A. Supplementary

A.1. Training and Evaluation Details

In the following section, we provide additional implementation and evaluation details.

The losses described in Sec. 3.2.3 are weighted by the following weights: $\lambda_{\text{simple}} = 5$, $\lambda_{\text{tran}} = 5$, $\lambda_{\text{smpl}} = 1$, and $\lambda_{\text{vel}} = 0.3$. The velocity loss is activated after 10 epochs, once the prediction has stabilized.

UDP uses an LSTM with a hidden state size of 512 with 2 layers and a dropout rate of 0.4 for the denoiser network. The motion history length is set to $N_H = 30$, and the motion window size is $N = 170$ frames at 60 FPS during training. The model is trained for 150 epochs with a weight decay of $1\text{e}{-5}$, an initial learning rate of $1\text{e}{-4}$, and step decay with a factor of $\gamma = 0.33$ every 50 epochs. We use a square root diffusion noise schedule.

The rotation estimator uses a hidden state dimension of 512, 3 layers, and a dropout rate of 0.2. We enable UWB-Diffusion Guidance at diffusion step $t = 20$, with $\lambda = 500$ and a weaker $\lambda = 50$ for the GIP-DB and UIP-DB, where UWB-signals are less reliable. Overall, the model contains 11.3M parameters, and training takes approximately 5 hours on an RTX 4090 GPU with a batch size of 256.

Following prior work [4, 19, 45], the model is initially trained on the synthetic AMASS dataset with DanceDB held out and directly evaluated on TotalCaptureReal and DanceDB. This is the complete list of Datasets used during training: HumanEva, MPI_HDM05, SFU, MPI_mosh, Transitions_mocap, SSM_synced, CMU, TotalCapture, Eyes_Japan_Dataset, KIT, BMLmovi, EKUT, TCD_handMocap, ACCAD, BioMotionLab_NTroje, BMLhandball, MPI_Limits, DFaust_67.

DIP-IMU and TotalCapture provide IMU signals coming from an XSense suit [40]. For TotalCapture, we use the DIP sensor calibration. All AMASS training data and DanceDB use synthesized IMU data, obtained via finite differences on the motion data following [4, 45]. UWB measurements are synthesized from the motion ground truth for AMASS training data, DanceDB, TotalCapture, and DIP-IMU. UIP-DB [4] and GIP-DB [42] obtain all IMU data through off-the-shelf, low-cost commercial IMU and UWB sensors, and we use these real-world IMU signals and UWB measurements directly. For DIP-IMU, the model is fine-tuned on the training split for 30 epochs and then evaluated on the test split. As UIP-DB, and GIP-DB present a more challenging scenario with lots of real-world sensor noise, we train this model with noise augmentations. We introduce Gaussian noise with $\sigma_{\text{acc}} = 0.6$ to the acceleration measurements and apply random rotation augmentation with Gaussian noise of $\sigma_{\text{ori}} = 0.15$, and add a random Gaussian bias of $\sigma_{\text{ori-bias}} = 0.054$ and a stronger weight decay of $4e - 5$. For UWB distance noise, we add Gaussian noise with the standard deviation based on the reported mean UWB distance errors from UIP-DB [4] scaled by 1.5. The model is pretrained for 50 epochs using 200 diffusion steps to improve robustness to noise. For fine-tuning, we take the model pretrained on purely synthetic data and train it for 5 epochs on the training split of either UIP-DB or GIP-DB (depending on which dataset is being evaluated). We then evaluate the model on the corresponding test split.

For consistency, we report results from the original authors or from previously published works. If the required metrics are not available, we reproduce the results ourselves by utilizing the provided weights. We retrain the network if no weights are provided for that evaluation. Since PIP does not provide training code, we implemented our own training pipeline and trained the model from scratch, ensuring that no test data from DanceDB leaked into the training process. PNP [46] does not provide data processing code and provide the reported results on TotalCapture and DIP. DynaIP does not predict translation; hence, we only report the results of the translation-free DIP dataset. UMotion uses a slightly different evaluation scheme; we used the authors' provided model weights and reran the results. During inference, we disable the use of ground-truth distances for updating the Kalman filter covariances, see here. Instead, we assume an 8 cm standard deviation for the UWB measurements and set the measurement covariance accordingly. Additionally, we report results in two settings: First, the zero-shot results reported on UIP-DB and GIP-DB use the raw UWB distances. Second, the finetuned results that clean the raw UWB distances using the training split, as seen here in the authors code. The reported results are obtained following the same evaluation framework as previous work [4, 45, 46]. Evaluation metrics follow the conventions established in prior work [4, 45, 46], excluding end joints that cannot be reliably estimated from sparse sensor input—specifically, the wrists, hands, root, toes, and ankles.

A.2. Additional Evaluations

We provide additional qualitative comparisons in the accompanying video and in Figures 5 and 6. As shown, UDP achieves better foot and arm placement than prior work, particularly in squatting motions, where it better estimates foot positioning while maintaining precise torso alignment.

Conversely, methods relying on post-optimization physics simulations (e.g., PIP, UIP, PNP) often struggle with highly dynamic motions, such as dance routines involving rapid twirls or intentional foot sliding (see the bottom row of Fig. 5). In these scenarios, the physics optimizer may be a bottleneck due to rigid constraint modeling. Specifically, when the network predicts a foot contact, the optimizer enforces strict zero-velocity or high-friction constraints. Consequently, intentional sliding motions may lead to wrong motion estimation when the model inaccurately

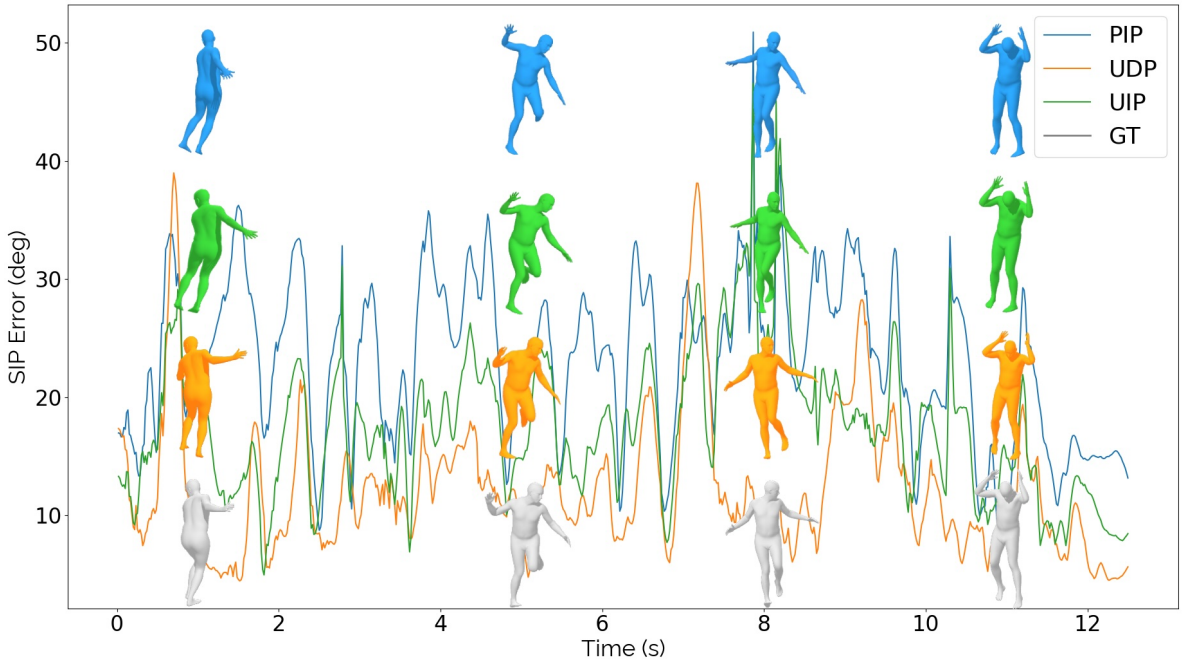

Figure 4. Qualitative results over a sequence of DanceDB. UDP has generally lower SIP error compared to previous methods.

predicts foot contacts. In contrast, UDP, being purely data-driven, is not bound by contact enforcement, friction models, or hyperparameter choices within the physics model and avoids such interference, allowing for the accurate reconstruction of complex maneuvers.

While physics optimizers remain a compelling alternative for applications where strict physical plausibility (e.g., non-penetration) takes precedence over tracking fidelity, our results demonstrate that UDP yields smooth, accurate motion, particularly in limb placement, while benefiting from the simplicity of a purely data-driven approach.

A.2.1. Additional Ablation Studies

The following sections extend the analysis in the main paper by providing detailed ablation studies regarding inference dynamics, noise scheduling, temporal lookahead, and architectural alternatives.

A.2.2. Sampling Steps, Inference and Realtime

UDP operates in real-time and significantly improves inference speed over UIP. Unlike UIP, it avoids the post-processing physics optimizer, which formulates refinement as a Quadratic Programming (QP) problem. This QP solver acts as a computational bottleneck, as it relies on serial CPU execution and cannot benefit from GPU acceleration. In contrast, UDP utilizes purely neural-based inference that is fully accelerated on the GPU.

Tab. 5 shows that UDP achieves 3.6 times faster inference compared to UIP when running on an RTX 4090. By using DDIM [28], we can reduce sampling steps to just 5. This increases inference speed by a factor of 14 compared to DDPM sampling with 50 steps, while the SIP error increases by only 4.8%. We observe that the pose prediction quality stabilizes quickly: the transition from 50 steps (DDPM) to 25 steps (DDIM) induces the largest drop in quality, while further reducing the steps to 5 has a negligible impact. In comparison to UIP, UDP with 5 sampling steps enables 50 times faster inference while still delivering superior output quality.

Table 5. Evaluation of the total inference time (TT) on the TotalCapture dataset (approx. 2,466,s duration) and the mean inference time per frame (MTPF). For 60FPS real-time inference, 16,ms MTPF or lower is required. Reducing the number of diffusion steps reduces pose estimation accuracy slightly but increases inference speed drastically.

Metric	SIP (°) ↓	JPE (cm) ↓	TT (s) ↓	MTPF (ms) ↓
TIP	11.36	5.15	568	3.938
UIP	10.70	5.11	1,342	7.545
UDP DDIM 5	9.38	4.01	**26**	**0.151**
UDP DDIM 10	9.34	3.99	46	0.265
UDP DDIM 25	9.30	3.98	110	0.627
UDP DDPM 50	**8.95**	**3.76**	371	2.117

A.2.3. Module Ablation

We showed that the SPL module and UWB-Diffusion Guidance improve pose estimation under ideal sensing conditions (TotalCapture). Tab. 6 demonstrates that both com-

Figure 5. Qualitative results on DanceDB. UDP often estimates fast motions more accurately than other methods.

ponents also improve accuracy under noisy sensor conditions. Notably, adapting the UWB guidance coefficient λ to the sensor noise level enables improved performance, as noisy distance measurements with strong guidance can induce wrong limb predictions.

A.2.4. Noise Schedules

Additionally, we evaluated various noise schedules, as diffusion models are often sensitive to the specific noise distribution of the data. We hypothesized that schedules adding noise more gradually (e.g., cosine or exponential) would be beneficial, given that error accumulation along the kinematic chain causes big joint errors with even minimal joint

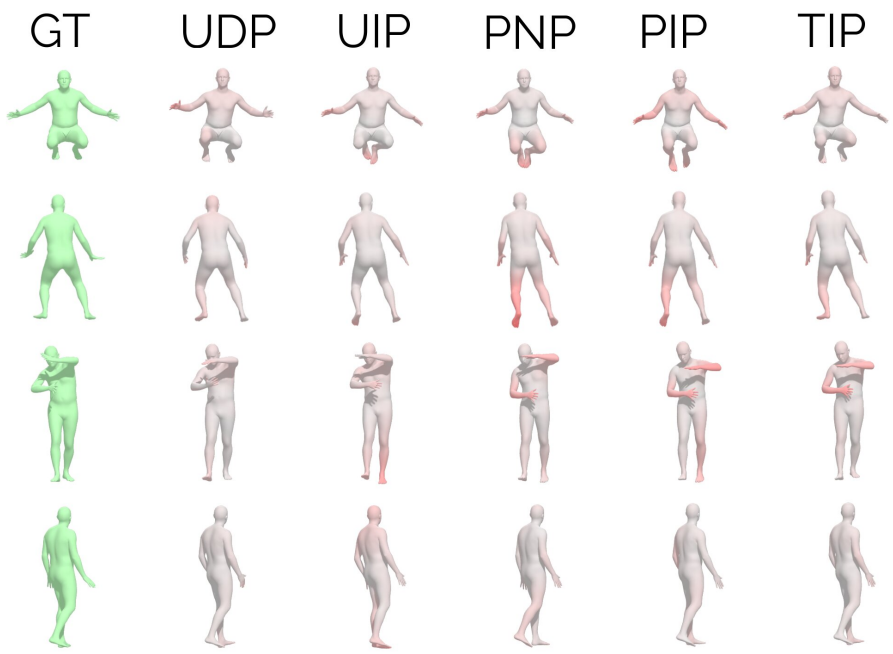

Figure 6. Qualitative results on TotalCapture. UDP often estimates limb positions more accurately than others.

Table 6. GIP-DB ablation on GIP-DB finetuned: SPL and UWB-Guidance improve metrics under real-world noise.

Metric	SIP (°) ↓	GAE (°) ↓	JPE (cm) ↓
UDP w/o SPL	16.03	15.09	7.08
UDP w/o UWB Guidance	16.30	14.86	6.90
UDP $\lambda = 500$	15.35	15.12	7.40
UDP	**15.33**	**14.34**	**6.68**

angle noise. As illustrated in Fig. 7, aggressive schedules like the square root schedule can rapidly degrade the pose structure, leading to visually distorted samples even at early noise steps. Counter-intuitively, the more aggressive square root schedule led to the best performance. As shown in Tab. 7, the square root schedule consistently outperforms all other tested schedules across all metrics.

Metric	SIP (°) ↓	GAE (°) ↓	JPE (cm) ↓
cosine	9.18	10.25	3.87
exponential	9.29	10.42	4.00
linear	9.46	10.48	4.09
sqrt	**8.95**	**10.19**	**3.76**

Table 7. UDP trained with different noise schedules.

A.2.5. Shaped Evaluation

UDP predicts SMPL joint angles and root translation, while the SMPL shape parameters β serve as optional conditioning input when available. Our main evaluation sets $\beta = \mathbf{0}$ (mean shape) for the model input and evaluation. β can be user-provided or estimated from UWB distances (as in UMotion [20]). We show that utilizing the known β reduces UDP's errors (Tab. 9), improving robustness across diverse body proportions. When using ground-truth shape parameters both as model input and for evaluation, UDP achieves consistent gains across all metrics, most notably a 10% reduction in SIP error. Since shape influences both inter-sensor distances and inertial dynamics, providing β further enhances motion estimation accuracy.

A.2.6. No Lookahead

We evaluate the model's performance in an online setting by restricting the evaluation to the last frame of the input window (Last Frame Only - LFO). In this scenario, the model relies solely on past observations without access to future context. As shown in Tab. 10, removing future context results in slightly higher errors, i.e., approximately 2.7% in SIP, while the global angle error remains largely unaffected. Additionally, reducing the context window from 3s to 1s increases JPE by 3%, with a marginal impact on angle errors. Crucially, even in this restricted online setting with no lookahead, UDP continues to outperform baseline methods.

Figure 7. Noise schedules of different schedules visualized on $q(x_t|x_0)$ in equal steps where the maximum step is $T = 50$.

Table 8. Ablation Study results on TotalCapture.

Metric	SIP Error (°) ↓	GAE (°) ↓	JPE (cm) ↓	TE @ 2m ↓	TE @5m ↓	Jitter (0.465)
UDP w/ Transformer	10.70	12.10	4.79	0.35	0.55	0.115
UDP w/ velocity representation	9.30	10.35	3.95	0.28	0.49	0.123
UDP w/ FK loss	9.01	10.29	3.89	**0.20**	**0.32**	0.124
UDP (ours)	**8.95**	**10.19**	**3.76**	**0.20**	0.33	0.124

Table 9. Evaluation DanceDB. Shape improves motion estimation.

Model	SIP (°) ↓	GAE (°) ↓	JPE (cm) ↓
UDP w/ known shape	**10.68**	**9.35**	**4.30**
UDP w/o known shape	11.79	9.91	4.67

LFO	W length	SIP (°) ↓	GAE (°) ↓	JPE (cm) ↓
✓	1 s	9.36	10.45	4.14
	1 s	9.15	10.49	3.84
✓	3 s	9.19	**10.11**	4.01
	3 s	**8.95**	10.19	**3.76**

Table 10. Ablation on lookahead and window size. 'LFO' denotes evaluating only the last frame of the window.

A.2.7. Removal of FK loss

Unlike previous methods [33, 49], we do not utilize a forward kinematics loss during training. Using the FK loss increases the training time from approximately 5h to 14h. Furthermore, as shown in Tab. 8, the forward kinematics loss does not compromise performance; in fact, it yields a marginal reduction in pose estimation accuracy. We attribute this to unstable gradients arising from the kinematic chain: despite similar loss values, the FK loss produces up to 10× higher gradient norms and 1000× larger peak gradients with respect to SMPL joint angles compared to a direct L1 loss. This indicates that direct supervision of local joint

angles is sufficient.

A.2.8. Model Architecture

Previous works on pose estimation [17, 19, 33] and pose generation [30] have employed transformers. However, our experiments show that a transformer encoder [34] performs significantly worse as a denoising model, with a 27% increase in joint position error, see Tab. 8. We implemented a 4-layer transformer encoder with the same token dimension as UDP, using sinusoidal position encodings. Overall, our results indicate that the local neighborhood priors captured by LSTMs are more effective than those learned by transformers, given the dataset size used in our experiments.

A.3. Translation Results

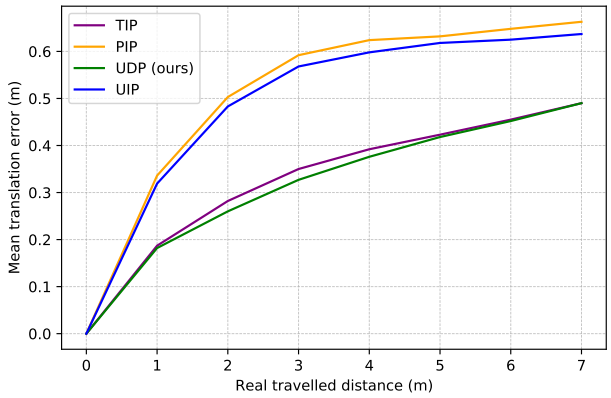

Figure 8. The cumulative translation error over segment of d meters on DanceDB. Purely learned methods UDP and TIP, tend to perform better on complicated dance moves.

UDP's primary contribution lies in improving the pose prediction by modeling geometric constraints using inter-sensor measurements. As demonstrated in the main paper, the Spatial Layout Module and UWB-Diffusion Guidance significantly improve local pose estimation, particularly for arm and leg placement. In this section, we describe the modeling of translation prediction and the evaluation of UDP's translation prediction.

Translation Representation. Unlike previous methods [4, 19, 33, 45, 46] that model translation implicitly by predicting frame-to-frame velocity, UDP outputs absolute translation predictions for individual window, each originating at the origin. To form a continuous, long-term sequence, these windows are concatenated such that each window begins at the final position of the preceding one. Velocity-based approaches require numerical integration to obtain global positions, a process where small inaccuracies accumulate over time. By representing translation as an absolute position directly, we aim to mitigate this integration drift.

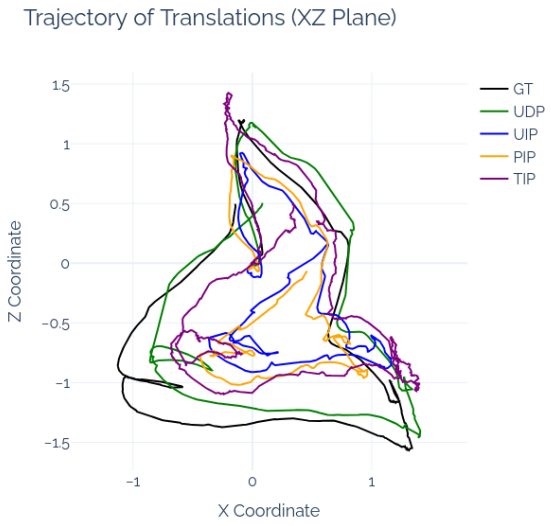

Figure 9. An example root trajectory in xz-plane. PIP and UIP tend to underestimate translation. UDP follows the trajectory closer. TIP has a higher level of jitter.

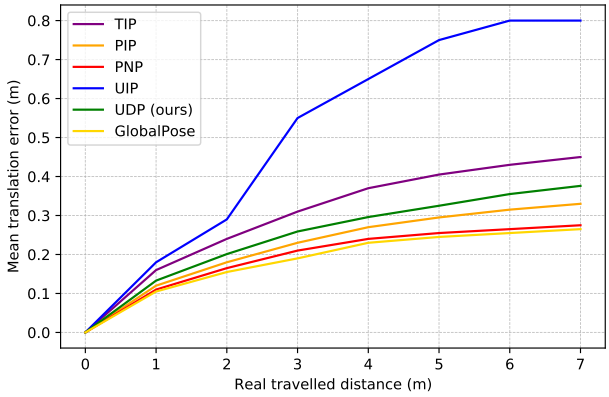

Figure 10. The cumulative translation error over segment of d meters on TotalCaptures. All models, but UIP, perform similarly in translation quality.

Translation Evaluation. We investigate translation prediction performance using the established cumulative translation error, which measures the drift from the ground truth trajectory after the subject has traveled a distance of d m. We report results on DanceDB and TotalCapture for all methods for which metrics are reported or could be reproduced directly. Overall, UDP achieves translation accuracy that is competitive with existing approaches across all datasets. On DanceDB, it outperforms the baseline methods, see Fig. 8. In contrast, on TotalCapture, methods that incorporate a physics optimizer, i.e., PIP, PNP, and GlobalPose, achieve lower drift, outperforming UDP by approxi-

mately 5 cm over 5 m (see Fig. 10).

We investigate the varying results qualitatively. On DanceDB, UDP generally yields more dynamic and accurate translations, whereas PIP and UIP tend to underestimate displacement, especially during twirls or intentional foot sliding. This is visible in Fig. 9, where UDP follows the ground truth trajectory closely.

On TotalCapture, UDP exhibits drift when the translation direction is estimated incorrectly. While the magnitude of fast, large-scale movements is tracked well, small directional errors can occasionally accumulate into larger drift.

We attribute these findings to two primary factors. First, UDP significantly improves foot and leg placement. Since human translation is intrinsically linked to gait, improved estimation of stride length and foot positioning reduces translation error, as observed on DanceDB. Second, physics optimized baselines benefit from explicit contact and dynamics constraints, which aid in tracking translation, as seen on TotalCapture. However, strict contact priors, such as zero-velocity assumptions, may degrade performance when the motion violates these constraints, such as during the intentional sliding maneuvers in DanceDB.

Overall, UDP demonstrates strong performance in local pose estimation due to the incorporation of on-body inter-sensor constraints, though it yields mixed results regarding global translation. Future work could investigate how pairing physics optimization with UDP, or the incorporation of external UWB anchors, can further improve translation accuracy.

A.3.1. Ablation: Absolute Translation Prediction

Existing methods such as UIP, PIP, TIP, and DiffusionPoser predict between-frame translation, i.e., velocity. In contrast, we employ an absolute translation representation, normalized to the origin. This representation improves not only the translation results but also marginally enhances local pose estimation. Experimentally, switching to a velocity-based representation increases translation drift by 16 cm over a 5 m distance. As shown in Tab. 8, UDP yields superior performance in both local pose and translation when predicting positions directly, highlighting the benefit of this approach.