limhpone / computervision-final-prep

Code Issues Pull requests Actions Projects Wiki Security In

computervision-final-prep / lab / Lab 09 (Tracking)-20251128
/ 9.1_simple_object_tracking_with_color.py

limhpone lab 9-obj tracking 267755b · 2 hours ago

118 lines (96 loc) · 3.93 KB

Code Blame Raw

```python
# import the necessary packages
from collections import deque
from imutils.video import VideoStream
import numpy as np
import argparse
import cv2
import imutils
import time


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video",
    help="path to the (optional) video file")
ap.add_argument("-b", "--buffer", type=int, default=64,
    help="max buffer size")
args = vars(ap.parse_args())

# define the lower and upper boundaries of the "green"/"yellow" objects in the HSV color space

greenLower = (29, 86, 6)
greenUpper = (64, 255, 255)

yellowLower = (15, 80, 80)
yellowUpper = (35, 255, 255)

# initialize the list of tracked points
pts = deque(maxlen=args["buffer"])

# Handle the video stream from either webcam or video file
use_file = bool(args.get("video"))
vs = cv2.VideoCapture(args["video"]) if use_file else VideoStream(src=0).start()
```

```python

use_file = bool(args.get("video"))
if use_file:
    vs = cv2.VideoCapture(args["video"])
else:
    vs = cv2.VideoCapture(0, cv2.CAP_ANY)
    vs.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*"MJPG"))
    vs.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    vs.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    vs.set(cv2.CAP_PROP_FPS, 30)

# allow the camera or video file to warm up
time.sleep(1.0)

fail_count = 0
max_fail = 30

try:
    # LOOP OVER THE FRAMES OF THE VIDEO
    while True:
        # grab the current frame
        if use_file:
            grabbed, frame = vs.read()
            if not grabbed:
                break
        else:
            grabbed, frame = vs.read()
            if not grabbed or frame is None:
                fail_count += 1
                if fail_count >= max_fail:
                    break
                time.sleep(0.02)
                continue
            fail_count = 0

        #resize the frame, blur it, and convert it to the HSV color space
        frame = imutils.resize(frame, width=600)
        blurred = cv2.GaussianBlur(frame, (11, 11), 0)
        hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

        # construct a mask for the color "yellow", followed by a series of dilations and erosi
        mask = cv2.inRange(hsv, yellowLower, yellowUpper)
        mask = cv2.erode(mask, None, iterations=2)
        mask = cv2.dilate(mask, None, iterations=2)

        # find contours in the mask
        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)

```

```
            center = None

            # only proceed if at least one contour was found

            if len(cnts) > 0:
                # find the largest contour in the mask
                c = max(cnts, key=cv2.contourArea)
                # determine the radius and center of the enclosing circle
                ((x, y), radius) = cv2.minEnclosingCircle(c)

                M = cv2.moments(c)
                if M["m00"] != 0:
                    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
                if radius > 10:
                    cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
                    if center is not None:
                        cv2.circle(frame, center, 5, (0, 0, 255), -1)

            # update the points queue
            pts.appendleft(center)

            # loop over the set of tracked points
            for i in range(1, len(pts)):
                # if current or previous point is None, ignore!
                if pts[i - 1] is None or pts[i] is None:
                    continue

                # compute the thickness of the points in line and draw the connecting lines
                thickness = int(np.sqrt(args["buffer"] / float(i + 1)) * 2.5)
                cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)

            cv2.imshow("Frame", frame)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
    finally:
        vs.release()
        cv2.destroyAllWindows()
```