# VISUAL INNOVATIONS:

## MODUL INOVATIF PEMBELAJARAN PEMROGRAMAN COMPUTER VISION

### CASE-STUDY OPTICAL FLOW

2023

PROGRAM STUDI INFOMATIKA

INSTITUT TEKNOLOGI NASIONAL BANDUNG

By: Irma Amelia Dewi

```
# **Study Case Interaktif Optical Flow**
```

# Study Case Interaktif Optical Flow

Optical flow adalah teknik dalam pengolahan citra komputer yang digunakan untuk melacak pergerakan objek atau piksel dalam bingkai (frame) berurutan dalam sebuah video atau sekuens citra. Tujuan utama dari optical flow adalah untuk memahami bagaimana piksel dalam bingkai bergerak dari satu bingkai ke bingkai berikutnya.

Dalam konteks optical flow, pergerakan objek atau piksel diukur dalam bentuk vektor pergeseran (perubahan posisi) pada setiap piksel. Vektor ini menunjukkan arah dan kecepatan pergerakan objek atau piksel dari satu bingkai ke bingkai berikutnya dengan ditandai dengan warna.

Berikut adalah tutorial Optical Flow untuk melacak pergerakan dari frame ke frame berikutnya dengan ditandai dengan warna.

## 1. Import Library

Kita akan mengimpor pustaka OpenCV ( `cv2` ), fungsi `from google.colab.patches import cv2_imshoww` untuk menampilkan citra di Google Colab, dan pustaka NumPy ( `numpy` ) untuk pengolahan numerik

```
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
```

## 2. Buka video

Lalu kita akan membuka video menggunakan fungsi `cv2.VideoCapture` . Kita akan coba menggunakan 'people.mp4'

```
cap = cv2.VideoCapture('people.mp4')
```

## 3. Baca dua bingkai pertama

Selanjutnya kita baca dua bingkai pertama dari video dan mengonversi bingkai pertama ke citra grayscale. Citra grayscale ini akan digunakan sebagai bingkai sebelumnya untuk menghitung perubahan antara bingkai saat ini dengan bingkai sebelumnya.

latar belakang gambar diinisialisasi citra bingkai pertama dari video, yang seharusnya adalah gambar latar belakang aktual dari video. Hal ini dicapai dengan menggunakan `frame1` sebagai latar belakang awal sebelum menghitung optical flow. Dengan cara ini, latar belakang gambar tidak akan menjadi hitam, tetapi akan mempertahankan konten dari bingkai pertama sebagai latar belakangnya.

```
ret, frame1 = cap.read()
prev_gray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
```

## 4. Buat window untuk visualisasi optical flow

Kita buat variabel kosong `hsv` dengan ukuran yang sama dengan bingkai pertama dan mengatur komponen saturasi (S) ke 255, sehingga citra akan memiliki warna

```
hsv = np.zeros_like(frame1)
hsv[..., 1] = 255
```

## 5. Konfigurasi video output

Kita konfigurasikan output video dengan nama 'output_optical_flow.avi' menggunakan codec MJPG dengan frame rate 10 fps (frame per detik) dan ukuran frame yang sama dengan video input.

```
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
out = cv2.VideoWriter('output_optical_flow.avi', cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (frame_width, frame_height))
```

## 6. Looping untuk memproses setiap bingkai dalam video

Kita buat looping untuk membaca dan memproses setiap frame dalam video. Didalam looping ini kita akan membaca dari frame ke frame selanjutnya. Jika tidak ada frame selanjutnya (video telah selesai), maka looping akan berhenti.

```
while True:
    ret, frame2 = cap.read()
    if not ret:
        break
```

## 7. Ubah Frame citra ke grayscale

ini juga masih bagian dari loopimg dimana kita akan mengonversi frame saat ini ke citra grayscale.

**Isikan Baris kode dibawah dengan benar!**

```
# Ubah Frame citra ke grayscale
```

## 8. Menghitung Optical Flow menggunakan algoritma Lucas-Kanade

Kita masuk ke langkah penting di mana optical flow dihitung antara bingkai sebelumnya ( `prev_gray` ) dan bingkai saat ini ( `gray` ) menggunakan algoritma Lucas-Kanade dengan parameter yang ditentukan.

**Isikan fungsi `cv2.calcOpticalFlowFarneback` dan parameter yang kosong dibawah dengan benar!**

```
# Menghitung Optical Flow menggunakan algoritma Lucas-Kanade

flow = *** ( *** , *** , None, 0.5, 3, 15, 3, 5, 1.2, 0)
```

## 9. Konversi ke koordinat polar

Magnitude (besar) dan arah (angle) optical flow dihitung dari hasil perhitungan sebelumnya. Dengan mengonversi vektor optical flow ke koordinat polar, visualisasi optical flow menjadi lebih informatif dan mudah dibaca. Magnitude memberikan informasi tentang seberapa besar perubahan, sedangkan arah memberikan informasi tentang arah perubahan. Kombinasi keduanya memberikan gambaran lengkap tentang bagaimana pergerakan terjadi antara dua bingkai dalam video, yang membantu dalam pemahaman dan analisis pergerakan objek.

```
magnitude, angle = cv2.cartToPolar(flow[..., 0], flow[..., 1])
```

## 10. Berikan Warna pada citra berdasarkan arah dan magnitude optical flow

Warna pada citra diatur berdasarkan arah optical flow (hue) dan magnitude optical flow (nilai kecerahan). Jadi, dalam hasil outputnya, warna merah biasanya menunjukkan pergerakan ke kanan, kuning menunjukkan pergerakan ke atas, Warna hijau pergerakan ke arah bawah, dan Warna biru pergerakan ke arah kiri. Tergantung pada arah pergerakan yang terdeteksi oleh algoritma Optical Flow. Warna-warna ini digunakan untuk memberikan visualisasi tentang arah dan kecepatan pergerakan objek dalam video.

```
hsv[..., 0] = angle * 180 / np.pi / 2
hsv[..., 2] = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)
bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

## 11. Gabungkan Visualisasi optical flow ke bingkai asli

Kita tambahkan visualisasi optical flow ke citra yang akan disimpan dalam variabel `combined_frame` , yang merupakan hasil dari citra bingkai asli yang telah digabungkan dengan visualisasi optical flow.

Dalam konteks kode tersebut, "bingkai asli" mengacu pada bingkai dari video input yang disimpan dalam variabel `frame2` . Visualisasi optical flow, yang disimpan dalam variabel `bgr` , ditambahkan ke citra frame2, dan hasilnya disimpan dalam variabel `combined_frame` .

**Isikan fungsi `cv2.addWeighted` dan parameter yang kosong dibawah dengan benar!**

```
# Gabungkan Visualisasi optical flow ke bingkai asli

combined_frame = *** ( *** , 1, ***, 2, 0)
```

## 12. Tampilkan Hasil penggabungan

Citra yang telah digabungkan ditampilkan di jendela Google Colab menggunakan `cv2_imshow`

```
cv2_imshow(combined_frame)
```

## 13. Simpan frame yang telah digabung ke dalam video output

Bingkai yang telah digabungkan disimpan ke dalam video output

```
out.write(combined_frame)
```

## 14. Tombol 'Esc' untuk keluar

Supaya kita tidak mau semua frame dibaca maka kita akan buat saat tombol 'Esc' ditekan, loop/program akan dihentikan.

```
if cv2.waitKey(25) & 0xFF == 27:
    break
```

## 15. Ganti bingkai sebelumnya dengan bingkai saat ini

Citra grayscale saat ini akan digunakan sebagai bingkai sebelumnya untuk perhitungan optical flow pada bingkai berikutnya.

```
prev_gray = gray.copy()
```

## 16. Tutup video output dan semua jendela

Setelah loop/program selesai, video output akan ditutup, objek video input juga akan ditutup, dan semua jendela OpenCV akan ditutup.

```
out.release()
cap.release()
cv2.destroyAllWindows()
```