opencv / samples / dnn / openpose.py

berak samples/dnn: better errormsg in openpose.py 3 years ago

122 lines (102 loc) · 6.45 KB

Code Blame Raw

```
# To use Inference Engine backend, specify location of plugins:
# source /opt/intel/computer_vision_sdk/bin/setupvars.sh
import cv2 as cv
import numpy as np
import argparse

parser = argparse.ArgumentParser(
        description='This script is used to demonstrate OpenPose human pose estimation network '
                    'from https://github.com/CMU-Perceptual-Computing-Lab/openpose project using OpenCV. '
                    'The sample and model are simplified and could be used for a single person on the frame.')
parser.add_argument('--input', help='Path to image or video. Skip to capture frames from camera')
parser.add_argument('--proto', help='Path to .prototxt')
parser.add_argument('--model', help='Path to .caffemodel')
parser.add_argument('--dataset', help='Specify what kind of model was trained. '
                                      'It could be (COCO, MPI, HAND) depends on dataset.')
parser.add_argument('--thr', default=0.1, type=float, help='Threshold value for pose parts heat map')
parser.add_argument('--width', default=368, type=int, help='Resize input to specific width.')
parser.add_argument('--height', default=368, type=int, help='Resize input to specific height.')
parser.add_argument('--scale', default=0.003922, type=float, help='Scale for blob.')

args = parser.parse_args()

if args.dataset == 'COCO':
    BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
                   "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
                   "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "REye": 14,
                   "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }

    POSE_PAIRS = [ ["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder", "RElbow"],
                   ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
                   ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAnkle"], ["Neck", "LHip"],
                   ["LHip", "LKnee"], ["LKnee", "LAnkle"], ["Neck", "Nose"], ["Nose", "REye"],
                   ["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEar"] ]
elif args.dataset == 'MPI':
    BODY_PARTS = { "Head": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
                   "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
                   "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "Chest": 14,
                   "Background": 15 }

    POSE_PAIRS = [ ["Head", "Neck"], ["Neck", "RShoulder"], ["RShoulder", "RElbow"],
                   ["RElbow", "RWrist"], ["Neck", "LShoulder"], ["LShoulder", "LElbow"],
                   ["LElbow", "LWrist"], ["Neck", "Chest"], ["Chest", "RHip"], ["RHip", "RKnee"],
                   ["RKnee", "RAnkle"], ["Chest", "LHip"], ["LHip", "LKnee"], ["LKnee", "LAnkle"] ]
elif args.dataset == 'HAND':
    BODY_PARTS = { "Wrist": 0,
                   "ThumbMetacarpal": 1, "ThumbProximal": 2, "ThumbMiddle": 3, "ThumbDistal": 4,
                   "IndexFingerMetacarpal": 5, "IndexFingerProximal": 6, "IndexFingerMiddle": 7, "IndexFingerDistal": 8,
                   "MiddleFingerMetacarpal": 9, "MiddleFingerProximal": 10, "MiddleFingerMiddle": 11, "MiddleFingerDistal": 12,
                   "RingFingerMetacarpal": 13, "RingFingerProximal": 14, "RingFingerMiddle": 15, "RingFingerDistal": 16,
                   "LittleFingerMetacarpal": 17, "LittleFingerProximal": 18, "LittleFingerMiddle": 19, "LittleFingerDistal": 20
                 }

    POSE_PAIRS = [ ["Wrist", "ThumbMetacarpal"], ["ThumbMetacarpal", "ThumbProximal"],
                   ["ThumbProximal", "ThumbMiddle"], ["ThumbMiddle", "ThumbDistal"],
```

```
                    ["Wrist", "IndexFingerMetacarpal"], ["IndexFingerMetacarpal", "IndexFingerProximal"],
                    ["IndexFingerProximal", "IndexFingerMiddle"], ["IndexFingerMiddle", "IndexFingerDistal"],
                    ["Wrist", "MiddleFingerMetacarpal"], ["MiddleFingerMetacarpal", "MiddleFingerProximal"],
                    ["MiddleFingerProximal", "MiddleFingerMiddle"], ["MiddleFingerMiddle", "MiddleFingerDistal"],
                    ["Wrist", "RingFingerMetacarpal"], ["RingFingerMetacarpal", "RingFingerProximal"],
                    ["RingFingerProximal", "RingFingerMiddle"], ["RingFingerMiddle", "RingFingerDistal"],
                    ["Wrist", "LittleFingerMetacarpal"], ["LittleFingerMetacarpal", "LittleFingerProximal"],
                    ["LittleFingerProximal", "LittleFingerMiddle"], ["LittleFingerMiddle", "LittleFingerDistal"] ]
else:
    raise(Exception("you need to specify either 'COCO', 'MPI', or 'Hand' in args.dataset"))

inWidth = args.width
inHeight = args.height
inScale = args.scale

net = cv.dnn.readNet(cv.samples.findFile(args.proto), cv.samples.findFile(args.model))

cap = cv.VideoCapture(args.input if args.input else 0)

while cv.waitKey(1) < 0:
    hasFrame, frame = cap.read()
    if not hasFrame:
        cv.waitKey()
        break

    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]
    inp = cv.dnn.blobFromImage(frame, inScale, (inWidth, inHeight),
                               (0, 0, 0), swapRB=False, crop=False)
    net.setInput(inp)
    out = net.forward()

    assert(len(BODY_PARTS) <= out.shape[1])

    points = []
    for i in range(len(BODY_PARTS)):
        # Slice heatmap of corresponding body's part.
        heatMap = out[0, i, :, :]

        # Originally, we try to find all the local maximums. To simplify a sample
        # we just find a global one. However only a single pose at the same time
        # could be detected this way.
        _, conf, _, point = cv.minMaxLoc(heatMap)
        x = (frameWidth * point[0]) / out.shape[3]
        y = (frameHeight * point[1]) / out.shape[2]

        # Add a point if it's confidence is higher than threshold.
        points.append((int(x), int(y)) if conf > args.thr else None)

    for pair in POSE_PAIRS:
        partFrom = pair[0]
        partTo = pair[1]
        assert(partFrom in BODY_PARTS)
        assert(partTo in BODY_PARTS)

        idFrom = BODY_PARTS[partFrom]
        idTo = BODY_PARTS[partTo]

        if points[idFrom] and points[idTo]:
            cv.line(frame, points[idFrom], points[idTo], (0, 255, 0), 3)
            cv.ellipse(frame, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)
            cv.ellipse(frame, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv.FILLED)

    t, _ = net.getPerfProfile()
    freq = cv.getTickFrequency() / 1000
    cv.putText(frame, '%.2fms' % (t / freq), (10, 20), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))

    cv.imshow('OpenPose using OpenCV', frame)
```