ProtoRes: Proto-Residual Architecture for Deep Modeling of Human Pose

Boris N. Oreshkin
Unity Technologies, Labs
Montreal, QC, CANADA
boris.oreshkin@unity3d.com

Florent Bocquelet
Unity Technologies, Labs
Montreal, QC, CANADA
florent.bocquelet@unity3d.com

Félix G. Harvey
Unity Technologies, Labs
Montreal, QC, CANADA
felix.harvey@unity3d.com

Bay Raitt
Unity Technologies, Labs
Olympia, WA, US
bay.raitt@unity3d.com

Dominic Laflamme
Unity Technologies, Labs
Montreal, QC, CANADA
dom@unity3d.com

Abstract

Our work focuses on the development of a learnable neural representation of human pose for advanced AI assisted animation tooling. Specifically, we tackle the problem of constructing a full static human pose based on sparse and variable user inputs (*e.g.* locations and/or orientations of a subset of body joints). To solve this problem, we propose a novel neural architecture that combines residual connections with prototype encoding of a partially specified pose to create a new complete pose from the learned latent space. We show that our architecture outperforms a baseline based on Transformer, both in terms of accuracy and computational efficiency. Additionally, we develop a user interface to integrate our neural model in Unity, a real-time 3D development platform. Furthermore, we introduce two new datasets representing the static human pose modeling problem, based on high-quality human motion capture data, which will be released publicly along with model code.

1 Introduction

Modeling human pose and learning pose representations have received increasing attention recently due to their prominence in applications, including computer graphics and animation [15, 47]; pose and motion estimation from video [31, 36, 45]; immersive augmented reality [5, 12, 29, 49]; entertainment [30, 46]; sports and wellness [25, 39]; human machine interaction [8, 16, 41] and autonomous driving [27]. In the gaming industry, state-of-the-art real-time pose manipulation tools, such as CCD [23], FABRIK [2] or FinalIK [38], are popular for rapid execution and rely on forward and inverse kinematic models defined via non-learnable kinematic equations. *Inverse kinematics* (IK) is the process of defining the internal geometric parameters (e.g. local rotations) of a kinematic system that provides the desired configuration (e.g. global positions) for a subset of system's joints [34]. Moreover, *forward kinematics* (FK) refers to the use of the kinematic equations to compute the positions of joints from specified values of internal geometric parameters. While mathematically accurate, non-learnable kinematic models do not guarantee that the underconstrained solutions derived from sparse constraints (e.g. positions of a subset of joints) result in plausible human poses. In

Preprint. Under review.

contrast, more often than not sparse constraints give rise to perturbations of pose parameters that look unnatural even to the untrained eye. The main reason behind it is the lack of inductive bias to resolve an ill-posed problem of recovering the full pose from a small set of constraints.

In this paper we develop a neural modeling approach to human pose that bridges the gap between skeleton-aware human pose representation based on IK/FK ideas and the neural embedding of human pose. We pay special attention to modelling the semantics of joints and their interactions using a novel prototypical residual (ProtoRes) architecture that compactly represents a partially specified human pose and accurately reconstructs a plausible fully specified pose in a computationally efficient manner. Our approach learns the statistics of natural poses using datasets derived from high-quality motion capture (MOCAP) sequences. We show that in terms of the pose reconstruction accuracy, the proposed novel architecture outperforms existing gaming industry tools (FinalIK [38]) as well as out-of-the-box machine-learning solution based on Transformer [44], which also happens to be 6-7 times less effective in terms of training speed than the proposed architecture. Finally, we develop a set of user-facing tools integrating our learned neural pose representation in the Unity game engine to solve an important practical problem of authoring a human pose with minimal user inputs. We hope that our model and tools will help speed up the animation process and alleviate game artist animation skill requirements thus simplifying and democratizing the game development process.

1.1 Background

We consider the full-body pose authoring animation task depicted in Fig. 1. The animator provides a few inputs, which we call *effectors*, that the target pose has to respect. For example, in Fig. 1, the look-at effector specifies that the head should be facing the orange dot, the positional effectors constrain the right foot and the right hand to be pinned to the pink dots and the rotational effector, shown in cyan, constrains the world-space rotation of the pelvis. We assume that the animator can generate arbitrary number of such effectors placed on skeletal joints (one joint can be driven by more than one effector). The task of the model is to combine all the information provided via effectors and generate the full-body pose making sure that the pose looks plausible and the initial effector constraints are respected. We define the full-body pose as the set of all kinematic parameters necessary to recreate the appearance of the body in 3D.

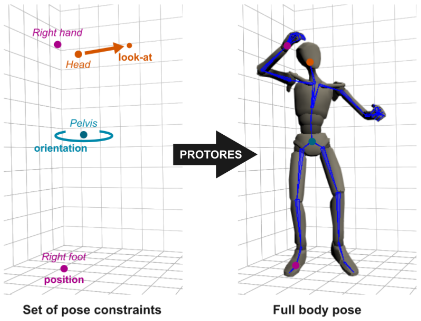

Figure 1: ProtoRes completes full human pose using a combination of 3D coordinates, look-at targets and world-space rotations specified by a user for an arbitrary subset of body joints.

Mathematically, we assume that each effector can be represented in the space $\mathbb{R}^{d_\text{eff}}$, where d_eff is taken to be maximum over all effector types. Suppose we have 3D position and 6D rotation effectors: d_eff is 6. In position effectors, 3 extra values are 0. We formulate the pose authoring problem as learning the mapping $\Upsilon_\theta : \mathbb{R}^{N \times d_\text{eff}} \rightarrow \mathbb{R}^{d_\text{kin}}$ with learnable parameters $\theta \in \Theta$. Υ_θ maps the input space $\mathbb{R}^{N \times d_\text{eff}}$ of variable effector dimensionality N (the number of effectors is not known in advance) to the space $\mathbb{R}^{d_\text{kin}}$, containing all kinematic parameters to reconstruct full-body pose. For example, a body with J joints can be fully defined using a tree model with 6D local rotation per joint and 3D coordinate for the root joint, in which case $d_\text{kin} = 6J + 3$, assuming fixed bone lengths. Given a sufficiently representative dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^M$ of poses containing (at a minimum) pairs of inputs $\mathbf{x}_i \in \mathbb{R}^{N \times d_\text{eff}}$ and outputs $\mathbf{y}_i \in \mathbb{R}^{d_\text{kin}}$ it is viable to define the empirical risk minimization problem to learn Υ_θ:

$$\Upsilon_\theta = \arg\min_{\theta \in \Theta} \frac{1}{M} \sum_{\mathbf{x}_i, \mathbf{y}_i \in \mathcal{D}} L(\Upsilon_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{1}$$

The rest of this section summarises our contributions and reviews literature. Section 2 describes our proposed approach. Section 3 presents empirical results and Section 4 concludes the paper.

1.2 Summary of Contributions

The contributions of our paper can be summarized as follows.

- The definition of the 3D character posing task in eq. (1) and two public datasets to solve it and support research in the direction of AI assisted character posing and animation
- A novel neural architecture embodying Υ_θ for representing and recovering a partially specified human pose based on an arbitrary set of heterogeneous effector inputs
- A novel loss function $L(\cdot)$ combining position, rotation and direction error terms via randomized weights based on randomly generated effector tolerance levels

1.3 Related Work

Joint representations. In principle, unconstrained 3D position joint representations can be used to specify a pose, which is an approach some works employ [4, 10, 24]. However, this approach is sub-optimal as it does not enforce fixed-length bones, nor specifies joint rotations. Respecting bone length constraints plays important role in modeling realistic human poses [35] and joint rotations are crucial in downstream applications, such as deforming a 3D mesh on top of the skeleton without unexpected twisting. Enforcing these constraints during optimization may complicate problem (1). A more practical solution is to predict joint rotations, making bones part of the model and automatically satisfying bone lengths [35]. This is viable via skeleton representations based on Euler angles [14], rotation matrices [17, 20, 50] and quaternions [35]. 6D representation of joint rotation based on the first two rows of the full 3x3 rotation matrix addresses the continuity issues reminiscent of quaternion and Euler based representations [51]. This is the angular representation we use in this work.

Pose modeling architectures. Multi-Layer Perceptrons (MLPs) [11, 24, 32] and kernel methods [13, 18] have been used to learn single pose representations. The former can be used for frame-based motion classification [11] or dimension reduction [32]. Beyond single pose, skeleton moving through time can be modeled as a spatio-temporal graph [22] or as a graph convolution [32, 48]. A common limitation of these approaches is their reliance on a fixed set of inputs, whereas our architecture is specifically designed to handle a variable number of inputs, allowing sparsely specified poses.

Pose prediction from sparse constraints. MOCAP systems and gamepad controllers produce intermittent and noisy data. A variety of approaches tackle the task of generating full-body poses from signals produced by low-fidelity sensors. Real-time methods based on nearest-neighbor search, local dynamics and motion matching have been used on sparse marker position and accelerometer data [3, 9, 37, 43]. MLPs and Recurrent Neural Networks (RNNs) have been used for real-time processing of sparse signals such as accelerometers [19, 21, 28, 42] and VR constraints [29, 49]. These approaches rely on the fixed set of inputs and past pose information to disambiguate next frame prediction and as such are not applicable to our problem. Our approach aims at finding the best pose for the current variable set of sparse constraints. Fast iterative IK algorithms for real-time applications such as CCD [23] or FABRIK [2] have been very popular for their flexibility and rapid execution, but suffer from limited realism when used for human full-body IK, as they are not data-driven. Learning-based methods strive to alleviate this by providing learned model of human pose. Grochow et al. [13] proposed a kernel based method for learning a pose latent space in order to produce the most likely pose satisfying sparse effector constraints via online constrained optimization. The more recent commercial tool Cascadeur uses a cascade of several MLPs (each dealing with fixed set of positional effectors: 6, 16, 28) to progressively produce all joint positions without respecting bone constraints [7, 24]. Unlike our approach, Cascadeur cannot handle arbitrary effector combinations, rotation or look-at constraints and requires heavy heuristic post processing to respect bone constraints.

2 ProtoRes

The proposed architecture, depicted in Fig. 2, follows the encoder-decoder pattern and produces predictions in three steps. First, the variable number and type of user supplied inputs are processed for translation invariance and embedded. Second, the architectural core, a proto-residual encoder, transforms the pose specified via effectors into a single vector (pose embedding). Finally, the pose decoder expands the pose embedding into the full-body pose representation including local rotation and global position of each joint. Next, we describe the proposed architecture in more detail.

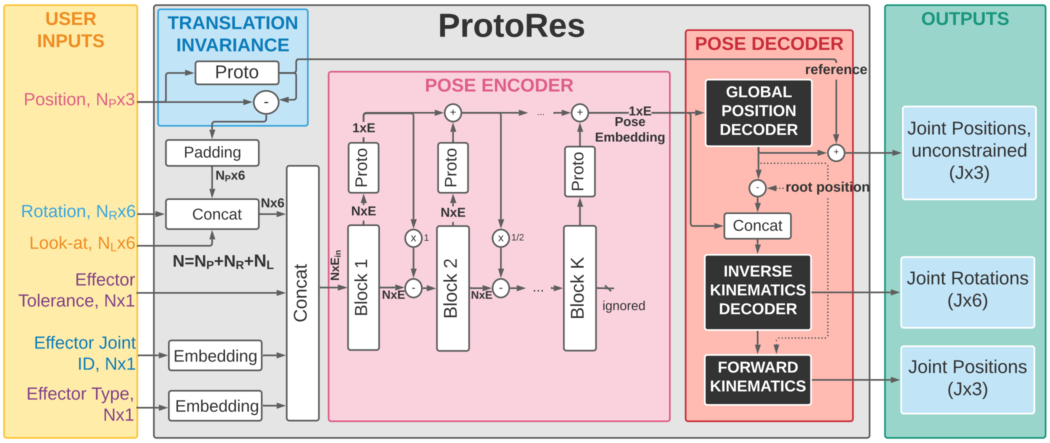

Figure 2: Block diagram of the ProtoRes architecture.

2.1 Architecture

User inputs. ProtoRes accepts position (3D coordinates), rotation (6D representation [51]) or look-at (3D target position and 3D local facing direction) effectors. For positional effectors, the remaining 3 out of total 6 dimensions are set to zeros. All positions are re-referenced relative to the centroid of the positional effectors to achieve translation invariance. Translation invariance simplifies the handling of poses in global space while not relying on a precise reference frame, which can be tricky to define for heterogeneous MOCAP sources. Each effector is further characterized by tolerance, joint ID and type. Tolerance is a positive value. Smaller tolerance implies that the effector has to be more strictly respected in the reconstructed pose. Joint ID is an integer in $[0, J)$ indicating which joint is affected by a given effector. Effector type is an integer in $[0, 2]$ indicating a positional (0), rotational (1) or look-at (2) effector. Type and joint ID variables are embedded into vectors and concatenated with effector data, resulting in the input to the encoder architecture, a matrix $\mathbf{x}_{in} \in \mathbb{R}^{N \times E_{in}}$ with E_{in} corresponding to the combined dimension of all embeddings plus 7D effector data and tolerance.

Pose Encoder is a two-loop residual network. The first residual loop is implemented inside each block (depicted in Fig. 4 (left), Appendix A). The second residual loop (see Fig. 2) is formed by forward link prototypes. The residual links are used to (i) improve gradient flow and (ii) achieve the interaction between the encodings of individual joints and the encoding of the entire pose created at each encoder block. We assume the encoder input to be $\mathbf{x}_1 = \mathbf{x}_{in} \in \mathbb{R}^{N \times E_{in}}$. We skip the batch dimension for brevity, in which case the fully-connected layer $\text{FC}_{r,\ell}$, with $\ell = 1...L$, in the residual block r, $r = 1 \dots R$, with weights $\mathbf{W}_{r,\ell}$ and biases $\mathbf{a}_{r,\ell}$ can be conveniently described as $\text{FC}_{r,\ell}(\mathbf{h}_{r,\ell-1}) \equiv \text{RELU}(\mathbf{W}_{r,\ell}\mathbf{h}_{r,\ell-1} + \mathbf{a}_{r,\ell})$. The prototype layer is defined as the mean over the leading dimension of $\mathbf{x}$, $\text{PROTOTYPE}(\mathbf{x}) \equiv \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}[i,:]$, resulting in the following pose encoder equations:

$$\begin{aligned} \mathbf{x}_r &= \text{RELU}[\mathbf{b}_{r-1} - 1/(r-1) \cdot \mathbf{p}_{r-1}], \\ \mathbf{h}_{r,1} &= \text{FC}_{r,1}[\mathbf{x}_r], \ \dots, \ \mathbf{h}_{r,L} = \text{FC}_{r,L}[\mathbf{h}_{r,L-1}], \\ \mathbf{b}_r &= \text{RELU}[\mathbf{L}_r \mathbf{x}_r + \mathbf{h}_{r,L}], \ \mathbf{f}_r = \mathbf{F}_r \mathbf{h}_{r,L}, \\ \mathbf{p}_r &= \mathbf{p}_{r-1} + \text{PROTOTYPE}[\mathbf{f}_r]. \end{aligned} \tag{2}$$

The second and third equations implement the MLP and the first residual loop. The zest is added to the architecture in the first and fourth equations. The fourth equation collapses the forward encoding of individual effectors into the representation of the entire pose via prototyping. The representation is accumulated across blocks to form the final pose representation at the end of the encoder. The intermediate representations are used in the first equation to implement the interaction between the representations of the individual effectors captured by the residual link, $\mathbf{b}_{r-1}$, and the global prototype representation of the pose, $\mathbf{p}_{r-1}$, based on the information passed from the previous blocks. The $1/(r-1)$ factor aligns the scales of $\mathbf{b}_{r-1}$ and $\mathbf{p}_{r-1}$. Without the first equation, each effector would be processed separately, with no conditioning on other effectors, leading to sub-optimal results.

Pose Decoder has two blocks: global position decoder (GPD) and inverse kinematics decoder (IKD). Both rely on the fully-connected residual (FCR) architecture depicted in Fig. 4 of Appendix A. GPD unrolls the pose embedding generated by encoder into the unconstrained predictions of 3D joint positions. IKD generates the internal geometric parameters (joint rotations) of the skeleton that are guaranteed to generate feasible joint positions after forward kinematics pass.

GPD accepts the encoded pose embedding, $\widetilde{\mathbf{b}}_0 = \mathbf{p}_R \in \mathbb{R}^E$, and produces 3D position predictions $\widetilde{\mathbf{f}}_R \in \mathbb{R}^{3J}$ of all skeletal joints using the FCR whose r-th ($r = 1 \dots R$) block is described as follows:

$$\begin{aligned} \mathbf{h}_{r,1} &= \mathrm{FC}^{gpd}_{r,1}[\widetilde{\mathbf{b}}_{r-1}], \; \dots, \; \mathbf{h}_{r,L} = \mathrm{FC}^{gpd}_{r,L}[\mathbf{h}_{r,L-1}], \\ \widetilde{\mathbf{b}}_r &= \mathrm{ReLU}[\mathbf{L}^{gpd}_r \widetilde{\mathbf{b}}_{r-1} + \mathbf{h}_{r,L}], \; \widetilde{\mathbf{f}}_r = \widetilde{\mathbf{f}}_{r-1} + \mathbf{F}^{gpd}_r \mathbf{h}_{r,L}. \end{aligned} \tag{3}$$

Since GPD produces predictions with no regard to skeleton constraints, its predictions do not respect bone lengths. For the IKD to provide correct rotations, the origin of the kinematic chain in world space must be given, and GPD conveniently provides the prediction of the reference (root) joint. Moreover, the draft pose it generates, albeit physically infeasible, is used to guide and condition IKD.

IKD accepts the concatenation of the encoder-generated pose embedding, $\mathbf{p}_R \in \mathbb{R}^E$, and the output of GPD, $\widetilde{\mathbf{f}}_R \in \mathbb{R}^{3J}$, forming the input to the IKD, $\widehat{\mathbf{b}}_0 \in \mathbb{R}^{E+3J}$. IKD predicts the 6DoF angle for each joint, $\widehat{\mathbf{f}}_R \in \mathbb{R}^{6J}$, and its r-th ($r = 1 \dots R$) block operates as follows:

$$\begin{aligned} \mathbf{h}_{r,1} &= \mathrm{FC}^{ikd}_{r,1}[\widehat{\mathbf{b}}_{r-1}], \; \dots, \; \mathbf{h}_{r,L} = \mathrm{FC}^{ikd}_{r,L}[\mathbf{h}_{r,L-1}], \\ \widehat{\mathbf{b}}_r &= \mathrm{ReLU}[\mathbf{L}^{ikd}_r \widehat{\mathbf{b}}_{r-1} + \mathbf{h}_{r,L}], \; \widehat{\mathbf{f}}_r = \widehat{\mathbf{f}}_{r-1} + \mathbf{F}^{ikd}_r \mathbf{h}_{r,L}. \end{aligned} \tag{4}$$

Forward Kinematics (FK) pass is applied to the output of the IKD, transforming local joint rotations and global root position into the global joint rotations and positions using skeleton kinematic equations. The FK pass relies on the offset vector $\mathbf{o}_j = [o_{x,j}, o_{y,j}, o_{z,j}]^\intercal$ and the rotation matrix $\mathbf{R}_j$ for each joint j. The offset vector is a fixed non-learnable vector representing bone length constraint for joint j. It provides the displacement of this joint with respect to its parent joint when joint j rotation is zero. Rotation matrix $\mathbf{R}_j$ is constructed from IKD's prediction $\widehat{\mathbf{f}}_{R,j}$ using robust representation [51] relying on vector norm $\overrightarrow{\mathbf{u}} \equiv \mathbf{u}/\|\mathbf{u}\|_2$ and vector cross product $\mathbf{u} \times \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\gamma) \overrightarrow{\mathbf{n}}$ (γ is the angle between $\mathbf{u}$ and $\mathbf{v}$ in the plane containing them and $\overrightarrow{\mathbf{n}}$ is the normal to the plane):

$$\widehat{\mathbf{r}}_{j,x} = \overrightarrow{\widehat{\mathbf{f}}_{R,j}[1:3]}, \quad \widehat{\mathbf{r}}_{j,z} = \overrightarrow{\widehat{\mathbf{r}}_{j,x} \times \widehat{\mathbf{f}}_{R,j}[4:6]}, \quad \widehat{\mathbf{r}}_{j,y} = \widehat{\mathbf{r}}_{j,z} \times \widehat{\mathbf{r}}_{j,x}, \quad \widehat{\mathbf{R}}_j = [\widehat{\mathbf{r}}_{j,x} \, \widehat{\mathbf{r}}_{j,y} \, \widehat{\mathbf{r}}_{j,z}]. \tag{5}$$

Provided with the local offset vectors and rotation matrices of all joints, the global rigid transform of any joint j is predicted following the tree recursion from the parent joint $p(j)$ of joint j:

$$\widehat{\mathbf{G}}_j = \widehat{\mathbf{G}}_{p(j)} \begin{bmatrix} \widehat{\mathbf{R}}_j & \mathbf{o}_j \\ \mathbf{0} & 1 \end{bmatrix}. \tag{6}$$

The global transform matrix $\widehat{\mathbf{G}}_j$ of joint j contains its global rotation matrix, $\widehat{\mathbf{G}}^{13}_j \equiv \widehat{\mathbf{G}}_j[1:3, 1:3]$, and its 3D global position, $\widehat{\mathbf{g}}_j = \widehat{\mathbf{G}}_j[1:3, 4]$.

2.2 Losses

We use three losses to train the architecture in a multi-task fashion. The total loss combines loss terms additively with weights chosen to equalize their magnitude orders.

L2 loss penalizes the mean squared error between the prediction $\widehat{\mathbf{y}}$ and the ground truth $\mathbf{y}$:

$$\mathrm{MSE}(\mathbf{y}, \widehat{\mathbf{y}}) = \|\mathbf{y} - \widehat{\mathbf{y}}\|_2^2. \tag{7}$$

L2 loss is used to supervise the GPD as well as the IKD output after FK pass. In the latter case it drives IKD to learn to predict rotations that lead to small position errors after FK.

Geodesic loss penalizes the errors of the IKD's rotational outputs. It represents the smallest arc (in radians) to go from one rotation to another over the surface of a sphere. The geodesic loss is defined for the ground truth rotation matrix $\mathbf{R}$ and its prediction $\widehat{\mathbf{R}}$ as (see e.g. [40]):

$$\mathrm{GEO}(\mathbf{R}, \widehat{\mathbf{R}}) = \arccos \left[(\mathrm{tr}(\widehat{\mathbf{R}}^T \mathbf{R}) - 1)/2 \right]. \tag{8}$$

We believe that using a combination of L2 and Geodesic losses is necessary to learn a high-quality pose representation. This particularly appeals intuition when the task is to reconstruct a sparsely

Algorithm 1 Loss calculation for a single item in the training batch of ProtoRes.

```
Require: $\mathbf{R}_j, \mathbf{G}_j; N \sim \textsc{Uniform}[3,16]$   ▷ Ground truth for all joints $j \in [0, J)$; number of effectors
Ensure: $\mathbf{x}$   ▷ Sample inputs
  $I_1, \ldots, I_N \leftarrow \textsc{Multinomial}(\{0, \ldots, J-1\}, N)$   ▷ Effector IDs
  $T_1, \ldots, T_N \leftarrow \textsc{Multinomial}(\{0, 1, 2\}, N)$   ▷ Effector type
  for $i$ in $1 \ldots N$ do
    $\Lambda_i \leftarrow \textsc{Uniform}[0,1]$   ▷ Effector tolerance
    $\sigma(\Lambda_i);\ W(\Lambda_i) \leftarrow \sigma_M \Lambda_i^{\eta};\ \min(W_M, 1/\sigma(\Lambda_i))$   ▷ Effector noise std and weight
    $\mathbf{x}[i,:] \leftarrow \textsc{NoiseModel}(\mathbf{G}_{I_i}, \sigma(\Lambda_i), T_i)$   ▷ Generate noisy effector
  end for
Predict: $\widetilde{\mathbf{f}}_{R,j}, \widehat{\mathbf{R}}_j, \widehat{\mathbf{G}}_j \quad \forall j$
  $\mathcal{L}^{rnd}_{gpd-L2} \leftarrow \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{T_i=0} W(\Lambda_i)} \sum_{i=1}^{N} \mathbb{1}_{T_i=0} W(\Lambda_i) \operatorname{MSE}(\mathbf{g}_{I_i}, \widetilde{\mathbf{f}}_{R,I_i})$   ▷ Randomized GPD position loss
  $\mathcal{L}^{rnd}_{ikd-L2} \leftarrow \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{T_i=0} W(\Lambda_i)} \sum_{i=1}^{N} \mathbb{1}_{T_i=0} W(\Lambda_i) \operatorname{MSE}(\mathbf{g}_{I_i}, \widehat{\mathbf{g}}_{I_i})$   ▷ Randomized IKD position loss
  $\mathcal{L}^{det}_{gpd-L2} \leftarrow \sum_{j=1}^{J} \operatorname{MSE}(\mathbf{g}_j, \widetilde{\mathbf{f}}_{R,j})$   ▷ Deterministic GPD position loss
  $\mathcal{L}^{det}_{ikd-L2} \leftarrow \sum_{j=1}^{J} \operatorname{MSE}(\mathbf{g}_j, \widehat{\mathbf{g}}_j)$   ▷ Deterministic IKD position loss
  $\mathcal{L}^{det}_{loc-geo} \leftarrow \sum_{j=1}^{J} \operatorname{GEO}(\mathbf{R}_j, \widehat{\mathbf{R}}_j)$   ▷ Deterministic local rotation loss
  $\mathcal{L}^{rnd}_{glob-geo} \leftarrow \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{T_i=1} W(\Lambda_i)} \sum_{i=1}^{N} \mathbb{1}_{T_i=1} W(\Lambda_i) \operatorname{GEO}(\mathbf{G}^{13}_{I_i}, \widehat{\mathbf{G}}^{13}_{I_i})$   ▷ Randomized global rotation loss
  $\mathcal{L}^{det}_{lat} \leftarrow \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{T_i=2}} \sum_{i=1}^{N} \mathbb{1}_{T_i=2} \operatorname{LAT}(\mathbf{x}[i, 1:3], \mathbf{x}[i, 4:6], \widehat{\mathbf{G}}^{13}_{I_i})$   ▷ Randomized Look-at loss
  $\mathcal{L} \leftarrow \frac{W_{pos}}{J}(\mathcal{L}^{rnd}_{gpd-L2} + \mathcal{L}^{rnd}_{ikd-L2} + \mathcal{L}^{det}_{gpd-L2} + \mathcal{L}^{det}_{ikd-L2}) + \frac{1}{J}(\mathcal{L}^{det}_{lat} + \mathcal{L}^{rnd}_{glob-geo} + \mathcal{L}^{det}_{loc-geo})$   ▷ Total loss
```

specified pose, giving rise to multiple plausible reconstructions. We argue that a model trained to reconstruct both plausible joint positions and rotations is better equipped to solve the task accurately. Empirical evidence presented in Section 3.4 supports this intuition: a model trained on both L2 and Geodesic generalizes better on both losses than models trained only on one of those terms.

Look-at loss, proposed in this paper, enables the "look-at" feature, *i.e.* the ability to rotate a joint to face a particular global position in a scene (for example, having the head looking at a given object). It is generic in that it allows the model to align any direction vector $\mathbf{d}_j \in \mathbb{R}^3$ of a joint, expressed in its local frame of reference, towards a global target location $\mathbf{t}$. Given the predicted global transform matrix $\widehat{\mathbf{G}}_j$ of this joint, the look-at loss is defined as follows:

$$\operatorname{LAT}(\mathbf{t}, \mathbf{d}_j, \widehat{\mathbf{G}}_j) = \arccos\left[\overrightarrow{(\mathbf{t} - \widehat{\mathbf{g}}_j)} \cdot \widehat{\mathbf{G}}^{13}_j \mathbf{d}_j\right]. \tag{9}$$

$\overrightarrow{(\mathbf{t} - \widehat{\mathbf{g}}_j)}$ is a vector pointing at the target object in world space. $\widehat{\mathbf{G}}^{13}_j$, when multiplied by $\mathbf{d}_j$, represents the global predicted look-at direction. The look-at loss trains the IKD to produce $\widehat{\mathbf{G}}^{13}_j$ consistent with the look-at direction defined by $\mathbf{t}$ and $\mathbf{d}_j$, both provided as network inputs.

2.3 Training Methodology

The training methodology involves techniques targeting to (i) regularize model via data augmentation, (ii) learn handling of sparse inputs and (iii) effectively combine multi-task loss terms.

Data augmentation is based on the rotation and mirror augmentations. The former rotates the skeleton around the vertical Y axis by a random angle in $[0, 2\pi]$. Rotation w.r.t. ground XZ plane is not applied to avoid creating poses implausible according to the gravity direction. Mirror augmentation removes any implicit left- or right-handedness biases by flipping the skeleton w.r.t. the YZ plane.

Sparse inputs modeling relies on effector sampling. First, the total number of effectors is sampled uniformly at random in the range [3, 16]. Given the total number of effectors, the effector IDs (one of 64 joints) and types (one of 3 types: position, rotation, or look-at) are sampled from the Multinomial without replacement. This sampling scheme produces an exponentially large number of different permutations of effector types and joints, resulting in strong regularizing effects.

Table 1: Key quantitative results: ProtoRes vs. baselines. Lower values are better.

	miniMixamo			miniAnonymous		
	$\mathcal{L}^{det}_{gpd-L2}$	$\mathcal{L}^{det}_{ikd-L2}$	$\mathcal{L}^{det}_{loc-geo}$	$\mathcal{L}^{det}_{gpd-L2}$	$\mathcal{L}^{det}_{ikd-L2}$	$\mathcal{L}^{det}_{loc-geo}$
5-point benchmark						
FinalIK [38]	5.53e-3	8.54e-3	0.5287	3.76e-3	7.83e-3	0.5164
Masked-FCR	1.29e-3	2.38e-3	0.2596	1.11e-3	2.38e-3	0.2124
Transformer	1.24e-3	2.32e-3	0.2703	0.99e-3	2.05e-3	0.2112
ProtoRes	**1.00e-3**	**2.07e-3**	**0.2529**	**0.74e-3**	**1.71e-3**	**0.2019**
Random benchmark						
Masked-FCR	1.73e-2	3.85e-2	0.3157	1.57e-2	3.23e-2	0.2694
Transformer	1.67e-3	4.75e-3	0.2564	1.53e-3	4.18e-3	0.1992
ProtoRes	**1.38e-3**	**4.25e-3**	**0.2390**	**0.91e-3**	**3.19e-3**	**0.1810**

Effector tolerance and randomized loss weighting. For each sampled effector, we further uniformly sample $\Lambda \in [0, 1]$ treated as effector tolerance. Given an effector tolerance Λ, noise (noise models used for different effector types are described in detail in Appendix B) with variance proportional to Λ is added to effector data before feeding them to the neural network:

$$\sigma(\Lambda) = \sigma_M \Lambda^{\eta}. \tag{10}$$

We use $\eta > 10$ to shape the distribution of σ to smaller values. Furthermore, to each effector is attached a randomized loss weight reciprocal to $\sigma(\Lambda)$, capped at W_M if $\sigma(\Lambda) < 1/W_M$:

$$W(\Lambda) = \min(W_M, 1/\sigma(\Lambda)). \tag{11}$$

Λ drives network inputs and is simultaneously used to weigh losses by $W(\Lambda)$. Thus ProtoRes learns to respect effector tolerance, leading to two positive outcomes. First, ProtoRes provides a tool allowing one to emphasize small tolerance effectors ($\Lambda \approx 0$) and relax the large tolerance ones ($\Lambda \approx 1$). Second, randomized loss weighting improves the overall accuracy in the multi-task training scenario.

Algorithm 1 summarizes the procedure to compute the ProtoRes loss based on one batch item. First, we sample (i) the number of effectors and (ii) their associated type and ID. For each effector, we randomly sample the tolerance level and compute the associated noise std and loss weight. Given noise std, an appropriate noise model is applied to generate input data based on effector type as described in Appendix B. Then ProtoRes predicts draft joint positions $\widetilde{\mathbf{f}}_{R,j}$, local joint rotations $\widehat{\mathbf{R}}_j$, as well as world-space rotations and positions $\widehat{\mathbf{G}}_j$ for all joints $j \in [0, J)$. We conclude by calculating the individual deterministic and randomized loss terms, whose weighted sum is used for backpropagation.

3 Empirical Results

Our empirical results based on two discrete pose datasets demonstrate that (i) ProtoRes reconstructs sparsely defined pose more accurately than existing non-ML IK solution, (ii) ProtoRes is more accurate than two baseline ML models, one based on Transformer and the other based on missing input masking, (iii) the proposed two-stage GPD+IKD decoding is more effective than a single-stage decoding, (iv) the proposed randomized loss weighting is effective in our multi-task training scenario, (v) the multi-task training using Geodesic and L2 losses produces synergetic effects.

3.1 Datasets

miniMixamo We use the following procedure to create our first dataset from the publicly available MOCAP data available from `mixamo.com`, generously provided by Adobe/Mixamo [1]. We download a total of 1598 clips and retarget them on our custom 64-joint skeleton using the Mixamo online tool. This skeleton definition is used in Unity to extract the global positions as well as global and local rotations of each joint at the rate of 60 frames per second (total 356,545 frames). The resulting dataset is partitioned at the clip level into train/validation/test splits (with proportion 0.8/0.1/0.1, respectively) by sampling clip IDs uniformly at random. Splitting by clip makes the evaluation framework more realistic and less prone to overfitting: frames belonging to the same clip are often similar. At last, the final splits retain only 10% of randomly sampled frames (miniMixamo has 33,676 frames total after

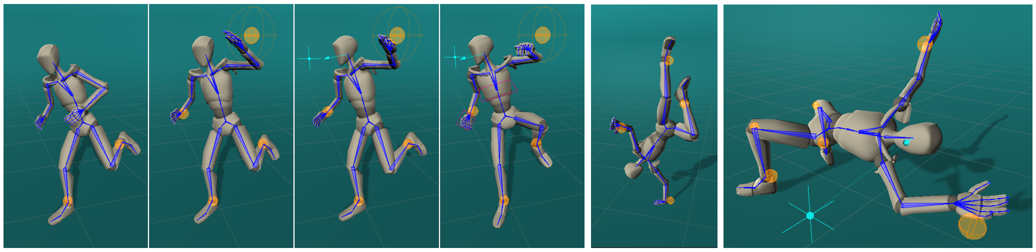

Figure 3: Qualitative posing results. Left 4 poses: adding position, look-at and rotation effectors to specify pose. Right 2 poses: achieving interesting poses with sparse constraints (4 and 7 effectors).

subsampling) and all the clip identification information (clip ID, meta-data/description, character information, etc.) is discarded. This scrambling and anonymization guarantee that the original sequences from `mixamo.com` cannot be reconstructed from our dataset, allowing us to release the dataset for reproducibility purposes without violating the original dataset license [1].

miniAnonymous To collect our second dataset we predefine a wide range of human motion scenarios and hire a qualified MOCAP studio to record 1776 clips (967,258 total frames @60 fps). Then we create a dataset of a total of 96,666 subsampled frames following exactly the same methodology that was employed for miniMixamo.

3.2 Training and Evaluation Setup

We use Algorithm 1 to sample batches of size 2048 from the training subset. The number of effectors is sampled once per batch and is fixed for all batch items to maximize data throughput. The training loop is implemented in PyTorch [33] using Adam optimizer [26] with a learning rate of 0.0002. Hyperparameter values are adjusted on the validation set (see Appendix C for hyperparameter settings). We report $\mathcal{L}^{det}_{gpd-L2}$, $\mathcal{L}^{det}_{ikd-L2}$, $\mathcal{L}^{det}_{loc-geo}$ metrics calculated on the test set, using models trained on the training set. These metrics characterise both the 3D position accuracy and the bone rotation accuracy. The evaluation framework tests model performance on a pre-generated set of seven files containing 6 to 12 effectors each. Skeleton is split in six zones, with four main zones including each limb, the hip zone and the head zone. In each file, we first sample one positional effector from each main zone. Remaining effectors are sampled randomly from all zones and effector types, mimicking pose authoring scenarios observed in practice. Metrics are averaged over all samples in all files, assessing the overall quality of pose reconstruction in scenario with sparse and variable inputs. All tables present results averaged over 4 random seed retries and metric values computed every 10 epochs over last 500 epochs, rounded to the last statistically significant digit.

3.3 Key Results

To demonstrate the advantage of the proposed architecture, we perform two evaluations. First, we compare ProtoRes against two ML baselines in the random effector evaluation setup described in Section 3.2. The first baseline, Masked-FCR, is a brute-force unstructured baseline that uses a very wide $J \cdot 3 \cdot 7$ input layer (J joints, 3 effector types, 6D effector plus one tolerance value) to handle all possible effector permutations. Each missing effector is masked with one of $3 \cdot J$ learnable 7D placeholders. Masked-FCR has 3 encoder and 6 decoder blocks to match ProtoRes. The second baseline is based on the Transformer [44] encoder (see Appendix E for hyperparameter settings), which receives variable length inputs $\mathbf{x}_{in} \in \mathbb{R}^{N \times E_{in}}$ (same as ProtoRes). The embeddings of joint IDs are used to query the Transformer output, producing one encoder embedding for each of J joints. The J encodings are fed into the 6-block FCR decoder (to match the total number of decoder blocks in ProtoRes) with two heads: one predicting rotation and one predicting unconstrained position. This is similar to the use of Transformer to predict bounding box class IDs and sizes for object detection [6]. Predictions of rotations and of the root joint are used in the forward kinematics pass, just as in ProtoRes. The bottom of Table 1, summarizing this study, shows clear advantage of ProtoRes w.r.t. both baselines. Additionally, training Transformer baseline on NVIDIA A6000 GPU for 15k epochs of miniAnonymous takes 200 hours @40GB GPU RAM usage, whereas training ProtoRes in the same conditions takes 33 hours @24GB GPU RAM usage. ProtoRes is clearly more compute efficient.

Table 2: Ablation studies on the random benchmark. Lower values are better.

		miniMixamo			miniAnonymous		
		$\mathcal{L}^{det}_{gpd-L2}$	$\mathcal{L}^{det}_{ikd-L2}$	$\mathcal{L}^{det}_{loc-geo}$	$\mathcal{L}^{det}_{gpd-L2}$	$\mathcal{L}^{det}_{ikd-L2}$	$\mathcal{L}^{det}_{loc-geo}$
		ProtoRes baseline					
		1.38e-3	**4.25e-3**	**0.2390**	**0.91e-3**	**3.19e-3**	**0.1810**
GPD	blocks, GPD/IKD	Ablation of GPD					
✗	0/6	1.51e-3	4.48e-3	0.2419	0.95e-3	3.44e-3	0.1838
$\mathcal{L}^{*}_{ikd-L2}$	$\mathcal{L}^{*}_{loc-geo}$	Ablation of rotation and position loss terms					
✓	✗	1.60e-3	4.39e-3	0.2794	1.15e-3	3.68e-3	0.2386
✗	✓	1.83e-3	5.85e-3	0.2422	1.46e-3	4.91e-3	0.1884
W_{pos}	Randomized Loss	Ablation of randomized loss weighting					
100	✗	1.80e-3	4.99e-3	0.2556	1.16e-3	3.58e-3	0.1902
1000	✗	1.66e-3	4.65e-3	0.2648	1.08e-3	3.41e-3	0.2035

Second, Table 1 (top) compares ProtoRes against a non-ML IK solution FinalIK [38], as well as Transformer and Masked-FCR, on a 5-point evaluation benchmark. The 5-point benchmark tests the reconstruction of the full pose from five position effectors: chest, left and right hands, left and right feet. It is chosen, because generating the exponentially large number of FinalIK configurations to process all heterogeneous effector combinations in the random benchmark is not feasible. Note that the 5-point benchmark and random benchmark results are not directly comparable. We can see that all ML methods significantly outperform FinalIK in reconstruction accuracy, ProtoRes being the best overall. This implies that the ML methods learn the right inductive biases from the data to solve the ill-posed sparse input pose reconstruction problem, unlike the pure non-learnable IK method FinalIK.

Third, qualitative posing results are shown in Fig. 3, demonstrating that visually appealing poses can be obtained with small number of effectors (4 and 7 effectors for the two poses on the right). The left 4 poses demonstrate how pose can be refined successively by adding more effectors. Please refer to Appendix F and supplementary videos for more demonstration examples.

3.4 Ablation Studies

Ablation of the GPD is shown in Table 2 (top). We keep all hyperparameters at defaults described in Section 3.2, remove GPD and increase IKD depth to 6 blocks to match the capacity of IKD+GPD. Comparing to the baseline, we see that GPD is useful as it creates small but consistent gain across metrics and datasets. GPD seems to provide some information to IKD that is hard to get otherwise.

Ablation of loss terms, shown in Table 2 (middle), studies the effect of (i) removing all L2 loss terms from the output of the FK pass and (ii) removing all Geodesic loss terms from the rotation output of IKD. Interestingly, removing either of the loss terms results in the degradation of all monitored metrics on both datasets. We conclude that jointly penalizing positions with L2 and rotations with Geodesic results in positive synergetic effects and improves the overall quality of pose model.

Ablation of randomized loss weighting is shown in Table 2 (bottom). The randomized loss weighting scheme is disabled by replacing all randomized loss terms with their deterministic counterparts. For example, $\mathcal{L}^{rnd}_{ikd-L2}$ is replaced with $\sum_{j=1}^{J} \mathrm{MSE}(\mathbf{g}_j, \widehat{\mathbf{g}}_j)$. It is clear that the inclusion of randomized weighting significantly improves generalization performance on all datasets and metrics. Additionally, when L2 weight W_{pos} increases with disabled randomized weighting, position L2 metrics improve, but at the expense of declining rotation metrics. Therefore, randomized weighting scheme contributes positive effect that cannot be achieved by simple tweaking of the deterministic loss weights.

The limitations of the current work, discussed in detail in Appendix G, include (i) the lack of temporal consistency as we focus on the problem of authoring a discrete pose, (ii) constraints are satisfied approximately, as opposed to the more conventional systems, (iii) exotic poses significantly deviating from the training data distribution (a common ML/DL problem) may be hard to achieve, (iv) a single skeleton layout with specific bone offsets is supported, (vi) the model allows interactive real-time rate of about 100 FPS, which is very good for interactive pose design. However, the current model cannot be used for run-time applications such as driving characters directly in real-time games.

4 Conclusions

We define and solve the discrete full-body pose authoring task using sparse user inputs. We define and release two datasets to support the development of ML models for discrete pose authoring and animation. We propose ProtoRes, a novel ML architecture which processes a variable number of heterogeneous user inputs (position, angle, direction) to reconstruct a full-body pose. We compare ProtoRes against two strong ML baselines, Masked-FCR and Transformer, showing superior results for ProtoRes, both in terms of accuracy and computational efficiency. We also show that ML models reconstruct full-body poses from sparse user inputs more accurately than existing non-learnable inverse kinematics models. We develop a suite of UI tools for the integration of our model in Unity and provide demos showing how our model can be used effectively to solve the discrete pose authoring problem by the end user. Our results have a few implications. First, our ML based tools will have positive impacts on the simplification and democratization of the game development process by helping a wider audience materialize their creative animation ideas. Second, our novel approach to neural pose representation could be applied in a wide variety of tasks where efficient and accurate reconstruction of full-body poses from noisy intermittent measurements is important.

References

[1] Adobe Inc. Adobe general terms of use. https://www.adobe.com/legal/terms.html, March 16, 2020. Accessed: 2021-05-06.

[2] A. Aristidou and J. Lasenby. Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011.

[3] M. Büttner and S. Clavet. Motion matching - the road to next gen animation. In *Proc. of Nucl.ai 2015*, 2015. URL https://www.youtube.com/watch?v=z_wpgHFSWss&t=658s.

[4] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, J. Yuan, and N. M. Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[5] N. Capece, U. Erra, and G. Romaniello. A low-cost full body tracking system in virtual reality based on microsoft kinect. In L. T. De Paolis and P. Bourdot, editors, *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 623–635. Springer International Publishing, 2018.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *ECCV 2020*, pages 213–229, 2020.

[7] Cascadeur. How to use deep learning in character posing. https://cascadeur.com/ru/blog/general/how-to-use-deep-learning-in-character-posing, 2019. Accessed: 2021-05-06.

[8] D. Casillas-Perez, J. Macias-Guarasa, M. Marron-Romera, D. Fuentes-Jimenez, and A. Fernandez-Rincon. Full body gesture recognition for human-machine interaction in intelligent spaces. In F. Ortuño and I. Rojas, editors, *Bioinformatics and Biomedical Engineering*, pages 664–676. Springer International Publishing, 2016.

[9] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. In *ACM Transactions on Graphics (ToG)*, volume 24, pages 686–696. ACM, 2005.

[10] Y. Cheng, B. Wang, B. Yang, and R. T. Tan. Graph and temporal convolutional networks for 3d multi-person pose estimation in monocular videos. *AAAI*, 2021.

[11] K. Cho and X. Chen. Classifying and visualizing motion capture sequences using deep neural networks. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, pages 122–130. IEEE, 2014.

[12] Facebook Reality Labs. Inside facebook reality labs: Research updates and the future of social connection. https://tech.fb.com/inside-facebook-reality-labs-research-updates-and-the-future-of-social-connection/. Accessed: 2021-04-30.

[13] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*, pages 522–531. 2004.

[14] F. Han, B. Reily, W. Hoff, and H. Zhang. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding*, 158:85–105, 2017.

[15] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. 39(4), 2020.

[16] C. Heindl, M. Ikeda, G. Stübl, A. Pichler, and J. Scharinger. Metric pose estimation for human-machine interaction using monocular vision. *ArXiv*, 2019.

[17] D. Holden. Robust solving of optical motion capture data by denoising. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[18] D. Holden, J. Saito, and T. Komura. Learning an inverse rig mapping for character animation. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 165–173, 2015.

[19] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[20] D. Holden, O. Kanoun, M. Perepichka, and T. Popa. Learned motion matching. *ACM Transactions on Graphics (TOG)*, 39(4):53–1, 2020.

[21] Y. Huang, M. Kaufmann, E. Aksan, M. J. Black, O. Hilliges, and G. Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[22] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5308–5317, 2016.

[23] B. Kenwright. Inverse kinematics–cyclic coordinate descent (ccd). *Journal of Graphics Tools*, 16(4):177–217, 2012.

[24] E. Khapugin and A. Grishanin. Physics-based character animation with cascadeur. In *ACM SIGGRAPH 2019 Studio*, SIGGRAPH '19, New York, NY, USA, 2019. Association for Computing Machinery.

[25] H. Kim, A. Zala, G. Burri, and M. Bansal. Fixmypose: Pose correctional captioning and retrieval. In *AAAI*, 2021.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[27] C. Kumar, J. Ramesh, B. Chakraborty, R. Raman, C. Weinrich, A. Mundhada, J. Arjun, and F. B. Flohr. VRU Pose-SSD: Multiperson pose estimation for automated driving. In *AAAI 2021*, 2021.

[28] K. Lee, S. Lee, and J. Lee. Interactive character animation by learning multi-objective control. In *SIGGRAPH Asia 2018 Technical Papers*, page 180. ACM, 2018.

[29] J. Lin and J. O'Brien. Temporal ik: Data-driven pose estimation for virtual reality. 2019.

[30] K. McDonald. Dance x machine learning: First steps. `https://medium.com/@kcimc/discrete-figures-7d9e9c275c47`. Accessed: 2021-05-03.

[31] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt. XNect: Real-time multi-person 3D motion capture with a single RGB camera. volume 39, 2020.

[32] M. S. Mirzaei, K. Meshgi, E. Frigo, and T. Nishida. Animgan: A spatiotemporally-conditioned generative adversarial network for character animation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2286–2290. IEEE, 2020.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS 2019*, pages 8024–8035, 2019.

[34] R. Paul. *Robot Manipulators: Mathematics, Programming, and Control : The Computer Control of Robot Manipulators*. The MIT Press Series in Artificial Intelligence. MIT Press, 1992.

[35] D. Pavllo, D. Grangier, and M. Auli. QuaterNet: A quaternion-based recurrent model for human motion. In *British Machine Vision Conference (BMVC)*, 2018.

[36] D. Rempe, L. J. Guibas, A. Hertzmann, B. Russell, R. Villegas, and J. Yang. Contact and human dynamics from monocular video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[37] Q. Riaz, G. Tao, B. Krüger, and A. Weber. Motion reconstruction using very few accelerometers and ground contacts. *Graphical Models*, 79:23–38, 2015.

[38] RootMotion. Advanced character animation systems for Unity. `http://root-motion.com`, 2021. Accessed: 2021-04-30.

[39] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H.-P. Seidel. Markerless motion capture of man-machine interaction. In *CVPR*, pages 1–8, 2008.

[40] S. S. M. Salehi, S. Khan, D. Erdogmus, and A. Gholipour. Real-time deep pose estimation with geodesic loss for image-to-template rigid registration. *IEEE transactions on medical imaging*, 38(2):470–481, 2018.

[41] J. Schwarz, C. C. Marais, T. Leyvand, S. E. Hudson, and J. Mankoff. Combining body pose, gaze, and gesture to determine intention to interact in vision-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3443—3452, New York, NY, USA, 2014. Association for Computing Machinery.

[42] S. Starke, Y. Zhao, T. Komura, and K. Zaman. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)*, 39(4):54–1, 2020.

[43] J. Tautges, A. Zinke, B. Krüger, J. Baumann, A. Weber, T. Helten, M. Müller, H.-P. Seidel, and B. Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (ToG)*, 30(3):18, 2011.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30, 2017.

[45] D. Xiang, H. Joo, and Y. Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[46] Xpire. Using ai to make nba players dance. `https://tinyurl.com/y3bdj5p5`. Accessed: 2021-05-03.

[47] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020.

[48] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI 2018*, volume 32, 2018.

[49] D. Yang, D. Kim, and S.-H. Lee. Real-time lower-body pose prediction from sparse upper-body tracking signals. *arXiv preprint arXiv:2103.01500*, 2021.

[50] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018.

[51] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The limitations are discussed in Section 3 and Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We carefully analyzed potential negative societal impacts of our work, but it was hard to identify any. For example, in the context of our problem, our architecture is more computationally efficient than transformer and has higher accuracy. Therefore, it is supposed to have a positive impact on (*i.e.* reduce) resources necessary to perform large-scale model training. Moreover, our goal is to provide AI assisted animation tools, which should lead to democratizing game development, lowering the bar for high-quality pose and animation creation. Therefore, our tools should enable small studios and game developers contribute to the game development process and successfully monetize the results of their work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and data, as well as instructions to reproduce the main experimental results will be released publicly if the paper is accepted

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We repeat experiments 4 times with respect to random seed. We average the results with respect to the 4 repeats and with respect to the metric values computed every 10 epochs in the last 500 epochs. We round the reported numbers to the last statistically significant digit. This implies that the experiment uncertainty for a number written as 1.000, for example, is in the range 0.991-1.009.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [No] The new assets will be released publicly if the paper is accepted

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We have personally communicated with representatives of Mixamo and described to them the detailed data curation/randomization/anonymization procedure we employed on their data to derive miniMixamo dataset. We obtained written permission to release the miniMixamo dataset alongside the published paper. We have described the data curation procedure in Section 3 and referred to the original data license.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data have been anonymized

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Supplementary Material for ProtoRes: Proto-Residual Architecture for Deep Modeling of Human Pose

Table of Contents

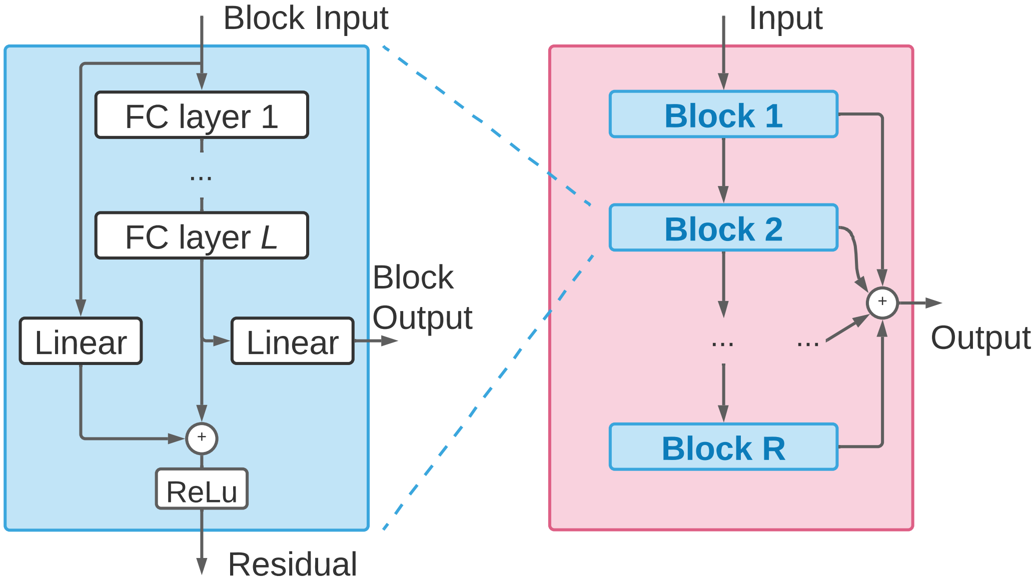

Figure 4: Block diagram of the fully-connected residual (FCR) decoder architecture. Left: the diagram of one residual block of the FCR decoder. Note that the basic residual block of the encoder architecture is exactly the same. Right: residual blocks connected in the FCR architecture.

A Architecture Details

Encoder The residual block depicted in Fig. 4 (left) is used as the basic building block of the ProtoRes encoder.

Decoders The block diagram of the global position and the inverse kinematics decoders used in the main architecture (see Fig. 2) is presented in Fig. 4. The architecture has fully connected residual topology consisting of multiple fully connected blocks connected using residual connections. Each block has residual and forward outputs. The forward output contributes to the final output of the decoder. The residual connection sums the hidden state of the block with the linear projection of the input and applies a ReLU non-linearity.

In the main text we use a convention that the number of layers and blocks in the encoder, as well as in GPD and IKD decoders is the same and is given by L and R respectively. Obviously, using a different number of layers and residual blocks in each of the blocks might be more optimal.

B Effector noise model

This section describes the details of the of the NoiseModel that is used in Algorithm 1 to corrupt model effector input $\mathbf{x}[i,:]$ based on appropriate noise level $\sigma(\Lambda_i)$.

B.1 Position effector noise model

If effector type is positional ($T_i = 0$), *i.e.* effector i is a coordinate in 3D space, typically corresponding to the desired position of joint I_i in 3D space, we employ Gaussian white noise model:

$$\mathbf{x}[i,1:3] = \mathbf{g}_{I_i} + \sigma(\Lambda_i)\varepsilon_i; \quad \mathbf{x}[i,4:6] = 0. \tag{12}$$

Here $\mathbf{x}[i,:]$ is the i-th model input, $\mathbf{g}_{I_i}$ is the ground truth location of joint I_i, $\sigma(\Lambda_i)$ is the noise standard deviation computed based on eq. (10) and ε_i is a 3D vector sampled from the zero-mean Normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$.

Hyperparameter	Value	Grid
Epochs, miniMixamo/ miniAnonymous	40k/15k	[20k, 40k, 80k] / [10k, 15k, 40k]
Losses	MSE, GEO, LAT	MSE, GEO, LAT
Width (d_h)	1024	[256, 512, 1024, 2048]
Blocks (R)	3	[1, 2, 3]
Layers (L)	3	[2, 3, 4]
Batch size	2048	[512, 1024, 2048, 4096]
Optimizer	Adam	[Adam, SGD]
Learning rate	2e-4	[1e-4, 2e-4, 5e-4, 1e-3]
Base L2 loss scale (W_{pos})	1e2	[1, 10, 1e2, 1e3, 1e4]
Max noise scale ($\sigma_{M,0}$, $\sigma_{M,1}$)	0.1	[0.01, 0.1, 1]
Max effector weight (W_M)	1e3	[10, 1e2, 1e3, 1e4]
Noise exponent, η	13	13
Dropout	0.01	[0.0, 0.01, 0.05, 0.1, 0.2]
Embedding dimensionality	32	[16, 32, 64, 128]
Augmentattion	mirror, rotation	[mirror, rotation, translation]

Table 3: Settings of ProtoRes hyperparameters and the hyperparameter search grid.

B.2 Rotation effector noise model

If effector type is angular ($T_i = 1$), *i.e.* effector i is a 6DoF rotation matrix representation, we employ random rotation model that is implemented in the following stages. First, suppose $\mathbf{f}_{I_i}$ is the ground truth 6DoF representation of the global rotation of joint I_i corresponding to effector i. We transform it to the rotation matrix representation $\mathbf{G}^{13}_{I_i}$ using equation (5). Second, we generate the random 3D Euler angles vector ε_i from the zero-mean Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma(\Lambda_i)\mathbf{I})$[1] and convert it to the random rotation matrix Ψ_i using eq. (13):

$$\Psi_i = \begin{bmatrix} \cos\varepsilon_i[1] & -\sin\varepsilon_i[1] & 0 \\ \sin\varepsilon_i[1] & \cos\varepsilon_i[1] & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varepsilon_i[2] & 0 & \sin\varepsilon_i[2] \\ 0 & 1 & 0 \\ -\sin\varepsilon_i[2] & 0 & \cos\varepsilon_i[2] \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varepsilon_i[3] & -\sin\varepsilon_i[3] \\ 0 & \sin\varepsilon_i[3] & \cos\varepsilon_i[3] \end{bmatrix}. \quad (13)$$

Third, we apply random rotation to the ground truth matrix, $\mathbf{G}^{13\prime}_{I_i} = \Psi_i \mathbf{G}^{13}_{I_i}$. Finally, we convert the randomly perturbed rotation matrix back to the 6DoF representation:

$$\mathbf{x}[i, 1:3] = \mathbf{G}^{13\prime}_{I_i}[:, 1], \quad \mathbf{x}[i, 4:6] = \mathbf{G}^{13\prime}_{I_i}[:, 2]. \quad (14)$$

B.3 Look-at effector noise model

If effector type is look-at ($T_i = 2$), *i.e.* effector i is a position of the target at which a given joint is supposed to look, we employ random sampling of the target point along the ray formed by the ground truth global rotation of a given joint.

First, we sample the local direction vector $\mathbf{d}_i$ from the zero-mean normal 3D distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and normalize it to unit length. Second, we sample the distance between the joint and the target object, $d_{\mathbf{t}}$, from the normal distribution $\mathcal{N}(0, 5)$ folded over at 0 by taking the absolute value. The location of the target object is then determined as $\mathbf{t}_i = \mathbf{g}_{I_i} + d_{\mathbf{t}}\mathbf{d}_i + \sigma(\Lambda_i)\varepsilon_i$. Finally, the output is constructed as follows:

$$\mathbf{x}[i, 1:3] = \mathbf{t}_i, \quad \mathbf{x}[i, 4:6] = \mathbf{d}_i. \quad (15)$$

As previously, ε_i is a 3D vector sampled from the zero-mean Normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

C Training Setup: Details

Hyperparameter settings This section presents the details of hyperparameter setting for the proposed architecture. We tried to use SGD optimizer to train the architecture, but it was very

[1] Note that in the case of angles, sampling from the Tikhonov (a.k.a. circular normal or von Mises) distribution might be a better idea, but Gaussian worked well in our case.

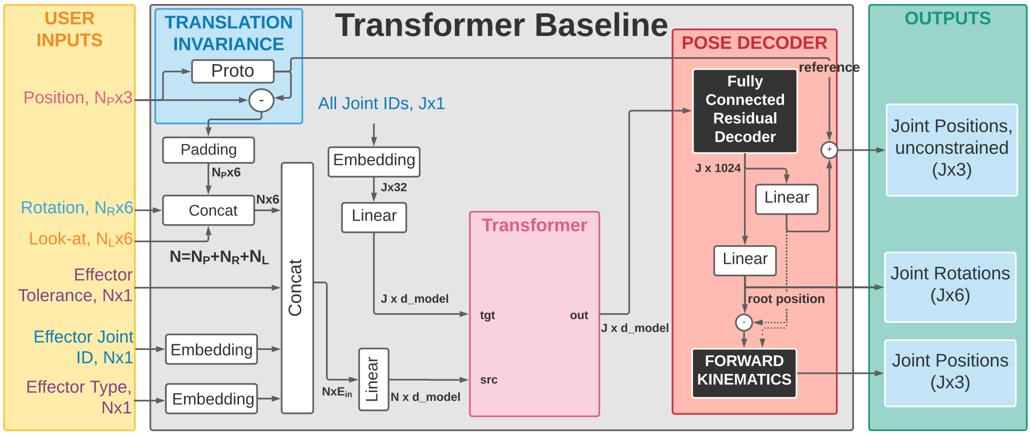

Figure 5: Block diagram of the Transformer baseline architecture.

difficult to obtain stable results with it. Adam optimizer turned out to be much more suitable for our problem. The learning rate was selected to be 0.0002, which is lower than Adam's default. Obtaining stable training results with higher learning rates was not feasible, even with increased batch size. Batch size is selected to be 2048 to accelerate training speed. In practice we observed slightly better generalization results with smaller batch size (1024 and 512). The detailed settings of ProtoRes hyperparameters are presented in Table 3.

D Masked-FCR baseline architecture

Masked-FCR is a brute-force unstructured baseline that uses a very wide $J \cdot 3 \cdot 7$ input layer (J joints, 3 effector types, 6D effector plus one tolerance value) to handle all possible effector permutations. Each missing effector is masked with one of $3 \cdot J$ learnable 7D placeholders. Masked-FCR has 3 encoder and 6 decoder blocks to match ProtoRes.

E Transformer baseline architecture

We implement Transformer baseline using the default Transformer module available from PyTorch `https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html`. The block diagram of the Transformer baseline architecture is shown in Fig. 5. We use a standard transformer application scenario in which the transformer source input is fed with the variable length input and the required outputs are queried via target input. In our case the variable length input corresponds to the effector data concatenated with embedded effector categorical variables. The query for the output consists of the embeddings of all joints. Note that the joint embedding is reused both for source and target inputs and both inputs are projected to the internal d_model dimensionality of transformer. Internally, Transformer processes both source and target inputs via self-attention first and then applies the multi-head attention between source and target after self-attention. This results in the output embedding for each skeleton joint that depends on all the input information as well as the learned interactions across all output skeleton joints. The output embedding of transformer is then decoded to unconstrained position and rotation outputs using two-headed Fully-Connected residual stack (this is the same architecture as the one used in ProtoRes decoders). Note that this is a classical Transformer application scheme that has recently been used to achieve SOTA results in object detection, for example [6]. Table 4 lists the hyperparameter settings for the Transformer baseline. Note that only hyperparameters that are unique to this baseline or different from the ProtoRes defaults appearing in Table 3 are listed.

F Videos and demonstrations

We provide here descriptions related to each video found in the supplementary materials. Note that most of the demonstrations are done using a ProtoRes model trained on a large internal dataset not

Hyperparameter	Value	Grid
FCR decoder parameteres		
FCR Blocks (R)	6	N/A
FCR Width (d_h)	1024	N/A
Transformer parameteres		
d_model	128	[64, 128, 256]
nhead	8	[1, 2, 4, 8]
num_encoder_layers	2	[1,2,3]
num_decoder_layers	2	[1,2,3]
dim_feedforward	1024	[256, 512, 1024]
dropout	0.01	0.01
activation	relu	relu
Base L2 loss scale (W_{pos})	10	[10, 100]

Table 4: Settings of Transformer baseline hyperparameters and the hyperparameter search grid.

evaluated in this work. One of our demonstrations, described below in Appendix F.5 qualitatively shows some of the most severe impacts of training on ablated datasets.

F.1 ProtoRes Demo

The video presents an overview of the integration of ProtoRes as a posing tool inside the Unity game engine, showcasing how different effector types can be manipulated and how they can influence the resulting pose. In this demo, a user-defined configuration is presented in the UI, allowing one to choose which effectors are enabled within that configuration. Note that this configuration could contain more or less effectors of each type, and can be built for specific posing needs. The most generic configuration would present all possible effectors inside each effector type sub-menu.

F.2 Posing from Images

This video presents screen recordings of a novice user using ProtoRes to quickly prototype poses taken from 2D silhouette images. Note that one can reach satisfactory results in less than a minute in each case, with a relatively low number of manipulations. Note also that fine-tuning the resulting poses can always be achieved by adding more effectors and applying more manipulations.

F.3 Loss Ablation

This recording shows a setup where different models are used with identical effector setup. Both models use the ProtoRes architecture. On the left, the model uses the total loss presented in Algorithm 1, whereas the right-hand side model uses positional losses only (GPD and IKD positional losses) and both local and global rotational losses, as well as look-at losses are disabled. This demonstration clearly shows how positional constraints, even when respected, do not suffice to produce realistic human poses. Joint rotations have to be modeled as well.

F.4 FinalIK Comparison

This recording shows a setup where ProtoRes is compared to a full body biped IK system provided by FinalIK [38], with an identical effector setup. Note that FinalIK solves constraints by modifying the current pose, often resulting in smaller changes in the output, when compared to ProtoRes that predicts a full pose at each update. The lack of a learned model of human poses in FinalIK becomes quickly noticeable when manipulating effectors significantly.

F.5 Datasets Comparison

In these demonstrations, we showcase how training ProtoRes on different datasets can impact the results. We showcase models trained on the two ablated datasets presented in this work, i.e.

miniAnonymous (left) and miniMixamo (right). We also show performance of a model trained on the full Mixamo [1] dataset (center) to qualitatively show how performance can be improved with more training data. In all of these recordings, one can notice differences in the resulting poses, emphasizing the fact that human posing from few effectors, when no extra conditioning signals are used, is an ambiguous task that will be influenced by the training data.

The first sequence makes this fact especially obvious on the finger joints and the head's look-at direction. The second sequence shows how good data coverage in the training set can significantly impact performance in special or rare effector configurations. Finally, the third sequence shows a similar pattern for look-at targets, where the difference in training data can be noticed in the general posture of the character and the varying levels of robustness with respect to those targets.

F.6 Limitations

The final video shows examples of some specific limitations of the approach that are listed in Appendix G. Namely, we first expose specific consequences of the lack of temporal consistency in our problem formulation, where smoothly moving an effector can cause flickering on some joints, such as the fingers. We also show how between some effector configurations, the character must be flipped completely to stay in a plausible pose, and how it's possible to place some effectors to reach an invalid pose coming from that *flip region* of the latent manifold. Finally, we showcase some problematic behaviors that can be caused by extreme look-at targets. In some cases, especially with many other constraints, ProtoRes will tend to produce a plausible pose that will not respect the look-at constraint. In other cases, the extreme look-at target may cause an unrealistic pose, e.g. by causing the character to have an impossible neck rotation. It is interesting to note how invalid poses from look-at effectors tend to happen more often than from other effector types with novice users. We hypothesize that the plausible region of a look-at target, given a current character pose, is less intuitive to grasp than for other effector types. Indeed, the current pose of the character seems to guide more precisely the placement of positional and rotational effectors than look-at effectors, leading more often to configurations outside of the training distribution for look-at effectors.

G Limitations

The limitations of our work can be summarized as follows:

- Constraints are not satisfied exactly, as opposed to the conventional systems. This is the price to pay for the ability of the model to inject the data-driven inductive bias that can be used to reconstruct pose from very sparse inputs. This could be mitigated using a conventional solver on top of the trained model. In this case, the model will produce a globally plausible pose, whereas the solver will only do the final pass to strictly satisfy certain constraints. Also, to provide additional flexibility in solving some of the constraints more strictly than the others, our model provides an effector tolerance mechanism that can help the user trade off the strictness of satisfying certain effectors vs. some others.
- Lack of temporal consistency. Our work solves the problem of creating a discrete pose. Therefore, it is limited in how it can be applied to modify an underlying smooth animation clip. For example, we can see flickering of joints (especially fingers) when effectors follow an underlying smooth animation (the finger embedding space is not smooth and has a high ambiguity). This happens to a smaller degree with the head when it is not constrained with look-at or rotation inputs.
- Exotic poses significantly deviating from the the training data distribution (a common ML/DL problem) may be hard to achieve. For example, the Lotus yoga pose is very hard to achieve with small number of effectors. Extreme or rare effector configurations may not be respected. Extreme look-at targets may not be followed or can cause artifacts in the resulting pose.
- Some effector displacement can cause a complete flip in the final pose as it makes more sense to be e.g. left-oriented or right-oriented to reach a hand position. This is normal, but we can sometimes reach "in-between" poses on the boundary of the hand effector that causes the flip, leading to weird poses

- We also noted a limitation as "aiming" poses (holding something in the hands). For example, Finger poses are generally wrong w.r.t. to a gun without additional finger constraints. It may be cumbersome to place hands + look-at for each aiming pose/angle?
- Runtime. In its current state, the model allows interactive real-time rate (about 100 FPS). This is very good for the primary application area of the model in the interactive pose design. However, the current model cannot be used for runtime applications such as driving charachters directly in real-time games, because it would consume too high of a time budget (about 10ms, which is too much to be usable in the game runtime context).
- No contextual input is supported (text description or environment awareness), in particular for finger posing and feet collisions
- Single skeleton layout with specific bone offsets is currently supported. A new skeleton requires either a re-trained model, or a retargeting pass, which may be expensive and has the potential to reduce the realism of the reconstructed pose.
- The approach relies on MOCAP data. This type of data may be hard to obtain for certain charachters, for example an octopus.
- Good for realism, but might limit creativity. In particular, no bone stretching support, which is sometimes used by animators to add more expressiveness to non-realistic characters.
- The current model struggles when a large number of fine-grain controls, especially fingers are used simultaneously. Perhaps, a more structured hierarchical approach can be used to enable this functionality.