Animatable Musician with InstantAvatar

洪聖祥 109062315, 張雅涵 109062206, 蔡侑廷 109000129

1 Introduction

The selected topic is Using NeRF to Construct Animatable Musicians. Our goal is to input a piece of music, our system will synthesize novel views of the person playing the instrument. This results in a novel view, novel pose, and mixed reality video. We have utilized a popular 3D novel view synthesis technique in recent years - Neural Radiance Field (NeRF) to achieve our goal.

2 Motivation

Creating human animation is a human-laboring work (a few months). Most people use computer graphics tools to create animations of human motion. For constructing virtual players, we are inspired by the work from Scream Lab, NCKU. However, the pipeline in practice is quite complicated. For example, multiple software is required to complete a whole task, including Native Access, Reaper, and Unity, not to mention more than one programming language and manual operation are involved. Plus, using Unity to generate animation means that it can't be vivid, the virtual player is mainly created by computer graphics tools so it can not look like the "real" player in reality. The above problem can be solved by introducing NeRF and simplifying the pipeline to use Python and related Python libraries.

3 Method

Given a monocular video of a moving human, and a piece of music that might be composed by many instruments, our goal is to create a system that can reconstruct a realistic animatable human model, and then output a video of the human playing an instrument. In the output video, the virtual musician will look identical to the human in the given monocular video and will be playing the given piece of music using one of the instruments.

To reconstruct a realistic human model, we leverage the characteristics of NeRF, a model that produces highly realistic photos of a reconstructed 3D model from any point of view. Due to our limited filming and computing resources, among plenty of open-source animatable human reconstructing models, we chose to use InstantAvatar [1]. InstantAvatar is a NeRF-based model that can be easily trained by inputting monocular video, with relatively short training and photo rendering time.

To animate the virtual musician, so that it plays notes from the given piece of music with the correct tempo, we implemented a keyframe algorithm to reduce labor costs. We then add the instrument into the scene using the Z-buffer algorithm, and finally, output a novel view video of our virtual musician playing the given piece of music. In the next section, we will go through our pipeline, and dive into the implementation details of each stage.

4 Our Pipeline

To explain, I separate our pipeline into the training phase and testing phase and give details in the following sections. The overall pipeline is shown in Figure 1

- Training phase: Train Instant Avatar (Section 4.1) with a monocular video
- Testing phase:
    1. Motion capture (Section 4.2): Use an existing mocap system to capture a person (different from the person in the training phase) playing the instrument and extract human pose parameters
    2. Adding music and synchronizing with the human pose (Section 4.3): Use a MIDI file to decide when to execute the human pose
    3. Inference Instant Avatar (Section 4.1): Feed SMPL[3] parameters and camera poses (the view you want to look at) into trained Instant Avatar
    4. Scene composition (Section 4.5): Adding instruments to the scene using z-buffer algorithm

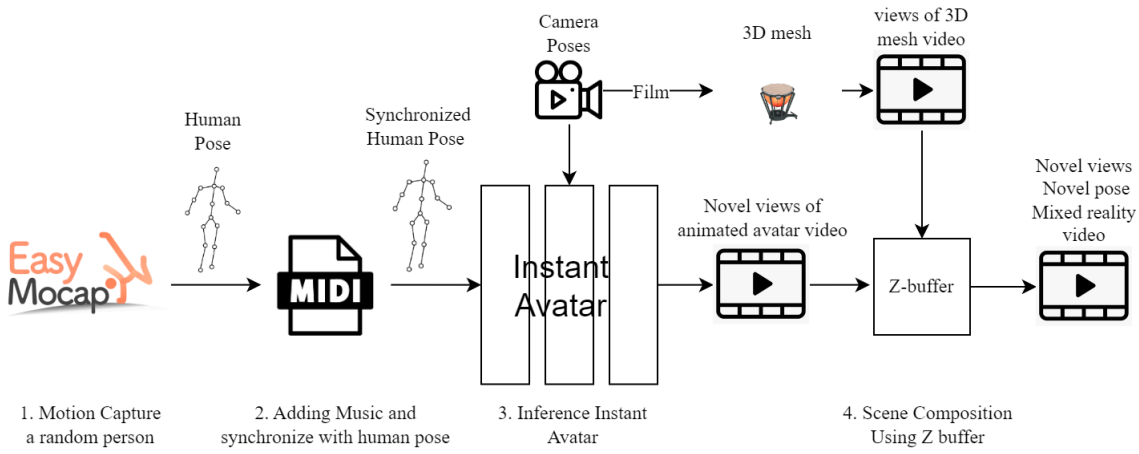


Figure 1: Overall Pipeline

4.1 Instant Avatar

InstantAvatar is a NeRF-based model with InstantNGP as backbone that can learn avatars from monocular video within a short period. The overview of the method is shown in Figure 2. First, given the rotation, translation, and body pose parameters (under the format of SMPL parameters) of the person and camera poses, for each frame, points are sampled along the rays in posed space. Then these points are transformed into a normalized space where the global orientation and translation of the person are removed. In this normalized space, points in empty space are filtered using an occupancy grid that is shared over all training frames. The remaining points are deformed to canonical space using an articulation module and then fed into the canonical neural radiance field to evaluate the color and density.

The SMPL parameters of the person can be obtained using a motion capture system (Section 4.2). To render a frame of the human under a novel pose, we change the pose parameters to animate the avatar.

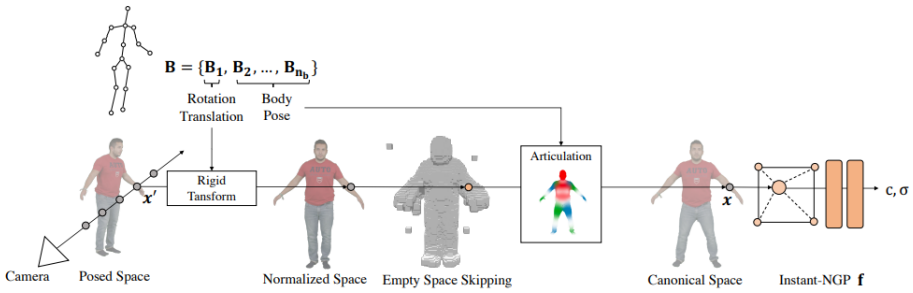


Figure 2: InstantAvatar Method Overview

4.2 Motion Capture

Once we finish training Instant Avatar, we move to the testing phase of our pipeline. Since we are going to synthesize views of a person playing instruments, we must first capture the human poses of a random person playing instruments and extract SMPL parameters to feed into Instant Avatar. However, it is impossible to capture 3D motion with only a single-view video, which is mostly filmed by. Therefore, we simulate a virtual person playing a drum in Blender and film the player from different views.

After filming different views of a person playing drum, we use an existing motion capture system - EasyMocap to extract SMPL parameters. The overall pipeline of motion capturing a virtual person playing a drum is shown in Figure 3.

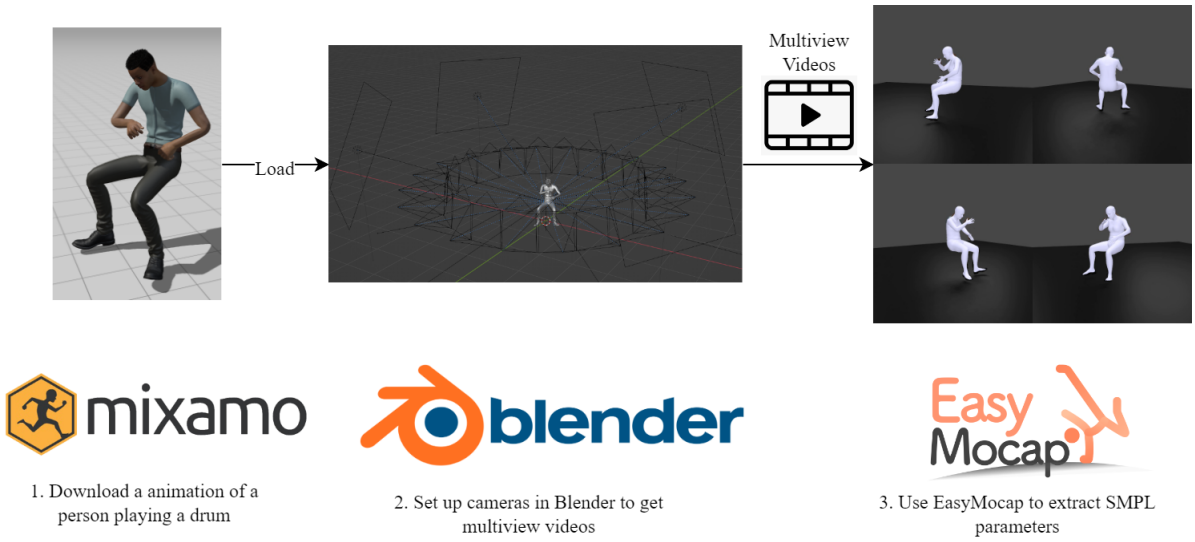


Figure 3: Pipeline of motion capturing a virtual person

4.3 Adding Music and Synchronize with Human Pose

4.3.1 Analyze Music

To decide when to execute the human pose, we need to analyze the music first. Thus, we take advantage of the paper Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi [2]. They provide a Python library that can work with MIDI data easily. The steps are listed below:

- Input the midi data
- Divide the instrument that the player would play and analyze its property.
- For instruments like drums, we only need to record the time (in seconds) when the player should play, but for other more complex instruments such as double bass, we not only need to record the time but also the index of pose that he or she should be playing.
  To be more specific, when the player plays double bass, he or she will move his finger according to the musical note that he or she is playing. The movement of the finger would also cause the body parameters to change accordingly. By generating all the possible human body parameters beforehand, we can know exactly how the player should act when analyzing the music notes of the MIDI file.

4.3.2 Keyframe Interpolation

When playing an instrument, each note corresponds with a pose. The process of playing a piece of music can be seen as continuously transforming from one pose to another. We use the motion capture system to obtain pose parameters for each note and then set a keyframe when a note is hit along the timeline.

To interpolate frames between two keyframes, we leverage the observation that transforming from one pose to another equals rotating body joints, where SMPL pose parameters describe the rotation of 23 body joints using an axis angle. We use the spherical linear interpolation(Slerp) method to obtain 23 joint angles for each frame between two keyframes. Sleep describes an interpolation (with constant angular velocity) along the shortest path (a.k.a. geodesic) on the unit hypersphere between two quaternions q1 and q2. It is defined as:

$$\mathrm{Slerp}(q_1, q_2; u) = q_1 \left({q_1}^{-1} q_2\right)^u$$

Figure 4: Slerp Formula

The parameter u moves from 0 (where the expression simplifies to q1) to 1 (where the expression simplifies to q2). For each joint, we first transform the axis angle to quaternions, next substitute the first keyframe's joint angle to q1, and the second keyframe's joint angle to q2, and then finally apply Slerp.

To make the avatar hit every note at the right time, we first set the fps for our output video. By preprocessing the MIDI file, we get an array that records the time a note is hit. Let the time a note is hit be t1, the time the next note is hit be t2, and then the number of frames between the two notes can be obtained by multiplying fps by (t1 - t2).

4.4 Adding Instruments

We use a 3D mesh to represent an instrument and film the scene with an instrument only along with the same camera path as rendering our avatar. As shown in Figure 6a, the circle-like line is the camera path, and we render the drum scene along the circle-like path. This eventually gets a video of viewing different angles of the drum.

4.5 Scene Composition - Z-buffer algorithm

4.5.1 Z-buffer algorithm

The next step is to add instruments to the scene. We use a simple algorithm for scene composition: Z-buffer algorithm. The Z-buffer algorithm is a simple algorithm that can composite 2 images into 1 image with a correct occlusion effect. The pseudo-code of this algorithm is described in Algorithm 1.

Algorithm 1 Z-buffer Algorithm

Input : Images of 2 Scenes Im_1, Im_2 and Depths of 2 images D_1, D_2
Output: Composited Image $NewIm$ with correct occlusion effect
$W \leftarrow$ Image width of Im_1 and Im_2
$H \leftarrow$ Image height of Im_1 and Im_2
for $i \leftarrow 1$ **to** H **do**
  for $j \leftarrow 1$ **to** W **do**
    if $D_1(i,j) < D_2(i,j)$ **then**
      $NewIm(i,j) \leftarrow Im_1(i,j)$
    else
      $NewIm(i,j) \leftarrow Im_2(i,j)$
    end
  end
end

4.5.2 Get the depth of NeRF

From section 4.5.1, we give a solution to composite 2 different scenes. However, how do we get the depth of the two scenes? For instruments, since it is a 3D mesh, we know exactly where the surface of the mesh is. Therefore, it is easy to get the depth of images. For the human avatar, it is quite difficult, since the human avatar is a **neural implicit representation**. In other words, we **can't** get the (x, y, z) information of the human surface explicitly.

Therefore, we utilize an attribute of NeRF to get the depth image. To illustrate, recap what NeRF is doing:

Illustrated in Figure 5, if we want to predict the color $\hat{C}(r)$ of pixel p, we trace along the RayA and sample K points. For the k^{th} sample points, we feed the (x, y, z) coordinate value of the k^{th} sample point and viewing direction (θ, ϕ) into an MLP. The MLP outputs color c_k and density δ_k of the k^{th} sample point. Repeat the process until all K sample points on RayA are done. Once all c_k and δ_k are known, we do the volume rendering process:

$$\hat{C}(r) = \sum_{k=1}^{K} w_k c_k \tag{1}$$

$$w_k = T_k(1 - exp(-\sigma_k \delta_k)) \tag{2}$$

$$T_k = exp(-\sum_{k'=1}^{k} \sigma'_k \delta'_k) \tag{3}$$

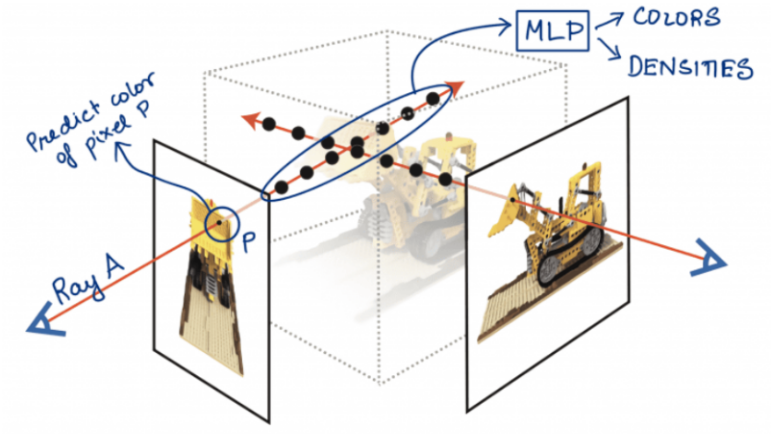


Figure 5: NeRF illustration

where $\delta_k = t_{k+1} - t_k$ is the distance between two adjacent samples and t_k is the distance between camera origin and the k^{th} sample point.

For calculating the depth of a pixel, we can reference from Equation 1 and utilize w_k. This gives the depth equation:

$$\hat{Z}(r) = \sum_{k=1}^{K} w_k t_k \tag{4}$$

where t_k is the distance between camera origin and the k^{th} sample point.

Once all depth images of two scenes are known, we can apply the Z-buffer algorithm (Algo 1) to composite 2 scenes.

5 Result

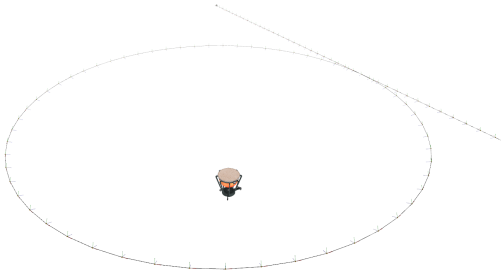
We composite 2 scenes, one is the drum scene and another is the human scene. Figure 6 is an illustration of the third point of view of the 2 scenes we want to render. Also, we follow the circle-like camera path to render the scenes. Note that Figure 6b is a virtual person, not the one trained from Instant Avatar. The person from Instant Avatar can only be represented implicitly, and cannot be represented explicitly.

For demonstration, we render 120 images along the camera path illustrated in Figure 6. The result is shown in Figure 7. The first row is the result rendered from Instant Avatar. The second row is the result rendered from a 3D mesh. The third row is the result of a composite of 2 scenes with the correct occlusion effect which are novel view, novel pose, and mixed reality images.

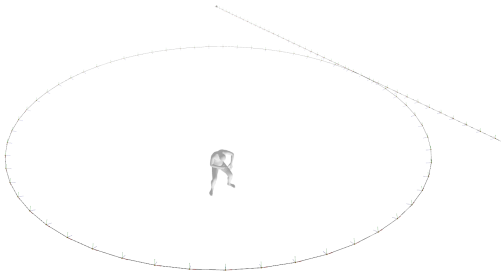
References

[1] Tianjian Jiang, Xu Chen, Jie Song, Otmar Hilliges, "InstantAvatar: Learning Avatars from Monocular Video in 60 Seconds", In CVPR, 2023

[2] Colin Raffel and Daniel P. W. Ellis, "INTUITIVE ANALYSIS, CREATION AND MANIPULATION OF MIDI DATA WITH pretty_midi"



(a) Drum Scene



(b) Virtual Human Scene

Figure 6: Third point of view of 2 scenes

[3] Loper, Matthew and Mahmood, Naureen and Romero, Javier and Pons-Moll, Gerard and Black, Michael J. , ”SMPL: A Skinned Multi-Person Linear Model”, SIGGRAPH Asia, 2015

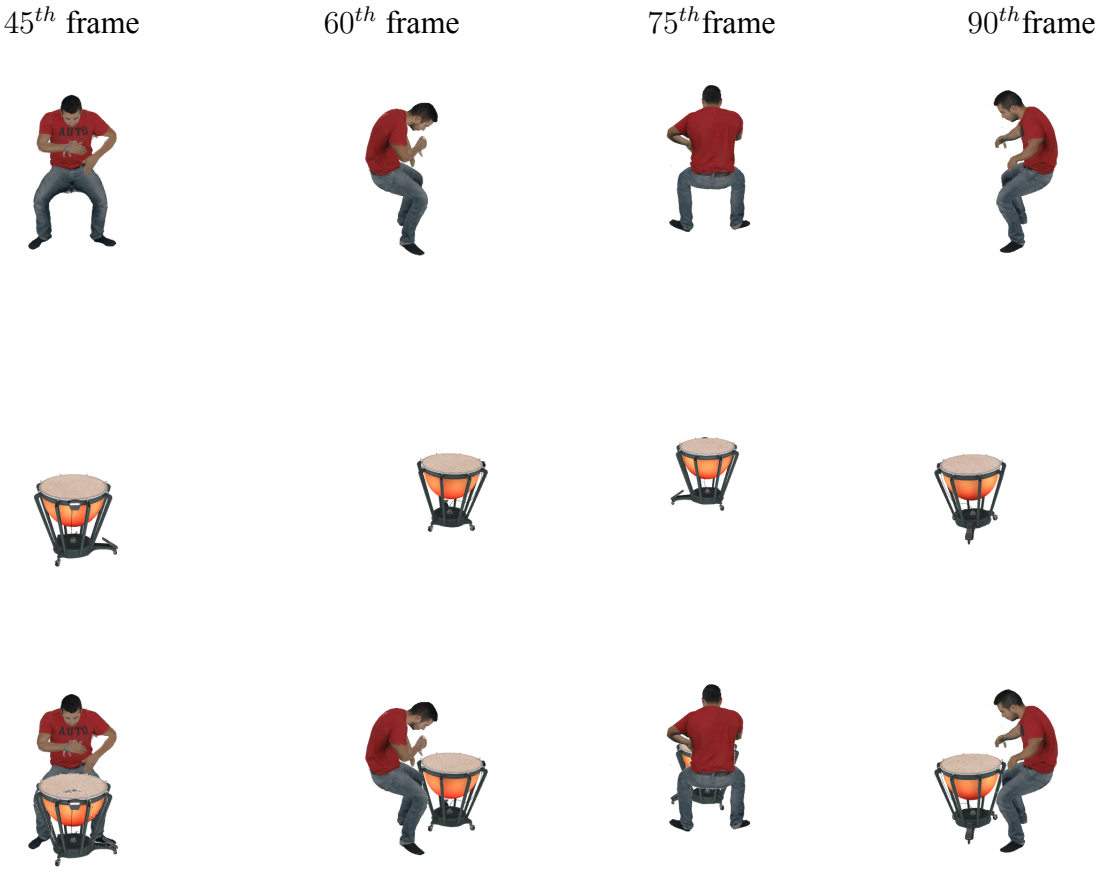


Figure 7: Result of scene composition