Assignment 2 - Motion Processing

Highlights

- Submission DDL: November 12th (Tue)

Introduction

This assignment will provide a practical introduction to working with animation data through various algorithms such as interpolation and concatenation. Additionally, you will learn to consider various variables from motion data to enhance the performance of the motion matching method.

The experiment environment is re-designed by GAMES105, thanks for the open sourcing.

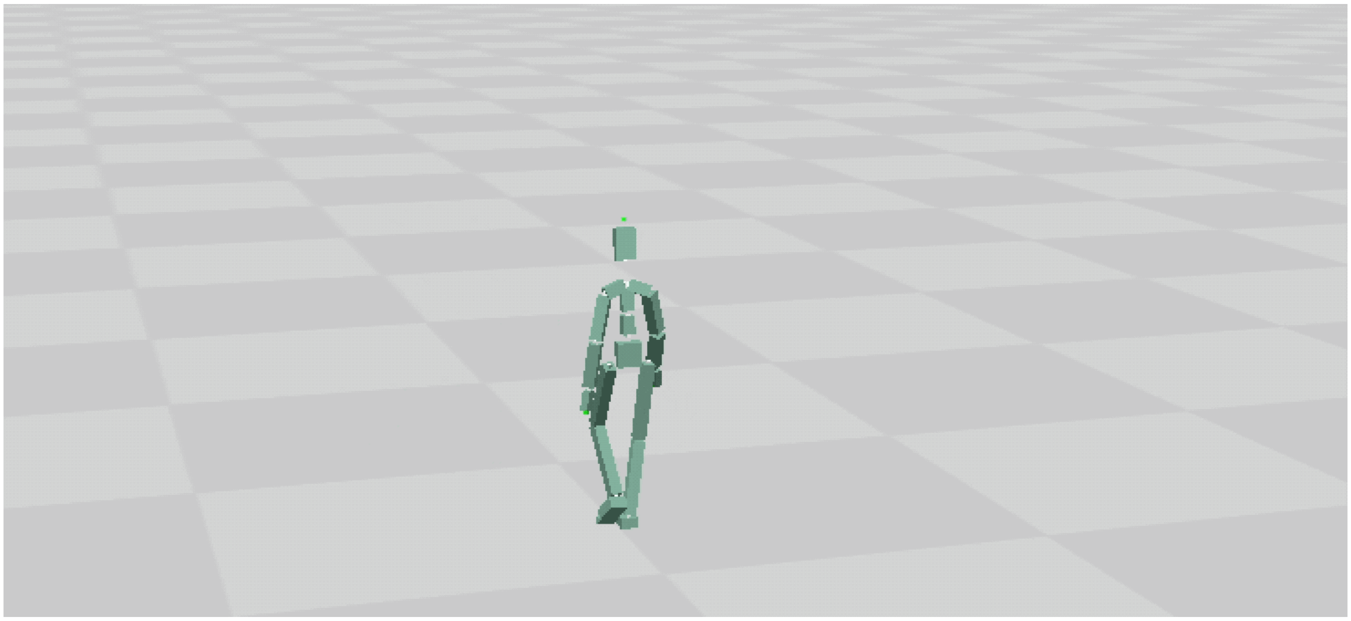
Submission

File format: A compress file **[uid_name_assignment2.zip]** with:

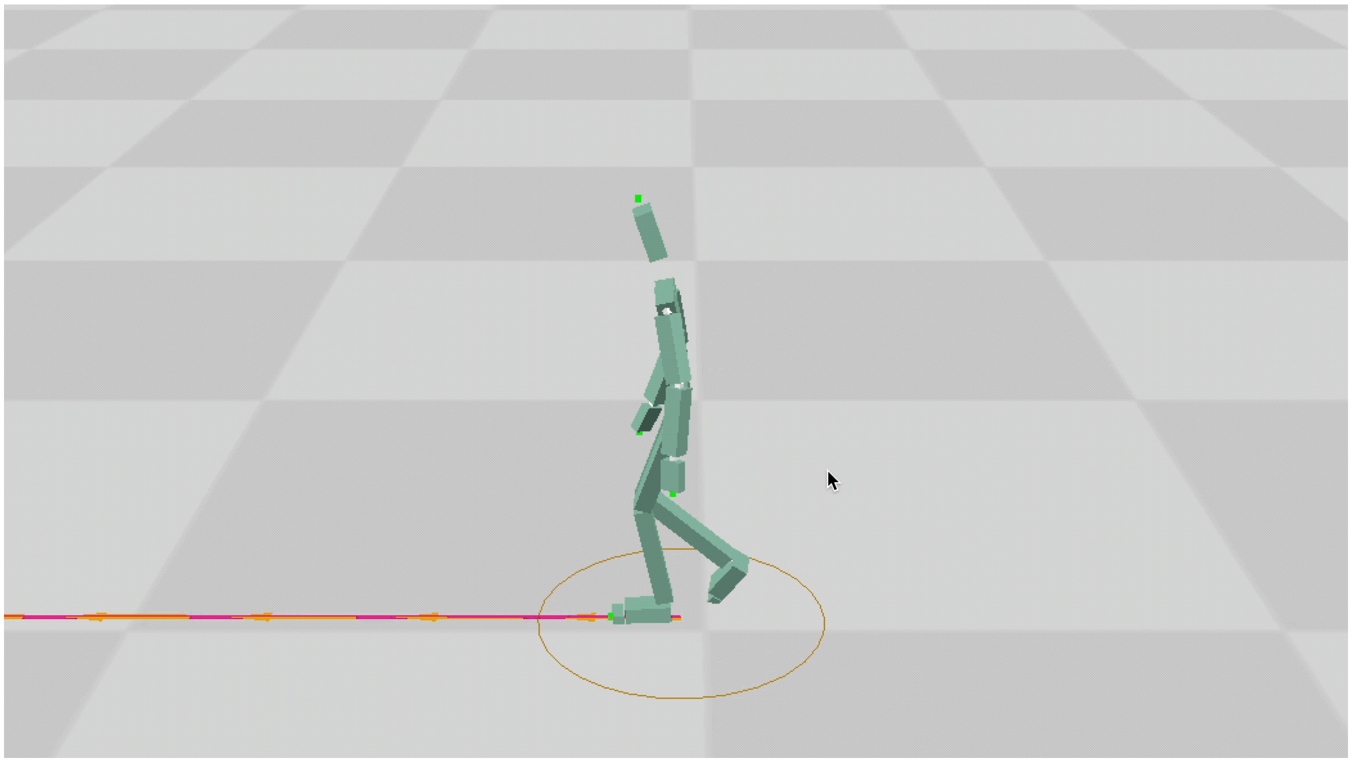
1. video_file1 [motion concatenation], video_file2 [motion_matching](if you video file is too big, you can also prepare a link to google drive or onedrive)
2. task1_motion_editing.py, task2_motion_matching.py (do not rename these files)
3. uid_name_2.pdf

Examples

Desired results for Motion Concatenation



Desired results for Motion Matching



Before Task 1 & 2

Enviroment Setting

As same as assignment 1, the Task 1 & 2 also requires Python 3 runtime and Panda3D library.

```
# recommend to use Anaconda to manage enviroment
conda create -n comp7508 python=3.8
conda activate comp7500
conda install numpy scipy
pip install panda3d

# Enviromtns verification. After running, you should see a skeleton in a 3D space
cd ./assignment_2
python env_test.py
```

Pre-knowledge

Human Skeleton and Motion

- As introduced in the tutorial, a body skeleton is a collection of bones representing the connection of body joints. The body joint is a 3D point in space, with (x, y, z) positional information as primary data. Sometimes, the offset between two joints will also be used, formulated by (x2-x1, y2-y1, z2-z1) with a vector.
- When we rotate a bone, a rotation value will be applied to the parent joint of this bone. For example, if we have joint_1 in (0, 0) and joint_2 in (1, 0), then a 45-degree anticlockwise rotation on joint_1 will yield the positional movement of joint_2, from (1, 0) to (sin45, sin45).
- Global/Local Rotation. The local rotation always means the relative rotation from the parent to the child's joint, and the global rotation (orientation) represents the global rotation of the whole 3D space. In the above case, after the rotation, the orientation of J1 and J2 is both 45 degrees, but the local rotation of J2 keeps the same as 0 degrees, only the local rotation of J1 will be updated to 45 degrees.
- BVH file is a combination of joint offset and local rotations, and you can find more details with this link.
- Quaternion is a rotation representation, you can find the details and its history with wiki. In this project, (x, y, z, w) is the default order.

Interpolation:

- Linear Interpolation
  We assume the offset between each two iteams is the same

```
start_value, end_value = 0, 11
between_num = 10
offset = (start_value - end_value)/between_num
betweens = [start_value + offse*i for i in range(between_num)]
```

- SciPy Interpolation: https://docs.scipy.org/doc/scipy/reference/interpolate.html
Imagain the interpolation as a function, we know some x and corresponded y, and send it to scipy function, it will return a reasonable known function that can be used further with a new_x.

```
from scipy.spatial.transform import Slerp

key_quaternions = R.from_quat(q1), …(q2)
keys = [0 , 1]
slerp_function = Slerp(keys, key_quaternions)
new_keys = np.linspace(0, 1, 10)          ->       0, 0.1, 0.2 … 1
interp_quaternions = slerp_function(new_keys)
```

More Tips:

- Numpy: Indexing of numpy array: https://numpy.org/doc/stable/user/basics.indexing.html
- Motion Matching: https://www.youtube.com/watch?v=KSTn3ePDt50
- VS code debugging feature: https://www.youtube.com/watch?v=KEdq7gC_RTA

Task 1 - Keyframing Animation

For this task, you will be responsible for developing two interpolation functions and implementing them using various keyframing settings. By default, we require that you utilize linear interpolation for position data and slerp for rotation data.

You are required to implement the interpolation functions at line 43 in *task1_motion_editing.py.* After doing so, you should uncomment lines 184-187 one by one and execute the script in order to call the function. Successful execution should result in a walking motion being displayed.

From Line 53, the primary keyframing process is detailed. First, the motion data is loaded, and then keyframes is taken every N (time_step) frames. Next, the interpolation function is called to generate M (target_step) posed frames. If N > M, the motion will play out more slowly, and if N < M, it will be faster. No modifications are required for this section of code.

Screenshot of walking motion will be expected in the assignment report.

Task 2 - Motion Concatenation

In this task, you will be required to implement a simple method for concatenating two distinct motions: walking and running. These motions are positioned globally in different locations and have distinct motion styles, directions, and velocities. As such, you will need to consider these differences to achieve a realistic concatenation.

The basic structure of the pipeline is presented at line 92 in *task1_motion_editing.py*. Upon completing the implementation, you should uncomment line 189 and examine the final result, which will be a combination of the walking and running motions as shown as above figure.

The basic pipeline consists of five steps, each of which will be considered in your grading: 1. Locating the search windows 2. Calculating the similarity matrix 3. Determining the closest frames with real frame indices 4. Shifting motion2 to motion1 5. Utilizing your task1 function for interpolation.

Additionally, you will be eligible for bonus points if you can find ways to address the following issues:

1. If the frame interval is set too high, noticeable on-the-fly motion will occur. How can we determine the optimal number of between frames? Alternatively, is there a way to enhance the realism of root position in these frames?
2. The current pipeline does not consider velocity. Inertialization is a viable option: https://theorangeduck.com/page/spring-roll-call#inertialization
3. Any other improvements are also welcome.

Task 3 - Motion Matching

Motion matching is a sophisticated technology used in real-time character control systems. In this section, using a basic motion matching framework, you will required to design the matching variables and experiement how these variables impact the matching quality.

If you want to modify the variables in the motion matching code *task2_motion_matching.py*, you should pay attention to the following three parts:

1. Line 21 contains a feature mapping variable that records all feature shapes. While some features are already prepared, additional features with specific names can also be included by yourself.
2. From line 450, you must determine which features to use by filling in their names in the *selected_feature_names* variable and their weights in the *selected_feature_weights* variable. Note that all features used must be present in the mapping variable.

So basiclly, just remember: **For the feature you wanna use, you should present it in the mapping varaible.** And also, around L176, it should have the implementation about its calculation.

We suggest using fewer variables to achieve realistic performance. And the grading of this part will depend on your variable number.

Assessments

- part1_key_framing (30%)
  - Linear interpolation (10%); Slerp Interpolation (15%)
  - Report the different performance by giving different numbers (5%)
- part2_concatenation (35%)
  - Define the search window (10%) + Calculate the sim_matrix (10%);
  - Find the real_i and real_j (10%);
  - The shifting on the root joint position (5)
- part3_motion_matching (25%)
  - Less variables, and better performance(total 15%, 22% - your_variable_num)
  - System analysis (10%) about variable selection, future frame range, etc.
- Report (8%) + 2 videos (2%)
  - Including necessary experiment results by *different parameters* (4%)

and your *thinking*(4%) for how to produce high quality motions.

Task 4 - Report

- PDF format, no page size requriment so you can also prepare it with powerpoint or keynote
- The first two lines should introduce your NAME and UID.
- Including necessary experiment results from **different parameters** (4%) and your **thinking**(4%) about how to produce high quality motions