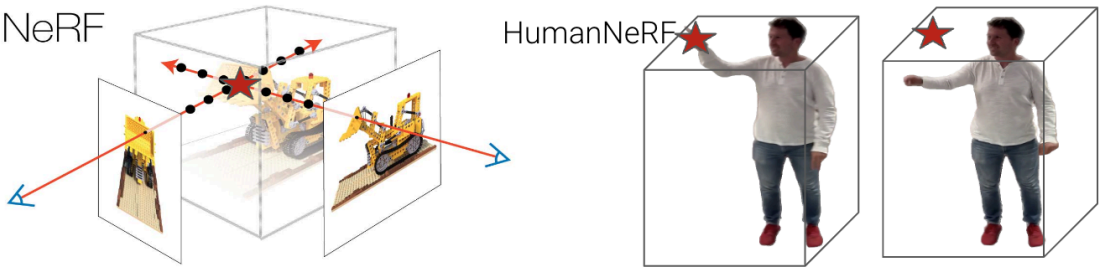
AllInOneAvatar

Team

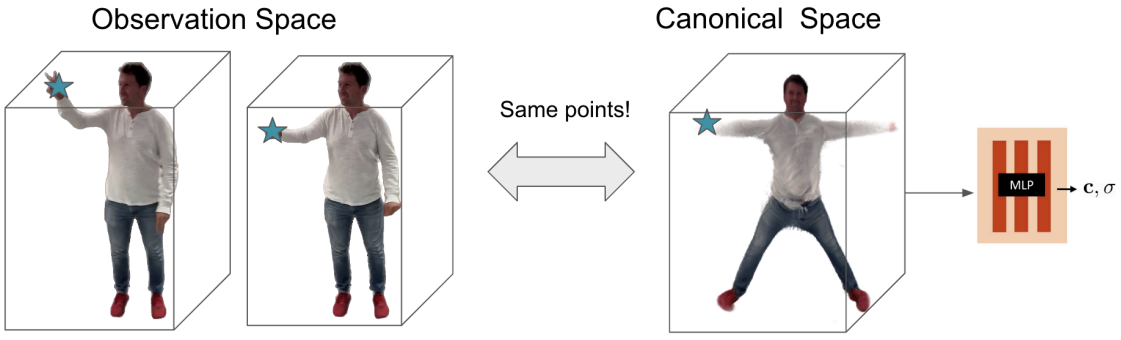
Boyu Zhang (bzhang99)
Senyang Jiang (senyangj)
Yining Mao (yiningm)

Background/Related Work

HumanNeRF vs. NeRF

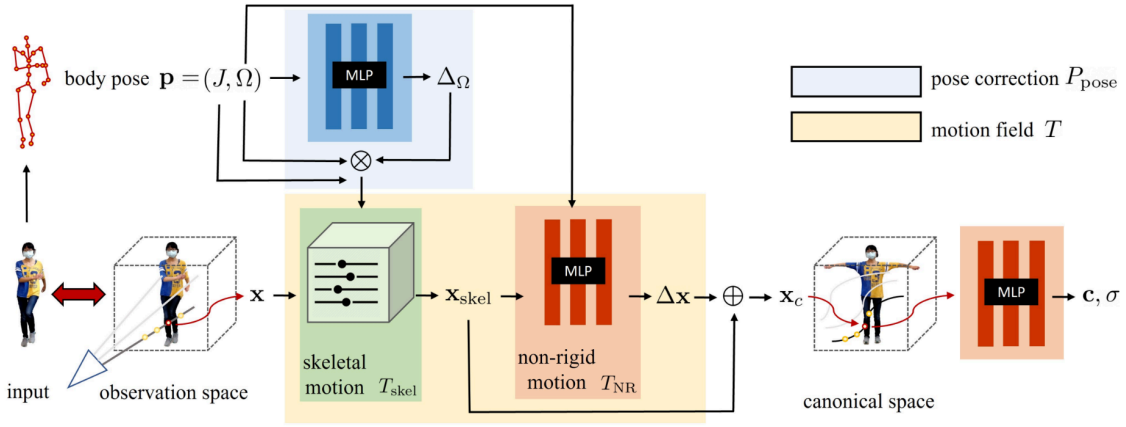


In NeRF, the scene is usually some static object. As a result, the color and density at a sample point in 3D space is constant at different times. However, in HumanNeRF we remove the static-object constraint, but instead the human in the video can move across different frames. To restore consistency across different frames, HumanNeRF introduces the concept of observation space and canonical space, as shown in the following figure.



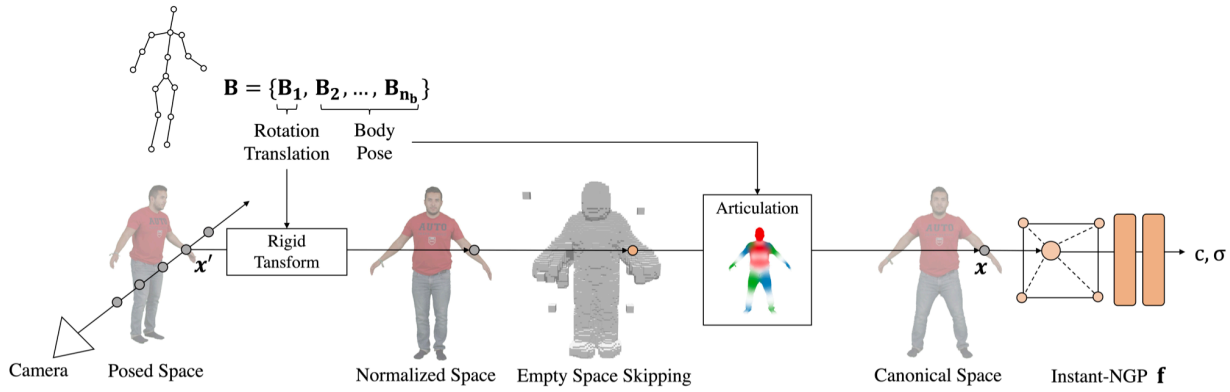
The input images are in observation space, which are inconsistent across time. By detecting the pose of the person in each frame, we can warp the person in observation space to canonical space, where the person's pose is fixed across times, hence restoring the static-object requirement of NeRF. Finally, we can train a NeRF representation of the human in canonical space.

Original HumanNeRF Architecture



The details in the architecture are not important, but notice that in addition to the MLP for training NeRF in canonical space, we have two more MLPs for learning the warp from observation space to canonical space. As a result, the training of the original HumanNeRF model is very slow, taking at least 10 hours for each dataset using multiple high-end GPUs.

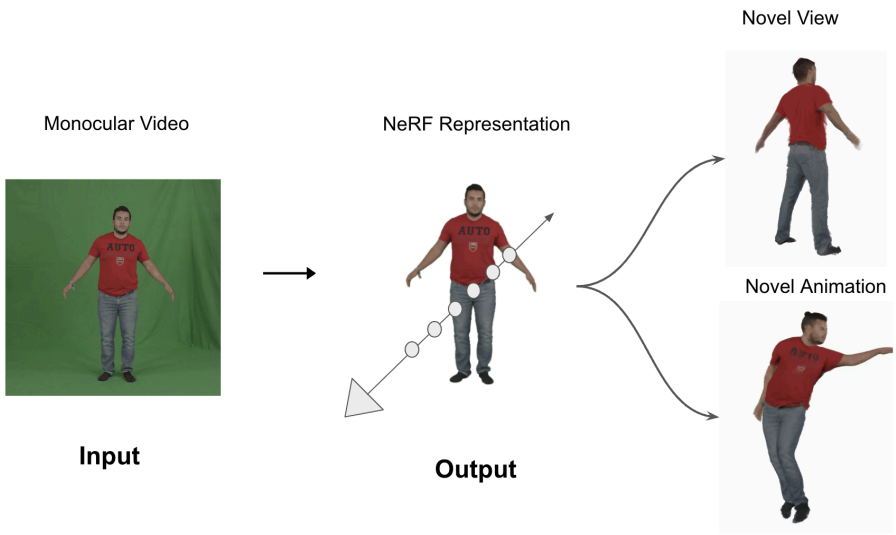
InstantAvatar Pipeline



InstantAvatar is the fastest HumanNeRF model we can find. It involves two major optimizations. Firstly, it replaces two MLPs used for warping in original HumanNeRF with a more efficient articulation module, Fast-SNARF. Secondly, it replaces the NeRF MLP with instant-NGP, which is much faster for training and inference.

Problem Setup

Inputs/Outputs



Our system takes in a monocular video of a moving person, trains the NeRF representation, and outputs the novel view and novel animation videos of the person.

Goals

Our goal is twofold:
1. Create an end-to-end web application that takes in a wild-video and produces Human NeRF reconstruction, as well as novel view and animation videos.
2. Optimize code in the InstantAvatar repo, with the goal of reducing the end-to-end execution time to within 5 minutes, while maintaining the quality of reconstruction.

Constraints

Our main constraint is that the input is only a monocular video, which means that it needs to be preprocessed to extract parameters, such as poses and masks, that are required by HumanNeRF.
As for computing resources, we will use a PC with a RTX4090 as our server for the web application backend.

Major Challenges

Our challenges involve optimizing and integrating the InstantAvatar processing pipeline with web applications. This requires us to understand InstantAvatar architecture and its implementation in code. In addition to experimenting with several performance optimization techniques, we also experienced some initial issues with reconstruction quality, which are resolved by tuning processing parameters and options.

Approach

Starting Codebase

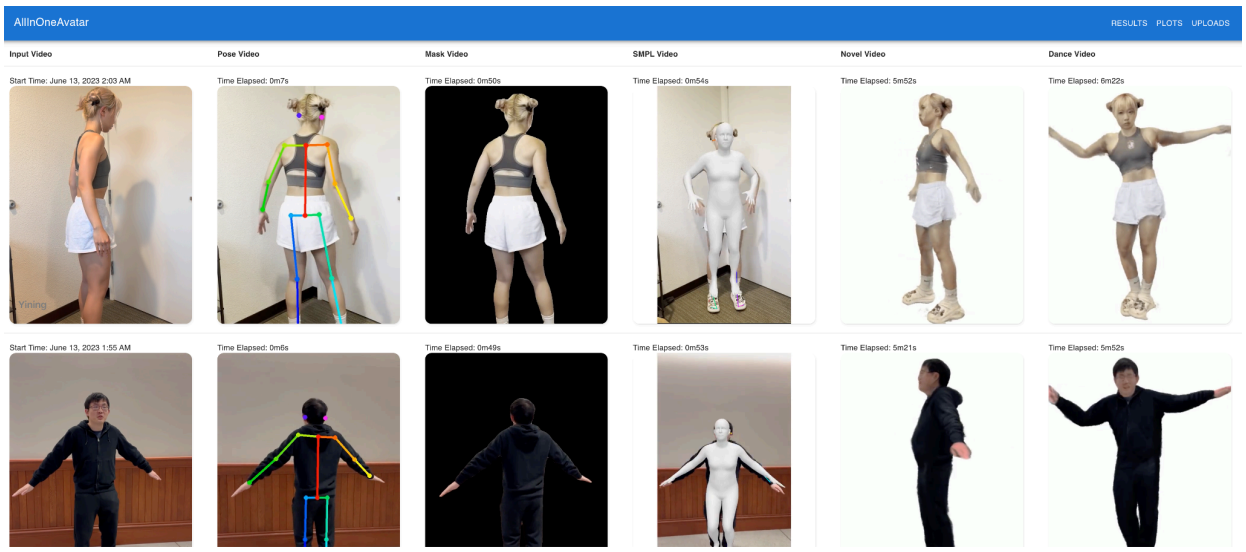
We start with the [InstantAvatar] Repo, which is a HumanNeRF architecture with several speed optimizations. We ran the processing scripts on an example dataset and collected their execution times.

	Preprocess	Fit(500 epochs)	Train(200 epochs)	Novel View	Novel Animation
Time	1m5s	8m26s	3m8s	10s	34s

The whole processing pipeline takes around 14 minutes. Considering the overheads of transferring data between frontend and backend in a web application, the total end-to-end execution time could be longer.

End-to-End System

We create a web application that takes in user videos, runs the processing pipeline, and displays the result novel view and animation videos.



Performance/Quality Optimization

We experimented with several techniques with the potential of improving either the speed or quality of reconstruction. We will explain these techniques in this section and evaluate them in the next section.

Parallelizing preprocessing steps

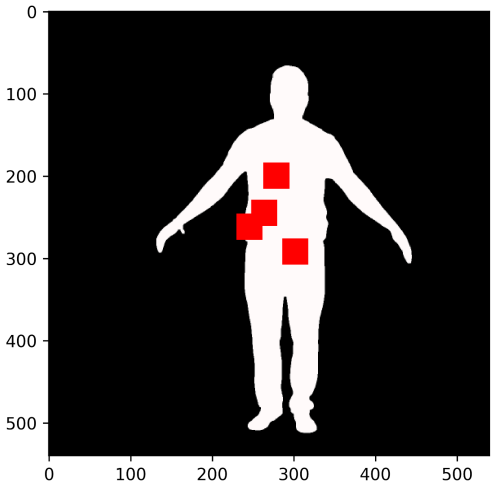
Since the input of our system is only a monocular video, we need to preprocess the video to extract parameters like poses and masks. The instant avatar repo has a preprocessing script. It has the following steps:

1. Converts video to images
2. Uses Openpose to estimate the 2D key points
3. Uses Segment-Anything to segment the scene
4. Uses ROMP to estimate camera and smpl parameters

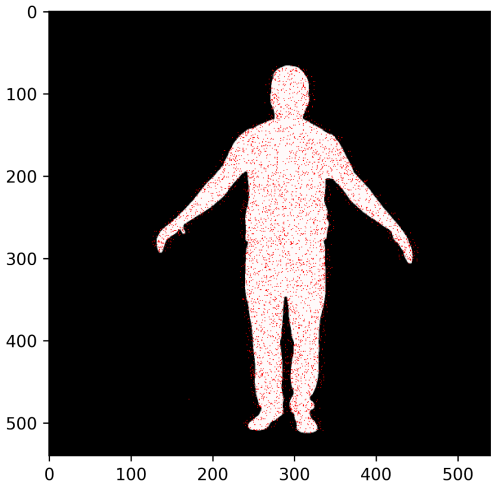
Each step is executed sequentially. Since the last three steps are independent of each other, we expect running them in different processes at the same time will reduce the preprocessing time.

Sampler

InstantAvatar defaults to using patch samplers, and we want to experiment with edge samplers. We expect patch samplers to perform better in terms of quality, since sampling in a window is more robust than sampling randomly.



Patch Sampler



Edge Sampler

Fast-SNARF

We want to determine the relative speed and quality tradeoff between SMPL, which is a simpler deformation module, to Fast-SNARF, which is InstantAvatar's default. We expect the training using SMPL is faster than Fast-SNARF, but the resulting reconstruction quality is lower.

Pretrain

Since HumanNeRF reduces the problem domain to human reconstruction, the resulting density field should be very similar for all NeRF reconstructions (color field should still be very different, though). If during training, we load the NeRF model parameters from another trained model, we

could potentially start with a very good density field, which could result in a better-quality reconstruction with less epochs.
To enable loading pretrained models, we modified the Python script for training to load any pretrained model on startup, and checkpoint the latest model when training ends. To experiment with pretraining, we train and save a model on one dataset, copy the saved model to the pretrained folder, and train another model using the pretrained model.

Early Stopping

We looked at the progression of fitting and training, and found out that the quality of reconstruction does not improve significantly past the early epochs. We also found that the training loss stops decreasing pretty early. In addition to reducing the max number of training epochs, we also implement an early-stopping strategy: training stops early if training PSNR value does not improve in the last 25 training epochs. We hope that early stopping would help with optimizing the training time while preserving the quality of reconstruction.
To enable early stopping, we modified the Python script for fitting and training to add an early-stop callback to the Pytorch Lightning model trainer.

Evaluation and Results

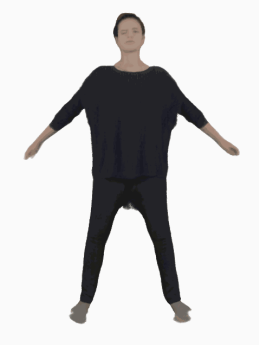
Since we have a couple of speed and quality optimization techniques, it is necessary to measure the effectiveness of each optimization separately.
For techniques targeting quality improvement, our evaluation consists of qualitative results of final reconstruction (image) / some quantitative metrics. We consider the technique as successful if there is improvement in visual quality or metrics.
For techniques targeting speed improvement, we evaluate the execution time and also the reconstruction quality, and we consider it success if there is a speedup without hurting the reconstruction quality.
We use training time as the speed metric, and use PSNR (higher is better), SSIM (higher is better), LPIPS (lower is better) as quality metrics.
Finally, we will do an end-to-end experiment with optimization techniques that proved to be useful, in order to determine whether we achieved our main goal of HumanNeRF in 5 minutes.

Parallelizing Preprocessing steps

	Not parallelized	parallelized
Execution time	1m5s	55s

Parallelizing the preprocessing script doesn’t reduce the execution time a lot. We might hit the I/O bound instead of the compute bound.

Sampler

	Patch Sampler	Edge Sampler
female-3-casual		
male-3-casual		

Qualitatively, we see edge samplers create more noisy reconstruction results. This agrees with our expectation that patch samplers are more robust.

Fast-SNARF

In this experiment, we evaluate fast-SNARF vs. SMPL deformer using two example datasets, female-4-casual and male-4-casual, with the expectation that SMPL will be faster than SNARF, but results in lower quality reconstruction.

Table: SMPL vs. Fast-SNARF

	female-4-casual					male-4-casual				
	Train Epochs	Train Time	PSNR	SSIM	LPIPS	Train Epochs	Train Time	PSNR	SSIM	LPIPS
SMPL	80	2m6.74s	23.54	0.9392	0.03543	80	2m27.1s	22.88	0.9351	0.04815
Fast-SNARF	80	2m5.9s	23.56	0.9396	0.03630	80	2m28.4s	22.97	0.9357	0.05102

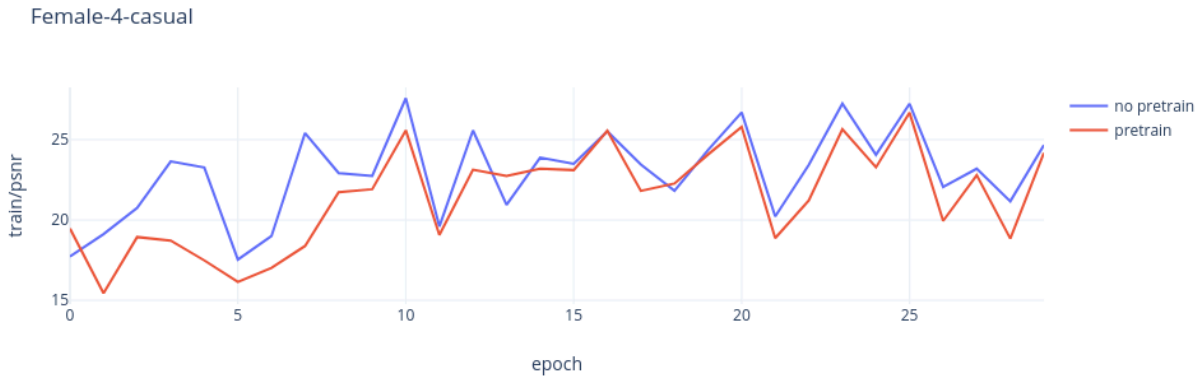
Surprisingly, these two deformers have similar run time and reconstruction quality. Since we will use a smaller epoch count than 80, we conclude changing the deformers does not help with training speed and quality much.

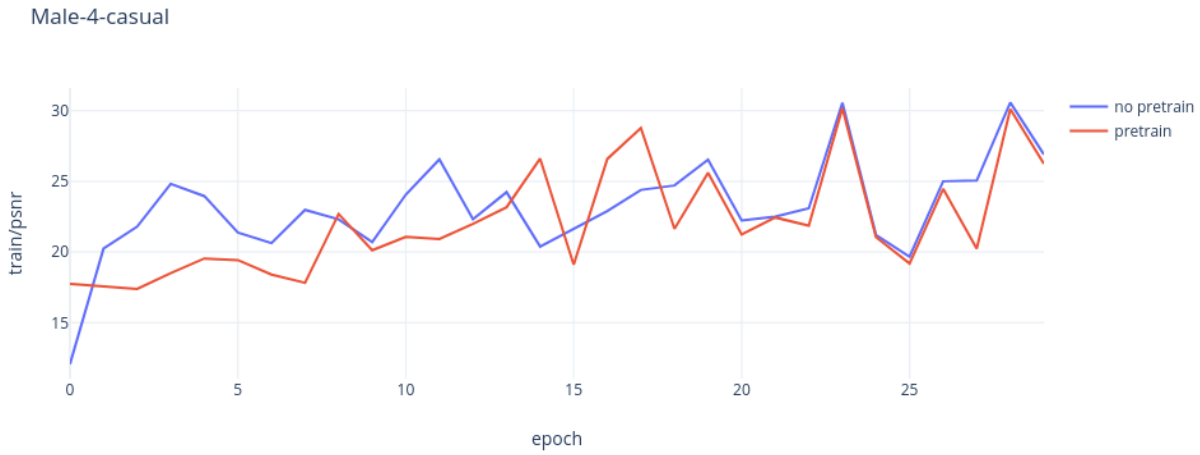
Pretrain

In this experiment, we evaluate the pretrain technique using the same two example datasets, female-4-casual and male-4-casual, with the expectation that pretraining will result in better reconstruction quality using a low number of training epochs (30 epochs).

Table: Pretrain using female-3-casual, 200 train epochs										
	female-4-casual					male-4-casual				
	Train Epochs	Train Time	PSNR	SSIM	LPIPS	Train Epochs	Train Time	PSNR	SSIM	LPIPS
InstantAvatar	30	0m50.9s	23.44	0.9392	0.04068	30	0m53.3s	23.01	0.9354	0.05467
InstantAvatar + Pretrain	30	0m48.8s	23.23	0.9374	0.04165	30	0m50.8s	22.89	0.9343	0.05504

Table: Pretrain using female-3-casual, 200 train epochs								
	female-4-casual				male-4-casual			
	InstantAvatar		InstantAvatar + Pretrain		InstantAvatar		InstantAvatar + Pretrain	
	PSNR	Loss	PSNR	Loss	PSNR	Loss	PSNR	Loss
Epoch 1	17.72	0.02410	19.46	0.01509	12.07	0.03267	17.74	0.01954
Epoch 2	19.12	0.01966	15.44	0.03400	20.23	0.01695	17.59	0.01902
Epoch 3	20.74	0.01145	18.93	0.01403	21.78	0.01186	17.38	0.01693
Epoch 4	23.63	0.00889	18.71	0.01682	24.82	0.00791	18.50	0.01470
Epoch 5	23.26	0.00970	17.48	0.01817	23.95	0.00799	19.53	0.01464
Epoch 10	22.73	0.00637	21.91	0.00795	20.70	0.01150	20.12	0.01417
Epoch 15	23.87	0.00508	23.19	0.00583	25.90	0.00413	26.59	0.00472
Epoch 20	24.35	0.00475	24.07	0.00497	24.61	0.00396	25.60	0.00368
Epoch 25	24.05	0.00423	23.19	0.00469	21.20	0.00692	21.05	0.00725
Epoch 30	24.65	0.00406	24.17	0.00444	26.90	0.00237	26.23	0.00297





We see that except at the first epoch, pretrain does not help with improving the quality. This is because from our experience, the model can locate the occupancy space of the person very quickly, which is the main objective of using pretraining. Instead, most of the training time is spent on regressing on the details of the human body (especially face), which pretraining does not help at all. Hence we conclude pretraining does not provide any speedup for HumanNeRF.

Early Stopping

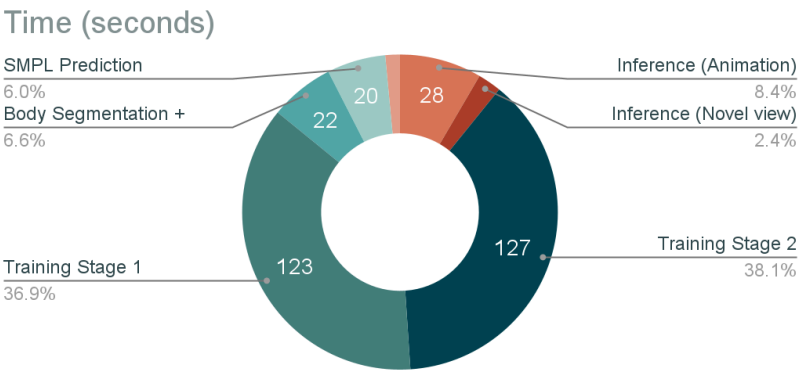
In this experiment, we evaluate whether early stopping can reduce the training time while preserving the quality. We choose an arbitrary stopping criteria of ‘no improvement in train/psnr metric in 25 training epochs’, which can be further optimized.

Stopping Criteria: No improvement in train/psnr in 25 train epochs										
	female-4-casual					male-4-casual				
	Train Epochs	Train Time	PSNR	SSIM	LPIPS	Train Epochs	Train Time	PSNR	SSIM	LPIPS
InstantAvatar	200	5m44.0s	23.45	0.9390	0.03413	200	5m3.7s	22.85	0.9361	0.04873
InstantAvatar + Early Stopping	94	2m22.8s	23.47	0.9397	0.03601	45	1m16.7s	22.84	0.9343	0.05321

We notice that early stopping is able to reduce the training epochs while maintaining a similar quality metric.

End-to-End Time Analysis

We ran our end-to-end pipeline on male-3-casual dataset, and below is the breakdown of time spent in each stage of our pipeline.



Preprocessing time: ~40s
Average training time: ~120s for each stage
Average training epoch number (early stopping): ~55 epochs for each stage
Novel View & Animation: ~40s
Total time: ~5mins

Results

We visualize more results on People Snapshot dataset.

Team Responsibilities

All team members contribute equally to this project
Senyang: Pretrain/Early-Stopping implementation and evaluation
Yining: End-to-end system pipeline. Preprocessing implementation and parallelization. Sample method optimization. End-to-end time analysis.
Boyu: Frontend and backend of the web application. Parallelize the image segmentation script.

References

[1] T. Jiang, X. Chen, J. Song, and O. Hilliges, “InstantAvatar: Learning Avatars from Monocular Video in 60 Seconds,” arXiv.org, Dec. 20, 2022. https://arxiv.org/abs/2212.10550 (accessed Jun. 13, 2023).
[2] C.-Y. Weng, B. Curless, P. P. Srinivasan, J. T. Barron, and I. Kemelmacher-Shlizerman, “HumanNeRF: Free-viewpoint Rendering of Moving People from Monocular Video,” arXiv:2201.04127 [cs], Jan. 2022, Available: https://arxiv.org/abs/2201.04127
[3] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding,” arXiv:2201.05989 [cs], Jan. 2022, Available: https://arxiv.org/abs/2201.05989
[4] M. Loper, J. Romero, and M. Black, “SMPL: A Skinned Multi-Person Linear Model.” Available: https://files.is.tue.mpg.de/black/papers/SMPL2015.pdf
[5] X. Chen et al., “Fast-SNARF: A Fast Deformer for Articulated Neural Fields,” arXiv.org, Dec. 01, 2022. https://arxiv.org/abs/2211.15601 (accessed Jun. 13, 2023).