DeFeeNet: Consecutive 3D Human Motion Prediction with Deviation Feedback ——Supplementary Material——

Xiaoning Sun[1], Huaijiang Sun[1]✉, Bin Li[2], Dong Wei[1], Weiqing Li[1], Jianfeng Lu[1]
[1]Nanjing University of Science and Technology, China
[2]Tianjin AiForward Science and Technology Co., Ltd., China
{sunxiaoning,sunhuaijiang}@njust.edu.cn, libin@aiforward.com

1. Details of Loss Function

During our two-round training phase, $\mathcal{L}_r$ should be specifically adapted with slight changes for fair comparison with original baselines [1, 5, 7].

On Human3.6M [2], the prediction loss of LTD-DeFee for round r $(r = 1, 2)$ is expressed as:

$$\mathcal{L}_{r(\text{LTD-DeFee})} = \frac{1}{J(N+T)} \sum_{t=1}^{N+T} \sum_{j=1}^{J} \|\hat{\mathbf{p}}_{r,(t,j)} - \mathbf{p}_{r,(t,j)}\|_2, \tag{1}$$

where $\hat{\mathbf{p}}_{r,(t,j)} \in \mathbb{R}^3$ denotes the predicted jth joint position of frame t in round r, and $\mathbf{p}_{r,(t,j)}$ the corresponding ground truth, with J the number of human skeletal joints. We also, like [5], predict on both observed part and target part, and sum L_2 errors on the $N + T$ region.

For STS-DeFee, the prediction loss for round r is:

$$\mathcal{L}_{r(\text{STS-DeFee})} = \frac{1}{J \times T} \sum_{t=1}^{T} \sum_{j=1}^{J} \|\hat{\mathbf{p}}_{r,(t,j)} - \mathbf{p}_{r,(t,j)}\|_2, \tag{2}$$

where $\hat{\mathbf{p}}_{r,(t,j)}$ and $\mathbf{p}_{r,(t,j)}$ share the same notation meanings as in Eq. 1. Prediction and error calculation are only on the T region like [7].

The per-round prediction loss of MotMix-DeFee is formulated as:

$$\mathcal{L}_{r(\text{MotMix-DeFee})} = \frac{1}{J \times T} \sum_{t=1}^{T} \sum_{j=1}^{J} \|v(\hat{\mathbf{p}}_{r,(t,j)}) - v(\mathbf{p}_{r,(t,j)})\|_2, \tag{3}$$

where $\hat{\mathbf{p}}_{r,(t,j)}$ and $\mathbf{p}_{r,(t,j)}$ have the same notation meanings as in Eq. 1. $v(\cdot)$ denotes the joint position displacement between two adjacent frames, as [1] predicts future displacement rather than position, i.e., velocity prediction strategy.

On BABEL [6], we refer to [4] to calculate MSE between the predicted pose parameter vector of $\mathbb{R}^K$ and the corresponding ground truth. The per-round prediction loss at round r $(r = 1, 2)$ for LTD-DeFee, STS-DeFee and MotMix-DeFee are

$$\mathcal{L}_{r(\text{LTD-DeFee})} = \frac{1}{N+T} \sum_{t=1}^{N+T} \|\hat{\mathbf{y}}_{r,t} - \mathbf{y}_{r,t}\|_2^2, \tag{4}$$

$$\mathcal{L}_{r(\text{STS-DeFee})} = \frac{1}{T} \sum_{t=1}^{T} \|\hat{\mathbf{y}}_{r,t} - \mathbf{y}_{r,t}\|_2^2, \tag{5}$$

and

$$\mathcal{L}_{r(\text{MotMix-DeFee})} = \frac{1}{T} \sum_{t=1}^{T} \|v(\hat{\mathbf{y}}_{r,t}) - v(\mathbf{y}_{r,t})\|_2^2, \tag{6}$$

with $\hat{\mathbf{y}}_{r,t} \in \mathbb{R}^K$ as the prediction at frame t at round r, and $\mathbf{y}_{r,t}$ the corresponding ground truth.

2. Implementation Details

		Human3.6M	BABEL
LTD-DeFee	dev_in	[batch_size,9,66]	[batch_size,9,60]
	dev_out	[batch_size,66,256]	[batch_size,60,256]
STS-DeFee	dev_in	[batch_size,3,9,22]	[batch_size,3,9,20]
	dev_out	[batch_size,3,10,22]	[batch_size,3,10,20]
MotMix-DeFee	dev_in	[batch_size,9,66]	[batch_size,9,60]
	dev_out	[batch_size,10,50]	[batch_size,10,50]

Table 1. Detailed input/output feature size of DeFeeNet on different baselines and different datasets.

In Table 1, we provide the detailed feature size of DeFeeNet when inserted into different baselines and on different datasets. *dev_in* and *dev_out* denote the prediction deviation (i.e., the input of DeFeeNet) and the latent deviation representation (i.e., the output of DeFeeNet). Specifically, as we set $T = 10$ in our experiments, the temporal dimension of the velocity-based deviation is 9. The spatial dimension is 66 (or [3, :, 22]) on Human3.6M and 60 (or [3, :, 20]) on BABEL. Our MLP-based DeFeeNet only increases parameter numbers of the original [1, 5, 7] by 12.2%, 5.2%

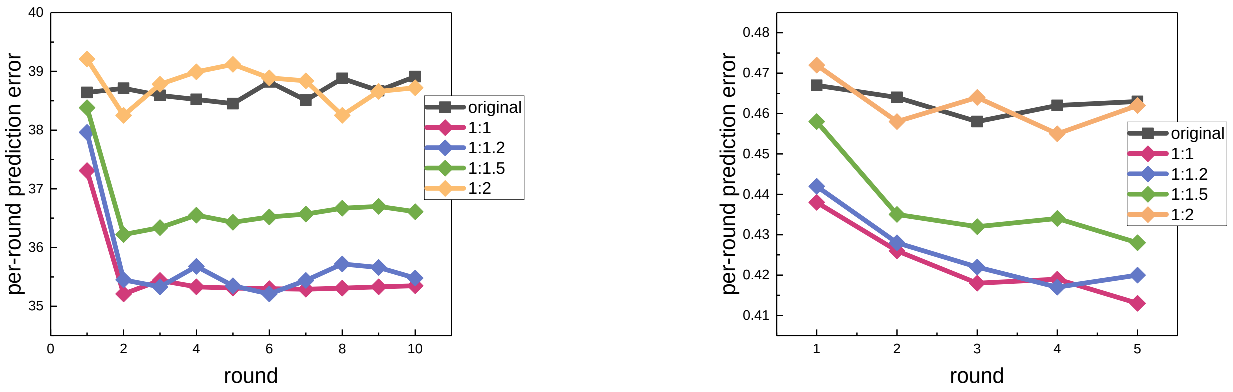

Figure 1. Influence of weight changing on $\mathcal{L}_{\text{total}}$. Dark grey *original* in both figure represent directly using baselines to implement multi-round prediction. $1 : x$ represent the weight ratio between λ_1 and λ_2. When $\lambda_1 : \lambda_2 = 1 : 1$, our multi-round prediction stably produce lower errors than baselines. Left: Baseline LTD-GCN added with DeFee(MLP) on Human3.6M. Per-round prediction error is calculated by averaging the errors of predicted frame 2, 4, 8, 10. Right: Baseline STS-GCN added with DeFee(GRU) on BABEL w/ transi, with per-round prediction error calculated by averaging the errors on frame 3, 6, 8, 10.

and 11.8%, respectively; our GRU-based version increase only about 0.29M parameters for three baselines.

Running Time Analysis. On RTX 1080Ti, for 10-round prediction on Human3.6M test set, LTD-GCN costs 94.93s, with MLP-based DeFeeNet 103.33s, and with GRU-based DeFeeNet 101.63s; for 5-round prediction on BABEL test set, STS-GCN costs 4.548s, with MLP-based DeFeeNet 4.943s, and with GRU-based DeFeeNet 4.886s. Our framework could compensate a small amount of computation time (7%∼9%) for prediction accuracy improvement.

Hyperparameter Setting. For the two-round training, we set both the hyperparameter in

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 \tag{7}$$

to 1, as the prediction accuracy of both rounds are of equal importance. Here we analyze the influence of weight changing by adjusting the relative weight of λ_1 and λ_2.

From Figure 1, our consecutive motion prediction stably yield lower errors than baselines when $\lambda_1 : \lambda_2 = 1 : 1$. When we put heavier weight on $\mathcal{L}_2$ (during training) in the attempt to increase the importance of deviation feedback learning, the multi-round testing performance, on the contrary, get worser results. In fact, to keep the accuracy of round 1 prediction is crucial. For one thing, the low-quality prediction in round 1 brings about low-quality deviation feedback between adjacent rounds, which may harm the following deviation-aware prediction round by round; for another, the heavy weight on $\mathcal{L}_2$ forces the baseline part more dependent on DeFeeNet (i.e., losing their original ability to predict solely), and therefore when DeFeeNet is nonactivated in round 1, the prediction error becomes even higher than the sole baseline, which is not our intention.

3. Experimental Details

3.1. BABEL Dataset Preprocessing

Based on the preprocessed BABEL [6] dataset file provided in [4] with 20 action categories, we remove categories with small sequence number and leave 11 actions (for example, action *hop* in [4] with only 58 sequences is removed in our task). (a) For isolated motion prediction task, we randomly cut two "observe then predict" units from each sequence in each action, as training/testing samples. (b) For consecutive motion prediction, we randomly cut two *groups* of two-round training/testing samples from each sequence in each action. To validate the improvement stability of our DeFeeNet, we randomly cut one group of five-round structure from each sequence in each action.

	original			two-round			five-round		
	w/o transi	transi	w/transi	w/o transi	transi	w/transi	w/o transi	transi	w/transi
train	17370	3544	20914	8485	1772	10257	-	-	-
test	6172	1212	7384	3029	606	3635	1376	606	1982

Table 2. Detailed dataset sample numbers on BABEL after our preprocessing. To train *original* baselines on BABEL, we arrange samples as current isolated-unit structure. To realize consecutive motion prediction, we need *two-round* sample structure for deviation learning and effectiveness validation. *Five-round* testing samples are for stability validation. Note that num(w/ transi) = num(w/o transi) + num(transi).

When choosing group numbers mentioned above, we fully consider two issues: (i) Some sequences are too short to cover the two-round structure; (ii) The random seeds are required to ensure that samples cut from sequences should cover their transition periods if there exist any. As shown in Table 2, the numbers of sequences in *original* are about twice as large as numbers in *two-round*. In other words, the amount of data used to train the original unit-based baselines and the two-round baseline-DeFee structures is almost the same, which allows for fair comparison.

3.2. More Experiments on AMASS and 3DPW

We additionally provide experimental comparisons between LTD-GCN and ours on AMASS [3] and 3DPW [8]. From Table 3, our DeFeeNet is also effective on these two datasets.

	AMASS		3DPW	
LTD-GCN [5]	27.70		29.78	
	-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)
r1	27.58	27.77	29.61	29.88
r2	**25.86**	**26.11**	**27.52**	**27.90**
r3	**26.02**	**25.74**	**27.85**	**27.48**
r4	**25.46**	**25.61**	**27.76**	**27.59**
r5	**25.88**	**25.90**	**27.62**	**28.12**

Table 3. Comparisons between LTD-GCN w/o and w/ DeFeeNet inserted. Results are average prediction errors at frame 2, 4, 8, 10. DeFeeNet is abbreviated as *D*. Bold values indicate lower errors.

3.3. Velocity-Based Vs. Position-Based Deviation Representation

To evaluate the superiority of our velocity-based prediction deviation, we change our $\mathbf{D}^{(v)}_{r-1} = v(\mathbf{x}_{r,N-T:N}) - v(\hat{\mathbf{y}}_{r-1,1:T}) \in \mathbb{R}^{K\times(T-1)}$ into the position-based representation:

$$\mathbf{D}^{(p)}_{r-1} = \mathbf{x}_{r,N-T:N} - \hat{\mathbf{y}}_{r-1,1:T} \in \mathbb{R}^{K\times T}, \tag{8}$$

and feed this representation into DeFeeNet to analyze the corresponding consecutive prediction performance (shown in Table 4 and 5).

	LTD-GCN [5]		STS-GCN [7]		MotionMixer [1]	
isolated	38.64		41.16		35.52	
	-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)
velocity-based	**35.52**	**35.98**	**38.63**	**39.06**	**32.49**	**32.87**
position-based	36.48	36.21	39.12	39.88	33.86	33.73

Table 4. Comparisons of average prediction errors at round 2 to 10 on Human3.6M when baseline-DeFee are fed with velocity-based and position-based deviation (per-round error is calculated by averaging the errors on frame 2, 4, 8, 10).

From the tables, both velocity-based and position-based representations are valid for deviation feedback, but velocity-based version produce lower errors. Compared to position-based representation that considers the general appearance of human poses, our velocity-based deviation allows for more focus on motion status and the body parts that are prone to error, which benefits DeFeeNet that may confront multiple factors of prediction deviation such as prediction not in place or action status changing.

	LTD-GCN [5]		STS-GCN [7]		MotionMixer [1]	
isolated	0.4221		0.4673		0.3925	
	-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)
velocity-based	**0.3879**	**0.3869**	**0.4247**	**0.4195**	**0.3560**	**0.3591**
position-based	0.3897	0.3888	0.4301	0.4226	0.3610	0.3622

Table 5. Comparisons of average prediction errors at round 2 to 5 on BABEL (w/ transi) when baseline-DeFee are fed with velocity-based and position-based deviation (per-round error is calculated by averaging the errors on frame 3, 6, 8, 10).

3.4. Ablation about Deviation Feedback

To further demonstrate the effectiveness of deviation feedback, we provide performance comparison between deviation feedback enabled and disabled in Table 6, where prediction errors cannot be obviously reduced when deviation is disabled. The corresponding visualized comparison is also given in Figure 2.

	round	dev ✘	dev		round	dev ✘	dev
LTD-DeFee on H3.6M	r1	37.31	37.31	STS-DeFee on BABEL	r1	0.4385	0.4385
	r2	37.27	**35.21**				
	r3	37.23	**35.44**		r2	0.4383	**0.4265**
	r4	37.19	**35.33**				
	r5	37.15	**35.31**		r3	0.4382	**0.4189**
	r6	37.14	**35.30**				
	r7	37.13	**35.29**		r4	0.4380	**0.4192**
	r8	37.16	**35.31**				
	r9	37.18	**35.33**		r5	0.4381	**0.4133**
	r10	37.22	**35.35**				

Table 6. When deviation feedback is disabled (✘), prediction errors cannot be obviously reduced.

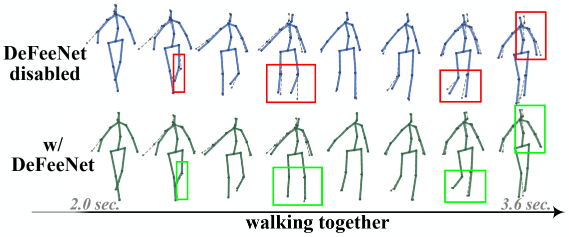

Figure 2. Visualized comparison between DeFeeNet disabled during testing vs. DeFeeNet enabled. Errors marked in red boxes are alleviated when DeFeeNet enabled (marked in green).

3.5. Passing GT as Deviation into DeFeeNet

To ensure that the baseline completely mirrors the DeFeeNet-inserted architecture, we pass GT observation as deviation to DeFeeNet. Results in Table 7 indicate that such operation has no obvious improvement on prediction accuracy, and that is why we are motivated to use past deviation. Compared to GT observation, the past deviation could reflect the mistake that the model has made before, i.e., containing error information that just happened, where our DeFeeNet can learn to derive certain "patterns" from it and then constrain the model to make better predictions.

LTD-GCN [5] on H3.6M		STS-GCN [7] on BABEL w/transi	
38.64		0.4673	
passing GT observation into DeFeeNet			
-D(MLP)	-D(GRU)	-D(MLP)	-D(GRU)
38.30	38.45	0.4629	0.4648

Table 7. Passing GT observation as deviation into DeFeeNet.

3.6. Stretching the Length of Observation

Since the accuracy improvement produced by our deviation-aware prediction is *essentially* due to the introduction of additional information outside the current unit (i.e., deviation feedback from the previous adjacent one), there might be concerns whether the same effect could be achieved if we simply stretch the observed length to introduce more information for prediction (see Figure 3).

We conduct an ablation study as follows. We stretch the *observed* length to $N+T$, which is of the same length as the prediction round in our consecutive motion prediction task, and construct each sample with $N+T$ poses observed and T poses to be predicted. We re-train baselines using such $(N+2T)$-pose sequence samples (isolated unit-based), and test them round by round. Note that, during testing phase, each round of prediction is produced given *only* the corresponding observation with no deviation involved.

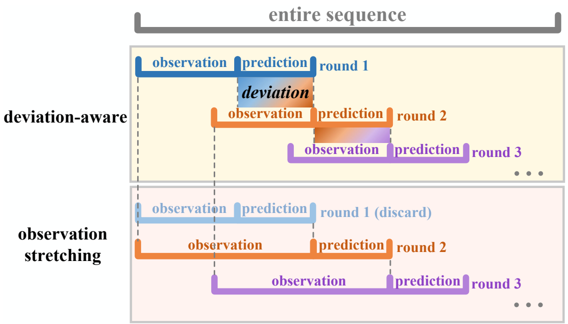

Figure 3. Frameworks of deviation-aware prediction and observation stretching.

From Table 8, the re-trained baselines do not enjoy accuracy advantage. The reason lies in that blindly extending the observation length may not bring significant improvement in the prediction performance. Instead of solely, evenly exploit global information from historical window, with the help of our DeFeeNet, the existing predictor could focus more on local information (i.e., newly detected deviation), and therefore more sensitive to prediction parts that are prone to error, such as action status changing period that *just* began to happen. That is also why we set the prediction length to 10 but not longer, as too long the predicted sequence is not conducive to generate the deviation feedback that is sufficiently effective yet easy to capture.

	Human3.6M					BABEL w/ transi			
frame num.	2	4	8	10	frame num.	3	6	8	10
LTD-GCN (obs=10)	12.69	26.06	52.28	63.53	STS-GCN (obs=10)	0.24	0.43	0.55	0.64
LTD-D(MLP) (obs=10)	**10.38**	**22.68**	**48.29**	**59.92**	STS-D(GRU) (obs=10)	**0.23**	**0.39**	**0.49**	**0.58**
LTD-GCN (obs=20)	12.21	26.24	52.79	64.02	STS-GCN (obs=20)	0.23	0.44	0.56	0.65

Table 8. Comparisons of prediction errors between original baselines [5, 7], Baseline-DeFee and baselines re-trained with stretching observation length on Human3.6M and BABEL w/ transi. Values in bold indicate lower errors. For Baseline-DeFee marked in purple background color, errors in Human3.6M is the average of round 2 to 10 prediction errors, while in BABEL the average of round 2 to 5 prediction errors.

3.7. More Visualized Results

More visualizations are in Figure 4 and 5 (next page).

References

[1] Arij Bouazizi, Adrian Holzbock, Ulrich Kressel, Klaus Dietmayer, and Vasileios Belagiannis. Motionmixer: Mlp-based 3d human body pose forecasting. *arXiv preprint arXiv:2207.00499*, 2022. 1, 3

[2] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. 1

[3] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *ICCV*, pages 5442–5451, 2019. 3

[4] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. Weakly-supervised action transition learning for stochastic human motion prediction. In *CVPR*, pages 8151–8160, 2022. 1, 2

[5] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *ICCV*, pages 9489–9497, 2019. 1, 3, 4

[6] Abhinanda R Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J Black. Babel: Bodies, action and behavior with english labels. In *CVPR*, pages 722–731, 2021. 1, 2

[7] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. Space-time-separable graph convolutional network for pose forecasting. In *ICCV*, pages 11209–11218, 2021. 1, 3, 4

[8] Timo Von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, pages 601–617, 2018. 3

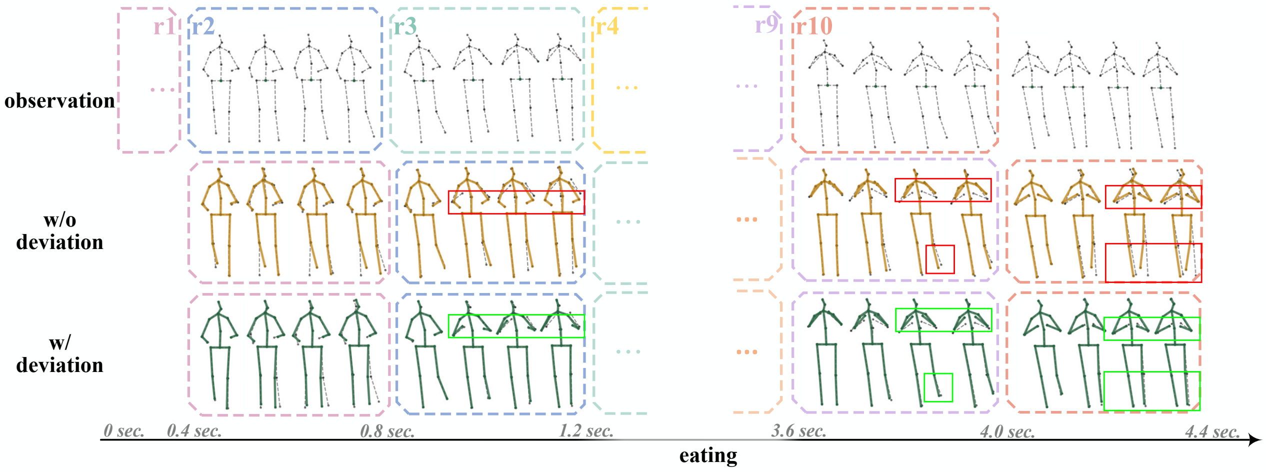

Figure 4. Top: Visualization of a sample *eating* in Human3.6M. Dashed lines indicate observation/GT. Prediction errors highlighted in red boxes are alleviated by our deviation-aware prediction (highlighted in green boxes).

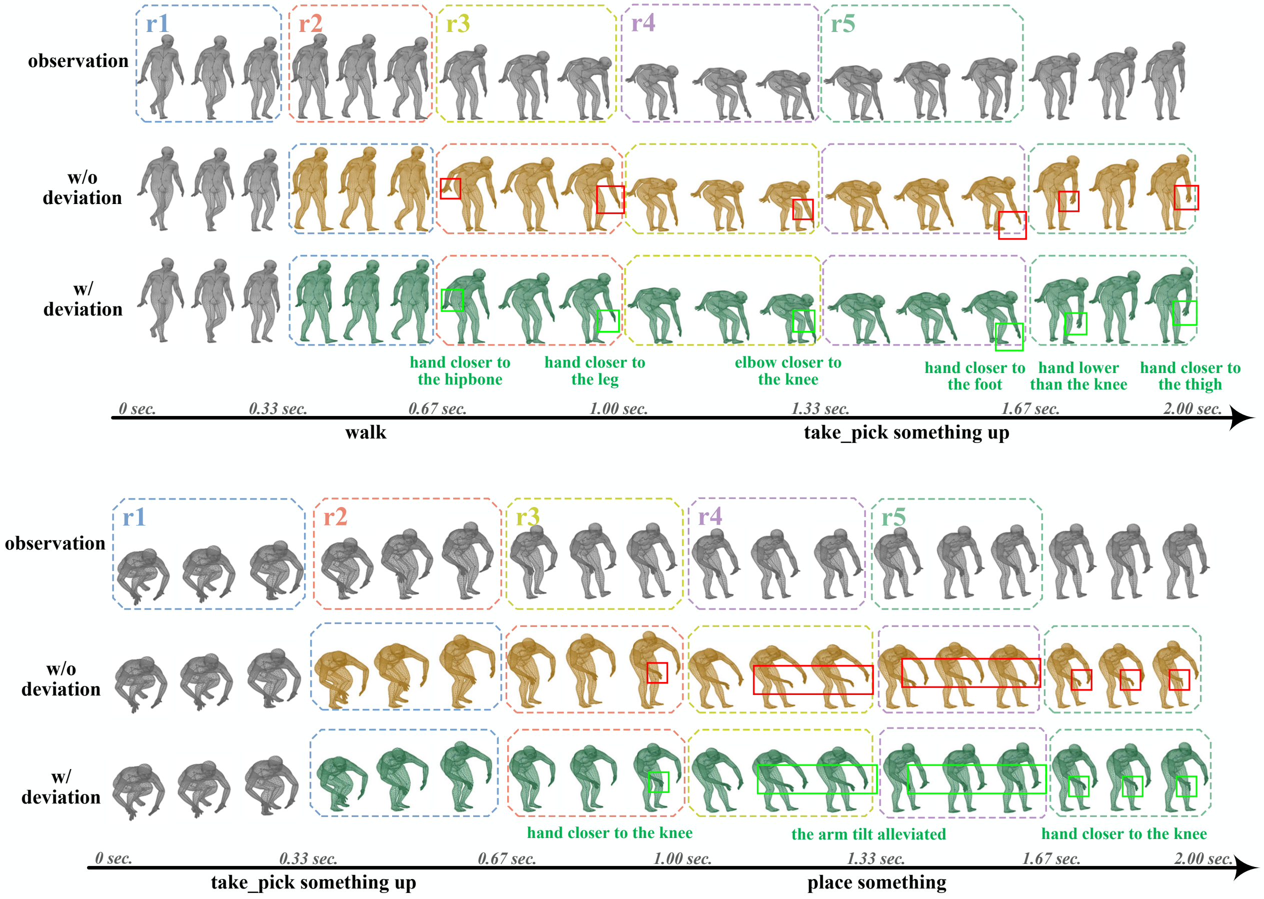

Figure 5. Top: Visualization of a sample from *walk* to *take_pick something up* in BABEL. Bottom: Visualization of a sample from *take_pick something up* to *place something* in BABEL. Grey poses represent observation/GT. Prediction errors highlighted in red boxes are alleviated by our deviation-aware prediction (highlighted in green boxes).