객체 추적 알고리즘

최준혁2

In [ ]:

```
import cv2
import numpy as np

video_path = './data/slow_traffic_small.mp4'

def MeanShift(path):
    # init rectangle for mean shift tracking
    track_window = None #temp for object loc data
    roi_hist = None #temp for histogram
    term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

    cap = cv2.VideoCapture("./data/slow_traffic_small.mp4")

    ret, frame = cap.read()
    # print(ret, frame)


    x, y, w, h = cv2.selectROI("selectROI", frame, False, False)

    #calculate init histogram of tracked obj
    roi = frame[y:y+h, x:x+w]

    # cv2.imshow("roi test", roi)
    # cv2.waitKey(0)

    hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
    roi_hist = cv2.calcHist([hsv_roi], [0], None, [180], [0, 100])
    cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)

    # set init window for tracked obj
    track_window = (x, y, w, h)

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

        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([hsv],[0],roi_hist, [0, 180], 1)

        _, track_window = cv2.meanShift(dst, track_window, term_crit)

        x, y, w, h = track_window
        print("추적 결과 좌표", x, y, w, h)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0),2)
        cv2.imshow("MeanShift Tracking",frame)

        if cv2.waitKey(30) & 0xFF==ord('q'):
            exit()
    cap.release()
    cv2.destroyAllWindows()

def Kalman(path):
    kalman = cv2.KalmanFilter(4,2)
    kalman.measurementMatrix = np.array([[1,0,0,0],
                                         [0,1,0,0]], np.float32)

    kalman.transitionMatrix = np.array([[1, 0, 0, 0],
                                        [0, 1, 0, 1],
                                        [0, 0, 1, 0],
                                        [0, 0, 0, 1]], np.float32)

    kalman.processNoiseCov = np.array([[1, 0, 0, 0],
                                     [0, 1, 0, 0],
                                     [0, 0, 1, 0],
                                     [0, 0, 0, 1]], np.float32) * 0.05


    cap = cv2.VideoCapture(path)
    bbox = cv2.selectROI("Select Object", frame, False, False)
    kalman.statePre = np.array([[bbox[0]],
                                [bbox[1]],
                                [0],
                                [0]], np.float32)

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

        if not ret:
            break
        kalman.correct(np.array([[np.float32(bbox[0] + bbox[2] / 2)],
                                 [np.float32(bbox[1] + bbox[3] / 2)]]))
        kalman.predict()
        predicted_bbox = tuple(map(int, kalman.statePost[:2, 0]))
        cv2.rectangle(frame, (predicted_bbox[0] - bbox[2] // 2, predicted_bbox[1] -
                      (predicted_bbox[0] + bbox[2] //2 , predicted_bbox[1] + bbox[3
                      (0, 255, 0), 2)

        cv2.imshow("Kalman Filter Tracking", frame)

        if cv2.waitKey(30) & 0xFF == ord('q'):
            break


    cap.release()
    cv2.destroyAllWindows()

def SIFT(path):
    limited = (input("Set max limitation ? (y/n)") == 'y')
    cap = cv2.VideoCapture(path)

    sift = cv2.SIFT_create()
    MAX_KEYPOINTS = 100

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

        if not ret:
            break

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

        keypoints, descriptors = sift.detectAndCompute(gray, None)

        if (len(keypoints) > MAX_KEYPOINTS) & limited:
            keypoints = sorted(keypoints, key= lambda x: -x.response)[:MAX_KEYPOINT


        frame = cv2.drawKeypoints(frame, keypoints, None, flags=cv2.DRAW_MATCHES_FL

        cv2.imshow("SIFT", frame)

        if cv2.waitKey(30) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

def ORB(path):
    cap = cv2.VideoCapture(path)

    orb = cv2.ORB_create()


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

        if not ret:
            break

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

        keypoints = orb.detect(gray, None)
        frame = cv2.drawKeypoints(frame, keypoints, None, color=(0, 255, 0), flags=

        cv2.imshow("ORB", frame)

        if cv2.waitKey(30) & 0xFF == ord('q'):
            break


    cap.release()
    cv2.destroyAllWindows()


n = input('''Choose tracking algo \n
          1. Mean-Shift\n
          2. Kalman Filter\n
          3. SIFT\n
          4. ORB\n''')
```

```
if n == '1':
    MeanShift(video_path)
elif n=='2':
    Kalman(video_path)
elif n=='3':
    SIFT(video_path)
elif n=='4':
    ORB(video_path)
```