Analyzing the Performance of 3D Pose Estimators to Optimize Humanoid Robot Control

CS231N Project Report Spring 2024

Ethan Whitmer
Stanford University
Dept. of Mechanical Engineering
erw12@stanford.edu

Abstract

This project aims to analyze the ability of current 3D Pose Estimators to accurately control the end effectors of a humanoid robot in 3D space using a single 2D (monocular) video feed. Two models (VideoPose3D (pre-trained), and a Baseline Neural Net) were used on two datasets (HumanEva and a small self-collected dataset) to predict 3D joint locations based on pixel space joint key-points. The predictions of these models were fed into three controllers, one that just controls for the position of the hands (baseline), one that uses task Null-Space to control the angles of the robot's joints, and one that uses task Null-Space to control the position of the robot's joints.

Overall the third controller performed remarkably better than the first two, indicating that secondary joint position data plays a key role in controlling a humanoid robot. The baseline models outperformed the pre-trained VideoPose3D model on both datasets, especially on the self-collected dataset. This was largely due to the baseline's increased ability to accurately perceive movement directly at the camera, something that other models struggled with. The Base model combined with the Null-Space position controller was able to achieve a mean hand position error of 0.0541 meters on the HumanEva test video, and the baseline model, combined with the Null-Space angle controller was able to achieve a mean hand position error of 0.0808 meters on the self-collected test video.

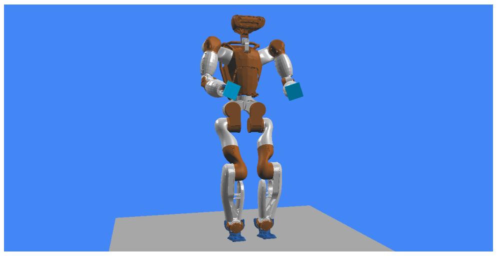


Figure 1. Toro Robot in Simulation

1. Introduction

Human Pose Estimation has been a longstanding problem in the fields of Computer Vision and Robotics. In recent years, strides have been made to train models that can accurately predict the 3D positions of human joints from video, however, accurate pose estimation is just one part of the robotics pipeline. This project will focus on the larger problem of mapping a human's movements onto a robotic avatar using just a 2D monocular video feed. Such a mapping would allow for efficient, accurate teleoperation without the need for haptic devices, camera arrays, wearable trackers, or other specialized cameras, greatly increasing the convenience of teleoperation setups.

Specifically, this project analyzed and evaluated the use of 2 models, a current state-of-the-art pose estimation model[1][2] and a baseline feed-forward neural net[3], to control the motion of the Toro [4] robot in simulation (Fig 1). Models were evaluated on how well Toro's end effectors (hands) were able to track the movement of a person in a video using three different control schemes (described below). The overarching goal of this project is to evaluate what the most important aspects of a model are for humanoid control. Note that only upper body control was considered for this project in order to avoid the complicated problem of humanoid robot locomotion.

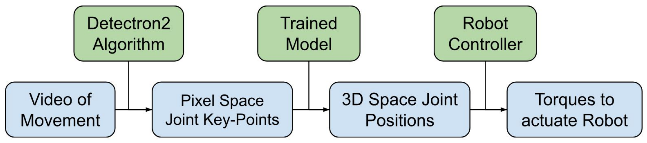


Figure 2. Video to Robot Pipeline

There are 3 main processing steps within this project's pipeline (Fig 2), the first takes a monocular video of a single subject as input and passes it through a pretrained Detectron2 algorithm[5], which outputs a temporal array of pixel locations of the subject's joints (x, y coordinates in pixel space of each joint for every video frame). These joint key-points are then passed through the model (VideoPose3D[2], or the baseline[3]), these models then output the predicted 3D locations of each joint (x, y, z coordinates in world space of each joint for every video frame), which are transformed into Toro's reference frame and then used to control the robot's hand (end effector) positions.

2. Related Work

2.1. 3D Pose Estimation

There are many different strategies for estimating the 3D position of the joints of a subject. [1] offers an excellent overview of the near-current state of the field. There are two main categories for pose estimation, the first is a model that processes a single frame of footage at a time, it has no access to previous or future data points in a video. This project uses one such model as described in [3] as a baseline (figure 3 above shows the architecture). This model trains and predicts on already extracted 2D key points. Some other models, however, train and predict directly on the image data. They typically use a CNN [6] or GCN (Graphical Convolutional Network) [7]. This can be an effective method as some information is lost when 2D key-points are extracted, but the compute required for such models is much larger. In particular, GCNs show promise as human poses are often represented using a skeletal framework (see figure 3), this skeletal framework is a graph, so using a model designed for graphs makes sense.

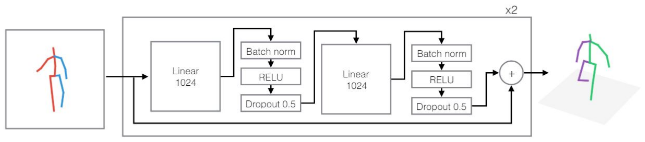


Figure 3. Baseline Model Architecture

The second main category for pose estimation contains models that use a time sequence of data to train and predict. This project evaluates the use of one such model [2], that performs convolutions across the temporal axis to make predictions on an entire sequence of data at once. This is in contrast to other methods that use recurrent networks such as LSTMs [8]. Temporal convolutions are nice because they do not suffer from the same vanishing/exploding gradient issues that commonly occur in recurrent models, but they suffer in practicality because they often lack the ability to make real-time predictions on data. LSTMs on the other hand can perform very well on time-sequenced data without needing access to future data. In particular [9] makes use of two LSTMs in parallel to predict the 2D Pose skeleton of a subject, and local image patches of each joint, and then integrates both sets of features to make a final 3D depth prediction.

2.2. Humanoid Robotic Control

Teleoperational control of a humanoid robot also has many different strategies, [10] offers a survey of many current popular methods. Much of humanoid teleoperation is done using a haptic or exoskeleton system [11]. These types of controllers are great because they can provide very accurate smooth control, but they are often expensive and require extensive setup. [12] and [13] both use teleoperation for dexterous manipulators using a monocular video feed. They found success using CNNs but did not scale the system for full-body motion. [14] Discusses learning on monocular vision to teach efficient full body control, they achieved some success but were severely limited by the vision-based pose estimators of the time (2007). Many vision based teleoperation systems these days use either a depth-equipped camera [15], or Motion Capture technology [16].

3. Dataset and Features

3.1. HumanEva

Two datasets were used for training and evaluation in this project. The first was HumanEva I [17]. This is a video dataset consisting of 10-15 second clips of subjects performing various motions in a room. There are 3 different camera angles for each motion, all recording in 640x480 resolution at 60 FPS. For each video, there is motion capture data that gives the ground truth 3D positions of 20 markers placed on the subjects, these markers are placed at each of the subject's joints as well as a few extra on the torso.

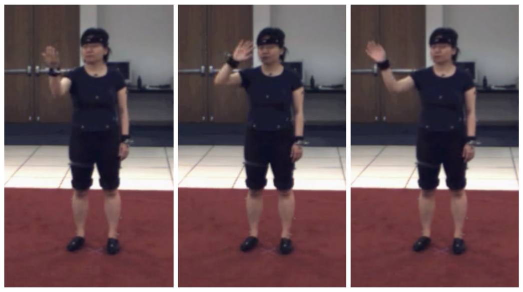


Figure 4. Frames from HumanEva Test Sequence

Overall there were 27769 available frames, A 14-second clip (796 frames) was chosen for evaluation with Toro, and the rest of the frames were split randomly into training (21074) and validation (5269) using an 80/20 split. The clip chosen for testing is of Subject 1 standing still, waving, and beckoning with her right hand while her left hand lies at her side (Figure 4). This clip was chosen because the subject's body remains still throughout the duration and has some interesting hand movements for Toro to try and replicate. Joint pixel key-points are provided for every tracker so a model trained on this dataset has up to 40 (20x2) potential input features, and up to 60 (20x3) potential outputs.

3.2. Self Collected

The second dataset was one that I collected and prepared myself. It has 2269 available frames, 550 were used in a clip for evaluation with Toro (described below), and the remaining frames were split randomly into training (1547) and validation (172) using an 90/10 split. Everything was shot from a single camera angle/relative position on an iPhone 11 Pro (720p, 30 FPS). In the videos, I perform a variety of upper-body movements while keeping my lower body relatively stationary. While I perform these motions, an Optitrack system[18] measures the real-time location of 3 trackers, one on each of my hands, and my head, in the room. This data is then transformed to be within the Toro's frame of reference, an advantage of this is that predictions from models trained on this data can be fed directly into a controller. These measurements were then carefully synced with the video and sampled at each video frame to produce ground truth measurements for this dataset.

The purpose of this dataset is to evaluate the performance of the system for a specific robotic control scenario. Many pose estimation models are trained to be robust against a variety of camera positions, angles, and resolutions, for robotic control, however, there will likely be a specific setup the user has with a relatively consistent camera position. The question then becomes, can a simple model trained on even just a few minutes of footage from this specific scenario outperform a larger more general model?

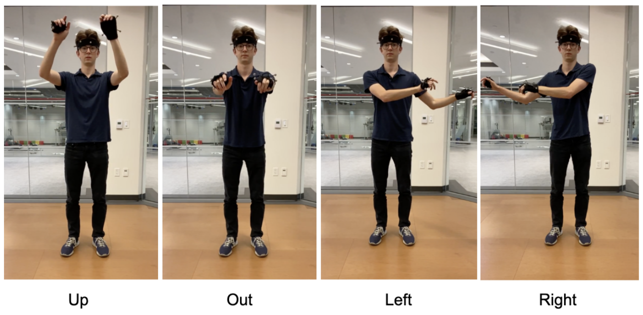


Figure 5. Frames from Self-Collected Test Sequence

The test sequence for this dataset is 18 seconds long and consists of me raising my arms up, then out, left, and right, and then moving them in a circle. Figure 5 shows frames at each of the 4 (up, out, left, right) extremes. This test video was chosen to evaluate the system's performance in each of the cardinal directions and for general upper body/arm control. Note that the existing Inference API in [2] was used to process videos into 2D joint key-points via Detectron2. This produces 17 joint key-points per frame, so a model trained on this dataset has up to 34 (17x2) potential input features and up to 9 (3x3) potential outputs.

4. Methods

4.1. Control Schemes

Three control schemes were implemented for this project. The first (Eqn. 1) is the most simple (a baseline) and represents a simple operational space PD controller. Here x_d represents the desired hand position (model prediction), x and $\dot{x}$ represent the position and velocity of robot's end effector (hand), k_p and k_v are positional and damping gains, J_v is the robot's jacobian at the end effector, and Γ is the vector of torques to apply at each joint (note that these equations are simplified and do not include the mass matrices, Coriolis force, or gravity force compensators that are used in control). This scheme is convenient for its simplicity but can result in inefficient poses that limit motion.

$$\Gamma = J_v^T(k_p(x_d - x) - k_v\dot{x}) \tag{1}$$

The second control scheme (Eqn. 2) is similar to the first, but attempts to keep the posture of the robot as "neutral" as possible. Controlling the position of the robot's hands requires 6 degrees of freedom (3 coordinates per hand), Toro, however, has many more degrees of freedom than 6 (39 total). This controller still uses PD control to move the hands to the desired location, but now uses the extra degrees of freedom to attempt to keep the angle at each joint as close to 0 (neutral initial position) as possible. This helps prevent the robot from twisting up and ensures a wide range of motion is always available. Here q and $\dot{q}$ are the angles and angular velocities at each joint, q_d is the desired joint angle (0), and N represents the null space of the jacobian.

$$\Gamma = J_v^T(k_p(x_d - x) - k_v\dot{x}) + N^T(k_{pj}(q_d - q) - k_{vj}\dot{q}) \tag{2}$$

The third control scheme (Eqn. 3) is similar to the second, but instead of using null space control to keep the joint angles close to 0, it uses null space control to match the posture of the subject, specifically the head, shoulder, elbow, and feet positions. This controller is designed to evaluate the importance of human posture in robotic control. Robot arms are not human arms and therefore their ideal posture may differ from that of a human's. Here n_d are the perceived positions of the subject's shoulders and elbows, while n and $\dot{n}$ are the positions and velocities of the robot's shoulders and elbows. Note that because there was no ground truth data available for the self-collected dataset, this controller was not used with the baseline model trained on the self-collected dataset.

$$\Gamma = J_v^T(k_p(x_d - x) - k_v\dot{x}) + N^T J_v^T(k_p(n_d - n) - k_v\dot{n}) \tag{3}$$

4.2. Models

4.2.1 Baseline

Two models were used in evaluation for this project. The first is the baseline feed-forward neural net described in [3]. This model's architecture is shown above in Figure 3, it consists of 2 outer blocks, each containing 2 linear, batchnorm, ReLU, dropout combos in sequence, connected with a residual. Input and output coordinates are flattened for training.

I trained/tuned four of these models using PyTorch [19], two on each dataset. The models trained on HumanEva had an input field of 40 (20 joints x 2 coords) and an output field of 27 (9 joints x 3 coords), the output joints of interest are the 9 joints used by Controller 3. The models trained on the Self Collected dataset had an input field of 34 (17 joints x 2 coords) and an output field of 6 (2 hands x 3 coords).

For the two models on each dataset, a true baseline model and a modded baseline model were used. The true baseline models replicated the architecture and training procedure described in [3] they were trained using a standard mean squared error (MSE) loss function (Eqn. 4). For the modded baseline models, architectural parameters were allowed to vary during tuning, also a weighted (MSE) loss function (Eqn. 5) was used where w_i is an assigned weight associated with each output.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4}$$

$$\mathcal{L}_{\text{Weighted MSE}} = \frac{1}{N}\sum_{i=1}^{N}w_i(y_i - \hat{y}_i)^2 \tag{5}$$

For the Self-Collected dataset, a tunable weight parameter to every x (camera direction) coordinate output was added. During experimentation, models seemed to particularly struggle with accurately predicting movement that was directed at the camera. During such movement, the joint pixel key-points do not change much, so it can be difficult to discern any movement. Increasing the weight of these outputs should help the model combat this error. For the HumanEva modded baseline model, a tunable hand weight parameter that increases the weight of the hand coordinates was added. This helps ensure that the hand locations are as accurate as possible while still predicting useful positions for the other joints to be used by Controller 3.

4.2.2 VideoPose3D

The second model used in this project is the pre-trained VideoPose3D Model [2]. This model was trained on Human 3.6M [20], a dataset similar to HumanEva but much larger. This model uses dilated temporal convolutions to predict 3D joint positions from 2D joint pixel key-points. This strategy offers an advantage over the baseline model because it utilizes the inherent information stored in time-sequenced data. Given that human bodies move continuously the 3D location of a joint of one point can never be too far from the next, meaning that extra information may be learned by looking at previous and future data. This helps to boost accuracy but limits the model's ability to predict on real-time data as an entire sequence must be predicted on at once, that is, that is, the model must see the last frame of the video before predicting on the first frame. This is in contrast to an RNN or LSTM which do not require future data.

4.3. Model Output to Control Scheme

The last piece of this puzzle is connecting the model predictions to the controllers. Outputs from models not trained on the Self Collected dataset are in some unknown camera of world reference frame. In order to be useful for control, they must be transformed into Toro's reference frame. This involves 3 transformations, a scale, a shift, and a rotation. An identical procedure is used to transform both the ground truth and predicted positions for fairness in evaluation.

The initial positions of the subject's shoulders and head were used to perform these three transformations. To scale the data, every point was multiplied by d_T/d_i where d_T is the known distance between Toro's shoulder joints, and d_i is the measured distance between the initial shoulder positions. To shift the data, a vector representing the difference between Toro's head and the measured head positions was added to every data point. Finally, to rotate the data, every point was multiplied by the rotation matrix required to rotate the plane formed by the 3 shoulder/head points to be along the YZ world plane, this is done by finding the vector normal to the plane formed by the 3 points, and rotating it to be parallel to a vector pointing in the positive X direction.

This transformation works well but does make several critical assumptions. The first is that all humans are equally proportional, the data is scaled based on shoulder width which is not directly correlated to human wingspan, if a subject's proportions are very different than Toro, their hands may move outside of Toro's reachable workspace. This issue could be fixed by using a more detailed per-subject calibration phase before operation. A second assumption is that the subject's initial position has good posture, not facing away from the camera. That is, the shoulders are not twisted relative to the torso, the head is not bent extremely forward or backward, and the camera can see the subject's chest (otherwise 180°rotation errors may occur).

After the transformation, the predictions are smoothed using a sliding average with window size 4, alleviating jitters that tend to be inherent to model predictions and can be detrimental to robotic control. Finally, because the controller runs at a much faster frequency than the video frame rate, the model predictions are linearly interpolated between each frame to provide continuous smooth control.

5. Experiments and Results

5.1. Model Training

Dataset:	HumanEva		Self Collected	
Model:	Base	Base Modded	Base	Base Modded
Train MSE	0.00161	0.00268	0.00684	0.00227
Valid MSE	0.00295	0.00264	0.00756	0.00237
Test MSE	0.00329	0.00304	0.00976	0.00313

Table 1: Mean Squared Errors for Baseline Models

Shown in Table 1 are the mean squared errors for each of the 4 trained models on the training, validation, and testing sets. The Base HumanEva model followed the same procedure as described in [3], and was trained for 100 epochs, this shows clear evidence of overfitting as the train MSE is much lower than the Valid and Test MSE. For the Base Modded Model, L2 regularization was added, the batch size was lowered to 64, the learning rate was decreased, and the model was trained for 200 epochs. This clearly reduces the overfitting, resulting in better Valid and Test MSEs despite a worse training MSE.

Similarly for the Base Model trained on the Self Collected dataset, there are some signs of overfitting and altogether worse performance. For the modded version of this model L2 regularization was added, batch size was lowered to 32 to accommodate the smaller data set, and the model was trained for 200 epochs. The parameter space was also increased by adding an extra outer block in an effort to decrease bias. These changes helped improve performance massively, cutting down the MSEs by a factor of 3. For reference the VideoPose3D model performed slightly worse on the HumanEva dataset with an MSE of 0.00412, and quite a bit worse on the Self-Collected dataset with an MSE of 0.02268. Understandably though, considering it was trained on a separate dataset.

5.2. Controller Results

Shown in Table 2 are the mean per joint position errors (Eqn. 6) in meters for each model with each controller where N is the number of joints, f is the number of frames, and $P_j(i)$ is the position vector of the i^{th} joint during the j^{th} frame, here $N = 2$ as only the hand positions are being evaluated. The results are broken up into 3 categories to identify the main source of error. The first is the error between the model predictions and the ground truth predictions, this is essentially the square of the model's MSE loss post-transformation. The second category is the error between the model predictions and the actual positions of Toro's hands, this is essentially a measure of how well the controller is tracking the desired hand locations. Finally there is the error between the ground truth positions and Toro's actual hand positions, this is the overall metric for how well the system performs.

$$\mathcal{L}_{\text{MPJPE}} = \frac{1}{f}\sum_{i=1}^{f}\frac{1}{N}\sum_{j=1}^{N}||P_j(i) - \hat{P}_j(i)||_2 \tag{6}$$

It is clear that the majority of the error comes from the model predictions, this indicates that in the goal of accurate control, improving models should be focused on more than improving control schemes. One interesting observation is that while the Base Modded HumanEva model performed slightly better than the Base HumanEva model on the test set, post-transformation, the Base model performed slightly better (on the order of 3 millimeters). This indicates that the transformation process is not perfectly one-to-one, which makes sense as models may predict different shoulder/head initial positions than the ground truths, resulting in slightly different transformations being applied. Overall the best result on the HumanEva test clip was on the Base model using Controller 3, with around 5 centimeters of average error, and the best result on the Self-Collected test clip was the Base Modded model using Controller 2.

Dataset:	HumanEva			Self Collected		
Model:	Base	Base Modded	VideoPose3D	Base	Base Modded	VideoPose3D
Ground Truth vs. Preds	0.0504	0.0530	0.0643	0.1116	0.0740	0.1506
Contoller 1 Toro vs. Preds	0.0163	0.0151	0.0133	0.0383	0.0301	0.0353
Contoller 2 Toro vs. Preds	0.0317	0.0427	0.0262	0.0409	0.0284	0.0281
Contoller 3 Toro vs. Preds	0.0176	0.0130	0.0131	N/A	N/A	0.0331
Contoller 1 Toro vs. Ground Truth	0.0546	0.0565	0.0644	0.1154	0.0836	0.1540
Contoller 2 Toro vs. Ground Truth	0.0765	0.0651	0.0801	0.1164	0.0808	0.1496
Contoller 3 Toro vs. Ground Truth	0.0541	0.0555	0.0641	N/A	N/A	0.1517

Table 2: Mean Per Joint Error Across Models and Controllers (meters)

The best controller/model combo for each test clip can be viewed by following the video links below.
HumanEva Test Clip,
Self-Collected Test Clip

On both the HumanEva dataset and especially the Self-Collected dataset the trained models tended to perform better than the pre-trained VideoPose3D model. On the Self-Collected dataset, a large part of this was due to an increased ability to perceive motion directed at the camera. In Figure 6, you can see that when I stick my arms straight out in the video (increasing the x coordinate), there is practically no response from the VideoPose3D model, whereas, in Figure 7, there is a clear response to drastic x coordinate motion by the Base Modded Model. This increased response is likely due to a combination of the added weights, training on a single camera angle, and using a small dataset (the model can ”overfit” to the specific camera setup).

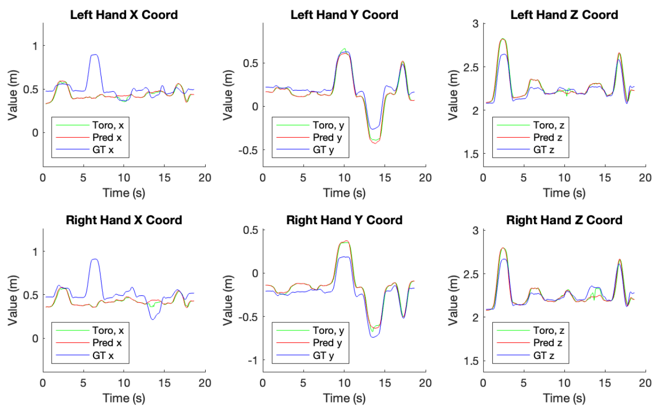


Figure 6. Hand Coords for VideoPose3D Self-Collected Test Clip

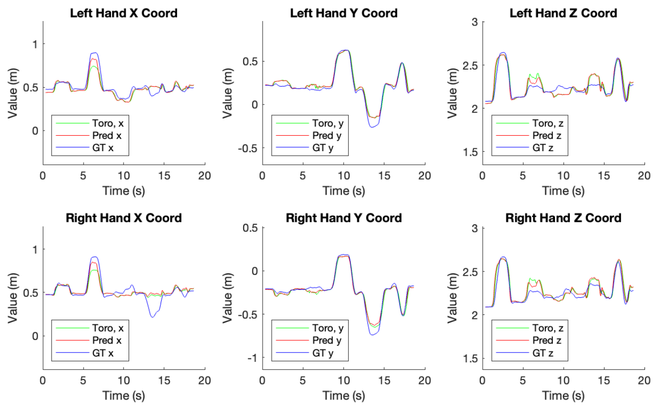


Figure 7. Hand Coords for Base Modded Self-Collected Test Clip

In terms of controller performance, Controller 3 performed the best across the board, providing mean errors between 1 and 3.5 centimeters. This was closely followed by Controller 1, which averaged slightly worse, followed by Controller 2 which averaged between 2.5 and 4.5 centimeters in average error. These levels of error are decent for most control tasks, however, tasks that require a greater deal of precision, such as operating a keypad, would likely require a fine-tuned controller. It is important to recognize that Controller 1 is not typically viable to use in a real world environment as it often results in the robot contorting its joints in strange ways (see Figure 8). This is not a problem in simulation as the base joint is fixed in space and not subject to gravity, but would not work well in the real world. For real-world control, something like Controller 2 is often used, this does increase error as some control torque is now being used to control joint angles, but it does help to avoid singularities and kinked limbs. At a bare minimum, some task should be assigned to control all degrees of freedom.

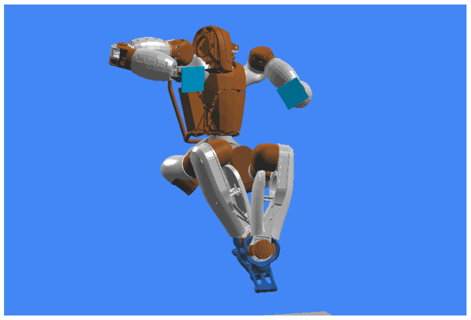
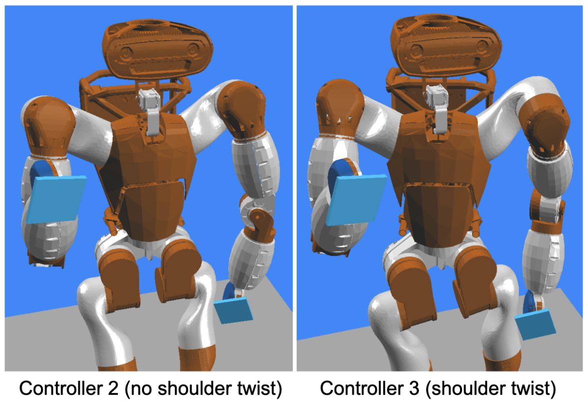


Figure 8. Contorted Toro Position while using Controller 1

The results demonstrate that a positional null space controller such as Controller 3 can perform better than a angular null space controller. This indicates that the natural motion of our bodies when humans move their hands is useful information to provide to a robot. Such information is much more difficult to acquire, but it appears that when possible, models for pose estimation should ideally be trained on data containing these secondary joint positions even if said joints are not immediately required for the task. An excellent example of the advantage this offers is shown in Figure 9.


Controller 2 (no shoulder twist)
Controller 3 (shoulder twist)
Figure 9. Toro Waving using Controller 2 and Controller 3

The two Toros waving in Figure 9 are both using the Base Modded Model to predict on the HumanEva test clip. When using Controller 2, Toro keeps its shoulders perfectly aligned with its pelvis during this motion as the Null Space control is instructing it to. When using Controller 3, however, Toro twists its upper body slightly moving the shoulder connected to the waving arm back. This is a very subtle subconscious motion that we humans tend to do when we lift one of our arms (try lifting one arm naturally with your elbow bent and observe the motion of our shoulders), such motion makes it much easier to perform motions with our raised arm (try tensing your shoulders, waving with one arm, then relaxing your shoulders and waving), by moving the base joint (shoulder) of our arm back, which in turn decreases the amount that we have to bend our elbow, allowing it a more natural position. It is a small motion that makes a large impact that would be quite difficult to hardcode into a controller, but our model provides the data to do this automatically when it predicts the shoulder joint locations of the subject. It is many small things like this that allow Controller 3 to outperform Controller 2 and demonstrates the importance of driven secondary joint control.

6. Conclusion and Future Work

6.1. Conclusion

Overall the baseline models trained on small datasets outperformed the larger pre-trained model. This is indicative of the importance of camera angle, position, and subject when it comes to pose estimation. The pre-trained model still performed remarkably well, a testament to its robust nature. Slight modifications to the architecture and training procedure described in [3] decreased overfitting and allowed for slight improvements on the HumanEva dataset, and significant improvements on a small Self-Collected dataset. In particular, the ability to better predict movement toward and away from the camera greatly aided these model's performance.

A control scheme that controlled for the position of the hands as a primary task and used that task's Null Space to control the location of secondary joints outperformed a base controller with no Null Space Control and a controller that used Angle-based Null Space Control. This highlighted the importance of secondary joint data as a way to encode the natural movement of a human body. The best controller for the HumanEva test clip had an average mean positional error of around 1.76 centimeters which, when coupled with the base model produced an overall average error of 5.41 centimeters. The best controller/model combo on the Self-Collected test clip had an average error of 8.08 cm. This level of error are acceptable for rudimentary control tasks, but any detail-oriented control tasks would likely require a more fine-tuned controller and better-performing model.

6.2. Future Work

There is a lot more work that I would have loved to do on this project given more time/resources. To start with, there were many models that I wanted to test but just didn't have time to implement and or train. Specifically using an LSTM such as the one described in [21] is something I would be very interested in. Robotic control is inherently a time-sequenced problem, there are no abrupt jumps or discontinuities that should in smooth robotic control. Employing a recurrent model that capitalizes on this time sequence would likely be a good strategy to minimize error. Also training a CNN or recurrent model to predict joint positions directly from video data such as described in [6] would be very interesting I think. It is great that 2D key-points allow such accurate prediction because they make our models much simpler, but I imagine there must be some important information for 3D pose prediction in the frames themselves that is lost when converting to 2D key-points. Also finding a better transformation method from predictions to world space would be very interesting, using frame-by-frame transformations, or even trying to learn a transformation to apply are two possible methods.

Another very interesting avenue is orientation control. When attempting to use a robotic end effector, the position is only half of the battle (and usually the easier half). Controlling the orientation of an end effector is just as important in order for it to be able to properly interact with its environment. Orientation prediction is not nearly as widespread as position prediction, although one key example comes from [22]. This is largely due to a lack of well annotated data, many pose datasets only use a single tracker per joint to track motion meaning only positional data is available, the nice thing about an Optitrack setup is that each joint has a rigid body of trackers associated with it, meaning that it can output real-time orientation data for each joint. Such a system will likely be essential for collecting enough data to make serious headway in orientation prediction. Collecting such data and training models on it would make a fantastic project of its own.

Finally, there is the aspect of more complex/general humanoid robot motion. This project just focused on a very small and simple subset of the wide range of motion available from a humanoid robot. In particular, unlocking the lower body opens up wide range of control questions that need to be answered. Walking/Locomotion has been a long-standing problem in humanoid control, maintaining balance during a stride is often the main issue, augmenting existing controllers for this using leg joint data and videos would be another very interesting path to explore. Not to mention the whole world of control possibilities that reinforcement learning and generative AI opens up. In conclusion, there is a lot more work that could be done on the subject.

7. Contributions

This project was done tangentially to another project I am working on for CS225A, Experimental Robotics. For that project we are using the Optitrack outputs directly in real time to control Toro's hands in order to play a simulated game of volleyball against an autonomous opponent. As part of that project, I collaborated with Rhea Malhotra (rheamal@stanford.edu) and Oskar Wendt (wendt@stanford.edu) to write the code that converts the Optitrack data into Toro's frame of reference. That code was used in this project to convert the Optitrack data in the Self-Collected dataset. We have not written our final report for that class yet, but I will email a copy of the PDF to my assigned TA mentor by June 11th once it is done.

Also as part of CS225A, skeleton code was provided for simulating Toro, but I built each of the three controllers specifically for this project. The repo for the Sai2 libraries used for control and simulation is linked below.

https://github.com/manips-sai-org

In order to interface with Detectron2 to gather joint key-point data and use VideoPose3D, I followed steps 1-5 of the INFERENCE.md document from this Github Repo. I also used code from this repo to generate the input and reconstruction portions of the video demos.

https://github.com/facebookresearch/VideoPose3D/tree/main

Everything else was done by me (Ethan Whitmer) for CS231N.

References

[1] J. Wang, S. Tan, X. Zhen, S. Xu, F. Zheng, Z. He, and L. Shao, “Deep 3d human pose estimation: A review,” *Computer Vision and Image Understanding*, vol. 210, p. 103225, 2021.

[2] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, “3d human pose estimation in video with temporal convolutions and semi-supervised training,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] J. Martinez, R. Hossain, J. Romero, and J. J. Little, “A simple yet effective baseline for 3d human pose estimation,” *CoRR*, vol. abs/1705.03098, 2017.

[4] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schäffer, “Overview of the torque-controlled humanoid robot toro,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 916–923, 2014.

[5] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” https://github.com/facebookresearch/detectron2, 2019.

[6] Z. Li, X. Wang, F. Wang, and P. Jiang, “On boosting single-frame 3d human pose estimation via monocular videos,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[7] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, “Semantic graph convolutional networks for 3d human pose regression,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] M. R. I. Hossain and J. J. Little, “Exploiting temporal information for 3d pose estimation,” *CoRR*, vol. abs/1711.08585, 2017.

[9] B. X. Nie, P. Wei, and S.-C. Zhu, “Monocular 3d human pose estimation by predicting depth on joints,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3467–3475, 2017.

[10] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi, and D. Pucci, “Teleoperation of humanoid robots: A survey,” *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1706–1727, 2023.

[11] J. Ramos and S. Kim, “Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation,” *Science Robotics*, vol. 4, no. 35, p. eaav4282, 2019.

[12] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, “Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9164–9170, 2020.

[13] Q. Gao, J. Li, Y. Zhu, S. Wang, J. Liufu, and J. Liu, “Hand gesture teleoperation for dexterous manipulators in space station by using monocular hand motion capture,” *Acta Astronautica*, vol. 204, pp. 630–639, 2023.

[14] J. B. Cole, D. B. Grimes, and R. P. N. Rao, “Learning full-body motions from monocular vision: dynamic imitation in a humanoid robot,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 240–246, 2007.

[15] A. Sripada, H. Asokan, A. Warrier, A. Kapoor, H. Gaur, R. Patel, and R. Sridhar, “Teleoperation of a humanoid robot with motion imitation and legged locomotion,” in *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 375–379, 2018.

[16] L. N. M., D. Dajles, and F. Siles, “Teleoperation of a humanoid robot using an optical motion capture system,” in *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, pp. 1–8, 2018.

[17] L. Sigal, A. Balan, and M. Black, “Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion,” *International Journal of Computer Vision*, vol. 87, pp. 4–27, 03 2010.

[18] G. Nagymate and R. Kiss, “Application of optitrack motion capture systems in human movement analysis a systematic literature review,” *Recent Innovations in Mechatronics*, vol. 5, 07 2018.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” *CoRR*, vol. abs/1912.01703, 2019.

[20] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, “Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[21] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, “Long short-term memory kalman filters: Recurrent neural estimators for pose regularization,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[22] L. Yang, Z. Qi, Z. Liu, S. Zhou, Y. Zhang, H. Liu, J. Wu, and L. Shi, “A light cnn based method for hand detection and orientation estimation,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2050–2055, 2018.