Session 5 - Human Pose Estimation

Due Sep 5 by 11:59pm **Points** 3,000 **Submitting** a text entry box or a website url
Available Aug 22 at 10am - Sep 5 at 11:59pm 15 days

This assignment was locked Sep 5 at 11:59pm.

Session 5 - Monocular Human Pose Estimation

AND ONNX Models

- [Awesome HPE Reference (https://github.com/wangzheallen/awesome-human-pose-estimation)](https://github.com/wangzheallen/awesome-human-pose-estimation)

What is human pose estimation?

Human pose estimation is the process of estimating the configuration of the body (pose) from a single, typically monocular, image.

It can be applied to many applications such as action/activity recognition, action detection, human tracking, in movies and animation, virtual reality, human-computer interaction, video surveillance, medical assistance, self-driving, sports motion analysis, etc.

Key Problems in Pose Estimation

[[SOURCE - Disney Research (https://www.cs.ubc.ca/~lsigal/Publications/SigalEncyclopediaCVdraft.pdf)](https://www.cs.ubc.ca/~lsigal/Publications/SigalEncyclopediaCVdraft.pdf)]Human pose estimation is one of the key problems in computer vision that has been studied for well over 15 years. The reason for its importance is the abundance of applications that can benefit from such technology.

For example, human pose estimation allows for higher-level reasoning in the context of human-computer interaction and activity recognition; it is also one of the basic building blocks for marker-less motion capture (MoCap) technology.

MoCap technology is useful for applications ranging from character animation to clinical analysis of gait pathologies.

Despite many years of research, however, pose estimation remains a very difficult and still largely unsolved problem.

Among the most significant challenges are:

(1) variability of human visual appearance in images,

(2) variability in lighting conditions,

(3) variability in the human physique,

(4) partial occlusions due to self-articulation and layering of objects in the scene,

(5) the complexity of human skeletal structure,

(6) the high dimensionality of the pose, and

(7) the loss of 3d information that results from observing the pose from 2d planar image projections

complex_poses.jpg

HPE Method Categories

1. generative and discriminative (3D Single Person)
    1. generative - human body model-based
    2. discriminative - human body model-free
2. top-down and bottom-up (Multi-Person)
    1. top-down - from high-level abstraction to low-level pixel evidence
    2. bottom-up - from low-level pixel evidence to high-level abstraction

3. regression and detection based (Single Person)
    1. regression - directly mapping from input images to body joint points
    2. detection - generating intermediate image patches or heatmaps of join-locations
4. one-stage and multi-stage:
    1. one-stage - end-to-end training
    2. multi-stage - stage-by-stage training

These divisions are applicable in general to other problem areas as well, so let's take a deeper look at them

Generative vs Discriminative

The main difference between generative and discriminative methods is whether a method uses human body models or not. Based on the different representations of human body models, generative methods can be processed in different ways such as prior beliefs about the structure of the body model, geometrically projection from different views to 2D or 3D space, high-dimensional parametric space optimization in regression manners.

Discriminative methods directly learn a mapping from input sources to human pose space (learning-based) or search in existing examples (example-based) without using human

body models. Discriminative methods are usually faster than generative methods but may have less robustness for poses never trained with.

Top-down vs Bottom-up

For multi-person pose estimation, HPE methods can generally be classified as top-down and bottom-up methods according to the starting point of the prediction: high-level abstraction or low-level pixel evidence. Topdown methods start from high-level abstraction to first detect persons and generate the person locations in bounding boxes. Then pose estimation is conducted for each person.

In contrast, bottom-up methods first predict all body parts of every person in the input image and then group them either by human body model fitting or other algorithms. Note that body parts could be joints, limbs, or small template patches depending on different methods. With an increased number of people in an image, the computation cost of top-down methods significantly increases, while keeps stable for bottom-up methods. However, if there are some people with a large overlap, bottom-up methods face challenges to group corresponding body parts.

Regression vs Detection

Based on the different problem formulations, deep learning-based human pose estimation methods can be split into regression-based or detection-based methods. The regression-based methods directly map the input image to the coordinates of body joints or the parameters of human body models. The detection-based methods treat the body parts as detection targets based on two widely used representations: image patches and heatmaps of joint locations.

Direct mapping from images to joint coordinates is very difficult since it is a highly nonlinear problem, while small-region representation provides dense pixel information with stronger robustness. Compared to the original image size, the detected results of small-region representation limit the accuracy of the final joint coordinates.

One-stage vs Multi-stage

The deep learning-based one-stage methods aim to map the input image to human poses by employing end-to-end networks, while multi-stage methods usually predict human pose in multiple stages and are accompanied by intermediate supervision. For example, some multiperson pose estimation methods first detect the locations of people and then estimate the human pose for each detected person.

Other 3D human pose estimation methods first predict joint locations in the 2D surface, then extend them to 3D space. The training of one-stage methods is easier than multi-stage methods, but with less intermediate constraints.

Human Body Models

human_body_models.jpg

3d-keypoints-human-pose-estimation.png

HPE Benchmarks & Datasets

Summary of 2D single person HPE

model_summary.png

Summary of 2D multi-person HPE

multi_model_summary.png

Datasets

datasets.jpg

HPE Evaluation Metrics

evaluation_metrics.png

OKS = Object Keypoint Similarity

PCK = Percentage of Correct Keypoint

PCKh = The PCKh performance metric is the percentage of joints with predicted locations that are no further than half of the head segment length from the ground truth.

Most important work in HPE

REF (https://theaisummer.com/Human-Pose-Estimation/)

OpenPose

OpenPose (https://arxiv.org/abs/1812.08008)

openpose.png

OpenPose is the most popular open-source tool for body, foot, hand, and facial keypoint detection. It makes use of Part Affinity Fields (PAFs), a set of 2D vector fields to encode the location and orientation of limbs over the image domain.

As shown in the image **F** is passed through several convolutional layers to generate the PAFs (**L**) and confidence maps **S** for every joint location. The process is repeated for some iterations and the network refines its predictions at every stage. OpenPose is widely used in research projects.

openpose_example.png

A simple yet effective baseline

[A simple yet effective baseline for 3d human pose estimation (https://arxiv.org/pdf/1705.03098.pdf)](https://arxiv.org/pdf/1705.03098.pdf)

3d-pose-estimation.png

In this work, the authors implemented a lightweight and fast network able to process 300 frames per second. After extracting 2d joint location, due to the low dimensionality of 2d space, they use a simple neural network that has a small number of parameters, to estimate the coordinates of joints in 3d space.

DensePose

[DensePose (http://densepose.org/)](http://densepose.org/)

densepose.png

DensePose adopts the architecture of MaskRCNN with the feature pyramid network features, and ROI-Align pooling so as to obtain dense part labels and coordinates with each of the selected regions. The method uses a fully convolutional network on top of ROI-pooling that is entirely devoted to generating per-pixel classification results for selection of surface part and regressing local coordinates within each part.

Simple baseline for HPE and Tracking

[PAPER (https://arxiv.org/pdf/1804.06208.pdf)](https://arxiv.org/pdf/1804.06208.pdf)

simple_pose.png

As other approaches have become complex,
this work aimed to ease the problem by asking a question,
"how good could a simple method be?"

This approach involves a few deconvolutional layers added on a backbone network, ResNet.
This approach adds a few deconvolutional layers over the last convolution stage in the ResNet, called C_5.

By default, 3 Deconv layers with BN and ReLU as used.
Each layer has 256 filters with 4x4 kernels.
The stride is 2.

A 1x1 convolutional layer is added at the last to generate predicted heatmaps for all k key points.
MSE is used as the loss function.

The targeted heatmap for join k is generated by applying a 2D gaussian centered on the k^{th} joint's ground truth location.

simple_pose_bench.png

Open Neural Network Exchange or ONNX

ONNX is an open format built to represent machine learning models.
ONNX defines a common set of operators - the building blocks of machine learning and deep learning models
- and a common file format to enable AI developers to use models with
a variety of frameworks, tools, runtimes, and compilers.

ONNX Runtime is a performance-focused engine for ONNX models,
which inferences efficiently across multiple platforms and hardware (Windows, Linux, and Mac and on both CPUs and GPUs).

ONNX Runtime has proved to considerably increase performance over multiple models

deploy.png

frameworks-1.png

Practice ONNX:

1. **PyTorch YoloV4 ONNX (https://github.com/Tianxiaomo/pytorch-YOLOv4)**
2. **AlexNet to ONNX (https://michhar.github.io/convert-pytorch-onnx/)**
3. **ResNet to ONNX (https://colab.research.google.com/github/bentoml/gallery/blob/master/onnx/resnet50/res**
4. **SuperResolution to ONNX (https://pytorch.org/tutorials/advanced/super_resolution_with_onnxruntime.html)**

We will cover **[ONNX.js](https://microsoft.github.io/onnxjs-demo/#/) (https://microsoft.github.io/onnxjs-demo/#/)** in the next session, so make sure that you know what ONNX is, and how to convert models to ONNX format.

Assignment

1. You are implementing "**Simple Baseline for HPE and tracking (https://github.com/Microsoft/human-pose-estimation.pytorch)**". Read the **paper (https://arxiv.org/pdf/1804.06208.pdf)** and write a detailed readme file describing the model architecture as well as the JointsMSELoss class. - 1000 pts
2. Download the smallest **model (https://onedrive.live.com/?authkey=%21AFkTgCsr3CT9%2D%5FA&id=56B9F9C97F261712%2110709&cid=56B9F9C97** and upload to Lambda for HPE detection - 1000 pts
3. Make sure to draw the points on the image, as well as connect the joints in the right fashion. - 1000 pts

OR

1. 1 application of human pose gesture control (zooming a chrome browser page or opening/closing an application). - 3000 pts

VIDEO