Dynamics-Regulated Kinematic Policy for Egocentric Pose Estimation **Appendix**

Zhengyi Luo[1] **Ryo Hachiuma** [2 *] **Ye Yuan**[1] **Kris Kitani**[1]
[1] Carnegie Mellon University [2] Keio University
https://zhengyiluo.github.io/projects/kin_poly/

Summary

A Qualitative Results (Supplemantry Video)

As motion is best seen in videos, we provide extensive qualitative evaluations in the supplementary video. Here we list a timestamp reference for evaluations conducted in the video:

- Qualitative results from real-world videos (00:12).
- Comparison with the state-of-the-art methods on the MoCap dataset's test split (01:27).

[*] Work done at Carnegie Mellon University.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), virtual.

- Comparison with the state-of-the-art methods on the real-world dataset (02:50).
- Failure cases for the dynamics-regulated kinematic policy (04:23).
- Qualitative results from the Universal Humanoid Controller (UHC) (4:39).
- Failure cases for the UHC (05:20).

B Dynamics-regulated Kinematic Policy

B.1 Evaluation Metrics Definition

Here we provide details about our proposed evaluation metrics:

- Root error: $\mathbf{E}_{\text{root}}$ compares the estimated and ground truth root rotation and orientation, measuring the difference in the respective 4×4 transformation matrix ($\boldsymbol{M}_t$): $\frac{1}{T}\sum_{t=1}^{T} \|I - (\boldsymbol{M}_t \widehat{\boldsymbol{M}}_t^{-1})\|_F$. This metric reflects both the position and orientation tracking quality.
- Mean per joint position error: $\mathbf{E}_{\text{mpjpe}}$ (mm) is the popular 3D human pose metric [19, 18, 20] and is defined as $\frac{1}{J}\|\boldsymbol{j}^{\text{pos}} - \widehat{\boldsymbol{j}}^{\text{pos}}\|_2$ for J number of joints. This value is root-relative and is computed after setting the root translation to zero.
- Acceleration error: E_{acc} (mm/frame^2) measures the difference between the ground truth and estimated joint position acceleration: $\frac{1}{J}\|\ddot{\boldsymbol{j}}^{\text{pos}} - \ddot{\widehat{\boldsymbol{j}}}^{\text{pos}}\|_2$.
- Foot sliding: FS (mm) is computed similarly as in [22], *i.e.* $\text{FS} = d\left(2 - 2^{h/H}\right)$ where d is the foot displacement and h is the foot height of two consecutive poses. We use a height threshold of $H = 33$ mm, the same as in [22].
- Penetration: PT (mm) is provided by the physics simulation. It measures the per-frame average penetration distance between our simulated humanoid and the scene (ground and objects). Notice that Mujoco uses a soft contact model where a larger penetration will result in a larger repulsion force, so a small amount of penetration is expected.
- Camera trajectory error: $\mathbf{E}_{\text{cam}}$ is defined the same as the root error, and measures the camera trajectory tracking instead of the root. To extract the camera trajectory from the estimated pose $\boldsymbol{q}_t$, we use the head pose of the humanoid and apply a delta transformation based on the camera mount's vertical and horizontal displacement from the head.
- Human-object interaction success rate: S_{inter} measures whether the desired human-object interaction is successful. If the humanoid falls down at any point during the sequence, the sequence is deemed unsuccessful. The success rate is measured automatically by querying the position, contact, and simulation states of the objects and humanoid. For each action:
  - Sitting down: successful if the humanoid's pelvis or the roots of both legs come in contact with the chair at any point in time.
  - Pushing a box: successful if the box is moved more than 10 cm during the sequence.
  - Stepping on a box: successful if the humanoid's root is raised at least 10 cm off the ground and either foot of the humanoid has come in contact with the box.
  - Avoiding an obstacle: successful if the humanoid has not come in contact with the obstacle and the ending position of the root/camera is less than 50 cm away from the desired position (to make sure the humanoid does not drift far away from the obstacle).

B.2 Fail-safe during evaluation

For methods that involve dynamics, the humanoid may fall down mid-episode and not complete the full sequence. In order to compare all methods fairly, we incorporate the "fail-safe" mechanism proposed in EgoPose [50] and use the estimated kinematic pose to restart the simulation at the timestep of failure. Concretely, we measure point of failure by thresholding the difference between the reference joint position $\tilde{\boldsymbol{j}}_{\boldsymbol{t}}^{\text{pos}}$ and the simulated joint position $\boldsymbol{j}_{\boldsymbol{t}}^{\text{pos}}$. To reset the simulation, we use the estimated kinematic pose $\tilde{\boldsymbol{q}}_{\boldsymbol{t}}$ to set the simulation state.

B.3 Implementation Details

The kinematic policy is implemented as a Gated Recurrent Unit (GRU) [6] based network with 1024 hidden units, followed by a three-layer MLP (1024, 512, 256) with ReLU activation. The value function for training the kinematic policy through reinforcement learning is a two-layer MLP (512, 256) with ReLU activation. We use a fixed diagonal covariance matrix and train for 1000 epoches using the Adam [**?**] optimizer. Hyperparameters for training can be found in Table. 3:

Table 1: Hyperparameters used for training the kinematic policy.

	γ	Batch Size	Value Learning Rate	Policy Learning Rate	PPO clip ϵ	Covariance Std
Value	0.95	10000	3×10^{-4}	5×10^{-4}	0.2	0.04
	w_{hp}	w_{hq}	$w_{\text{jr}}^{\text{gt}}$	$w_{\text{jv}}^{\text{gt}}$	$w_{\text{jr}}^{\text{dyna}}$	$w_{\text{jp}}^{\text{dyna}}$
Value	0.15	0.15	0.2	0.1	0.2	0.2

B.4 Additional Experiments about Stochasticity

Our kinematic policy is trained through physics simulation and samples a random sequence from the MoCap dataset for each episode. Here we study the stochasticity that rises from this process. We train our full pipeline with three different random seeds and report its results with error bars on both the MoCap test split and the real-world dataset. As can be seen in Table 2, our method has very small stochasticity and maintains high performance on *both* the MoCap test split and the real-world dataset, demonstrating the robustness of our dynamics-regulated kinematic policy. Across different random seeds, we can see that “stepping" is consistently the hardest action and “avoiding" is the easiest. Intuitively, “stepping" requires precise coordination between the kinematic policy and the UHC for lifting the feet and pushing up, while “avoiding" only requires basic locomotion skills.

B.5 Additional Analysis into low Per Joint Error

As discussed in the results section, we notice that our Mean Per Joint Position Error is relatively low compared to third-person pose estimation methods, although egocentric pose estimation is arguably a more ill-posed task. To provide an additional analysis of this observation, here we report the per-joint positional errors for the four joints with the smallest and largest errors, in ascending order:

As can be seen in the results, the toes and hands have much larger errors. This is expected as inferring hand and toe movements from only the egocentric view is challenging, and our network is able to extrapolate their position based on physical laws and prior knowledge of the scene context. Different from a third-person pose estimation setting, correctly estimating the torso area can be much easier from an egocentric point of view since torso movement is highly correlated with head motion. In summary, the low MPJPE reported on our MoCap dataset is the result of 1) only modeling a subset of possible human actions and human-object interactions, 2) the nature of the egocentric pose estimation task, 3) our network's incorporation of physical laws and scene context, which reduces the number of possible trajectories.

C Universal Humanoid Controller

C.1 Implementation Details

Proxy humanoid. The proxy humanoid we use for simulation is created automatically using the mesh, bone and kinematic tree defined in the popular SMPL [23] human model. Similar to the procedure in [52], given the SMPL body vertices $V = 6890$ and bones $B = 25$, we generate our humanoid based on the skinning weight matrix $\boldsymbol{W} \in \mathbb{R}^{V \times B}$ that defines the association between each vertex and bone. The geometry of each bone's mesh is defined by the convex hull of all vertices assigned to the bone. The mass of each bone is in turn defined by the volume of the mesh. To simplify the simulation process, we discard all body shape information from the AMASS [25] dataset, and use the mean body shape of the SMPL model. Since AMASS and our MoCap dataset are recorded by people with different height, we manually adjust the starting height of the MoCap pose to make sure each of the humanoid's feet are touching the ground at the starting point of the episode.

Table 2: Results of our dynamics-regulated kinematic policy on the test split of MoCap and real-world datasets using different random seeds. The "loco" motion in the MoCap dataset corresponds to the generic locomotion action, containing all sequences from the EgoPose [50] Dataset.

MoCap dataset										
$S_{inter}\uparrow$	$E_{root}\downarrow$	$E_{mpjpe}\downarrow$	$E_{acc}\downarrow$	FS $\downarrow$	PT $\downarrow$	Per class success rate $S_{inter}\uparrow$				
						Sit	Push	Avoid	Step	Loco
$96.87\% \pm 1.27\%$	0.21 ± 0.01	39.46 ± 0.52	6.27 ± 0.1	3.22 ± 0.11	0.69 ± 0.03	100%	$97.20\% \pm 3.96\%$	100%	$86.70\% \pm 4.71\%$	$97.4\% \pm 3.63\%$

Real-world dataset							
$S_{inter}\uparrow$	$E_{root}\downarrow$	FS $\downarrow$	PT $\downarrow$	Per class success rate $S_{inter}\uparrow$			
				Sit	Push	Avoid	Step
$92.17\% \pm 1.41\%$	0.49 ± 0.01	2.72 ± 0.03	1.03 ± 0.16	$94.7\% \pm 4.20\%$	$93.10\% \pm 1.84\%$	100.0%	$77.1\% \pm 2.37\%$

Table 3: Per-joint error on the MoCap dataset

Torso	Left_hip	Right_hip	Spine	Left_toe	Right_toe	Right_hand	Left_hand
7.099	8.064	8.380	15.167	65.060	66.765	74.599	77.669

Policy network architecture. Our Universal Humanoid Controller (UHC)'s workflow and architecture can be seen in Fig. 1. $\boldsymbol{\pi}_{\text{UHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \widehat{\boldsymbol{q}}_{t+1})$ is implemented as a multiplicative compositional policy (MCP) [**?**] with eight motion primitives, each being an MLP with two hidden layers (512, 256). The composer is another MLP with two hidden layers (300, 200) and outputs the multiplicative weights $\boldsymbol{w}_t^{1:n}$ for the n motion primitives. As studied in MCP [**?**], this hierarchical control policy increases the model's capacity to learn multiple skills simultaneously. The output $\boldsymbol{a}_t \in \mathbb{R}^{75}$ is a vector concatenation of the target angles of the PD controller mounted on the 23 no-root joints (each has 3 DoF), plus the residual force [51]: $\boldsymbol{\eta}_t \in \mathbb{R}^6$. Recall that each target pose $\widehat{\boldsymbol{q}}_t \in \mathbb{R}^{76}$, $\widehat{\boldsymbol{q}}_t \triangleq (\widehat{\boldsymbol{r}}_t^{\text{pos}}, \widehat{\boldsymbol{r}}_t^{\text{rot}}, \widehat{\boldsymbol{j}}_t^{\text{rot}})$ consists of the root position $\widehat{\boldsymbol{r}}_t^{\text{pos}} \in \mathbb{R}^3$, root orientation in quaternions $\widehat{\boldsymbol{r}}_t^{\text{rot}} \in \mathbb{R}^4$, and body joint angles in Euler angles $\widehat{\boldsymbol{j}}_t^{\text{rot}} \in \mathbb{R}^{69}$ of the human model. The use of quaternions and Euler angles follows the specification of Mujoco [41]. As described in the main paper, our UHC first transforms the simulation state to a feature vector using $\boldsymbol{T}_{\text{AC}}\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_{t+1}, D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t)\right)$ to output a 640 dimensional vector that is a concatenation of the following values:

$$\begin{aligned}(\boldsymbol{h}_t'^q, \boldsymbol{q}_t', \widehat{\boldsymbol{q}}_t', (\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t), \dot{\boldsymbol{q}}_t, (\boldsymbol{\psi}_t - \widehat{\boldsymbol{\psi}}_t), \widehat{\boldsymbol{j}}_t'^{\text{pos}}, (\boldsymbol{j}_t'^{\text{pos}} - \widehat{\boldsymbol{j}}_t'^{\text{pos}}), \boldsymbol{j}_t'^{\text{rot}}, (\boldsymbol{j}_t'^{\text{rot}} \ominus \widehat{\boldsymbol{j}}_t'^{\text{rot}})) \\ = \boldsymbol{T}_{\text{AC}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_{t+1}, D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t)).\end{aligned} \tag{1}$$

It consists of: root orientation $\boldsymbol{h}_t'^q \in \mathbb{R}^4$ in agent-centric coordinates; simulated pose $\boldsymbol{q}_t' \in \mathbb{R}^{74}$ ($\boldsymbol{q}_t' \triangleq (\boldsymbol{r}_t'^z, \boldsymbol{r}_t'^{rot}, \boldsymbol{j}_t^{\text{rot}})$, root height $\boldsymbol{r}_t'^z \in \mathbb{R}^1$, root orientation $\boldsymbol{r}_t'^{rot} \in \mathbb{R}^4$, and body pose $\boldsymbol{j}_t^{\text{rot}} \in \mathbb{R}^{69}$ expressed in Euler angles) in agent-centric coordinates; target pose $\widehat{\boldsymbol{q}}_t' \in \mathbb{R}^{74}$ in agent-centric coordinates; $(\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t) \in \mathbb{R}^{76}$ is the difference between the simulated and target pose (in world coordinate), calculated as $(\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t) \triangleq (\widehat{\boldsymbol{r}}_t^{\text{pos}} - \boldsymbol{r}_t^{\text{pos}}, \widehat{\boldsymbol{r}}_t^{\text{rot}} \ominus \boldsymbol{r}_t^{\text{rot}}, \widehat{\boldsymbol{j}}_t^{\text{rot}} - \boldsymbol{j}_t^{\text{rot}})$, where $\ominus$ calculates the rotation difference; $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{75}$ is the joint velocity computed by Mujoco; $(\boldsymbol{\psi}_t - \widehat{\boldsymbol{\psi}}_t) \in \mathbb{R}^1$ is the difference between the current heading (yaw) of the target and simulated root orientation; $\widehat{\boldsymbol{j}}_t'^{\text{pos}} \in \mathbb{R}^{72}$ and $(\boldsymbol{j}_t'^{\text{pos}} - \widehat{\boldsymbol{j}}_t'^{\text{pos}}) \in \mathbb{R}^{72}$ are joint position differences, calculated in the agent-centric space, respectively; $\widehat{\boldsymbol{j}}_t'^{\text{rot}} \in \mathbb{R}^{96}$ and $(\boldsymbol{j}_t'^{\text{rot}} \ominus \widehat{\boldsymbol{j}}_t'^{\text{rot}}) \in \mathbb{R}^{96}$ are joint rotation differences in quaternions (we first convert $\widehat{\boldsymbol{j}}_t'^{\text{rot}}$ from Euler angles to quaternions), calculated in the global and agent-centric space, respectively.

Reward function. The imitation reward function per timestep, similar to the reward defined in Yuan *et al*. [51] is as follows:

$$r_t = w_{\text{jr}} r_{\text{jr}} + w_{\text{jp}} r_{\text{jp}} + w_{\text{jv}} r_{\text{jv}} + w_{\text{res}} r_{\text{res}}, \tag{2}$$

where $w_{\text{jr}}, w_{\text{jp}}, w_{\text{jv}}, w_{\text{res}}$ are the weights of each reward. The joint rotation reward r_{jr} measures the difference between the simulated joint rotation $\boldsymbol{j}_t^{\text{rot}}$ and the target $\widehat{\boldsymbol{j}_t^{\text{rot}}}$ in quaternion for each joint on the humanoid. The joint position reward $\boldsymbol{r}_{\text{jp}}$ computes the distance between each joint's position $\boldsymbol{j}_t^{\text{pos}}$ and the target joint position $\widehat{\boldsymbol{j}_t^{\text{pos}}}$. The joint velocity reward $\boldsymbol{r}_{\text{jv}}$ penalizes the deviation of the

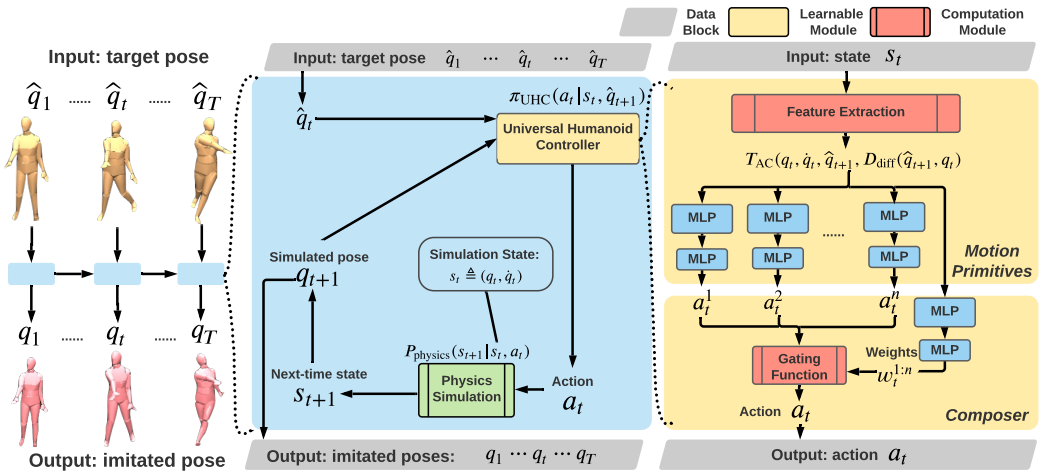

Figure 1: Overview of our Universal Dynamics Controller. Given a frame of target pose and current simulation state, our UHC π_{UHC} can dirve the the humanoid to match the target pose.

estimated joint angular velocity $\dot{\boldsymbol{j}}_t^{\text{rot}}$ from the target $\hat{\dot{\boldsymbol{j}}}_t^{\text{rot}}$. The target velocity is computed from the data via finite difference. All above rewards include every joint on the humanoid model (including the root joint), and are calculated in the world coordinate frame. Finally, the residual force reward $\boldsymbol{r}_{\text{res}}$ encourages the policy to rely less on the external force and penalize for a large $\boldsymbol{\eta}_t$:

$$
\begin{aligned}
r_{\text{jr}} &= \exp\left[-2.0\left(\|\boldsymbol{j}^{\text{rot}}{}_t \ominus \hat{\boldsymbol{j}}_t^{\text{rot}}\|^2\right)\right], \quad r_{\text{jp}} = \exp\left[-5\left(\left\|\boldsymbol{j}_t^{\text{pos}} - \hat{\boldsymbol{j}}_t^{\text{pos}}\right\|^2\right)\right], \\
r_{\text{jv}} &= \exp[-0.005\left\|\dot{\boldsymbol{j}}_t^{\text{rot}} \ominus \hat{\dot{\boldsymbol{j}}}_t^{\text{rot}}\right\|^2], \quad r_{\text{res}} = \exp\left[-\left(\|\boldsymbol{\eta}_t\|^2\right)\right].
\end{aligned} \tag{3}
$$

We train our Universal Humanoid Controller for 10000 epoches, which takes about 5 days. Additional hyperparameters for training the UHC can be found in Table 4:

Table 4: Hyperparameters used for training the Universal Humanoid Controller.

	γ	Batch Size	Value Learning Rate	Policy Learning Rate	PPO clip ϵ	Covariance Std
Value	0.95	50000	3×10^{-4}	5×10^{-5}	0.2	0.1
	w_{jr}	w_{jp}	w_{jv}	w_{res}	Sampling Temperature	
Value	0.3	0.55	0.1	0.05	2	

Training data clearning We use the AMASS [25] dataset for training our UHC. The original AMASS dataset contains 13944 high-quality motion sequences, and around 2600 of them contain human-object interactions such as sitting on a chair, walking on a treadmill, and walking on a bench. Since AMASS does not contain object information, we can not faithfully recreate and simulate the human-object interactions. Thus, we use a combination of heuristics and visual inspection to remove these sequences. For instance, we detect sitting sequences through finding combinations of the humanoid's root, leg, and torso angles that correspond to the sitting posture; we find walking-on-a-bench sequences through detecting a prolonged airborne period; for sequences that are difficult to detect automatically, we conduct manual visual inspection. After the data cleaning process, we obtain 11299 motion sequences that do not contain human-object interaction for our UHC to learn from.

C.2 Evaluation on AMASS

To evaluate our Universal Humanoid Controller's ability to learn to imitate diverse human motion, we run our controller on the full AMASS dataset (after removing sequences that include human-object interactions) that we

Table 5: Evaluation of motion imitation for our UHC using target motion from the AMASS dataset.

	AMASS dataset			
Method	$S_{\text{inter}} \uparrow$	$E_{\text{root}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	$E_{\text{acc}} \downarrow$
DeepMimic	24.0%	0.385	61.634	17.938
UHC w/o MCP	95.0%	0.134	25.254	5.383
UHC	**97.0%**	**0.133**	**24.454**	**4.460**

trained on. After data cleaning, the AMASS dataset contains 11299 high quality motion sequences, and contains challenging sequences such as kickboxing, dancing, backflipping, crawling, etc. We use a subset of metrics from egocentric pose estimation to evaluate the motion imitation results of UHC. Namely, we report S_{inter}, E_{root}, E_{mpjpe}, E_{acc}, where the human-object interation S_{inter} indicates whether the humanoid has become unstable and falls down during the imitation process. The baseline we compare against is the popular motion imitation method DeepMimic [30]. Since our framework uses a different physics simulation (Bullet [**?**] vs Mujoco [41]), we use an in-house implementation of DeepMimic. From the result of Table 5 we can see that our controller can imitate a large collection (10956/11299, 96.964%) of realistic human motion with high fidelity without falling. Our UHC also achieves very low joint position error on motion imitation and, upon visual inspection, our controller can imitate highly dynamic motion sequences such as dancing and kickboxing. Failure cases include some of the more challenging sequences such as breakdancing and cartwheeling and can be found in the supplementary video.

C.3 Evaluation on H36M

To evaluate our Universal Humanoid Controller's ability to generalize to *unseen motion sequences*, we use the popular Human 3.6M (H36M) dataset [**?**]. We first fit the SMPL body model to ground truth 3D keypoints similar to the process in [**?**] and obtain motion sequences in SMPL parameters. Notice that this fitting process is imperfect and the resulting motion sequence is of less quality than original MoCap sequences. These sequences are also never seen by our UHC during training. As observed in Moon *et al*. [**?**], the fitted SMPL poses have a mean per joint position error of around 10mm. We use the train split of H36M (150 unique motion sequences) as the target pose for our UHC to mimic. From the results shown in Table 6, we can see that our UHC can imitate the unseen motion in H36M with high accuracy and success rate, and outperforms the baseline method significantly. Upon visual inspection, we can see that the failure cases often result from losing balance while the humanoid is crouching down or starts running suddenly. Since our controller does not use any sequence level information, it has no way of knowing the upcoming speedup of the target motion and can result in instability. This indicates the importance of the kinematic policy adjusting its target pose based on the current simulation state to prevent the humanoid from falling down, and signifies that further investigation is needed to obtain a better controller. For visual inspection of motion imitation quality and failure cases, please refer to our supplementary video.

Table 6: Evaluation of motion imitation for our UHC using target motion from the H36M dataset.

H36M dataset				
Method	S_{inter} ↑	E_{root} ↓	E_{mpjpe} ↓	E_{acc} ↓
DeepMimic	0.0%	0.609	107.895	28.881
UHC w/o MCP	89.3%	0.200	**36.972**	4.723
UHC	**92.0%**	**0.194**	40.424	**3.672**

D Additional Dataset Details

D.1 MoCap dataset.

Our MoCap dataset (202 training sequences, 64 testing sequences, in total 148k frames) is captured in a MoCap studio with three different subjects. Each motion clip contains paired first-person footage of a person performing one of the five tasks: sitting down and (standing up from) a chair, avoiding an obstacle, stepping on a box, pushing a box, and generic locomotion (walking, running, crouching). Each action has around 50 sequences. The locomotion part of our dataset is merged from the egocentric dataset from EgoPose [49] since the two datasets are captured using a compatible system. MoCap markers are attached to the camera wearer and the objects to get the 3D full-body human pose and 6DoF object pose. To diversify the way actions are performed, we instruct the actors to vary their performance for each action (varying starting position and facing gait, speed *etc*.). We followed the Institutional Review Board's guidelines and obtained approval for the collection of this dataset. To study the diversity of our MoCap dataset, we plot the trajectory taken by the actors in Fig. 2. We can see that our trajectories are diverse and are spread out around a circle with varying distance from the objects. Table 7 shows the speed statistics for our MoCap dataset.

Table 7: Speed analysis of our MoCap dataset and real-world dataset. Unit: (meters/second)

MoCap dataset					Real-world dataset				
Action	Mean	Min	Max	Std	Action	Mean	Min	Max	Std
Sit	0.646	0.442	0.837	0.098	Sit	0.556	0.227	0.891	0.171
Push	0.576	0.320	0.823	0.119	Push	0.526	0.234	0.762	0.127
Avoid	0.851	0.567	1.084	0.139	Avoid	0.668	0.283	0.994	0.219
Step	0.844	0.576	1.029	0.118	Step	0.729	0.395	1.092	0.196

D.2 Real-world dataset.

Our real world dataset (183 testing sequences, in total 55k frames) is captured in everyday settings (living room and hallway) with an additional subject. It contains the same four types of interactions as our MoCap dataset and is captured from a head-mounted iPhone using a VR headset (demonstrated in Fig.3). Each action has around 40 sequences. As can be seen in the camera trajectory in Fig. 2, the real-world dataset is more heterogeneous than the MoCap dataset, and has more curves and banks overall. Speed analysis in Table 7 also shows that our real-world dataset has a larger standard deviation in terms of walking velocity and has a larger overall spread than the MoCap dataset. In all, our real-world dataset has more diverse trajectories and motion patterns than our MoCap dataset, and our dynamics-regulated kinematic policy can still estimate the sequences recorded in this dataset.

Figure 3: Our real-world dataset capturing equipment.

Notice that our framework is starting *position and orientation invariant*, since all of our input features are transformed into the agent-centric coordinate system using the transformation function $\boldsymbol{T}_{\text{AC}}$.

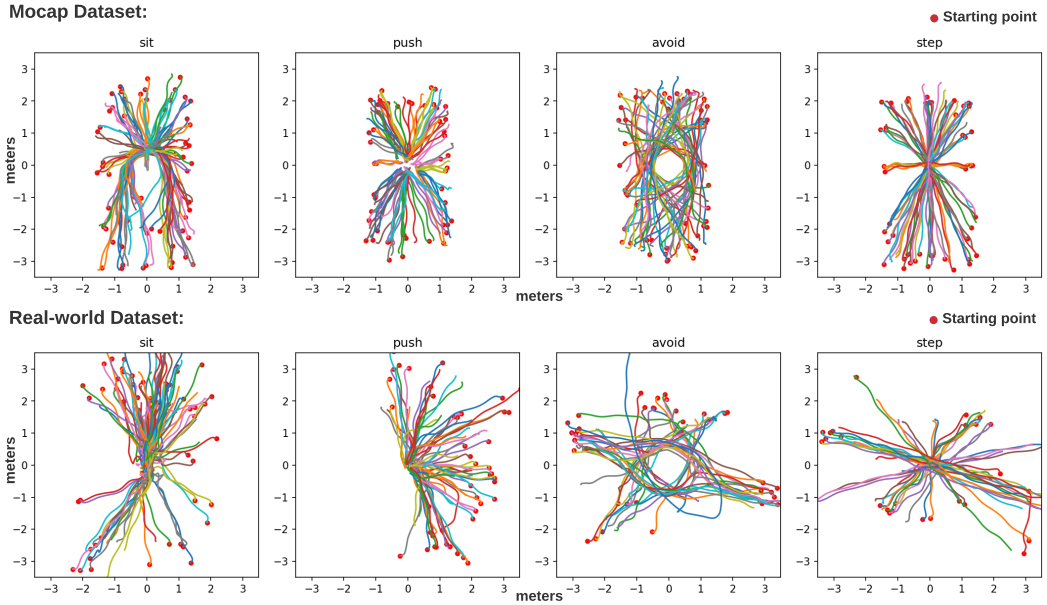

Figure 2: Trajectory analysis on our MoCap and real-world datasets. **Here we recenter each trajectory using the object position and plot the camera trajectory**, the objects are positioned differently across trajectories. The starting point is marked as a red dot.

D.3 Dataset Diversity

E Broader social impact.

Our overall framework can be used in extracting first-person camera wearer's physically-plausible motion and our humanoid controller can be a plug-and-play model for physics-based humanoid simulation, useful in the animation and gaming industry for creating physically realistic characters. There can be also negative impact from this work. Our humanoid controller can be used as a postprocessing tool to make computer generated human motion physically and visually realistic and be misused to create fake videos using Deepfake-like technology. Improved egocentric pose estimation capability can also mean additional privacy concerns for smart glasses and bodycam users, as the full-body pose can now be inferred from front-facing cameras only. As the realism of motion estimation and generation methods improves, we encourage future research in this direction to investigate more in detecting computer generated motion [**?**].

References

[1] Kevin Bergamin, Simon Clavet, Daniel Holden, and J. Forbes. Drecon. *ACM Transactions on Graphics (TOG)*, 38:1 – 11, 2019.

[2] Federica Bogo, A. Kanazawa, Christoph Lassner, P. Gehler, J. Romero, and Michael J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *The European Conference on Computer Vision (ECCV)*, 2016.

[3] M. Brubaker, L. Sigal, and David J. Fleet. Estimating contact dynamics. *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396, 2009.

[4] Yu-Wei Chao, Jimei Yang, Weifeng Chen, and Jia Deng. Learning to sit: Synthesizing human-chair interactions via hierarchical control. *ArXiv*, abs/1908.07423, 2019.

[5] N. Chentanez, M. Müller, M. Macklin, Viktor Makoviychuk, and S. Jeschke. Physics-based motion capture imitation with deep reinforcement learning. *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, 2018.

[6] Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *ArXiv*, abs/1406.1078, 2014.

[7] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456, Los Alamitos, CA, USA, Dec. 2013. IEEE Computer Society.

[8] Georgios V. Georgakis, Ren Li, Srikrishna Karanam, Terrence Chen, Jana Kosecka, and Ziyan Wu. Hierarchical kinematic human mesh recovery. *ArXiv*, abs/2003.04232, 2020.

[9] Riza Alp Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[10] I. Habibie, W. Xu, D. Mehta, G. Pons-Moll, and C. Theobalt. In the wild human pose estimation using explicit 2d features and intermediate 3d representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10897–10906, Jun. 2019.

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Mir Rayat Imtiaz Hossain and J. Little. Exploiting temporal information for 3d human pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2018.

[13] Soo hwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38:1 – 11, 2019.

[14] Apple Inc. Estimating camera pose with arkit. `https://developer.apple.com/documentation/arkit/arcamera`, 2021. Accessed: 2021-03-16.

[15] Apple Inc. Scanning and detecting 3d objects with arkit. `https://developer.apple.com/documentation/arkit/content_anchors/scanning_and_detecting_3d_objects`, 2021. Accessed: 2021-03-16.

[16] Mariko Isogawa, Ye Yuan, Matthew O'Toole, and Kris M Kitani. Optical non-line-of-sight physics-based 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7013–7022, 2020.

[17] Hao Jiang and Kristen Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3509, Jun. 2016.

[18] A. Kanazawa, Michael J. Black, D. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[19] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5252–5262, 2020.

[20] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2252–2261, 2019.

[21] Sijin Li and Antoni B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, 2014.

[22] Hung Yu Ling, Fabio Zinno, George H. Cheng, and M. V. D. Panne. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39:40:1 – 40:12, 2020.

[23] M. Loper, Naureen Mahmood, J. Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34:248:1–248:16, 2015.

[24] Zhengyi Luo, S. Alireza Golestaneh, and Kris M. Kitani. 3d human motion estimation via motion compression and refinement. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[25] Naureen Mahmood, N. Ghorbani, N. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019.

[26] J. Merel, S. Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, T. Erez, Greg Wayne, and N. Heess. Reusable neural skill embeddings for vision-guided whole body movement and object manipulation. *ArXiv*, abs/1911.06636, 2019.

[27] Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *IEEE Conference on International Conference on Computer Vision (ICCV)*, pages 10113–10142, Oct. 2019.

[28] Evonne Ng, Donglai Xiang, Hanbyul Joo, and Kristen Grauman. You2me: Inferring body pose in egocentric video via first and second person interactions. *CoRR*, abs/1904.09882, 2019.

[29] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[30] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, 7 2018.

[31] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37(6), November 2018.

[32] Davis Rempe, L. Guibas, Aaron Hertzmann, Bryan C. Russell, R. Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *SCA*, 2020.

[33] Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. Egocap: Egocentric marker-less motion capture with two fisheye cameras. *ACM Trans. Graph.*, 35(6), November 2016.

[34] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[35] John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[36] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture. *ACM Transactions on Graphics*, 40(4), aug 2021.

[37] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and C. Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Trans. Graph.*, 39:235:1–235:16, 2020.

[38] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[39] J. Tan, K. Liu, and G. Turk. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, 31(4):34–44, Jul. 2011.

[40] Bugra Tekin, Isinsu Katircioglu, M. Salzmann, Vincent Lepetit, and P. Fua. Structured prediction of 3d human pose with deep neural networks. *ArXiv*, abs/1605.05180, 2016.

[41] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, Oct. 2012.

[42] Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. xr-egopose: Egocentric 3d human pose from an hmd camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 7728–7738, Oct. 2019.

[43] M. Vondrak, L. Sigal, J. Hodgins, and O. C. Jenkins. Video-based 3d motion capture through biped control. *ACM Transactions on Graphics (TOG)*, 31:1 – 12, 2012.

[44] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, May 2017.

[45] Tingwu Wang, Yunrong Guo, Maria Shugrina, and S. Fidler. Unicon: Universal neural controller for physics-based character motion. *ArXiv*, abs/2011.15119, 2020.

[46] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39(4), 2020.

[47] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. Mo^2Cap^2 : Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019.

[48] Yuanlu Xu, S. Zhu, and Tony Tung. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7759–7769, 2019.

[49] Ye Yuan and Kris Kitani. 3d ego-pose estimation via imitation learning. In *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[50] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. In *IEEE International Conference on Computer Vision (ICCV)*, pages 10082–10092, Oct. 2019.

[51] Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. In *Advances in Neural Information Processing Systems*, 2020.

[52] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] Hongwen Zhang, Jie Cao, Guo Lu, Wanli Ouyang, and Z. Sun. Learning 3d human shape and pose from dense body parts. *ArXiv*, abs/1912.13344, 2019.