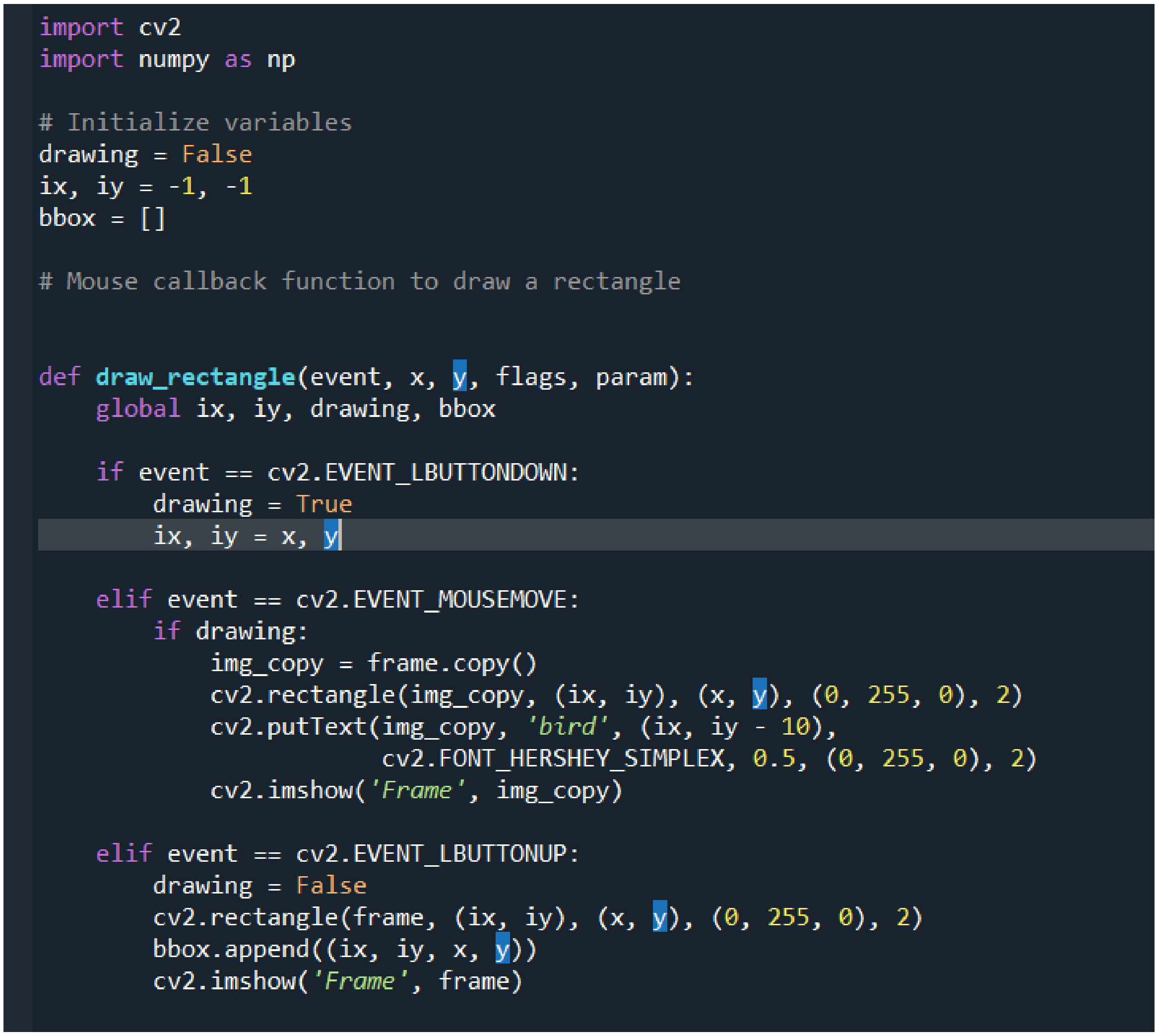
Name: Nguyễn Hoàng Thắng

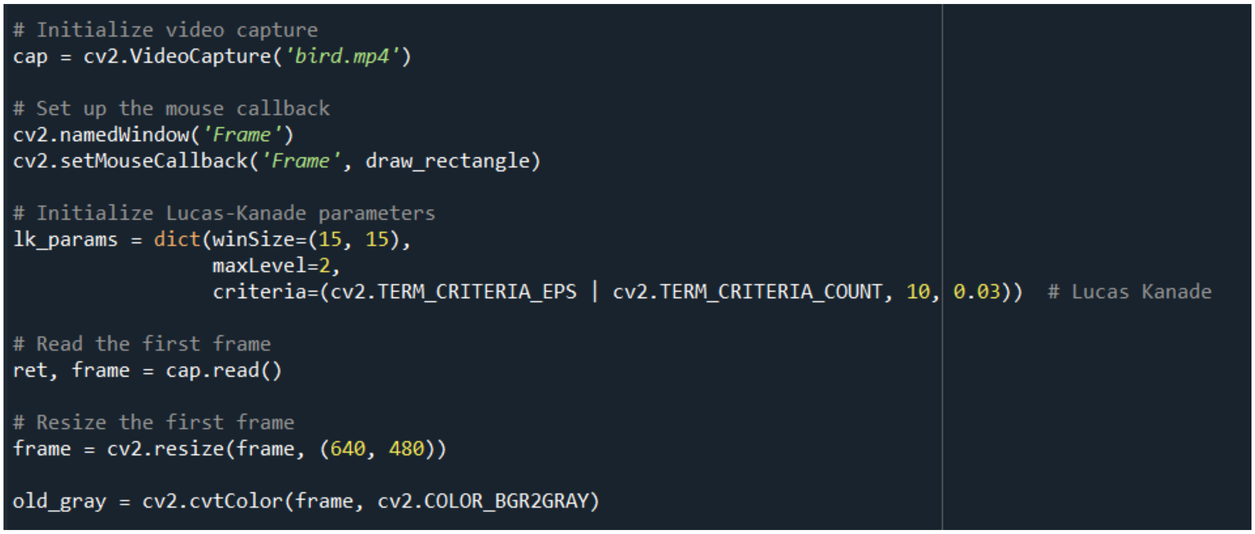
Student ID: 2151012009

Exercise 1: Object Tracking

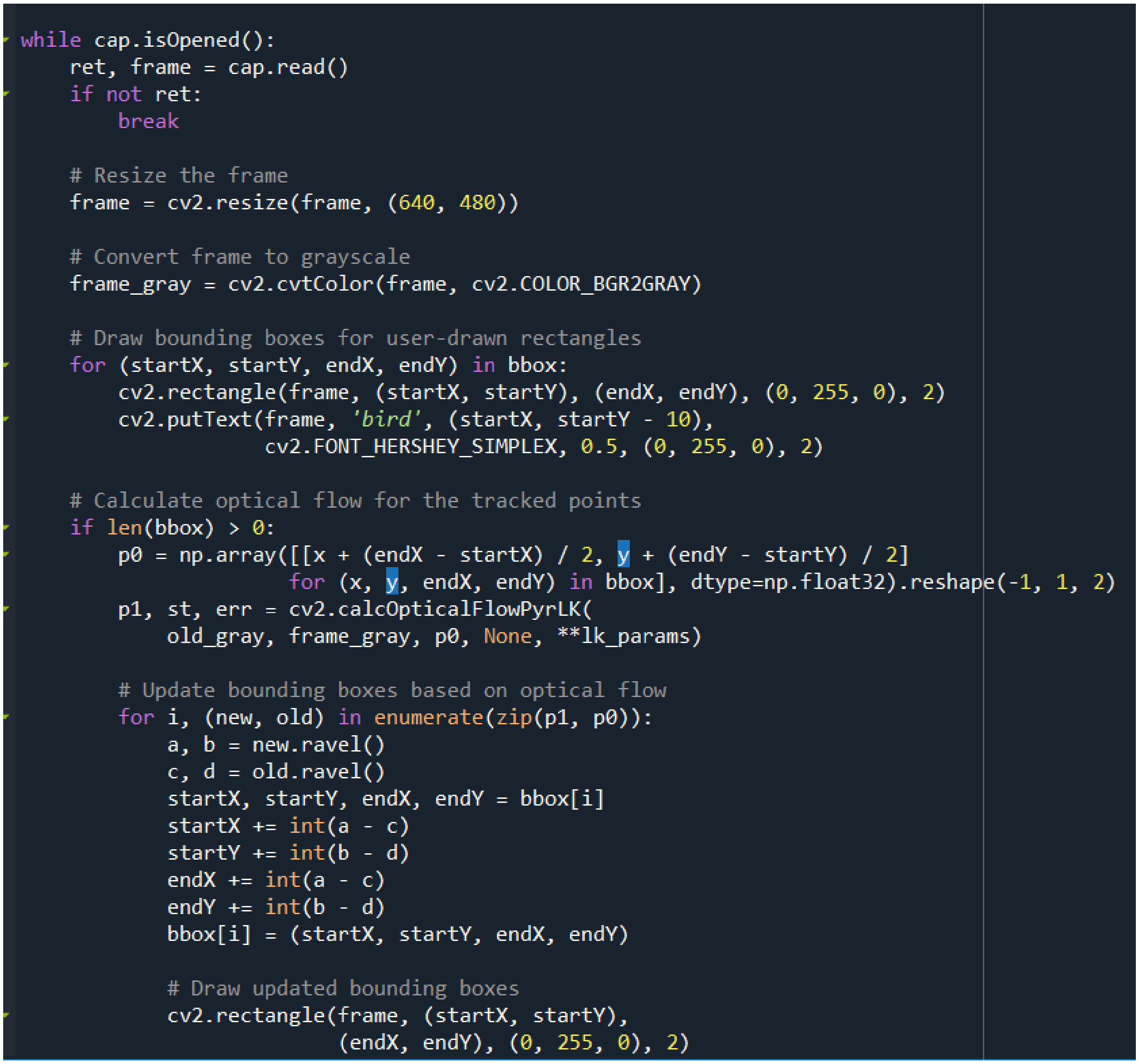
Source Code:



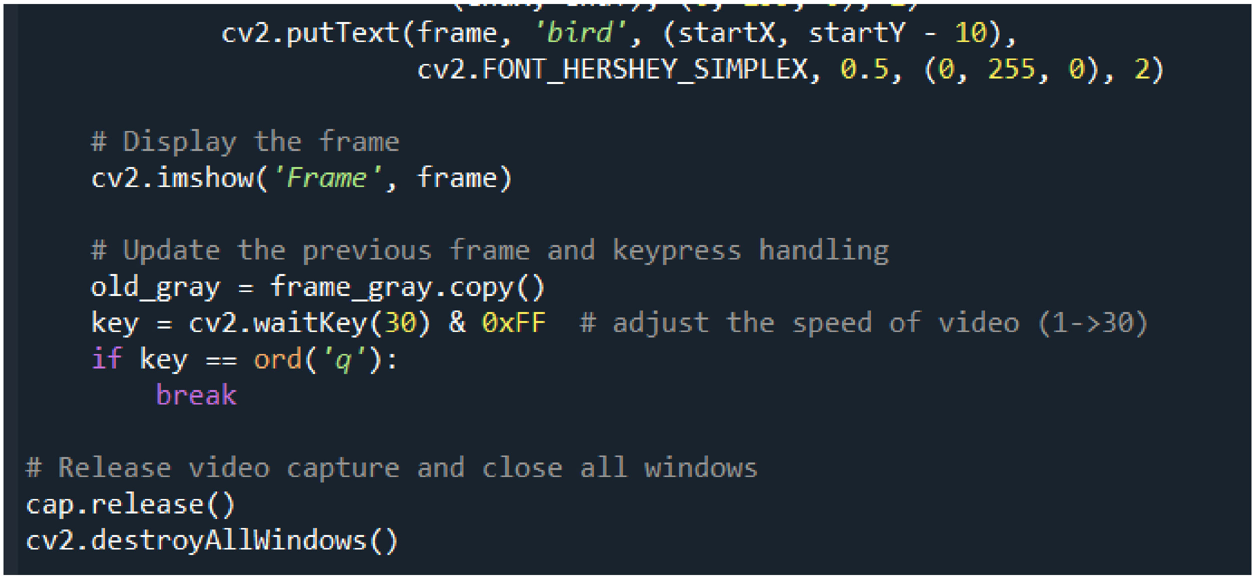
```python
import cv2
import numpy as np

# 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, 'bird', (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)
```



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

# 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))  # Lucas Kanade

# 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)
```

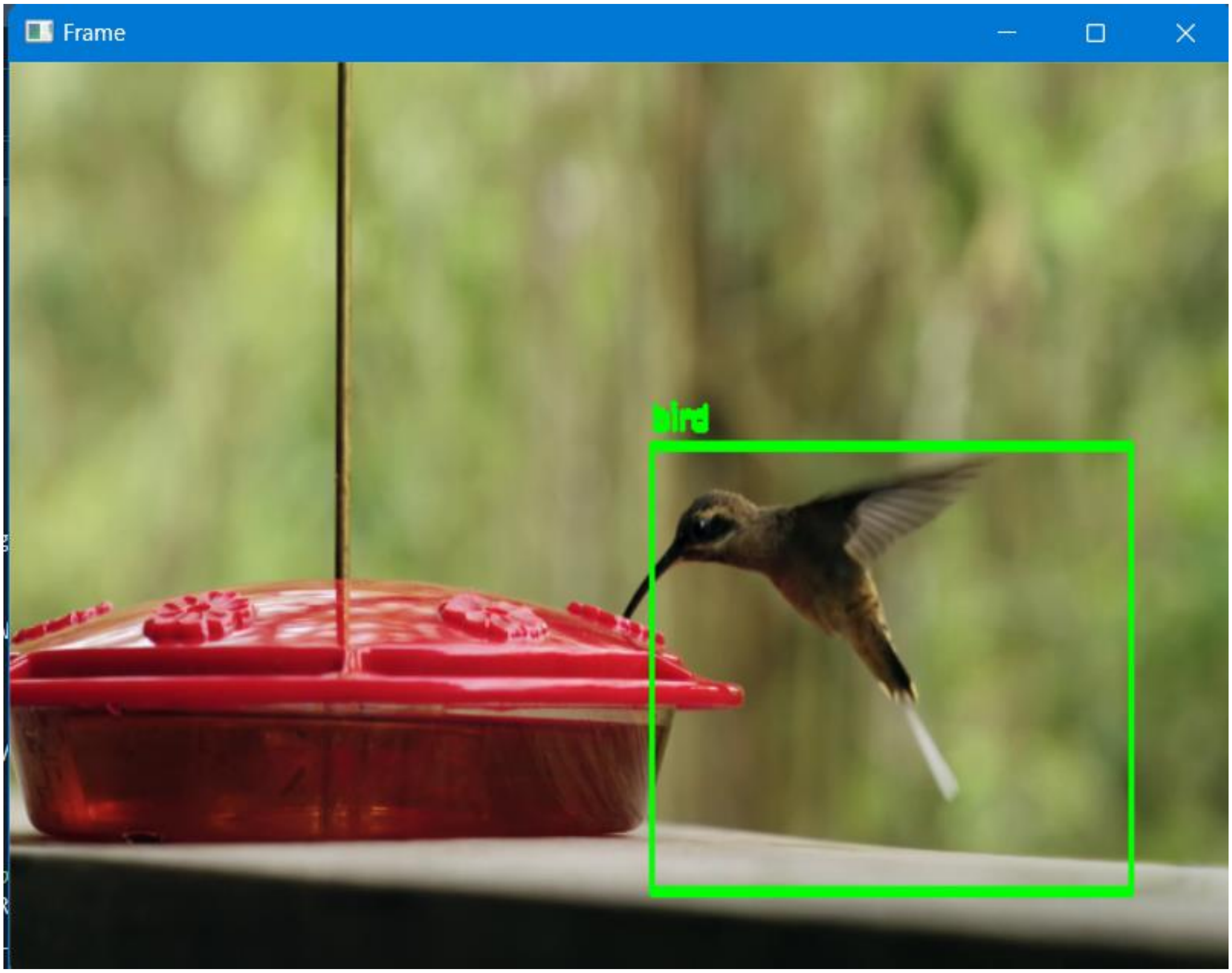


```python
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # 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, 'bird', (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # 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
        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)

            # Draw updated bounding boxes
            cv2.rectangle(frame, (startX, startY),
                          (endX, endY), (0, 255, 0), 2)
```



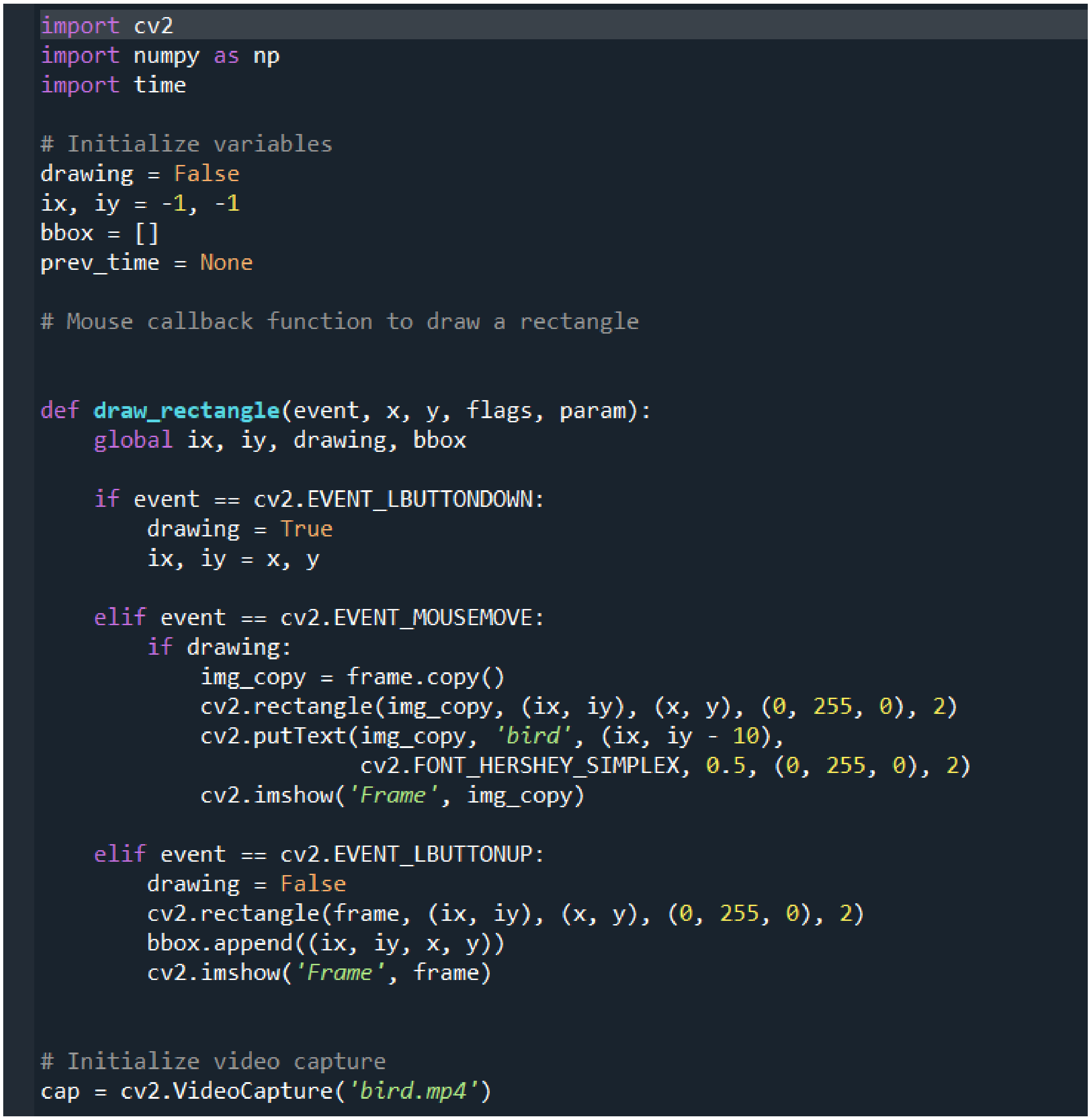
```python
            cv2.putText(frame, 'bird', (startX, startY - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # 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:

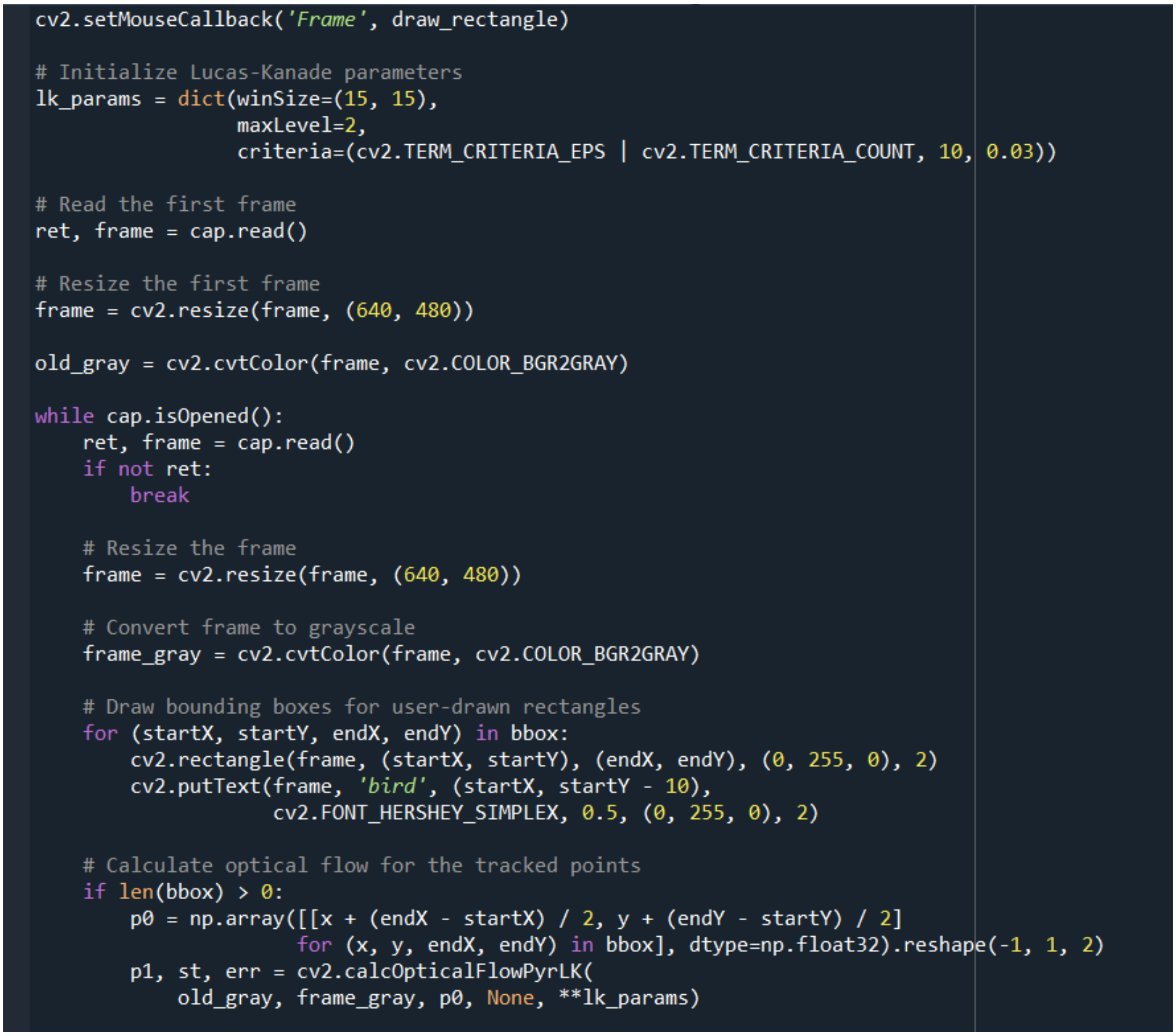


Exercise 2: Object Speed Estimation

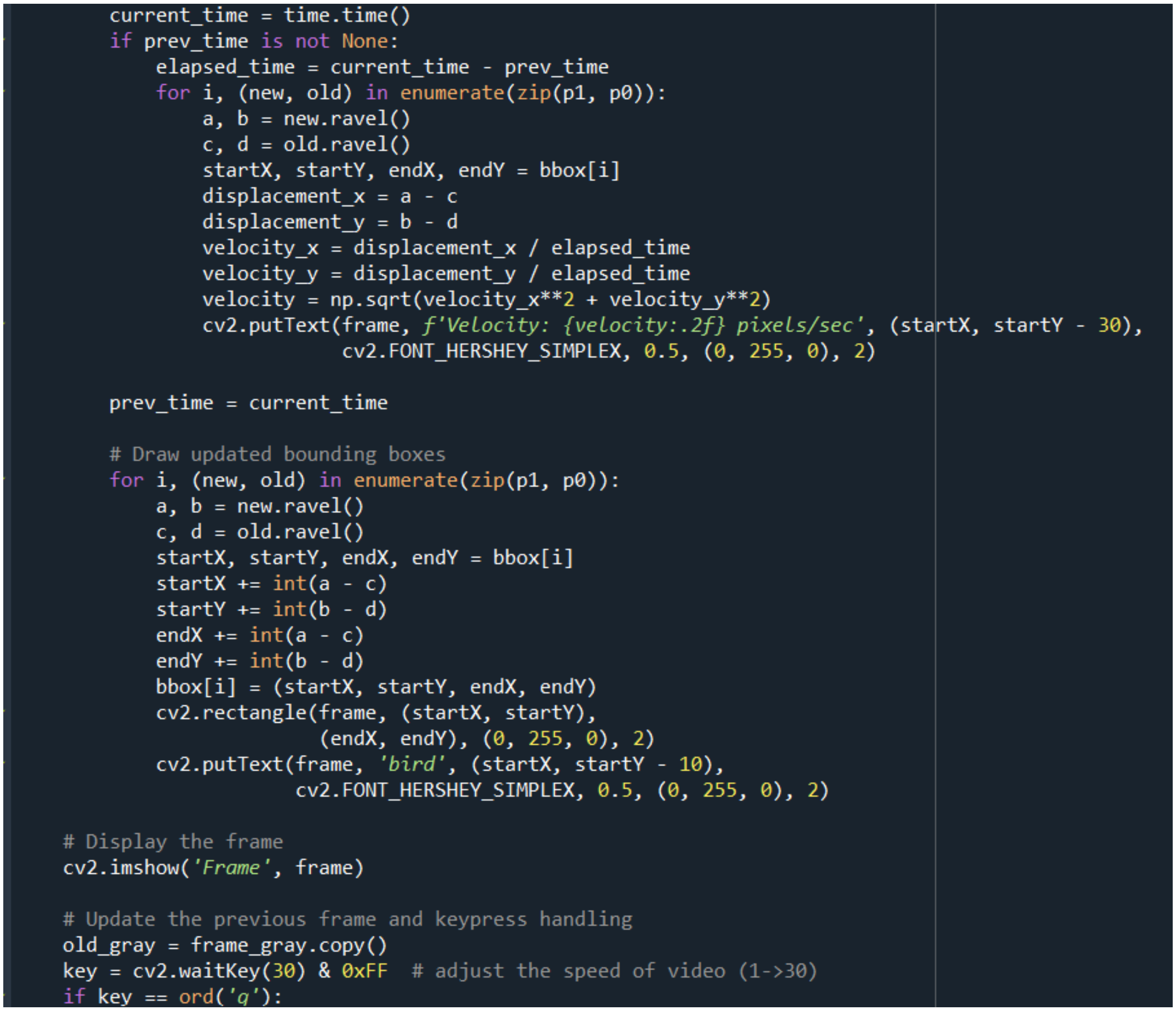
Source Code:



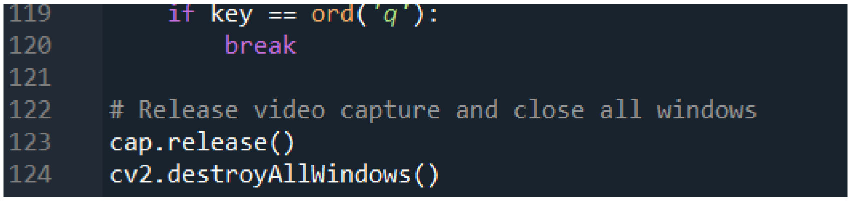
```python
import cv2
import numpy as np
import time

# 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, 'bird', (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('bird.mp4')
```



```python
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, 'bird', (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # 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)
```

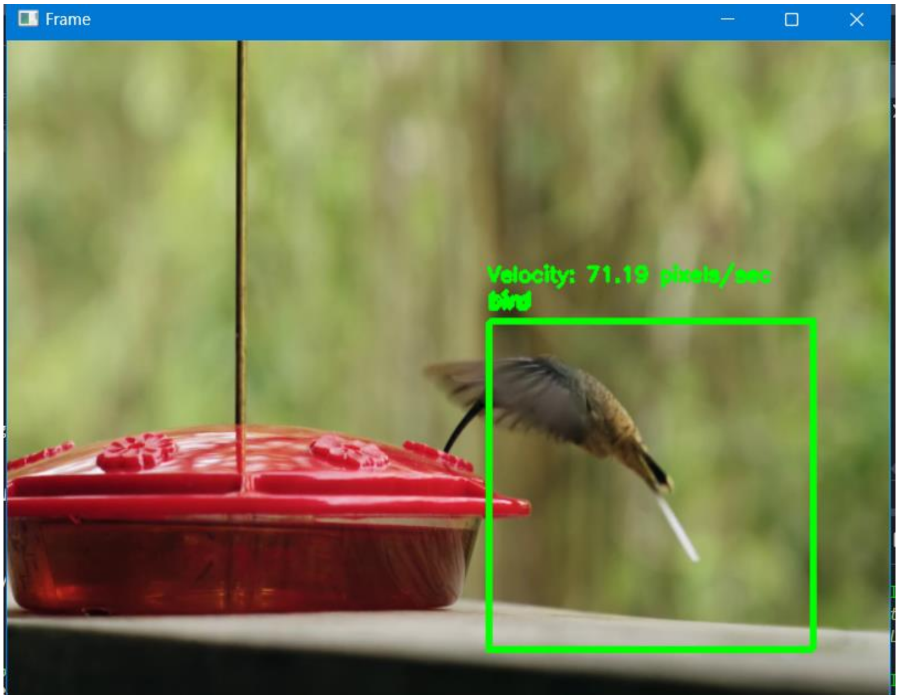


```
        current_time = time.time()
        if prev_time is not None:
            elapsed_time = current_time - prev_time
            for i, (new, old) in enumerate(zip(p1, p0)):
                a, b = new.ravel()
                c, d = old.ravel()
                startX, startY, endX, endY = bbox[i]
                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, 'bird', (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'):
```



```
    if key == ord('q'):
        break

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

Result:


Frame
Velocity: 71.19 pixels/sec
bird