Day 14: Polishing the Color-Based Object Tracker

Outcomes:

- Make the tracker **more stable**
- Reduce jitter and sudden jumps
- Visualize motion clearly
- Improve usability and presentation

Without polishing:

- Bounding box jitters
- Center jumps frame-to-frame
- Noise causes false movement
- Looks “unfinished”

We’ll fix these **without ML**.

Smoothing the Center Point

Instead of using only the current center, **average recent positions**.

```
from collections import deque
trail_points = deque(maxlen=20)
```

Add each center:

```
points.append((cx, cy))
```

Draw motion trail:

```
for i in range(1, len(points)):
    cv.line(frame, points[i-1], points[i], (0,255,255), 2)
```

`Deque()` Function

deque stands for **Double-Ended Queue**.

It is a built-in Python data structure provided by the ‘collections’ module.

Consider the line:

trail_points = deque(maxlen=30)

This creates a deque that can store 30 items at max, when the 31st item is added, the oldest item is automatically removed.

Data Type of `trail_points`: `collections.deque`

`deque` is a special list-list structure.

Area Threshold Tuning

Ignore sudden small detections:

```
if cv.contourArea(largest) > 800:
    # track
```

This removes noise-induced jumps.

FPS counter

```
import time


fps = 1 / (curr_time - prev_time)
prev_time = curr_time


cv.putText(frame, f"FPS: {int(fps)}", (10,30),
           cv.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
```

Freeze last position (no sudden disappear)

If object disappears briefly:

- Keep last known position
- Don't reset instantly

```
if largest is not None:
    last_center = (cx, cy)
else:
    cx, cy = last_center
```

pipeline: Conversion → Mask → Morphology → Contouring → Largest Contour → Smoothed Center → Trail + box + FPS

14-polishedObjectTracker.py

```python
import cv2 as cv
import numpy as np
from collections import deque
import time

cap = cv.VideoCapture(0)
if not cap.isOpened():
    print("Camera not accessible")
    exit()

cv.namedWindow('Trackbars')
cv.imshow("Trackbars", np.zeros((90, 400), dtype=np.uint8))

def nothing(x):
    pass

cv.createTrackbar("LH", "Trackbars", 0, 179, nothing)
cv.createTrackbar("LS", "Trackbars", 0, 255, nothing)
cv.createTrackbar("LV", "Trackbars", 0, 255, nothing)
cv.createTrackbar("UH", "Trackbars", 179, 179, nothing)
cv.createTrackbar("US", "Trackbars", 255, 255, nothing)
cv.createTrackbar("UV", "Trackbars", 255, 255, nothing)

kernel = np.ones((5,5), np.uint8)

trail_points = deque(maxlen=30)
center_points = deque(maxlen=5)
```

```
prev_time = time.time()

while True:
    ret, frame = cap.read()
    if not ret:
        print("An error occurred")
        break

    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    lh = cv.getTrackbarPos("LH", "Trackbars")
    ls = cv.getTrackbarPos("LS", "Trackbars")
    lv = cv.getTrackbarPos("LV", "Trackbars")
    uh = cv.getTrackbarPos("UH", "Trackbars")
    us = cv.getTrackbarPos("US", "Trackbars")
    uv = cv.getTrackbarPos("UV", "Trackbars")

    lower_bound = np.array([lh, ls, lv])
    upper_bound = np.array([uh, us, uv])

    mask = cv.inRange(hsv, lower_bound, upper_bound)
    opened = cv.morphologyEx(mask, cv.MORPH_OPEN, kernel)

    contours, hierarchy = cv.findContours(opened,
cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    largest = None
```

```
    max_area = 0

    for cnt in contours:
        if cv.contourArea(cnt) > max_area:
            largest = cnt
            max_area = cv.contourArea(cnt)

    if largest is not None:
        x, y, w, h = cv.boundingRect(largest)

        cx = x + w // 2
        cy = y + h // 2

        center_points.append((cx, cy))

        avg_x = int(np.mean([p[0] for p in center_points]))
        avg_y = int(np.mean([p[1] for p in center_points]))

        trail_points.append((avg_x, avg_y))

        cv.rectangle(frame, (x, y), (x + w, y + h), (0, 255,
0), 2)

        cv.circle(frame, (avg_x, avg_y), 5, (0, 0, 255), -1)

        for i in range(1, len(trail_points)):
            cv.line(frame, trail_points[i - 1],
trail_points[i], (255, 0, 0), 2)
```

```
    curr_time = time.time()
    fps = 1 / (curr_time - prev_time)
    prev_time = curr_time
    cv.putText(frame, f"FPS: {int(fps)}", (10,30),
cv.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)

    cv.imshow("Frame", frame)
    cv.imshow("Mask", opened)

    if cv.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```