TesseTrack: End-to-End Learnable Multi-Person Articulated 3D Pose Tracking Supplementary Material

N Dinesh Reddy[1] Laurent Guigues[2*] Leonid Pischulini[2*] Jayan Eledath[2] Srinivasa Narasimhan[1]
[1] Carnegie Mellon University [2] Amazon

1. Run Time

Compute specification. All of our experiments have been conducted using a machine with 8 v100 gpus and 96 cores CPU. The machine contained RAM memory of 748 GB for loading the data befor passing it into the GPU for processing. All through the training the algorithm has been implemented to maxout the utilization of GPU resource. We used 20 worker threads per GPU for distributed data loading in Pytorch.

Training time. We train TesseTrack on the haggling sequences of panoptic studio dataset for 10 epochs. The frame resolution used across all the datasets is 384x384 pixels. We rescale the images to the specified dimensions and stack them for the duration of the temporal window before passing to the GPU. TesseTrack has a batch size of 1 during training due to memory constraints. To train the network the total compute time on a machine with 8 v100 GPUs and 96 cores CPU is 36 hours. The bottleneck for compute was the GPU memory and can be trained faster with GPUs containing larger memory.

Evaluation latency. We have used multiple state-of-the-art datasets to evaluate the algorithm. For each dataset the TesseTrack detection framework has been fine-tuned to predict the locations of the center of the person. The ground truth for producing the center of each person has been computed by performing an RANSAC based triangulation of the center of the person on the detections from different views. This fine-tuning is computed at random time instances of the dataset. During evaluation the model is able to produce accurate 3D detections.

During evaluation, We use parameters similar to the training time specifications. The input images are still scaled to 384x384 images. We stack the frames from a temporal window before passing to the GPU. Table 2 and Table 1 show the time taken to run each module on a V100 GPU. Since TesseTrack is a Top-Down Approach, We observe that the test time increases with number of people detected by the detection network. The two segments heavily dependent on the number of people are the tesseract convolution and deconvolution layers. The run time is also dependent on the temporal window, Reducing the window size reduces the computed time. The tracking pipeline is the most time consuming step of the algorithm. This can be attributed to the MLP layer converting the tesseract to a single dimension feature vector.

During Inference, We use batch size of 1 and 5 for multi-view and monocular sequences respectively. We found that the run-time for monocular is 3.3 millisecond for each time instance and nearly 30 millisecond for multi-view inference. Based on requirement run-time can be improved with change in parameters.

	Time(in Seconds)			
	No. of People			
Module	1	3	5	7
Backbone	0.03	0.03	0.03	0.03
Detection	0.05	0.05	0.05	0.05
Tracking	0.14	0.2	0.26	0.32
3D Pose	0.03	0.09	0.15	0.21
Total	0.25	0.37	0.49	0.61

Table 1: Run time(in seconds for each module) on monocular input based on number of people with a constant window size of 5.

		Time(in Seconds)			
		No. of People			
Views	Module	1	3	5	7
3	Backbone	0.08	0.08	0.08	0.08
3	Detection	0.07	0.07	0.07	0.07
3	Tracking	0.17	0.23	0.37	0.49
3	3D Pose	0.06	0.18	0.3	0.42
5	Backbone	0.12	0.12	0.12	0.12
5	Detection	0.11	0.11	0.11	0.11
5	Tracking	0.19	0.25	0.37	0.49
5	3D Pose	0.06	0.18	0.3	0.42

Table 2: Run time(in seconds for each module) on muti-view input based on number of people with a constant window size of 5.

*Equal contribution

Comparisions. Table3 shows the per joint accuracy compared with other baselines. This is an extension of Table 1 from the main manuscript.

Inference Discrepancies: We want to point out that the model has been only trained on Panoptic studio and the 3D pose estimation has never be fine-tuned or retrained on any of the other datasets. The failure of 3D pose estimation on some frames of the Tagging sequence can be attributed to the person being very close or far from camera or the face keypoints. We believe finetuing the pose model in real-world situations similar to the Tagging sequence should boost performance on "In the Wild" Sequences.

2. Network Architecture

Backbone. We use HRnet[1] as the backbone in our model. The pre-final layer is passed through a convolutional layer to create a feature vector of 32. This feature dimension is consistently used in both the detection and the tracking layer.

Detection. We use 32 as the feature vector for our voxel grid. Inspired from [2], we pass the voxel feature map through three 3D-ResNet blocks with three 3D-maxpooling layers for 3D convolutions. Similarly it is passed through three 3D-ResNet blocks and unpooling layers during 3D deconvolution. The stride and kernel for all the 3D operations was 3.

Tracking. The tesseract follows similar dimensions to the detection voxel grid but with a temporal component and has a feature vector of 32. We apply three 4D convolutions on the tesseract to create the tesseract feature vector passed to the matching stage. We use a 4D kernel of 3x3x3x3 and a similar stride to compute the convolutions. This produces a tesseract feature of 32x1x8x8x8 feature vector. This feature is passed through a MLP to produce a 256 vector which is the input to the matching framework. The matching network solves the optimal assignment problem in differentiable fashion.

Pose estimation. Once the tracking is computed the merged pose is passed through multiple 4D convolutions and upsampled to produce the tesseract heatmap. Similar to the 4D convolutions the deconvolutions follow the same stride and kernel size.

Model	HI	FI	FT	FTGA	FTGL	FTDL
Neck	15.2	13.1	7.6	7.6	7.1	**6.9**
Head	15.6	12.9	8.0	7.9	7.4	**7.3**
Shou.	16.0	13.6	7.7	7.8	7.0	**6.8**
Elbow	16.4	14.5	8.0	8.5	**7.6**	7.7
Wrist	16.9	14.7	8.5	8.7	7.9	**8.1**
Hip	17.3	13.7	8.4	8.3	7.8	**7.5**
Knee	17.8	14.1	8.7	8.6	7.5	**7.0**
Ankle	16.1	14.3	8.6	8.8	7.9	**7.4**
Total	16.3	13.8	8.0	8.1	7.5	**7.3**

Table 3: MPJPE per-joint 3D pose reconstruction error for various design choices on the Panoptic dataset. These results represent a per-joint breakdown of MPJPE metric reported in Tab. 1 in the paper.

References

[1] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 2

[2] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. Voxelpose: Towards multi-camera 3d human pose estimation in wild environment. ECCV, 2020. 2