Report

Gait Analysis and Pose Estimation in Unity

Brentan Simone 239959

Introduction

The general outline of this project involves first the computation of the gait analysis on a simulated animation and then the analysis of the gathered data. Therefore, for convenience, the structure of the project can be divided into two main components: the Animator and the Analyser.
The Animator's goal is to offer a way to simulate and display a virtual animation, whatever it may be, and recognize the person's position in any of its frames. Through the combination of *Unity* as a 3D animation tool and the *OpenPose* library it was possible to compute the pose estimation of a simulated animation.
After gathering the necessary posture data, the Analyser aims to offer a way to perform gait analysis on the collected information. This was possible through the usage of a set of charts to plot the joint's position of the animation target over time or in relation to each other.

Technology

The technologies involved in this project were: Unity, OpenPose and FastAPI.
First of all, the Unity engine was chosen to be the simulation tool for its simplicity and versatility, offering a good amount of features and customizations. A Unity Scene was created so that an animation of a walking person could be imported and simulated within. Next, the code available in the Unity Scene was combined with the OpenPose library in order to compute the gait analysis. This conjunction of tools permitted the extrapolation of the multiple joints positions of the simulated walking person. Afterwards, the retrieved gait figure was drawn in the Unity Camera, the one the user actually sees, so that it could display the person's position that the algorithm detected.
For the *Analyser* role instead, during the process of establishing the best tool to use, a lot of options were considered. In the end, the chosen approach involves the usage of a small web application through the definition of simple APIs. The FastAPI framework for Python was chosen for its simplicity and efficiency in both development and execution.
Other two additional worthy mentions, when talking about the technology used, are Mixamo and Plotly. Mixamo website, maintained by *Adobe*, offered a good list of walking animations and skins for the simulation generation. Plotly, instead, is the library that was used to plot the gait analysis data inside the web application, through a simple and powerful interface.

Animator Component

The main components of the Unity scene, leaving aside the ones used for rendering, are:

- **Animation** controller, which is the one responsible for the correct execution of the walking animation, also controlling its speed
- **OpenPose** object, responsible for the interaction with external programs, such as the OpenPose library and the web application. Considering the custom scripts are all assigned to this component, it can be viewed as the main part of the *Animator*. During each iteration, it first sends the current frame to OpenPose and retrieves the detected posture, and then sends the extracted joints positions to the web application
- **Canvas** object, which is basically responsible for controlling and displaying the menu during the simulation. It allows the user to edit settings such as: the activation of each joint, the animation speed, the background image toggle and other OpenPose neural network settings

Analyser Component

The purpose of the FastAPI web application is to retrieve the positions of simulation joints, save this data into a document, and perform an analysis on it. In order to effectively manage this process, the application was divided into two pages:

- The **first page** manages the interaction with Unity and the retrieval of real-time data. It shows a preview of the data charts from the obtained points and offers an interface for saving such data into a document. It also allows you to stop or restart the stream of data or refresh the charts
- The **second page** manages the actual gait analysis. The page allows you to select a document, which was previously saved from the real-time page, and open it in order to plot its data. Each plot is made with the Plotly library, which offers all the controls for panning and zooming the graph to better analyse its data

Evaluation

For the performance evaluation of the project, the results are **satisfactory** and the program is able to analyse around *6 to 9 frames per second*. The OpenPose library needs a *NVIDIA* graphics card for its Cuda drivers. Even though it can also work with only the CPU, the performance significantly drops, analysing only 0.5 frames per second. Also the *Analyser* is quite performing considering it runs locally and uses a light and fast framework.

Conclusions

This project successfully integrates Unity, OpenPose, and FastAPI to perform gait analysis on simulated animations. The Animator component effectively simulates and captures gait using Unity and OpenPose. The Analyser component, built with FastAPI, efficiently manages and analyses the collected data through Plotly charts. Performance is also satisfactory, with analysis rates of 6 to 9 frames per second using a GPU. Overall, the project demonstrates the potential of combining these technologies for detailed gait analysis, with future improvements focusing on performance and expanded capabilities.