar kode.
-

Aturan Pengumpulan

- Semua file kerja Anda (notebook `.ipynb`, gambar, dan hasil) **wajib diunggah ke GitHub repository tugas sebelumnya**.
 - Gunakan struktur folder berikut di dalam repo Anda:

```
/Nama_NIM_Repo/ # Nama repo sebelumnya
  ├── assets_ws4/    # berisi semua gambar atau video asli
  (input)
  └── results_ws4/   # berisi semua hasil modifikasi dan output
  └── worksheet4.ipynb
  └── NIM_Worksheet4.pdf
```
- File yang dikumpulkan ke **Tally** hanya berupa **hasil PDF** dari notebook Anda, dengan format nama:

NIM_Worksheet4.pdf
- Pastikan notebook telah dijalankan penuh sebelum diekspor ke PDF.
- Sertakan tautan ke repository GitHub Anda di bagian atas notebook atau di halaman pertama PDF.

Catatan Akhir

Worksheet 4 ini bertujuan mengasah pemahaman Anda tentang manipulasi citra digital secara praktis. Gunakan kreativitas Anda untuk menghasilkan hasil visual yang menarik dan penjelasan konseptual yang jelas.