Supplemental Document of Paper "D&D: Learning Human Dynamics from Dynamic Camera"

Jiefeng Li[1], Siyuan Bian[1], Chao Xu[2], Gang Liu[2], Gang Yu[2], and Cewu Lu[1]

[1] Shanghai Jiao Tong University [2] Tencent
{ljf_likit,biansiyuan,lucewu}@sjtu.edu.cn
{dasxu,sylvainliu,skicyyu}@tencent.com

In the supplemental document, we provide:

A Derivations of the Dynamics Quantities

Inertia Matrix. Following Featherstone et al. [1], the inertia matrix M is derived as:

$$\mathrm{M} = \sum_{i}^{N_j} m_i J_{v_i}^{\mathsf{T}} J_{v_i} + J_{\omega_i}^{\mathsf{T}} I_{c_i} J_{\omega_i}, \tag{1}$$

where I_{c_i} denotes the inertia tensor of the i-th body joint and $J_{\omega_i} \in \mathbb{R}^{3\times(3N_j+6)}$ denotes the angular Jacobian matrix that relates angular velocity to pose velocity. J_{ω_i} can be computed recursively:

$$J_{\omega_i} = J_{\omega_j} + J_{\omega_{j\to i}}, \tag{2}$$

where $j = \mathcal{P}(i)$ is the parent index of the i-th joint and $J_{\omega_{j\to i}}$ represents the relative angular Jacobian matrix that can be computed from q. The linear Jacobian matrix of the i-th joint J_{v_i} can be computed based on J_{ω_i}:

$$J_{v_i} = J_{v_j} - [\Delta r_i]_\times J_{\omega_i}, \tag{3}$$

where $[\Delta r_i]_\times$ is the skew-symmetric matrix of the body-part vector Δr_i.

Angular Jacobian Matrix. The angular velocity of the i-th body joint ω_i can be computed recursively:

$$\omega_i = \omega_j + \omega_{j \to i}, \tag{4}$$

where $j = \mathcal{P}(i)$ is the parent index of the i-th joint and $\omega_{j \to i}$ denotes the relative angular velocity. Notice that the angular Jacobian matrix of the i-th body joint J_{ω_i} should satisfy:

$$\omega_i = J_{\omega_i} \dot{q}. \tag{5}$$

Therefore, we can build the recursive equation for J_{ω_i} by replacing ω_i with J_{ω_i} in Eqn. 4:

$$J_{\omega_i} = J_{\omega_j} + J_{\omega_{j \to i}}, \tag{6}$$

where

$$\omega_{j \to i} = J_{\omega_{j \to i}} \dot{q}. \tag{7}$$

To compute J_{ω_i}, we now need to compute $J_{\omega_{j \to i}}$ in each step. Denote $(\alpha_i, \beta_i, \gamma_i)$ as the relative rotation of the i-th joint in Euler angles. Then $J_{\omega_{j \to i}}$ is defined as:

$$J_{\omega_{j \to i}} = \left[\mathbf{0}^{3 \times (3j+3)} \;\; \mathcal{W}_{j \to i} \;\; \mathbf{0}^{3 \times (3N_j - 3j)} \right], \tag{8}$$

where

$$\mathcal{W}_{j \to i} = \begin{bmatrix} \cos\beta_i \cos\gamma_i & -\sin\gamma_i & 0 \\ \cos\beta_i \sin\gamma_i & \cos\gamma_i & 0 \\ -\sin\beta_i & 0 & 1 \end{bmatrix}. \tag{9}$$

For the root joint that has no parent, we have:

$$J_{\omega_0} = \mathbf{0}^{3 \times (3N_j + 6)}. \tag{10}$$

Linear Jacobian Matrix. The linear velocity of the i-th body joint v_i can be computed based on the angular velocity:

$$v_i = v_j + \omega_i \times \Delta r_i = v_j - [\Delta r_i]_\times \omega_i, \tag{11}$$

where Δr_i denotes the vector of the i-th body part. Notice that the linear Jacobian matrix of the i-th body joint J_{v_i} should satisfy:

$$v_i = J_{v_i} \dot{q}. \tag{12}$$

Therefore, we can compute J_{v_i} by replacing v_i with J_{v_i} and ω_i with J_{ω_i} in Eqn. 11:

$$J_{v_i} = J_{v_j} - [\Delta r_i]_\times J_{\omega_i}. \tag{13}$$

For the root joint that has no parent, we have:

$$J_{v_0} = \left[\mathbf{E}^{3 \times 3} \;\; \mathbf{0}^{3 \times (3N_j + 3)} \right], \tag{14}$$

where $\mathbf{E}^{3 \times 3}$ denotes the 3×3 identity matrix.

Inertia Tensor. Let I_i denote the inertia tensor of the i-th body joint under the rest pose that can be pre-computed. The inertia tensor of the i-th body joint under the pose q can be computed as:

$$I_{c_i} = R_i I_i R_i^{\mathsf{T}}, \tag{15}$$

where R_i denotes the rotation matrix of the i-th body joint.

B Differentiable Sampling

Non-differentiable Process. The Gumbel-Max trick [2, 6] provides a simple way to draw samples from a categorical distribution. The contact distribution of the j-th joint follows the Bernoulli distribution:

$$\Pr(b_j = 1) = p_j = 1 - \Pr(b_j = 0). \tag{16}$$

To draw sample $\widehat{b}_j$ with class probability p_j, we can conduct:

$$\widehat{b}_j = \arg\max_k [g_{jk} + \log \Pr(b_j = k)], \tag{17}$$

where $k \in \{0, 1\}$. Then the corresponding ground reaction torque can be computed as:

$$\widehat{\mathrm{h}}_{\mathrm{grf}} = \sum_j^{N_c} \mathbb{1}(\widehat{b}_j = 1) J_{v_j}^{\mathsf{T}} \lambda_j = \sum_j^{N_c} \widehat{b}_j J_{v_j}^{\mathsf{T}} \lambda_j. \tag{18}$$

Differentiable Process. However, Gumbal-Max is not differentiable. Therefore, we adopt Gumbel-Softmax [3] to conduct differentiable sampling from the predicted distribution. Gumbel-Softmax is a continuous and differentiable approximation to Gumbel-Max by replacing `argmax` with the softmax function:

$$\widehat{b}_j = \frac{\exp(\log \Pr(b_j = 1) + g_{j1})}{\sum_{k \in \{0,1\}} \exp(\log \Pr(b_j = k) + g_{jk})}. \tag{19}$$

Therefore, the corresponding ground reaction torque can be computed as:

$$\widehat{\mathrm{h}}_{\mathrm{grf}} = \sum_j^{N_c} \frac{p_j e^{g_{j1}}}{p_j e^{g_{j1}} + (1 - p_j) e^{g_{j0}}} J_{v_j}^{\mathsf{T}} \lambda_j, \tag{20}$$

Alternatively, we can compute the expectation of the ground reaction torque, which is also differentiable:

$$\bar{\mathrm{h}}_{\mathrm{grf}} = \sum_j^{N_c} p_j J_{v_j} \lambda_j. \tag{21}$$

However, using the expectation to generate motion cannot encourage well-calibrated probabilities [4], i.e., DyNet is not encouraged to generate high probabilities for

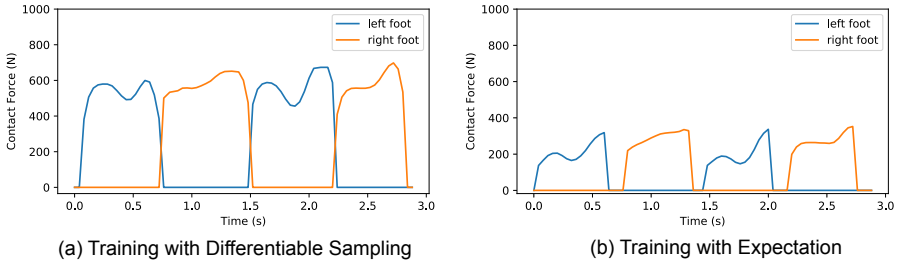

Fig. 1. Estimated contact forces of the walking sequence.

the correct contact states. Consequently, the contact forces might be incorrect since the model is trained without direct supervision. We plot the contact forces of the walking motion using the model trained with the torque expectation in Fig. 1(b). It shows that using the torque expectation makes the contact forces lie in an unreasonable range.

C Network Architecture

C.1 Kinematics Backbone

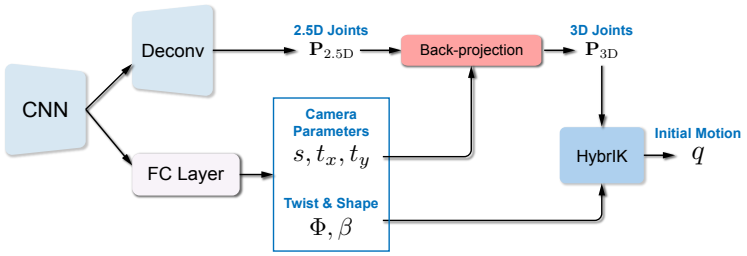

Fig. 2. The detailed architecture of the kinematics backbone.

The detailed network architecture of the kinematics backbone is illustrated in Fig. 2. We implement an extended version of HybrIK [5] as the backbone network. The original HybrIK model first predicts 2.5D joints of the body skeleton. Then the RootNet [8] is adopted to predict the distance of the root joint to the camera plane and obtain the 3D joints in the camera frame via back-projection. In our implementation, we design an integrated model that directly estimates the 3D joint positions. Specifically, we add a fully-connected layer to regress the camera parameters (s, t_x, t_y). Therefore, we can obtain the 3D joint positions as well as the initial motion within a single model.

C.2 DyNet

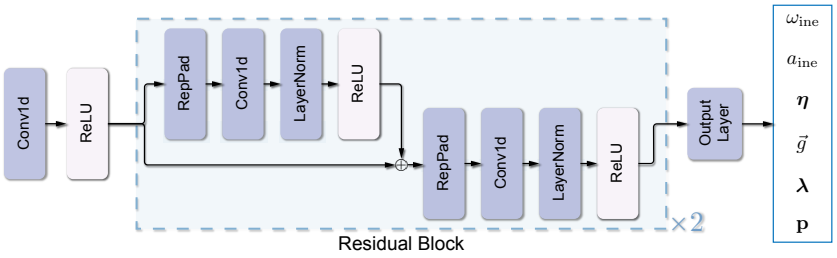

Fig. 3. The detailed architecture of DyNet.

The detailed network architecture of the DyNet is illustrated in Fig. 3. For all 1D convolutional layers, we use the kernel of size 3 and channel of size 256. The output layer consists of a bilinear GRU layer with the hidden dimension of 1024 and a fully-connected layer to predict the physical properties.

C.3 Attentive PD Controller

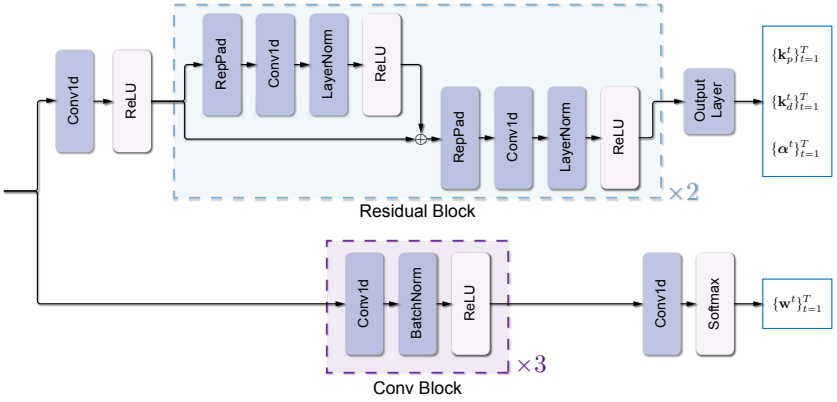

Fig. 4. The detailed architecture of attentive PD controller.

The detailed network architecture of the attentive PD controller is illustrated in Fig. 4. The kernel size and channel number are set to 3 and 256, respectively. To predict the attention weights that satisfy $\sum_{j=1}^{T} w^{tj} = 1$, we use a softmax layer for normalization.

D Details for Contact Annotations

To provide supervision signals for contact states, we generate contact annotations on the AMASS dataset [7]. Given a video sequence, we first retrieve the toe joints of the body in each frame and then fit the ground plane using the toe positions. Finally, the joints within 4cm of the ground are labeled as in contact. Following previous works [10, 9], we use $N_c = 4$ contact joints, i.e., toes and heels. In daily scenes, the human body might contact the environment with other joints, such as hips and hands. The current model will compensate these undefined contact forces with the residual force, and it is easy for our model to extend to more contact joints.

E Comparason with Baselines.

To further assess the effectiveness of D&D, we conduct experiments to compare D&D with two baselines: the acceleration network (AccNet) and the velocity network (VelNet). Instead of predicting physical properties like forces and contacts, AccNet and VelNet directly predict the pose acceleration and velocity, respectively. The acceleration and velocity are used to control the human motion. The architectures of AccNet and VelNet are similar to DyNet, with an output layer that predicts the pose acceleration and velocity. Other training settings are the same as the proposed D&D. As shown in Tab. 1, D&D obtains superior performance to AccNet and VelNet. It demonstrates that the improvement of D&D comes from the explicit modeling of the physical properties, not controlling human motion with acceleration or velocity.

Table 1. Comparason with baselines on 3DPW and Human3.6M datasets.

	3DPW			Human3.6M		
	MPJPE ↓	PA-MPJPE ↓	ACCEL ↓	MPJPE ↓	PA-MPJPE ↓	ACCEL ↓
AccNet	101.4	72.1	10.1	90.1	58.2	14.1
VelNet	79.3	47.0	8.4	74.1	48.8	8.5
D&D (Ours)	**73.7**	**42.7**	**7.0**	**52.5**	**35.5**	**6.1**

F Physical-based Results on the 3DPW Dataset.

We report two physical-based metrics, foot sliding (FS) and ground penetration (GP), to measure the physical plausibility on the 3DPW dataset. We remove the video sequences that contain stairs since it is inaccurate to measure the ground penetration in these cases. Quantitative results are provided in Tab. 2. It shows that D&D can predict physically plausible motion in daily scenes with dynamic camera movements.

Table 2. Physical-based Results on the 3DPW dataset.

Method	FS ↓	GP ↓
HybrIK [5]	37.3	29.9
Ours	**9.8**	**1.5**

G Global Trajectory Results on the Human3.6M Dataset.

We compare the predicted global trajectory using the simulated moving camera with the results using the static camera on the Human3.6M dataset. Quantitative results are reported in Tab. 3. When testing on the static camera, we directly use the predicted trajectory from HybrIK [5] in camera coordinates as the global trajectory. We then follow the standard evaluations for open-loop reconstruction (e.g., SLAM and GLAMR [11]) to remove the effect of the accumulative error. G-MPJPE and G-PVE are computed using a sliding window (10 seconds) and align the root translation with the GT at the start of each window. We can see that using static camera obtains better results than the closed-loop results of D&D with the moving camera. When we follow the open-loop protocol to eliminate the accumulative errors, D&D obtains better results than using the static camera.

Table 3. Results of the global trajectory on the Human3.6M dataset.

Method	G-MPJPE ↓	G-PVE ↓
Static Camera	674.7	681.5
Ours	785.1	793.3
Ours (open-loop)	525.3	533.9

H Computation Complexity.

The whole system is run online by using a sliding window with a length of 16 frames and a stride of 16 frames. The system takes 1349ms for each window (84.3ms for each frame).

I Pseudocode

The pseudocode of the proposed D&D is given in Alg. 1.

Algorithm 1 Pseudocode of D&D in a PyTorch-like style.

```
# inp_video: [B, T, 3, H, W]
# init_motion: [B, T, 75]
# betas: [B, 10]
init_motion, betas = kinematics_net(inp_video)

# Dynamics Networks
eta, a_ine, w_ine, g, lambda, p = DyNet(init_motion)
kp, kd, alpha, w = AttenPDController(init_motion)

# Initialize pose at time 0
final_q, final_dq = [q_0], [dq_0]
final_qtrans, final_dqtrans = [qtrans_0], [dqtrans_0]
rot_cam = 0

# Analytical Computation
for t in range(time_len - 1):
    current_q, current_dq = final_q[t], final_dq[t]
    current_qtrans, current_dqtrans = final_qtrans[t], final_dqtrans[t]

    # Compute Inertia Matrix and Jacobian
    M, Jv, Jw, m = get_jacobian(current_q, betas)
    # Compute physical torques
    target_q_state = torch.sum(w[t + 1] * initial_motion, dim=1)
    tau = kp[t] * (target_q_state - current_q) - kd[t] * current_dq + alpha[t]

    I = compute_inertia(m, Jv[t], a_ine[t], w_int[t])
    h_g = compute_gravity(m, Jv[t], g[t])
    h_grf = PCT(Jv, p[t], lambda[t])

    # Inertial Forward Dynamics
    ddq = M.inv().matmul(tau + h_g + h_grf + I)

    rot_cam = rot_cam + w_ine * delta_t

    # Trajectory Forward Dynamics
    ddqtrans = rot_cam.to_matrix().T.matmul(eta[t] + h_grf[:, 0:3] + h_g[:, 0:3]) / m[:, 0]

    # Constrained Update
    dq = current_dq + ddq * delta_t
    dqtrans = current_dqtrans + ddqtrans * delta_t

    dq, dqtrans = cvx_layer(dq, dqtrans, Jv[t], p[t], rot_cam)

    q = current_q + dq * delta_t
    qtrans = current_qtrans + dqtrans * delta_t

    final_q.append(q)
    final_dq.append(dq)
    final_qtrans.append(qtrans)
    final_dqtrans.append(dqtrans)

final_q = torch.cat(final_q, dim=1)
final_qtrans = torch.cat(final_qtrans, dim=1)
```

J Qualitative Results

Additional qualitative results are shown in Fig. 5, Fig. 6 (the Human3.6M dataset), Fig. 7, Fig. 8, Fig. 9 and Fig. 10 (the 3DPW dataset).

Fig. 5. Qualitative results on the Human3.6M dataset.

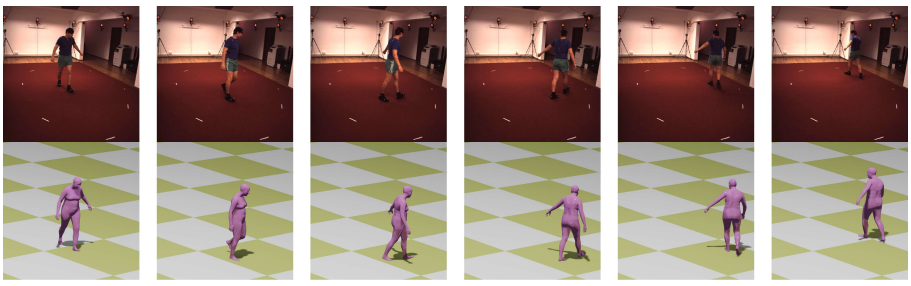

Fig. 6. Qualitative results on the Human3.6M dataset.

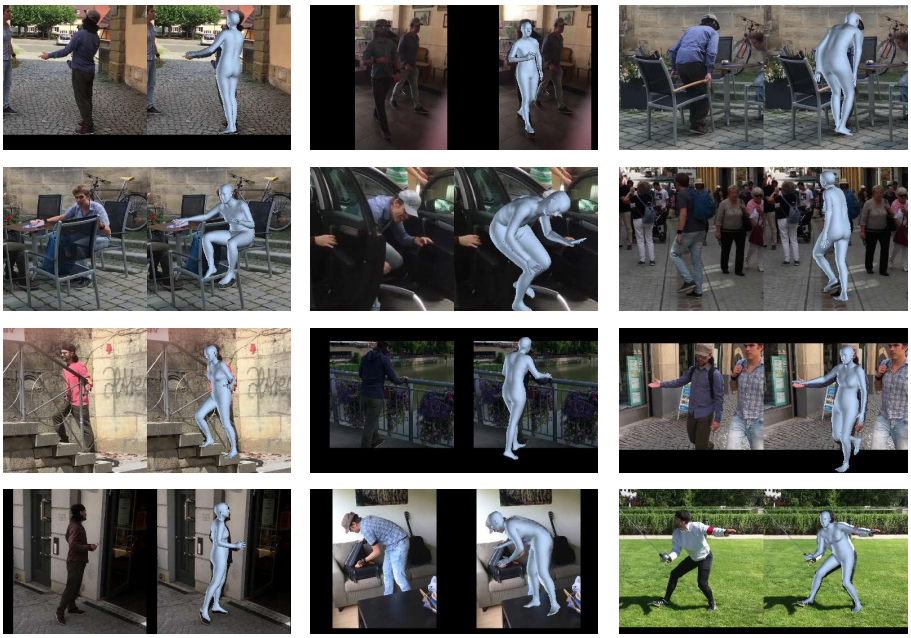

Fig. 7. Qualitative results on the 3DPW dataset.

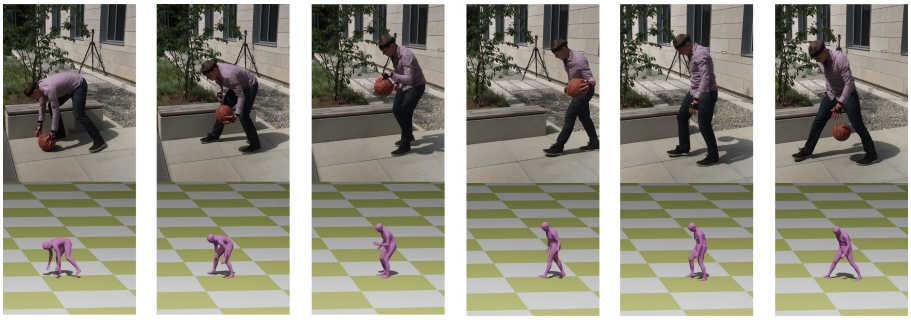

Fig. 8. Qualitative results on the 3DPW dataset.

Fig. 9. Qualitative results on the 3DPW dataset.

Fig. 10. Qualitative results on the 3DPW dataset.

References

1. Featherstone, R.: Rigid body dynamics algorithms. Springer (2014)
2. Gumbel, E.J.: Statistical theory of extreme values and some practical applications: a series of lectures (1954)
3. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: ICLR (2017)
4. Li, J., Chen, T., Shi, R., Lou, Y., Li, Y.L., Lu, C.: Localization with sampling-argmax. Advances in Neural Information Processing Systems **34**, 27236–27248 (2021)
5. Li, J., Xu, C., Chen, Z., Bian, S., Yang, L., Lu, C.: Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In: CVPR (2021)
6. Maddison, C.J., Tarlow, D., Minka, T.: A* sampling. NeurIPS (2014)
7. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: Amass: Archive of motion capture as surface shapes. In: ICCV (2019)
8. Moon, G., Chang, J.Y., Lee, K.M.: Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In: ICCV (2019)
9. Shimada, S., Golyanik, V., Xu, W., Pérez, P., Theobalt, C.: Neural monocular 3d human motion capture with physical awareness. TOG (2021)
10. Shimada, S., Golyanik, V., Xu, W., Theobalt, C.: Physcap: Physically plausible monocular 3d motion capture in real time. TOG (2020)
11. Yuan, Y., Iqbal, U., Molchanov, P., Kitani, K., Kautz, J.: Glamr: Global occlusion-aware human mesh recovery with dynamic cameras. In: CVPR (2022)