CSE 586 Project 1 — Project Proposal

Body Pose Forecasting

Project group: Tzuhsin Chiu.

1 Project Title & Objective

Title: Body Pose Forecasting with Transformer.

Objective: We predict future body poses from past poses. We use pose data from AMASS in the form of SMPL parameters, encoded into Vposer's 32-D latent space. Predicting 2–5 seconds ahead is hard because small errors can grow over many frames. We use a **Transformer** to model the sequence and aim to beat the **zero-velocity** and **constant-velocity** baselines. Unlike language models like makemore (discrete tokens), we predict continuous vectors at each step. For quantitative evaluation we will use **MPJPE** and compare against zero-velocity and constant-velocity baselines (details in Evaluation Plan).

2 Proposed Architecture (Transformer)

Input/Output: We feed in a sequence of 32-D latent vectors (one per frame). The model outputs a 32-D vector for the next frame, or a block of future vectors. At the end we use a linear layer that outputs 32 numbers. We do not use softmax or cross-entropy, since we are doing regression, not classification.

Components: The core is a **Transformer** with **self-attention** blocks. Attention lets the model use information from all past frames when predicting the next one. Each position in the sequence holds one 32-D vector. We add positional encoding to let the model know the order of frames. After decoding the predicted latent back to SMPL with Vposer, we can get 3D joint positions for evaluation.

Loss: We use **MSE** or **MAE** between the predicted and true latent vectors. We may try both and pick the one that trains better.

3 Long-term Forecasting Strategy

At 120 fps, 5 seconds is 600 steps. Predicting one frame at a time for that long often causes errors to build up. We plan to try:

- **Subsample:** Use **30 fps** instead of 120 fps (keep every 4th frame). That cuts the number of steps by 4 and can reduce drift.
- **Block prediction:** Instead of predicting only the next vector, predict a block $[z_{t+1}, \ldots, z_{t+k}]$ in one forward pass. Then use the last predicted vector as input and predict the next block. This way we do fewer autoregressive steps.
- **Multi-step targets:** During training, sometimes ask the model to predict k steps ahead (e.g., 1–2 s) instead of only the next step. This can help the model learn longer-range motion.

4 Evaluation Plan

Baselines: We compare against two simple methods. **Zero-velocity:** repeat the last observed pose for all future frames (Martinez et al. show this is already strong). **Constant-velocity:** compute velocity from the last two frames and move the pose forward with that velocity. Our model should get lower error than both.

Metric: MPJPE (mean per joint position error). We decode predicted latent vectors to SMPL, get 3D joint positions, and take the mean Euclidean distance to the ground-truth joints. Lower is better. We report MPJPE at several horizons: 80 ms, 320 ms, 560 ms, 1 s, 2 s, and up to 5 s after the input ends.

Data and setup: We split AMASS into separate train, test and optionally validation sets. We use a fixed body shape and no global rotation or translation, so poses are aligned and comparison is straightforward. We will also show a few example sequences (best, average, worst) as visualizations.

5 Computational Platform

We plan to use **Google Colab** for most training and experiments. If we need more compute or longer runs, we may use a personal laptop or school lab machines.