Supplementary material

A Motion-DVAE training

A.1 Implementation of the generative model

We recall the generative model of Motion-DVAE:

$$p_\theta(x_{1:T}, z_{1:T}|x_0) = p_\theta(z_1|x_0)p_\theta(x_1|z_1, x_0)\prod_{t=2}^{T} p_\theta(z_t|z_{1:t-1}, x_0)p_\theta(x_t|z_{1:t}, x_0), \tag{1}$$

We implement $p_\theta(x_t|z_{1:t}, x_0) = \mathcal{N}(x_t; \mu_{\theta_x}(z_{1:t}, x_0), Id)$ with:

- $h^x_0 = d_{ini}(x_0)$,
- $h^x_t = d_{h^x}(h^x_{t-1}, z_t)$,
- $\mu_{\theta_x}(h^x_t) = d_x(h^x_t)$.

For $p_\theta(z_t|z_{1:t-1}, x_0) = \mathcal{N}(z_t; \mu_{\theta_z}(z_{1:t-1}, x_0), \sigma_{\theta_z}(z_{1:t-1}, x_0))$, we take:

- $h^z_0 = d_{ini}(x_0)$
- $h^z_t = d_{h^z}(h^z_{t-1}, z_t)$
- $[\mu_{\theta_z}(h^z_t), \sigma_{\theta_z}(h^z_t)] = d_z(h^z_t)$

d_{h^x} and d_{h^z} are recurrent neural networks and are both implemented with the same LSTM. d_{ini}, d_x, and d_z are MLPs with 2 hidden layers using group normalization with 16 groups and ReLU activations. The latent dimension is 48, while the LSTM hidden state dimension is 1024.

A.2 Implementation of the approximate posterior

The inference model is:

$$q_\phi(z_{1:T}|x_{0:T}) = q_\phi(z_0|x_{0:T})\prod_{t} q_\phi(z_t|z_{1:t-1}, x_{t:T}, x_0). \tag{2}$$

$q_\phi(z_t|z_{1:t-1}, x_{t:T}, x_0) = \mathcal{N}(z_t; \mu_\phi(z_{1:t-1}, x_{t:T}, x_0), \sigma_\phi(z_{1:t-1}, x_{t:T}, x_0))$ is implemented by:

- $h^{glob}_t = [h^{lat}_{t-1}, h^{data}_t]$,
- $h^{lat}_0 = e_{ini}(x_0)$,
- $h^{lat}_t = e^{lat}_h(h^{lat}_{t-1}, z_t)$,
- $h^{data}_0 = 0$,
- $h^{data}_t = e^{data}_h(h^{data}_{t+1}, x_t)$,
- $[\mu_\phi(h^{glob}_t), \sigma_\phi(h^{glob}_t)] = e_{glob}(h^{glob}_t)$.

e^{lat}_h and e^{data}_h are LSTM neural networks, while e_{ini} and e_{glob} are MLPs. Implementations of LSTM and MLP are similar to the generative model.

A.3 Loss function

From Eq. (3) and Eq. (6), following [84], the ELBO of Motion-DVAE is:

$$\mathcal{L}(\theta, \phi; x_{0:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi}[\log p_\theta(x_t|z_{1:t}, x_0)] - \sum_{t=1}^{T} \mathbb{E}_{q_\phi}[D_{KL}(q_\phi(z_t|z_{1:t-1}, x_{t:T}, x_0)||p_\theta(z_t|z_{1:t-1}, x_0))]. \tag{3}$$

We define the loss function of Motion-DVAE:

$$\mathcal{L}^{train}(\theta, \phi; x_{0:T}) = \mathcal{L}_{rec}(\theta, \phi; x_{0:T}) + \mathcal{L}_{KL}(\theta, \phi; x_{0:T}) + \mathcal{L}_{reg}(\theta, \phi; x_{0:T}). \tag{4}$$

The first term can be developed as:

$$\begin{aligned}
\mathcal{L}_{rec}(\theta, \phi; x_{0:T}) &= -\sum_{t=1}^{T} \mathbb{E}_{q_\phi} \left[\log p_\theta(x_t|x_0, z_{1:t})\right] \\
&= -\sum_{t=1}^{T} \mathbb{E}_{q_\phi} \left[\log \mathcal{N}(x_t; \mu_{\theta_x}(z_{1:t}, x_0), Id)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{q_\phi} \left[\| \mu_{\theta_x}(z_{1:t}, x_0) - x_t \|_2^2\right].
\end{aligned}$$

This is a reconstruction term between the input and the output of Motion-DVAE.

The second term is a Kullback-Leibler divergence pushing the posterior distribution towards the prior:

$$\mathcal{L}_{KL}(\theta, \phi; x_{0:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi}[D_{KL}(q_\phi(z_t|z_{1:t-1}, x_{t:T}, x_0)||p_\theta(z_t|z_{1:t-1}, x_0))].$$

We also add a regularization term using the SMPL model to enforce the final human mesh to be as close as possible to the original mesh. It consists of a squared reconstruction error on meshes and joints:

$$\mathcal{L}_{reg}(\theta, \phi; x_{0:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi} \left[\| \mathcal{M}_\beta(\mu_{\theta_x}(z_{1:t}, x_0)) - \mathcal{M}_\beta(x_t) \|_2^2\right].$$

A.4 Learning settings

We train Motion-DVAE with sequences of 30 frames from the AMASS [85] dataset, previously downsampled to 30Hz. Then, learning sequences last 1 second. We choose this duration because we argue that even if human motion can last more than 1 second, direct dependencies between poses rarely exceeds 1 second. To ease learning, following [86], we align the first frame of each sequence in the canonical coordinate frame, meaning that translation r_0 and the first two components of root-orient Φ_0 are 0. This enables focusing on learning spatial-temporal dependencies independently from the starting point of the motion, which makes the motion prior more general.

We use batches of 64 sequences and train Motion-DVAE for 200 epochs. Similar to HuMoR [86], we use Adamax [87] with the same settings and learning rate decays. We also use KL-annealing [88] during the first 50 epochs. However, since our model does not use past predictions for current state prediction, we do not need to perform scheduled sampling.

B Unsupervised learned denoising posterior distribution

Let's recall the joint distribution in the context of motion denoising:

$$p_\theta(y_{0:T}, v_{0:T}, z_{1:T}|\beta, x_0) = p(v_0)p(y_0|x_0, \beta, v_0) \prod_{t=1}^{T} p_\theta(z_t|z_{1:t-1}, x_0)p(v_t)p_\theta(y_t|z_{1:t}, \beta, v_t, x_0). \tag{5}$$

We want to express the posterior distribution $p_\theta(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T})$:

$$\begin{aligned}
p_\theta(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T}) &= p(\beta|v_{0:T}, z_{1:T}, x_0, y_{0:T})p_\theta(v_{0:T}, z_{1:T}, x_0|y_{0:T}) \\
&= p(\beta|v_{0:T}, z_{1:T}, x_0, y_{0:T})p(v_{0:T}|z_{1:T}, x_0, y_{0:T})p_\theta(z_{1:T}|x_0, y_{0:T})p(x_0|y_{0:T}) \\
&= p(\beta|v_{0:T}, z_{1:T}, x_0, y_{0:T}) \prod_{t=1}^{T} \Big[p(v_t|z_{1:t}, y_t, x_0)p_\theta(z_t|z_{1:t-1}, y_{t:T}, x_0)\Big] p(v_0|y_0, x_0)p(x_0|y_{0:T}) \\
&\simeq p(\beta|y_{0:T}) \prod_{t=1}^{T} \Big[p(v_t|z_{1:t}, y_t, x_0)p_\theta(z_t|z_{1:t-1}, y_{t:T}, x_0)\Big] p(v_0|y_0, x_0)p(x_0|y_{0:T}).
\end{aligned}$$

As can be seen in the above calculations, we approximate the posterior of β. We choose to ignore the noise and the latent motion to simplify the model. In practice, β is computed as the average of the observed sequence of body shapes.

As a reminder, we define the following approximate posterior:

$$q(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T}) = q(\beta|y_{0:T}) \prod_{t=1}^{T} \Big[q_\gamma(v_t|z_{1:t}, y_t, x_0) q_\phi(z_t|z_{1:t-1}, y_{t:T}, x_0) \Big] q_\gamma(v_0|y_0, x_0) q_\omega(x_0|y_{0:T}). \quad (6)$$

C ELBO derivation and loss functions

For finetuning Motion-DVAE, we aim to minimize the Kullback-Leibler divergence:

$$\min_{\phi,\gamma,\omega} D_{KL}(q_{\phi,\gamma,\omega}(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T})||p_\theta(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T})). \quad (7)$$

We will do it by maximizing the ELBO:

$$\mathcal{L}(\phi, \gamma, \omega) = \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\log p_\theta(y_{0:T}, v_{0:T}, z_{1:T}, x_0, \beta) - \log q_{\phi,\gamma,\omega}(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T}) \right]. \quad (8)$$

C.1 ELBO decomposition

One can notice that the first term of the ELBO involves the joint distribution defined in Eq. (5). Taking the logarithm and expectation of this joint distribution, we obtain:

$$\mathbb{E}_{q_{\phi,\gamma,\omega}} \left[\log p(v_0) + \log p(y_0|x_0, \beta, v_0) \right] + \sum_{t=1}^{T} \mathbb{E}_{q_{\phi,\gamma,\omega}} \left[\log p_\theta(z_t|x_0, z_{1:t-1}) + \log p(v_t) + \log p_\theta(y_t|x_0, z_{1:t}, \beta, v_t) \right].$$

Similarly, the second term involves:

$$q_{\phi,\gamma,\omega}(x_0, \beta, z_{1:T}, v_{0:T}|y_{0:T}) = q(\beta|y_{0:T}) \prod_{t=1}^{T} \Big[q_\gamma(v_t|z_{1:t}, y_t, x_0) q_\phi(z_t|z_{1:t-1}, y_{t:T}, x_0) \Big] q_\gamma(v_0|y_0, x_0) q_\omega(x_0|y_{0:T}),$$

which becomes after taking the logarithm and expectation:

$$\begin{aligned} &\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\log q(\beta|y_{0:T}) + \log q_\gamma(v_0|y_0, x_0) + \log q_\omega(y_{0:T}) \right] \\ &+ \sum_{t=1}^{T} \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\log q_\gamma(v_t|z_{1:t}, y_t, x_0) + \log q_\phi(z_t|z_{1:t-1}, y_{t:T}, x_0) \right]. \end{aligned}$$

The decomposed ELBO is then:

$$\begin{aligned} \mathcal{L}(\phi, \gamma, \omega) &= \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\log p(y_0|x_0, \beta, v_0) + \sum_{t=1}^{T} \log p_\theta(y_t|z_{1:t}, \beta, v_t, x_0) \right] \\ &+ \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[-\log q_\omega(x_0|y_{0:T}) \right] \\ &+ \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\sum_{t=1}^{T} \log p_\theta(z_t|z_{1:t-1}, x_0) - \sum_{t=1}^{T} \log q_\phi(z_t|z_{1:t-1}, y_{t:T}, x_0) \right] \\ &+ \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[\sum_{t=0}^{T} \log p(v_t) - \sum_{t=1}^{T} \log q_\gamma(v_t|z_{1:t}, y_t, x_0) \right] \\ &+ \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})} \left[-\log q(\beta|y_{0:T}) \right], \end{aligned}$$

which can be rewritten as:

$$\begin{aligned}
\mathcal{L}(\phi,\gamma,\omega) &= \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log p(y_0|x_0,\beta,v_0) + \sum_{t=1}^{T}\log p_\theta(y_t|z_{1:t},\beta,v_t,x_0)\right] \\
&- \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log q_\omega(x_0|y_{0:T})\right] \\
&- \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,v_{0:T}|z_{1:T},y_{0:T})}\left[D_{KL}(\prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1},y_{t:T},x_0) \parallel \prod_{t=1}^{T} p_\theta(z_t|z_{1:t-1},x_0))\right] \\
&- \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T}|v_{0:T},y_{0:T})}\left[D_{KL}(\prod_{t=0}^{T} q_\gamma(v_t|z_{1:t},y_t,x_0) \parallel \prod_{t=0}^{T} p(v_t))\right] \\
&- \mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log q(\beta|y_{0:T})\right].
\end{aligned}$$

C.2 ELBO terms calculation

C.2.1 Data commitment term

The first term is a data commitment term. We have:

$$\begin{aligned}
&\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log p(y_0|x_0,\beta,v_0) + \sum_{t=1}^{T}\log p_\theta(y_t|x_0,z_{1:t},\beta,v_t)\right] \\
&= \mathbb{E}_{q_{\phi,\gamma,\omega}}\left[\log \mathcal{N}(y_0;\mathcal{M}_\beta(x_0),v_0) + \sum_{t=1}^{T}\log \mathcal{N}(y_t;\mathcal{M}_\beta(\mu_{\theta_x}(x_0,z_{1:t})),v_t))\right] \\
&= \frac{1}{2}\mathbb{E}_{q_{\phi,\gamma,\omega}}\left[\sum_{j,d}\frac{1}{v_{0,j,d}}(y_{0,j,d}-\mathcal{M}_\beta(x_0)_{j,d})^2 + \sum_{t,j,d}\frac{1}{v_{t,j,d}}(y_{t,j,d}-\mathcal{M}_\beta(\mu_{\theta_x}(x_0,z_{1:t}))_{j,d})^2\right]
\end{aligned}$$

As expected, those are Mean Squared Errors weighted by the inverse of the variance.

C.2.2 Initial state predictor term

The second term is linked to the initial state predictor:

$$\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log q_\omega(x_0|y_{0:T})\right] = \mathbb{E}_{q_\omega(x_0|y_{0:T})}\left[\log q_\omega(x_0|y_{0:T})\right].$$

This term corresponds to the negative entropy of $q_\omega(x_0|y_{0:T})$. We chose not to use it during learning because it would increase the initial state predicted variance, which is not a desired behavior.

C.2.3 Motion prior term

The third term of the ELBO uses Motion-DVAE to implement a motion prior:

$$\begin{aligned}
&\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,v_{0:T}|z_{1:T},y_{0:T})}\left[D_{KL}\left(\prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1},y_{t:T},x_0) \parallel \prod_{t=1}^{T} p_\theta(z_t|z_{1:t-1},x_0)\right)\right] \\
&= \sum_{t=1}^{T}\mathbb{E}_{q_\omega q_\phi}\Big[D_{KL}\big(q_\phi(z_t|z_{1:t-1},y_{t:T},x_0)||p_\theta(z_t|x_0,z_{1:t-1})\big)\Big] \\
&= \sum_{t=1}^{T}\mathbb{E}_{q_\omega q_\phi}\Big[D_{KL}\big(\mathcal{N}(z_t;\mu_\phi(z_{1:t-1},x_{t:T},x_0),\sigma_\phi(z_{1:t-1},x_{t:T},x_0)) \parallel \mathcal{N}(z_t;\mu_{\theta_z}(z_{1:t-1},x_0),\sigma_{\theta_z}(z_{1:t-1},x_0))\big)\Big] \\
&= -\frac{1}{2}\sum_{t=1}^{T}\mathbb{E}_{q_\omega q_\phi}\left[\log\frac{\sigma_{\mu_{\theta_z}}}{\sigma_\phi} - 1 + \frac{\sigma_\phi}{\sigma_{\mu_{\theta_z}}} + \frac{(\mu_{\mu_{\theta_z}}-\mu_\phi)^2}{\sigma_{\mu_{\theta_z}}}\right].
\end{aligned}$$

C.2.4 Noise prior term

The fourth term of the ELBO is a noise prior term:

$$
\begin{aligned}
&\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T}|v_{0:T},y_{0:T})}\left[D_{KL}(\prod_{t=0}^{T} q_\gamma(v_t|z_{1:t},y_t,x_0) \parallel \prod_{t=0}^{T} p(v_t)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{q_\omega q_\phi}\left[D_{KL}(q_\gamma(v_t|z_{1:t},y_t,x_0) \parallel p(v_t)\right] \\
&= \sum_{t,j,d} \mathbb{E}_{q_\omega q_\phi}\left[D_{KL}\left(\mathcal{IG}\left(v_{t,j,d},\alpha_\gamma(z_{1:t},y_{t,j,d},x_0),\beta_\gamma(z_{1:t},y_{t,j,d},x_0)\right) \parallel \mathcal{IG}\left(v_{t,j,d},\frac{\lambda}{2},\frac{\lambda}{2}\right)\right)\right] \\
&= \sum_{t,j,d} \mathbb{E}_{q_\omega q_\phi}\left[\left(\alpha_\gamma - \frac{\lambda}{2}\right)\psi\left(\alpha_\gamma\right) - \log\Gamma\left(\alpha_\gamma\right) + \log\Gamma\left(\frac{\lambda}{2}\right) + \frac{\lambda}{2}\left(\log\beta_\gamma - \log\frac{\lambda}{2}\right) + \alpha_\gamma\frac{\frac{\lambda}{2}-\beta_\gamma}{\beta_\gamma}\right],
\end{aligned}
$$

where Γ and ψ are the Gamma and Digamma functions.

C.2.5 Body shape term

The last term is about body shape:

$$
\mathbb{E}_{q_{\phi,\gamma,\omega}(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\left[\log q(\beta|y_{0:T})\right] = \mathbb{E}\left[\delta(\beta - \bar{\beta}_{SPIN}(y_{0:T}))\right]
$$

Since that term does not depend on any learned parameter, we ignore it during training.

D Final predictions from observations

D.1 Initial state

We start by predicting the initial state:

$$
\hat{x}_0^{3d} = \mathbb{E}_{q_\omega(x_0|y_{0:T})}[\mathcal{M}_\beta(x_0)] \simeq \mathcal{M}_\beta(\mu_\omega(y_{0:T})), \tag{9}
$$

where μ_ω is the mean vector of $q_\omega(x_0|y_{0:T})$.

D.2 Motion prediction

Then we need to compute $\hat{x}_{1:T} = \mathbb{E}_{p_\theta(x_{1:T}|y_{0:T})}[x_{1:T}]$ where

$$
\begin{aligned}
p(x_{1:T}|y_{0:T}) &= \int p_\theta(x_{1:T},x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})\; d_{x_0}d_\beta d_{z_{1:t}}d_{v_{0:T}} \\
&= \int p_\theta(x_{1:T}|x_0,\beta,z_{1:T},v_{0:T},y_{0:T})p_\theta(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})\; d_{x_0}d_\beta d_{z_{1:t}}d_{v_{0:T}} \\
&= \mathbb{E}_{p_\theta(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})}\Big[p_\theta(x_{1:T}|x_0,\beta,z_{1:T},v_{0:T},y_{0:T})\Big].
\end{aligned}
$$

Approximating the posterior $p_\theta(x_0,\beta,z_{1:T},v_{0:T}|y_{0:T})$ with $q_{\phi,\gamma,\omega}$ we obtain:

$$
\hat{x}_{1:T} = \mathbb{E}_{q_{\phi,\gamma,\omega}}\Big[\mathbb{E}_{p_\theta(x_{1:T}|x_0,\beta,z_{1:T},v_{0:T},y_{0:T})}[x_{1:T}]\Big]. \tag{10}
$$

We need to calculate $p_\theta(x_{1:T}|x_0,\beta,z_{1:T},v_{0:T},y_{0:T})$:

$$
\begin{aligned}
\log p_\theta(x_{1:T}|x_0,\beta,z_{1:T},v_{0:T},y_{0:T}) &\overset{c}{=} \log p_\theta(x_{1:T},x_0,\beta,z_{1:T},v_{0:T},y_{0:T}) \\
&\overset{c}{=} \sum_t \log p(y_t|x_t,v_t)p_\theta(x_t|z_{1:T},x_0) \\
&= \sum_t \log\mathcal{N}(y_t;x_t,\mathrm{diag}(v_t))\mathcal{N}(x_t;\mu_{\theta_x},\sigma_{\theta_x}),
\end{aligned}
$$

where $\stackrel{c}{=}$ denotes equality up to an additive constant that does not depend on $x_{1:T}$. We can further develop the last line and identify:

$$\log p_\theta(x_{1:T}|x_0, \beta, z_{1:T}, v_{0:T}, y_{0:T}) = \sum_{t,j,d} \log \mathcal{N}\left(x_{t,j,d}; \frac{v_{t,j,d}(\mu_{\theta_x})_{j,d} + (\sigma_{\theta_x})_{j,d} y_{t,j,d}}{v_{t,j,d} + (\sigma_{\theta_x})_{j,d}}, \frac{v_{t,j,d}(\sigma_{\theta_x})_{j,d}}{v_{t,j,d} + (\sigma_{\theta_x})_{j,d}}\right).$$

This finally leads to:

$$\hat{x}^{3d}_{t,j,d} = \mathbb{E}_{q_{\phi,\gamma,\omega}}\left[\frac{v_{t,j,d}\mathcal{M}_\beta(\mu_{\theta_x}(x_0, z_{1:t-1})_{j,d}) + y^{3d}_{t,j,d}}{v_{t,j,d} + 1}\right]. \tag{11}$$

E Unsupervised denoising learning

E.1 Neural networks implementation

During the unsupervised denoising training, we introduce 2 neural networks: an initial state predictor and a noise predictor. Those 2 models were not necessary for training Motion-DVAE on clean motion capture data since x_0 was known and there was no noise in the observations.

As described in the main paper, the initial state predictor implements $q_\omega(x_0|y_{0:T}) = \mathcal{N}(x_0; \mu_\omega(y_{0:T}), \sigma_\omega(y_{0:T}))$. It is implemented by:

- $h_0 = 0$
- $h_t = r_h(h_{t+1}, y_t)$
- $\hat{x_0} = e_{ini}(h_T)$

r_h is an anticausal LSTM neural network. We want it to take the observations $y_{0:T}$ backward in time because x_0 should depend more on y_0 than on y_T. e_{ini} is an MLP.

For the noise predictor, we implement $q_\gamma(v_t|z_{1:t}, y_t, x_0) = \prod_{j,d} \mathcal{IG}\left(v_{t,j,d}; \alpha_\gamma(z_{1:t}, y_t, x_0), \beta_\gamma(z_{1:t}, y_t, x_0)\right)$ as follows:

- $h^z_0 = d_{ini}(x_0)$
- $h^z_t = d_{h^z}(h^z_{t-1}, z_t)$
- $[\alpha_\gamma, \beta_\gamma] = e_b(h_t, y_t)$

d_{ini} and d_{h^z} are the MLP and the LSTM neural networks previously defined for Motion-DVAE. e_b is an MLP only used by the noise predictor.

E.2 Learning settings

Unsupervised denoising training for regression mode is performed on training data for 500 epochs, with early stopping when the validation loss does not improve for 10 epochs. Note that for noisy AMASS [85] data, the training converges in about 50 epochs only, probably due to a large amount of training data. In optimization mode, we fix the number of iterations (200 iterations on i3DB [89], and 50 for AMASS).

During unsupervised denoising learning, the decoder and prior networks are fixed, preserving the motion prior. We optimize the weights of the encoder, initial state predictor, and noise predictor networks. As for training the original Motion-DVAE, we use KL-annealing [88].

F SPIN-t

SPIN-t is a procedure built on SPIN's [90] frame-wise local pose estimations (in frame coordinates) to obtain global motion (in world coordinates). SPIN provides SMPL pose and shape parameters Θ and β, as well as orientation ϕ relative to the camera. To obtain motion in global coordinates, we need the global translation r. For smooth and realistic 3D trajectories, we decide to optimize both global translation and rotation. We define v as the output of the SMPL model forward passes with optimization variables r and ϕ and SPIN [90] predictions β and Θ. During the optimization, v it only depends on the global translation and rotation. We use L-BFGS [91] to solve the following optimization problem:

$$\min_{r,\phi} \quad \lambda_{data}\, \xi_{data}(r, \phi) + \lambda_{smooth}\, \xi_{smooth}(r, \phi), \tag{12}$$

where $\xi_{data}(r, \phi) = \sum_t \sum_j \sigma_{t,j} \rho(\Pi(v_{t,j}) - y_{t,j})$, with t the temporal index, j the observed joint, $y_{t,j}$ and $\sigma_{t,j}$ the Openpose [92] 2D detection and its associated confidence, ρ the robust Geman McClure function [93], and Π the pinhole projection; and $\xi_{smooth} = ||v_{1:T} - v_{0:T-1}||_2^2$. ξ_{data} ensures that the projection of the predicted 3D joints corresponds to the 2D detections, while ξ_{smooth} smooths the motion over time. λ_{data} and λ_{smooth} are two hyperparameters.

G Ablation of SOTA optimization processes

The optimization process of HuMoR can be divided into 2 parts. A first part called VPoser-t minimizes the projection error of 3D predictions relative to 2D keypoints [92], with a pose prior [94] and regularization terms. The second part of optimization uses a CVAE motion prior to predict more plausible motions. We first decrease the number of iterations of the last part, which is the most time-consuming and the less important for the final results. We then decreased the number of iterations for VPoser-t, leading to significant degradation in performance.

Pose-NDF uses 5 steps of 50 iterations. At each iteration, the observation reconstruction weight becomes less and less important compared to the pose prior and smoothing term. Then, changing the number of iterations per step would change the balance of the loss functions, and we thus reduce the number of steps instead.

H Additional experiments

H.1 Plausibility versus speed

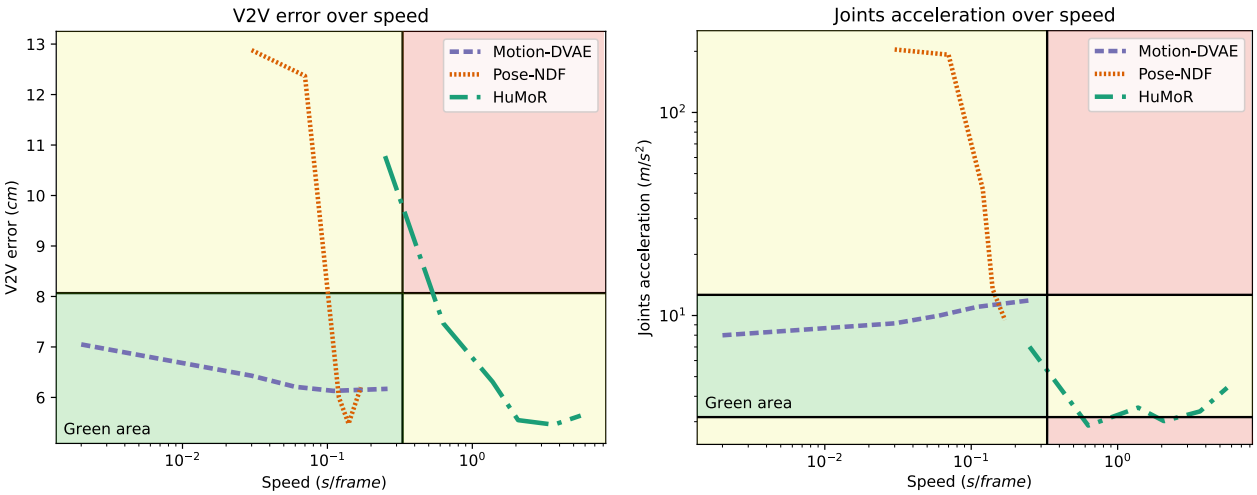

Figure 1: Evolution of the performance versus speed. Vertex-to-vertex (V2V) error is an accuracy metric, while joints acceleration measures plausibility.

This experiment is similar to the "Accuracy versus Speed" experiment in the main paper. We consider that joint acceleration should be between half and double of the ground-truth joint acceleration. Results are shown in Fig. 1.

HuMoR obtains satisfying plausibility performance, but it is very slow. Pose-NDF produces motions that are not smooth enough. Considering plausibility and speed Motion-DVAE is clearly the best choice. We observe that the more optimization iterations are performed with the proposed method, the better accuracy, but also the less smooth predictions. This result is not surprising since optimizing the model for specific samples leads to a loss of generalization.

H.2 Experiments on i3DB

We provide supplementary experiments on i3DB [89]. The i3DB dataset [89] contains in-the-wild videos with medium to heavy occlusions due to person-environment interaction. It provides global 3D annotations on joint coordinates that we use for computing evaluation metrics. We initialize our method using an off-the-shelf 2D pose estimation [92] and SPIN [90], a method providing frame-wise SMPL pose estimates.

Figure 2: Qualitative results for motion estimation from videos.

H.2.1 Evaluation on RGB videos

We evaluate the proposed method on i3DB [89]. Following [86], we only use six scenes with accurate annotations (Scenes 05, 07, 10, 11, 13, and 14). Due to this dataset's low number of videos, we evaluate our approach in a cross-validation setting, with each fold: 4 videos for training, 1 for validation, and 1 for testing. As in the previous experiment, the method is evaluated in the two regression and optimization modes.

As for the motion denoising experiment, we use very high values for noise parameter λ for learning the denoising process on the noisy training set. However, we use a robust learned noise model in the optimization mode. This gives a Student-t prior distribution for the noise on the 3D observations. Fitting a robust noise model on noisy observation sequences allows us to adapt to the observed noise in an unsupervised manner, which is expected to be beneficial in terms of prediction accuracy. Indeed, as explained in the section 4.3 of the main paper, this unsupervised adaptation allows us to find a compromise between the noisy observations and the Motion-DVAE output, depending on the estimated noise variance. A lower variance will give more importance to the observations and vice versa.

We use the open-source implementations of SPIN [90], VIBE [95], and HuMoR [86] for comparing with SOTA algorithms. We also include MVAE [96] in the comparison, but taking the results given in the paper of [86]. We do not have the running time for this method, but we can assume that it is similar to HuMoR since MVAE can be thought of as ablation of the HuMoR CVAE using the same optimization procedure. The running time of 2D keypoints detection by off-the-shelf models is not considered in speed calculation since it is used by all methods. However, we add SPIN execution time to our method for a fair comparison with SPIN [90] and other methods. Results are presented in Tab. 1.

Table 1: Results on i3DB: We compare the proposed method (bottom part) and SOTA (top part) performance in terms of global and local Mean Per Joint Positional Error (MPJPE) (cm) and execution time (sec/frame) on visible (Vis) and occluded (Occ) joints. We achieve an accuracy competitive with the SOTA methods while showing reasonably high speed.

	Speed ↓	G-MPJPE ↓			MPJPE ↓		
	(sec/frame)	All	Vis	Occ	All	Vis	Occ
SPIN [90]	<u>0.05</u>	-	-	-	14.95	11.89	23.90
VIBE [95]	**0.02**	116.46	90.05	192.55	15.08	12.06	23.78
MVAE [96]	-	40.91	37.54	50.63	19.17	16.00	28.32
VPoser-t [86]	1.05	<u>31.88</u>	<u>28.84</u>	<u>40.63</u>	16.36	12.81	26.57
HuMoR [86]	11.37	**28.68**	**26.01**	**36.37**	15.29	12.57	23.10
SPIN-t (ours)	0.25	42.45	36.54	59.71	14.28	11.66	21.94
Regression (ours)	0.26	41.41	36.55	55.45	**13.82**	<u>11.54</u>	**20.40**
Optimization (ours)	0.56	41.09	36.01	55.78	<u>14.08</u>	**11.48**	<u>21.58</u>

We can notice that the proposed SPIN-t improves the SPIN predictions regarding the MPJPE by adjusting the global rotation. SPIN-t outperforms all SOTA methods in terms of MPJPE. However, VPoser-t is more accurate globally (i.e., in terms of G-MPJPE) by about 8 cm.

Motion-DVAE outperforms all SOTA models by more than 1cm in terms of MPJPE. Regarding the global error, Motion-DVAE is similar to MVAE and significantly outperforms VIBE, which gives inaccurate results for occluded joints. HuMoR obtains better global predictions but is prohibitively slow for real-world applications.

Table 2: Ablation study: MPJPE and G-MPJPE are in cm.

		G-MPJPE		MPJPE	
		Vis	Occ	Vis	Occ
Full	HuMoR [86]	26.01	36.37	12.57	23.10
	Regression	36.55	55.45	11.54	20.40
	Optimization	36.01	55.78	11.48	21.58
NN output	Regression	39.54	56.64	13.69	22.32
	Optimization	36.42	55.33	11.80	21.95
No prior	HuMoR [86]	25.64	36.12	12.13	22.58
	Regression	37.51	58.77	11.64	20.90
	Optimization	36.75	58.68	11.56	21.74
No noisy train	Regression	68.19	81.72	32.30	47.56
	Optimization	39.97	57.64	12.26	22.49
Gaussian noise	Optimization	35.84	56.93	11.52	21.76

Qualitative results are shown in Fig. 2. When there are no occlusions, all methods perform great visually. However, with occlusions, there are clear differences between predictions. VIBE fails to estimate the global translation, and the arms pose is not as accurate as other methods. Motion-DVAE and HuMoR give good local predictions, with more realistic floor contacts for the latter. HuMoR seems to underestimate the distance between the person and the camera for global predictions as the prediction looks closer than the person on the image, while Motion-DVAE predicted translation is a bit too far relative to the camera. Qualitative results for videos are made available in the supplementary material.

H.2.2 Ablation study

We also perform an ablation study to evaluate the impact of the different components of the denoising procedure learned in an unsupervised fashion. We chose i3DB [89] for the ablation study because this dataset has many occlusions, enabling us to evaluate the model components for dealing with occlusions. Results are shown in Tab. 2. Most ablations are tested in regression (Reg) and optimization modes (Opt), except for the last experiment, which only concerns the optimization mode. Results are compared with the performance without any ablation (Full).

The first ablation experiment, "NN output", predicts the initial state, the latent motion, and the noise by taking directly the expectations outputted by the associated inference model networks successively instead of sampling the approximate posterior distributions as explained in the section 4.3 of the main paper. This approximation is made in most works, but neural network decoders are not linear functions. Thus taking the expectation of the latent representation and decoding it does not give the expectation of the network output. Results obtained without the proposed inference scheme are less accurate by about 1.5 cm in the regression mode and 0.5 cm in the optimization mode.

Next, we remove $\mathcal{L}_{KL}^{DVAE}(\phi, \omega)$ from the equation (13) of the main paper for both training and optimization ("No prior"). We compare our results with HuMoR [86] by setting the motion prior weight to 0 in optimization mode. Surprisingly, when removing the motion prior, HuMoR obtains better results for occluded and visible joints. On the contrary, removing the motion prior penalizes our method in regression and optimization modes. In the optimization mode, the results get better on every metric using the motion prior. This shows that the proposed motion prior is efficient for unsupervised optimization.

In "No noisy train", we skip the unsupervised training step on noisy data: we perform direct inference and optimize from Motion-DVAE learned on AMASS. As expected, the results deteriorate significantly in the regression mode because we use a model trained on clean data for processing noisy observations. In optimization mode, we do not reach the accuracy of the regression mode with no ablation. This demonstrates the efficiency of the introduced unsupervised learning procedure for denoising new unknown motions.

Finally, in "Gaussian noise" we test the proposed approach in optimization mode using a Gaussian noise prior distribution instead of Student-t. The results are improved on visible joints but deteriorate on occlusions. However, the results in the regression mode are still better than the results in the optimization mode, which shows the excellent generalization of the unsupervised learning approach.

Supplementary References

[84] Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. *Foundations and Trends® in Machine Learning*, 15, 2021.

[85] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision (ICCV)*, 2019.

[86] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

[87] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[88] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[89] Aron Monszpart, Paul Guerrero, Duygu Ceylan, Ersin Yumer, and Niloy J. Mitra. Imapper: Interaction-guided scene mapping from monocular videos. *ACM Transactions on Graphics*, 38, 2019.

[90] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *International Conference on Computer Vision (ICCV)*, 2019.

[91] Jorge Nocedal and Stephen J Wright. Nonlinear equations. *Numerical Optimization*, 2006.

[92] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[93] Stuart Geman. Statistical methods for tomographic image reconstruction. *Bulletin of International Statistical Institute*, 4, 1987.

[94] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016.

[95] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[96] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. Character controllers using motion vaes. *ACM Transactions on Graphics*, 39, 2020.