

1. Pengambilan Sinyal Respirasi dengan Mediapipe Pose

MediaPipe adalah framework open-source yang dikembangkan oleh Google untuk pemrosesan multimedia dan visi komputer, terutama untuk aplikasi real-time seperti deteksi pose, wajah, tangan, objek, dan lain-lain. Sebelum melakukan percobaan, terlebih dahulu dilakukan pengambilan video dengan resolusi 1080 30fps dengan durasi ± 60 detik. Lalu, definisikan library yang akan digunakan.

In [20]:

```
import cv2
import mediapipe as mp
import numpy as np
from scipy.signal import butter, filtfilt, find_peaks
import matplotlib.pyplot as plt
```

Berikutnya adalah membuat filter `butterworth bandpass` dengan lowcut dan highcut berdasarkan frekuensi sampling (fs). Filter ini bertujuan untuk mempertahankan frekuensi pernafasan normal (biasanya dalam rentang 0.1-0.7 Hz).

In [21]:

```
# Fungsi untuk filter sinyal (Butterworth bandpass)
def butter_bandpass_filter(data, lowcut, highcut, fs, order=4):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    y = filtfilt(b, a, data)
    return y
```

Kemudian definisikan video yang akan digunakan untuk eksperimen. Proses ini juga akan mendapatkan fps untuk dilakukan filtering dan inisialisasi model Mediapipe Pose untuk mendeteksi landmark tubuh secara real-time.

In [24]:

```
# Baca video
video_path = 'dsp1menit.mp4' # Ganti dengan path video Anda
cap = cv2.VideoCapture(video_path)
fps = cap.get(cv2.CAP_PROP_FPS)

mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False, min_detection_confidence=0.5)
```

Program kemudian akan membaca frame video satu per satu. OpenCV akan membaca video dengan format warna BGR (Blue, Green, Red) secara default, sementara Mediapipe membaca dan memproses format RGB. Oleh karena itu, digunakan fungsi `cv2.cvtColor` untuk mengubah format warna sehingga tidak akan terjadi salah interpretasi pada prosesnya. Program juga akan memproses frame untuk mendapatkan landmark pose. Landmark yang digunakan adalah pada titik 11 dan 12 yang merupakan bahu kanan dan kiri. Posisi vertikal dari landmark akan diambil dan hasil konversi format warna akan disimpan dalam bentuk array agar mudah diproses. Setelah proses selesai, program akan menutup video.

```
In [25]: chest_y = []
frame_count = 0

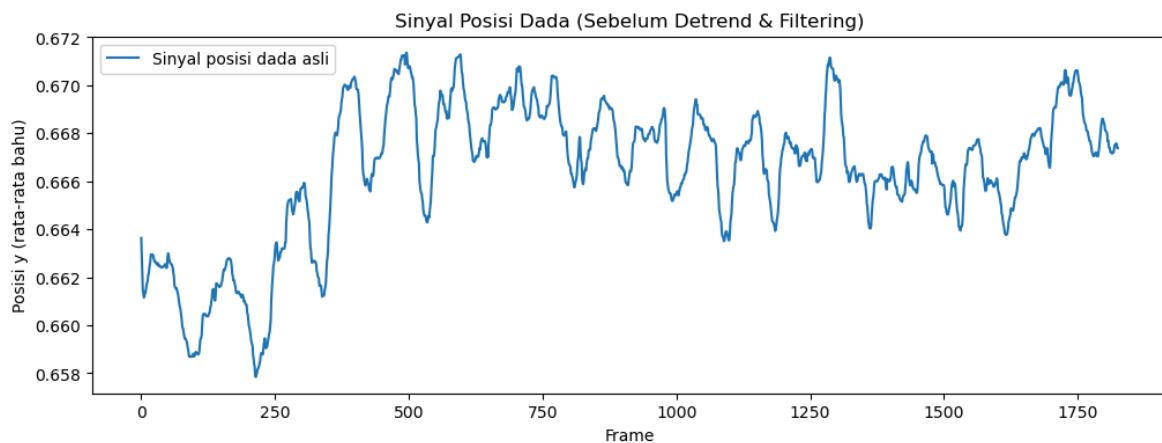
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = pose.process(frame_rgb)
    if results.pose_landmarks:
        # Ambil titik dada (Landmark 11 dan 12: left/right shoulder)
        left_shoulder = results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEF
        right_shoulder = results.pose_landmarks.landmark[mp_pose.PoseLandmark.RI
        chest_y.append((left_shoulder.y + right_shoulder.y) / 2)
    frame_count += 1

cap.release()
pose.close()

chest_y = np.array(chest_y)
```

Kemudian plot sinyal respirasi sebelum dilakukan post-processing.

```
In [26]: # Visualisasi sinyal posisi dada asli sebelum post-processing
plt.figure(figsize=(12,4))
plt.plot(chest_y, label='Sinyal posisi dada asli')
plt.title('Sinyal Posisi Dada (Sebelum Detrend & Filtering)')
plt.xlabel('Frame')
plt.ylabel('Posisi y (rata-rata bahu)')
plt.legend()
plt.show()
```



Berikutnya adalah post-processing. Lakukan detrending sinyal untuk menghilangkan trend. Trend pada sinyal adalah pergerakan umum naik atau turun yang berlangsung perlahan dan menutupi pola osilasi atau fluktuasi yang lebih kecil. Trend harus dihilangkan karena dapat mengganggu proses filtering dan deteksi puncak. Selain itu, dilakukan juga filtering sinyal dengan bandpass filter dan deteksi puncak sinyal sebagai indikasi awal siklus pernafasan.

```
In [27]: # Hilangkan trend (detrend)
chest_y_detrended = chest_y - np.mean(chest_y)

# Filter sinyal (bandpass 0.1-0.7 Hz, laju nafas normal manusia)
```

```

filtered = butter_bandpass_filter(chest_y_detrended, 0.1, 0.7, fps)

# Temukan puncak (peak) sebagai siklus nafas
peaks, _ = find_peaks(filtered, distance=fps*1.5) # Jarak antar peak minimal 1.

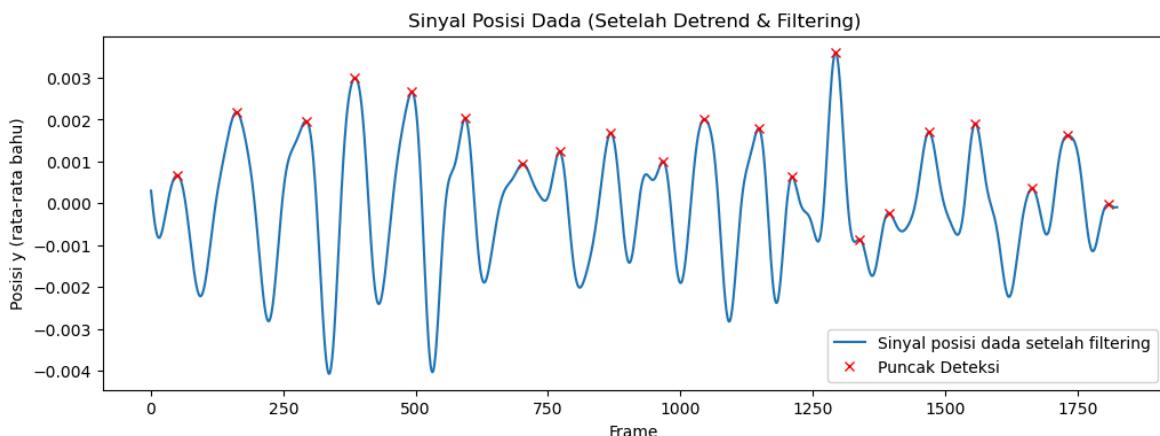
```

Kita lakukan plot untuk melihat sinyal setelah dilakukan post-processing.

```

In [28]: # Visualisasi sinyal posisi dada setelah post-processing
plt.figure(figsize=(12,4))
plt.plot(filtered, label='Sinyal posisi dada setelah filtering')
plt.plot(peaks, filtered[peaks], "rx", label='Puncak Deteksi')
plt.title('Sinyal Posisi Dada (Setelah Detrend & Filtering)')
plt.xlabel('Frame')
plt.ylabel('Posisi y (rata-rata bahu)')
plt.legend()
plt.show()

```



Berikutnya adalah mencari laju pernafasan (BPM) berdasarkan puncak dan durasi video dalam detik.

```

In [29]: # Hitung laju pernafasan (breaths per minute)
duration_sec = len(chest_y) / fps
breaths_per_minute = len(peaks) / duration_sec * 60

```

Diperoleh hasil laju pernafasan adalah 20.70 BPM. Diketahui bahwa nilai normal laju pernafasan orang dewasa (18 tahun ke atas) saat istirahat adalah sekitar 12-20 BPM. Hasil 20.70 BPM menunjukkan bahwa objek dalam video memiliki laju pernafasan **normal** walau nilainya berada sedikit di atas rentang normal. Hal tersebut mungkin dipengaruhi oleh kualitas video atau kondisi objek saat dilakukan pengambilan video (gugup, dll).

```

In [78]: duration_sec_flow = len(chest_y) / fps

print(f"Jumlah puncak: {len(peaks)}")
print(f"Durasi sinyal (detik): {duration_sec_flow:.2f}")

print(f"Laju pernafasan (Mediapipe): {breaths_per_minute:.2f} breaths per minute"

```

Jumlah puncak: 21
 Durasi sinyal (detik): 60.87
 Laju pernafasan (Mediapipe): 20.70 breaths per minute

2. Pengambilan Sinyal Respirasi dengan Lucas-Kanade Optical Flow

Optical flow adalah pola pergerakan piksel atau intensitas cahaya antar dua frame gambar berurutan dalam video. Dengan kata lain, ini adalah cara untuk menghitung arah dan kecepatan gerakan objek atau permukaan dalam citra berdasarkan perubahan intensitas dari satu frame ke frame berikutnya.

Metode Lucas-Kanade adalah salah satu teknik klasik untuk menghitung optical flow yang dikembangkan oleh Bruce D. Lucas dan Takeo Kanade pada tahun 1981. Metode ini berasumsi bahwa intensitas cahaya piksel tidak berubah secara signifikan antara dua frame berturut-turut meskipun posisi piksel berubah (brightness constancy assumption). Tujuannya adalah menemukan vektor perpindahan (u, v) di bidang gambar/video yang menunjukkan perpindahan posisi piksel antara frame pertama dan kedua.

Proses pengambilan sinyal respirasi dilakukan dengan video yang sama dengan sebelumnya. Mula-mula definisikan library yang akan digunakan.

In [87]:

```
import cv2
import numpy as np
from scipy.signal import butter, filtfilt, find_peaks
import matplotlib.pyplot as plt
```

Di sini juga digunakan filter `butterworth bandpass` yang sama dengan metode sebelumnya.

In [88]:

```
# Fungsi filter Butterworth bandpass (sama seperti sebelumnya)
def butter_bandpass_filter(data, lowcut, highcut, fs, order=4):
    from scipy.signal import butter, filtfilt
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    y = filtfilt(b, a, data)
    return y
```

Kemudian program akan membuka video dan mendapatkan fps (frame per detik) untuk parameter filtering dan penghitungan waktu.

In [89]:

```
video_path = 'dsp1menit.mp4'

# --- Pengambilan Sinyal Respirasi dengan Lucas-Kanade Optical Flow ---
cap = cv2.VideoCapture(video_path)

# Definisi fps setelah membuka video ulang
fps = cap.get(cv2.CAP_PROP_FPS)
```

Berikutnya, atur paramater Lucas-Kanade, dimana:

- `winSize` : ukuran jendela pencarian pergerakan.
- `maxLevel` : jumlah level pyramid untuk skala (multi resolusi).

- `criteria` : kondisi penghentian iterasi algoritma.

```
In [90]: lk_params = dict(winSize=(15, 15), maxLevel=2,
                      criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
```

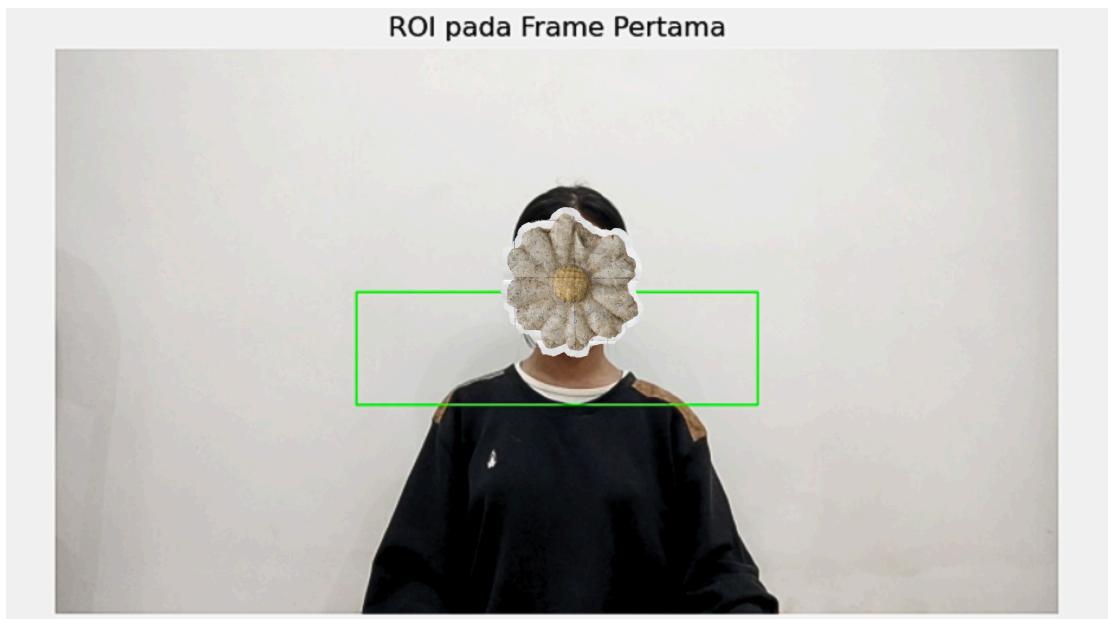
Program lalu akan mengambil frame pertama dan mengonversi ke format grayscale karena metode Lucas-Kanade bekerja dalam format grayscale.

```
In [91]: ret, first_frame = cap.read()
if not ret:
    raise RuntimeError("Tidak bisa membaca frame pertama video.")
first_gray = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)
h, w = first_gray.shape
```

Lalu, tentukan area fokus ROI pada frame pertama sebagai referensi. ROI akan berada di 30% dari lebar frame mulai dari kiri, 43% dari tinggi frame mulai dari atas. Luas ROI sekitar 40% lebar dan 20% tinggi frame.

```
In [92]: roi_x = int(w * 0.3)
roi_y = int(h * 0.43)
roi_w = int(w * 0.4)
roi_h = int(h * 0.2)
roi = first_gray[roi_y:roi_y+roi_h, roi_x:roi_x+roi_w]
```

Berikut adalah ROI pada frame pertama.



Kemudian program akan mendeteksi titik-titik kuat yang dapat dilacak dengan Lucas-Kanade di dalam ROI.

```
In [93]: p0 = cv2.goodFeaturesToTrack(roi, mask=None, maxCorners=50, qualityLevel=0.01, m
if p0 is not None:
    p0[:, 0, 0] += roi_x
    p0[:, 0, 1] += roi_y
else:
    raise RuntimeError("Tidak ditemukan fitur pada ROI.")
```

Kemudian buat sebuah variabel `flow_y` yang akan menyimpan posisi vertikal rata-rata titik fitur dari setiap frame (menandakan gerakan naik-turun). Serta variabel `prev_gray` dan `prev_pts` untuk menyimpan frame dan titik fitur sebelumnya sebagai referensi.

```
In [94]: flow_y = []

prev_gray = first_gray.copy()
prev_pts = p0.copy()
```

Kemudian akan dilakukan looping untuk membaca frame. Untuk setiap frame akan dilakukan konversi ke format grayscale, menghitung optical flow titik fitur dari frame sebelumnya ke frame sekarang, mengambil rata-rata dari posisi vertikal dan simpan di variabel `flow_y`, mengupdate titik fitur dan frame sebelumnya untuk iterasi berikutnya, dan mengulang nilai terakhir apabila tracking gagal. Hasil dari proses akan dikonversi menjadi array numpy.

```
In [95]: while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    p1, st, err = cv2.calcOpticalFlowPyrLK(prev_gray, frame_gray, prev_pts, None)
    if p1 is not None and st.sum() > 0:
        good_new = p1[st == 1]
        mean_y = np.mean(good_new[:, 1])
        flow_y.append(mean_y)
        prev_pts = good_new.reshape(-1, 1, 2)
    else:
        flow_y.append(flow_y[-1] if flow_y else 0)
    prev_gray = frame_gray.copy()

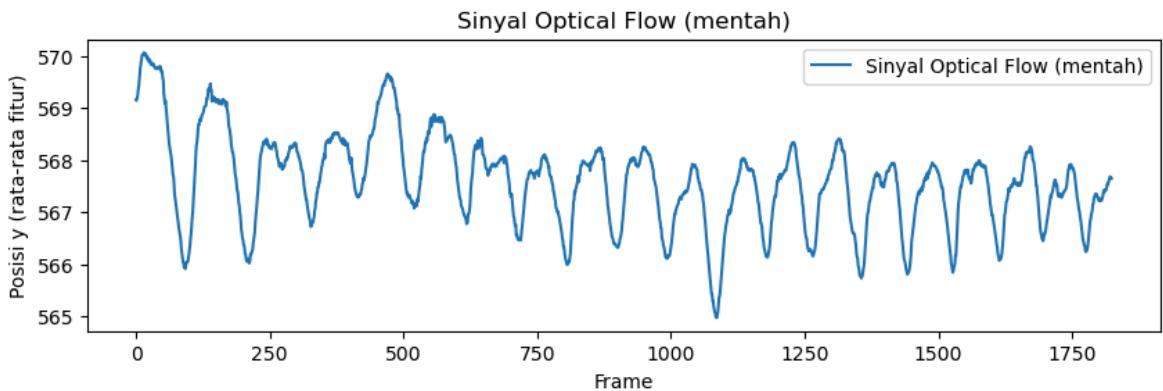
cap.release()

flow_y = np.array(flow_y)
```

Lakukan visualisasi sinyal sebelum dilakukan post-processing.

```
In [96]: #visualisasi sinyal sebelum post-processing
plt.figure(figsize=(10,6))
plt.subplot(2, 1, 1)
plt.plot(flow_y, label='Sinyal Optical Flow (mentah)')
plt.title('Sinyal Optical Flow (mentah)')
plt.xlabel('Frame')
plt.ylabel('Posisi y (rata-rata fitur)')
plt.legend()
```

```
Out[96]: <matplotlib.legend.Legend at 0x291cde07bc0>
```



Kemudian lakukan post-processing. Hilangkan trend agar sinyal berosilasi di sekitar nol, dan filtering dengan bandpass untuk rentang pernafasan normal.

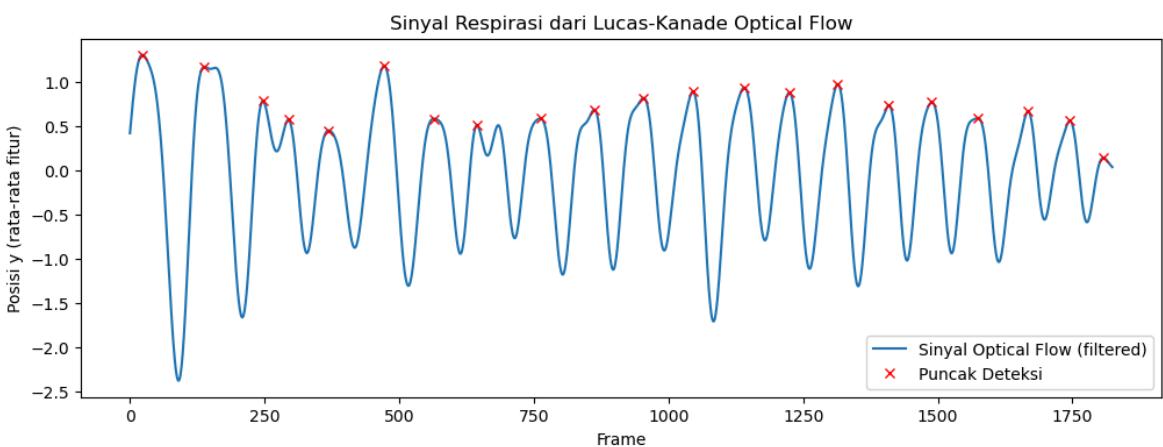
```
In [97]: # Post-processing
flow_y_detrended = flow_y - np.mean(flow_y)
flow_filtered = butter_bandpass_filter(flow_y_detrended, 0.1, 0.7, fps)
```

Temukan puncak (peak) sebagai siklus nafas, hitung durasi video, dan hitung laju pernafasan dalam BPM.

```
In [100...]: peaks_flow, _ = find_peaks(flow_filtered, distance=fps*1.5)
duration_sec_flow = len(flow_y) / fps
bpm_flow = len(peaks_flow) / duration_sec_flow * 60
```

Lakukan visualisasi untuk sinyal setelah dilakukan post-processing.

```
In [101...]: plt.figure(figsize=(12,4))
plt.plot(flow_filtered, label='Sinyal Optical Flow (filtered)')
plt.plot(peaks_flow, flow_filtered[peaks_flow], "rx", label='Puncak Deteksi')
plt.title('Sinyal Respirasi dari Lucas-Kanade Optical Flow')
plt.xlabel('Frame')
plt.ylabel('Posisi y (rata-rata fitur)')
plt.legend()
plt.show()
```



Diperoleh hasil laju pernafasan adalah 20.71 BPM. Hasil 20.71 BPM menunjukkan bahwa objek dalam video memiliki laju pernafasan **normal** walau nilainya berada sedikit di atas rentang normal.

```
In [102...]: duration_sec_flow = len(flow_y) / fps
print(f"Jumlah puncak: {len(peaks_flow)}")
print(f"Durasi sinyal (detik): {duration_sec_flow:.2f}")
print(f"Laju pernafasan (Lucas-Kanade Optical Flow): {bpm_flow:.2f} breaths per minute")

Jumlah puncak: 21
Durasi sinyal (detik): 60.83
Laju pernafasan (Lucas-Kanade Optical Flow): 20.71 breaths per minute
```

Perbandingan Data Kedua Metode

Berikut adalah tabel data hasil pencatatan pernafasan yang dihitung dengan stopwatch pada HP ketika proses pengambilan video. Video diambil selama ± 60 detik. Diperoleh sekitar 20 BPM melalui pengukuran dengan stopwatch.

Nafas-ke	Second	Milisecond
1	4	13
2	8	27
3	11	84
4	15	15
5	18	54
6	21	52
7	24	69
8	27	83
9	31	11
10	34	13
11	37	00
12	40	21
13	43	32
14	46	19
15	48	90
16	51	73
17	54	69
18	57	36
19	60	40
20	61	88

Kemudian kita bandingkan hasil laju pernafasan (BPM) yang diperoleh dengan kedua metode seperti berikut ini. Durasi sinyal yang diproses oleh kedua metode sedikit berbeda, sehingga BPM yang diperoleh juga sedikit berbeda. Pada metode Mediapipe

Pose diperoleh sekitar 20.70 BPM, sementara pada metode Lucas-Kanade diperoleh sekitar 20.71 BPM.

Metode	Jumlah puncak	Durasi sinyal	BPM
Mediapipe	21	60.87	20.70
Lucas-Kanade	21	60.83	20.71

Jumlah BPM yang diperoleh dengan kedua metode ini juga mengalami sedikit perbedaan dengan pengukuran menggunakan stopwatch, yaitu sekitar 20 BPM.

Kesimpulan

Berdasarkan seluruh percobaan yang telah dilakukan, diperoleh bahwa kedua metode ekstraksi laju pernafasan dari video, yaitu menggunakan Mediapipe Pose dan Lucas-Kanade Optical Flow, menghasilkan estimasi laju pernafasan yang sangat mirip, yaitu sekitar 20.70 BPM (Mediapipe) dan 20.71 BPM (Lucas-Kanade). Nilai ini juga sangat dekat dengan hasil pengukuran manual menggunakan stopwatch (sekitar 20 BPM). Hal ini menunjukkan bahwa kedua metode dapat digunakan untuk memantau laju pernafasan secara non-invasif dari video dengan tingkat akurasi yang baik. Perbedaan kecil yang muncul kemungkinan disebabkan oleh perbedaan durasi sinyal yang terdeteksi, kualitas video, serta sensitivitas masing-masing metode terhadap noise dan artefak gerakan. Secara keseluruhan, baik metode Mediapipe maupun Lucas-Kanade Optical Flow dapat diandalkan untuk aplikasi pemantauan laju pernafasan berbasis video.

Referensi

[Link Referensi](#)