Privacy-centric Deep Motion Retargeting for Anonymization of Skeleton-Based Motion Visualization (Supplementary Materials)

S1. Implementation Details

Our model was implemented in PyTorch and experiments were conducted on a server with two NVIDIA GeForce RTX 3090's and an AMD Threadripper 3960x CPU (64 GB RAM). We utilize the Adam optimizer for the auto-encoder portion, the embedding classifiers, and the discriminator. The PMR models boasts an inference time is 0.006s for 75 frames ($\sim$2.5s) of motion video. The training complexity is $\sim$ $1.5\times$ that of the DMR model due to the adversarial/cooperative learning components. The final model took 6.5 hours to train.

S1.1. Model Architecture

Table S1 shows our PyTorch implementation of the encoders, decoder, embedding classifiers, and the quality controller, where the acronyms are explained in Table S2. The embeddings size is (256, 32).

S1.2. Training Stages

Table S3 details the epochs and stages of training. Initial pre-training primes the models briefly. The first two stages utilize paired and unpaired data for embedding separation. The next two stages ready the embedding classifiers for iterative cooperative and adversarial training. In the unpaired stage, the model is learning how motion data works and learns how to break down and reconstruct the skeletons. Most of the emphasis is put on the paired training, where the motion retargeting is fine-tuned.

S1.3. DMR Baseline Implementation

Since the original DMR was designed for animation datasets (e.g., Mixamo), we adapted it for real-world capture data:

- Re-implemented the character-agnostic variant following official code
- Added loss components from the skeleton-aware variant to handle NTU's noisy real-capture data
- Used same data preprocessing pipeline as PMR for fair comparison
- Training hyperparameters: learning rate 0.001, batch size 64, 200 epochs

S2. Dataset Analysis

A major difference in our experiment versus those in standard motion retargeting is that our data is imperfect. Other datasets like Mixamo, as used in PMR, are great for motion retargeting training as all the actors complete actions in the same way. Additionally, the data is created and not captured, leaving much less noise. In a real-world dataset, such as NTU, actors complete the task how they would normally do them. Environments could also be different, some may be closer to the camera, some may complete the task faster or slower. Identifiable information is captured in just the way someone completes a task. The data has individualistic ways the tasks are completed, which makes it perfect for evaluating the privacy, but may have flaws in the retargeting. This becomes especially notable in the cross-reconstruction loss L_{cross}.

S3. Privacy-Utility Trade-off

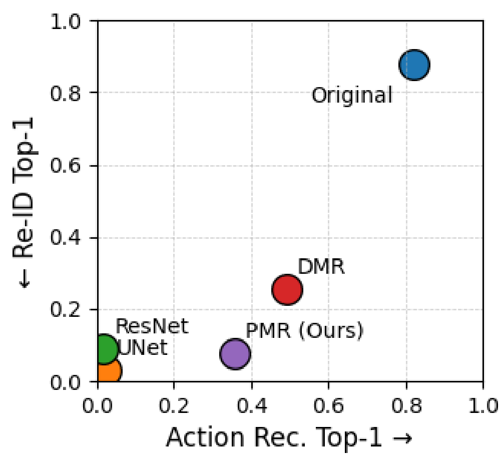


Figure S1. Privacy-utility trade-off on NTU-60. The plot shows re-identification accuracy (privacy risk) vs. action recognition accuracy (utility). PMR achieves the best balance, positioned closest to the lower-right corner (low privacy risk, high utility).

Name	Layers	k	s	p	in/out
Encoder	C2D + LR + MP + RP2D	3x3	1	-	75/12
	C2D + LR + MP + RP2D	3x3	1	-	12/24
	C2D + LR + MP + RP2D	3x3	1	-	24/32
	C2D + LR + MP + RP2D	3x3	1	-	32/256
Decoder	CT2D + LR + Up + RP2D	3x3	1	1	512/ 256
	CT2D + LR + Up + RP2D	3x3	1	1	256/128
	CT2D + LR + Up + RP2D	3x3	1	1	128/96
	CT2D + LR + Up + RP2D	3x3	1	1	96/75
Embedding Classifier	CT1D + BN + R	3	1	1	256/128
	CT1D + BN + R	3	1	1	128/256
	CT1D + BN + R + AP	3	1	1	256/512
	FL + LR + R	-	-	-	512/1024
	LR + R	-	-	-	1024/512
	LR + R + SM	-	-	-	512/Y
Quality Controller (Discriminator)	CT1D + LR + Up + RP1D	3	1	1	T/64
	CT1D + LR + Up + RP1D	3	1	1	64/32
	CT1D + LR + Up + RP1D	3	1	1	32/16
	CT1D + LR + Up + RP1D	3	1	1	16/8
	FL + LR + R	-	-	-	80/32
	LR + Sig	-	-	-	32/1

Table S1. Model Implementations. **k** represents kernel size, **s** represents stride, and **p** represents padding.

Acronym	**Definition**
C2D	Convolutional 2D
CT2D/1D	Convolutional Transpose 2D/1D
LR	Leaky ReLU
R	ReLU
Up	Upsample
MP/AP	Max/Average Pooling
RP/2D	Reflection Pad/2D
MLP	Multi-Layer Perceptron
BN	Batch Normalization
Y	Number of classes
FL	Flatten
SM	Softmax
Sig	Sigmoid

Table S2. Definitions of Acronyms Used

Stage	**Paired**	**Epochs**
Pre-training the Auto-Encoder	Yes	5
Pre-training the Auto-Encoder	No	20
Pre-training the Embedding Classifiers	Yes	20
Pre-training the Embedding Classifiers	No	50
Unpaired	No	100
Paired Training	Yes	100

Table S3. Training stages used for final model

S4. Task Specific Metrics

In many deployment scenarios, one may only need to preserve discriminative accuracy on *certain critical actions*, rather than all classes. For example, healthcare or assisted living might focus on fall detection. We therefore measure **per-class** F1 on a subset of medically or safety-relevant actions from the NTU-60 dataset:

- **Falling Down** (A43),
- **Staggering** (A42),
- **Chest Pain** (A45),
- **Headache** (A44).

Method	Class-level F1			
	Falling	Stagger	Chest Pain	Headache
Original	0.965	0.981	0.905	0.884
DMR	0.999	0.814	0.676	0.491
PMR (Ours)	0.974	0.981	0.905	0.884

Table S4. Per-class accuracy and F1 for selected medical-related actions on NTU-60. PMR and DMR are reasonably close on these key actions, while PMR drastically lowers re-ID (shown in the main paper).

Table S4 shows that both PMR and DMR degrade performance vs. the original skeleton (which is expected, since retargeting can alter nuance). Nevertheless, PMR still achieves fairly strong detection of these critical classes, at only a modest cost in F1 relative to DMR. Meanwhile, PMR's re-ID remains below 8–10%, whereas DMR's is $\sim 25\%$. In a safety-oriented application where privacy is paramount, PMR offers a compelling balance.

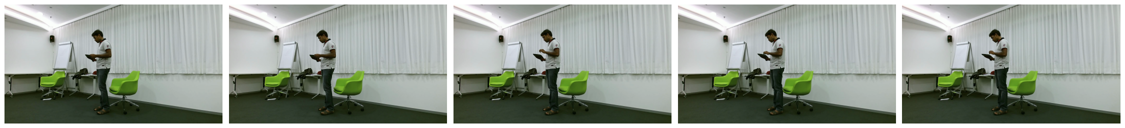
S5. Visualizations

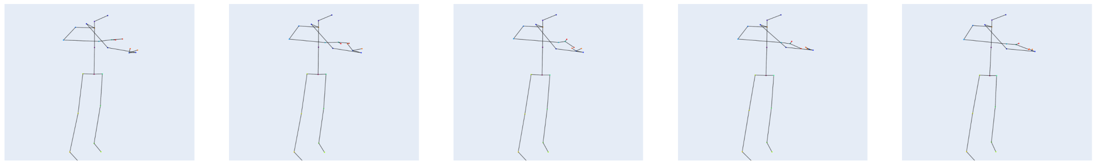
Figure S2 shows another example ("Type on a Keyboard") from NTU-60. Figure S3 shows the anonymization results for the "Apply Cream on Hand" action from NTU-120. The Moon paper was only trained on NTU60 and could not be fairly tested for the "Apply Cream on Hand" action. Additionally, provided in the GitHub are GIF files demonstrating the frame by frame movement. These GIFs are rendered in 30 fps to align with the capture rate from the NTU dataset.

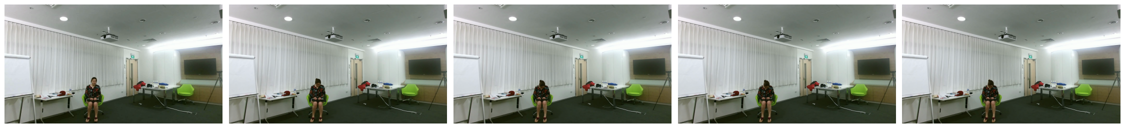
S6. Embedding Classifier Accuracy

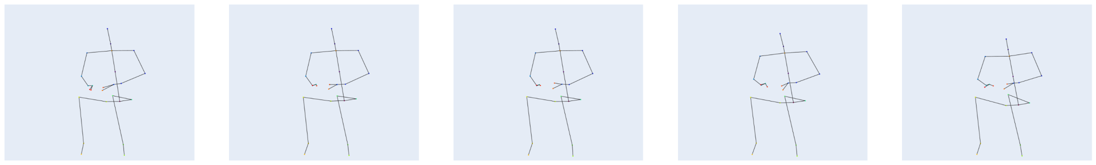
Figure S4 illustrates the privacy classifier's accuracy on training and validation on the primary training on the NTU60 dataset. Red and green lines represent accuracy on the privacy embedding, whereas blue and purple lines show accuracy on the motion embedding. The aim is low performance on the motion embedding, indicating minimal PII presence.

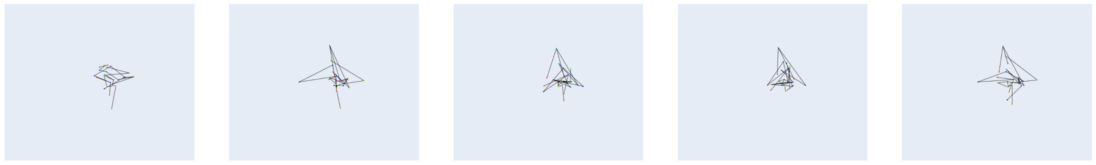
Figure S5 displays the utility classifier's accuracy in training and validation on the primary training on the NTU60 dataset. Here, red and green lines indicate accuracy on the motion embedding, focusing on the skeleton's utility. Blue and purple lines represent performance on the privacy embedding. The goal is high accuracy on the motion embedding and low on the privacy embedding, ensuring motion information exclusivity.

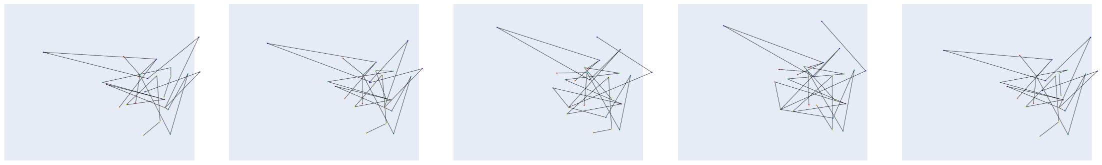

(a) Original RGB Video

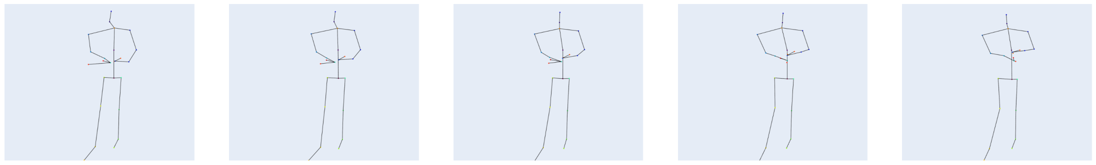

(b) Original


(c) Dummy


(d) Dummy


(e) UNet


(f) ResNet


(g) DMR

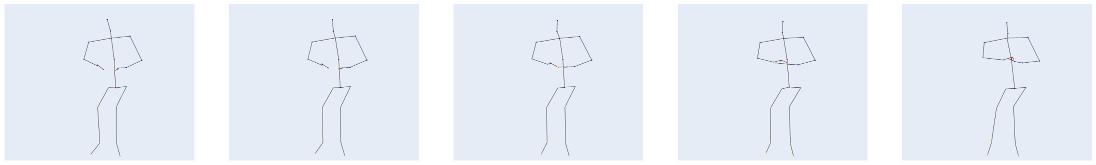
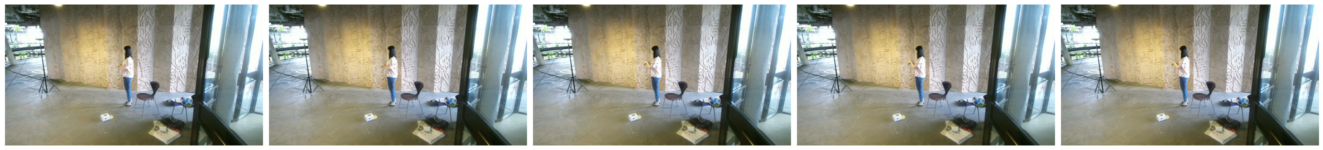
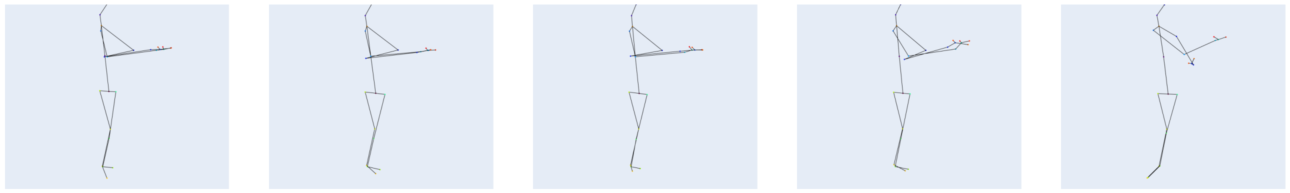
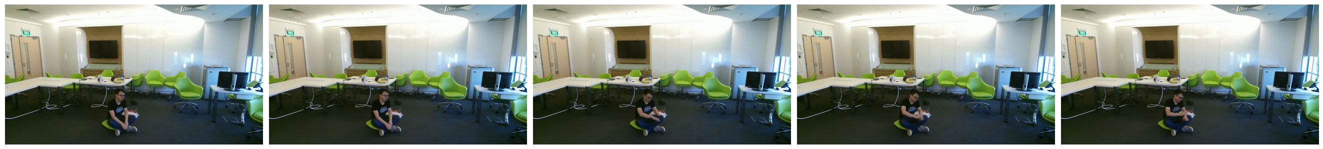
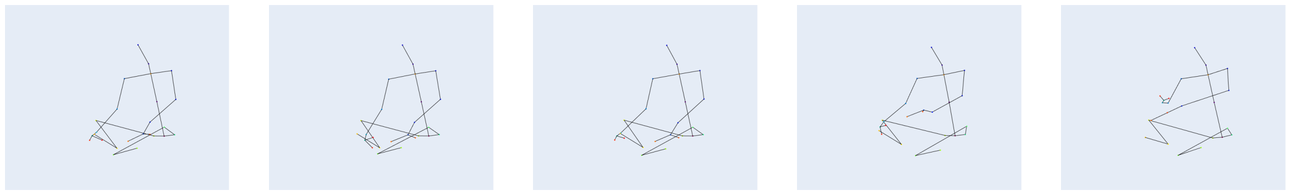

(h) PMR

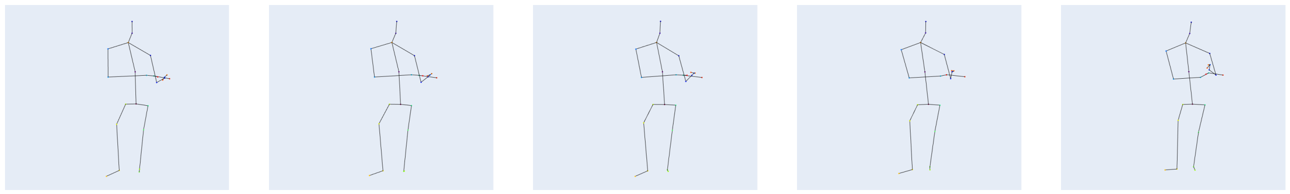
Figure S2. Example visualization: Actor 19 performing the ”Type on a Keyboard” action, cast to Actor 36. The PMR model attempts to make the generated skeleton sit, while maintaining end-effector positioning.


(a) Original RGB Video


(b) Original


(c) Dummy RGB Video


(d) Dummy


(e) DMR

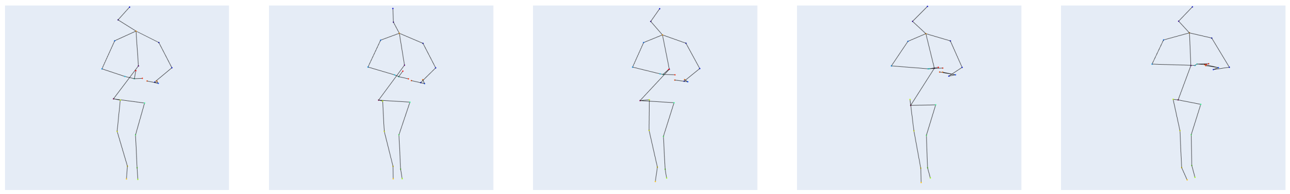

(f) PMR

Figure S3. Example visualization: Actor 61 performing the "Apply Cream on Hand" action, cast to Actor 8. PMR makes the generated skeleton sit and places arms similarly to that of the dummy, preserving the privacy of the original sequence.

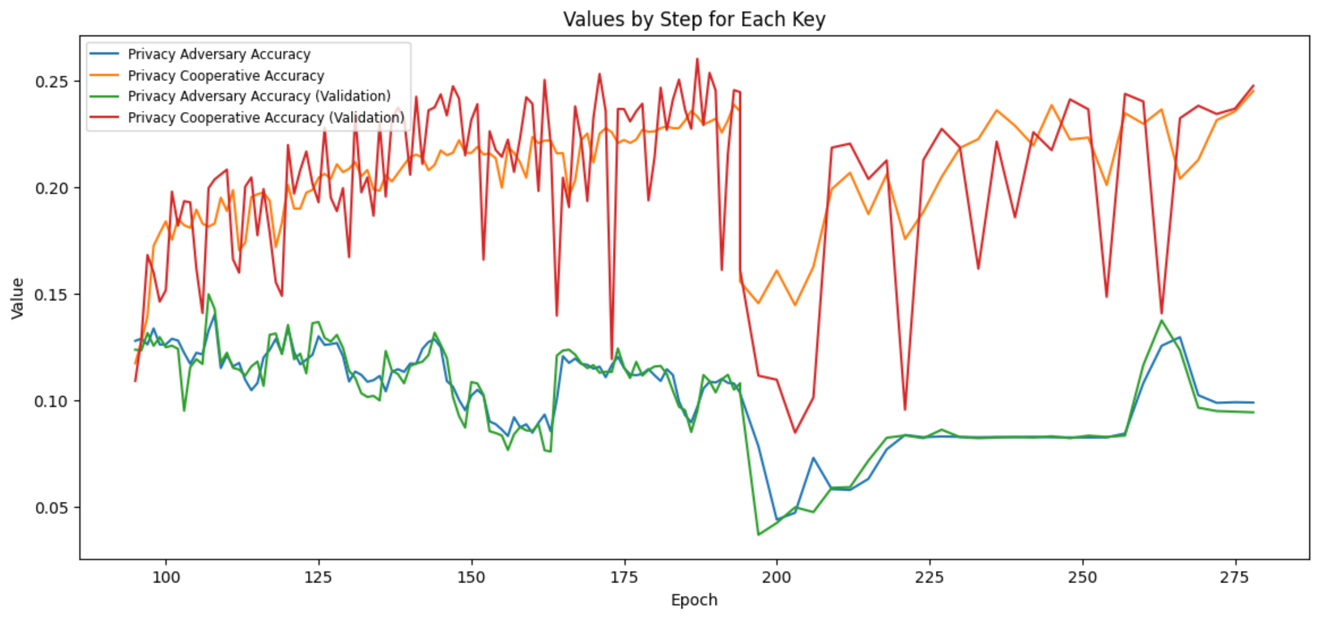


Figure S4. Privacy Classifier Accuracy

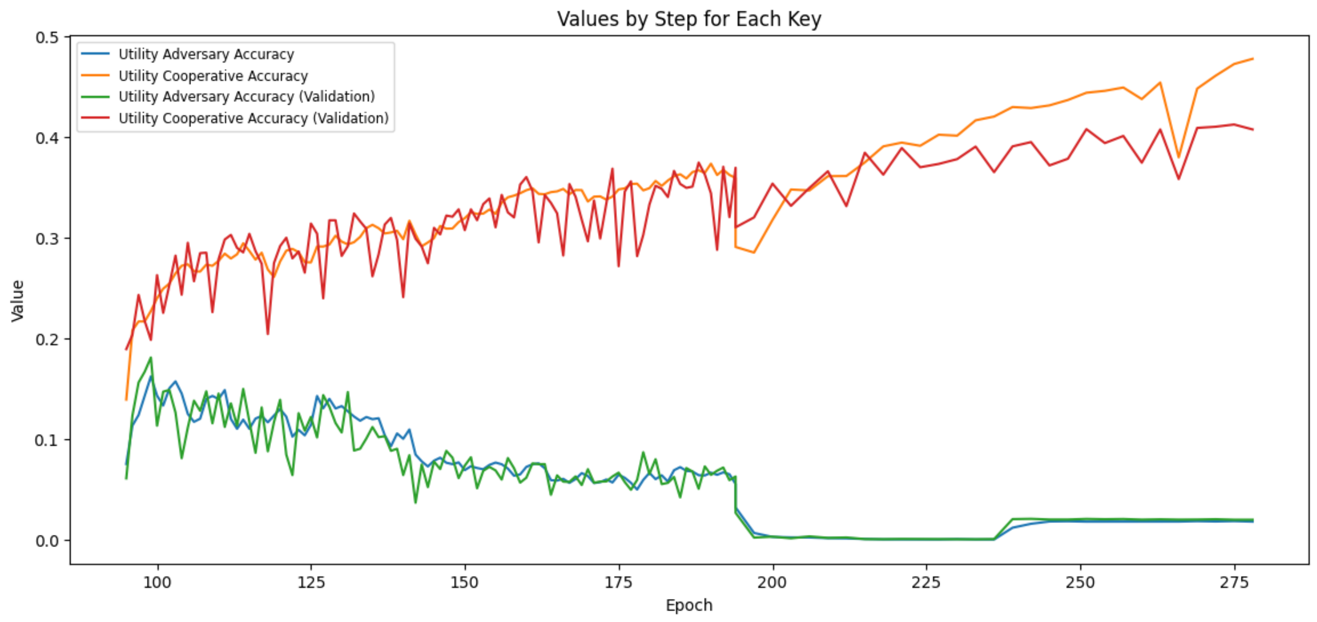


Figure S5. Utility Classifier Accuracy