Robotics - Final Project

3D Motion Imitation in Robotics

LanGe Chen
Yuxuan Zhu

Course Organizer: Dr. Erwin M. Bakker

Leiden Institute of Advanced Computer Science (LIACS)
Faculty of Science
Universiteit Leiden

May 31, 2021

Abstract

In this project, we present an implementation of 3D Pose Imitation on virtual robots, which enables the NAO robot to imitate motions from recorded videos or real-time motions from camera. For this task, the sample images are given an OpenPose integration in Unity using ONNX and the Barracuda library, and estimates the 3D joint locations of the skeleton. Those joint locations of the avatar are then mapped to those of the NAO robot in the virtual scene. Finally, the joint positions are used as inverse kinematics targets in order to achieve an imitation of the original movement on the robot model, including joint limits, different bone lengths and body proportions. The results demonstrate that the robot can achieve a plausible motion imitation based on the input videos.

1 Introduction and Motivation

The interaction between robots and human has attracted many interests in humanoid robotics research. However, it is easy for human to complete a given task by observing how other people perform it, while robots are usually programmed to learn by imitating humans. To achieve human-like gestures for robots, controlling the motion of a robot remains a challenging task. The advances in computing nowadays opens the door for many exciting AI applications in human-robot interaction and motion imitation. A particularly strong interest in AI is to extract 3D body information from 2D image content via computer vision techniques, which can be used as features for learning or optimization frameworks [3]. The main focus of this project is to enable the NAO robot kinematically imitating the motion of a human in a video clip reference, which requires reproducing similar movements with different joint limits and body proportions for the robot than in the reference. In computer animation and robotics, inverse kinematics is a widely used approach which computes the joint values that will cause the end effectors to reach a desired goal state, for example a robot manipulator arm reaching a given position and orientation relative to the root of the chain. Using deep learning, 3D joint location estimates of human skeletons can be computed from 2D video data. By combining those techniques, a real-time framework for kinematic robot motion imitation was developed in this project.

2 Project Design and Implementation

This section describes the implementation of the different components to built the 3D motion imitation pipeline (see Fig. 1). The project is using Unity as the virtual environment and visualization. In our framework, given the raw video data as input, the reference joint positions are computed via OpenPose [1], then retargeted to the proportion and space of the robot, and finally solved using BioIK [5] to optimize a pose in joint space to produce a robot pose that follows the motion in the reference video.

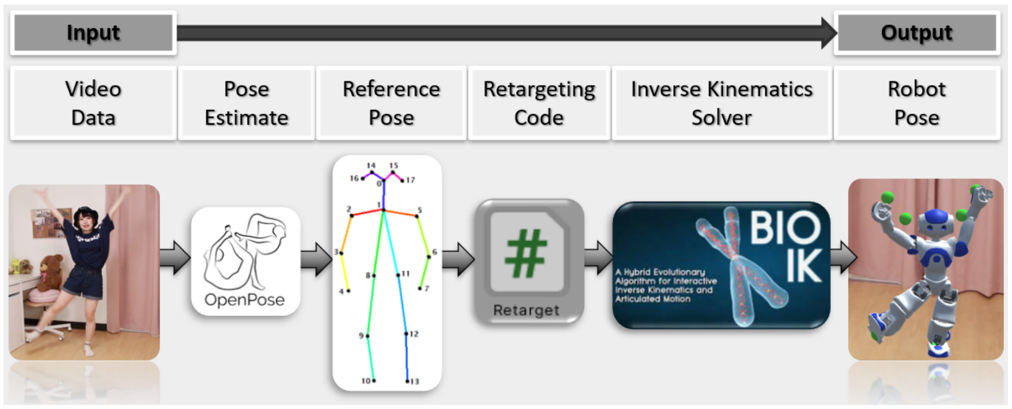
Input
Output
Video Data
Pose Estimate
Reference Pose
Retargeting Code
Inverse Kinematics Solver
Robot Pose
OpenPose
Retarget
BIO IK
A Hybrid Evolutionary Algorithm for Interactive Inverse Kinematics and Articulated Motion

Figure 1: Implemented 3D Motion Imitation Architecture.

2.1 Integrating the OpenPose Plugin for *Pose Estimation*

In this project, the OpenPose is integrated to detect human joints on single images. OpenPose [1] is a popular human pose estimation library in Unity, and it has been the first framework demonstrating the capability of jointly detecting human body, face, and foot keypoints. The detection pipeline of OpenPose is as follows:

- First, an input RGB image is fed into baseline Convolutional Neural Network(CNN) to extract the feature maps of the input.
- Then CNN will produce two outputs. One branch predicts the confidence maps of body parts like left elbow, right foot and so on. The other branch predicts the affinity fields which represents the degree of freedom between different joints.
- Finally, the confidence maps and affinity fields that are genearted above are being processed by greedy bipartite matching algorithm to obtain 2D key points as outputs for people in the input image.

It is a stable and suitable tool for real-time single- and multi-person keypoint detection, and we will use ThreeDPoseUnityBarracuda [7] implementation which computes an ONNX network model using Barracuda to do 3D pose estimation in Unity. In this sample, the avatar can perform similar movements as the person in the video in real time, estimating the 3D joint locations, excluding rotations. Those positions are in format of the VNect character avatar, and which mapping to the NAO robot model will be described in next section.

2.2 Goal retargeting from VNect avatar to NAO robot

The NAO robot is imported using its URDF (Universal Robot Description Format) [6] file, which is a standardized way of describing properties of robots. URDF files are XML files that allow us to specify these visual, collision, and physical properties in a human-readable markup language. To import robots into Unity, we use the open-source Unity package *URDF-Importer* [4], which returns a kinematic robot model links and meshes along the hierarchy. Since NAO robot is one of the most popular robots with an endearing appearance and a kinematic hierarchy similar to those of humans, we decided to use NAO robot for the motion imitation task. The retargeting workflow includes the following steps:

- First, we used the *nao.urdf* file to import the robot model with 22 DOF into the Unity scene (see Fig. 2). The URDF-Importer parses the XML file and stores the links, joints and motion axes with their respective lower/upper limits. A hierarchy of GameObjects is then created.

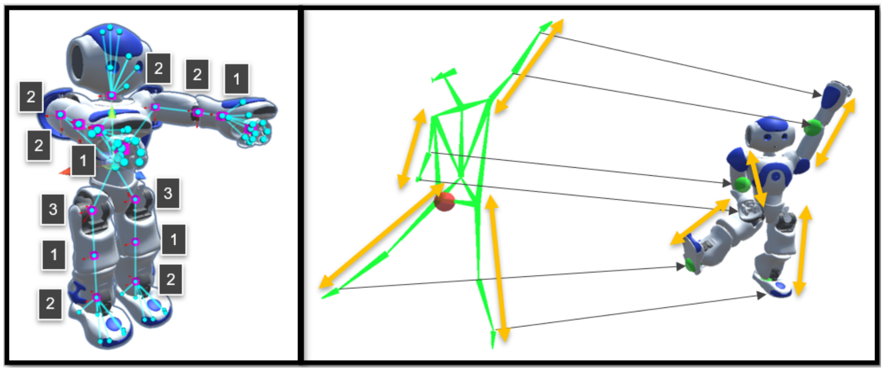

Figure 2: 22 DOF NAO model with joint mapping and automatic target rescaling based on bone lengths.

- *Retarget.cs* script is created for mapping joints from the avatar to the robot. For better accuracy in solving inverse kinematics, the root (hip) of NAO is first aligned with the hip of the VNectModel in the world space. We then map the key joint positions of *LeftHand*, *RightHand*, *LeftElbow*, *RightElbow*, *LeftFoot*, *RightFoot* to the NAO robot and use them as targets for the joints of NAO.

- Considering the body proportions between the VNectModel avatar and NAO model may cause difficulties during retargeting process, as we need to ensure that the robot can actually reach the given targets. A global rescale parameter may not consider different lengths of individual limbs, such as arms or legs. Therefore, we implemented functions to compute the chain length of the robot and the chain length of the avatar for each target joint to the root, and from that get a rescaling factor for each limb using the follwing equation which helps to properly adjust the target position (see Fig. 2).

$$rescale = \frac{NAO.ChainLength}{Skeleton.ChainLength}$$

2.3 Using Inverse Kinematics to solve the *Target Position*

Inverse kinematics is the process to determining the pose of a robot to reach a desired goal location, or more generally to find the optimal joint values $\theta = f^{-1}(\mathcal{X}_{1,\ldots,k})$ for a set of objectives $\mathcal{X}$, where f is the forward kinematics function of the robot. In this project, given the 3D joint position estimates from OpenPose, we use the BioIK [5] solver to find the joint values within the limits of the Nao robot. BioIK implements a fast and robust evolutionary algorithm, which can solve fully constrained generic inverse kinematics with multiple end effectors and goal objectives. Thus, accurate solutions for full-body inverse kinematics can be found in real-time while avoiding local minima where alternative approaches, such as Jacobian-based methods, would get stuck in joint limits and returning inaccurate results. The algorithm of BioIK requires two parameters for the number of individuals and elites to be set, which we choose as *individuals=25* and *elites=1*.

3 Experiments and Evaluation

In this section, we evaluated the performance using BioIK to let the robot follow the target locations provided by OpenPose. We further integrated a CCD (Cyclic Coordinate Descent) solver, which is another well-known algorithm used for inverse kinematics to operate on multi-joint chains. As shown in Fig. 3, the CCD solver leads to poor performance compared to BioIK, and the motion of NAO looks unreasonably broken which is far away from imitation. This could be due to poor incorporation of joint limits in CCD because it does not directly operate in joint space. Instead, it computes a rotation quaternion to update the robot in Cartesian space directly. The NAO robot using BioIK achieves much higher accuracy and a good result, producing

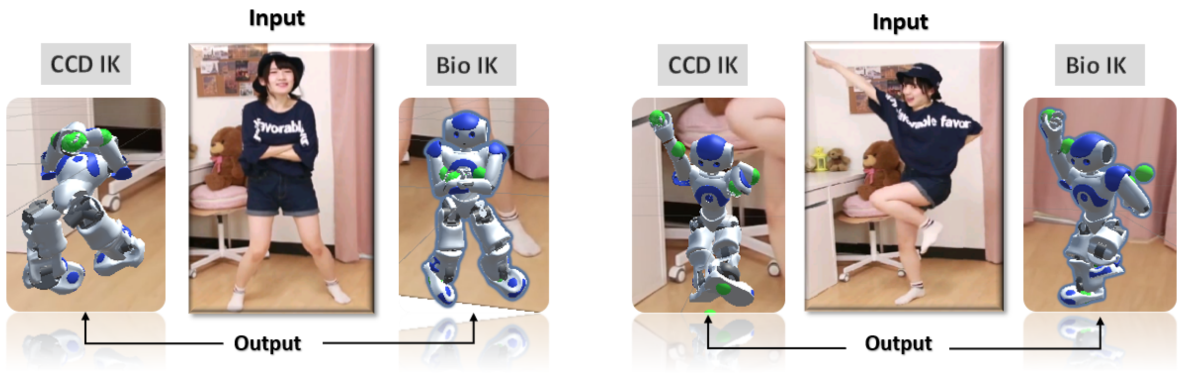

Figure 3: Example results of our framework to produce robot poses from video.

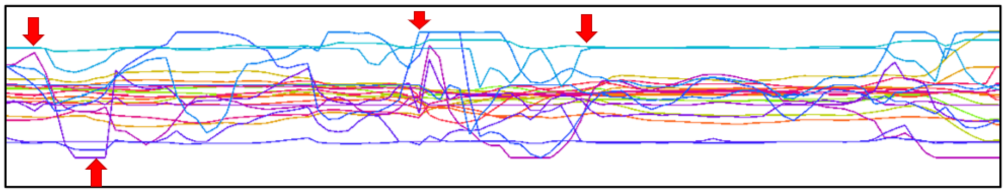
Figure 4: Plot of joint values over time during following the motion.

a motion that looks similar to the video input pose. However, we found our setup sometimes produces unrealistic end effector rotations as only position targets are given, as well as oscillations. Thus, we plot the joint values (y axis) over time (x axis) with the aim to observe at which situation motion artifacts are occurring. As shown in Fig. 4, at some moments there are joint values (marked with red arrow) suddenly jumping from one solution to another, mostly after being constant for a while (which means they have been at joint limits).

4 Conclusion, Discussion and Future Work

In this project, we achieved enabling the NAO robot to follow a human motion from video in real-time. To achieve this task, we combined 3D pose estimation using neural networks, evolutionary inverse kinematics, and retargeting operations to map the goal joint locations from the virtual avatar to those of the robot. Generally, we found inverse kinematics to be suitable tool for this task, but it can be challenging to produce the expected pose as there can typically be several solutions equally possible. In particular, defining more target objectives (such as elbows instead of only hands) helps constraining the solution space and to produce more similar poses, but providing too many can lead to overconstraining the objective function and inaccurate poses. Thus, defining the number of objectives is a trade-off between stability and optimization. Based on results, however, the most obvious problem is the foot contact, which is caused by missing joint rotation information. Estimating and resolving such rotation objectives would further improve the pose quality. Finally, inspired by [2], it may be possible to transform the motions from kinematic reference to physically-simulated movements and then to real robot via deep reinforcement learning. This is a challenging but very interesting direction to work on in the future study.

References

[1] Zhe Cao et al. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields". In: *arXiv preprint arXiv:1812.08008*. 2018.

[2] Xue Bin Peng et al. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: *Robotics: Science and Systems*. July 2020. DOI: 10.15607/RSS.2020.XVI.064.

[3] Xue Bin Peng et al. "SFV: Reinforcement Learning of Physical Skills from Videos". In: *ACM Trans. Graph.* 37.6 (Nov. 2018).

[4] *Robotics simulation in Unity is as easy as 1, 2, 3!* URL: https://blog.unity.com/technology/robotics-simulation-in-unity-is-as-easy-as-1-2-3.

[5] Sebastian Starke, Norman Hendrich, and Jianwei Zhang. "Memetic Evolution for Generic Full-Body Inverse Kinematics in Robotics and Animation". In: *IEEE Transactions on Evolutionary Computation* 23.3 (2019), pp. 406–420. DOI: 10.1109/TEVC.2018.2867601.

[6] Ioan Sucan and Jackie Kay. *urdf.* URL: http://wiki.ros.org/action/fullsearch/urdf?action=fullsearch&context=180&value=linkto%3A%22urdf%22.

[7] *ThreeDPoseUnityBarracuda.* 2019. URL: https://github.com/digital-standard/ThreeDPoseUnityBarracuda#threedposeunitybarracuda.