Name: Tiển Chí Sâm

Student ID: 2151010319

Exercise 1: Object Tracking

Source Code:

```python
import cv2
import numpy as np

# Initialize variables
drawing = False
ix, iy = -1, -1
bbox = []

# Mouse callback function to draw a rectangle


def draw_rectangle(event, x, y, flags, param):
    global ix, iy, drawing, bbox

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            img_copy = frame.copy()
            cv2.rectangle(img_copy, (ix, iy), (x, y), (0, 255, 0), 2)
            cv2.putText(img_copy, 'person', (ix, iy - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.imshow('Frame', img_copy)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        cv2.rectangle(frame, (ix, iy), (x, y), (0, 255, 0), 2)
        bbox.append((ix, iy, x, y))
        cv2.imshow('Frame', frame)


# Initialize video capture
cap = cv2.VideoCapture('video_walking.mp4')
```

```python
37     # Set up the mouse callback
38     cv2.namedWindow('Frame')
39     cv2.setMouseCallback('Frame', draw_rectangle)
40
41     # Initialize Lucas-Kanade parameters
42     lk_params = dict(winSize=(15, 15),
43                      maxLevel=2,
44                      criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
45
46     # Read the first frame
47     ret, frame = cap.read()
48
49     # Resize the first frame
50     frame = cv2.resize(frame, (640, 480))
51
52     old_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
53
54     while cap.isOpened():
55         ret, frame = cap.read()
56         if not ret:
57             break
58
59         # Resize the frame
60         frame = cv2.resize(frame, (640, 480))
61
62         # Convert frame to grayscale
63         frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
64
65         # Draw bounding boxes for user-drawn rectangles
66         for (startX, startY, endX, endY) in bbox:
67             cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 255, 0), 2)
68             cv2.putText(frame, 'person', (startX, startY - 10),
69                         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
70
71         # Calculate optical flow for the tracked points
72         if len(bbox) > 0:
73             p0 = np.array([[x + (endX - startX) / 2, y + (endY - startY) / 2]
74                            for (x, y, endX, endY) in bbox], dtype=np.float32).reshape(-1, 1, 2)
75             p1, st, err = cv2.calcOpticalFlowPyrLK(
76                 old_gray, frame_gray, p0, None, **lk_params)
77
78             # Update bounding boxes based on optical flow
79             for i, (new, old) in enumerate(zip(p1, p0)):
80                 a, b = new.ravel()
81                 c, d = old.ravel()
82                 startX, startY, endX, endY = bbox[i]
83                 startX += int(a - c)
84                 startY += int(b - d)
85                 endX += int(a - c)
86                 endY += int(b - d)
87                 bbox[i] = (startX, startY, endX, endY)
88
89                 # Draw updated bounding boxes
90                 cv2.rectangle(frame, (startX, startY),
91                               (endX, endY), (0, 255, 0), 2)
92                 cv2.putText(frame, 'person', (startX, startY - 10),
93                             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
94
95         # Display the frame
96         cv2.imshow('Frame', frame)
97
98         # Update the previous frame and keypress handling
99         old_gray = frame_gray.copy()
100        key = cv2.waitKey(30) & 0xFF   # adjust the speed of video (1->30)
101        if key == ord('q'):
102            break
```
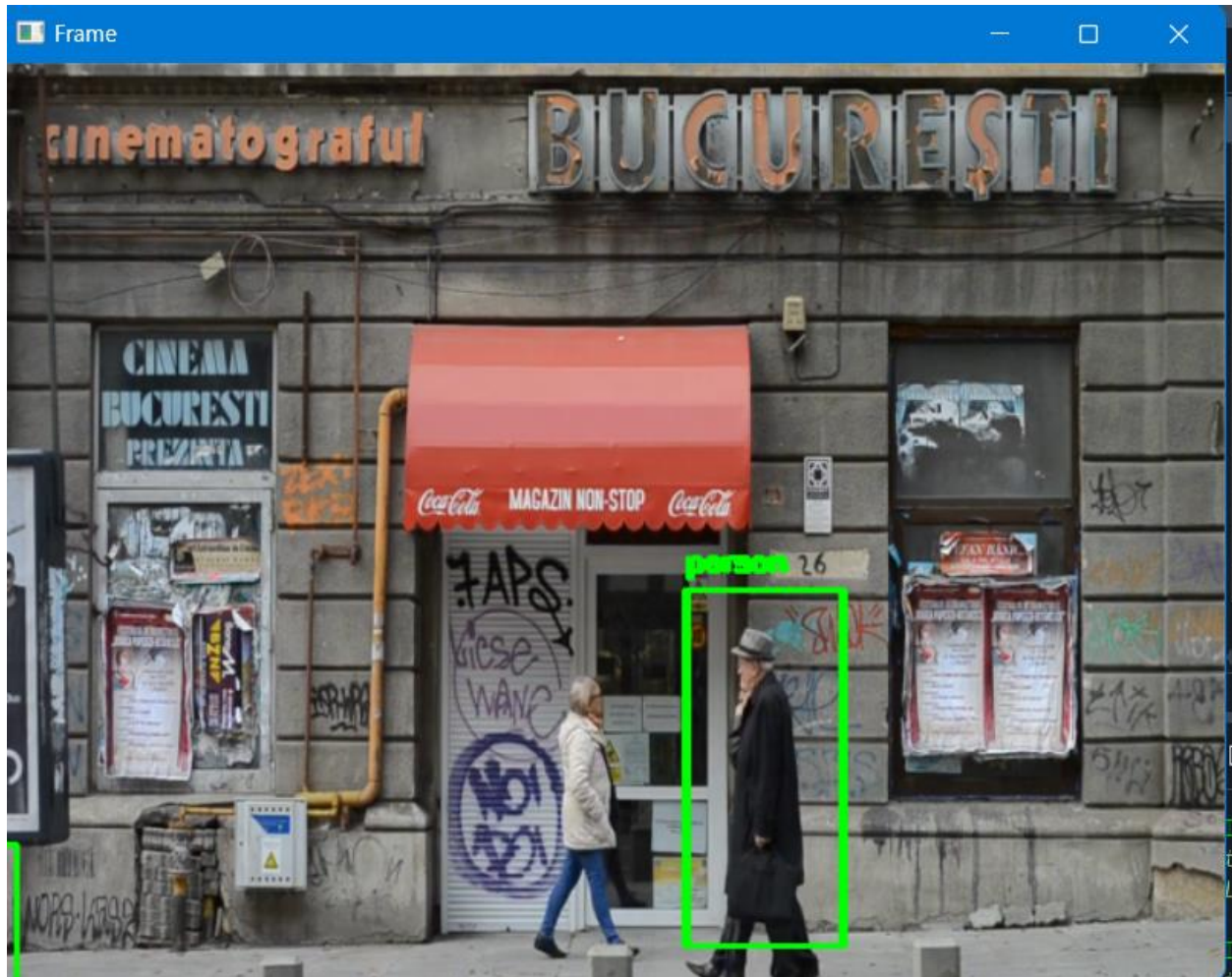
```
103
104    # Release video capture and close all windows
105    cap.release()
106    cv2.destroyAllWindows()
107
```

Result:

Exercise 2: Object Speed Estimation

Source Code:

```python
import cv2
import numpy as np
import time

# Initialize variables
drawing = False
ix, iy = -1, -1
bbox = []
prev_time = None

# Mouse callback function to draw a rectangle


def draw_rectangle(event, x, y, flags, param):
    global ix, iy, drawing, bbox

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            img_copy = frame.copy()
            cv2.rectangle(img_copy, (ix, iy), (x, y), (0, 255, 0), 2)
            cv2.putText(img_copy, 'person', (ix, iy - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.imshow('Frame', img_copy)

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        cv2.rectangle(frame, (ix, iy), (x, y), (0, 255, 0), 2)
        bbox.append((ix, iy, x, y))
        cv2.imshow('Frame', frame)


# Initialize video capture
cap = cv2.VideoCapture('video_walking.mp4')
```

```python
# Set up the mouse callback
cv2.namedWindow('Frame')
cv2.setMouseCallback('Frame', draw_rectangle)

# Initialize Lucas-Kanade parameters
lk_params = dict(winSize=(15, 15),
                 maxLevel=2,
                 criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

# Read the first frame
ret, frame = cap.read()

# Resize the first frame
frame = cv2.resize(frame, (640, 480))

old_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Resize the frame
    frame = cv2.resize(frame, (640, 480))

    # Convert frame to grayscale
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Draw bounding boxes for user-drawn rectangles
    for (startX, startY, endX, endY) in bbox:
        cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 255, 0), 2)
        cv2.putText(frame, 'person', (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

```python
        # Calculate optical flow for the tracked points
        if len(bbox) > 0:
            p0 = np.array([[x + (endX - startX) / 2, y + (endY - startY) / 2]
                          for (x, y, endX, endY) in bbox], dtype=np.float32).reshape(-1, 1, 2)
            p1, st, err = cv2.calcOpticalFlowPyrLK(
                old_gray, frame_gray, p0, None, **lk_params)

            # Update bounding boxes based on optical flow and estimate velocity
            current_time = time.time()
            if prev_time is not None:
                elapsed_time = current_time - prev_time
                for i, (new, old) in enumerate(zip(p1, p0)):
                    a, b = new.ravel()
                    c, d = old.ravel()
                    startX, startY, endX, endY = bbox[i]
                    displacement_x = a - c
                    displacement_y = b - d
                    velocity_x = displacement_x / elapsed_time
                    velocity_y = displacement_y / elapsed_time
                    velocity = np.sqrt(velocity_x**2 + velocity_y**2)
                    cv2.putText(frame, f'Velocity: {velocity:.2f} pixels/sec', (startX, startY - 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

            prev_time = current_time

            # Draw updated bounding boxes
            for i, (new, old) in enumerate(zip(p1, p0)):
                a, b = new.ravel()
                c, d = old.ravel()
                startX, startY, endX, endY = bbox[i]
                startX += int(a - c)
                startY += int(b - d)
                endX += int(a - c)
                endY += int(b - d)
                bbox[i] = (startX, startY, endX, endY)
                cv2.rectangle(frame, (startX, startY),
                              (endX, endY), (0, 255, 0), 2)
                cv2.putText(frame, 'person', (startX, startY - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # Display the frame
    cv2.imshow('Frame', frame)

    # Update the previous frame and keypress handling
    old_gray = frame_gray.copy()
    key = cv2.waitKey(30) & 0xFF  # adjust the speed of video (1->30)
    if key == ord('q'):
        break

# Release video capture and close all windows
cap.release()
cv2.destroyAllWindows()
```

Result: