

3D human interaction synthesis for action recognition data augmentation

Master Thesis



3D human interaction synthesis for action recognition data augmentation

Master Thesis

June, 2024

By

Anders Bredgaard Thuesen

Copyright: Reproduction of this publication in whole or in part must include the customary bibliographic citation, including author attribution, report title, etc.

Cover photo: Vibeke Hempler, 2012

Published by: DTU, Department of Applied Mathematics and Computer Science,
Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk

ISSN: [0000-0000] (electronic version)

ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)

ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over six months at the Section for Indoor Climate, Department of Civil Engineering, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Engineering, MSc Eng.

It is assumed that the reader has a basic knowledge in the areas of statistics.

Anders Bredgaard Thuesen - s183926

.....
Signature

.....
Date

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Acknowledgements

Anders Bredgaard Thuesen, MSc Civil Engineering, DTU

I would like to thank my supervisors for their feedback during my writing of this thesis.

Morten Rieger Hannemose, [Assistant Professor], [affiliation]

[text]

[Name], [Title], [affiliation]

[text]

Contents

Preface	ii
Abstract	iii
Acknowledgements	iv
1 Introduction	1
2 Background	3
2.1 Skinned Multi-Person Linear Model (SMPL)	3
2.2 Human Pose Estimation and Mesh Recovery	4
2.3 Denoising Diffusion Probabilistic Models (DDPM)	5
2.3.1 Variance schedules	7
2.3.2 Classifier and Classifier-Free Guidance (CFG)	8
2.4 Transformer architecture	9
2.5 Human Motion Generation	10
2.5.1 Human Motion Diffusion Model (MDM)	10
2.5.2 InterGen	10
3 Data	11
3.1 HumanML3D	11
3.2 InterHuman	11
3.3 Teton (raw) dataset	11
4 Methods	13
4.1 3D scene reconstruction	13
4.2 Human pose sequence optimization	13
4.3 Unification of datasets	14
4.4 Motion and scene representation	15
4.5 Diffusion model	15
5 Results	17
6 Discussion & Conclusion	19
Bibliography	21
A Appendices	23
A.1 Derivation of posterior distribution	23
A.2 L_{t-1} closed form derivation	23

1 Introduction

In healthcare environments such as hospitals and care homes, data pertaining to critical incidents like falls are scarce due to their infrequent nature and the high privacy requirements surrounding such data. This scarcity poses significant challenges for training robust machine learning models, particularly in applications related to video classification where detailed understanding of such events is crucial. The primary goal of this thesis is to enhance the performance of video classification tasks by leveraging synthetic data, which is designed to closely mirror the underlying distribution of real-world incidents while enabling focused studies on specific, rare events.

To address the challenges inherent in collecting and utilizing real-world data from sensitive environments, this research proposes a novel approach using synthetic data generation. Synthetic data not only adheres to the distributional characteristics of genuine data but also provides flexibility to explore less common scenarios—specifically, those at the tail of the distribution which are typically underrepresented in available datasets.

This work introduces a sophisticated framework for generating synthetic data by explicitly modeling three-dimensional (3D) environments. This includes detailed interactions both among humans and between humans and their surroundings. By integrating these complex interactions as a strong inductive bias, the proposed generative diffusion model enhances the realism and applicability of the synthetic data.

To construct and train this model, we utilize publicly available datasets such as HumanML3D and InterHuman. These datasets include motion capture data of individuals and pairs interacting, each accompanied by textual descriptions. These are combined with 3D scene reconstructions derived from video captures in actual hospital and care home settings. This integration of human motion and scene specifics forms the foundation for our synthetic data generation process.

To effectively reconstruct 3D scenes from the captured videos, we employ state-of-the-art models such as ProHMR and Depth Anything. These models are instrumental in generating per-frame human pose estimations and scene depth labels. These outputs, along with 2D keypoint annotations, are fed into a joint optimization process. This process is critical as it unifies the coordinate systems of the human models and the environment, ensuring that the motion trajectories are smooth and coherent. The result is a highly accurate 3D representation of the scenes, which serves as a vital input for our synthetic data generation.

2 Background

2.1 Skinned Multi-Person Linear Model (SMPL)

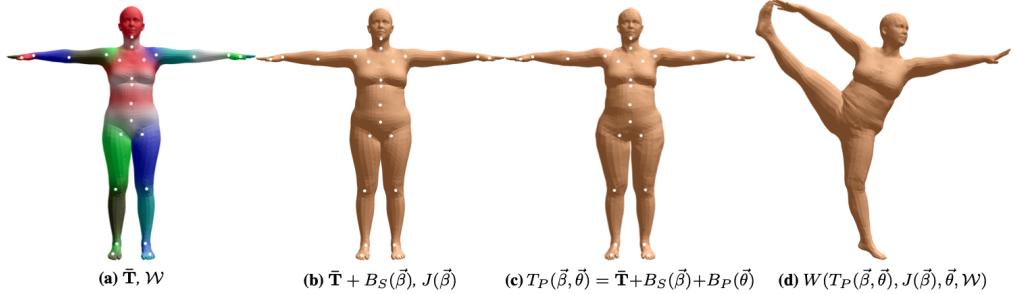


Figure 2.1: SMPL model (Image credit: Loper et al. 2015)

The Skinned Multi-Person Linear (SMPL) model (Loper et al. 2015) is a parametric model designed to accurately represent a diverse array of human body shapes and poses. The model is constructed on the principles of Linear Blend Skinning (LBS), augmented with corrective blend shapes derived from an extensive dataset of body scans. This approach allows the SMPL model to capture the nuanced deformations associated with different body shapes and poses, ensuring high fidelity and realism in representation. Furthermore, the model is compatible with standard graphics pipelines, facilitating its easy integration and rendering.

Since its inception, the SMPL model has been extended through various enhancements. Notable among these are SMPL-H (Romero, Tzionas, and Black 2017), which models detailed hand movements, and SMPL-X (Pavlakos et al. 2019), which extends the model to include both detailed hand movements and facial expressions. Additionally, Dynamic-SMPL or DMPL (Loper et al. 2015) incorporates dynamic soft-tissue deformations, which simulate the natural movement of soft tissues during movement.

The SMPL model is parameterized by $\vec{\beta}$, which encapsulates the deviations from a mean body shape, and $\vec{\theta}$, which specifies the axis-angle rotations of the 23 joints in the template skeleton. These rotations are local and are applied using a kinematic tree structure, which ensures that the hierarchical relationships between body parts are accurately represented. Mathematically, the model can be expressed as:

$$M(\vec{\beta}, \vec{\theta}) = W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W}) \quad (2.1)$$

where $T_P(\vec{\beta}, \vec{\theta})$ denotes the vertices of the rest pose, incorporating deformations due to body shape and pose, and is defined as:

$$T_P(\vec{\beta}, \vec{\theta}) = \bar{T} + B_S(\vec{\beta}) + B_P(\vec{\theta}) \quad (2.2)$$

Here, \bar{T} represents the template mesh, $B_S(\vec{\beta})$ accounts for shape-dependent deformations, and $B_P(\vec{\theta})$ represents pose-dependent deformations.

The joint locations, $J(\vec{\beta})$, are derived from the shaped template vertices using a learned regression matrix \mathcal{J} , and can be formulated as:

$$J(\vec{\beta}) = \mathcal{J}(\bar{T} + B_S(\vec{\beta})) \quad (2.3)$$

The skinning function, W , which can be either LBS or Dual-Quaternion Blend Skinning (DQBS), applies these deformations to the skeletal structure, with \mathcal{W} representing the blend weights.

Overall, the SMPL model’s parametric nature and its capability to accurately represent a wide range of human anatomies and movements establish it as a valuable tool in the fields of computer graphics, animation, and related areas.

2.2 Human Pose Estimation and Mesh Recovery

Human Pose Estimation (HPE) and Mesh Recovery (HMR) are pivotal areas within computer vision and graphics, involving the inference of three-dimensional human body shapes and their articulated poses from images or videos. These tasks have widespread applications, including virtual reality, motion capture, animation, and human-computer interaction.

HPE traditionally refers to predicting the positions of key joints in a human body. Advances in this field have transitioned from 2D pose estimation, where joint positions are determined in image space, to 3D pose estimation, which involves inferring joint positions in three-dimensional space. HMR extends beyond pose estimation by reconstructing a detailed 3D mesh of the human body, capturing the surface geometry and volume in addition to joint locations. The SMPL model is often used in this context, providing a standardized framework for representing human body shapes and poses, thereby enabling more accurate and consistent mesh recovery. Additionally, the SMPL model reduces the number of parameters required to represent the human body, simplifying the computational complexity while maintaining a high level of detail.

One traditional approach to HPE involves fitting 3D models to 2D keypoints extracted from images. This method, exemplified by techniques such as SMPLify by Bogo et al. 2016, first detects 2D joint positions (bottom-up) using a keypoint detector. These detected keypoints are then used (top-down) to fit a pre-defined 3D human body model, such as SMPL. This fitting process involves optimizing the model parameters to minimize the difference between the projected 3D keypoints and the detected 2D keypoints in the image. Such methods often require a strong prior to avoid generating highly implausible poses, as the optimization process can otherwise lead to unrealistic human body configurations. An example of such a prior is vPoser (Pavlakos et al. 2019), which uses a variational autoencoder to learn a latent space of plausible human poses, thereby helping to constrain the optimization process to more realistic pose configurations.

More recent approaches bypass the intermediate step of 2D keypoint detection and directly regress the parameters of 3D models from images. This end-to-end regression is performed using deep convolutional neural networks (CNNs) that predict the 3D pose and shape parameters directly from input images. HMR (Kanazawa et al. 2018) represent this category, where the models are trained to predict the SMPL parameters directly, thereby streamlining the pipeline and potentially increasing the robustness to errors in 2D keypoint detection.

Above methods perform HPE on a per-image basis without considering temporal information. Each frame is processed independently, which simplifies the modeling process but can lead to inconsistencies in predictions across consecutive frames. Temporal models, on the other hand, leverage information from multiple frames to ensure consistent and stable pose estimations over time. This is particularly important in video analysis where the continuity of human motion must be preserved. VIBE (Kocabas, Athanasiou, and Black 2020) is an example of a temporal model that uses recurrent neural networks (GRUs) to

incorporate temporal information, resulting in more stable and accurate pose estimations in videos.

Deterministic models provide a single estimate of the human pose for a given input, without accounting for uncertainties. Most early HPE methods fall into this category, where the predicted pose is a fixed output of the model's inference process. Probabilistic models, in contrast, explicitly model the uncertainty in pose estimation, providing a distribution over possible poses rather than a single deterministic output. This approach allows the model to capture ambiguities inherent in 2D-to-3D mapping. ProHMR (Kolotouros et al. 2021) is a representative of probabilistic models, where the network outputs a probability distribution over 3D poses, enabling it to better handle ambiguities and provide more informative predictions.

2.3 Denoising Diffusion Probabilistic Models (DDPM)

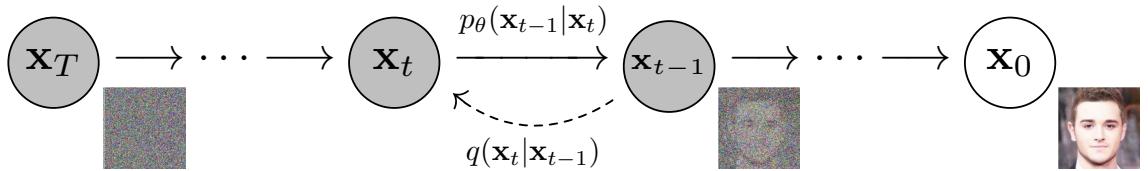


Figure 2.2: Diffusion forward and backward process (taken from Ho, Jain, and Abbeel 2020)

In recent years, several types of generative models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), autoregressive models and flow-based models have shown remarkable results in data generation of varying data modalities, such as images, audio, videos and text. Most recently, Denoising Diffusion Probabilistic Models (DDPMs) have gained large popularity especially within the field of image generation due to several reasons such as high-quality data generation, versatility in several data domains as well as controllability, allowing one to steer the generation towards desired outputs.

A DDPM is a parametrized Markov chain trained using variational inference to reverse a (forward) diffusion process, $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$, as seen in fig. 2.2 wherein the signal of the data, \mathbf{x}_0 , is gradually destroyed by adding gaussian noise according to predefined noise schedule $\{\beta_t \in (0, 1)\}_{t=1}^T$ giving rise to increasingly noisy samples, $\mathbf{x}_1 \dots \mathbf{x}_T$.

The forward process is defined as follows:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} \mathbf{x}_t, \beta_t \mathbf{I}) \quad (2.4)$$

with T being the discretized number of diffusion steps before all original information is completely discarded. The goal of the inverse or backwards process then becomes to iteratively remove the noise, in order to arrive at the original data. More formally, the process is defined as:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (2.5)$$

taking starting point in pure noise $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, incrementally removing the noise through the learned functions, $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$ commonly parameterized by a deep

neural network. Using the reparameterization trick, we are able to sample any noisy version of our data, \mathbf{x}_t , at time step t given our original data \mathbf{x}_0 . Recall our forward transition probability function, $q(\mathbf{x}_t | \mathbf{x}_{t-1})$. Letting $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ and using the reparameterization trick the expression can be rewritten as:

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \epsilon_{t-1} \quad (2.6)$$

where $\epsilon_{t-1} \sim \mathcal{N}(0, 1)$. Expanding the recursive definition then gives:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t} \left(\sqrt{\alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_{t-1}} \epsilon_{t-2} \right) + \sqrt{1 - \alpha_t} \epsilon_{t-1} \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\epsilon}_{t-2} \end{aligned}$$

where $\bar{\epsilon}_{t-2}$ merges the two independent Gaussians ϵ_{t-1} and ϵ_{t-2} into a single Gaussian with new variance as the sum of variances $\alpha_t(1 - \alpha_{t-1}) + (1 - \alpha_t) = 1 - \alpha_t \alpha_{t-1}$. Recursively applying the definition of \mathbf{x}_t and merging the gaussian noise terms results in the simplified expression:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (2.7)$$

Conversely, given \mathbf{x}_0 , the reverse conditional probability:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I} \right)$$

becomes tractable to compute using Bayes rule (derivation in appendix A.1) with mean and variance given by:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\alpha_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t \stackrel{\text{Using 2.7}}{=} \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right) \quad (2.8)$$

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \quad (2.9)$$

The model is trained by optimizing the variational lower bound on the log likelihood:

$$L_{VLB} = \mathbb{E}_q \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] = \mathbb{E}_q \left[\log p(\mathbf{x}_T) + \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] \leq \mathbb{E} [\log p_\theta(\mathbf{x}_0)] \quad (2.10)$$

which in practice means minimizing the negative variational lower bound. Ho, Jain, and Abbeel 2020 rewrites this into a sum of KL-divergences:

$$L_{VLB} = \mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right] \quad (2.11)$$

where the authors model L_0 from separate discrete decoder. As L_T doesn't depend on our parameters, θ , and is therefore constant, it can be ignored during optimization. The rest of the L_{t-1} terms can be efficiently computed in the closed form, by fixing the variance $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ to only depend on the current timestep (authors propose $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t$). L_{t-1} can then be written in closed form (derivation in appendix A.2) as:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C_t \quad (2.12)$$

Algorithm 1 Training

```

1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
       $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon, t)\|^2$ 
6: until converged

```

Algorithm 2 Sampling

```

1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 

```

where C_t is a constant depending on the choice of σ_t^2 and the noise schedule. Using eq. (2.8) Ho, Jain, and Abbeel 2020 reparameterize the expression in terms of predicting the noise:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2 \right] \quad (2.13)$$

as they find it leads to better unconditional sample quality when training on the CIFAR10 dataset. Furthermore, they report the best sample quality when using the "simple" objective, ignoring the weighing:

$$L_{\text{simple}} = \mathbb{E}_{t \sim \mathcal{U}(1, T), \epsilon} [\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}} \epsilon, t)] \quad (2.14)$$

The final training and sampling scheme, as outlined in algorithm 1 and algorithm 2 can be summarized as follows. The training algorithm optimizes the model to predict and remove noise from data, while the sampling algorithm uses the trained model to iteratively transform random noise into structured data.

2.3.1 Variance schedules

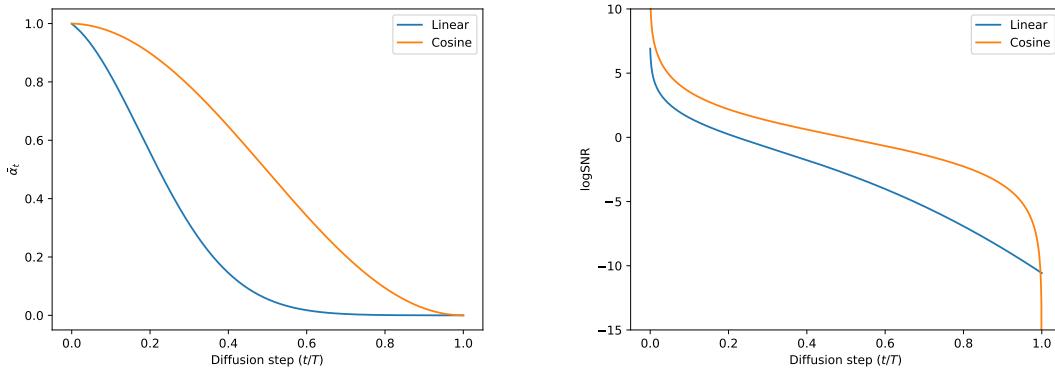


Figure 2.3: Linear and cosine schedule and signal-to-noise ratio.

In the DDPM paper Ho, Jain, and Abbeel 2020 choose a variance schedule with β_t linearly increasing from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. However, this results in \mathbf{x}_t almost entirely losing its signal in the last quarter of the schedule as problematized by A. Q. Nichol and Dhariwal 2021. The authors instead propose a cosine variance schedule, where the squared signal proportion is given by:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos \left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2} \right)^2 \quad (2.15)$$

where s controls the offset of the noise schedule. The authors set $s = 0.008$ as they found that $s = 0$ resulted in minuscule noise near $t = 0$ making it hard for the model to predict ϵ . From $\bar{\alpha}_t$ one can then compute $\beta_t = \min\left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999\right)$ with the min to prevent singularities. A comparison of the cosine and linear schedules can be seen in fig. 2.3.

2.3.2 Classifier and Classifier-Free Guidance (CFG)

In practice, it is often desirable to steer the generation process in order to output data belonging to a specific class, such as cats or dogs in the context of image generation, or incorporating some other information relevant for the output. This process can be framed within the context of probability theory as conditional generation, where the aim is to maximize the probability of the data, \mathbf{x} , given some conditioning signal, y , (e.g. a class label). According to Bayes' rule, the conditional probability can be expressed as:

$$p(\mathbf{x} | y) = \frac{p(\mathbf{x}, y)}{p(y)} \propto p(y | \mathbf{x})p(\mathbf{x}) \quad (2.16)$$

Hence, it is proportional to the joint probability of the data and label, which from basic probability theory is equal to the product of the unconditional probability of the data, $p(\mathbf{x})$, and the probability of the conditioning signal given the data, $p(y | \mathbf{x})$ (for labels; that the data belonging to the given class). In the DDPM paper Ho, Jain, and Abbeel 2020 establishes connection between diffusion models and Noise-Conditioned Score Networks (NCSN) and shows how the reverse diffusion process can be seen as a temporal discretized type of annealed Langevin dynamics given by the stochastic differential equation:

$$\frac{d}{dt} \log p(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t} s(\mathbf{x}_t, t) + \sqrt{1 - \bar{\alpha}_t} \mathbf{z}_t \quad (2.17)$$

where $\mathbf{z}_t \sim \mathcal{N}(0, 1)$ and $s(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ is referred to as the score function. It turns out that the denoising network can be used to approximate the score function:

$$s(\mathbf{x}_t, t) \approx -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \quad (2.18)$$

Returning to the factorization of the conditional probability, taking the gradient of the log of both sides reveals the relationship with the score function:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | y) = \nabla_{\mathbf{x}_t} \log p(y | \mathbf{x}_t) + \underbrace{\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)}_{s(\mathbf{x}_t, t)} \quad (2.19)$$

Dhariwal and A. Nichol 2021 trains a classifier, $f_\phi(y | \mathbf{x}_t) \approx p(y | \mathbf{x}_t)$, separately to predict the class label of noisy images. By using the reformulated denoising function:

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y | \mathbf{x}_t) \quad (2.20)$$

where s controls guidance strength, it is possible to guide the diffusion sampling e.g. in order to generate images of a specific category, hence the name classifier guidance.

In contrast to classifier guidance, an alternative method known as classifier-free guidance, proposed by Ho and Salimans 2021, eliminates the need for a separate guidance network. This method leverages the same generative model to perform conditional generation by incorporating the conditioning signal, y , during the training process. Specifically, the model is trained with the conditioning signal but occasionally discards it with a certain probability, p . This technique allows the model to learn both the conditional and unconditional generation processes.

During the forward diffusion process, the model is trained to predict the noise added to the data both conditionally, $\epsilon_\theta(\mathbf{x}_t, t, y)$, and unconditionally, $\epsilon_\theta(\mathbf{x}_t, t, y = \emptyset)$. The conditional prediction uses the conditioning signal y (e.g., class labels), while the unconditional prediction is made without any conditioning.

At inference time, classifier-free guidance combines these predictions to steer the generation process towards the desired condition. This is achieved by interpolating between the conditional and unconditional predictions:

$$\bar{\epsilon}_\theta(\mathbf{x}_t, t, y) = \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset) + s \cdot (\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset)) \quad (2.21)$$

Here, s is a scaling factor that controls the strength of the guidance. This formulation allows the model to flexibly adjust the influence of the conditioning signal on the generation process, effectively guiding the reverse diffusion towards samples that match the desired condition.

By leveraging the same network for both conditional and unconditional noise predictions, classifier-free guidance simplifies the implementation and improves the efficiency of the conditional generation process. This method has demonstrated significant improvements in generating high-quality samples that adhere to the specified conditions, making it a powerful alternative to traditional classifier guidance approaches.

2.4 Transformer architecture

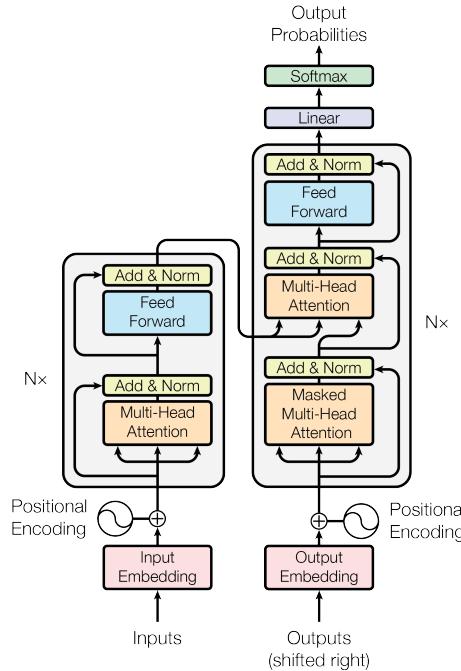


Figure 2.4: The transformer (Image from Vaswani et al. 2017.)

The transformer model, introduced by Vaswani et al. 2017, revolutionized natural language processing (NLP) and machine translation. This model was designed as an alternative to the traditional recurrent sequence models, such as the seq2seq model by Sutskever, Vinyals, and Le 2014, which had been widely used in NLP tasks. The seq2seq model (as illustrated in ??) encodes an input sequence into a context vector and then decodes the output sequence from this vector in a sequential manner. This approach often

results in an information bottleneck, particularly when dealing with long-range dependencies within the data.

In contrast, the transformer model utilizes an attention mechanism, enabling more effective communication between tokens without relying on sequential processing of hidden states. The architecture comprises an encoder and a decoder. The encoder uses multi-headed self-attention to create semantic representations for each token in the input sequence, allowing the model to process the entire context simultaneously.

The decoder then uses these representations through cross-attention and causal self-attention mechanisms. Cross-attention integrates information from the encoder's output, while causal self-attention ensures that the model only considers previously generated tokens by masking out future tokens in the training phase. At inference time, the output tokens are generated autoregressively.

One of the key strengths of the transformer model is its training efficiency, owing to the parallelizable nature of the attention mechanism. During the generation process, tokens are produced in an autoregressive fashion, with each generated token serving as context for the subsequent one.

The transformer's attention mechanism is based on scaled dot-product attention, defined by the following equation:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.22)$$

Q , K and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors.

For causal self-attention, the model masks future tokens for each key to prevent information leakage, ensuring the model attends only to past and present tokens during both training and inference.

Since the transformer architecture does not inherently encode positional information, Vaswani et al. introduced a sinusoidal positional encoding scheme. This scheme incorporates positional information into each element of the input sequence, defined as:

2.5 Human Motion Generation

Generating realistic human motion has several applications...

2.5.1 Human Motion Diffusion Model (MDM)

2.5.2 InterGen

3 Data

Dataset	# Sequences	Multi-person	Text descriptions	Scene object
HumanML3D		X	✓	X
InterHuman	7779	✓(exactly 2)	✓	X
Teton (processed)	27.656	✓	X	✓

Figure 3.1: Overview of different dataset properties.

3.1 HumanML3D

The HumanML3D dataset combines the HumanAct12 and AMASS datasets, integrating human motion captured using advanced motion capturing systems and converting the data to a unified parameterization. It covers a broad range of daily human activities, providing 14,616 motions in total, accompanied by 44,970 single-sentence descriptions. Each motion clip includes 3-4 descriptions, and the entire dataset amounts to 28.59 hours of recorded motion. Additionally, the data is augmented by mirroring all motions, with corresponding adjustments to descriptions, such as changing “clockwise” to “counterclockwise.”

3.2 InterHuman

The InterHuman dataset is a comprehensive, large-scale 3D dataset designed to capture human interactive motions involving two individuals. It includes approximately 107 million frames detailing a wide range of human interactions, from professional activities to daily social behaviors. Each motion sequence is paired with natural language annotations, totaling 23,337 descriptions, which provide context and detail for the captured interactions, enhancing the dataset’s utility for training and evaluating models.

7779 sequences resulting in 6.56h

3.3 Teton (raw) dataset



Figure 3.2: Examples from the Teton dataset

The Teton dataset contains approximately 50,000 image sequences hierarchically organized by hospital/carehome site, room and timestamp. Each sequence consists of 100 frames recorded at approximately 10 frames per second, resulting in a sequence duration of about 10 seconds and totaling roughly 140 hours of data.

Each person in the frame has been manually annotated with their class (person, staff, patient), center point, bounding box, keypoints and current action (e.g., "laying in bed", "standing on floor"). Using an image similarity threshold, the 100 frames are reduced to a smaller set of key frames. Each key frame is annotated with segmentation masks for people in the frame as well as the floor, walls and furniture (e.g., beds, sofas and chairs).

As the dataset is inherently two dimensional and does not contain 3D annotations such as poses or depth maps, we instead rely on pseudo-ground truth data derived from off-the-shelf models. In the case of human poses the HMR2.0 model by Goel et al. 2023 is initially run on the image sequences to predict a set of poses. Poses with high reprojection error are filtered out, and the remaining poses with low reprojection error are used as pseudo-ground truth.

For reconstructing depth maps the monocular metric depth estimation model, Depth Anything by Yang et al. 2024, trained on the indoor NYUv2 dataset (Nathan Silberman and Fergus 2012), is applied to the set of key frames.

4 Methods

4.1 3D scene reconstruction

Extracting SMPL poses

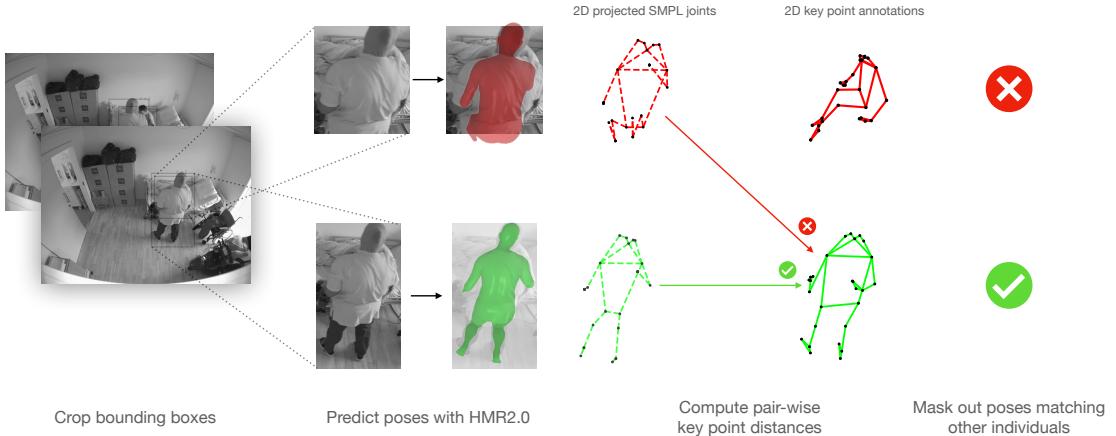


Figure 4.1: Illustration of pose extraction process with occlusion handling.

We leverage the pretrained HMR2.0 model from Goel et al. 2023 for pose extraction, applying it frame-wise to crops of each human bounding box in the frame. This approach ensures proper alignment with existing dataset annotations without the need for additional tracking. To handle potential occlusions that may cause duplicate predictions of the occluding pose, pairwise euclidean keypoint distances are computed for all edges in the bipartite graph of annotated 2D keypoints and SMPL joint 2D projections for individuals in the current frame. Poses with the minimum keypoint distance to other individuals' annotated keypoints are masked out.

Restoring scene depth

We combine both metric and relative variants of the Depth Anything model (Yang et al. 2024) to restore the depth of the scene. Using the low-resolution 384x512 metric depth predictions, $m_{1:K}$ (upsampled to match higher resolution disparity maps), we restore the scaling and offset from the higher resolution predicted disparity maps, $d_{1:K}$, by linearly fitting the disparity scaling α and offset β according to $\text{argmin } \alpha \cdot d_{1:K} + \beta - 1/m_{1:K}$ for all K key frames to get our upscaled metric depths, $m_{1:K}^* := 1/(\alpha \cdot d_{1:K} + \beta)$.

4.2 Human pose sequence optimization

Due to poses being predicted independently per frame, the combined pose sequences often lack temporal consistency, resulting in janky or erratic movements. To address this issue, we refine the pose sequences by optimizing for smooth trajectories across all $1 \dots T$ frames by minimizing a composite smooth loss objective assuming constant velocity:

$$\mathcal{L}_{\text{smooth}} = \sum_t \|\hat{\mathbf{j}}^{(t)} - \mathbf{j}^{(t)}\|_2 + \lambda_j \sum_t \mathcal{R}_j^{(t)} \quad (4.1)$$

where $\hat{\mathbf{j}}^{(t)} = \mathbf{j}^{(t-1)} + \mathbf{v}^{(t-1)}$ is the predicted joint location at time t assuming constant velocity $\mathbf{v}^{(t)} = \mathbf{j}^{(t)} - \mathbf{j}^{(t-1)}$, and $\mathcal{R}_{\bar{\mathbf{j}}}^{(t)} = \|\mathbf{j}^{(t)} - \bar{\mathbf{j}}^{(t)}\|_2$ a regularization term preventing the optimized location $\mathbf{j}^{(t)}$ to not diverge too far from the initially predicted joint location $\bar{\mathbf{j}}^{(t)}$.

4.3 Unification of datasets

Before concatenating the HumanML3D, InterHuman and our own Teton dataset we make sure they all conform to the same coordinate system and scaling by implicitly representing the floor as the XY-plane with positive Z indicating the upwards direction using metric units for coordinates. This removes the need to explicitly condition the model on the floor plane possibly simplifying the learning task for more efficient training.

Aligning floor with XY-plane

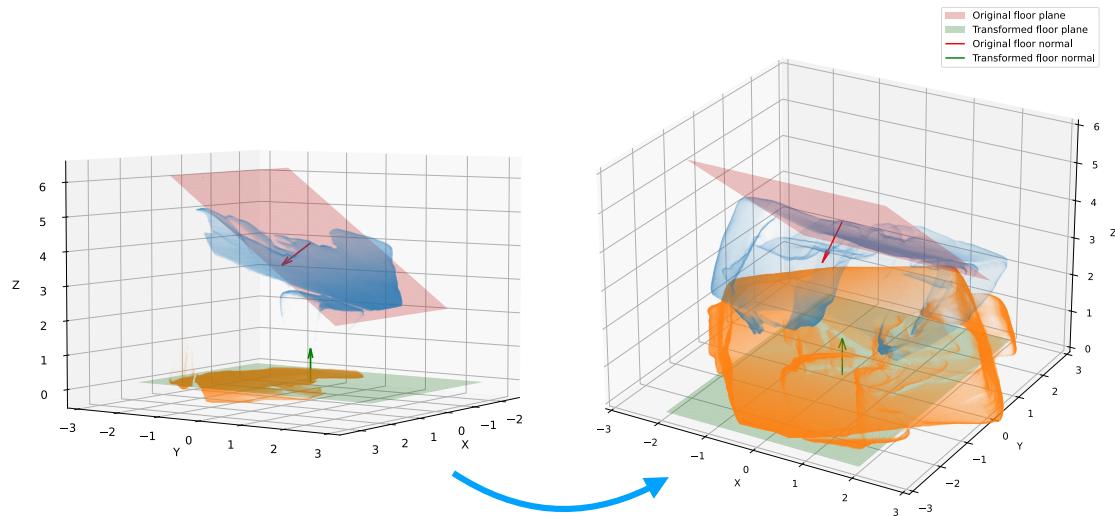


Figure 4.2: Translation and rotation of room point cloud aligning the floor with the XY-plane.

Using the floor segmentation masks we isolate the floor point cloud and use the outlier robust RANSAC regressor with inline threshold of 10cm to fit the plane equation:

$$z = \beta_z + \beta_x x + \beta_y y \quad (4.2)$$

To compute the rotation matrix that aligns the normal vector $\hat{\mathbf{n}}_{\text{floor}} = \mathbf{n}_{\text{floor}} / \|\mathbf{n}_{\text{floor}}\|$ (where $\mathbf{n}_{\text{floor}} = (0, 1, b_y)^T \times (1, 0, b_x)^T$) of the floor plane in the camera coordinate system with the normal vector $\mathbf{n}_{XY} = (0, 0, 1)$ of the XY-plane, we can utilize Rodrigues' rotation formula.

First, we construct the skew-symmetric matrix \mathbf{K} from the components of the cross product vector $\mathbf{v} = \hat{\mathbf{n}}_{\text{floor}} \times \mathbf{n}_{XY}$. The skew-symmetric matrix \mathbf{K} is defined as:

$$\mathbf{K} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \quad (4.3)$$

Next, we compute the rotation matrix \mathbf{R} using Rodrigues' rotation formula. The formula incorporates the identity matrix \mathbf{I} , the skew-symmetric matrix \mathbf{K} , and a scaling term based on the angle between the vectors. The resulting rotation matrix is given by:

$$\mathbf{R} = \mathbf{I} + \mathbf{K} + \mathbf{K}^2 \left(\frac{1 - c}{s^2} \right) \quad (4.4)$$

Here, \mathbf{I} is the identity matrix, c is the dot product of \mathbf{a} and \mathbf{b} , and s is the norm of the cross product vector \mathbf{v} . This formula ensures that the rotation matrix \mathbf{R} not only aligns \mathbf{a} with \mathbf{b} but also preserves the orthogonality and orientation of the coordinate system.

Finally, we transform the point cloud by translating it to align the floor plane intercept with the origin, followed by rotating it using the computed rotation matrix:

$$\mathbf{p}^* = \mathbf{R} (\mathbf{p} - (0, 0, b_z)^T) \quad (4.5)$$

In a similar fashion, we transform our poses by applying eq. (4.5) to our pose root joint translations. Since our joint rotations are relative to the root global orientation, \mathbf{O} , we first compute its inverse global-to-local rotation matrix, \mathbf{O}^{-1} . After applying the new rotation matrix \mathbf{R} , we then invert the result to convert back from local-to-global orientation:

$$\mathbf{O}^* = (\mathbf{R}\mathbf{O}^{-1})^{-1} \quad (4.6)$$

The resulting transformation of the point cloud and poses in aligning the floor with the XY-plane is illustrated in fig. 4.2.

4.4 Motion and scene representation

We follow the work by Liang et al. 2024 and represent each pose in the motion sequence using their proposed non-canonical representation, but disregard the binary foot-ground contact features:

$$\mathbf{x}_t = [\mathbf{j}_g^p, \mathbf{j}_g^v, \mathbf{j}^r] \quad (4.7)$$

where \mathbf{j}_g^p are the global joint positions, \mathbf{j}_g^v the global joint velocities and \mathbf{j}^r the relative SMPL 6D joint rotations.

4.5 Diffusion model

otion data for each person is flattened into a single array of poses. Each pose in the sequence is embedded using a linear layer. To embed the motion timestamp, we add positional encoding to the latent variables (z_t).

Identity encoding is achieved by randomly selecting (without replacement) a learned ID embedding for each person and adding it to each pose embedding. Additionally, class information (such as medical staff, patients, etc.) is encoded by adding a learned class embedding.

For textual descriptions, we use the CLIP model to encode the text, and the resulting output is embedded using a linear layer. This embedded text is then prepended to the input sequence for the transformer.

Object point clouds are processed by clustering the coordinates into groups of 64, which are then embedded using a linear layer. These embeddings are integrated into the model to provide spatial context.

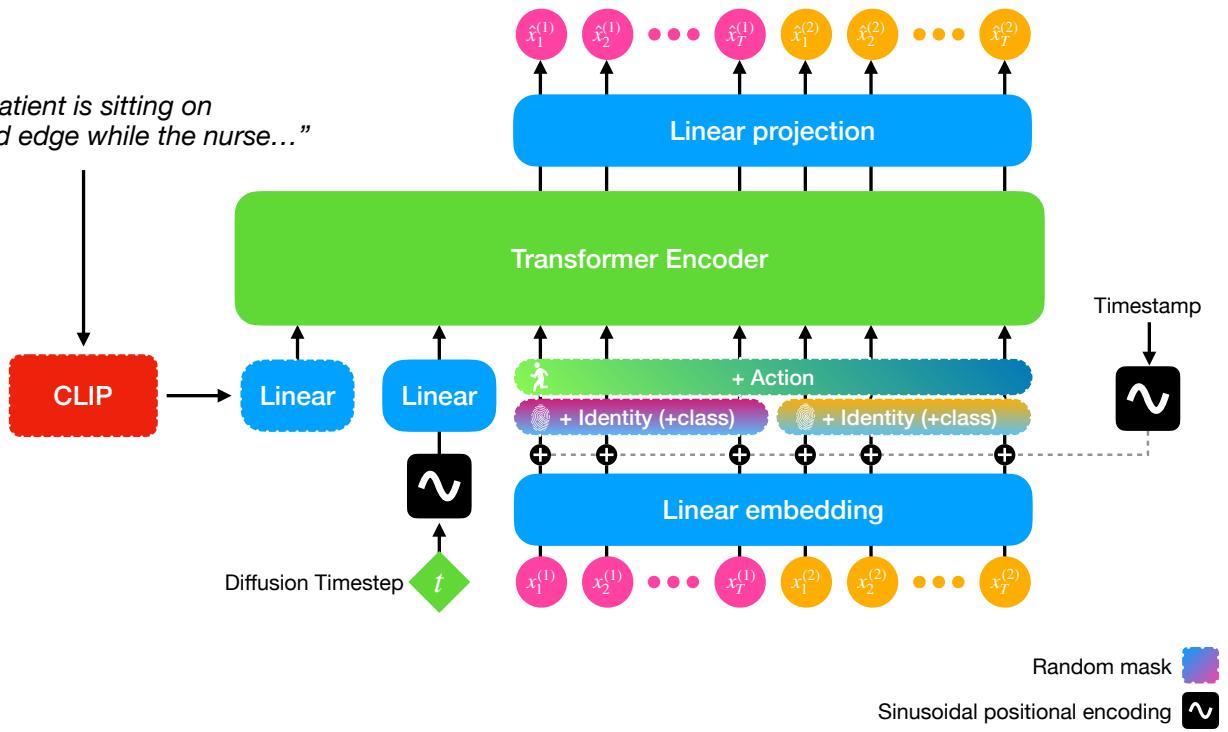


Figure 4.3: Diffusion architecture

5 Results

6 Discussion & Conclusion

Bibliography

- Loper, Matthew et al. (Oct. 2015). "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6, 248:1–248:16. DOI: 10.1145/2816795.2818013.
- Romero, Javier, Dimitrios Tzionas, and Michael J. Black (Nov. 2017). "Embodied Hands: Modeling and Capturing Hands and Bodies Together". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*. 245:1–245:17 36.6.
- Pavlakos, Georgios et al. (2019). "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.
- Bogo, Federica et al. (Oct. 2016). "Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image". In: *Computer Vision – ECCV 2016*. Lecture Notes in Computer Science. Springer International Publishing.
- Kanazawa, Angjoo et al. (2018). "End-to-end Recovery of Human Shape and Pose". In: *Computer Vision and Pattern Recognition (CVPR)*.
- Kocabas, Muhammed, Nikos Athanasiou, and Michael J. Black (June 2020). "VIBE: Video Inference for Human Body Pose and Shape Estimation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kolotouros, Nikos et al. (2021). "Probabilistic Modeling for Human Mesh Recovery". In: *ICCV*.
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33, pp. 6840–6851.
- Nichol, Alexander Quinn and Prafulla Dhariwal (July 2021). "Improved Denoising Diffusion Probabilistic Models". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8162–8171. URL: <https://proceedings.mlr.press/v139/nichol21a.html>.
- Dhariwal, Prafulla and Alexander Nichol (2021). "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., pp. 8780–8794. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.
- Ho, Jonathan and Tim Salimans (2021). "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. URL: <https://openreview.net/forum?id=qw8AKxfYbl>.
- Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb0d053c1c4a845aa-Paper.pdf.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Proceedings of the 27th International Conference on Neural*

Information Processing Systems - Volume 2. NIPS'14. Montreal, Canada: MIT Press, pp. 3104–3112.

Goel, Shubham et al. (2023). “Humans in 4D: Reconstructing and Tracking Humans with Transformers”. In: *International Conference on Computer Vision (ICCV)*.

Yang, Lihe et al. (2024). “Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data”. In: *CVPR*.

Nathan Silberman Derek Hoiem, Pushmeet Kohli and Rob Fergus (2012). “Indoor Segmentation and Support Inference from RGBD Images”. In: *ECCV*.

Liang, Han et al. (2024). “Intergen: Diffusion-based multi-human motion generation under complex interactions”. In: *International Journal of Computer Vision*, pp. 1–21.

A Appendices

A.1 Derivation of posterior distribution

$$\begin{aligned}
q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} \\
\Leftrightarrow \log q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= \log q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) + \log q(\mathbf{x}_{t-1} | \mathbf{x}_0) - \log q(\mathbf{x}_t | \mathbf{x}_0) \\
&= \frac{1}{2} \left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} + \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right) + K \\
&= \frac{1}{2} \left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t} \mathbf{x}_t \mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_{t-1} \mathbf{x}_0 + \bar{\alpha}_{t-1} \mathbf{x}_0^2}{1 - \bar{\alpha}_{t-1}} + C(x_t, x_0) \right) +
\end{aligned}$$

We collect all terms for \mathbf{x}_{t-1} and \mathbf{x}_{t-1}^2 .

A.2 L_{t-1} closed form derivation

The closed form of the KL-divergence between two Gaussians $\mathcal{N}_0(\mu_0, \Sigma_0)$, $\mathcal{N}_1(\mu_1, \Sigma_1)$ is given as:

$$D_{\text{KL}}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2} \left(\text{tr}(\Sigma_1^{-1} \Sigma_0) - k + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) + \ln \frac{\det \Sigma_1}{\det \Sigma_0} \right) \quad (\text{A.1})$$

Inserting $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$ and $p(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$ yields:

$$D_{\text{KL}}(q \| p) = \frac{1}{2} \left(k \frac{\tilde{\beta}_t}{\sigma_t^2} - k + \frac{1}{\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 + \ln(\sigma_t^2) - \ln(\tilde{\beta}_t) \right) \quad (\text{A.2})$$

Letting $\sigma_t^2 = \tilde{\beta}_t$ ($= \beta_t = \frac{1-\bar{\alpha}_t}{1-\bar{\alpha}_{t-1}} \tilde{\beta}_t$) simplifies the above to:

$$D_{\text{KL}}(q \| p) = \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 (+ C_t) \quad (\text{A.3})$$

where C_t is a constant depending only on the noise schedule.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk