Reconstruct 3D Human Pose with NeRF

Minzhe Guo
University of Michigan
vincegmz@umich.edu

Lyuyongkang Yuan
University of Michigan
lyuyong@umich.edu

Tianyuan Du
University of Michigan
alexdu@umich.edu

Yikai Wang
University of Michigan
danwyk@umich.edu

Human pose estimation is a challenging task in computer vision, and Openpose [1] has made it possible to extract 2D human keypoint information from images. However, Openpose [1] still encounters errors, and transforming 2D keypoint information to 3D human pose estimation can be challenging. In this paper, we present a pipeline for 3D human pose estimation that addresses the challenges of extracting 2D human keypoint information from images and transforming it into 3D pose estimation. Our pipeline includes a pose machine for 2D keypoint extraction, and afterwards, uses InstantNGP [3] and D-NeRf [4] for 3D reconstruction. We also introduce two extensions for pure human pose and dynamic scene reconstruction. To evaluate our pipeline, we built a reality-aware dataset and conducted qualitative and quantitative analyses. Our results demonstrate that our proposed pipeline achieves promising results in terms of 3D reconstruction quality and accuracy. The pipeline has the potential to impact various industries, including sports analysis and virtual reality, and future work can explore its potential for more robust 3D reconstruction and real-time applications.

1. Introduction

Human pose estimation has been a challenging task in computer vision and has various applications in fields like robotics, sports analysis, and healthcare. Extracting 2D human keypoint information from images has been made possible with the advent of Openpose [1], which is an open-source library for real-time multi-person keypoint detection and multi-threading written in C++ with Python and MATLAB bindings. However, despite its popularity, Openpose [1] still encounters errors that would render it unfit for certain tasks. Additionally, transforming the 2D keypoint information to 3D human pose estimation can be a challenging task.

In this paper, we propose a pipeline that extracts 2D human keypoint information with our pose machine that is specifically constructed for this task, overlays the 2D keypoints on the original image, and applies InstantNGP [3] for 3D reconstruction. To train and evaluate our pipeline, we built a dataset based on various scenarios. Our proposed pipeline has the capability to transform the 2D keypoint information into 3D human pose estimation with high accuracy by utilizing InstantNGP [3] which is used for generating new views through constructing 3D scenes. Additionally, we introduce two extensions to our pipeline. The first extension is a pure human pose without overlaying, which explored potential representation of the human pose without any interference from the background. The second extension is dynamic scene reconstruction using D-NeRF [4], which enables us to reconstruct a human pose in a dynamic scene.

Our proposed pipeline offers a significant contribution to the field of computer vision and has the potential to impact various industries positively, including sports analysis and virtual reality. We believe future works could explore its potentials for more robust 3D reconstruction and real-time applications.

2. Basic Approach

2.1. Extract 2D human keypoint-information with Openpose

Our goal was to extract 2D human pose key points from a video in order to reconstruct the 3D human pose. We chose to use Openpose [1], which is an open-source software library for detecting and tracking human body key points in images and videos. Our approach consisted of the following steps:

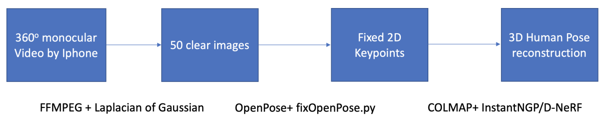
360º monocular Video by Iphone
50 clear images
Fixed 2D Keypoints
3D Human Pose reconstruction
FFMPEG + Laplacian of Gaussian
OpenPose+ fixOpenPose.py
COLMAP+ InstantNGP/D-NeRF

Figure 1. Pipeline

(1) Preprocessing the input video: We applied a Laplacian filter to each frame of the video to assess the noise score. We then selected the top 50 images based on the Laplacian scores, as these frames were deemed to have the most informative and representative human poses.

(2) Running Openpose: We applied Openpose [1] to each selected frame to extract 2D keypoint information. Openpose [1] uses a deep neural network to detect and track human key points in real time. The output of Openpose [1] is a set of 2D keypoint-locations, including the position of body joints such as the shoulders, elbows, and knees and facial landmarks such as the eyes, nose, and temples.

2.2. Fix Openpose Error

We observed that some inaccuracies in the 2D keypoint-data were caused by camera perspective. Specifically, the head key points were sometimes inaccurate due to the limitations of Openpose [1] in detecting certain facial landmarks. For example, when the camera is positioned behind the person, the key points of the eyes may be missed. To address this issue, we developed a prediction algorithm that estimated the spatial position of the head based on the presence or absence of specific keypoint-landmarks. We identified three conditions:

(1) all key points of the nose tip, temples, and eyes are detected;
(2) only half of the temples and eyes are detected;
(3) only the key points of the temples are detected;

For each condition, we derived a different prediction algorithm that estimated the spatial position of the head. We then replaced the original inaccurate head data with the new estimates, resulting in more accurate 2D keypoint-data for subsequent analysis and 3D reconstruction.

2.3. Overlay 2D Keypoints on the original image

The next step after fixing the errors on the ”head” position calculated by Openpose [1] is to overlay the 2D keypoints of human pose on the original images we captured from our input video. To accomplish this, we use the draw_lines() function, which takes in the fixed keypoint-positions (.json) from our Openpose [1] calculation and uses those positions to draw lines on the video captured images.

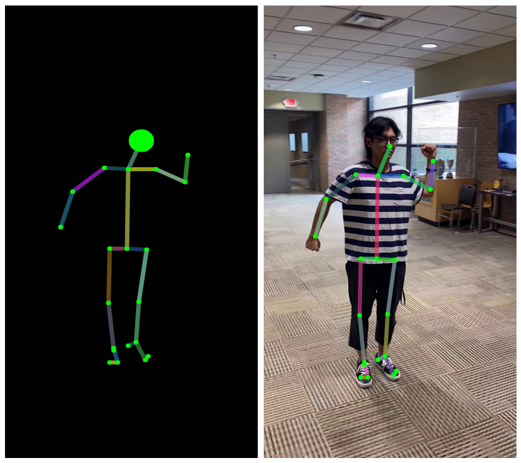

Figure 2. Drawing human pose on a blank canvas and Drawing human pose on an original image

Our sample video captured a static human model in 1080p 30fps for 25 seconds. The person who took the video shot a 360-degree video of the human model by circling around him. To simplify the training process, we extracted a picture every 15 frames, resulting in a total of 50 images.

The draw_lines() function was applied to each of the 50 images. We first tested out the keypoint-positions by drawing lines between corresponding 2D keypoints of human pose and to mark he keypoints on a blank canvas. With a clear indication of the relationship between 2D keypoint-positions, we then overlaid the edges and the keypoints of human pose on top of the original images as shown in Figure 2.

The resulting images are then saved as 2D keypoints labeled images and used as input for our instantNGP [3] for a later 3D reconstruction.

2.4. Apply InstantNGP for 3D Reconstruction

After getting images with 2D keypoints overlaid on top, we train InstantNGP [3] to obtain 3D keypoints and human reconstruction. We first run COLMAP [5] on those images, a multi-view stereo software that outputs information such as camera poses and camera intrinsics. This information is essential because we need to get 3d rays shot from camera origin to every image pixel from which we can do volume rendering and train a NeRF [2]. For instantNGP [3] specifically, this information is stored in a transforms.json file and it takes about a few seconds to train InstantNGP [3] compared to the original NeRF [2] which takes 15- 20 hours to train a moderate model.

3. Extension 1: Pure human pose without overlaying

We note that users just want to see the pose only because it can be distracting for these users to see overlaid poses on humans. So in the first extension, we train InstantNGP [3] on pure human pose. We first test a naive way: directly feeding the key points into InstantNGP [3]. We then process the pose output in the hope of better COLMAP [5] feature matching:

(1) Add more connections of keypoints to make the pose structure more complicated.
(2) Differentiate each edge with different colors
(3) Increase the size of head keypoint
(4) thickening edges.

4. Extension 2: Dynamic Scene with D-NeRF

To make our task even more challenging, we also attempted to reconstruct 3D human Pose in a dynamic scene with D-NeRF [4]. D-NeRF [4] is a NeRF [2] model that handles dynamic scenes of synthetic data. D-NeRF [4] has a very different data type and structure from InstantNGP [3].The D-NeRF [4] data is structured in the following format: train/, val/, test/ ,transforms_train.json, transformas_val.json and transforms_test.json. The first three folders store the RGBA images for D-NeRF [4] training, validation and testing. The A channel of RGBA denotes how opaque each pixel is and can be understood as a mask. However, D-NeRF [4] does not provide any script for generating the data in the above format. So we first run the COLMAP [5] in InstantNGP [3] on a 30 fps video to get video frames in 2 fps and transforms.json for these frames. The frames in transforms.json are in random order and we write a script that splits the first 80% frames to be the training set, the next 10% frames to be the test set and the last 10% to be the validation set and create a transforms_¡set¿ .json for each set. Lastly, to follow the Blender data type which has no background, we use rembg, a library that removes the background and leaves the human in scenes only.

After the preprocessing, we train D-NeRF [4] with the following configuration: learning rate = 2e-4, learning rate decay=500, N_iterations(number of total training iterations) = 800000, N_samples(number of coarse samples per ray) = 64, N_importance (number of fine samples per ray)= 128 , N_rand(batch size) = 1024, half_res = True. We originally trained with a batch size of 500 rays and learning rate of 5e-4. However, the rendering quality declines as training iterations increase. So, we increase the batch size to 1024 and decrease the learning rate to 1e-4.

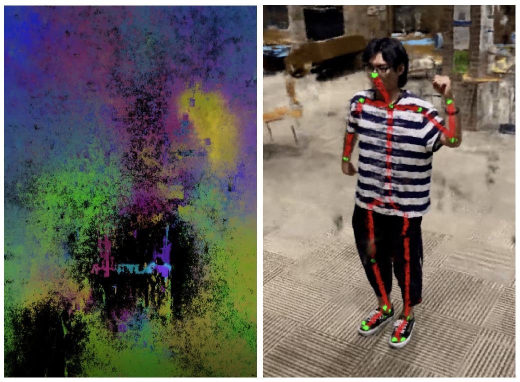
Figure 3. Result of Pure Pose Input

5. Experiment

5.1. InstantNGP

We feed instantNGP [3] with a set of photos, 50 images in total. Each of them contains a human figure. The human figure is labeled with human pose keypoints and the edges by connecting those keypoints. The human pose keypoints are generated from Openpose [1]. The result is a 3D reconstruction of the human figure in our photos.

Our results indicate that instantNGP [3] was able to produce high-quality 3D reconstructions of the human figures in the dataset. The reconstructions contained enough details of the human figures, without rendering unnecessary background scenes. This was likely due to the high level of detail present in the photos.

After getting images with 2d keypoints overlaid on top, we train InstantNGP [3] to obtain 3D keypoints and human reconstruction. Initially, we tried to use pure pose as input. However, the results after over 1000 iterations of training did not reign a reasonable performance. As shown in the figure, the output resembles more as a random point cloud of multiple colors, which implies it did not extract the features the poses intended to provide.

5.2. D-NeRF:

The data for testing the dynamic dancing pose is 10% of the total selected video frames, which are not seen in the training.

To evaluate our result, we use both qualitative and quantitative metrics. Specifically, a pair of estimated pose and ground truth the rendered video/frames will be provided in the report for an intuitive understanding of the quality of human pose reconstruction. For the quantitative metric, we will use the average of PCK (Percentage correct points) from 1 test image.

Figure 4 shows one of the rendered test images. The quality does not look high as blank space frequentl y appears and the move does not match either. After close

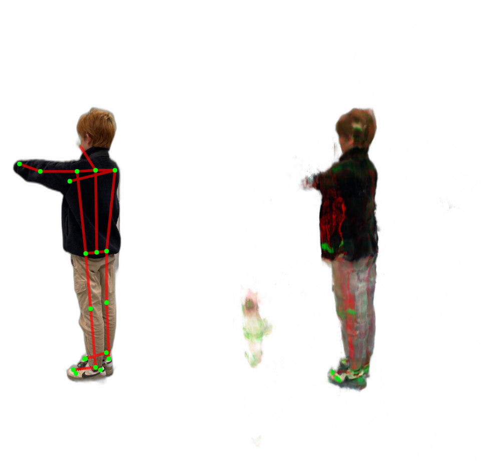
Figure 4. D-NeRF: Ground Truth & Estimate

examination, we hypothesize that the dance moves vary greatly and unpredictably across our custom dataset, while the moves of D-NeRF [4] dataset such as standing up are very predictable and do not vary greatly across images except the view changes. The PCK is only 6/25 = 24% and all the correctly labeled keypoints lie on the static part of the dancer, which suggest D-NeRF [4] are incapable of handling the real life dancing dataset.

References

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018. 1, 2, 3

[2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. 2, 3

[3] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2, 3

[4] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *CoRR*, abs/2011.13961, 2020. 1, 3, 4

[5] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-Motion revisited. June 2016. 2, 3