```python
# -*- coding: utf-8 -*-
"""Object_tracker

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1ZjxR1hsy5IXAPUqVDJru4C-xr3scy5hd
"""

import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow

def imshow(title="Image",image=None,size=10):
  w,h = image.shape[0], image.shape[1]
  aspect_ratio = w/h
  plt.figure(figsize = (aspect_ratio*size,size))
  plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
  plt.title(title)
  plt.show()

!wget https://github.com/rajeevratan84/ModernComputerVision/raw/main/walking_short_clip.mp4
!wget https://github.com/rajeevratan84/ModernComputerVision/raw/main/walking.avi

cap = cv2.VideoCapture('walking.avi')

width = int(cap.get(3))
height = int(cap.get(4))

output = cv2.VideoWriter('output_walking3.avi', cv2.VideoWriter_fourcc('M','J','P','G'), 15, (width,height))

feature_params = dict( maxCorners = 100,
                       qualityLevel = 0.5,
                       minDistance = 7,
                       blockSize = 7 )

lucas_kanade_params = dict( winSize  = (15,15),
                  maxLevel = 2,
                  criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

color = np.random.randint(0,255,(100,3))

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

prev_corners = cv2.goodFeaturesToTrack(prev_gray, mask = None, **feature_params)

mask = np.zeros_like(prev_frame)

tracked_objects = []

while(1):
    ret, frame = cap.read()

    if ret == True:
      frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      # calculate optical flow
      new_corners, status, errors = cv2.calcOpticalFlowPyrLK(prev_gray,
                                                             frame_gray,
                                                             prev_corners,
                                                             None,
                                                             **lucas_kanade_params)

      # Select and store good points
      good_new = new_corners[status==1]
      good_old = prev_corners[status==1]

      # Draw the tracks
      for i,(new,old) in enumerate(zip(good_new, good_old)):
          a, b = map(int, new.ravel())
          c, d = map(int, old.ravel())

          velocity = np.sqrt((a - c)**2 + (b - d)**2)
          direction = np.arctan2(b - d, a - c)

          tracked_objects.append({
                'position': (a, b),
                'velocity': velocity,
                'direction': direction
            })
```

```
            mask = cv2.line(mask, (a,b),(c,d), color[i].tolist(), 2)
            frame = cv2.circle(frame, (a,b), 5, color[i].tolist(),-1)

            text = f"Vel: {velocity:.2f}, Dir: {np.degrees(direction):.2f}"
            frame = cv2.putText(frame, text, (int(a), int(b)), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

        img = cv2.add(frame,mask)

        # Save Video
        output.write(img)
        # Show Optical Flow
        #imshow('Optical Flow - Lucas-Kanade',img)

        # Now update the previous frame and previous points
        prev_gray = frame_gray.copy()
        prev_corners = good_new.reshape(-1,1,2)

    else:
      break

cap.release()
output.release()

!ffmpeg -i /content/output_walking3.avi output_walking3.mp4 -y

import cv2
import numpy as np
import argparse

# Parse command-line arguments
parser = argparse.ArgumentParser(description='Object Tracking using Optical Flow')
parser.add_argument('test_video.avi', help='Path to the input video file')
args = parser.parse_args()

# Open the video file
cap = cv2.VideoCapture(args.video_path)

width = int(cap.get(3))
height = int(cap.get(4))

output = cv2.VideoWriter('output_video.mp4', cv2.VideoWriter_fourcc('M','J','P','G'), 15, (width,height))

feature_params = dict( maxCorners = 100,
                       qualityLevel = 0.5,
                       minDistance = 7,
                       blockSize = 7 )

lucas_kanade_params = dict( winSize  = (15,15),
                  maxLevel = 2,
                  criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

color = np.random.randint(0,255,(100,3))

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

prev_corners = cv2.goodFeaturesToTrack(prev_gray, mask = None, **feature_params)

mask = np.zeros_like(prev_frame)

tracked_objects = []

while(1):
    ret, frame = cap.read()

    if ret == True:
      frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

      # calculate optical flow
      new_corners, status, errors = cv2.calcOpticalFlowPyrLK(prev_gray,
                                                             frame_gray,
                                                             prev_corners,
                                                             None,
                                                             **lucas_kanade_params)

      # Select and store good points
      good_new = new_corners[status==1]
      good_old = prev_corners[status==1]

      # Draw the tracks
      for i,(new,old) in enumerate(zip(good_new, good_old)):
          a, b = map(int, new.ravel())
          c, d = map(int, old.ravel())
```

```
          velocity = np.sqrt((a - c)**2 + (b - d)**2)
          direction = np.arctan2(b - d, a - c)

          tracked_objects.append({
                'position': (a, b),
                'velocity': velocity,
                'direction': direction
            })

          mask = cv2.line(mask, (a,b),(c,d), color[i].tolist(), 2)
          frame = cv2.circle(frame, (a,b), 5, color[i].tolist(),-1)

          text = f"Vel: {velocity:.2f}, Dir: {np.degrees(direction):.2f}"
          frame = cv2.putText(frame, text, (int(a), int(b)), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

      img = cv2.add(frame,mask)

      # Save Video
      output.write(img)
      # Show Optical Flow
      #imshow('Optical Flow - Lucas-Kanade',img)

      # Now update the previous frame and previous points
      prev_gray = frame_gray.copy()
      prev_corners = good_new.reshape(-1,1,2)

    else:
      break

cap.release()
output.release()

# Release the video capture object
cap.release()
output.release()
cv2.destroyAllWindows()

from google.colab import drive
drive.mount('/content/drive')
```