

From Sparse Signal to Smooth Motion: Real-Time Motion Generation with Rolling Prediction Models

Supplementary Material

This supplementary material is structured as follows. In [Sec. A](#), we provide additional details on the protocol followed to collect and process the new GORP dataset. Then, we describe all the implementation details of our Rolling Prediction Model (RPM) in [Sec. B](#). We also give more insights on both the synthetic and real benchmarks in [Sec. C](#). In [Sec. D](#) and [E](#), we show quantitative studies on the effects of modifying the free-running length, and the uncertainty function, respectively. In [Sec. F](#), we further discuss why RPM is not fed with the previous prediction, and provide additional experiments supporting our choice. In [Sec. G](#) we include additional experiments showing how RPM provides the best accuracy-smoothness trade-off. In [Sec. H](#), we analyze RPM’s robustness over long runs. In [Sec. I](#), we support experimentally the choice of the hyperbolic tangent as the scaling function for PCAF. Finally, we show and discuss additional qualitative comparisons with the state of the art in [Sec. J](#).

A. GORP dataset

Data collection setup. The hardware setup includes a Meta Quest 3 headset, an Optitrack motion capture system with an eSync2 unit [5] for synchronization, and a proprietary synchronization device, *QuestSync*, as shown in [Fig. A](#). The *QuestSync* triggers data capture for both systems, and provides timestamps from a shared timeline to align the captured data. The *QuestSync* pairs with Quest 3 wirelessly through the same mechanism as controller pairing. It also utilizes the same synchronization mechanism between the headset and controllers to synchronize both using its timing signals. The *QuestSync* connects to eSync2 with a DIN-to-BNC cable, and provides it with a 30-Hz SMTPE time-code [21]. The eSync2 can further subdivide this 30 Hz signal to 120 Hz for the Optitrack cameras. Since both the Quest 3 cameras and the Optitrack cameras rely on Infrared (IR) light for blob tracking, the *QuestSync* offsets trigger signals to eSync2 to prevent IR lights from Optitrack interfering with the Quest 3.

Both the Quest 3 and Optitrack stream data in real time to a PC that runs a proprietary data collection software. The Quest 3 streams tracking data through the Android Debug Bridge (ADB) and Meta Quest Link [17]. Optitrack streams 3D marker data through NatNet SDK [18]. These data streams include accurate capture timestamps from *QuestSync* in their protocol, so transmission and software latency is not a concern.

The Quest 3 already calibrates the headset tracking data

with controller tracking data and hand tracking data internally. With synchronized marker data from Optitrack, we want to transform all Quest 3 data into the Optitrack space. To do so, we attach a marker tree on the Quest 3 headset as an Optitrack rigid body during data capture. The rigid transformation of the marker tree in Optitrack space, \mathbf{M}_t^{MT} , should represent the same motion as the Quest 3 headset tracking output in the Quest 3 space, \mathbf{M}_t^{VR} . We compute the constant transformation \mathbf{T} from the Quest 3 space to the Optitrack space using [Eq. A](#) for each capture sequence, where \mathbf{O} represents the constant offset of the headset in the marker tree rigid body space.

$$\mathbf{T}^*, \mathbf{O}^* = \underset{\mathbf{T}, \mathbf{O}}{\operatorname{argmin}} \sum_{t=0}^N \|(\mathbf{T} \cdot \mathbf{M}_t^{VR}) - (\mathbf{M}_t^{MT} \cdot \mathbf{O})\|^2 \quad (\text{A})$$

Thanks to the accurate data synchronization, we achieve a calibration error of less than one millimeter on average across the entire dataset.

Data streams. We retrieve a rich set of egocentric tracking signals from Quest 3. For both the headset and the controllers, it provides a rigid transformation, linear and angular velocities, and linear and angular accelerations at every time step, all synchronized and calibrated to the same space. The rigid transformation is computed through Visual Inertial Odometry (VIO) from stereo images and Inertial Measurement Units (IMUs) at 30 Hz. The velocities and accelerations are computed from high-frequency IMUs and then downsampled to 30 Hz, so they are much more accurate than finite differences of the positional data. In addition, each device has a flag to indicate whether its data is valid. This flag is always true for the headset in our dataset, but it may be false for the controllers when they are outside of the field of view of the headset cameras for an extended period. The origin of the headset is at the center between the eyes, and the origins of the controllers are somewhere between the thumb and the index finger, depending on how a user holds them.

Hand tracking data is exclusive to controller data in our dataset because it was collected before Multimodal mode [23] was available on the Quest 3. The hands are tracked from the headset cameras [9] at 30 Hz. For the purpose of our project, we captured only the rigid transformations of the wrists, as well as tracking confidence of each hand.

For body motion ground truth, we use SOMA and Mosh++ [8, 16] to fit SMPL body parameters to the Optitrack markers.



Figure A. Left: the QuestSync device. Right: Meta Quest 3 with two marker trees attached.

Data protocol. Our dataset consists of 28 participants playing VR games using controllers and hands for 15 minutes respectively. We recruit participants from diverse demographics with various VR experience levels. There are 17 male participants and 11 female participants, ranging from 20 to 65 years old, who have played in VR for 0 to 10+ times. In a capture session, a participant wears a tight suit with reflective markers attached and puts on the Quest 3 headset, as shown in Fig. B. After they feel comfortable in the VR environment, they are asked to choose one of two controller-based games, Beat Saber [1] or Fruit Ninja [7]. They can choose the game settings freely and start playing. After about 15 minutes or when they want to stop, they will take a 5 minutes break. Next, they are asked to choose one of two hand tracking based games, Hand Physics Lab [10] or Rogue Ascent VR [19]. They then play in the settings they choose for another 15 minutes or until they want to stop. They can also break or stop any time during the session. In total, we collected about 17 hours of realistic VR gameplay data with ground truth body motions.

B. Implementation details

Architecture details. The design of our model, f_θ , is inspired by [22]. First, we concatenate the sequence of past motion \mathcal{X}_t with a sequence of W blank pose tokens (i.e., all zeros), which are passed through a linear layer and merged with a sinusoidal positional encoding (PE). A cross-attention layer then uses these tokens as queries to attend to the past and current tracking inputs, \mathcal{C}_t , which are fed as keys and values after adding a sinusoidal PE. The result is processed by four Transformer encoder layers [24] with GeLU activation functions [11]. The first M output feature vectors are discarded, retaining only the last W . Each attention layer incorporates layer normalization and a residual connection. Finally, a linear layer transforms the output features into the predicted poses, $f_\theta(\mathcal{X}_t, \mathcal{C}_t)$, which are then used by our Prediction Consistency Anchor Function (PCAF) to update the previous prediction, P_{t-1} , into P_t . The dimension of the Transformer latent space is set to 512. Feature vectors have a dimension of 132 for poses in \mathcal{X}_t and P_t , and 54 for tracking inputs in \mathcal{C}_t .

Further implementation details. We optimize RPM using Adam [15] and train it for 80k iterations with a batch size of 512. The initial learning rate is set to $3e-4$, which decreases to $3e-5$ after 50k iterations. Weight decay is set to $1e-4$. The weights for all losses in Eq. 3 are set to 1. During training, we simulate tracking signal losses by dropping a segment of tracking inputs of length $L \sim \mathcal{U}(1, I+1+FR)$ with a 10% probability, where I is the number of past tracking inputs used, and FR the maximum free-running length. For all baselines, $L \sim \mathcal{U}(1, I_{\text{baseline}})$, where I_{baseline} is the length of the tracking input sequence used by each baseline. For the AGRoL baseline [4], we generate motion for the first 196 frames offline, as proposed in the original implementation. We then use this output to autoregressively inpaint a new pose at the end. For RPM, both the motion context and the initial prediction are initialized with ground truth poses. To avoid evaluating during this period, the first second of all sequences is discarded.

C. Benchmarks details

Synthetic inputs: AMASS (A-P1 and A-P2). We build the motion controllers (MC) synthetic setup on AMASS in the same way as prior works [4, 12], that is, taking the head and wrists joint positions as tracking inputs. For the hand-tracking (HT) setup, we simulate gaps in the tracking inputs. Specifically, for each wrist in each motion sequence, we simulate periods of tracking signal loss with length $L \sim \mathcal{U}(0.5, 2)$ seconds, with a 2.5% probability of starting at a given timestep. We also enforce a distance of 2 s between consecutive gaps so we can measure the models’ ability to recover from a tracking signal loss.

Real inputs: GORP. The GORP dataset was randomly split into subject-independent training and test sets, ensuring an 80/20 proportion of total duration, respectively. All models were trained using both sequences with real MC or HT sensing signals. We approximated the 6-DOF of the wrists and the SMPL head by adding a fixed offset to the motion controllers and headset position. The confidence threshold of the hand-tracking system was set to 0.8, resulting in 3.16/1.50% of missing left/right hands in the training set and 4.05/1.49% in the test set. The average length of tracking signal loss segments is 0.84s for the training set and 0.77s for the test set.

D. Analysis of the free-running length

In Fig. C, we show the effects of the free-running length on the accuracy and smoothness of the generated motion for A-P1. Overall, we observe that longer free-running periods result in higher accuracy (i.e., lower MPJRE and MPJPE), with values converging to the best performance at around 50 frames (0.83s). Regarding smoothness, longer FR values cause a slight increase in jitter (from 3.8 to 4.4) and a

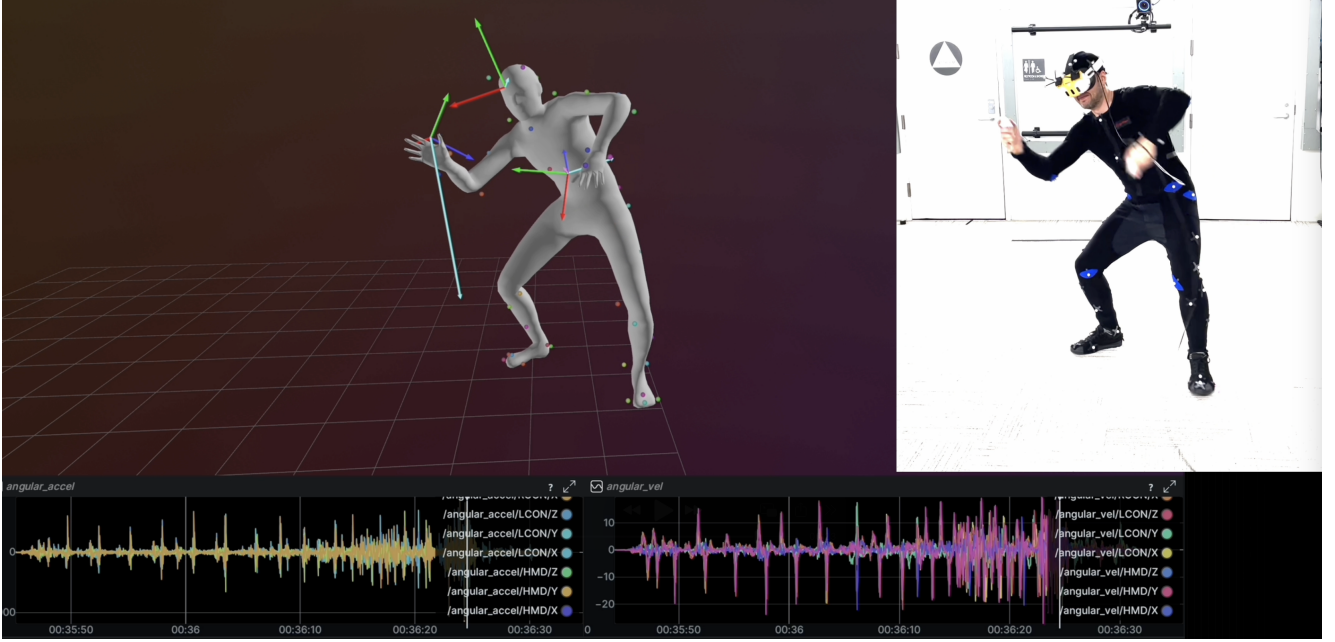


Figure B. We capture both tracking signals from the Quest 3 headset and ground truth body motions from Optitrack. The Quest 3 tracking signal includes transformations of the headset and the controllers (or the wrists – not shown here), linear velocities (purple arrows), linear accelerations (cyan arrows), angular velocity (bottom right), and angular accelerations (bottom left). The velocities and accelerations are from high-frequency IMU data. Ground truth body motions are solved from labeled 3D marker data (colored points).

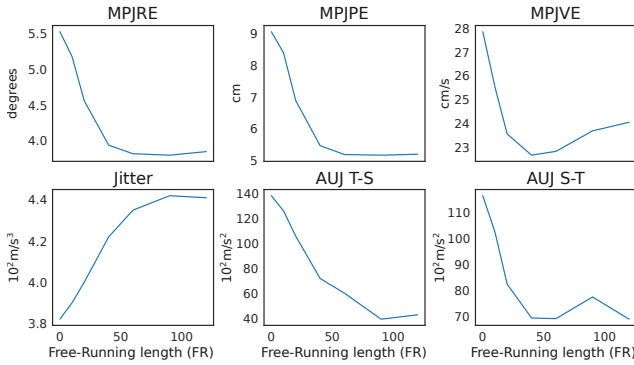


Figure C. **Free-running length.** We observe that the free-running stage during RPM’s training is essential for making the network robust to mismatches between tracking inputs (\mathcal{C}_t) and previously generated motion (\mathcal{X}_t). Both accuracy (MPJRE, MPJPE) and smoothness during transitions (AUJ T-S/S-T) improve significantly with longer free-running periods.

reduction in the AUJ for both types of transition. We hypothesize that the improved smoothness during transitions is due to a better simulation at training time of conditions encountered during inference, such as mismatches between motion context and tracking inputs after periods of tracking signal loss (see Fig. 3, right). The best smoothness values during transitions are achieved at around FR=90 frames (1.5s). For all our experiments, we use FR=60 frames (1s),

as it offers a good balance between accuracy, smoothness, and training efficiency.

E. Analysis of the uncertainty function

In Sec. 3.2, we introduce the PCAF reparameterization (Eq. 2), which incorporates an uncertainty function to regulate the correction magnitude allowed at each future time horizon. The objective is to model the increasing uncertainty of future motion as the prediction extends further into the future. Consequently, the corrections applied to previous predictions should scale proportionally with this uncertainty. We explored three different uncertainty functions: cosine (Eq. B), cosine squared (Eq. C), and linear (Eq. D):

$$f_{\cos}(\tau) = 1 - \cos\left(\frac{\tau+1}{W} \cdot \frac{\pi}{2}\right), \quad (\text{B})$$

$$f_{\cos \text{ sq.}}(\tau) = 1 - \cos\left(\frac{\tau+1}{W} \cdot \frac{\pi}{2}\right)^2, \quad (\text{C})$$

$$f_{\text{lin}}(\tau) = \frac{\tau+1}{W}, \quad (\text{D})$$

where $\tau \in [0, W-1]$ represents the distance to the present, in frames. A visualization of the three functions with $W=10$ is provided in Fig. D. The results of RPM trained on A-P1 are presented in Tab. A. While accuracy metrics are comparable across the three functions, the cosine function

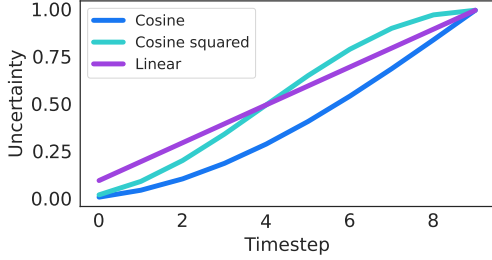


Figure D. **Visualization of the uncertainty functions.** We explored three uncertainty functions: cosine (Eq. B), cosine squared (Eq. C), and linear (Eq. D). In all cases, the uncertainty level increases with the distance from the present.

Uncertainty (U)	MPJRE	MPJPE	MPJVE	Jitter	AUJ _{T-S}	AUJ _{S-T}
Cosine	3.82	5.18	22.83	4.35	60.51	69.02
Cosine squared	3.92	5.37	22.51	4.28	60.85	78.97
Linear	3.92	5.33	22.05	4.25	71.55	80.47

Table A. **Comparison of different uncertainty functions.** We observe that modifying the uncertainty function of PCAF primarily impacts the smoothness during synthesis-to-tracking (AUJ_{S-T}) and tracking-to-synthesis (AUJ_{T-S}) transitions.

produces smoother motion during transitions (as shown by lower AUJ_{T-S} and AUJ_{S-T}). Given the importance of smooth transitions for the problem we tackle in this work, we selected the cosine function as the uncertainty function for all our experiments.

F. On the iterative prediction refinement

RPM uses the PCAF reparameterization to refine the previous prediction, \mathcal{P}_{t-1} . However, this previous prediction is not fed directly to f_θ , which only observes the previously generated motion, \mathcal{X}_t , and the history of tracking inputs, \mathcal{C}_t . At the end of Sec. 3.2, we note that \mathcal{X}_t serves as a proxy for the previous prediction \mathcal{P}_{t-1} under the deterministic paradigm, which assumes that only a single future is predicted, as opposed to a multimodal distribution (i.e., stochastic prediction). To validate this claim, we train a version of RPM that is explicitly fed with \mathcal{P}_{t-1} , thereby providing direct access to the previous prediction. The results, shown in Tab. B, support our hypothesis: explicit access to \mathcal{P}_{t-1} neither improves accuracy nor generates smoother results. We also train another version where, instead of feeding the previous prediction, we provide a noisy version of it to simulate the uncertainty of the future as a decreasing signal-to-noise ratio, inspired by rolling diffusion models [20, 25]. Similarly, no significant improvement is observed over our simpler definition. Nonetheless, we hypothesize that these alternative definitions could be beneficial if RPM were extended to refine stochastic human motion

Forward function	MPJRE	MPJPE	MPJVE	Jitter	AUJ _{T-S}	AUJ _{S-T}
$f_\theta(\mathcal{X}_t, \mathcal{C}_t)$	3.82	5.18	22.83	4.35	60.51	69.02
$f_\theta(\mathcal{X}_t, \mathcal{C}_t, \mathcal{P}_{t-1})$	3.80	5.19	22.47	4.12	82.75	65.22
$f_\theta(\mathcal{X}_t, \mathcal{C}_t, \mathcal{P}_{t-1}^{\text{noisy}})$	3.78	5.13	22.97	5.40	55.74	103.23

Table B. Study on different techniques to refine the rolling prediction with RPM. We observe that explicitly feeding the previous prediction (\mathcal{P}_{t-1}) leads to results similar to those of our simpler model (1st row). Similarly, simulating the uncertainty of the previous prediction by adding noise with increasing variance to it ($\mathcal{P}_{t-1}^{\text{noisy}}$), as proposed in [20], does not lead to a better performance than RPM either.

predictions, where previous predictions may not be directly inferable from \mathcal{X}_t due to their multimodal nature.

G. Accuracy-smoothness trade-off

In Sec. 4, we discussed how current state-of-the-art methods lag behind RPM in terms of smoothness and, consequently, motion realism. In return, our model sacrifices some accuracy. One might think that state-of-the-art methods could also produce such smooth results if they were willing to sacrifice accuracy, for instance, by incorporating a low-pass filter. To test this hypothesis, we applied a 1€ filter [2] to the output of our baselines. The filter parameters were optimized through grid search to maximize the smoothness metrics. However, we found that achieving competitive smoothness introduced excessive latency to the generated motion, significantly reducing accuracy. To further explore this accuracy-smoothness trade-off, we identified another set of filter parameters offering the best balance between accuracy and smoothness. We refer to these two sets of parameters as smooth and reactive, respectively. Results in Tab. C demonstrate that all baselines still produce motion with more discontinuities than RPM when using a mild low-pass filter that minimally affects accuracy. When targeting smoothness levels comparable to RPM for synthesis-to-tracking transitions, their accuracy drops by more than 50%, placing them far behind RPM in terms of accuracy. We illustrate this trade-off in Fig. E. Our model provides the optimal accuracy-smoothness trade-off, particularly during synthesis-to-tracking transitions, which was the primary goal of our work.

H. Performance over time

Autoregressive methods may experience degeneration issues over time. To showcase RPM’s robustness and its potential for real-world deployment, we plot the error evolution over 25 seconds in the real MC/HT setups using the GORP dataset in Fig. F. We observe that the error remains stable in both setups. Additionally, the wrist error does not escalate and stays close to the overall error’s magnitude.

Model	MPJRE	MPJPE	MPJVE	Jitter	PJ _{T-S}	AUJ _{T-S}	PJ _{S-T}	AUJ _{S-T}
AvatarPoser [12]	5.62	8.38	44.26	26.10	89.87	2215.07	86.93	2133.20
+ 1€ (reactive)	5.77	8.84	40.45	9.34	26.30	429.34	25.93	389.75
+ 1€ (smooth)	6.13	10.24	43.69	6.21	17.46	129.79	16.48	110.13
EgoPoser [14]	4.61	6.29	42.91	29.15	461.18	3356.38	567.42	3773.29
+ 1€ (reactive)	4.75	6.71	34.63	7.57	85.27	491.61	108.95	606.55
+ 1€ (smooth)	5.26	8.51	40.44	4.89	28.37	93.64	38.00	190.45
SAGE [6]	4.21	5.50	46.38	33.55	1056.33	6337.79	1073.65	4131.33
+ 1€ (reactive)	4.40	6.06	35.90	9.82	178.16	1064.44	188.51	733.46
+ 1€ (smooth)	4.99	8.09	41.22	6.28	53.90	308.76	56.25	306.36
AvatarJLM [26]	4.18	4.59	27.30	12.79	114.36	811.13	901.75	2008.13
+ 1€ (reactive)	4.44	5.44	29.35	4.67	19.82	35.86	156.25	242.25
+ 1€ (smooth)	5.07	7.83	39.66	4.03	10.47	87.31	46.31	102.73
HMD-Poser [3]	3.34	4.04	22.34	7.35	23.84	302.58	461.91	1236.47
+ 1€ (reactive)	3.71	5.00	27.18	3.98	9.83	69.17	68.38	186.21
+ 1€ (smooth)	4.55	7.54	39.06	4.10	10.55	75.98	23.75	60.14
RPM - Reactive	3.82	5.18	22.83	4.35	15.28	60.51	18.98	69.02
RPM - Smooth	3.98	5.44	24.04	4.29	8.41	84.81	12.12	50.23

Table C. **Comparison of RPM with baselines using 1€ filters on A-P1-HT.** Applying both versions of the 1€ filter (reactive and smooth versions, see Sec. G) to baselines improves smoothness but significantly reduces accuracy because of latency. To match RPM’s smoothness levels (AUJ_{S-T}), baselines must sacrifice over 50% of their accuracy (MPJPE).

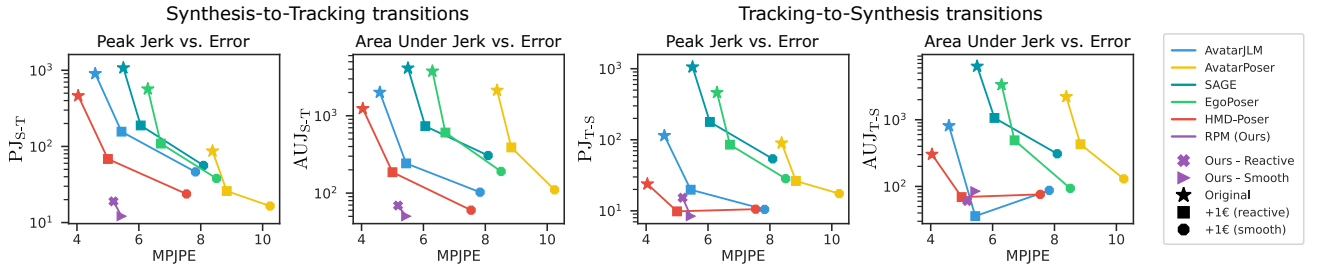


Figure E. **Accuracy-smoothness trade-off on A-P1-HT.** Baselines require significant accuracy loss to achieve competitive smoothness when combined with traditional low-pass filters. On synthesis-to-tracking transitions, RPM achieves the best balance to date, positioning itself in the lower-left of the plot and surpassing the previous Pareto frontier set by HMD-Poser. At the same time, RPM attains a similar trade-off as HMD-Poser and AvatarJLM on tracking-to-synthesis transitions.

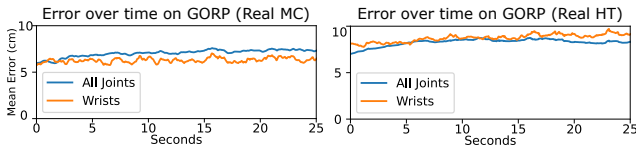


Figure F. **Error evolution over time.** RPM’s error on both real MC/HT setups in the GORP dataset remains stable over time, demonstrating its robustness in long executions. Wrist error also remains stable and is comparable in magnitude to the overall error.

Interestingly, despite the tracker position being available, the network still struggles to match it closely. This issue is commonly observed in methods synthesizing full-body motion from sparse tracking inputs [13]. We argue that this is expected in RPMs, as certain accuracy is traded off for smoother and more realistic motion, as discussed in Sec. G.

I. Scaling function in PCAF

At the end of Sec. 3.2, we discussed the rationale behind selecting the hyperbolic tangent as the scaling function in PCAF. In this section, we present the corresponding experimental validation. We trained RPM with PCAF using three functions: the hyperbolic tangent, a scaled sigmoid ($2 * \text{sigmoid}(\dots) - 1$), and a linear function constrained between -1 and 1 (referred to as *straight*). As shown in Tab. D, the model trained with the hyperbolic tangent produces the smoothest synthesis-to-tracking transitions. The *straight* function achieves similar accuracy to the hyperbolic tangent, but with slightly more abrupt transitions. In contrast, the sigmoid function leads to transitions that are too abrupt. We hypothesize that the sigmoid function generates large gradients when substantial correction is needed, promoting strong corrections during prediction refinement rather than improving long-term predictions. This issue does not occur

	MPJPE	MPJVE	Jitter	AUJ _{T-S}	AUJ _{S-T}
RPM - Reactive					
Tanh	5.18	22.83	4.35	60.51	69.02
Sigmoid	5.21	23.45	5.14	55.11	137.88
Straight	5.11	22.58	4.30	61.55	71.19
RPM - Smooth					
Tanh	5.59	23.80	5.05	58.70	175.86
Sigmoid	5.63	25.32	6.76	232.50	431.62
Straight	5.59	23.69	5.37	53.68	187.27

Table D. **Effect of PCAF scaling function iA-P1-HT.** We observe how the hyperbolic tangent and the straight functions provide the best accuracy and transitions smoothness.

with the other two functions. We selected the hyperbolic tangent for its continuity and differentiability.

J. Additional qualitative results

We provide additional visual comparisons in Fig. G for A-P1 and in Fig. H for GORP. In this section, we discuss several improvements that RPM offers compared to the state of the art in terms of visual quality. We refer the reader to the attached videos, which showcase all the qualitative results from the main paper and supplementary material. Additionally, we include a demo video highlighting RPM’s strengths in comparison to the state of the art.

Synthesis-to-tracking transitioning. These additional visual comparisons reinforce our observations from Sec. 4.3: only our method generates smooth and realistic synthesis-to-tracking transitions. While AvatarJLM and HMD-Poser repeatedly snap the hand to the recovered tracking signal after each loss, RPM instead smoothly catches up with the tracking signal’s position and motion dynamics.

Robustness to noise. RPM demonstrates greater robustness to noisy tracking inputs. By reformulating the pose generation task as a sequential refinement of previously predicted motion, the network learns to better handle sudden false hand-tracking detections (Fig. H-left). We hypothesize that, at training time, these false positives fail to provide a signal capable of refining high-frequency motion details, which are incorporated during the last refining step before the pose generation. Instead, the network leverages such new signal to correct the low-frequency characteristics of the long-term predicted motion. However, these corrections do not persist under noisy hand-tracking inputs, as subsequent accurate hand-tracking inputs override them.

Full-body pose coherence. Lastly, our method maintains the coherence of the full-body pose along time. The main reason behind this is RPM’s explicit conditioning on the previously generated motion (\mathcal{X}_t), instead of just the past and current tracking inputs (\mathcal{C}_t) as other methods do. In GORP, this ensures that dynamic upper-body mo-

tion leading to torso rotations does not result in unrealistic lower-body rotations (Fig. H-right). Consequently, we observe more expressive lower-body motion, including stepping while turning. Conversely, in AvatarJLM and HMD-Poser, the tendency to align the lower-body with an average pose often causes severe foot sliding – i.e., moving feet while standing as if the person floated. Notably, foot sliding remains a common issue across all methods for this task, including ours. Further research is needed in this area, as we hypothesize that the lack of body shape awareness and the reliance on headset-driven motion are potential reasons for it. We highlight the value of GORP as a benchmark for addressing this challenge, as its motion is predominantly in place, featuring frequent small in-place steps, knee torsion, and significant upper-body movement. These characteristics require networks to adapt the entire kinematic chain, including the lower-body and the spine, to minimize foot-sliding artifacts.

References

- [1] Beat Saber on Meta Quest, last visited: 2024/11/10. <https://www.meta.com/experiences/beat-saber/2448060205267927/>. 2
- [2] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530, 2012. 4
- [3] Peng Dai, Yang Zhang, Tao Liu, Zhen Fan, Tianyuan Du, Zhuo Su, Xiaozheng Zheng, and Zeming Li. Hmd-poser: On-device real-time human motion tracking from scalable sparse observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 874–884, 2024. 5
- [4] Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakoyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 481–490, 2023. 2
- [5] eSync2 - A PoE Synchronization Device, last visited: 2024/11/10. <https://optitrack.com/accessories/sync-networking/esync-2/>. 1
- [6] Han Feng, Wenchao Ma, Quankai Gao, Xianwei Zheng, Nan Xue, and Huijuan Xu. Stratified avatar generation from sparse observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 153–163, 2024. 5
- [7] Fruit Ninja on Meta Quest, last visited: 2024/11/10. <https://www.meta.com/experiences/fruit-ninja/2215140511885250/>. 2
- [8] Nima Ghorbani and Michael J. Black. SOMA: Solving optical marker-based mocap automatically. In *Proc. International Conference on Computer Vision (ICCV)*, pages 11117–11126, 2021. 1
- [9] Shangchen Han, Po-Chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang,

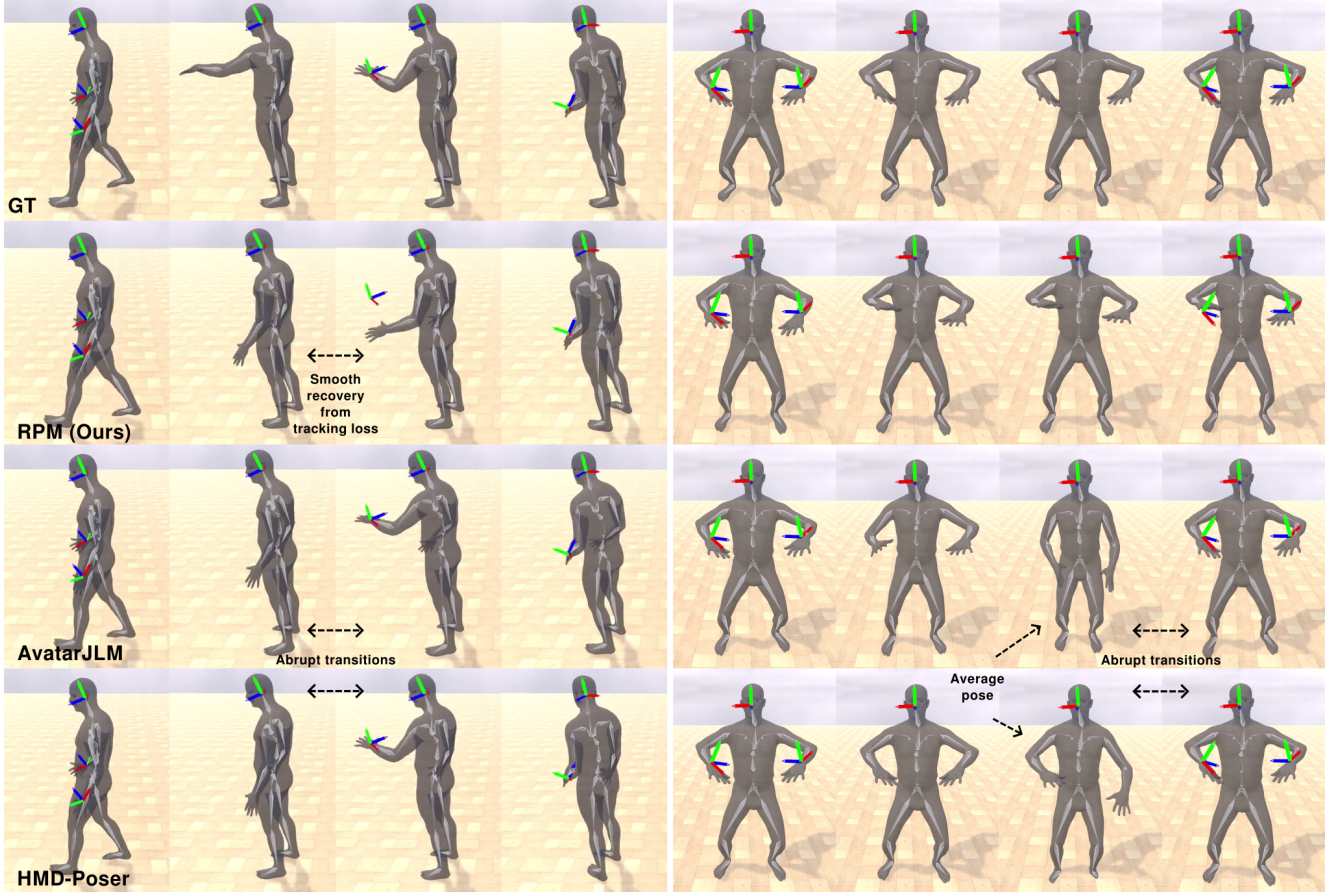


Figure G. **More qualitative results on synthetic hand-tracking (HT) inputs (A-P1).** On the left, we observe that when the left-hand tracking is lost, all methods fail to generate the *grabbing* action. Once the hand-tracking signal is recovered, AvatarJLM and HMD-Poser snap the hand abruptly to correct the mistake, whereas our method generates a smoother and more realistic transition to match the signal again. On the right, we present an example illustrating how state-of-the-art methods tend to generate an average pose as uncertainty increases during prolonged hand-tracking signal losses. In contrast, RPM produces motion that remains coherent with the past context.

- Yujun Cai, Tomas Hodan, Randi Cabezas, Luan Tran, Muzaffer Akbay, Tsz-Ho Yu, Cem Keskin, and Robert Wang. Umetrack: Unified multi-view end-to-end hand tracking for VR. In *SIGGRAPH Asia 2022 Conference Papers, SA 2022, Daegu, Republic of Korea, December 6-9, 2022*, 2022. 1
- [10] Hand Physics Lab on Meta Quest, last visited: 2024/11/10. <https://www.meta.com/experiences/hand-physics-lab/3392175350802835/>. 2
- [11] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2
- [12] Jiaxi Jiang, Paul Streli, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. In *European conference on computer vision*, pages 443–460. Springer, 2022. 2, 5
- [13] Jiaxi Jiang, Paul Streli, Xuejing Luo, Christoph Gebhardt, and Christian Holz. Manikin: biomechanically accurate neural inverse kinematics for human motion estimation. In *European Conference on Computer Vision*, pages 128–146. Springer, 2024. 5
- [14] Jiaxi Jiang, Paul Streli, Manuel Meier, and Christian Holz. Egoposer: Robust real-time egocentric pose estimation from sparse and intermittent observations everywhere. In *European Conference on Computer Vision*, pages 277–294. Springer, 2025. 5
- [15] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2
- [16] 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*, pages 5442–5451, 2019. 1
- [17] Meta Quest Link Cable — Connect PC to VR Headset, last visited: 2024/11/10. <https://www.meta.com/quest/accessories/link-cable/>. 1
- [18] Optitrack - NatNet SDK, last visited: 2024/11/10. <https://optitrack.com/software/natnet-sdk/>. 1
- [19] Rogue Ascent VR on Meta Quest, last visited: 2024/11/10. <https://www.meta.com/experiences/rogue-ascent-vr/5395586720470296/>. 2
- [20] David Ruhe, Jonathan Heck, Tim Salimans, and Emiel



Figure H. **More qualitative results on real hand-tracking (HT) inputs (GORP).** On the left, we observe that false-positive hand-tracking detections cause AvatarJLM and HMD-Poser to disrupt the continuity of the previous motion by abruptly aligning with the new tracking input. In contrast, RPM generates motion that maintains coherence with prior predictions, thereby avoiding instantaneous hand snapping. On the right, we see that state-of-the-art methods often generate lower-body average poses, simply aligning their orientation with the tracking inputs. Instead, RPM preserves coherence with the prior lower-body motion, producing more realistic full-body motion.

- Hoogetboom. Rolling diffusion models. In *Forty-first International Conference on Machine Learning*, 2024. 4
- [21] SMPTE Timecode, last visited: 2024/11/10. https://en.wikipedia.org/wiki/SMPTE_timecode. 1
- [22] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. 2
- [23] User Simultaneous Hands and Controllers (Multimodal), last visited: 2024/11/10. <https://developers.meta.com/horizon/documentation/unity/unity-multimodal/>. 1
- [24] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 2
- [25] Zihan Zhang, Richard Liu, Rana Hanocka, and Kfir Aberman. Tedi: Temporally-entangled diffusion for long-term motion synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 4
- [26] Xiaozheng Zheng, Zhuo Su, Chao Wen, Zhou Xue, and
- Xiaojie Jin. Realistic full-body tracking from sparse observations via joint-level modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14678–14688, 2023. 5