Development of a Pytorch-based Kinematic Engine to Facilitate the Use of Realistic Joint Structures in Machine Learning Applications

Claire V. Hammond[1], Wan M. R. Rusli[1], Ryo Ueno[1]
[1]Department of Research and Development, ORGO Inc., Sapporo, Japan
Email: cvhammond@orgo.co.jp

Summary

Significant breakthroughs in machine learning-based markerless motion capture research have allowed for improved accuracy in pose estimation from monocular video. Many of these methods use kinematic joint structures with extra non-anatomical degrees of freedom (DOFs). This study introduces a novel kinematic engine and model built from Pytorch-compatible data structures to facilitate future machine learning-based pose estimation algorithms [1].

Introduction

Pose estimation from monocular video utilizing markerless motion capture facilitates the movement analysis of human subjects without the need for direct calibration or advanced hardware technology. Typically, during the training of machine learning models for monocular markerless motion capture, subjects are represented by values describing their pose and body shape. The most commonly used kinematic model, SMPL, describes each joint with three DOFs [2]. As a result, non-anatomical poses may occur during both model training and model inference. This study introduces a novel automatically differentiable kinematic engine and model built from Pytorch-compatible data structures to facilitate future machine learning-based pose estimation algorithms.

Methods

Python code was developed utilizing the Pytorch library to build an engine for creating kinematic models containing bodies, joints, and markers. Each joint is defined by twelve parameters describing the transformations of the parent and child bodies. Each body is defined as having a single joint connecting to a parent body. The root body can be defined as having either zero or six DOFs. Markers are defined by their position in the reference frame of their attached. Models include a full-body scale factor and individual body segment scale factors to allow for subject-specific modeling.

Forward and inverse kinematic methods were also developed. Input marker and pose data and all intermediate calculations are computed using tensors containing time-varying data, allowing all time states to be solved simultaneously. The inverse kinematics method minimizes squared distance errors between experimental and modeled markers. It uses the Pytorch LBFGS function to optimize pose, full-body scaling, individual segment body scaling, and marker positions using Pytorch's built-in automatic differentiation.

The kinematic engine was used to perform inverse kinematics on a lower body model with a six DOF root body and six DOFs per leg (three hip, one knee, two ankle). Marker motion data of a walking motion from the CMU Graphics Lab Motion Capture Database was selected [3]. Pelvis markers were allowed to move in the X and Z axes and the full body scale factor and individual body scale factors were allowed to change during the optimization. The resulting kinematics were compared to a matched kinematic model in CuSToM with similar marker motion and body scaling functionality [4]. Forward kinematics was then performed to determine the marker distance errors between the experimental marker data and the inverse kinematics-derived motion.

Results and Discussion

The novel Pytorch-based kinematic engine showed high kinematic accuracy relative to CuSToM for all lower body coordinates except for the left ankle varus/valgus angle. After performing forward kinematics, the Pytorch-based kinematic engine's marker distance errors relative to the experimental marker data were 7.4mm while CuSToM's marker distance errors were 9.1mm.

The kinematic disparity may be due to differences in optimization methods between the two kinematic engines. CuSToM optimizes marker positions and individual segment body scale factors prior to inverse kinematics while the method presented here optimizes these components during inverse kinematics. Notably, the Pytorch-based kinematic engine showed lower marker distance error than CuSToM.

Conclusions

A novel kinematic engine utilizing Pytorch's tensor data structure and native automatic differentiation was developed and showed similar accuracy to existing kinematic engines.

References

[1] Paszke, A et al. (2019). *Adv. NIPS*, 8024-35.
[2] Loper, M et al. (2015). *ACM Trans. Graph*, **248**: 1-16.
[3] CMU Motion Database. *http://mocap.cs.cmu.edu/.*
[4] Muller, A et al. (2019). *JOSS*.

Table 1: Average lower body kinematic errors by coordinate between the novel Pytorch-based kinematic engine and CuSToM.

	Hip Flexion	Hip Adduction	Hip Rotation	Knee Flexion	Ankle Varus/Valgus	Ankle Flexion
Left (Degrees)	3.3	2.2	1.0	2.8	9.3	1.1
Right (Degrees)	3.0	2.5	0.6	2.4	3.9	0.5