SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Dipartimento di Ingegneria informatica, automatica e gestionale
PhD in Data Science

Thesis For The Degree Of Doctor Of Philosophy

Decoding Human Dynamics: Explorations in Motion Forecasting, Social Navigation, and Egocentric Perception

Thesis Advisor
Prof. Fabio Galasso

Correlatori
Prof. Indro Spinelli
Dr. Alessandro Nicolosi

Candidate
Luca Scofano
1762509

Academic Year MMXXI-MMXXIV (XXXVII cycle)

Abstract

Human dynamics—how individuals move, interact, and perceive their environment—pose significant challenges for theoretical understanding and practical implementation in robotics, human-computer interaction, and behavior analysis. Accurate models addressing these challenges are essential for developing intelligent systems capable of effectively collaborating with or understanding humans. This Ph.D. thesis investigates key aspects of human dynamics through Motion Forecasting, Social Navigation, and Egocentric Perception.

In Motion Forecasting, we explore both two-body pose prediction and global human motion prediction. We present best practices for improving collaborative motion prediction [261]. We introduce a staged, contact-aware framework for global human motion forecasting [282] that predicts human movements within broader environmental contexts. Our model surpasses existing methods by incorporating contact points and staged motion, enabling more accurate human pose and trajectory predictions.

In the context of social dynamics, we investigate the impact of latent variables on forecasting human interactions, especially in team-based settings. Introducing a role-based approach demonstrates that understanding these latent social roles can significantly improve trajectory prediction in multi-agent systems [281]. This concept extends to Social Navigation [280], where a robot's trajectory planning must account for human movement and be processed in real-time. Human dynamics are incorporated into the robot's reinforcement learning path-planning framework via a social dynamics module. This module distills human trajectories into latent codes, which serve as contextual input for the robot's policy model.

We also address challenges in Egocentric Perception and Mistake Detection. By developing a novel method, we tackle the need for real-time online detection of procedural mistakes from egocentric video streams. Our approach, PREGO [93], introduces an innovative model that recognizes current actions and predicts future ones to identify discrepancies and detect mistakes. We also present an extension of the latter, which offers an in-depth analysis and enhances the framework with an Automatic Chain of Thought mechanism. This addition improves the model's reasoning capabilities, enabling more nuanced error detection. Additionally, we contribute a framework for estimating social interactions and human meshes using egocentric video, improving pose estimation accuracy by incorporating wearer-interactee interactions.

Beyond direct applications to human dynamics, this thesis includes a contribution to Topological Deep Learning. We contributed to a technical paper introducing the first Python framework for Topological Deep Learning [119], offering new tools for researchers exploring machine learning on non-Euclidean data structures.

Overall, this thesis explores human motion forecasting, social interaction modeling, and egocentric perception while advancing methodologies in machine learning. The insights and tools developed contribute to understanding human behavior and pave the way for further research in intelligent systems and interactive environments.

Keywords: Motion Forecasting , Social Navigation, Egocentric Perception, Procedural Mistake, Topological Deep Learning, Mesh Estimation, Large Language Models (LLMs)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Predicting human motion and social interactions, a crucial aspect of human dynamics, is necessary across diverse domains, including robotics, sports analytics, and virtual reality. The inherent complexity of forecasting human behavior stems from many influential factors, encompassing environmental contexts, social dynamics, and individual variability. Conventional modeling approaches frequently struggle to adequately capture these intricate dynamics, facing challenges in the representation of multi-agent interactions and the ability to adapt in real-time within dynamic settings. These limitations underscore the need for more sophisticated methodologies to address human behavioral prediction's nuanced and multifaceted nature. Furthermore, egocentric perception, where the viewpoint is limited to what the individual directly sees, introduces additional challenges in procedural task monitoring, mistake detection, and social interaction understanding. Thus, there is a pressing need for advanced models that can predict human motion and social interactions in various contexts, identify errors, and adapt to dynamic environmental changes.

This thesis explores three core areas: motion forecasting, social navigation, and egocentric perception. It explores innovative approaches that leverage deep learning, role-based interaction models, and social dynamics to enhance the prediction and understanding of human behavior in complex environments. The significance of these works lies in their potential to bridge the gap between theoretical models of human behavior and practical applications in fields where understanding and predicting human motion is critical. Human dynamics, whether in social environments, collaborative tasks, or individual motion patterns, are central to advancing human-computer interaction, robotics, virtual reality, and team-based systems. Accurate motion forecasting allows for improved interaction models in sports analytics, where predicting players' future actions can provide a strategic edge. Similarly, anticipating human movements in robotics and human-robot collaboration fosters smoother and more intuitive interactions, enhancing safety and efficiency in shared environments. Furthermore, understanding procedural mistakes in egocentric videos is invaluable in healthcare and manufacturing, where real-time mistake detection can prevent costly errors.

Chapters 2 and 3 expand upon these ideas by working on social and environmental dynamics, moving beyond the limitations of isolated motion forecasting. Significant strides have been made in human motion forecasting, but challenges remain, especially when incorporating social interactions and environmental contexts.

Collaborative human motion forecasting, which focuses on predicting the future poses of multiple interacting people, is a particularly underexplored area. By considering interactions between individuals, we aim

to improve forecasting accuracy. While single-person techniques have yielded valuable insights, our work shows that not all approaches transfer well to multi-person scenarios, especially when dealing with complex social interactions [261].

Moreover, scene-aware global human motion forecasting, which considers the environmental context, has emerged as a critical field. It plays a pivotal role in applications like virtual reality, robotics, and sports, where predicting human trajectories and poses within a 3D environment is essential. To address these challenges, STAG [282] (STAGed contact-aware global human motion forecasting) models social interactions and incorporates environmental elements such as contact points within the scene. This staged method improves trajectory and pose prediction by first identifying fundamental scene interactions, then modeling the global human trajectory, and finally refining fine-grained motion details. Through this, we significantly improve state-of-the-art methods, particularly in environments with rich scene interactions.

Chapters 4, 5, and 6 expand on social interaction and modeling agent relationships. Social interaction is critical in human dynamics, particularly in environments where individuals' actions are tightly coupled with those around them. In the context of motion forecasting, these interactions are complex and are often guided by social norms, roles, and shared objectives. Team-based scenarios, such as sports, exemplify this complexity, where their teammates and opponents influence each player's movements. Recent advances in role-based forecasting models have highlighted the importance of understanding these interactions, where each participant assumes a specific function that significantly impacts their future actions. Role-based approaches, such as *RolFor* [281], demonstrate that leveraging latent roles through graph-based models can enhance trajectory prediction by optimizing the relationships between roles. These insights underscore the necessity of incorporating social roles when forecasting in multi-agent systems.

Beyond team sports, social interaction is also essential in human-robot collaboration, particularly in shared environments. In *Social Navigation*, the agent must balance assisting the human while avoiding interference with their movements. Human trajectories provide crucial cues for navigation, but these are often partially observable from the robot's egocentric perspective, presenting a significant computational challenge. Social Dynamics Adaptation [280] (SDA) leverages reinforcement learning to infer social dynamics from limited observational data. SDA enables real-time adaptation in social navigation tasks by encoding human trajectories and using historical state-action pairs, achieving state-of-the-art performance in environments such as Habitat 3.0.

We have also contributed to implementing *TopoModelx* [119], a framework for Topological Deep Learning (TDL). Initially, we explored this framework to represent relationships between agents that extend beyond pairwise interactions to include more complex, multi-agent dynamics. While traditional deep learning operates primarily on Euclidean domains, advancements in geometric deep learning (GDL) and TDL have expanded the scope of machine learning to non-Euclidean data structures, such as hypergraphs and simplicial complexes. These higher-order structures provide new ways to model complex interactions in human dynamics. However, practical implementation remains challenging despite their potential due to the lack of accessible software libraries.

Chapters 7, 8, and 9 explore Egocentric Perception within the context of social and environmental interactions. This perspective, captured from the viewpoint of the camera wearer, offers unique insights into human behavior, proving particularly valuable for applications in virtual reality (VR), augmented reality (AR), and real-time behavior analysis. A significant challenge in this field is the limited visibility of the

wearer's body, as head-mounted cameras often fail to capture substantial portions of the body. While recent advancements have leveraged scene context and ego-motion to estimate 3D body poses, many approaches overlook the crucial role of social interactions in this process.

Our proposed framework, *SEE-ME* (**S**ocial **E**gocentric **E**stimation of body **ME**shes), represents a pioneering effort to incorporate social interaction cues into the egocentric mesh estimation process. By conditioning the latent diffusion model on the scene and the interactions between the camera wearer and other agents, *SEE-ME* captures essential relational dynamics that significantly enhance pose estimation. We quantify how factors such as interpersonal distance and gaze direction influence the quality of ego-mesh estimation, ultimately achieving a substantial reduction in pose estimation errors.

Identifying procedural errors from egocentric videos in real-time is crucial in fields like manufacturing and healthcare, where immediate detection can prevent costly outcomes. Procedural errors are inherently open-set, meaning novel failures may occur, necessitating classifiers trained on correctly performed procedures. To address this challenge, we introduce *PREGO*, the first online one-class classification model for mistake detection in **PR**ocedural **EGO**centric videos [93]. PREGO combines an online action recognition module to model ongoing tasks and a symbolic reasoning module to predict the next steps. Mistake detection is achieved by comparing the recognized action with the anticipated action. We evaluate PREGO on two adapted egocentric video datasets—*Assembly* and *EPIC*—creating the *AssemblyO* and *EPICO* datasets to establish benchmarks for online procedural mistake detection.

Extending upon PREGO, we further explore the realm of real-time procedural error identification in egocentric videos. In this extended approach, we emphasize the role of LLMs in action anticipation, conducting a thorough analysis of their potential in predicting procedural actions. Our research explores various prompting schemes to effectively guide LLMs and investigates optimal strategies for aggregating outputs from both branches to create a robust mistake-detection system. Additionally, we address the challenges associated with per-frame evaluations, mainly focusing on the need for accurate predictions in dynamic settings. Our experimental results highlight the challenges and opportunities of integrating dual-branch architectures with LLM-based anticipation for open-set procedural mistake detection. This approach achieves state-of-the-art performance, demonstrating its efficacy in online applications and opening new avenues for future research in egocentric perception and procedural error detection.

A significant direction for future research lies in expanding the scope of egocentric perception for video understanding. Specifically, we aim to leverage the capabilities of Vision-Language Models (VLMs) and Large Language Models (LLMs) to enhance the analysis of egocentric videos. With their ability to integrate visual and textual data, these models offer promising avenues for better understanding complex human behavior from a first-person perspective. We aim to push the boundaries of procedural error detection, human motion forecasting, and immersive applications such as augmented and virtual reality by utilizing VLMs for scene interpretation and LLMs for action anticipation and reasoning.

Additionally, we plan to explore new methods for modeling social interaction in human motion forecasting and social navigation. One possible approach is to develop graph-based models that capture multi-agent interactions more effectively by incorporating higher-order relational dynamics between agents. This could extend current topological representations, such as those explored in *TopoModelx*, to better model group dynamics and collective behaviors. Furthermore, integrating reinforcement learning techniques with interaction models could enable agents to adapt more fluidly to changing social contexts in navigation tasks.

Chapter 2

Best Practices for 2-Body Pose Forecasting

2.1 Introduction

Human 2-body pose forecasting predicts the future body poses of two people in interaction jointly. The task is relevant to long-term pose tracking [12], to understanding interacting pairs in sports such as dancing [108] and to the collaborative assembly in industry [78, 163], towards human-robot collaboration [274]. Considering the concurrent prediction of two bodies helps in cases where the people act synergistically. However, this task has remained mostly unexplored and limited to the dataset of [108][1]. Also, this differs from the related task of human trajectory forecasting, where social interaction has been key to most recent progress [165, 228, 230, 350].

There has been vast progress in single human pose forecasting [69, 112, 208], which has not transferred to the 2-body counterpart. Single-person techniques [23, 66, 110] tested on two-people data underperform, which is unsurprising, as they neglect the body-body motion correlations [108]. This motivates the current work, where the most recent modeling advancements are analyzed and integrated. Here, we refer to the best and complementary modeling aspects as *best practices*, which we leverage to bootstrap research on 2-body forecasting.

We propose a systematic analysis of single-person skeleton-based best practices by considering three processing stages (cf. Fig. 7.1): input representation, encoding, and decoding. For the first stage, we identify Discrete Cosine Transform (DCT) [35, 108, 212, 214, 216] as an asset to cope with the periodic body movements. For the second stage, we set to encode the body kinematics by Graph Convolutional Networks (GCN), which power the vast majority of most recent techniques [108, 112, 212, 214, 216, 292] and subsume general MLP-based formulations [112]. Here we evaluate as best practices the separability of space and time dynamics [292], the learnable adjacencies versus kinematic trees [338], attention [108], and hierarchical body representations [69]. Finally, for the third stage, we contrast the widely-adopted [208, 274, 292] decoding with convolutional networks (*a.k.a.* Temporal Convolutional Network–TCN [18]) with the simpler Fully Connected (FC) layers [112].

We propose a novel initialization technique for the learnable GCN parameters in the encoder. A large body of literature asserts the importance of initialization for performance, convergence speed, and robustness, and theory has been devised for MLP [105] and ConvNets [126, 166]. Up until recently, there has been a limited necessity for ad-hoc GCN initialization theories since techniques leveraged mainly shallow net-

[1] Beyond [108], another multi-body dataset has been introduced by [90], but annotations are only available for one individual at the time of writing.

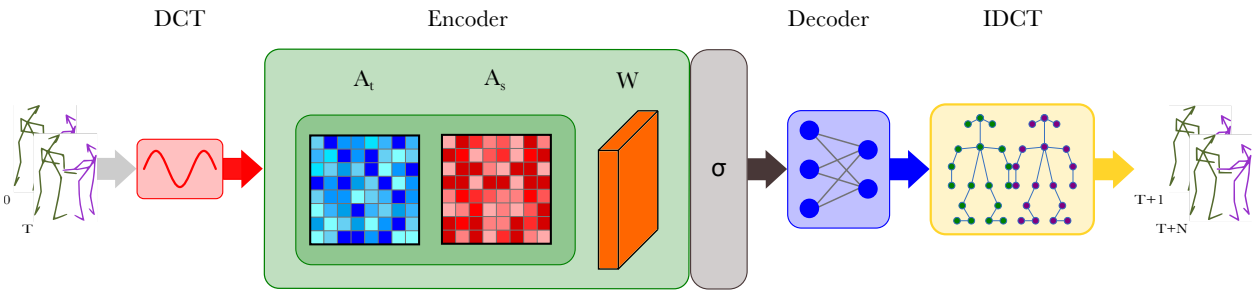


Figure 2.1: The general architecture of a 2-body pose forecasting model employing best practices. First, 3D joint coordinates are mapped to frequencies by DCT coefficients, a best input representation practice. Secondly, body kinematics are encoded by layers of a GCN $\sigma(A_s A_t X W)$, with separable space-time adjacency matrices $\sigma(A_t, A_s)$, learned unconstrainedly, upon our proposed parameter initialization. Thirdly, the FC-based decoder outputs future poses for the two people, mapped to 3D coordinates with inverse-DCT (IDCT).

works with fixed graphs structures (e.g., the people neighbors[17, 180, 306], the kinematic tree [69, 338]) or spectral normalizations [152, 155]. Since we determine that unconstrained learnable GCN affinities are best practices, we also develop a novel theory (See Sec. 2.3.4) and experimental study (See Sec. 2.4.3) on the initialization of GCN parameters.

Integrating the selected best practices into a 2-body pose forecasting model yields a large-margin improvement of 21.9% *wrt* the state-of-the-art (SoA) on the most recent ExPI dataset [108]. The best-practice model is also 5 times faster than the current best technique and only has 2% of its parameters. The improvement is similarly consistent in generalization tests, across unseen actions with an overall improvement of 14.7% (cf. Table 2.2) and 14.2% for unseen actors (cf. Table 2.3). And the same best-practice model performs on par (cf. Table 2.4) with the leading single-person pose forecasting techniques on the established Human3.6M dataset [139], without any hyper-parameter tuning. The novel initialization, proposed for the unconstrained learning of GCN affinities, contributes an average performance improvement of 3.5%, and it increases stability, as it reduces the long-term forecasting performance variance by (at least) a factor of 2.

The main contributions are summarized as follows:

- We thoroughly evaluate all leading best practices from single-person pose forecasting and bootstrap research on the 2-body task counterpart;
- We propose a novel theory and experimental study on the initialization of GCNs, applying to unconstrained learnable affinities, accounting for an increase in performance of 3.5% and a 2-fold increase in stability;
- On a closed-set dataset configuration, the best-practice model outperforms the 2-body forecasting SoA by a large margin of 21.9% while employing 2% of the parameters and running 5 times faster.

2.2 Related Work

Here we review related work from the field of human pose forecasting, specifically approaches of spatio-temporal pose modeling and hierarchical body representations. Additionally, we review relevant literature from initialization and multi-agent trajectory forecasting.

Human pose forecasting. Established methodologies for (single) human pose forecasting include Temporal Convolutional Network [180], Recurrent Neural Network [94, 208, 214, 316] and Transformer Net-

works [8, 108]. The MLP-based approach of [112] holds SoA performance.

Graph Convolutional Networks (GCN) [155, 338] are most popular on the task [69, 185, 274], due to their simplicity and effectiveness. GCNs model the kinematic body part interactions by a plain adjacency matrix at a fraction of the parameters of the otherwise required attention mechanism [108, 212]. In this realm, [212] integrates DCT to consider motion frequency; [69, 186] adopt multi-scale hierarchical representations, grouping joints to model relations between coarser body parts; [274, 292] factorize the spatial and temporal adjacency matrices, and they propose to learn them, unconstrainedly, without kinematic tree priors nor spectral normalization.

As we know, the only work that addresses multi-body pose forecasting is [319]. However, they utilize datasets that do not contain highly interactive actions. For comparison, we ran their model with our setup as a comparison with our proposed method (See Tab. 2.1). By contrast, for the task of 2-body pose forecasting, [108] provides the solely-available dataset (ExPI) and the only 2-body-specific technique, adaptation of [212] with cross-person attention. Not surprisingly, this outperforms single-person techniques.

Initialization. A proper initialization improves performance and accelerates convergence [168], limiting vanishing and exploding gradients [105, 126]. Techniques have been concerned with initializing the weights of linear [105] and convolutional [126, 166, 223] layers, generalizing from hyperbolic (tanh) to rectified-linear unit (ReLU) activations. For GCNs, spectral techniques [155, 188, 363] rely on the spectral normalization of the adjacency matrix to elude vanishing and exploding gradients, while spatial techniques [17] resort to degree-normalized transition matrices, derived from the adjacency. In all prior study cases, the graph connectivity is given. To the best of our knowledge, this work presents the first theoretical and empirical analysis of GCN initialization in the case of unconstrained learnable graph connectivity and edge weights.

Multi-agent trajectory forecasting. For trajectory forecasting, employed techniques include attention [135, 165, 350] and graph-based modeling [182, 230, 288]. The multi-agent relations may parallel the joint-joint interaction. However, nodes in a graph of joints have a fixed cardinality and a semantic meaning (head, torso, hand, etc.), which does not apply to general agent-agent graphs. Notably, best trajectory forecasting techniques model the agent-agent interaction [135, 165, 182, 230, 288, 350], which aligns with the motivation of this work, to forecast the poses of people jointly.

2.3 Methodology

We explore the best models for single-body pose forecasting [69, 112, 212, 214, 292] and select best practices for the 2-body task. We group and evaluate practices in three processing stages (cf. Fig. 7.1): 1) input representation (Sec. 2.3.1); 2) encoding of the body kinematics in the observed frames (Sec. 2.3.2); 3) decoding of the future poses (Sec. 2.3.3). In Sec. 2.3.4, we provide a theory for the proposed unconstrained-GCN initialization. To facilitate reading, we mark with a green check ✓ the selected *best practices* upon evaluation, cf. Sec. 8.5.2.

Problem formalization. Across T frames, we observe the motion of two human bodies $\mathcal{B}^1$ and $\mathcal{B}^2$, each consisting of J three-dimensional joints. At time t, the 3D body pose of each person is given by corresponding tensors $\mathcal{B}^1_t, \mathcal{B}^2_t \in \mathbb{R}^{3\times J}$. We define the concatenation of two bodies at timeframe t as $\mathbf{x}_t = \mathcal{B}^1_t || \mathcal{B}^2_t$, thus

the observed motion history in T frames is $\mathcal{X}_{in} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{T\times 3\times 2J}$. Our goal is to predict the future N frames' poses $\mathcal{X}_{out} = [\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+N}] \in \mathbb{R}^{N\times 3\times 2J}$.

Preliminaries on the encoder-decoder baseline. We adopt an encoder-decoder architecture [274, 292], and following [338, 360], we encode the observed body parts and their kinematic interaction through a GCN, defined as

$$\mathbf{Y} = \sigma(A\mathbf{X}W), \tag{2.1}$$

where A is the adjacency matrix, W learnable weights and σ an activation function. Other encodings such as RNNs [58, 77] and MLPs [112] have been proposed, whereas we opt for a graph-based model to exploit the non-euclidean nature of graphs. As a decoder, we examine either a single fully connected layer as in [112] or a convolutional architecture [208, 292].

2.3.1 Input Representation

Most recent techniques [7, 112, 212, 214] use Discrete Cosine Transform (DCT) to represent 3D coordinate input as frequencies, under the claim that this captures the dynamic patterns of moving people better.

Frequency encoding ✓
Given the j-th body joint and the t-th timeframe we define the i-th DCT coefficient as

$$\mathcal{F}(\mathcal{X}^{in})_{j,i} = \sqrt{\frac{2}{T}} \sum_{t=1}^{T} x_{j,t} \frac{1}{\sqrt{1+\delta_{i1}}} \cos(\alpha) \tag{2.2}$$

$$\alpha = \frac{\pi}{2T}(2t-1)(i-1), \tag{2.3}$$

where the Kronecker delta function $\delta_{ij} \in 0, 1$ has null value if $i \neq j$ and 1 otherwise. After inference, frequencies are remapped to the pose representation via the inverse DCT decoding function $\mathcal{F}^{-1}$. Previous works [212, 216] truncate high frequencies to avoid jittery motion; we consider the impact of the number of retained DCT coefficients and discover that employing all of them yields the best performances. Studies on the impact of DCT coefficients are shown in Sec. 8.5.2 and Table 2.5.

2.3.2 Encoding Best Practices

Best-performing single-pose forecasting GCN encoders have considered two main aspects: the space-time separability of adjacency weight matrices and learning the body kinematic graph connectivity and weights. We detail these two aspects and empirically compare them in Table 2.5. Furthermore, we also consider hierarchical representations of the skeleton proposed by [69], but this is not a best practice, as we determine experimentally. Nor is it a good practice to add attention, as we discuss in this section and quantitatively evaluate in the next.

Space-time separability ✓
Each graph's intra-relations are expressed through a GCN-based framework that encodes the spatiotemporal motion and the relationships between keypoints in one's skeleton [292, 338]. Tensor $\mathbf{X} \in \mathbb{R}^{T\times 2J\times C}$ represents a couple's skeleton pose and motion, adjacency matrices $A_s \in \mathbb{R}^{T\times 2J\times 2J}$ and $A_t \in \mathbb{R}^{2J\times T\times T}$ are responsible for learning spatial and temporal interactions respectively, as in [292]. Matrices are fully

learnable, no kinematic tree is used, and the model is free to grasp the relation between body joints. Thus, this module is formulated as follows:

$$\mathbf{Y} = \sigma(A_s A_t \mathbf{X} W), \tag{2.4}$$

where σ is an activation function and $W \in \mathbb{R}^{C \times C'}$ is a tensor of learnable weights defined as a convolution with kernel dimension $k = 1$. Thus, it is conceptually similar to a fully connected layer. However, unlike the MLP design of [112], GCN shares the weights of W across all channels.

Learning the graph connectivity and weights ✓

Some works [69, 338] use inductive biases based on the human body, such as kinematic trees or specifically-devised connectivity weights. In contrast, others learn the graph adding a constraint on the optimization by spectral normalization [153]. Instead, we follow what is done in the most recent work [292]: unconstrained optimization of graph edges and weights i.e., we set A_{st} for nonseparable GCN and A_s, A_t in case of space-time separable GCN as a fully learnable matrix. This is effectively a best practice, experimentally proven in Table 2.5.

Attention

A GCN model equipped with attention is also known as a Graph Attention Network (GAT) [314]. In a GAT, attention re-defines the adjacency matrix terms as a function of the node embeddings. We employ attention to encode the relation between the two actor embeddings B_h^1 and B_h^2:

$$\mathcal{B}_{\mathbf{h}}^{\mathbf{1}} = \mathcal{B}^1 W_1, \mathcal{B}_{\mathbf{h}}^{\mathbf{2}} = \mathcal{B}^2 W_2, \tag{2.5}$$

Where $\mathcal{B}^1, \mathcal{B}^2 \in \mathbb{R}^{T \times J \times C}$ and $W_1, W_2 \in \mathbb{R}^{C \times C}$ are learnable weights to map features in a high-dimensional space. We use these features to calculate attention weights as follows:

$$\eta = softmax\Big(\sigma(\mathcal{B}_h^1 W_3 || (\mathcal{B}_h^2 W_4)^{\mathsf{T}})\Big), \tag{2.6}$$

Where $\mathcal{B}_h^1, \mathcal{B}_h^2 \in \mathbb{R}^{T \times J \times C}$, $W_3, W_4 \in \mathbb{R}^{C \times 1}$ and σ is a LeakyRelu activation function. We apply softmax to get attention weights $\eta \in \mathbb{R}^{T \times n \times m}$ constituiting n joints in $\mathcal{B}^1$ and m joints in $\mathcal{B}^2$ and reweight B_h^1 and B_h^2 as follows:

$$\mathcal{B}_{\mathbf{out}}^{\mathbf{1}} = \mathcal{B}_h^1 \eta, \mathcal{B}_{\mathbf{out}}^{\mathbf{2}} = \mathcal{B}_h^2 \eta^{\mathsf{T}}, \tag{2.7}$$

Where $\mathcal{B}_h^1, \mathcal{B}_h^2 \in \mathbb{R}^{T \times J \times C}$ and B_{out}^1, B_{out}^2 are the outputs of attention module. We observe that in its more common use [314], graph attention is used to estimate the interaction coefficients of the adjacency matrix A. This is done by learning a function (general MLP) of two node embeddings. By contrast, when the nodes of the graph are semantically given (body parts of a leader and follower person), one may learn the interaction coefficient (i.e., each term of A) directly, with a joint function of all nodes (not just pairs). The direct estimation results in better performance, as shown by the experiments in Sec. 2.4.3. Hence, the GCN with fully-learned parameters is selected as a best practice rather than attention.

Hierarchical body parts

To the best of our knowledge, a high-level motion representation improves the prediction of human poses[186]. [69] achieves this by concatenating the higher level as an extra node and hand-crafting ad-hoc neighborhoods of nodes.

We integrate a module within the model that enables it to decrease the number of skeleton keypoints for both bodies. We allow the model to naturally learn aggregations between nodes by excluding artificial aggregations while shifting between hierarchies. We employ a linear layer that learns an optimized aggregation when downscaling, and the same is done when upscaling to retrieve the original size skeleton. Although we gain a small improvement by adopting hierarchies, it becomes a limiting factor rather than a gain when combined with other best practices.

2.3.3 Decoding Best Practices

In earlier works, convolutions have been employed for the decoding stage [98, 205, 292]. However, the most recent SoA method chose a plain, fully connected layer [112]. In this section, we will analyze the two solutions, and in Sec. 8.5.2, we will show why we choose the latter.

Convolutional-based decoder
In the convolutional-based decoder, convolutional layers applied to the temporal dimension are responsible for estimating the pose. It aims to forecast the subsequent frames, $t+1$ to $t+n$, given the first t frames. This structure is known as Temporal Convolutional Network (TCN) [98, 205, 292].

FC-based decoder ✓
The decoder consists of a single linear layer [112] in charge of mapping the observed T frames to the predicted N.

2.3.4 Novel Adjacency Matrix Initialization

We propose a novel initialization methodology, aiming to preserve variance during the forward pass, which matches the preservation of gradients in the backward. Since over several layers a non-unit variance results in vanishing or exploding signals, and neither of those is good for training, as they stall the gradient, we aim to preserve the variance. To do that, under the assumption of a neural network consisting of only linear layers and linear activation functions, [105] proposes to estimate the standard deviation by considering the number of neurons in both the current and previous layer.

It is particularly relevant for our model because it comprises 8 layers while GCNs are often shallow [155]. We propose to randomly initialize the fully learnable matrices A_s, A_t, and W according to a uniform distribution, whose bounds are defined in such a way that considers both the number of graph nodes and the number of timeframes.

Convolutions on graphs that adopt a normalized adjacency matrix [155, 306] use a well-known graph and do not let all nodes interact with each other. Furthermore, normalization avoids vanishing and exploding gradient, yet it limits the performance and, in the end, fully-learnable yields the best performances [274, 292]. Here is the importance of randomly initializing an *ad hoc* fully learnable adjacency matrix, avoiding exploding or vanishing gradients. The response from the Separable GCN at layer l, according to Eq. (2.4), is

$$\mathbf{X}^{l+1} = \sigma\left(A_s^l A_t^l \mathbf{X}^l W^l\right), \quad \forall l. \tag{2.8}$$

Let's assume matrices A_s, A_t, and W to be independent, have zero mean [105, 126] and uniformly distributed. To constrain variance, hence stabilize training and avoid exploding or vanishing gradient, constraining the variance of the output product of n^l neurons at layer l times W to 1 [126] is a sufficient

condition, i.e.,

$$\frac{1}{k} n^l Var[W^l] = 1, \quad \forall l, \tag{2.9}$$

where $k = 2$ in the case of Re-LU activations, which are asymmetric [126] (while $k = 1$ for symmetric activations such as the `tanh`). For the spatial matrix, rather than the number of neurons n^l, we consider the number of nodes v, which A_s integrates

$$\frac{1}{k}(n_v^l) Var[A_s^l] = 1, \quad \forall l. \tag{2.10}$$

Similarly, we consider t time frames to initialize the temporal matrix A_t,

$$\frac{1}{k}(n_t^l) Var[A_t^l] = 1, \quad \forall l. \tag{2.11}$$

When initializing W with a zero-mean uniform distribution, the constraint of Eq. (2.9) yields the following distribution for the initialization:

$$W^l \sim U\left[-\sqrt{\frac{k}{n^l}}, \sqrt{\frac{k}{n^l}}\right], \quad \forall l. \tag{2.12}$$

The spatial and temporal matrix constraints of Eqs. (2.10) and (2.11) translate to the following initializing distributions for A_s and A_t respectively:

$$A_s^l \sim U\left[-\sqrt{\frac{k}{n_v^l}}, \sqrt{\frac{k}{n_v^l}}\right], \tag{2.13}$$

$$A_t^l \sim U\left[-\sqrt{\frac{k}{n_t^l}}, \sqrt{\frac{k}{n_t^l}}\right], \quad \forall l. \tag{2.14}$$

Action	A1				A2				A3				A4				A5				A6				A7				Average ↓			
Time (msec)	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*
LTD [214]	70	125	157	189	131	242	321	426	102	194	260	357	62	117	155	197	72	131	173	231	81	151	200	280	112	223	315	442	90	169	226	303
HisRep [212]	52	103	139	188	96	186	256	349	57	118	167	240	45	93	131	180	51	105	149	214	61	125	176	252	71	150	222	333	62	126	177	251
MSR-GCN [69]	56	100	132	175	102	187	256	365	65	120	166	244	50	95	127	172	54	100	138	202	70	132	182	258	82	154	218	321	69	127	174	248
MRT [319]	50	98	134	188	79	155	212	307	53	106	152	229	47	95	131	185	52	105	149	215	58	118	166	242	65	136	199	299	58	116	163	238
siMLPe [112]	49	102	137	177	88	180	244	336	57	122	174	254	45	100	137	182	50	103	144	206	59	126	175	250	77	164	134	348	60	128	178	250
XIA [108]	49	98	140	192	84	166	234	346	51	105	154	234	41	84	120	161	43	90	132	197	55	113	163	242	62	130	192	291	55	112	162	238
Ours	**34**	**71**	**105**	**159**	**56**	**121**	**181**	**292**	**36**	**78**	**118**	**195**	**30**	**66**	**98**	**145**	**35**	**74**	**113**	**171**	**41**	**88**	**129**	**193**	**47**	**108**	**166**	**261**	**39**	**86**	**129**	**202**

Table 2.1: Results in millimeters for ExPI Common actions split. Our model achieves state-of-the-art results in all actions considered, at each predicted time instant.

Action	A8			A9			A10			A11			A12			A13			A14			A15			A16			Average ↓		
Time (msec)	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*
LTD [214]	252	333	387	174	228	268	139	184	217	239	324	394	175	226	259	148	191	220	176	240	286	143	178	192	146	193	226	177	233	272
HisRep [212]	157	219	257	134	190	233	96	146	187	195	283	358	121	169	206	92	129	**160**	129	193	245	80	104	121	112	154	187	124	176	218
MSR-GCN [69]	177	239	295	143	179	213	157	222	281	230	289	335	188	245	290	148	198	248	234	319	384	176	232	278	162	218	266	179	238	288
MRT [319]	170	231	308	145	199	270	141	245	338	225	327	481	131	180	253	120	169	238	165	229	322	110	151	209	105	144	201	146	205	291
siMLPe [112]	165	220	258	137	198	246	104	154	198	210	301	432	114	156	187	94	132	160	140	204	255	91	119	138	120	166	204	131	183	225
XIA [108]	156	216	256	126	175	213	96	152	205	191	287	377	118	165	203	91	129	162	122	183	232	81	**107**	**128**	106	150	185	121	174	218
Ours	**113**	**164**	**203**	**114**	**167**	**209**	**85**	**136**	**183**	**153**	**231**	**304**	**100**	**148**	**188**	**82**	**125**	162	**91**	**138**	**179**	**79**	109	132	**85**	**124**	**156**	**100**	**149**	**191**

Table 2.2: Results in millimeters for ExPI Unseen actions split. On average, we outperform the baseline considered over short and long time horizons.

2.4 Experiments

We thoroughly evaluate the proposed best practices on the most recent and challenging 2-body pose forecasting dataset ExPI [108], comparing against the SoA and the best single-pose forecasting techniques adapted to the task. The selected best practices also perform on par with the SoA in single-pose forecasting on the established Human3.6M dataset [139].

2.4.1 Benchmark and baselines

Datasets. The dataset used for multi-body pose forecasting, ExPI [108], is a collection of two different dancing pairs performing Lindy Hop sessions, dubbed "extreme human interaction" by the authors [108]. Data were collected in a multi-camera platform with 68 synchronized and calibrated RGB cameras and a motion capture system with 20 mocap cameras. The missing points were manually fixed to ensure good data quality. ExPI contains 115 sequences at 25 fps with 18 body joints for each of the two persons involved. These agents are grouped in two couples, dubbed $(\mathcal{A}_c^1, \mathcal{A}_c^2)$, which perform 16 different actions. Actions A1 to A7 are common to both couples; A8 to A13 performed only by $\mathcal{A}_c^1$ and A14-A16 by $\mathcal{A}_c^2$. Based on this, ExPI provides three different splits to test the model on:

- **Common.** Training and test set are composed only of actions performed by both couples. The ones belonging to $\mathcal{A}_c^2$ define the train set, and $\mathcal{A}_c^1$'s the test set.
- **Unseen.** Differently from the previous one, this split has common actions to both $\mathcal{A}_c^1$ and $\mathcal{A}_c^2$ as the train set and couple-specific actions as the test one. This subset allows us to test for generalization.
- **Single.** In this split, a single action from couple $\mathcal{A}_c^2$ is used as a train set, and the same action from couple $\mathcal{A}_c^1$ as the test set. It allows testing how the model generalizes to a new couple for each action.

We also test on Human3.6M [139], an established dataset for single-person pose forecasting. It consists of a total of 3.6 million poses, acquired at 25 fps, depicting seven actors performing 15-day real-life actions, e.g., walking, sitting, and talking on the phone. Following [69, 212, 216], we train on subjects S1, S6, S7, S8, S9, we use S11 for validation, and S5 for testing.

Evaluation metrics. We validate performance by the *Mean per joint position Error*, defined as the MPJPE [139, 214] and renamed as JME in [108] at a future frame t:

$$L_{\text{JME}} = L_{\text{MPJPE}} = \frac{1}{V} \sum_{v=1}^{V} ||\hat{x}_{vt} - x_{vt}||_2, \tag{2.15}$$

where $\hat{x}_{vt}$ and x_{vt} are the 3-dimensional vectors of a target joint and the ground truth, respectively. For the joint evaluation of the 2-body position error, the two body poses are normalized into the same reference system. In this work, we keep the MPJPE notation.

Baselines. We select the latest and best-performing single-body pose forecasting models, and we adapt them to predict the motion of two people. XIA-Transformer [108] is the only 2-body pose forecasting method in the literature. XIA uses a transformer to encode skeleton features and model the body-body interaction via attention. We consider [319] the only multi-body model based on a Transformer architecture. Due to the lack of multi-body pose forecasting models, we also compare them to single ones. LTD [214]

consists of a cascade of GCN blocks acting on frequencies, and its extension, HisRep [212], inserts a motion attention mechanism based on DCT coefficients operating on sub-sequences of the input. MSR-GCN [69] is a hierarchical GCN-based technique that applies multi-scale aggregations, so coarser scales represent groups of body joints and coarser motion. In Table 2.4 we compare ourselves, again, to LTD [214], HisRep [212] and MSR-GCN [69] and, additionally, on two recent single-body models. SeS-GCN [274] adopts an all-separable GCN with a teacher-student approach, and the SoA [112], which consists of MLPs encoding spatial and temporal relationships.

Action	A1				A2				A3				A4				A5				A6				A7			
Time (msec)	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*
LTD [214]	70	126	155	183	131	243	312	415	102	194	252	338	62	117	153	203	71	131	171	231	81	151	199	299	112	223	306	411
HisRep [212]	66	118	153	190	128	231	308	417	74	143	205	295	64	120	159	191	63	121	166	227	90	168	232	312	88	166	232	332
MSR-GCN [69]	64	108	136	**170**	119	210	282	385	79	144	189	265	59	103	134	173	65	118	162	225	86	151	201	283	96	178	255	362
MRT [319]	63	120	160	218	97	190	249	346	77	148	193	240	51	102	139	186	61	118	163	226	58	115	151	198	82	172	244	340
siMLPe [112]	60	113	145	200	104	202	268	373	76	150	205	305	58	110	151	203	64	123	163	218	76	152	207	277	93	180	254	341
XIA [108]	64	120	160	199	109	200	275	381	59	117	174	277	60	116	162	209	53	106	152	221	65	122	166	223	74	144	**203**	**301**
Ours	**52**	**94**	**128**	179	**89**	**176**	**242**	**329**	**42**	**90**	**129**	**200**	**49**	**96**	**134**	**185**	**48**	**99**	**140**	**196**	**52**	**105**	**144**	**198**	**68**	**140**	204	305

Table 2.3: Results in millimeters for ExPI Single actions split. We outperform in 6 out of 7 stocks all baselines considered according to the MPJPE metric. For the other stocks our model is comparable with the current state of the art.

2.4.2 Evaluation of human pose forecasting

We evaluate our model quantitatively and qualitatively on ExPI's [108] provided splits. We further test our model's generalization power on the single-body dataset Human3.6M.

ExPI Common Actions. Table 2.1 shows the results obtained from our best model with our selected best practices. These outperform every tested method by a large margin, both the SoA single-person and the SoA 2-body pose forecasting techniques. The overall mean improvement is 22% over all actions and all time horizons. In particular, on all actions, the improvement for short-term future predictions (200 msec) is 29% and 15% for the long-term.

ExPI Unseen Actions. Table 2.2 also showcases improvements using the proposed best practices. On average, across all forecasting horizons, the improvement is 14%.

ExPI Single Actions. In Table 2.3, also for the case of single actions, the best practices report an average improvement of 14.2%. They outperform all other tested techniques in 6 (out of 7) actions at all predicted time horizons. It confirms the generalization of our model to new people.

ExPI qualitative. In Fig. 2.2, the current SoA, ExPI [108], is compared against the best-practice model (*Ours*), qualitatively. The first three columns depict observations; the following four are future motion predictions. The light-colored pictograms represent ground-truth motion. The best practices provide, in general, better predictions. Best improvements are observed in the case of large motion displacements, cf. the last two rows, action "Cartwheel".

Evaluation of single-person pose forecasting. We test how the 2-body best practices transfer back to single-person pose forecasting for a sanity check. In Table 2.4, observe that the best practices (*Ours*) yield results within a small margin compared to SoA. Note that, for the sake of this experiment, we just run the

	MPJPE ↓			
Time Horizon (msec)	*160*	*400*	*560*	*1000*
LTD [214]	23.4	58.9	78.3	114.0
HisRep [212]	22.6	58.3	77.3	112.1
MSR-GCN [69]	25.5	63.3	81.1	114.1
SeS-GCN [274]	29.0	64.0	84.4	113.9
siMLPe [112]	**21.7**	**57.3**	**75.7**	**109.4**
Ours w/o init.	27.3	64.6	83.1	116.3
Ours	26.8	63.1	81.1	113.2

Table 2.4: Error in millimeters on Human3.6M dataset. We show how our method adapted to single-person human pose forecasting is comparable with the best-performing techniques on average.

	Model	**Input Repr.**	**Encoding**					**Decoding**	***MPJPE*** ↓				**Param.** ↓
		Freq. Enc. ✓	Learn. ✓	Sep. ✓	Init. ✓	Att.	Hier.	FC ✓	*200*	*400*	*600*	*1000*	(M)
1	[108]	✓	✓			✓			55	112	162	238	8.5
2	Space-time GCN		✓						108	152	255	379	1.08
3	*(kin. tree)*			✓					81	129	183	260	0.18
4			✓	✓					55	112	156	224	0.18
5	Input repr. practice	✓	✓	✓					41	88	135	219	0.18
6			✓	✓	✓				53	106	148	216	0.18
7	Encoder practices		✓	✓		✓†			55	112	157	228	9.9
8			✓	✓			✓		51	104	148	223	0.18
9	Decoder practices		✓	✓				✓	51	104	145	212	**0.17**
10		✓	✓	✓				✓	41	89	133	208	**0.17**
11		✓	✓	✓			✓	✓	51	104	146	217	**0.17**
12	**Best model**	✓	✓	✓	✓			✓	**39**	**86**	**129**	**202**	**0.17**

Table 2.5: Combinations of best practices. From left to right, we have frequency encoding, fully learnable connections, Space-time separability, initialization, attention mechanism, hierarchy, fully connected layer as a decoder. †: we implement a Graph Attention Network (GAT) tailored for GCNs, similar in spirit to [108] designed for transformers.

2-body best-practice model *as is*. Without any hyper-parameter tuning. Furthermore, the initialization gives an overall 2.4% over the counterpart model that does not use it.

2.4.3 Evaluation of Best Practices

In this section, we refer to Table 2.5 and thoroughly assess each selected practice. First, we select a baseline GCN model. Secondly, we assess each practice's performance, added as a standalone extension. Thirdly, we integrate practices. Best practices are assessed based on their standalone performance improvement and complementarity. Finally, in Table 2.6, we evaluate the impact of the proposed initialization in more detail.

Baseline selection. We first select a baseline model on which we test each best practice. We identify three possible GCN-based encoder architectures:

- Space-time GCN [338]: this is a plain GCN model $\sigma(AXW)$ with learnable A (learnable connectivity and graph weights)
- Space-time separable GCN with learnable kinematic tree: inspired by [338] and [292] to factorize the adjacency matrix into two spatial and temporal learnable matrices, whereby the spatial connectivity is constrained to the kinematic tree
- Space-time separable GCN with fully-learnable connections: lastly, we evaluate a space-time separable GCN with fully-learnable adjacencies matrices taking inspiration from [292].

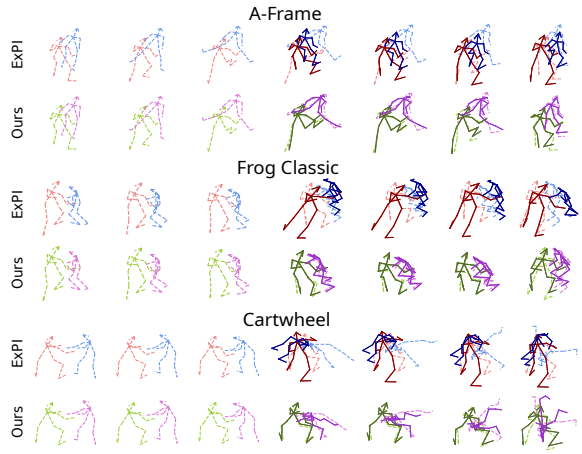


Figure 2.2: Visual comparison of our proposed best-practice model (*Ours*) against ExPI [108]. The first three columns are observed, and the last four are predicted poses. Light-colored and dashed skeletons are GT, and darker and solid ones are predictions. Note the improved larger-displacement motions (*Cartwheel*).

As shown in Table 2.5, the space-time GCN with separability (row 3) has an overall decrease of error by 25% compared to the base GCN (row 2). A considerable additional performance boost (18% over all frames) s also given by using the separability and fully-learnable connections (row 4) instead of limiting the learning procedure on the kinematic tree. The simple space-time separable GCN already outperforms XIA [108] while having a fraction of the parameters, although XIA includes DCT representations and attention. Thus GCN with separability and fully-learnable connections is a good baseline to build upon.

Standalone best practices. Table 2.5 shows input representation (row 5), encoding (rows 6-8), and decoding practices (row 9). When considering the input representation and decoding techniques, DCT, and fully connected (FC) layer as decoder, it is clear that both have a considerable impact. The DCT provides a significant boost in short-term predictions, up to 25%, while the FC-based decoder offers a more substantial increase in long-term predictions, up to 7% against TCN (when the box is not ✓). Regarding the encoder practices, the novel initialization procedure and a hierarchical architecture improve the chosen baseline by 5% and 4%, respectively. On the other hand, using the attention technique did not lead to any gain in performance and is hence not considered a best practice.

Integrated best practices. Rows 10-12 in Table 2.5 refers to the combinations of techniques that performed best independently.

Integrating the input representation using DCT coefficients and the FC-based decoder indicates how these two methods can be used in addition to the standard method. Secondly, we include a Graph Attention Network as explained in Sec. 2.3.2 to account for the interaction. The performance does not benefit from it, and the number of parameters is considerably higher. Lastly, a hierarchical structure lowers performance when combined with other practices, so we do not consider it a best practice. Our proposed initialization improves our best practice model by another 3.5%.

Impact of initialization. Table 2.6 shows the average of multiple runs for different initialization methods and the corresponding standard deviation. We compare our strategy with the Uniform sampling and the two established methodologies of [105], and [126]. Our proposed initialization exceeds or is on par with the

others on average, having more than 2.6% improvement over uniform sampling over the longer time horizon. Note also the lower standard deviation of performance for our proposed technique, especially for the most challenging long-term prediction horizon (at least 2x lower), which we interpret as improved stability.

	MPJPE ↓			
Time Horizon (msec)	*200*	*400*	*600*	*1000*
Uniform	39.7 ±0.7	87.6 ±0.7	132.2 ±0.5	207.7 ±1.1
Glorot et al.[105]	40.3 ±0.1	89.4 ±1.2	134.3 ±1.5	207.9 ±1.8
He et al.[126]	40.2 ±0.4	88.6 ±0.7	133.4 ±1.4	206.6 ±1.2
Ours	**39.2 ±0.4**	**86.4 ±0.6**	**129.4 ±1.0**	**202.2 ±0.5**

Table 2.6: Initialization procedures for best practices model.

2.5 Proof on initialization

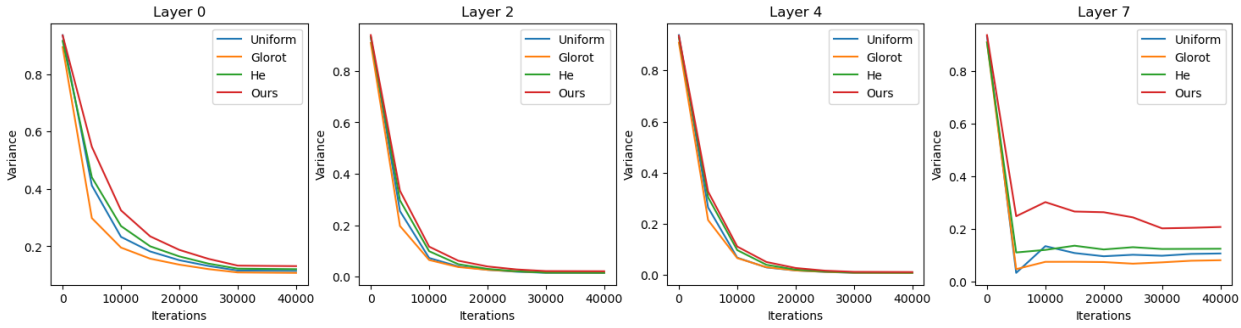


Figure 2.3: Comparison of feature activation variances, at layers 0, 2, 4 and 7, estimated during the model training, upon initialization with random "Uniform", "Glorot" [105], "He" [126] against "Ours", our proposed initialization technique.

Here we provide more detailed proof for Eqs. 10-11, 13-14 of the main paper. At each layer l, we assume learnable matrices $W^l \in \mathbb{R}^{C\times C'}$, $A_s^l \in \mathbb{R}^{T\times 2J\times 2J}$ and $A_t^l \in \mathbb{R}^{2J\times T\times T}$ to be independent, have zero mean and be uniformly distributed. With T being the number of timeframes, J being the number of joints in one person, and C and C' being the number of input and output channels.

First, we review and demonstrate the proposed initialization for the forward (Sec. 2.5.1) and backward passes (Sec. 2.5.2). Then, in Sec. 2.5.3, we illustrate how the initialization results in better training robustness.

2.5.1 Forward propagation

Let us consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to encode the body kinematics, with all joints at all observed frames as the $2J \times T$ nodes defining the vertex set V, and edges $\epsilon \in \mathcal{E}$ connecting them.

Following up on Eq. (4) from the main paper, the response of a separable GCN [292] layer is

$$\begin{cases} Y^l = A_s^l A_t^l X^l W^l \\ X^l = \sigma\left(Y^{l-1}\right), \end{cases} \tag{2.16}$$

where $X \in \mathbb{R}^{T\times 2J\times C}$ is the C-dimensional embedding of each node.

W may be interpreted as a fully connected layer acting on each of the graph node embeddings separately, i.e., on each of the joints from the two people at all times, for a total of $2J \cdot T$ connections. W may be

assumed to have C' neurons, i.e. to output $n = C'$ neural activations per node. The matrices A_s and A_t act on the spatial and temporal number of connections of the graph, respectively (please also see [292] for more details). Specifically, A_s may be considered to model the interaction of each node with all $2J$ others at the same frame, by means of $n_v = 2J$ neurons. Correspondingly, we may consider A_t to model the interaction of each node with those of the same joint at all T times, by means of $n_t = T$ neurons.

The number of interactions corresponds to the number of terms that are summed. Assuming matrices to be i.i.d. [126], the variance of the sum yields the sum of variances, thus

$$Var\left[Y^l\right] = n^l n_v^l n_t^l Var\left[A_s^l A_t^l X^l W^l\right]. \tag{2.17}$$

Assuming A_s^l, A_t^l, and W^l to have zero mean [126], the variance of the product of independent variables is

$$\begin{aligned} Var\left[Y^l\right] = n^l n_v^l n_t^l Var\left[A_s^l\right] Var\left[A_t^l\right] \\ \mathbb{E}\left[\left(X^l\right)^2\right] Var\left[W^l\right]. \end{aligned} \tag{2.18}$$

We consider the PReLU as our activation function, i.e.

$$\sigma\left(X^l\right) = \max\left(0, Y^{l-1}\right) + a\min\left(0, Y^{l-1}\right), \tag{2.19}$$

with a being a learnable parameter that, when set to 0, reduces to the ReLU[2]. This means that for a generic a, $\mathbb{E}\left[X^l\right] \neq 0$. Let A_s^{l-1}, A_t^{l-1}, and W^{l-1} have symmetric zero-centered distributions [126]. This may then be also implied for Y^{l-1}, and we may write

$$\mathbb{E}\left[\left(X^l\right)^2\right] = \frac{1+a^2}{2} Var\left[Y^{l-1}\right]. \tag{2.20}$$

Substituting for Eq. (2.20) in Eq. (2.18) we get

$$\begin{aligned} Var\left[Y^l\right] = \frac{1+a^2}{2} n^l n_v^l n_t^l Var\left[A_s^l\right] Var\left[A_t^l\right] \\ Var\left[Y^{l-1}\right] Var\left[W^l\right]. \end{aligned} \tag{2.21}$$

Considering L layers, this yields the following variance formulation for the entire separable GCN model:

$$\begin{aligned} Var\left[Y^L\right] = Var\left[Y^1\right] \prod_{l=2}^{L} \frac{1+a^2}{2} n^l n_v^l n_t^l \\ Var\left[A_s^l\right] Var\left[A_t^l\right] Var\left[W^l\right]. \end{aligned} \tag{2.22}$$

In order to have the same input and output signal variance for the entire model, it suffices to assume that each layer l has the same input and output signal variances. This corresponds to setting the variance induced by the multiplicative parameters to be 1 i.e.,

$$\frac{1+a^2}{2} n^l n_v^l n_t^l Var\left[A_s^l\right] Var\left[A_t^l\right] Var\left[W^l\right] = 1. \tag{2.23}$$

[2]Also recall that a small a e.g., 0.01, is the LeakyRelu and $a = 1$ is the linear case.

Towards this goal, it suffices to set each parameter initialization variance as follows

$$\frac{1+a^2}{2} n_v^l Var\left[A_s^l\right] = 1 \tag{2.24}$$

$$\frac{1+a^2}{2} n_t^l Var\left[A_t^l\right] = 1 \tag{2.25}$$

$$\frac{1+a^2}{2} n^l Var\left[W^l\right] = 1 \tag{2.26}$$

2.5.2 Backward propagation

Action	A1				A2				A3				A4				A5				A6				A7				Average ↓			
Time (msec)	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*
LTD [214]	51	92	116	132	51	91	116	148	43	80	103	130	38	70	89	111	39	70	90	116	42	75	94	123	52	101	139	198	45	83	107	137
HisRep [212]	34	69	97	130	44	84	115	150	32	65	91	121	27	56	82	112	28	58	85	121	34	66	88	115	42	83	120	171	34	69	97	131
MSR-GCN [69]	41	75	99	126	54	96	129	180	41	74	98	135	34	61	82	106	33	59	79	109	42	71	93	124	57	103	146	210	43	77	104	141
MRT [319]	34	69	95	128	39	78	106	142	30	59	83	115	28	57	79	110	28	57	79	108	34	68	91	120	39	80	114	160	33	67	92	126
siMLPe [112]	32	69	94	115	44	93	122	160	33	73	102	138	26	61	87	114	28	60	84	112	32	69	93	123	45	94	127	171	34	74	101	133
XIA [108]	32	68	99	128	41	82	116	163	29	58	84	116	24	50	73	96	24	51	75	109	31	62	86	114	41	81	115	160	32	65	93	127
Ours	**24**	**51**	**76**	**114**	**31**	**66**	**93**	**132**	**23**	**49**	**70**	**103**	**19**	**41**	**60**	**85**	**21**	**44**	**64**	**93**	**24**	**52**	**73**	**100**	**29**	**64**	**95**	**143**	**24**	**52**	**76**	**110**

Table 2.7: Results in millimeters for ExPI Common actions split. Our model achieves state-of-the-art results in all actions considered, at each predicted time instant.

Action	A8			A9			A10			A11			A12			A13			A14			A15			A16			Average ↓		
Time (msec)	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*	*400*	*600*	*800*
LTD [214]	106	136	155	91	119	135	72	96	116	95	123	146	85	106	116	74	91	101	86	115	137	98	125	134	85	110	124	88	113	129
HisRep [212]	86	120	142	73	104	128	54	82	104	101	144	476	61	82	94	49	67	80	73	105	129	53	73	86	64	89	104	68	96	116
MSR-GCN [69]	88	118	142	90	113	136	90	122	148	103	134	155	101	135	160	74	98	121	103	143	173	87	111	132	84	106	122	91	120	143
MRT [319]	89	121	161	79	108	145	69	100	147	97	133	174	71	96	127	66	88	117	83	113	149	72	98	132	67	92	121	77	105	141
siMLPe [112]	95	125	141	82	114	134	63	93	115	124	174	212	61	80	92	50	67	79	83	116	138	59	81	90	72	99	116	77	106	124
XIA [108]	82	116	142	69	97	120	**52**	79	104	95	137	171	58	80	93	51	70	**84**	70	105	134	**53**	**73**	**88**	63	88	104	66	94	**116**
Ours	**68**	**95**	**115**	**66**	**95**	**116**	**52**	**78**	**103**	**86**	**124**	**150**	**54**	**76**	**91**	**47**	**68**	**84**	**59**	**86**	**108**	**53**	77	94	**53**	**77**	**94**	**60**	**86**	121

Table 2.8: Results in millimeters for ExPI Unseen actions split. On average, we outperform the baseline considered over short and long time horizons.

Action	A1				A2				A3				A4				A5				A6				A7			
Time (msec)	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*	*200*	*400*	*600*	*1000*
LTD [214]	51	99	129	163	61	110	150	229	53	96	131	188	46	81	106	142	44	79	106	147	53	100	162	176	70	133	163	198
HisRep [212]	51	93	114	127	51	91	116	162	43	80	100	126	38	70	88	118	39	70	90	125	42	75	93	123	52	101	137	188
MSR-GCN [69]	45	83	106	118	57	102	135	178	39	72	100	132	41	77	103	119	35	70	97	125	46	82	107	137	48	90	121	169
MRT [319]	36	69	93	123	44	81	106	138	41	76	96	114	30	61	81	105	33	64	88	121	34	64	83	104	42	83	114	157
siMLPe [112]	43	84	107	137	55	107	142	182	47	91	120	164	39	76	101	129	38	75	99	128	47	90	118	150	58	110	150	197
XIA [108]	43	84	115	131	53	99	136	185	35	68	98	140	37	74	106	128	29	59	86	125	39	72	94	119	43	82	112	**152**
Ours	**34**	**63**	**86**	**115**	**41**	**79**	**105**	**138**	**27**	**55**	**77**	**110**	**31**	**64**	**88**	**119**	**27**	**55**	**77**	**107**	**30**	**58**	**78**	**103**	**38**	**78**	**109**	154

Table 2.9: Results in millimeters for ExPI Single actions split. We outperform in 6 out of 7 stocks all baselines considered according to the MPJPE metric. For the other stocks our model is comparable with the current state of the art.

The gradient of a separable GCN is

$$\begin{cases} \frac{\partial L}{\partial X^l} = A_s^l A_t^l \frac{\partial L}{\partial Y^l} \tilde{W}^l \\ \frac{\partial L}{\partial Y^l} = \frac{d\sigma}{dY^l} \frac{\partial L}{\partial X^{l+1}}, \end{cases} \tag{2.27}$$

with $\tilde{W} \in \mathbb{R}^{C' \times C}$, while A_s and A_t have the same dimensionality as in the forward pass. Our backward response number is $\tilde{n}^l = C$ for $\tilde{W}$, and it is still n_v and n_t for A_s and A_t, respectively, thus

$$Var\left[\frac{\partial L}{\partial X^l}\right] = n_v^l n_t^l \tilde{n}^l Var\left[A_s^l A_t^l \frac{\partial L}{\partial Y^l}\tilde{W}^l\right]. \tag{2.28}$$

We let A_s^l, A_t^l, $\tilde{W}^l$ and $\frac{\partial L}{\partial Y^l}$ be independent. Let us assume A_s^l, A_t^l, and $\tilde{W}^l$'s to be zero-centered symmetric distributions, and $\frac{\partial L}{\partial X^l}$ to have zero mean [126]. Similarly to the forward pass, we have to consider the PReLU activation function.

If we assume that $\frac{d\sigma}{dY^l}$ and $\frac{\partial L}{\partial X^{l+1}}$ are independent [126], we get

$$\mathbb{E}\left[\frac{\partial L}{\partial Y^l}\right] = \frac{1+a^2}{2}\mathbb{E}\left[\frac{\partial L}{\partial X^{l+1}}\right] = 0, \tag{2.29}$$

$$\mathbb{E}\left[\left(\frac{\partial L}{\partial Y^l}\right)^2\right] = \frac{1+a^2}{2}Var\left[\frac{\partial L}{\partial X^{l+1}}\right]. \tag{2.30}$$

Considering Eq. (2.28) and the assumed independence, we elaborate on Eq. (2.30) as follows

$$Var\left[\frac{\partial L}{\partial X^l}\right] = n_v^l n_t^l \tilde{n}^l Var\left[A_s^l\right] Var\left[A_t^l\right] \tag{2.31}$$

$$Var\left[\frac{\partial L}{\partial Y^l}\right] Var\left[\tilde{W}^l\right],$$

$$= \frac{1+a^2}{2} n_v^l n_t^l \tilde{n}^l Var\left[A_s^l\right] Var\left[A_t^l\right] \tag{2.32}$$

$$Var\left[\frac{\partial L}{\partial X^{l+1}}\right] Var\left[\tilde{W}^l\right].$$

For L layers, this yields

$$Var\left[X^2\right] = Var\left[\frac{\partial L}{\partial X^{L+1}}\right]\prod_{l=2}^{L}\frac{1+a^2}{2} n_v^l n_t^l \tilde{n}^l \tag{2.33}$$

$$Var\left[A_s^l\right] Var\left[A_t^l\right] Var\left[\tilde{W}^l\right].$$

In order to avoid exploding and vanishing gradients, a sufficient condition is to set the gradient of each layer to maintain the signal variance throughout the backpropagation

$$\frac{1+a^2}{2} n_v^l n_t^l \tilde{n}^l Var\left[A_s^l\right] Var\left[A_t^l\right] Var\left[\tilde{W}^l\right] = 1. \tag{2.34}$$

Finally, it suffices to set the variance initialization of each of the parameter matrices as follows:

$$\frac{1+a^2}{2} n_v^l Var\left[A_s^l\right] = 1 \tag{2.35}$$

$$\frac{1+a^2}{2} n_t^l Var\left[A_t^l\right] = 1 \tag{2.36}$$

$$\frac{1+a^2}{2} \tilde{n}^l Var\left[\tilde{W}^l\right] = 1 \tag{2.37}$$

Note that the result in Eq. (2.34) resonates with what was obtained for the forward pass, in Eq. (2.23).

This ensures that the same initialization may be adopted to yield the signal and gradient requirements both in the forward and backward passes.

2.5.3 Training variance

The model output should maintain unit variance during training for the sake of activation functions and robust training, avoiding vanishing gradient [105, 126]. This becomes more challenging when adopting deeper networks, which regards our case, as we consider a separable-GCN twice as deep as the original one of [292] (8 Vs. 4 layers).

In Fig. 2.3, we consider the variances of the feature activations at sampled layers during training and compare the result of our proposed initialization against the "Uniform", "Glorot" [105], and "He" [126] techniques. Observe how "Ours" yields feature variances that are consistently closer to the desired unit variance. This is especially true for the last layer (layer 7), arguably the most challenging.

2.6 AME results

Following [108], we also evaluate our model using a different metric for the error between poses, the *Aligned Mean per joint position Error* (AME). Both poses are independently normalized in advance to avoid positional errors, to correct errors due to having a root joint as the origin, we use a rigid alignment transformation T.

$$L_{\text{AME}} = \frac{1}{V}\sum_{v=1}^{V} ||\hat{n}_{vt} - T(\hat{n}_{vt}, n_{vt})||_2, \tag{2.38}$$

where n is the coordinate x after normalization as defined above. The results are reported in Tab. 2.7, Tab. 2.9 and Tab. 2.8. Results are consistent with those reported in the main paper, expressed in terms of *Mean per joint position Error*(*MPJPE*), as it favors comparability with other works [30, 69, 112, 212, 214, 216, 274, 292]

2.7 Implementation details

In this section, we thoroughly describe the implementation procedures that we have used in training and testing. Furthermore, we describe the iterative approach used during testing.

2.7.1 Training and testing details

We use 10 frames as input and 10 frames as output during training. We use an iterative mechanism at test time to make a 1-second prediction(25 frames). We exclusively use our predictions as input for subsequent iterations. We extensively analyze the iterative mechanism impact in the supplementary materials. We adopt the ADAM [150] optimizer and a learning rate of 1×10^{-5}, decayed to 5×10^{-8} after 30K iterations. The model converges in 40K iterations, i.e., the training takes 23 min on a single Nvidia P6000 GPU. We also get an average prediction time on CPU[3] of 0.07 seconds compared with the fastest [108]'s 0.4. At each layer, we adopt batch normalization [138] and residual connections.

2.7.2 Iterative approach

We test different combinations of T input frames and N output frames (See Tab. 2.10) and notice that some perform better than others. Most notable works in pose forecasting [108, 112, 292] use $T = 50$ input frames

[3] An AMD Ryzen 5 3600 6-Core processor.

and $N = 10$ output ones. Conversely, the best results are obtained using $T = 10$ a mechanism [108, 212, 214] that iteratively feeds both parts of the observed history and new predictions as an input.

			MPJPE (ms)↓			
	Input Frames	*Output Frames*	*200*	*400*	*600*	*1000*
	50	1	43	113	176	274
	50	5	52	114	162	243
	50	10	62	126	174	243
	50	50	68	131	177	244
	10	1	**36**	98	164	294
	10	5	37	89	141	238
Used	10	10	39	**86**	**129**	**202**

Table 2.10: Results in millimeters on the ExPI dataset, on average common actions split. We show the impact that different combinations of input-output frames have on performance. Using 10 input frames makes predictions in the short term more accurate, helping results to be more stable in the long term.

2.8 Complete list of actions

This section lists (ref. Tab. 2.11) each action A_i, with $i = 1, ..., 16$. Refer to the supplementary material in [108] for a more detailed explanation.

Action	Name
A_1	A-frame
A_2	Around the back
A_3	Coochie
A_4	Frog classic
A_5	Noser
A_6	Toss out
A_7	Cartwheel
A_8	Back flip
A_9	Big ben
A_{10}	Chandelle
A_{11}	Check the challenge
A_{12}	Frog-turn
A_{13}	Twisted toss
A_{14}	Crunch-toast
A_{15}	Frog-kick
A_{16}	Ninja-kick

Table 2.11: List of actions and their corresponding names

Actions $A_1, ..., A_7$ are performed by both couples $\mathcal{A}_1$ and $\mathcal{A}_2$. Actions $A_8, ..., A_{13}$ are exclusive to couple $\mathcal{A}_1$, and actions $A_{14}, ..., A_{16}$ to couple $\mathcal{A}_2$.

2.9 Sample videos

In addition, we include a video comparing the results of our and the current SoA's model [108] qualitatively. It is possible to see how our model is far more accurate when analyzing both basic and complex activities. For comparison, we use the pre-trained model provided by [108] and showcase only 10 of their 50 input frames to make it the same length as ours. Still, the number of output frames remains at 25 for both models. We release videos on our project page at https://www.pinlab.org/bestpractices2body.

2.10 Conclusion

This work has identified, reviewed, and experimentally evaluated best practices for 2-body pose forecasting, to bootstrap research in the mostly unexplored task. Best practices have a large impact on SoA performance, and the novel initialization adds further improvement in performance and stability. Notably, predicting the future of two people in interaction yields better estimates than considering each person separately, so 2-body forecasting is recommended for applications such as sports and collaborative assembly in factories.

Chapter 3

Staged Contact-Aware Global Human Motion Forecasting

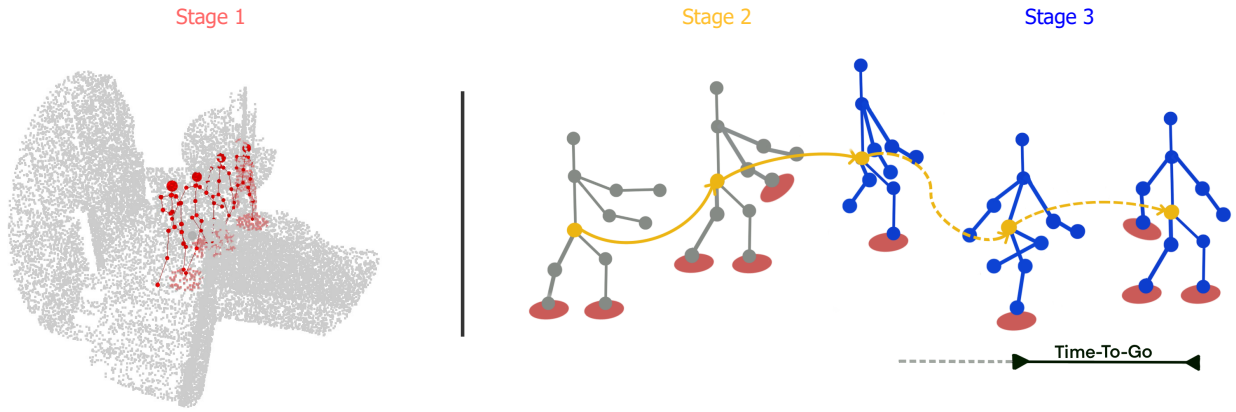


Figure 3.1: STAG forecasts scene-aware global human motion by three coarse-to-fine stages: (i) estimate the present and future contact points (light red) given the scene and the ground truth body joints (red); (ii) predict the future trajectory (dashed yellow), i.e. the future position of the root joints, given the past (solid yellow); (iii) predict the future body joints (blue) from the observed ones (gray). Each stage of STAG conditions on the previous, so trajectory forecasting leverages future estimated contact points, and pose forecasting leverages both other estimates. Awareness of the time-to-go (black arrow), the passing time between the current prediction and the end one, improves performance.

3.1 Introduction

Humans are inherently predicting the near future at all times [73, 79]. As humans and machines coexist more, predicting human motion in the immediate future becomes critical for human-robot interaction, e.g., in industrial environments [59, 164, 275] or breaking-in-time to avoid collisions [106, 238, 366]. Human motion forecasting generally includes *local* pose forecasting [109, 111, 213, 215, 217, 292, 339], in which the joint locations are predicted with respect to the root joint, and *global* pose forecasting [211, 365, 367], which takes into account the positions of joints and the root in relation to a global coordinate system.

One common issue in human motion forecasting is the omission of the environment. It leads to contrived motion when the model is used in more realistic scenarios, such as *ghost motions*, i.e., phasing through solid objects. To our knowledge, [211] is the only work that accomplishes scene-aware global human forecasting.

They first process the scene and emphasize the human-scene interaction through contact points. Subsequently, they employ an end-to-end approach to model individuals' trajectories and poses. Although contact points have shown effectiveness, employing end-to-end modeling for both trajectory and pose is suboptimal. The pose of an individual is influenced by their motion trajectory, interaction with the surrounding scene, and the pose in previous frames. However, the pose is typically not the underlying cause of the pathway. Essentially, global motion forecasting naturally aligns with a coarse-to-fine methodology that considers the scene, the trajectory, and the human pose.

We propose a novel model for *STAGed contact-aware global human motion forecasting* (*STAG*) that cascades three coarse-to-fine processing stages: (i) predicting the contact points, (ii) using them to forecast the trajectories, (iii) estimating the body pose (see Fig.7.1). Our three-stage pipeline predicts the future motion autoregressively, conditioning each stage on the previous ones. We condition the global motion on end goals and propose a time-to-go temporal encoding of the remaining duration until the endpoints are reached, i.e. informs the model on how many frames are missing.

We quantitatively evaluate the design choices of *STAG* and compare them to the SoA on the available GTA-IM dataset [211]. Overall, we get up to a 21.1% improvement on the path error with 16.2% on average, while on the pose error, we get up to 5.4% less error and 1.8% on average. We also show the generalizability of *STAG* by testing it on CMU-Mocap [63], a well-established multi-person dataset without scenes. To account for the missing scene, we only assume a planar ground. *STAG* sets a new SoA without leveraging social cues, which SoA methods use [3, 102, 320]. Overall, our contributions are threefold:

1. We introduce a novel three-stage, coarse-to-fine model, which cascadedly processes the contact points, the trajectories, and the poses of people.

2. We introduce a learnable temporal counter for the time-to-go to align the predictions with the missing time before the endpoint.

3. We perform a thorough analysis on GTA-IM [211], where we set a new SoA, and generalize *STAG* to the *scene-less* CMU-Mocap [63].

3.2 Related Work

We discuss literature relating to the three core aspects of contact-aware global human motion forecasting: human-scene interaction (Sec. 3.2.1), trajectory forecasting (Sec. 3.2.2), and human motion forecasting (Sec. 3.2.3).

3.2.1 Human-Scene Interaction

Human motion forecasting is inherently influenced by the scene context in which it occurs, thus, considering the interaction between humans and their surroundings is crucial for motion forecasting.

In motion synthesis, this shift towards including more contextual information can already be seen [51, 123, 134, 318, 367], and some works in trajectory forecasting also consider contextual information [53, 95, 175]. In human motion forecasting, scene information has been widely disregarded, with only a few works considering implicitly learning from the scene [41, 64]. However, this indirect modeling does not prevent *ghost motion*, i.e. body parts passing through objects or the scene. To the best of our knowledge, only [211] has investigated the explicit representation of human-scene interaction for human motion forecasting. [211]

proposes a two-stage pipeline, first predicting future joint-scene distances, then using this information to predict the global pose.

Working with scene context requires data that enables the model to infer environmental clues. 3D point clouds provide dense information about surfaces and objects in the scene, which is ideal for human-scene interaction and trajectory forecasting. *STAG* elaborates on the idea of contact maps and adds a component of contextual knowledge through trajectory forecasting.

3.2.2 Trajectory Forecasting

Trajectory forecasting can be divided into two main categories: *model-based* and *model-free* approaches. Model-based approaches [81, 128, 204, 311] impose physical constraints directly, while model-free approaches typically rely on implicitly learned physical plausibility [140, 273]. Some recent works [13, 353] have combined the two. Model-free approaches employ a variety of deep learning techniques such as transformers [313, 352], RNNs [129, 224] or GCNs [156, 225]. Deep learning approaches define the SoA on multiple benchmarks [190, 191, 195, 227, 353]. *STAG* follows best practices of SoA methods, adopting GCNs, attention, and trajectory endpoints. Additionally, we are the first to propose a temporal encoding for the *time-to-go*, informing the current model prediction of how long it is missing before the endpoint.

3.2.3 Human Motion Forecasting

Human motion forecasting can be divided into (local) pose forecasting and global pose forecasting. Local pose forecasting [109, 111, 213, 215, 217, 292, 339] only considers the position of the agent in relation to its root, while global pose forecasting [211, 365, 367] takes the absolute position within the given scene into account. Thus, global pose forecasting can be viewed as combining the trajectory and the (local) pose. Many applications such as human-robot collaboration [59, 164, 275], autonomous driving [106, 238, 366], sports [281, 351], augmented reality [294] or animation [234, 315] require knowledge about the global position of the agent in the scene.

Many human pose forecasting works use 2D image data only [21, 83, 177, 264, 265, 357]. However, in tasks such as industrial human-robot collaboration, where the agent's and objects' exact position in the scene is crucial, 3D data is often used [34, 45, 336]. We consider a 3D point cloud for our task, as they offer a rich scene representation. The agent in the scene can be represented as a graph of body joints or more complex representations such as meshes [107, 124, 209, 245].

While delicate tasks may require a more specific human model, the skeletal representation (adopted in *STAG*) suffices for human motion forecasting [109, 111, 213, 215, 217, 292, 339].

3.3 Methodology

STAG is designed as a three-stage model, which we overview in Sec. 3.3.1. The modeling of each stage is detailed in Secs. 3.3.2-3.3.4.

3.3.1 Proposed STAGed contact-aware global motion modelling

Our novel approach for predicting contact-aware global human motion, named *STAG*, is designed in a coarse-to-fine manner by using a three-stage pipeline (see Fig. 3.2). *STAG* is composed of a first stage that computes the contact points between the 3D scene and the body (cf. Sec 3.3.2). The second stage

uses the information from the previous stage and the past root trajectory to predict its future trajectory (cf. Sec. 3.3.3). In the third stage, the historical movement of the body, together with the contact points and human trajectory end goals, are used to predict the future global pose upon temporally encoding the time-to-go (cf. Sec 3.3.4).

Notation. We refer to $\mathbf{S}$ as the scene, $\mathbf{R}$ as the root joint trajectory, and to $\mathbf{M}$ as the global human motion. $\mathbf{S}$ represents a 3D scenes, where $\mathbf{S} \in \mathbb{R}^{N\times 3}$ contains N points, each expressed as a triplet (x, y, z). $\mathbf{R} = [R_0, \ldots, R_F]$ is a sequence of root trajectories, where $\mathbf{R}^T = [R_0, \ldots, R_{T-1}]$ represents the observed ones and $\mathbf{R}^F = [R_T, \ldots, R_F]$ the ones to predict. $R_i \in \mathbb{R}^3$ represents the 3D root coordinates. Similarly $\mathbf{M} = [M_0, \ldots, M_F]$ is a sequence of global body poses, where $\mathbf{M}^T = [M_0, \ldots, M_{T-1}]$ represents the observed ones and $\mathbf{M}^F = [M_T, \ldots, M_F]$ the ones to predict and $M_i \in \mathbb{M}^{V\times 3}$ represents the pose at timestamp i, consisting of V joints expressed as 3D coordinates.

Staged processing. In the first stage, the goal is to compute the contact points $\mathbf{C}$ defined as $[C_0, \ldots, C_T, \ldots, C_F]$, and $C_i \in \mathbb{R}^{V\times 4}$ [211] consisting of V points expressed as 4D coordinates, triplet (x, y, z) and one value $\{0, 1\}$ to indicate whether it is a contact point or not (cf. Sec 3.3.2). The second stage predicts the future trajectory $\mathbf{R}^F$ given the historical root coordinates $\mathbf{R}^T$, and the contact points $\mathbf{C}$ (cf. Sec 3.3.3). In the third stage, the objective is to predict the future body poses $\mathbf{M}^F$ by using $\mathbf{M}^T$, $\mathbf{C}$ and $\mathbf{R}^F$ (cf. Sec 3.3.4).

3.3.2 Contact Point Estimation

In the first stage, the goal is to predict the contact points $\mathbf{C}$ between the global human body motion $\mathbf{M}$ and the scene $\mathbf{S}$ (See Fig. 3.2). We use Point-Voxel CNN (PVCNN) [197] to model the scene $\mathbf{S}$ as a point cloud and encode $\mathbf{M}$ as a spatio-temporal graph [292], to capture the movement's proprieties. Following [211], we first compute the distance matrix $\mathbf{D} \in \mathbb{R}^{TV\times N}$ where each term represents the Euclidean distance between each joint in time TV and the N points in the scene. Since $\mathbf{D}$ is based on distances, it is smooth over time. We adopt a temporal encoding strategy of $\mathbf{D}$ based on the Discrete Cosine Transform (DCT) [215].

To leverage the DCT representation, we reformulate this problem by learning a mapping from the DCT coefficients of the past distance matrix $\mathbf{D}$ to those of the future one $\hat{\mathbf{D}}$. Following [211], we leverage PVCNN [197] to encode the 3D scene $\mathbf{S}$, as well as the encoded motion $\mathbf{M}^T$ and $\mathbf{D}$'s DCT coefficients.

Following leading pose forecasting literature, we use Graph Convolutional Networks [156] to encode the motion. Similarly to previous works [292, 335], we define a *spatial adjacency matrix* as $A_s \in \mathbb{R}^{T\times V\times V}$ to model the connections between joints and a *temporal adjacency matrix* $A_t \in \mathbb{R}^{V\times T\times T}$ to capture the temporal relationships.

$$\bar{\mathbf{M}}^T = \sigma(A_s A_t \mathbf{M}^T W) \tag{3.1}$$

We aim to obtain a latent vector representing the entire movement sequence and serving as a conditioning variable. To compress spatial and temporal information, we propose using two separate MLPs, MLP_S, and MLP_T:

$$\tilde{\mathbf{M}}^T = MLP_S(MLP_T(\bar{\mathbf{M}}^T)) \tag{3.2}$$

Each MLP consists of two linear layers and an equal number of activation functions. From now on, we will refer to this encoding technique as *GCN-MLP*. In summary:

$$\hat{\mathbf{D}} = IDCT(\mathbf{M}^T + f(\mathbf{S}, DCT(\mathbf{D}), \tilde{\mathbf{M}}^T)) \tag{3.3}$$

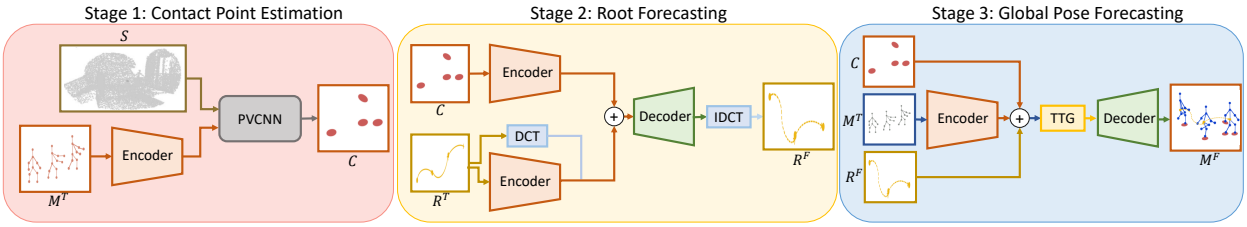


Figure 3.2: Overview of *STAG*'s three-staged pipeline. Stage 1 takes the scene and the human motion in input and predicts future interactions as contact points. Stage 2 feeds them to a trajectory forecasting model for a coarse prediction, and Stage 3 then refines it to predict future human poses.

Where f represents the trainable point-cloud encoder [197]. Ultimately, we reconvert the distance matrix $\hat{\mathbf{D}}$ to the contact points as in [211] resulting in the predicted contact points $\mathbf{C}$.

3.3.3 Root Forecasting

In the second stage, we propose to predict the person's trajectory to account for future global motion. We achieve it by predicting the future root joint $\mathbf{R}^F$ from the past $\mathbf{R}^T$. The second stage integrates the scene contacts $\mathbf{C}$, estimated in stage one. $\mathbf{R}^T$ is encoded twice, once by using DCT and secondly by using the encoder described in Sec. 3.3.2. The formulation is similar, however, $\mathbf{M}^T$ gets changed with $\mathbf{R}^T$ in Eq. (3.2) and the number of nodes $V = 1$, resulting in $\tilde{\mathbf{R}}^T$. The same encoding technique is used for the contact points, where $\mathbf{M}^T$ gets changed with $\mathbf{C}$ in Eq. (3.2) and results in $\tilde{\mathbf{C}}$. The latter encodings are concatenated and fed to an MLP, which decodes the feature dimension C and outputs $\hat{\mathbf{R}} \in \mathbb{R}^{T \times V \times C}$.

$$\hat{\mathbf{R}}^T = MLP(DCT(\mathbf{R}^T) \parallel \tilde{\mathbf{R}}^T \parallel \tilde{\mathbf{C}})), \tag{3.4}$$

where $\parallel$ indicates a concatenation operation. Lastly, the $IDCT$ reverts the transformation process to trajectories so that $\mathbf{R}^F = IDCT(\hat{\mathbf{R}}^T)$.

3.3.4 Global Pose Forecasting

For the third stage, we utilize the forecasted root trajectory $\mathbf{R}^F$ and the contact points $\mathbf{C}$ obtained from the preceding stages (see Fig. 3.2) as inputs. It enables us to predict the future pose and refine the trajectory, ultimately yielding the future global motion of the agent. We also encode the past body motion $\mathbf{M}^T$ as in Sec. 3.3.2 and concatenate the latter information. The decoding occurs autoregressively, where each future timeframe $\{i\}_{i=T}^{F}$ of the predicted body motion $\mathbf{M}_i^F$ is computed sequentially. We propose to temporally encode the scene contact points and the trajectory endpoints to raise the model understanding of the time-to-go, i.e. how long before it reaches them. At each i, we also concatenate the root's position $\mathbf{R}_e^F$, and the contact points $\mathbf{C}_e^F$ at the last frame F as end goal conditioning variables. Where respectively, $\mathbf{R}_e^F \in \mathbb{R}^C$ and $\mathbf{C}_e^F \in \mathbb{R}^{V \times C}$.

$$HM_i^F = MLP(\tilde{M}^T \parallel R_{i+1}^F \parallel \mathbf{R}_e^F \parallel \tilde{C}_{i+1} \parallel \tilde{\mathbf{C}}_e^F), \tag{3.5}$$

where HM_i^F is the the embedding at time $\{i\}_{i=T}^{F}$. Then we add the TE and decode the global body pose.

Time-to-go Temporal Encoding To insert time context, we use a learnable temporal encoder $\mathcal{T}_e$ to encode the time-to-go and add it to HM_i^F. During the autoregressive process, $\mathcal{T}_e$ measures how long is missing

before the contact and endpoints are reached. To decode the global motion we use an MLP layer. In summary:

$$M^i = MLP(HM^i + \mathcal{T}_e). \tag{3.6}$$

3.4 Experiments

In this section, we detail the dataset and metrics, compare it to the current SoA [211], and perform an extensive ablation on the staged modeling design and its components. Furthermore, we show how our model compares with SoA.

Dataset. The GTA-IM dataset [41] is a large-scale synthetic dataset that captures human-scene interactions, which consists of 50 different characters performing various activities in 7 scenes. We use [211]'s proposed pre-processing, employing 4 of the scenes as our training set, the remaining 3 as the test set, and 21 out of the 98 human joints provided by the dataset. Videos are recorded at 30fps, and we train our models to observe the past 30 frames and predict the future 60. We evaluate *STAG* in all its stages and outperform [211].

[211] also considers PROX [124] but they do not distribute the pre-processed scene-to-pointcloud nor the code for pre-processing. PROX is a real dataset captured using a Kinect-One sensor, and it contains noise at frames (e.g. jittering and corrupted pixels) and in time (missing frames). Upon best efforts, we could not replicate the pre-processed pointcloud, so we could not use it for comparison.

Metrics. The first stage is evaluated by the L_2-norm between our predicted contact point and the ground truth. For the second and third stages, we consider the *Mean Per Joint Positional Error* (MPJPE) across all joints and all the future timeframes [211]. The global movement is called Path Error, and the Pose Error represents the local body movement.

State of the art models and selected baserows. We evaluate *STAG* on the GTA-IM dataset and compare it with the current leading techniques. LTD [215] utilizes a graph convolutional network to encode motion representations in frequencies. DMGNN [187] employs a *dynamic multiscale* GNN for sequence encoding, with a decoder based on GRU. SLT [321] focuses on motion synthesis and employs an autoencoder architecture consisting of a multilayer perceptron as the initial stage, followed by motion generation using LSTM. The top-performing technique is Mao et al. [211], which combines MLP and RNN for motion encoding and employs an iterative prediction approach.

3.4.1 Comparison against SoA

First Stage - Contact points estimation. Table 3.1 is not present in [211]; thus, we ran their first stage and compared it to ours (Sec. 3.3.2). We have an overall 9.2% improvement, and it is due to our body movement's encoder, which more accurately extracts the latent representations.

Second Stage - Global pose error. This section focuses on our second stage's impact on the Path Error. As in [211], we experiment with three configurations of our model: (i) no contact point to condition on, (ii)

	L_2-norm (mm)				
	0.5s	1s	1.5s	2s	mean
Mao et al. [211]	26.2	45.5	67.5	96	47.8
STAG	**24.3**	**41.9**	**61.6**	**86,2**	**43.6**

Table 3.1: Distance between the predicted contact points and the ground truth ones.

conditioning on the predicted contact points, and (iii) on the ground truth ones. With the original configuration (i), we note a decline in performance, indicating that the second stage necessitates supplementary contextual information for accurate operation. This emphasizes the importance of considering the scene when predicting overall bodily motion. When the predicted contact points (ii) are added, *STAG* has a 16.2% more accurate prediction over the path error. Such improvement increases when considering the GT contact points (iii), reaching a 21% decrease in mean over path error. (iii) also highlights that having precise contact points coming from stage one can significantly improve the overall performance of the second stage.

Third Stage - Local pose error. As in the previous paragraph, we consider: (i) no contact point to condition on, (ii) conditioning on the predicted contact points, and (iii) on the ground truth ones. In this case, we outperform [211] in all settings, reaching 1.8% improvement in (iii) and 4% when considering GT contact points. It demonstrates how our body movement encoder is more capable of creating reasonable latent representations. While the improvement in pose may not be as pronounced as the improvement in path, it is crucial to consider the 3D nature of the scenario and ensure coherent body movements by accounting for the surroundings. With *STAG*, we observe an overall enhancement in both path and pose compared to the SoA methods. The staged pipeline assigns equal importance to both tasks, leading to these improvements.

	Path Error (mm)					**Pose Error (mm)**				
Models	**0.5s**	**1s**	**1.5s**	**2s**	**mean**	**0.5s**	**1s**	**1.5s**	**2s**	**mean**
LTD [215]	67.0	119.3	207.6	375.6	147.4	67.5	93.8	98.9	103.5	80.5
DMGNN [187]	82.7	158.0	227.8	286.9	156.2	47.5	69.1	85.6	95.3	64.9
SLT* [321]	**45.9**	117.0	186.7	267.1	121.8	70.8	181.4	150.2	196.0	112.6
Mao et al. [211] *w/o contact*	61.1	111.7	171.0	249.0	118.8	57.8	74.8	82.4	98.1	68.2
Mao et al. [211] *w/ pred contact*	58.0	103.2	154.9	221.7	108.4	50.8	67.5	75.5	**86.9**	61.4
Mao et al. [211] *w/ GT contact*	52.4	77.8	95.8	129.5	74.1	49.8	64.8	70.4	78.3	58.2
STAG w/o contact	64.0	133.0	210.4	302.0	141	55.8	72.9	82.8	96.2	67.1
STAG w/ pred contact	55.4	**89,6**	**127.9**	**179.3**	**92.3**	**48.1**	**65.3**	**75.6**	88.2	**60.3**
STAG w/ GT contact	50.3	65.1	70.1	99.2	60.0	46.9	61.5	68.0	76.3	55.6

Table 3.2: Path and pose error on the output obtained by pipelining the second and third stages on GTA-IM dataset.

3.4.2 Ablation study

We perform ablative studies to explore our model's components extensively. The results in Table 3.3 consider GT contact points and refer to the metrics used in Table 3.2. **stages** indicates the training mode of the second stage module: *2-stage e2e* means that stages two and three are learned in an e2e fashion, as is done in [211]; *2-stage ft.* indicates that stage two is pre-trained and fine-tuned during the training of stage thee; *3-stage (STAG)* is our proposed pipeline. **end** indicates whether the endpoint is used in the third stage. **TTG** is flagged if the proposed *time-to-go* is used in the third stage. Regarding how many joints are used to compute the contact points, we conducted a dedicated ablation study outlined in Table 3. The **cont.** column indicates which joints we consider for contact. With "all", every body part is considered to estimate contact with the scene. With "feet", only the feet can generate contact points, while with "feet, wrist", we consider contact points involving both the feet and hands.

The performance comparison in rows 1-3 reveals that even without the inclusion of end goals or TTG, the three-stage pipeline surpasses the performance of the two-stage pipeline. When end goals are introduced (rows 4-6), the performance gap becomes more apparent as they contribute to improved global performance. Lastly, using TTG in autoregressive prediction introduces time context and significantly enhances the results (rows 7-9). Moreover, it is preferable to consider the entire skeleton when calculating contact points, as the 3D scene is complex and involves multiple joints. Merely focusing on ground contact points (e.g., feet) or the most probable contact points (e.g., feet and hands) leads to unsatisfactory outcomes, as indicated in rows 10-13.

3.4.3 Comparison against global motion SoA models

Here we are testing the generalization of *STAG* to predict global motion without a given scene. The task aims to be comparable to other scene-free methods. The original version of *STAG* is evaluated under the assumption of a ground surface beneath the individual. This assumption is implemented by converting the floor into a scene representation as a 3D point cloud. Based on this information, the model estimates future contact points. It is worth mentioning that unlike competing techniques such as [3, 320], our model does not include multi-person joint forecasting or consider social relationships among individuals.

Dataset. We evaluate the performance of our model on additional datasets such as CMU-Mocap [63], which is widely used for absolute pose forecasting. The CMU dataset is captured at a rate of 30 frames per second (fps) using a marker system. Each sequence in the dataset consists of three individuals randomly selected from different scenes and merged together [320].

Comparison with state-of-the-art. Our model is compared to SoA approaches, among which are HRI [213], SocialPool [3], and MR-Trans [320]. HRI utilizes a motion attention mechanism to encode motion in both spatial coordinates and frequencies. SocialPool, on the other hand, is an RNN-based model that employs multiple GRU modules independently for each person in the scene, followed by a social module that considers the features of all individuals in the scene. MR-Trans, currently considered the SoA model, is a transformer-based approach that employs a discriminator to determine the suitability of pose and motion. Lastly, we also adapt Mao et al. [211] to the additional dataset *as is*.
The proposed approach performs similarly to the current best technique [320] in terms of overall error. However, when predicting future trajectories on the most challenging longer-term horizon, *STAG* slightly

					Path Error (mm)					Pose Error (mm)				
	stages	**end.**	**TTG**	**cont.**	**0.5s**	**1s**	**1.5s**	**2s**	**mean**	**0.5s**	**1s**	**1.5s**	**2s**	**mean**
1	2-stage e2e	×	×	all	55.8	77.7	87.5	121.5	71.3	48.8	64.15	70.7	77.9	57.8
2	2-stage ft.	×	×	all	53.6	72.5	83.7	115.8	68.4	48.8	64.2	70.8	77.9	57.8
3	3-stage (*STAG*)	×	×	all	53.4	72.4	84.1	117.8	68.5	48.8	64.2	70.9	78	57.8
4	2-stage e2e	✓	×	all	55.7	79.2	95.2	128.3	75.3	47.1	61.8	68.5	76.7	56
5	2-stage ft.	✓	×	all	51.9	68.7	78.1	113.1	65	47.1	61.8	68.5	76.6	55.9
6	3-stage (*STAG*)	✓	×	all	51.6	68.3	76.8	108.4	64	47.1	61.9	68.4	76.8	56
7	2-stage e2e	✓	✓	all	53.8	74.6	87.2	122.4	70	47.2	61.9	68.5	76.9	56
8	2-stage ft.	✓	✓	all	50.8	66.1	72.6	104.7	61.6	47.1	61.8	68.4	76.6	55.9
9	3-stage (*STAG*)	✓	✓	all	**50.3**	**65.1**	**70.1**	**99.2**	**60**	**46.9**	**61.5**	**68**	**76.3**	**55.6**
10	3-stage (*STAG*)	✓	✓	feet	55.1	79.9	96.7	136.5	76.2	47.5	62.1	68.7	77.3	56.3
11	2-stage e2e	✓	✓	feet, wrist	56.0	81.4	99.0	135.7	77.7	47.1	61.8	68.4	77.1	56
12	2-stage ft.	✓	✓	feet, wrist	55.6	82.0	102.0	140.0	79	47.1	61.4	68.9	79.9	56
13	3-stage (*STAG*)	✓	✓	feet, wrist	56.9	84.4	104.9	143.0	81.0	46.2	61.4	68.3	77.1	55.9

Table 3.3: Ablation study on the staged modeling.

underperforms compared to MR-Trans. On pose error, *STAG* outperforms the previous SoA model [320] by 33.5%, and by 8.8% with respect to [211] on the most challenging longer-term horizon.

	Path Error			Pose Error			Global Error		
Models	**1s**	**2s**	**3s**	**1s**	**2s**	**3s**	**1s**	**2s**	**3s**
LTD [215]	0.97	1.73	2.62	0.98	1.21	1.37	1.37	2.19	3.26
HRI [213]	0.96	2.06	3.11	1.05	1.37	1.58	1.49	2.60	3.07
SocialPool [3]	0.96	2.01	2.96	1.03	1.41	1.71	1.15	2.71	3.90
MR-Trans [320]	**0.60**	**1.12**	**1.71**	0.79	1.05	1.22	0.96	**1.57**	**2.18**
Mao et al. [211] *w/ pred contact*	0.78	2.19	3.99	0.59	0.93	0.95	1.01	2.47	4.16
STAG w/ pred contact	0.71	1.43	2.02	**0.57**	**0.76**	**0.87**	**0.95**	1.70	2.29

Table 3.4: Path, pose and global error in meters on CMU-Mocap dataset.

3.5 Conclusion

This paper has addressed the prediction of global pose in a three-dimensional environment as the staged modelling of three core elements: the scene, the human trajectory, and the pose. *STAG* is the first scene-aware global forecasting model which splits trajectory and pose motion to match the coarse-to-fine nature of the task. In fact, the pose of a person is the result of its motion pathway and the scene, rather than the cause of it.

STAG yields SoA performance on GTA-IM, the sole available for testing scene-aware global forecasting. *STAG* also sets the SoA on the CMU-Mocap dataset, under the assumption that the scene consists solely of a flat ground surface, therefore generalizing the task of global forecasting, which earlier methods have addressed without consideration of the scene.

Chapter 4

About latent roles in forecasting players in team sports

4.1 Introduction

Recent advances in visual recognition and sequence modeling have enabled novel objectives in athletic performance and sport analytics [220, 231, 267]. One novel and challenging task is the multi-agent trajectory forecasting (See Fig. 4.1) of the players as a result of their observed current motion [125, 183]. The difficulty is due to tactics, tight interaction of team players, the antagonist behavior of opponents, and the role assigned to each player in each action. Traditional trajectory forecasting techniques [9, 101, 114, 136, 226] fall short in performance due to their general formulations and lack of sport-specific dynamics. Furthermore, trajectory forecasting methods must deal with the variable numbers of people in each scene (usually absent in games) and do not consider the presence of two opposing teams, the ball, or the finality in the given sport (e.g. scoring). Most recent literature [125, 183] has started to address some of these objectives, but, to our knowledge, none has modeled the role of players for specific actions.

We propose RoleFor, a novel graph-based encoder-decoder model that performs a robust prediction of the players' future trajectory, utilizing roles to comprehend their interactions. The players' positions and movements on the court often follow pre-defined schemes, so we assume that each player may be assigned a specific role. By proposing a role-based ordering of nodes in the graph, it is possible to establish a player order and learn specific role-specific relationships.

The current best performers in-game forecasting [183, 226] are based on graph convolutional networks (GCN) [154], but they do not consider roles. On the contrary, we model latent roles as nodes in the graph. Our RoleFor model is composed of an ordering and a relational module. The former is an Ordering Network, which identifies latent roles and orders players according to them – we use a well-known sorting approximation [27] to order the latent projections of the players. In the latter, the game dynamics and trajectories are modeled using RoleGCN, based on [293] where the nodes are the newly assigned roles, and the edges are their relations. The adjacency matrix is learned, and each entry corresponds to learning the role-based player interaction.

We assume roles exist, and many characteristics could dictate them – e.g., marking the opponent, possessing the ball, and identifying the attacking and defending teams. However, we assume no prior knowledge about roles. Our goal is to learn latent roles with an end-to-end algorithm, only considering the future trajectory of all players. To test our intuition about roles, we pre-processed the basketball dataset by assigning

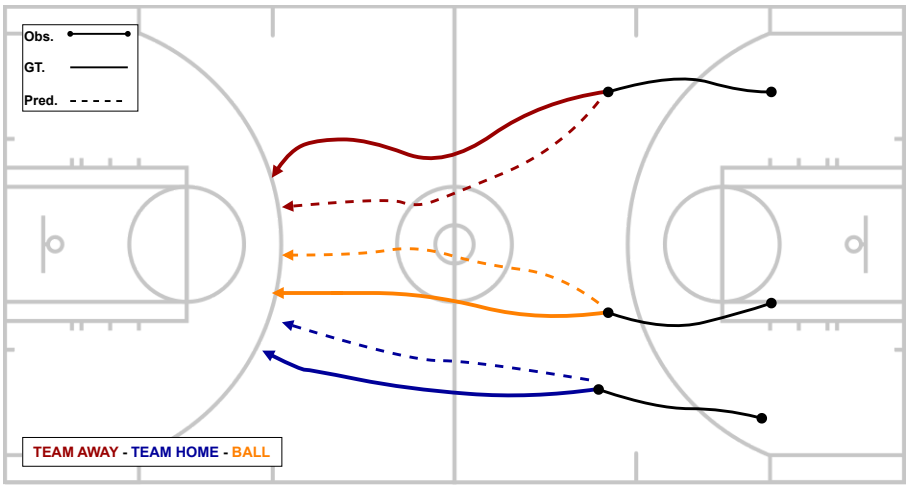


Figure 4.1: Example of multi-agent trajectory forecasting. We only plot one player for each team and the basketball for readability reasons.

roles based on different methods (Table 4.2) and using those in our RoleGCN. We produce SOTA results, confirming that finding good roles can improve model performances. Nevertheless, we found that current differentiable ordering methods face some limitations of backpropagation when inserted in complex models. In summary, our contributions are:

- We experimentally demonstrate that leveraging roles yields SoTA in trajectory forecasting.
- We propose an Order Neural Network module that creates a latent representation of the player's coordinates and orders them accordingly.
- We build a RoleGCN that learns the relations among roles.
- We empirically demonstrate that the current differentiable ordering approaches have some difficulties with backpropagation – enabling little to no gradients to flow through – when dealing with complex models.

4.2 Related Work

Trajectory Forecasting The forecasting of pedestrian movement has been studied to deal with realistic crowd simulation [249] or to improve vehicle collision avoidance [25]; it was also used to enhance the accuracy of tracking systems [62, 250, 337] and to study the intentions of individuals or groups of people [173, 333]. Different models have been proposed to predict such trajectories, like Long Short-Term Memory (LSTM) networks [131] with shared hidden states [9], multi-modal Generative Adversarial Networks (GANs) [114], or inverse reinforcement learning [157]. This group forecasting scenario resembles Game Forecasting, where it is necessary to model the movements of two opposing teams.

Game forecasting Associations such as National Basketball League or the English Premier League have used sophisticated tracking systems that allow teams to gain insight into each game [43]. Variational Autoencoders (VAEs) were used to model real-world basketball actions, showing that the offensive player trajectories are less predictable than the defense[84]. LSTM[283] were employed to predict near-optimal defensive positions for soccer and basketball, respectively, as for predicting the player's movements during

the game [125]. Variants of VAEs have also been used [297] to generate trajectories for NBA players. NBA player trajectory forecasting was also studied in [356] and [370], proposing a deep generative model based on VAE, LSTM, and RNN [125, 131, 142] and trained with weak supervision to predict trajectories for an entire team. Nonetheless, we did not encounter work estimating specific latent roles and learning the player interaction on those bases.

GCN-based forecasting Adopting a graph structure makes it possible to encode information and quantify shared information between nodes. SoA in pose forecasting learns specific terms for the specific joint-to-joint relation [293, 340]. Graphs are also widely used in trajectory forecasting and can be considered fully connected [183], sparse or weighted. These structures distinctly model the interrelationships between nodes, and their combination can be a crucial factor. Also, Graph attention layers (GAT) are widely used in trajectory forecasting [136, 184] to learn the inter-player dependencies. To model role-based interaction, we use the SoA pose forecasting model [293]. Pose forecasting is relevant since it considers the fixed node cardinalities and the learned interactions. However, players from various matches and teams do not have a fixed order, which is not an issue with pose forecasting. This encourages us to learn and re-order the players based on hidden roles.

Differentiable Ranking Sorting and Ranking are two popular operations in information retrieval that, in our case, can be useful in identifying the role of players. When used in composition with other functions, sorting induces non-convexity, rendering model parameter optimization difficult. On the other hand, the ranking operation outputs the positions, or ranks, of the input values in the sorted vector. As a piece-wise constant function, the computation of gradients is way more complex, preventing gradient backpropagation. Several recent works [27, 65] provide an approximation of the above operations to be used in a learnable framework.

4.3 Methodology

This section formally defines the problem and explains our strategy to tackle it, focusing on the role assignment and encoding methods. First, we briefly explain how the Role-based Forecasting model (RoleFor) performs latent mapping, role assignment, and trajectory prediction. We also focus on the main components: the Order Neural Network (OrderNN), which handles the ordering task, and the RoleGCN, which facilitates the learning process of relationships between roles in a game.

4.3.1 Problem Formalization

We target to predict the future trajectory of all players, given the observed positions at past time frames. We denote the players by 2D vectors $x_{p,t}$ representing player p at time t. The position of all players at time t are aggregated into a matrix of 2D coordinates $X_t \in \mathbb{R}^{2\times p}$. Motion history of players is denoted by the tensor $X_{in} = [X_1, X_2, ..., X_T]$, which is constructed out of the matrices X_t for frames $t = 1, ..., T$. The goal is to predict the future K players positions $X_{out} = [X_{T+1}, ..., X_{T+K}]$.

4.3.2 Role-based Forecasting model (RolFor)

RoleFor uses two main components, the first one being the OrderNN (Section 3.2.1), which orders players according to their latent roles. We postulate the existence of latent roles that when learned in an end-to-end architecture yield the best trajectory forecasting performance. From the OrderNN, we will consider R, the role vector, instead of P, the position vector. Notice that R and P have equal dimensions. The graph is now defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes indicate the roles of each player and the edges capture the interaction among roles during the game. The graph $\mathcal{G}$ has $|\mathcal{V}| = T \times R$ nodes, which represent all R roles across T observed time frames. Edges in $\mathcal{E}$ are represented by a Spatio-Temporal adjacency matrix $A^{st} \in \mathbb{R}^{RT \times RT}$, relating the interactions of all roles at all times. Note that A^{st} is learned, i.e., the model learns how players with different roles interact by learning how latent roles interact over time.

Order Neural Network

The Order Neural Network (Fig. 4.2) takes in input the initial coordinates X_{in} and maps them into a latent space. Additionally, it orders the latent vector into optimal roles X_{role_in}, thanks to the use of a differentiable ranking method [27], which has the same dimensionality of X_{in}. Note that roles get the corresponding position coordinates over subsequent time frames, so each role is now characterized by a spatio-temporal trajectory. A straightforward example of a role assignment involves sorting players in ascending order based on their Euclidean distance from the ball. This method is also used as a valuable proxy task, which we use for ablation studies (see Section 4.4 Table 4.2). However, since RolFor is trained end-to-end, OrderNN is free to learn the ideal ordering that yields the best forecasting performance.

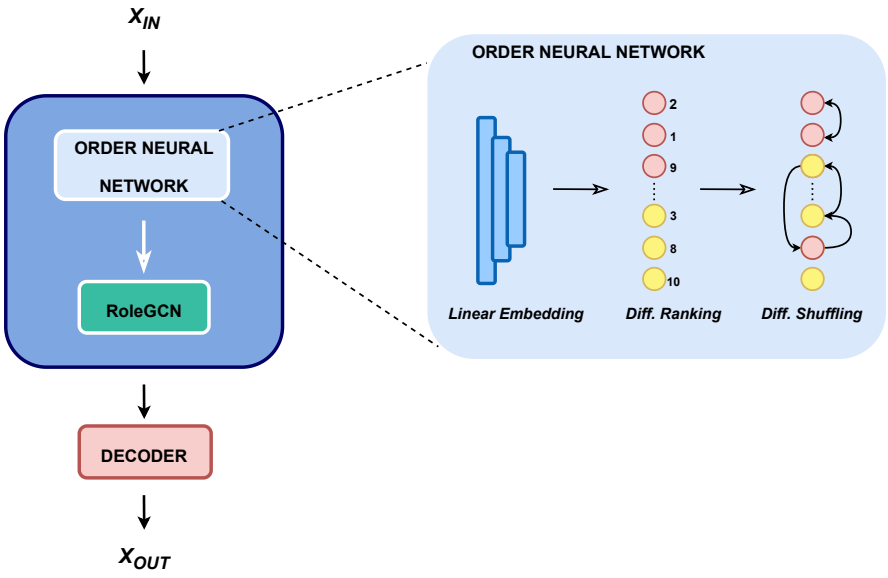


Figure 4.2: Architecture of RoleFor and a zoom into Order Neural Network

The differentiable ranking method SoftRank, [27] is a recent differentiable implementation of the classic sorting and ranking algorithm, empirically shown to achieve accurate approximation for both tasks. It

is designed by constructing differentiable operators as projections onto the permutahedron, i.e., the convex hull of permutations, and using a reduction to isotonic optimization. The key takeaway of the method is to cast sorting and ranking operations as linear programs over the permutahedron. More precisely, it formulates the argsort and ranking operations as optimization problems over the set of permutations Σ. SoftRank also relies on a regularization parameter ε, which creates a trade-off between the differentiability of the algorithm and the optimum's accuracy. The greater the regularization factor ($\varepsilon \to \infty$), the further the approximation will be from the permutation vertices, and the smoother the loss function gradient will be. And vice versa, by picking an $\varepsilon \to 0$, the algorithm will yield more accurate permutations with a lower degree of differentiability. After learning the ranking, we order the players according to it by employing a differentiable re-shuffling module. The outputs of SoftRank are noted as $\{s_i\}_{i=1}^{n}$ where *n* is the number of rankings considered. At this point, we use a so-called *base* matrix **B** with the number of rows and columns equal to the number of rankings. **B** will be used to store the real rankings $\{p_i\}_{i=0}^{n}$. We then compute a $\{\Delta_i\}_{i=1}^{n}$ matrix, which represents $\Delta_i = p_i - s_i$ for each position $\{p_i\}_{i=0}^{n}$. The matrix Δ is used as the input as a rescaling function. The re-shuffle process is a weighted combination: it yields a real shuffling when the approximated rankings are integer and a differentiable shuffling instead when the ranking is fractional. $M_i = e^{(\frac{-\Delta}{scale})^2}$ can be considered an array of weights for each position, with values closer to 1 being the predicted positions of each player. Finally, this will be used to recall the initial coordinates in an ordered manner:

$$P_i = \begin{cases} x'_i = \sum_{j=1}^{n} M_j \cdot x_j \\ y'_i = \sum_{j=1}^{n} M_j \cdot y_j \end{cases} \tag{4.1}$$

RoleGCN

Once the latent roles are inferred, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents each node $i \in \mathcal{V}$ as the player's role while the edges $(i, r) \in \mathcal{E}$ connect all the roles and describe their mutual interaction. RoleGCN (Fig 4.2) will capture the underlying graph's relationships, both between different nodes on the court in the same time frame and between one node and itself over different time-frames. GCN [154] is a graph-based operation that works with nodes and edges. For nodes, it aims to learn an embedding containing information about the node itself and its neighborhood for each node in the graph. Thus, the learned adjacency matrices yield a quantitative description of the interplay among roles. The space-time cross-talk is realized by factoring the space-time adjacency matrix (as in [293]) into the product of separate spatial and temporal adjacency matrices $A^{st} = A^S A^t$. A separable space-time graph convolutional layer *l* is written as follows:

$$H^{(l+1)} = \sigma(A^{s-(l)} A^{t-(l)} \mathcal{H}^{(l)} W^{(l)}) \tag{4.2}$$

It is similar to a classic GCN convolutional layer, where $A^{s-(l)} A^{t-(l)}$ is the factorized matrix $A^{st-(l)}$ of a GCN [154] layer. The critical difference is better efficiency and allows full learnability of the former.

Decoder

First, we de-shuffle the permuted roles according to the inverse of **B** to return to the original coordinates' position. The decoding is done with multiple temporal convolutional (TCN) layers [132] used to predict the following frames. We adopt TCN due to its performance and robustness.

4.4 Experimental Evaluation

In this section, we introduce the NBA benchmark dataset and metrics, the trajectory forecasting results and investigate why learning E2E roles is challenging.

4.4.1 Dataset

For our experiments, we use NBA SportVU [84]. It contains players and ball trajectories for 631 games from the 2015-2016 NBA season. Similar to previous work [297], we focus on just two teams and consider all their games. We obtain a dataset of $95,002$, 12-second sequences of players and ball overhead-view trajectories from 1247 games. Each sequence is sampled at 25 Hz, has the same team on offense for the entire duration and ends in a shot, turnover, or foul. As in [84], the data is randomly split into train, validation, and test sets with respectively $60,708$, $15,244$, and $19,050$ sequences.

4.4.2 Trajectory Forecasting Metrics

We use as metrics ADE (Average Displacement Error) and FDE (Final Displacement Error), as usual in literature [9, 84, 114, 125, 183]. They are used to measure the error of the whole trajectory sequence and the final endpoints for each player. Respectively:

$$ADE = \left\| \hat{T}_c - T_c \right\|_2^2 \tag{4.3}$$

$$FDE = \left\| \hat{E}_f - E_f \right\|_2^2 \tag{4.4}$$

Each observation has *five* frames, which correspond to 2.0 seconds in a basketball scenario. The goal is to forecast the successive *ten* frames (4.0 seconds). In Eq. 4.3, $\hat{T}_c$ represents the prediction for all future trajectories over the $c = 1, .., 10$ subsequent frames, and T_c is the ground truth. The same nomenclature is used in Eq. 4.4, where E is the matrix for the endpoints and $c = 1$ since we are only considering the last frame.

4.4.3 Trajectory Forecasting Results

So, do roles exist, and does learning the role interaction yield state-of-the-art performance? We answer this question by considering the most straightforward ordering: Euclidean distance of players from the ball. In Table 4.1, we report state-of-the-art techniques compared to the RolFor model, with the Euclidean distance ordering of players from the ball. [183] proposes multiple predictions via latent interaction graphs among multiple interactive agents. [114], similarly, is also a multi-modal model incorporating the social aspects of the players as well. [136] is based on a sequence-to-sequence architecture to predict the future trajectories of players. Lastly, [226] substitutes the need for aggregation by modeling the interactions as a graph. Similar to [340], it needs a pre-defined graph, allowing the leaning procedure only on the given edges. RolFor in Table 4.1 yields the SoA forecasting performance in terms of ADE, 5.55 meters, second best in terms of FDE, 9.99 meters. It sorts players according to their Euclidean distance from the ball, arranging them into a sequence of attackers (players detaining the ball in the considered action), alternating with defenders (not detaining the ball). Each attacker is followed by its marker, which RolFor considers the closest to it in terms of Euclidean distance. As for all other reported SoA algorithms, RolFor considers that the teams are known.

Finally, "Oracular Permutation" means that RolFor uses distances at the last future step, i.e., step 10 in the future. In contrast, any other reported algorithm uses only the observed five frames. We will investigate this more thoroughly in the next section. A neural network can learn the Euclidean distance, and softRank [27] should be able to sort the players according to it. Replacing the hand-defined distance computation with a Neural Network should be as effective. We expect that a model with a sorting unit that learns sorting E2E in relation to the final forecasting goal should be capable of doing better than this, assuming all modules were effectively differentiable.

Table 4.1: Comparison of our model with SoTA models

Model	ADE	FDE
EvolveGraph [183]	5.73	8.65
Social-STGCNN [226]	6.42	10.04
STGAT [136]	7.06	12.54
SGAN [114]	5.88	10.36
RolFor + Oracular Permutation	**5.55**	9.99

Further Experiments on Euclidean Ordering

We delve deeper into the results of RolFor in Table 4.1 and analyze the importance of each hand-defined Euclidean distance term in Table 4.2.

No ordering Vs. Simple ordering. The first forecasting result in the table neglects the player ordering and learns interaction terms between players, arranged in random order. It yields 6.34/11.5 ADE/FDE meters errors. Simple ordering stands for arranging all players in a list, according to their distance from the ball, at the last (5th) observed frame. This uncomplicated ordering is only negligibly better than no order. A GCN model may deal with players in random order well and only benefits from ordering if it is informative.

Distance from the ball and marking. Results in the third row of the Table 4.2 add marking to the ball distance ordering. Each player in the attacker team is matched with one from the defender team according to Euclidean distance. Performance improves in ADE, from 6.31 to 6.16 meters, and slightly degrades in terms of FDE, from 11.1 to 11.28. Overall All distances are computed at the last observed frame. Furthermore, all distances are plain Euclidean distances, which a simple Neural Network may replicate or improve with E2E learning.

Distance from the ball and marking at future frames. The last row of Table 4.2 considers the furthest future frame position for all distance computations. It should be noted that the model makes no assumptions about future locations. Future information is simply utilized to place players in order. This motivates us to replace the hand-defined ordering with an E2E-trained module, which we will do in the following section.

Table 4.2: Results for different types of ordering

Ball Dist.	Obs.	Future	Mark	ADE	FDE
-	-	-	-	6.34	11.5
✓	-	-	-	6.31	11.1
✓	✓	-	✓	6.16	11.28
✓	-	✓	✓	5.55	9.99

4.4.4 End to end Model with latent roles

In this section, we leverage the full RolFor model, E2E trained. Here the first module, OrderNN, sorts players into their roles in the action, then the RoleGCN module reasons on their role-based interaction. Sorting into roles has benefited forecasting in Sec. 4.4.3. Here we assume that roles are latent variables, which the OrderNN estimates, E2E, based on the best forecasting performance. In Table 4.3, we compare the hand-defined baseline (ball and marking distance on the last observed frame, scoring 6.16/11.28) against E2E model variants. *E2E* is learning to order, encode the role-role interaction, and forecast based on the encoder. This model is performing poorly at 12.12./15.02 ADE/FDE. Is this because the OrderNN is incapable of ordering, or is it because the OrderNN is not fully differentiable? Moreover, the *EuclDistEst* variant attempts to answer part of this question. Here we used a pre-trained Neural Network module to approximate the Euclidean distance based on the player's performance. We then use the pre-trained module to sort players according to the ball. If the Euclidean distance estimator model were perfect, performance would be 6.31/11.1 (ADE/FDE), cf. Table 4.2. *EuclDistEst* yields, however, 7.50/12.58. We attribute this mismatch to the residual errors in the Euclidean distance estimation, which, as it seems, matters. More surprisingly, *E2E-finetune* starts from the *EuclDistEst* variant, and it fine-tunes it, E2E. The error increases to 12.08/14.97, so the model neglects the initialization and reverts to the *E2E* performance. We attribute the discrepancy between *EuclDistEst* and *E2E* to the challenges in the SoftRank differentiability, as we further analyze in the next section.

Table 4.3: Different training configurations for RolFor

Model	Configuration	ADE	FDE
RolFor	*E2E*	12.12	15.02
RolFor	*E2E-finetune*	12.08	14.97
RolFor	*EuclDistEst*	7.50	12.58
RolFor	*Best non-or. dist.*	6.16	11.28

Table 4.4: Analysis of simulated errors in ordering

Model	ADE	FDE
Oracular Ordering	5.55	9.99
Light Swap	6.55	12.10
Light Insert	6.55	12.10
Light Swap + Light Insert	6.59	12.08
Heavy Swap + Heavy Insert	6.71	12.25

Analysis of the Order Neural Network

Here we focus on confirming our claims on the issues of the differentiability of Softrank. We set to order the players according to their ascending distance from the ball, at a specific frame, given their 2D coordinates. It allows us to test the first RolFor module, OrderNN, in isolation, cf.4.5. In Table 4.5, we compare *OrderNN E2E* against *OrderNN EuclDistEst*. The first E2E trains the order of players and re-shuffles them. The second supervises the network by tasking it to learn the Euclidean distance between the players and the ball and then sort the distances according to SoftRank. We measure the ordering accuracy p_{ord} as the percentage of players the models place in the correct order. In other words, we reproduce the top-k classification experiment as [65]. The authors propose a loss for top-k classification between a ground truth class $ord \in [n]$

and a vector of soft ranks $\hat{ord} \in \mathbb{R}^n$, which is higher if the predicted soft ranks correctly place y in the top-k elements.

Table 4.5: OrderNN *E2E* against OrderNN *EuclDistEst* top-k accuracy

Model	top-k	Accuracy
OrderNN *E2E*	10	1,00%
OrderNN *EuclDistEst*	10	71,00%
OrderNN *EuclDistEst*	5	77,00%
OrderNN *EuclDistEst*	3	82,00%
OrderNN *EuclDistEst*	1	92,00%

Observe from Table 4.5 that learning Euclidean distances from 2D positions is an easier task for a deep neural network since SoftRank yields 71% at the *top-10* ordering accuracy p_{ord}. It is also interesting to notice that when changing the *top-k* ordering accuracy into $5, 3, 1$ we get similar results to [27]. By contrast, learning the ordering E2E from the 2D coordinates yields surprisingly low performance. Note in the table that *OrderNN E2E* achieves a *top-10* ordering accuracy of only 1%.

4.4.5 Robustness of RolFor to ordering errors

How much does misordering impact forecasting? We measure ADE and FDE forecasting errors when randomly altering the order provided by our best performing oracle RolFor (5.55/9.99 ADE/FDE, Table 4.2). In more detail, we consider the swap of two players *Light Swap*, which can occur if the distance between them is relatively small. A more significant error can also occur, e.g., one role is not identified correctly and a player is inserted at the wrong position, making the whole order slip. We name this *Light Insert*. In Table 4.4, we consider the two potential sources of errors by randomly simulating one or both. The results are coherent with what we said previously Table 4.3, where the RolFor *EuclDistEst* has a *top-10* ordering accuracy of 71% yielding 7.50/12.58. At the same time, a Light Swap/Insert gives 6.55/12.10 in ADE/FDE and 80% *top-10* ordering accuracy. This last Table 4.4 highlights the importance of roles and their impact on the final trajectory accuracy.

4.5 Conclusions

Our goal was to show that roles and social relations in sports are quantifiable and can be effectively used to improve the current SoA models in game forecasting. We demonstrate that roles exist by testing different permutations over players. Then, we encode the player's coordinates into a latent space and use the encoding to find an optimal latent role ordering. The model employed to perform trajectory forecasting is called RolFor (Role Forecasting) and considers the input nodes of a graph indicating roles in a game. This single-graph framework favors the relation between roles and time, allowing better learning of the fully-trainable adjacency matrices for role-role and time-time interactions. The adoption of CNNs and the graph structure of the input allows the requirement of parameters to be only a fraction of the ones used in Transformers, GANs, and VAEs. Our observations emphasize the significant opportunity for future work to develop fully differentiable ordering modules to enable learning latent role-based interactions in graph-based models, also applicable to social networks and multi-agent systems.

Acknowledgements This work was partially supported by the MUR PNRR project FAIR (PE00000013)

Chapter 5

Following the Human Thread in Social Navigation

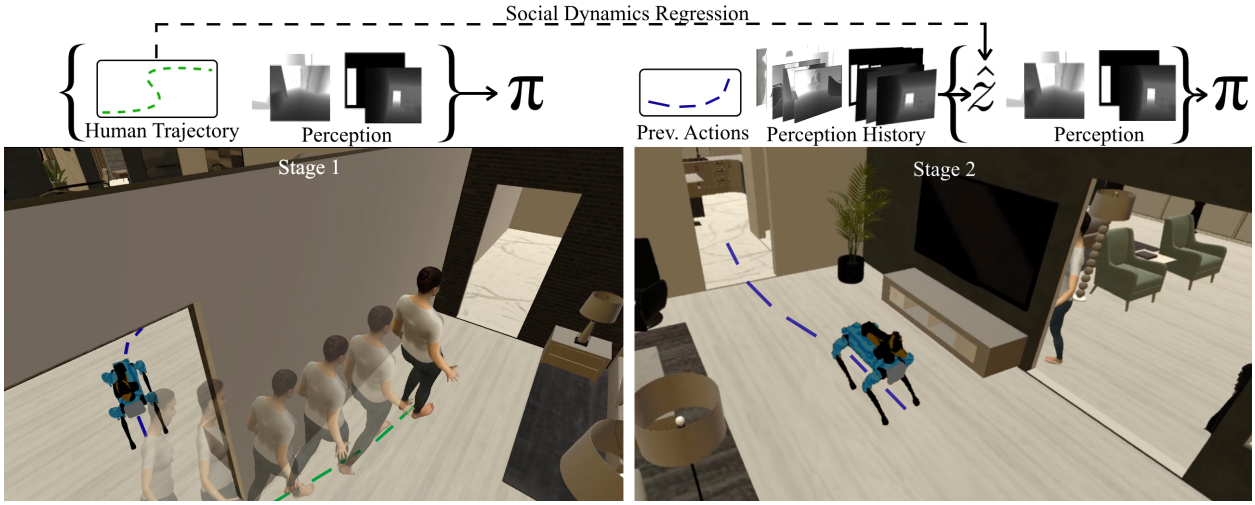


Figure 5.1: We present our novel Social Dynamic Adaptation model (SDA). The framework involves two stages of training that allow the model to infer, given its past observations and actions, another agent's Social Dynamics. In the first training stage, the model embeds the followed agent's trajectory, which, together with sensor perceptions, compose the input to the model's Social navigation Policy (π). The knowledge obtained from the human trajectory strongly helps the navigation policy in finding and following an agent. However, this information is often not available during deployment. In the second stage, SDA learns to adapt past statuses and actions, which are always available, to the first stage's Social Dynamics embedding $\hat{z}$. As depicted in the figure, the status contains depth maps and BB detection of the person, if observable from the egocentric robot view. $\hat{z}$ is then paired with current observations as input to the frozen π.

5.1 Introduction

Traditional navigation techniques within Embodied Artificial Intelligence (EAI) have marked a crucial advancement by introducing robots into real-life environments. However, these techniques have primarily focused on agents traversing vacant spaces. Conversely, the significance of social navigation within EAI has steadily increased. Social navigation entails agents' capacity to navigate human-centric environments while considering human movements and behaviours. These agents need to be able to locate, track, and interact with humans in a safe and socially acceptable manner. Previous studies predominantly characterized So-

cial Navigation as a variation of PointGoal Navigation, wherein agents strive to reach specified destinations while considering human movements [242, 331, 342]. Habitat 3.0 [255], a significant breakthrough in EAI, introduces a lifelike environment seamlessly incorporating human avatars. This integration enables investigating human-agent interactions within a controlled, risk-free, dynamic environment. What sets Habitat 3.0 [255] apart is its ability to replicate complex scenarios where human intentions are constantly changing. Nevertheless, this dynamism also presents particular challenges, such as collision avoidance and achieving success in locating and following humans.

Despite notable efforts in collision avoidance and safety, most existing methods for Social Navigation either rely on privileged information that is not available in real-world scenarios or do not adequately capture the social dynamics and norms of human behaviour. For instance, [331] and [342] use a GPS and compass sensor to provide the agent with perfect localization, which might be unrealistic even if using SLAM [82, 232] methods, whenever the human is not in the line of sight. While [242] and [346] do not account for the social factors that influence human behaviour. Therefore, this limitation hinders their practical applicability and adaptability to dynamic environments. [38], instead, models some social factors in the form of Proximity Tasks but fails to account for the cooperative nature that a social agent must possess, restricting itself to merely avoiding collisions with them. The current SoA model proposed in [255] necessitates privileged information, such as humanoid GPS, which offers polar coordinates, detailing the accurate distance and angle of the human from the robot, to attain high-performance outcomes. However, this requirement is highly impractical in real-world environments during inference.

This paper proposes a novel Social Dynamics Adaptation model (SDA), shown in Fig. 5.2, that effectively solves the robot's awareness of complex human behaviors, even temporarily losing sight of the person and fast robot motion. Specifically, the first stage trains a base policy considering human trajectories encoded into a latent vector. The latent vector is a low-dimensional, nonlinear projection of the human trajectories, and it is trained end-to-end with the base policy to extract the social factors that led the robots to choose better actions. The subsequent supervised adaptation stage regresses this latent vector using only the robot's state and action history. Unlike previous methods, such as [331] and [346], which often depend on simulated privileged information or simplified social behavior models, our approach adapts to dynamics resembling real-world conditions in real-time. In summary, SDA adapts and accounts for unpredictable human behaviors by exploiting privileged information during training and recovering this fundamental signal during deployment when it is often impractical to compute. Finally, the deployed robot features the motion policy, learned in the first stage, and the social dynamics, inferred from prior statuses and actions.

Out of extensive benchmarking, SDA outperforms the approach proposed in Habitat 3.0 [255] and a second adapted best-performing method [38] from Habitat 2.0 [300]. We conduct a thorough experimental evaluation of the core contribution of this work—learning to infer social dynamics from (privileged) information about the person. Although our primary focus is on algorithm development, we also seek to improve the robustness of our method for potential real-world applications. To this end, we conduct experiments that bridge the gap to real-world scenarios by introducing noise to the input, reducing the refresh rate of the sensors, and modifying the simulator to reflect more realistic human behavior. Our ablative studies reveal that human trajectories are not only strong input information for the robot control policy but also provide better supervision for inferring the social dynamics latent to the same policy. Other oracular information, such as the humanoid GPS (direction and distance from the person to the robot), serves as powerful sensors for the control policy but does not facilitate adaptable social dynamics for inferring human-robot-scene interactions.

5.2 Related Work

Embodied Navigation. Recently, there has been a surge in exploring indoor navigation within an embodied framework [70]. This upswing has been facilitated primarily by the availability of large-scale datasets comprising 3D indoor environments [44, 263, 286] and simulators designed for simulating navigation within these dynamic 3D spaces [162, 276, 286]. Nevertheless, these simulators are not equipped to handle human entities within the environments, restricting the investigation to navigation tasks in scenarios where the agent functions independently or, at most, alongside humans simulated via static meshes that simulate movement [332, 346]. These simulated humans are treated as dynamic obstacles and lack compliance with any social construct. This constraint has been effectively addressed with Habitat 3.0 [255], the simulator used for this research. Habitat 3.0 introduces the capability to simulate the behaviours of humans engaging in tasks within dynamic environments, thus overcoming the limitations mentioned above.

Thanks to these simulators the realm of EAI has witnessed the introduction of numerous tasks [70], including PointGoal Navigation [331], ObjectGoal Navigation [20], Embodied Question Answering [330], and Vision and Language Navigation (VLN) [11, 167]. Various modular approaches [37, 47, 48, 262, 266] have been proposed to address the challenges of navigating through static, single-agent environments. These approaches utilize maps constructed from depth images and conduct path planning directly on these maps. However, these approaches are unsuitable in social settings where dynamic objects (humans) move within the environment. This is because humans are observable only within the agent's field of view (FOV). Moreover, the agent must address the additional challenge of tracking and mapping. End-to-end RL-trained policies [36, 242, 325, 342, 343], should be adapted to learn also social clues similarly to [38], where the agent learned proxemics information about the humans moving in the environments thanks to two proximity tasks. In this paper, instead, we try to learn social behaviours directly by internally modeling the humanoid trajectories in the latent representation of the agent. Furthermore, differently from [38], the agent's aim is no longer just avoiding collisions. Still, it involves locating this dynamically acting human and following them for a specified number of steps while maintaining a safe distance. This evolution of the task demands a heightened level of social comprehension from the agent, requiring the anticipation of the person's intentions and the ability to trail them closely without compromising safety.

Socially-Aware Navigation. Research in robotics, computer vision, and the analysis of human social behavior explored socially aware representations and models [229]. Extending from the realm of collision-free multi-agent navigation [24, 55, 198, 312] and navigation in dynamic environments [14], researchers have further expanded their investigations to encompass scenarios involving human presence [49, 54, 87, 116, 202].

The approach presented in [54] incorporates collision avoidance algorithms like CADRL [55] while introducing common-sense social rules. This integration aims to reduce uncertainty while minimizing the risk of encountering the Freezing Robot Problem [308]. Other works [49, 87] seek to model human-agent interaction by employing techniques such as spatiotemporal graphs [202].

Typically, these methods undergo testing in a minimal number of environments that offer complete knowledge about the human positions and velocities [49, 55], featuring simple obstacles and often assuming collaboration between moving agents[49, 87]. In contrast, our focus revolves around SocialNav within expansive indoor environments, characterized by partial knowledge about them, since the agent perceives the environment only through its sensors from an egocentric perspective, and no information about the velocity or position of the human is given to the agent. Our SDA addresses the missing information from

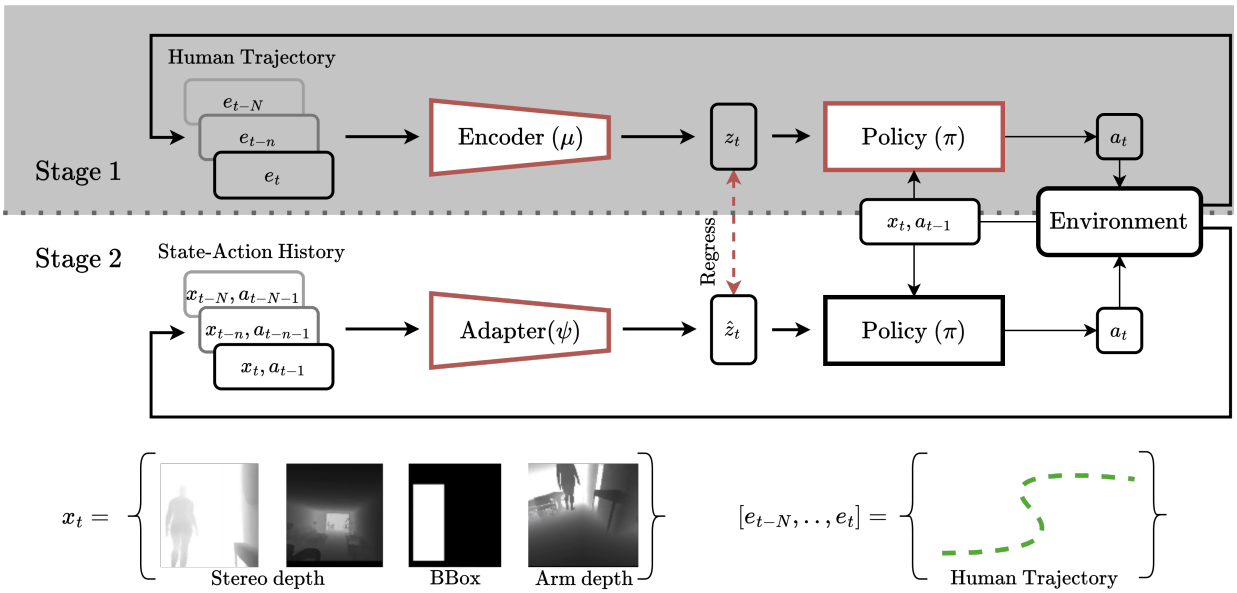


Figure 5.2: Pipeline of the novel methodology proposed. First, we jointly learn to encode human trajectories and a motion policy. In the next stage, given the previous states and actions, we infer the social dynamics and pass the estimated latent vector to the frozen policy.

the interacting human position, inferring it from the robot's history of actions and status.

Modelling dynamic environments. Simulated environments [179, 254, 309] offer privileged information about the scene whose exploitation can be computationally intense or unfeasible during deployment. While navigating social environments, it is vital to take into consideration human behaviour [158]. Ideally, one would want to forecast people's position for better path planning [244], but forecasting robot-person interactions is significantly slower [261] than navigation policies, hence being challenging to be considered for training or deployment. To overcome this problem, we leverage literature on system identification [4, 113] to infer the encoded privileged information during robot navigation. Once encoded in a latent space during the first training stage and used to train the primary policy, it is possible to asynchronously derive that same information from the state-action history [171, 172, 192, 200, 256, 364], influenced by the signal we want to identify. Unlike previous works, we are the first to identify the social dynamics (under the form of human trajectories), with the intuition that modeling human behavior is fundamental for efficient human-robot collaboration.

5.3 Methodology

This section introduces SDA: a novel framework for social navigation that incorporates human trajectories into the sequential decision-making process. The first stage of our approach focuses on encoding human trajectories into a latent social context to represent the social dynamics that are functional to the agent motion policy. In the second stage, we introduce the Adapter module, which enables the estimation of social information from the agent's past behavior. Recovering this signal allows the robot to operate without explicitly representing human behavior. We detail the training process, trajectory modeling, and optimization techniques utilized in our approach, highlighting its effectiveness in addressing the challenges of social navigation.

Problem formalization. The problem requires to locate and follow a humanoid in motion within an indoor environment, maintaining a distance of 1 to 2 meters for at least k steps [255]. With the status x_t, we represent the agent's "perception" at time t. We exploit depth images from different cameras placed on the robot and a preprocessed version containing a humanoid detection bounding box. These perceptions are processed with a ResNet [127] before being fed to the recurrent policy network, selecting the best action a_t at time t. We use Decentralized Distributed Proximal Policy Optimization (DD-PPO) [331] to iteratively improve the agent's policy while maximizing rewards derived from interactions with several environments executed in parallel, ensuring stability through controlled policy updates and lower training times.

The pipeline described above can be considered a baseline implementation that does not contain privileged information but relies only on the robot's onboard sensors. To address the social aspect required by the task, we consider additional details on the humanoid, e.g., "social dynamics", defined as $e_{t-N:t-1}$, where N refers to the trajectory length. e_{t-n} represents the position n steps before time t, and e_{t-N} refers to the earliest position in the trajectory under consideration. The notation $e_{t-N:t-1}$ is used as a shorthand to indicate the complete trajectory from N steps in the past to the current time, representing absolute x-y coordinates. In the following sections, we describe how exploiting this information at training time can improve performance during deployment when it is absent.

Stage 1: Social Policy. Recurrent policies such as DD-PPO take as input the current status, in our case, a collection of depth images processed via a Resnet x_t, and the action at the previous time-step a_{t-1}. We add another input to this pipeline, namely a latent vector z_t built by encoding the humanoid privileged information $e_{t-N:t-1}$:

$$z_t = \mu(e_{t-N:t-1}) \tag{5.1}$$

$$a_t = \pi(x_t, a_{t-1}, z_t) \tag{5.2}$$

Intuitively, by training everything with the same objective z_t encodes the social dynamics that led the agent to maximize its reward, adapting to human movement patterns. Additionally, including trajectory data allows the agent to learn from past interactions and experiences. In our approach, the trajectory encoder (μ) is implemented as Multilayer Perceptrons (MLPs). The objective retains its usual formulation without any explicit reference to the human trajectories:

$$L^{CLIP} = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{5.3}$$

where $r_t(\theta)$ is the ratio of the probability of action a_t under the current policy and the previous one that is being executed for gathering data. $\hat{A}_t$ represents the advantage function at time t, guiding the policy towards actions that yield higher expected rewards.

Defining what information is considered "privileged" and why is essential. In our context, it refers to detailed knowledge about the humanoid in the environment, such as the exact position of humanoids defining a trajectory (traj.), or the relative position with respect to the agent often denoted as humanoid GPS [255](hGPS). We can easily gather this information in simulated environments. However, collecting them in real-life scenarios is often impractical. This distinction is crucial as it highlights the challenge of transferring learned policies from simulation to the real world.

Stage 2: Social Dynamics Regression. We aim to extract and exploit social cues directly from the robot's perception and eliminate the need for auxiliary devices like GPS trackers on humanoids. Inspired by [171], we introduce the "social dynamics" module (Adapter), parametrized by an MLP ψ that takes as input the recent history of the robot's states $x_{t-N:t-1}$ and actions $a_{t-N:t-1}$ to generate a new latent vector $\hat{z}_t$:

$$\hat{z}_t = \psi(x_{t-N:t-1}, a_{t-N:t-1}) \tag{5.4}$$

We obtain the state-action history by deploying the agent in the environment with optimal policy π^* obtained after the first stage and the latent vector $\hat{z}_t$:

$$a_t = \pi^*(x_t, a_{t-1}, \hat{z}_t) \tag{5.5}$$

During this process, we optimize the Adapter, MLP, with a supervised regression objective, Mean Squared Error (MSE), to recover the original information contained in z_t that we compute relying on the preferential information trajectory:

$$\text{MSE}(\hat{z}_t, z_t) = \|\hat{z}_t - z_t\|_2^2 \tag{5.6}$$

Once we finalize the Adapter training, instead of relying on the privileged information, we can depend upon the robot's states $x_{t-N:t-1}$ and actions $a_{t-N:t-1}$ to generate $\hat{z}_t$, serving as an estimate of the actual latent social dynamics vector z_t. Doing so enables the agent to estimate social dynamics online, improving its performance in dynamic environments and enhancing its social navigation capabilities, freeing it from dependence upon external sensors.

5.4 Results

Section 9.7.2 outlines our findings on Social Navigation, along with an ablation analysis of adaptable information. Additionally, Section 5.4.2 offers a qualitative examination of our results, and an analysis of the role played by the latent vector $\hat{z}_t$. A more detailed analysis and further qualitative results can be found in the Appendix.

Simulator. We tested SDA on Habitat 3.0 [255], a simulation platform designed for human-robot interaction within domestic settings. This platform offers precise humanoid simulation capabilities with a focus on collaborative tasks such as Social Navigation and Social Rearrangement. It offers a vast library of avatars featuring multiple genders, body shapes, and appearances. Furthermore, it employs an oracle policy to generate movement and behavior, enabling programmable control of avatars for navigation, object interaction, and a range of other movements.

Baselines. The baselines we employ in our study are drawn from Habitat 3.0 [255] and consist of:

- **Heuristic Expert**: a heuristic baseline equipped with access to the environment map, employing a shortest path planner to devise a route to the current location of the humanoid. The heuristic expert operates on the following principles: When the agent is beyond a distance of 1.5 meters from the humanoid, it employs a "find" behavior, utilizing a path planner to approach the humanoid. Conversely, if the humanoid is within 1.5 meters, the expert executes a backup motion to prevent collision with the humanoid.

- **Baseline**: the current SoA method [255], a recurrent neural network policy trained with DD-PPO [331], operates on a "sensors-to-action" paradigm. Inputs to this policy consist of an egocentric arm and stereo depth sensors, a humanoid detector, and humanoid GPS coordinates, while the outputs are velocity commands in the robot's local frame. Table 5.1 compares our model in a realistic configuration, where the humanoid GPS data are unavailable.
- **Proximity tasks**: we also adapted the Proximity Tasks defined in [38] and applied them to the baseline [255]. These tasks were proposed for a different setup of SocialNav, where the agent acts in an environment with multiple humanoids and must navigate from point A to point B while avoiding collisions. We adapted the risk and compass proximity tasks to the SocialNav setting addressed in this article. In this context, the risk has a low value (close to 0) when the agent is within 1 to 2 meters from the humanoid and a value close to 1 when the distance is less than 1 meter or greater than 2 meters. Similarly, given the presence of only one humanoid in the environment, the compass was redefined to predict the angle between the humanoid and the agent. This adjustment aims to assist during the following phase, enabling the agent to follow the humanoid while maintaining a safe distance and staying aligned with the human. The proximity tasks are jointly trained with the policy and detached during evaluation.

Metrics. We used the metrics for the SocialNav task as defined in [255]. *Finding Success (S)* is the ratio of the episodes where the agent located and reached the human. *Finding Success Weighted by Path Steps (SPS)* measures the optimality of the path taken by the agent wrt the optimal number of steps needed to reach the human. *Following Rate (F)* is the ratio of steps during which the robot maintains a distance of 1-2 meters from the humanoid while facing towards it relatively to the maximum possible following steps. *Collision Rate (CR)* is the ratio of the episodes that ended with the robot colliding with the humanoid. *Episode Success (ES)* measures the ratio of the episodes where the agent found the human and followed it for the required number of steps, maintaining a safe distance in the 1-2 meters range.

Privileged information. In our work, the privileged information under consideration includes humanoid GPS (hGPS) and human trajectories (traj.). Humanoid GPS is represented in polar coordinates, a method of specifying a point's position in a plane using two parameters: the distance from the point to the origin (radius) and the angle formed between a reference direction (typically the positive x-axis) and a line connecting the origin to the point (polar angle or azimuth). In our context, the origin is defined as the robot's position; thus, its position is implicitly known along with that of the human. Conversely, trajectories solely consist of information derived from the human's position within the environment.

5.4.1 Quantitative Results

In Table 5.1, we present the results obtained in Habitat 3.0 [255] for the Social Navigation task. The table is divided into three sections. The first one displays outcomes achieved by utilizing a heuristic expert endowed with extensive information, including its position, map data, and the humanoid's position, granting it a competitive advantage over other methods. The subsequent section features models trained and tested using ground truth (GT) data (Baselines and Stage 1), thus establishing an upper limit for techniques utilizing more practical inputs feasible in real-world scenarios. Lastly, the final rows delineate results from methods conducting inference without privileged information. The models that use privileged information (GT) show

that S1 has lower performance than the two Baselines, especially in S and SPS. This is to be expected, considering that while trajectories solely depict the movement of the human in the environment, hGPS furnishes crucial details on locating the humanoid, offering insights into the distance and angle between them. In the second stage, despite the absence of human trajectory input, SDA keeps the performance level of Stage 1 by adapting to the social dynamics. Our model outperforms the baselines in the find task, increasing S and SPS by 6%.

Our approach generally improves performance in the episode success (ES) metric, which occurs when the agent finds the humanoid and follows it for 400 steps. However, we emphasize that the test-time episode ends at 1500 steps or when the agent collides with the humanoid [255], not after the 400 follow steps. In this context, S, SPS, and F metrics demonstrate how SDA, compared to the Baseline, more frequently locates the humanoid, follows a more optimal path (on average 438 vs. 540 steps), and follows it for a longer duration (390 vs. 218 steps). Therefore, as the agent follows the humanoid longer, it has more chances to collide with it, given that the episode does not necessarily end after the required steps. In a scenario where the test-time episode concludes either after 400 follow steps or immediately upon a collision between the agent and the humanoid, SDA and the Baseline show a comparable collision rate (CR), 0.39, and 0.38, respectively.

Table 5.1: Main results for Social Navigation. Within the table, GT denotes ground truth privileged information and * corresponds to reproduced results.

Models	hGPS	traj.	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
Heuristic Expert [255]	-	-	1.00	0.97	0.51	0.52	-
Baseline [255]	GT		$0.97^{\pm 0.00}$	$0.65^{\pm 0.00}$	$0.44^{\pm 0.01}$	$0.51^{\pm 0.03}$	$0.55^{\pm 0.01}$*
Baseline+Proximity [38][1]	GT		$0.97^{\pm 0.01}$	$0.64^{\pm 0.00}$	$0.57^{\pm 0.01}$	$0.58^{\pm 0.03}$	$0.63^{\pm 0.02}$
SDA - S1		GT	$0.92^{\pm 0.00}$	$0.46^{\pm 0.01}$	$0.44^{\pm 0.02}$	$0.61^{\pm 0.02}$	$0.50^{\pm 0.01}$
Baseline [255]			$0.76^{\pm 0.02}$	$0.34^{\pm 0.01}$	$0.29^{\pm 0.01}$	$\mathbf{0.48}^{\pm 0.03}$	$0.40^{\pm 0.02}$*
Baseline+Proximity [38]			$0.85^{\pm 0.02}$	$0.41^{\pm 0.02}$	$0.37^{\pm 0.01}$	$0.58^{\pm 0.02}$	$0.41^{\pm 0.01}$
SDA - S2			$\mathbf{0.91}^{\pm 0.01}$	$\mathbf{0.45}^{\pm 0.01}$	$\mathbf{0.39}^{\pm 0.01}$	$0.57^{\pm 0.02}$	$\mathbf{0.43}^{\pm 0.02}$

Adaptable information. Table 5.2 presents the results from Stage 1 (S1) and Stage 2 (S2) with the utilization of various privileged information, such as Humanoid GPS (hGPS) and human trajectories (traj.). During S1, particularly in S and notably in SPS, incorporating hGPS as an additional input leads to superior results. This advantage is likely attributed to hGPS containing implicit information about human and robot positions, facilitating more efficient path selection, especially in SPS scenarios.

However, this pattern is not evident in S2, where trajectories typically provide more adaptable information. As previously mentioned, hGPS inherently encompasses the robot's position, posing challenges in regressing this data during the second stage due to the lack of initial context regarding the robot's location relative to the environment. Furthermore, hGPS may be difficult to adapt when the human is detected and disappears around a wall. Since hGPS comprises polar coordinates, the distance between the robot and the human spans the wall. This issue does not arise when utilizing trajectories, as they do not require any information about the robot and can be adapted solely based on depth images and detection information.

[1] Code refactored and adapted from [38] to Habitat 3.0.

Table 5.2: Ablation studies for social dynamics estimation. GT denotes ground truth privileged information, and $_A$ indicates the adapted information.

Models	hGPS	traj.	hGPS$_A$	traj.$_A$	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
S1	GT	GT			**0.94**$^{\pm 0.01}$	0.58$^{\pm 0.00}$	0.45$^{\pm 0.02}$	0.64$^{\pm 0.03}$	**0.52**$^{\pm 0.01}$
S1	GT				0.93$^{\pm 0.00}$	**0.62**$^{\pm 0.01}$	**0.46**$^{\pm 0.01}$	0.64$^{\pm 0.02}$	0.48$^{\pm 0.01}$
S1 (*Proposed*)		GT			0.92$^{\pm 0.00}$	0.46$^{\pm 0.01}$	0.44$^{\pm 0.02}$	**0.61**$^{\pm 0.02}$	0.50$^{\pm 0.01}$
S2			✓	✓	0.57$^{\pm 0.06}$	0.21$^{\pm 0.04}$	0.05$^{\pm 0.01}$	**0.30**$^{\pm 0.02}$	0.02$^{\pm 0.00}$
S2			✓		0.70$^{\pm 0.02}$	0.31$^{\pm 0.02}$	0.05$^{\pm 0.01}$	0.70$^{\pm 0.03}$	0.03$^{\pm 0.01}$
S2 (*Proposed*)				✓	**0.91**$^{\pm 0.01}$	**0.45**$^{\pm 0.01}$	**0.39**$^{\pm 0.01}$	0.57$^{\pm 0.02}$	**0.43**$^{\pm 0.02}$

5.4.2 Qualitative Results

We qualitatively demonstrate the results in our proposed SDA. Firstly, we showcase the agent's ability to locate the humanoid within the environment by moving around, followed by its capability to follow the humanoid within the environment. Subsequently, we present two specific behaviors where the agent briefly spots the humanoid and one where it moves backward to create space for passage. Fig. 5.3 shows an episode where the agent and the human are located in different rooms at the start. Then, the agent begins its search for the humanoid by navigating within the environment until the encounter takes place. After the encounter, the agent then transitions into the follow phase.

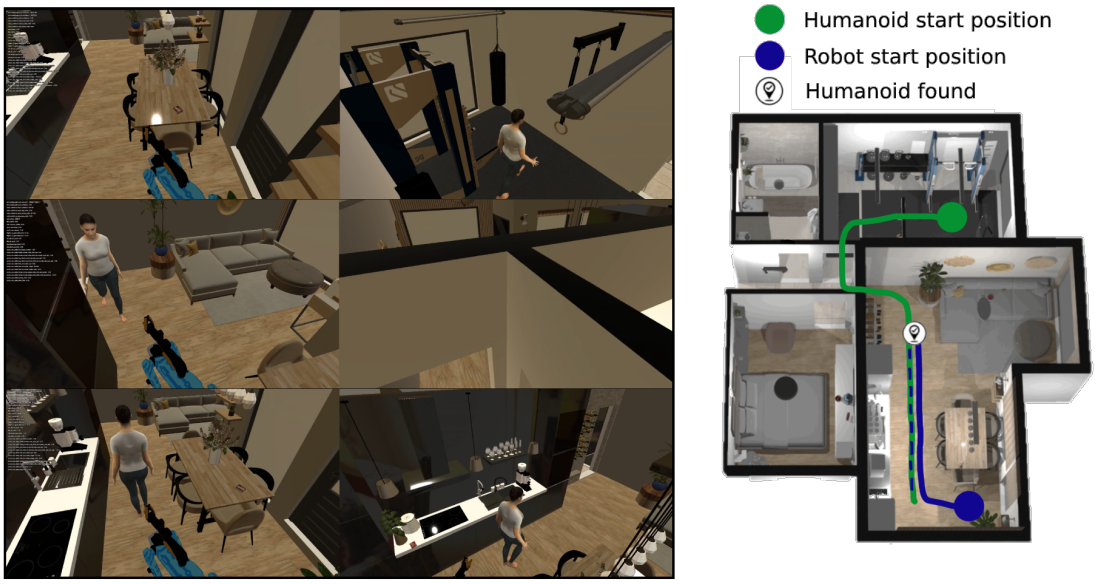


Figure 5.3: The agent and the humanoid start the episode in separate rooms. The agent navigates through the environment in search of the humanoid, and once found, begins to follow it.

Latent Analysis. We additionally explore the implications of our approach and the role of adapted human behavior. In Figure 5.4, we present the distribution of inferred human behaviors using the latent variable $\hat{z}$, computed across all timesteps in the test set. The t-SNE projection reveals four distinct behavioral stages in a well-separated manner, highlighting key aspects of human-robot interaction:

- **Find**: The robot has not yet detected any human in its RGB camera frames and remains outside the zone of interest, typically at a distance of 1-2 meters.

- **Seek**: The human has been detected but is still beyond the zone of interest. The robot is moving toward the person.
- **Lost**: The robot loses sight of the human it previously detected, but the person is still within the 1-2 meter range, prompting the robot to reorient.
- **Follow**: The human is actively being tracked by the robot, remaining within the 1-2 meter zone, allowing the robot to continue following the person.

We also overlay dots from a representative experiment, where the color gradient from black to white shows the progression of time. The plot effectively illustrates the robot's adaptive behavior as it transitions fluidly between modes: the robot starts in **Find** mode, shifts to **Seek**, and then enters **Follow** mode upon detecting the person. At times, the human moves behind an obstacle, such as a wall, prompting the robot to switch to **Lost** mode. Eventually, the robot reacquires the person and resumes following until the episode ends.

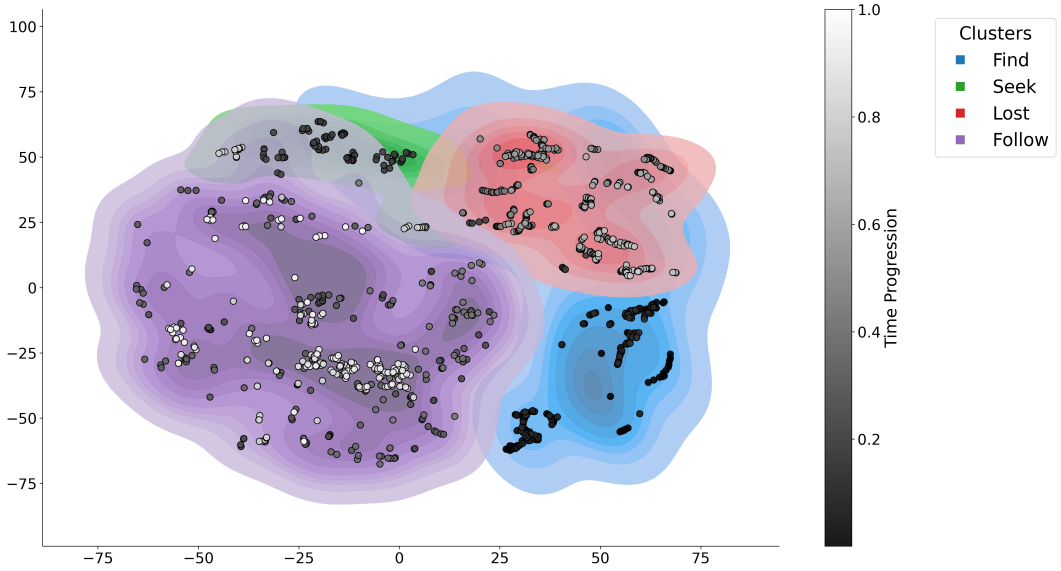


Figure 5.4: Latent Analysis

5.5 Toward real-world scenarios

While simulated environments have advanced significantly, they still fall short in capturing the full unpredictability of real-world interactions. To address this limitation, we take incremental steps toward real-world scenarios by proposing tests that incorporate more realistic social behavior, constrained computational resources, imperfect communication between the adaptation module and the primary policy, and noisy sensors. Although conducted in simulation, our work aligns with standard practices in the Embodied AI literature [36, 38, 47, 48, 266, 342, 343, 346], which utilize simulated environments to rigorously evaluate and iterate on agent behaviors. In contrast, "Sim2Real" studies are typically standalone works [99, 242] that focus specifically on transferring policies and learned behaviors from simulated environments to real-world applications. The anticipated conclusion is that more realistic human behavior and sensor readings likely contribute to the sim-to-real gaps. However, incorporating realistic training samples can partially recover

performance loss. Additionally, the performance gap identified between the Baseline [255] and our proposed SDA remains consistent across metrics, underscoring the algorithmic novelty of our simulated-only study.

ORCA. We have augmented the motion policy of humanoids in Habitat 3.0 by making them aware of the robot presence, with reciprocal collision avoidance (ORCA) [291]. When used on two agents (or more), it provides sufficient conditions for collision-free motion by letting each agent take half the responsibility of avoiding pairwise collisions. In our case, however, it is applied only to the humanoid; meanwhile, the robot still relies on its end-to-end policy. This removes unrealistic behaviors where the human sees the robot and goes straight into it. Table 5.3 shows a lower collision rate (CR), 37% instead of 57% and a higher episode success (ES) 48% vs. 43%; meanwhile, we notice a slight decrease in performance in the other metrics.

Table 5.3: Comparison of SDA performances on plain Habitat 3.0 versus the variant with ORCA.

SDA	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
Habitat 3.0	**0.91**$^{\pm 0.01}$	**0.45**$^{\pm 0.01}$	**0.39**$^{\pm 0.01}$	0.57$^{\pm 0.02}$	0.43$^{\pm 0.02}$
Habitat 3.0 + ORCA	0.90$^{\pm 0.01}$	0.43$^{\pm 0.02}$	0.38$^{\pm 0.01}$	**0.37**$^{\pm 0.01}$	**0.48**$^{\pm 0.01}$

Lower frequency updates. In Table 5.4, we simulate the scenario where the adaptation module works at a lower frequency (and a low update rate) due to computational constraints that may arise during deployment. In SDA, we update the $\hat{z}$ vector at each timestep. We evaluated two scenarios where the update is performed once every two or one hundred timesteps. Notably, only a slight performance decrease affects the metrics related to the finding part of the task.

Table 5.4: SDA performance considering missing readers on Habitat 3.0.

Update Rate	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
1 *(Proposed)*	**0.91**$^{\pm 0.01}$	**0.45**$^{\pm 0.01}$	0.39$^{\pm 0.01}$	**0.57**$^{\pm 0.02}$	0.43$^{\pm 0.02}$
1/2	0.87$^{\pm 0.01}$	0.39$^{\pm 0.01}$	**0.44**$^{\pm 0.01}$	0.63$^{\pm 0.02}$	**0.48**$^{\pm 0.02}$
1/100	0.85$^{\pm 0.01}$	0.38$^{\pm 0.01}$	0.43$^{\pm 0.01}$	0.64$^{\pm 0.03}$	0.46$^{\pm 0.01}$

Noisy inputs. We evaluate the addition of noise on both the sensor input (depth images and bounding boxes) and actuators. Table 5.5 presents the results after fine-tuning both SDA and Baseline policies for 1M steps. We used Gaussian noise on the Bounding box human-detector, Redwood noise on the Depth camera (the policy does not use the RGB) [61, 242], and Gaussian noise on the high-level actuators [61, 242], the agent's angle and velocity. Analyzing the results in Table 5.5, we note that the overall largest drop in performance regards Finding Success (S), dropping from 91% to 81-83%. As a consequence of this, the collision rate (CR) actually improves since the robot needs to follow the humanoid for less time. The performance in follow (F) is mostly affected by noise in the bounding boxes (from 39% to 30%). The performance in episode success is mostly affected by adding both depth camera and bounding box noise (from 43% to 25%). Importantly, the gap between SDA and the Baseline [255] remains consistent across all noise types and all metrics, which supports the validity of tests in simulated environments.

Table 5.5: Ablation study with RedWood Noise on the Depth Camera, and Gaussian Noise on the Bounding Box and Actuators.

Model	Noisy Input	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
Baseline	None	$0.76^{\pm 0.02}$	$0.34^{\pm 0.01}$	$0.29^{\pm 0.01}$	$0.48^{\pm 0.03}$	$0.40^{\pm 0.02}$
SDA	None	$0.91^{\pm 0.01}$	$0.45^{\pm 0.01}$	$0.39^{\pm 0.01}$	$0.57^{\pm 0.02}$	$0.43^{\pm 0.02}$
Baseline	Depth Camera	$0.70^{\pm 0.01}$	$0.32^{\pm 0.01}$	$0.26^{\pm 0.01}$	$0.43^{\pm 0.02}$	$0.20^{\pm 0.02}$
SDA	Depth Camera	$0.83^{\pm 0.02}$	$0.42^{\pm 0.03}$	$0.34^{\pm 0.02}$	$0.30^{\pm 0.01}$	$0.37^{\pm 0.01}$
Baseline	Bounding Box	$0.73^{\pm 0.01}$	$0.30^{\pm 0.01}$	$0.24^{\pm 0.01}$	$0.44^{\pm 0.02}$	$0.15^{\pm 0.02}$
SDA	Bounding Box	$0.83^{\pm 0.01}$	$0.41^{\pm 0.02}$	$0.30^{\pm 0.01}$	$0.28^{\pm 0.02}$	$0.35^{\pm 0.02}$
Baseline	Depth Camera + Bounding Box	$0.69^{\pm 0.01}$	$0.33^{\pm 0.01}$	$0.25^{\pm 0.01}$	$0.44^{\pm 0.02}$	$0.21^{\pm 0.02}$
SDA	Depth Camera + Bounding Box	$0.82^{\pm 0.01}$	$0.41^{\pm 0.02}$	$0.48^{\pm 0.02}$	$0.45^{\pm 0.01}$	$0.25^{\pm 0.02}$
Baseline	Actuators	$0.71^{\pm 0.01}$	$0.29^{\pm 0.01}$	$0.24^{\pm 0.01}$	$0.42^{\pm 0.02}$	$0.31^{\pm 0.02}$
SDA	Actuators	$0.81^{\pm 0.01}$	$0.41^{\pm 0.02}$	$0.35^{\pm 0.01}$	$0.42^{\pm 0.01}$	$0.40^{\pm 0.02}$

5.6 Additional Results

The content is structured as follows:

- Metrics - Sec. 5.6.1 expands on the metrics presented in the primary paper and introduces supplementary ones, as exemplified in [255].
- Training Details - Sec. 5.6.2 outlines the training methodology, detailing the training stages, the use of DD-PPO, and the hardware setup for training.
- Results - Sec. 5.6.3 includes a table listing all metrics along with an accompanying explanation of the results.
- Error Analysis - Sec. 5.6.4 analyzes failure cases, identifying key areas where the model encounters challenges, such as constrained movements and blind spots.
- Ablation Studies - Sec. 5.6.5 explores the adaptability of privileged information and trajectory length's influence on model performance.
- Qualitative Results - Sec. 5.6.6 showcases visual examples of episodes where the agent successfully tracks or avoids collisions with humans, with accompanying qualitative analysis.

Moreover, we show some generated episodes in the enclosed supplementary video.

5.6.1 Metrics

In our main paper, we utilize metrics including Finding Success (S), Finding Success Weighted by Path Steps (SPS), Following Rate (F), Collision Rate (CR), and Episode Success (ES). However, Habitat 3.0 [255] introduces in their supplementary material additional metrics such as Backup-Yield Rate (BYR), Total Distance (TD), and Following Distance (FD), providing further insights into the models. Similarly, in our supplementary material, we also incorporate these metrics. Subsequently, a comprehensive explanation will be given for all the former and latter metrics featured.

(1) **Finding Success** (*S*): This metric, denoted S, evaluates whether the robot successfully locates the humanoid within the maximum episode steps and reaches it within a close range of 1-2 meters while facing toward it. It is represented as:

$$S = \begin{cases} 1 & \text{if the robot successfully finds and reaches the humanoid,} \\ 0 & \text{otherwise} \end{cases}$$

This metric provides a binary indication of the robot's ability to locate and approach the humanoid within the designated constraints.

(2) **Finding Success Weighted by Path Steps** (*SPS*): The SPS metric, calculated as $SPS = S \cdot \frac{l}{\max(l,p)}$, evaluates the efficiency of the robot's path relative to an oracle with complete knowledge of the humanoid's trajectory and the environment map. Here, l represents the minimum steps an oracle would take to find the humanoid, and p denotes the agent's actual path steps. A higher SPS value indicates the robot's more efficient path toward finding the humanoid.

(3) **Following Rate** (F): The following rate F quantifies the ratio of steps during which the robot maintains a distance of 1-2 meters from the humanoid while facing towards it relative to the maximum possible following steps. It is calculated as:

$$F = \frac{w}{\max(E - l, w)}$$

E denotes the maximum episode duration, and w represents the number of steps during which the agent closely follows the humanoid. This metric provides insight into the robot's ability to consistently track the humanoid once it has been located.

(4) **Collision Rate** (CR): The collision rate CR measures the ratio of episodes that end with the robot colliding with the humanoid. It is computed as:

$$CR = \frac{\text{Number of episodes ending in collision}}{\text{Total number of episodes}}$$

This metric assesses the robot's collision avoidance capabilities during interactions with the humanoid.

(5) **Backup-Yield Rate** (BYR): The backup-yield rate BYR quantifies the frequency with which the robot performs backup or yield motions to avoid collision when the humanoid is nearby. A 'backup motion' refers to a backward movement executed by the robot when the distance between the robot and the humanoid is less than 1.5 meters. Similarly, a 'yield motion' denotes a robot's maneuver to avoid collision when the distance between them is less than 1.5 meters and the robot's velocity is less than 0.1 m/s. The BYR is computed as:

$$BYR = \frac{\text{Number of backup or yield motions}}{\text{Total number of episodes}}$$

This metric provides insights into the effectiveness of the robot's collision avoidance strategies.

(6) **Total Distance** between the robot and the humanoid (TD): The TD metric evaluates the average L2 distance between the robot and the humanoid over the total number of episode steps. It is calculated as:

$$TD = \frac{\sum \text{L2 distance between robot and humanoid}}{\text{Total number of episode steps}}$$

This metric measures the overall proximity between the robot and the humanoid throughout the episodes,

providing insights into the effectiveness of the robot's navigation and tracking capabilities.

(7) **Following Distance** between the robot and the humanoid after the first encounter (FD): The FD metric assesses the L2 distance between the robot and the humanoid after the robot initially encounters the humanoid. It quantifies the proximity between the two entities during the following stages of interaction. The FD should ideally be maintained within 1-2 meters, indicating effective tracking and following behavior. This metric is calculated as:

$$FD = \frac{\sum \text{L2 distance between robot and humanoid after first encounter}}{\text{Total number of episode steps}}$$

The FD metric provides valuable insights into the robot's ability to maintain an appropriate distance from the humanoid target after initial contact, which is crucial for effective interaction and task completion.

(8) **Episode Success** (ES) measures the ratio of the episodes where the agent found the human and followed it for the required number of steps, maintaining a safe distance in the 1-2 meters range.

5.6.2 Training details.

During training, we encode social dynamics using a ResNet, trained from scratch. The trajectory encoder μ consists of a 3-layer MLP, and the output z_t has a dimensionality of 128. The Adapter ψ comprises alternating spatial and temporal MLP layers, with the output $\hat{z}_t$ matching the dimensionality of z_t. We jointly train the policy π, the perceptions encoder (ResNet), and the social dynamics (Trajectory Encoder) encoder μ during the first stage. In the second stage, everything is frozen except the Adapter. In Stage 1, we utilize DD-PPO [331] for 250 million steps across 24 environments, following the training protocol presented in [255]. Furthemore, each time step is approximately 0.04 seconds, so we consider a trajectory of 0.8 seconds. An entire episode of 1500 steps corresponds to almost a 1-minute long video. Inputs to the policy are egocentric arm depth and a humanoid detector, and outputs are velocity commands (linear and angular) in the robot's local frame. This policy does not have access to a map of the environment. The training process takes approximately four days. In Stage 2, we employ supervised learning for 5 million steps across the same environments. The learning process lasts around 2 hours. Both stages utilize 4 A100 GPUs for efficient computation.

5.6.3 Results

In Table 5.6, we compare Baseline [255], Baseline+Prox. [38], and the novel SDA model on the additional metrics proposed by [255]. We showcase a comparable *Backup Yield Rate* across the methodologies, meaning that all models suggest an avoidance behaviour. Regarding *Following Distance*, all models, on average, fall within the ideal 1-2 meters range, with SDA exhibiting a conservative behavior, i.e., SDA remains at 1.80 meters further from the humanoid. The *Total Distance* is lower for SDA and Baseline+Prox. than for Baseline. This is due to the ability of the first two models to detect the human (SPS) earlier and to follow it for a longer duration (F). In summary, the inclusion of additional metrics such as BYR, TD, and FD leads to analogous conclusions, as previously outlined in the main paper (Sec. 4.1). Specifically, it underscores that the comprehensive advantage in adapting to social dynamics stems from following a more optimal trajectory, maintaining it over an extended period, and ensuring a safe distance.

[2]Code refactored and adapted from [38] to Habitat 3.0.

Table 5.6: Comparative evaluation of Social Navigation on Habitat 3.0 [255]. Within the table, GT denotes ground truth privileged information and * corresponds to reproduced results. Beyond what is reported in Table 1 of the main paper, we additionally report here: (1) Backup-Yield Rate (BYR), (2) The Total Distance between the robot and the humanoid (TD), and (3) The "Following" Distance (in *meters*) between the robot and the humanoid after the first encounter (FD).

Models	hGPS	traj.	S ↑	SPS ↑	F ↑	CR ↓	BYR	TD	FD	ES ↑
Heuristic Expert [255]	-	-	1.00	0.97	0.51	0.52	0.24	2.56	1.72	-
Baseline [255]	GT		0.97	0.65	0.44	0.51	0.19	3.43	1.70	0.55*
Baseline+Prox. [38][2]	GT		0.97	0.64	0.57	0.58	0.17	3.15	1.66	0.63
SDA - S1		GT	0.92	0.46	0.44	0.61	0.18	3.70	1.83	0.50
Baseline [255]			0.76	0.34	0.29	**0.48**	0.13	5.18	1.64	0.42*
Baseline+Prox. [38]			0.85	0.41	0.37	0.58	0.14	4.24	1.57	0.41
SDA - S2			**0.91**	**0.45**	**0.39**	0.57	0.12	4.24	1.80	**0.43**

5.6.4 Error Analysis

We include in Fig.5.5 We analyze the distribution of failure causes in social navigation analyzed by watching 100 failed episodes and categorizing the failures into five types:

- Constrained Movements (28%): The robot cannot avoid collision due to environmental constraints.
- Blindspot (25%): The robot cannot perceive the human due to blind corners or side collisions.
- Not Found (22%): The robot does not detect the humanoid within 1500 simulation steps.
- Moving Backwards (22%): The robot yield space backwards but fails to avoid collisions.
- Walking into a Humanoid (3%): Frontal collisions with the humanoid are the least common cause.

The failure analysis offers valuable insights into areas needing improvement. Walking directly into the humanoid is very unlikely and represent the worst scenario. As can be seen from Table 5.3, by including ORCA, we reduce the collision rate by 20 percentage points and this scenario will likely disappear. With ORCA, humans can either slow down or slightly change their direction, particularly for Constrained Movements and moving Backwards. Constrained Movements and Not Found could be improved by high-level planning capabilities that devise better exploration strategies to locate the person and let the robot back off until there is enough space to let the human pass. The addition of a microphone sensor could also help locate humans in the scenario in which someone could call their robot.

5.6.5 Additional Analysis

In the next section, we conduct a failure case analysis of SDA , examining the adaptability of privileged information and performing an ablation study on its design.

Fig. 5.6*(left)* illustrates the trend of the *First encounter step over episode* during Stage 2. The plot confirms that using trajectories facilitates the robot's finding the person, which it achieves after only approx. 450 steps, while the hGPS and hGPS+traj take more than 700 steps. Finding the robot first results in higher S and SPS metrics, but it may incur larger chances of collision (CR metric), due to having to then follow it for longer, cf. Sec. 9.7.2.

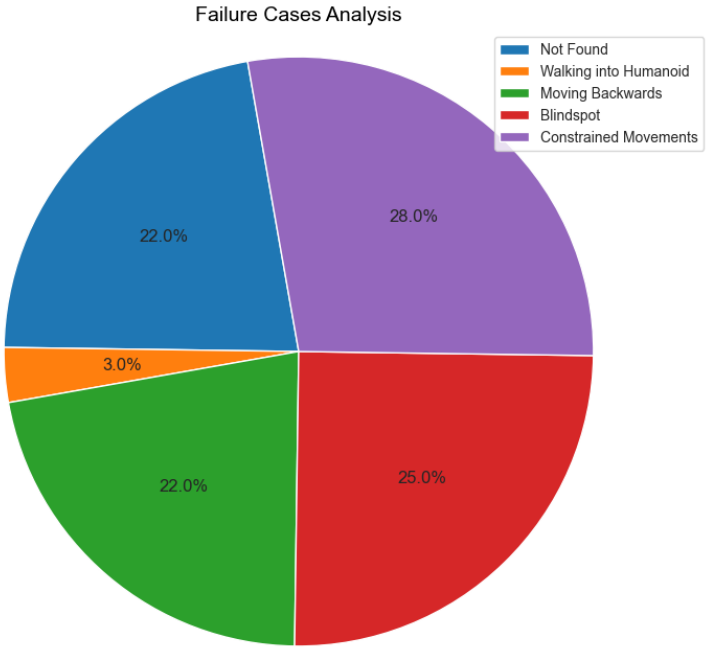


Figure 5.5: Failure Cases Analysis on 100 episodes

Fig. 5.6*(right)* illustrates the average distance between the agent and the humanoid after the first encounter. While hGPS and hGPS+traj stay at approx. 4 meters, the proposed trajectory-based approach ranges between 1.5 and 1.9, thus well within the 1-2 meter range, which yields the success in the task of following (larger F metric, but encompassing a larger risk of collision– CR metric).

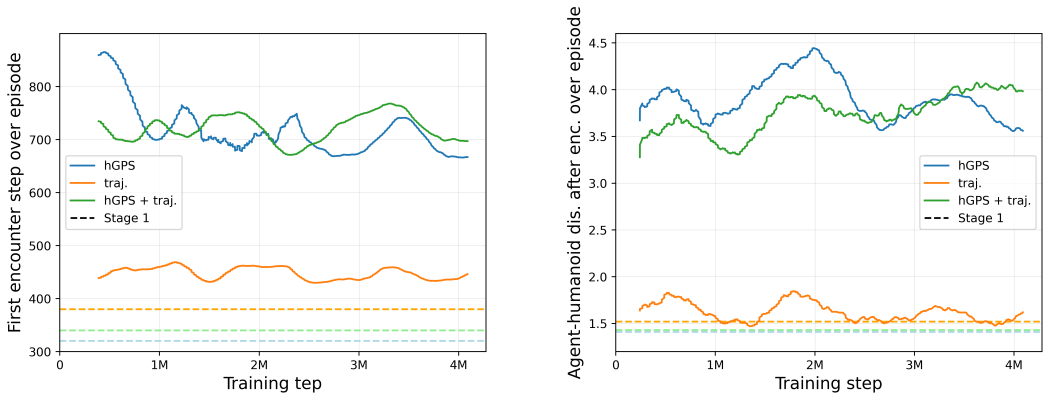


Figure 5.6: *(left)* Training step (x-axis) Vs the number of steps which it takes the robot to find the humanoid (y-axis), in the finding task; *(right)* Training step (x-axis) Vs the average distance which the robot manages to keep itself at from the humanoid (y-axis), in the task of following.

Design Analysis. RMA [171] was used to adapt an agent from simulation to real-world deployment. In contrast, SDA encodes human behavior in trajectories, which can be inferred at test time to develop a socially aware navigation policy. While RMA encodes the environment configuration vector e_t with information only at time t, we feed an entire trajectory encoding the position in the 20 timesteps up to time t. As shown in Table 5.7, encoding the human position only at time t results in unsatisfactory performance, making the direct application of RMA relatively poor. In fact, it achieves just 3% of Episode Success (ES) against the

43% of SDA.

Table 5.7: Ablation on the length of trajectories to consider during Stage 1

# States Considered	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
1	$0.55^{\pm 0.01}$	$0.14^{\pm 0.02}$	$0.03^{\pm 0.01}$	$0.75^{\pm 0.01}$	$0.03^{\pm 0.01}$
5	$0.62^{\pm 0.01}$	$0.25^{\pm 0.02}$	$0.05^{\pm 0.01}$	$0.64^{\pm 0.01}$	$0.06^{\pm 0.01}$
20 (*Proposed*)	$\mathbf{0.92}^{\pm 0.00}$	$\mathbf{0.46}^{\pm 0.01}$	$\mathbf{0.44}^{\pm 0.02}$	$\mathbf{0.61}^{\pm 0.02}$	$\mathbf{0.50}^{\pm 0.01}$
50	$0.70^{\pm 0.02}$	$0.29^{\pm 0.02}$	$0.08^{\pm 0.02}$	$0.78^{\pm 0.02}$	$0.08^{\pm 0.01}$

Given the sequential nature of the states that get encoded, we also evaluated Transformers and MLPs and ultimately selected the MLPs for SDA. In Table 5.8, we include this preliminary study. Note how the choice of the new sequence modeling mechanism is not trivial and strongly affects performance.

Table 5.8: Ablation on the type of encoder to consider during Stage 1.

Encoder Type	S ↑	SPS ↑	F ↑	CR ↓	ES ↑
MLP (*Proposed*)	$\mathbf{0.92}^{\pm 0.00}$	$\mathbf{0.46}^{\pm 0.01}$	$\mathbf{0.44}^{\pm 0.02}$	$\mathbf{0.61}^{\pm 0.02}$	$\mathbf{0.50}^{\pm 0.01}$
Transformer	$0.85^{\pm 0.01}$	$0.15^{\pm 0.02}$	$0.27^{\pm 0.01}$	$0.76^{\pm 0.01}$	$0.12^{\pm 0.01}$

5.6.6 Qualitative results

The episode in Fig. 5.7 (*top*) illustrates the agent's capability to track the human even when it is briefly observed. As the agent searches for the target in the bathroom, the humanoid swiftly passes in front of the door (Step 145) and disappears from the agent's view again (Step 150). Due to learned social dynamics, the agent exploits the humanoid's behavior and begins to follow it. In the episode in Fig. 5.7 (*bottom*), the agent has already located the human and its objective is to follow it. It is observed that at Step 340, the human decides to move backward, prompting the agent to move backward as well, creating the necessary space for passage by Step 360. Then, it continues its task of following.

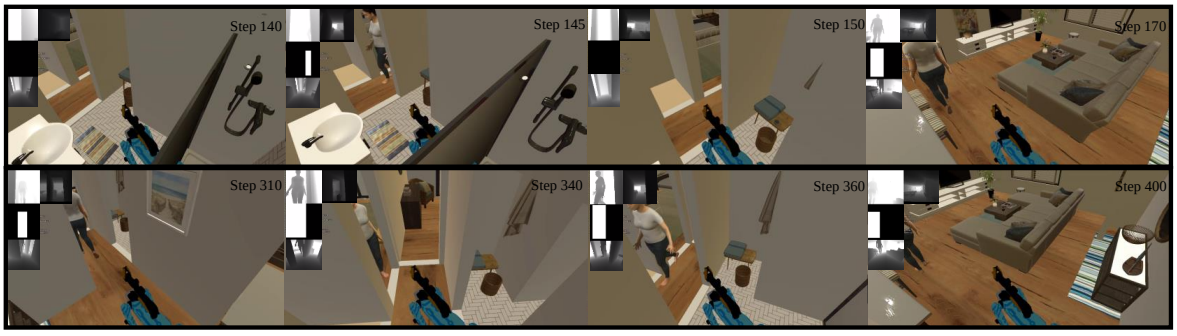


Figure 5.7: We showcase two different episodes. On the top, the agent successfully follows the human after it swiftly moves in front of the door. On the bottom, an episode where the human decides to move backward and the agent steps back to make way.

Limitations. Table 5.2 illustrates how the performance in stage 2 is affected by the adaptability of information, underscoring its importance in evaluating the model's effectiveness. Additionally, a limitation of our approach is that the proposed model has been benchmarked solely in simulation, relying on training information—such as trajectories—that may not be readily available in real-world settings.

5.7 Conclusions

Our study presents the Social Dynamics Adaptation model (SDA) for Social Navigation. Notably, it is the first to integrate privileged human dynamics information during training while adapting it in the following stage, enabling its application in realistic environments without relying on such privileged information. Our findings underscore the non-trivial nature of adapting information, highlighting the necessity for selective processes. In future research, we aim to extend our model by incorporating diverse human dynamics beyond trajectories, enhancing the robot's comprehension of human movement patterns.

Chapter 6

TopoX: A Suite of Python Packages for Machine Learning on Topological Domains

6.1 Introduction

Deep learning traditionally operates within Euclidean domains, focusing on structured data like images [169] and sequences [299]. However, to handle more diverse data types, geometric deep learning (GDL) [31, 40, 374] has emerged. GDL extends deep learning to non-Euclidean data by leveraging geometric regularities like symmetries and invariances. Recently, Topological Deep Learning (TDL) [121] has gained attention, exploring models beyond traditional graph-based abstractions to process data with multi-way relations, such as simplicial complexes and hypergraphs. These extensions allow for the representation of diverse data domains encountered in scientific computations [19, 86, 104, 118, 270, 277, 278, 341]. Despite theoretical advancements, practical implementation faces challenges due to the lack of accessible software libraries supporting deep learning models with higher-order structures.

In this paper, we present `TopoX`, an open-source suite of Python packages designed for machine learning and deep learning operations in topological domains. `TopoX` is organized into three Python packages: `TopoNetX`, `TopoEmbedX`, and `TopoModelX`. These packages enhance and generalize functionalities found in popular mainstream graph computations and learning tools, enabling them on topological domains. What sets `TopoX` apart is its abstract general design and the exploitation of the resulting modeling flexibility in the implementation of a broad spectrum of topological domains and TDL models (see Table 6.1). Further, every domain in `TopoNetX` offers utilities to work with various components such as nodes, edges, and higher-order cells. `TopoNetX` also supports computations using incidence matrices, (co)adjacency matrices, and up, down, and Hodge Laplacians. From a representation learning point of view, `TopoEmbedX` provides methods for embedding topological domains, or parts of these domains, into Euclidean domains. `TopoModelX` offers a wide range of TDL models based on a comprehensive implementation of higher-order message passing, built on the `PyTorch` framework [243].

The core objectives of `TopoX` are as follows: facilitate research in topological domains by providing foundational code to understand concepts, and offer a platform to disseminate algorithms; broaden the accessibility of the field by delivering user-friendly topological learning algorithms to the machine learning community; serve as a learning resource for topological domains and TDL, enriched by diverse examples, notebooks, and visualization capabilities; and provide a unified application programming interface (API) that operates on topological domains. Given that topological spaces generalize most data domains encoun-

tered in scientific computations, its unified API offers multiple advantages: it enhances interoperability, streamlines productivity, simplifies learning, and fosters collaboration. By providing a common framework, it reduces maintenance, promotes code portability, and supports parallel computing.

6.2 Implementation Overview

The `TopoNetX` package is organized into three main modules: `classes`, `algorithms`, and `transform`. The `classes` module implements numerous common topological domains, such as `SimplicialComplex`, `CellComplex`, and more; inheriting from the abstract class `Complex`. The `algorithms` module implements spectral methods, distance computation, and connected component analysis in topological domains. The `transform` module facilitates conversions between different topological domains. `TopoNetX` uses `Numpy` and `Scipy` backends, and offers toy datasets and examples to facilitate learning and improve understanding.

The `TopoEmbedX` package supports the learning of representations for all topological domains available in `TopoNetX`. This package contains the module `classes`, which implements topological representation learning algorithms that generalize the most popular graph-based representation learning algorithms. These algorithms include `DeepCell` and `Cell2Vec`. The `TopoEmbedX` package has an API inspired by `scikit-learn` [248] and utilizes the KarateClub [272] backends.

`TopoModelX` is a Python package for topological deep learning, providing efficient tools to implement topological neural networks (TNNs). The package consists of two main modules: `base` and `nn`. The `base` module implements higher-order message passing methods, allowing the construction of general-purpose TNNs using the tensor diagram formalism introduced in [121] and surveyed in [241]. The `nn` module implements various TNNs on popular topological domains, such as simplicial complexes, cell complexes, hypergraphs, and combinatorial complexes. Each implementation includes a corresponding Jupyter notebook tutorial for a user-friendly initiation into TDL. `TopoModelX` leverages `PyTorch` and `PyG` (PyTorch Geometric) backends [88]. The `TopoModelX` package is the outcome of a coding challenge that crowdsourced the implementations of TDL models [240].

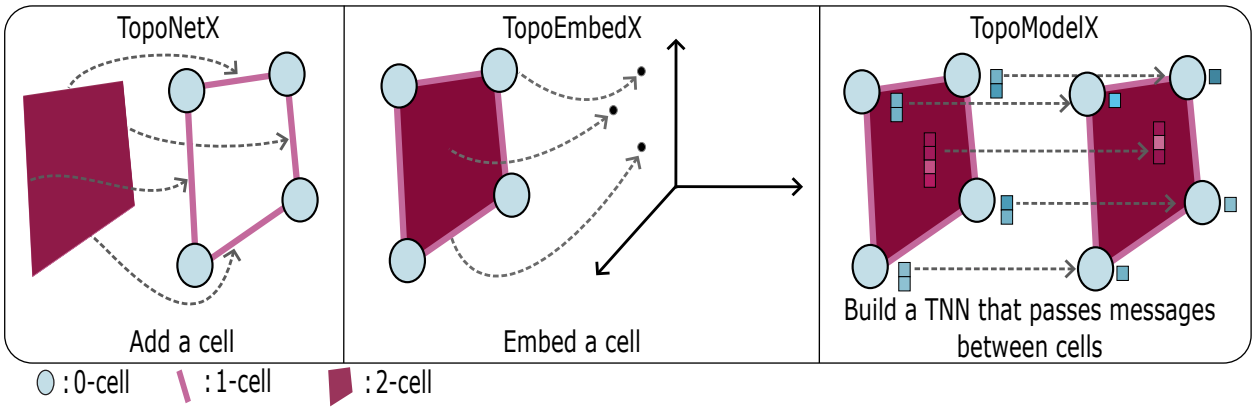


Figure 6.1: Building blocks from the three packages of the `TopoX` software suite. Left: `TopoNetX` enables building topological domains such as cell complexes. The figure demonstrates adding a 2-cell on a cell complex with `TopoNetX`. Middle: `TopoEmbedX` enables embedding of topological domains inside a Euclidian space. The figure illustrates embedding a 0-cell, a 1-cell and a 2-cell from a cell complex inside a Euclidean space with `TopoEmbedX`. Right: build a topological neural network (TNN) that processes data via higher-order message passing on a cell complex with `TopoModelX`.

Comparison between TopoNetX and other Python packages		
Packages	Domains	Operations
TopoNetX	Graphs, colored hypergraphs, simplicial complexes, path complexes, cell complexes, combinatorial complexes	add_node, add_simplex, add_cell, adjacency_matrix, coadjacency_matrix, incidence_matrix, hodge_laplacian_matrix
NetworkX	Graphs	add_node, add_edge, adjacency_matrix, incidence_matrix, laplacian_matrix
HyperNetX	Hypergraphs	add_node, add_edge, adjacency_matrix, incidence_matrix
XGI	Simplicial complexes, hypergraphs, dihypergraph	add_node, add_edge, adjacency_matrix, boundary_matrix, hodge_laplacian
Comparison between TopoEmbedX and other Python packages		
Packages	Domains	Embedding algorithms
TopoEmbedX	Graphs, colored hypergraphs, simplicial complexes, path complexes, cell complexes, combinatorial complexes	Cell2Vec, DeepCell, CellDiff2Vec, HigherOrderLaplacianEigenMap, HOPE, HOGLEE,
Karateclub	Graphs	Node2Vec, Graph2Vec, Diff2Vec, GL2Vec, IGE, Role2Vec, GraRep
Comparison between TopoModelX and other Python packages		
Packages	Domains	Push-forward operators
TopoModelX	Graphs, (colored) hypergraphs, simplicial complexes, path complexes, cell complexes, combinatorial complexes	(Higher order) message passing, merge operator, split operator
DGL	Graphs	Message passing
DHG	Graphs, hypergraphs	Message passing, hypergraph message passing
PyG	Graphs	Message passing

Table 6.1: TopoX provides a user-friendly and comprehensive suite for building blocks and computing on topological domains. The table shows a comparison between TopoX and other Python packages.

6.3 Comparison and Interaction with Other Packages

The libraries most closely related to TopoNetX are NetworkX [117] and HypernetX [193]. They facilitate computations on graphs and hypergraphs, respectively. TopoNetX utilizes a similar API to these two libraries to faciliaite rapid adoption of topological domains, such as simplicial complexes, cell complexes, and colored hypergraphs. The XGI package [174] offers hypergraph functionalities similar to HypernetX with additional support for simplicial complexes and directed hypergraphs.

The closest package to TopoEmbedX is KarateClub [272], which is a Python package consisting of methods for unsupervised learning on graph-structured data. TopoEmbedX extends the functionality of KarateClub to topological domains supported in TopoNetX. PyG and DGL (Deep Graph Library) [322], which are two popular geometric deep learning packages that support deep learning models on graphs, are closely related to TopoModelX. The DHG (Deep HyperGraph) package [86] supports deep learning models defined on hypergraphs. All three of our packages feature a continuous integration pipeline and are well-tested with code coverage of >=95% per package. This is comparable or better than many related graph-based learning libraries, such as PyG, DGL, XGI, NetworkX, and HyperNetX.

6.4 Usage: Elementary Examples

TopoNetX provides an user-friendly interface which allows to create a complex in two main steps: first, instantiate the complex; second, add cells to that complex, as shown in the three first lines of the code snippet below. Processing data on a complex requires matrices that describe the (co)adjacency of the incidence

relations among cells, as computed below.

```
cell_complex = CellComplex()
cell_complex.add_cell([1,2,3,4], rank=2)
cell_complex.add_cell([1,2,5], rank=2)
L2 = cell_complex.hodge_laplacian_matrix(2)
```

The following code snippet shows how `TopoEmbedX` embeds edges of the Stanford bunny dataset using the Cell2Vec algorithm [118]:

```
cell_complex = tnx.datasets.stanford_bunny("cell")
model = Cell2Vec()
model.fit(cell_complex, nbhd_type="adj", nbhd_dim={"adj": 1})
```

The following code snippet shows how to instantiate, and run the forward-pass of a simplicial neural network (SNN) with `TopoModelX`:

```
simplicial_complex = tnx.datasets.stanford_bunny("simplicial")
feature = simplicial_complex.get_edge_features()
nbhd = simplicial_complex.hodge_laplacian_matrix(1)
snn_model = SNN(input_feat_dim, output_feat_dim)
snn_model(feature, nbhd)
```

`TopoNetX` provides a high-level declarative interface. In `TopoEmbedX`, each embedding algorithm is compatible with all complexes available in `TopoNetX`. In `TopoModelX`, TNNs are classified according to the topological domain upon which they are defined. In each package, the directories that contain examples and notebooks offer an abundance of code snippets to assist users in starting their journey with `TopoX`.

6.5 ICML'23 TopoX Challenge

Graph neural networks (GNNs) have proven to be a powerful deep learning architecture for processing relational data. More specifically, GNNs operate in graph domains comprised of pairwise relations between nodes. *Topological neural networks* (TNNs) extend GNNs by operating on domains featuring higher-order relations. Such domains, called *topological domains*, feature part-whole and/or set-type relations (Fig. 6.2) [122], allowing a more expressive representation of the data. By operating on a topological domain, a TNN leverages the intricate relational structure at the heart of the data. Topological deep learning [28, 122] has shown great promise in many applications, ranging from molecular classification to social network prediction. However, the adoption of its architectures has been limited by the fragmented availability of open-source algorithms and lack of benchmarking between topological domains.

The challenge described in this white paper aims to fill that gap by implementing models in a unifying open-source software. In doing so, the challenge contributes to fostering reproducible research in topological deep learning. Participants were asked to contribute code for a published TNN, following `TopoModelX`'s API [122] and computational primitives, and implement a training mechanism for the algorithm's intended task.

This white paper is organized as follows. Section 6.6 describes the setup of the challenge, including its guidelines and evaluation criteria. Section 8.5.2 lists all qualifying submissions to the challenge and its winners.

6.6 Setup of the challenge

The challenge [1] was held in conjunction with the workshop Topology and Geometry in Machine Learning of the International Conference on Machine Learning (ICML) 2023 [2]. Participants were asked to contribute code for a previously existing TNN and train it on a toy dataset of their choice.

Guidelines Each submission took the form of an implementation of a pre-existing TNN listed in a survey of the field [241]. These models fall into four categories, defined by their topological domain. All submitted code was required to comply with `TopoModelX`'s GitHub Action workflow [122], successfully passing all tests, linting, and formatting.

Each submission consisted of a pull request to `TopoModelX` containing three new files:

1. A Python script implementing a layer of the model in a single class using `TopoModelX` computational primitives. One layer is equivalent to the message passing depicted in the tensor diagram representation for the model given in the survey [241].

2. A Jupyter notebook that builds a neural network out of the single layer, loads and pre-processes the chosen dataset, and performs a train-test loop on the dataset. Defining training and testing in a Jupyter notebook offers authors a natural way to communicate results that are reproducible, as anyone with access to the notebook may run it to attain analogous results.

3. A Python script which contains the unit tests for all methods stored in the class defining the model layer.

Teams were registered to the challenge upon submission of their pull request and there was no restriction on the number of team members, nor on the amount of submissions per team.

The principal developers of `TopoModelX` were not allowed to participate. Consistent with the aims of an open environment for sharing participation in this activity is completely voluntary and no support or endorsement of any of the participating parties by any of the other participating parties is provided. All submissions are the views of the individual participants only and should be taken, as is with all faults and without any guarantee, promise or endorsement of any kind.

Evaluation criteria The evaluation criteria were:

1. Does the submission implement the chosen model correctly, specifically in terms of its message passing scheme? (The training schemes do not need to match that of the original model).

2. How readable and clean is the code? How well does the submission respect `TopoModelX`'s APIs?

3. Is the submission well-written? Do the docstrings clearly explain the methods? Are the unit tests robust?

Note that these criteria were not designed to reward model performance, nor complexity of training. Rather, these criteria aimed to reward clean code and accurate model architectures that will foster reproducible research in topological deep learning.

[1] Challenge website: `https://pyt-team.github.io/topomodelx/challenge/index.html`

[2] Topology and Geometry in Machine Learning Workshop website: `https://www.tagds.com/events/conference-workshops/tag-ml23`

Evaluation Method The Condorcet method [347] was used to rank the submissions and decide on the winners. Each team whose submission respected the guidelines was given one vote in the decision process. Nine additional reviewers selected from PyT-team maintainers and collaborators were also each given a vote. Upon voting, participating teams and reviewers were each asked to select the best and second best model implementation in each topological domain, thus making eight choices in total. Participants were not allowed to vote for their own submissions.

Software engineering practices Challenge participants were encouraged to use software engineering best practices. All code had to be compatible with Python 3.10 and a reasonable effort had to be made for the code to adhere to PEP8 Python style guidelines. The chosen dataset had to be loaded from `TopoNetX` [122] or `PyTorch-Geometric` [89]. Participants could raise GitHub issues and/or request help at any time by contacting the organizers.

6.7 Submissions and Winners

In total, the challenge received 32 submissions, 28 of which adhered to the above outlined qualification requirements. Out of the qualifying submissions, 23 unique models were implemented. All four topological domains are represented in this set of models: 12 hypergraph implementations, 11 simplicial model implementations, 3 cellular implementations, and 2 combinatorial implementations.

Table 6.2 lists all qualifying submissions. [241] contains additional information on the architectures and message-passing frameworks for each of these models.

Table 6.2 also indicates the winning contributions, consisting of a first and second prize for each topological domain, as well as honorable mentions. The winners were announced publicly at the ICML Workshop on Topology, Algebra and Geometry in Machine Learning and on social medias. Regardless of this final ranking, we would like to stress that all the submissions were of very high quality. We warmly congratulate all participants.

6.8 Conclusion

We introduced the TopoX Python suite to cater to the computing, machine learning, and deep learning needs within the realm of topological domains. The goal is to offer off-the-shelf solutions for graph computations and learning in diverse topological settings, ensuring flexibility and fidelity to topological principles. However, we acknowledge that this can occasionally impact efficiency. Future efforts will be directed towards optimizing TopoX to balance efficiency and adherence to topological concepts.

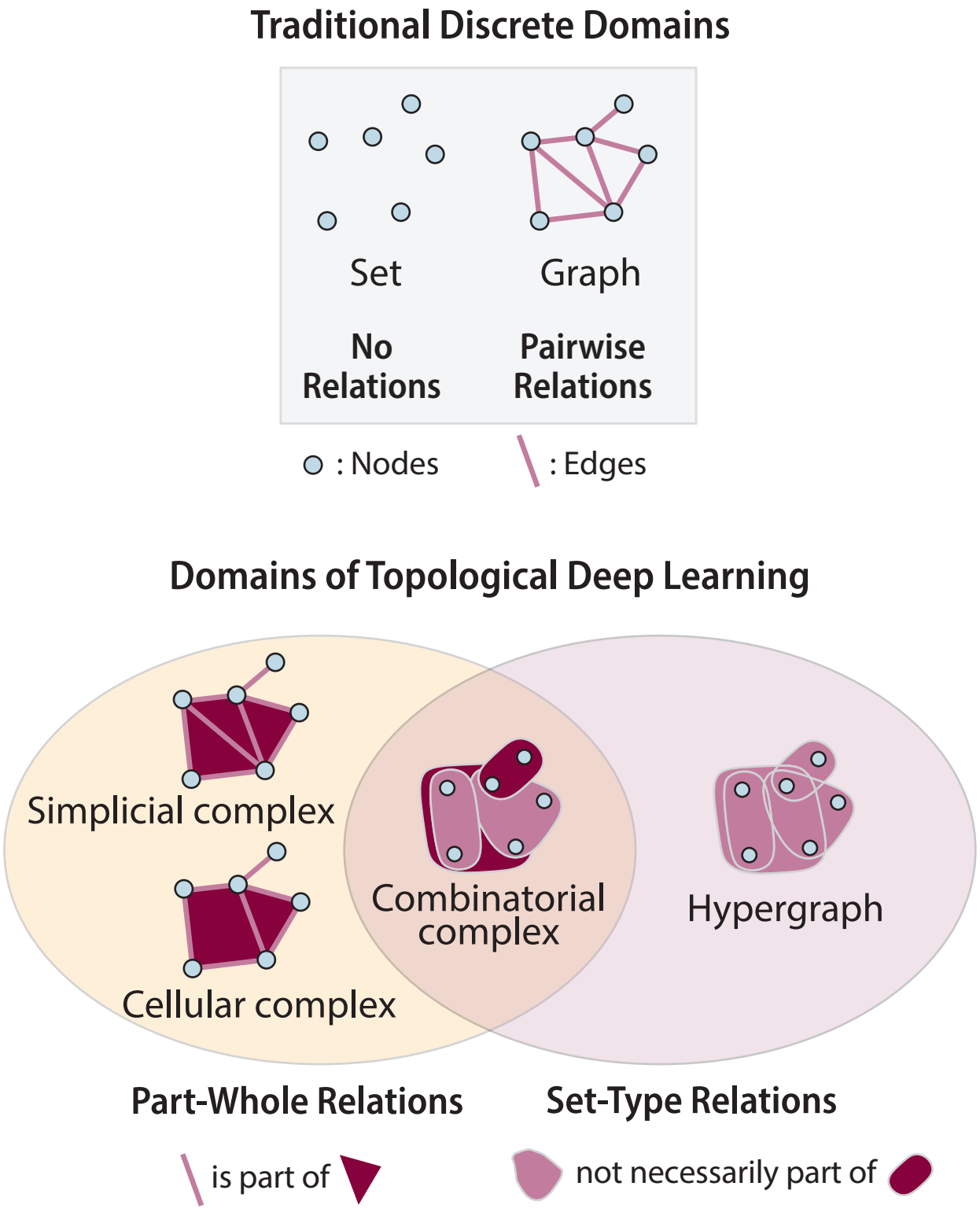


Figure 6.2: Domains: Nodes in light blue, (hyper)edges in pink, and faces in dark red. Adapted from [122].

Domain	Model	Task Level			Computational challenge submission authors
		Node	*Edge*	*Complex*	
HG	HyperSage [15]	✓			German Magai, Pavel Snopov
	AllSetTransformer [57]	✓			Luca Scofano, Indro Spinelli, Simone Scardapane, Simone Fiorellino, Olga Zaghen, Lev Telyatnikov, Claudio Battiloro, Guillermo Bernardez (first place)
	HyperGat [75]	✓			German Magai, Pavel Snopov
	HNHN [76]	✓	✓		Alessandro Salatiello (hon. mention), Sadrodin Barikbin
SC	SCCONV [33]			✓	Abdelwahed Khamis, Ali Zia, Mohammed Hassanin
	SAN [103]		✓		Luca Scofano, Indro Spinelli, Simone Scardapane, Simone Fiorellino, Olga Zaghen, Lev Telyatnikov, Claudio Battiloro (first place)
CC	CWN [29]		✓	✓	Dmitrii Gavrilev, Gleb Bazhenov, Suraj Singh (second place)
	CAN [104]			✓	Luca Scofano, Indro Spinelli, Simone Scardapane, Simone Fiorellino, Olga Zaghen, Lev Telyatnikov, Claudio Battiloro (first place), Abraham Rabinowitz
CCC	HOAN [120]		✓	✓	Rubén Ballester, Manuel Lecha, Sergio Escalera (first place), Aiden Brent (second place)

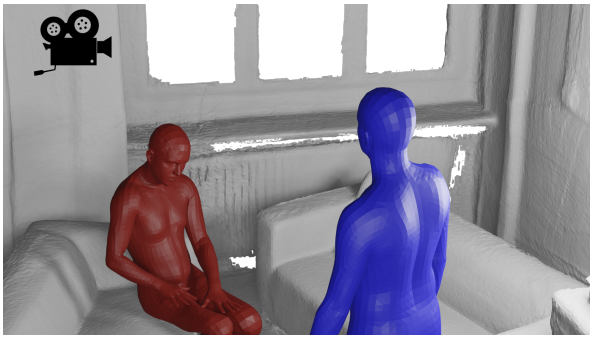
Table 6.2: Model implementations submitted to the Topological Deep Learning Challenge. We organize original models according to domain: hypergraph (HG), simplicial (SC), cellular (CC), and combinatorial (CCC). Task level indicates the rank on which a prediction is made.

Chapter 7

Social EgoMesh Estimation

7.1 Introduction

(a) Wearer's first person view, looking at the interactee.



(b) Third-person view of the wearer and the interactee in a scene.

Figure 7.1: *(Left)* Frame from the input egocentric video stream. We experience the immersive subjective perspective of the front-facing camera wearer, but the wearer is behind the wearable and, therefore, invisible. Still, we recognize the points of interest of the wearer, parts of the scene where the action happens, and, most importantly, the interactee engaged in communication with the wearer. *(Right)* Third-view reconstruction of the *ego* mesh of the camera wearer by our proposed **SEE-ME**. Our vantage point reveals the surrounding environment, featuring a sofa and a person from an overhead perspective, leading us to infer the wearer's likely standing position. Specifically, vicinity and gaze interactions are important cues for our reconstruction as we experimentally quantify.

Estimating the 3D motion of a person from an egocentric video sequence is a critical task with diverse applications in virtual reality (VR) and augmented reality (AR). This research is motivated by the necessity to realistically depict the entire body to enable total immersion in these environments [56]. The user typically wears a camera in such videos, capturing their surroundings from a first-person perspective. Large field of view (FOV) cameras can capture some body parts (hands, feet), which standard FOV cameras cannot, as illustrated from the first-person perspective in Figure 7.1.

The task, dubbed egopose estimation, is influenced by the type of camera used to record the sequences. Top-down head-mounted cameras have the best view of the wearer [305] but are intrusive due to the required displacement from the face. Front-facing cameras solve the ergonomic issue, but the downside is that the wearer is almost always invisible except when using very large FOV.

In [145], the authors introduce a two-stage framework for fish-eye head-mounted cameras, where a traditional SLAM algorithm [296] provides the camera rotation and translation. Simultaneously, a neural

network exploits visible body parts to predict the egopose. Recently, EgoEgo [181] proposed an updated pipeline, relaxing the assumption on the large FOV. However, they still rely on (deep learning-based) SLAM [302] to estimate the head's rotation and translation, fed to a generative model [130] to synthesize the pose for the rest of the body. These approaches exploit the video to extract the head pose, ignoring the surrounding environment and the other actors in the scene that can provide essential cues for the egopose estimation.

For this reason, we propose a new social egocentric mesh estimation task and contribute with a novel probabilistic framework for **S**ocial **E**gocentric **E**stimation of body **ME**shes (**SEE-ME**). Humans have a social nature and are often involved in social activities in real and virtual worlds, accomplishing tasks that usually require cooperation and coordination. You2Me [235] explicitly captures the interplay between two persons from a chest-mounted camera in a controlled environment where two people interact by conversing, playing hand games, or doing sports. However, their approach does not extract high-level scene information and predicts deterministic skeleton-like representation that may not be suitable for AR/VR applications. In contrast with our work, [235] employs a person-centric coordinate system and does not estimate global orientation or translation, while our approach enables end-to-end learning of both of them. [235] also relies on OpenPose [42] to extract the interactee, which faces challenges in accurately estimating occluded keypoints [196]. We exploit EgoHMR [361] to estimate the interactee's mesh, which demonstrates robustness against occlusions, by utilizing scenes as input. In contrast, our approach leverages the scene and interactions to understand the invisible camera wearer's pose, with no input body cues required.

The motivation behind our research stems from the wearer's perspective, which highlights both their surroundings and an overlooked aspect: the presence of another actor within the scene (see Figure 7.1). Our unified approach links the generation of the egomesh to both the 3D scene depiction and the wearer's social interaction with the interactee. We do not rely on SLAM for the head's orientation or translation but predict them with our model. Our approach pioneers conditional Latent Diffusion in egocentric human mesh estimation, which provides a notable speedup in generation times. We build the latent space encoding human poses using a Variational Autoencoder (VAE) [151]. Subsequently, we perform conditional diffusion on this latent space. The conditioning strategies aim to guide the process by modeling the 3D point cloud of the scene in which the wearer moves and the estimated mesh of the interactee recovered from the egocentric video feed. Our model depends on the 3D scene and other actors' poses. Therefore, provided the wearer's pose, we can predict the interactee's one even if it is not paired with an egocentric video stream.

To assess the performance of our approach, we evaluate it on EgoBody [362]: the solely available dataset that features multiple individuals, an egocentric perspective, and an environment. We reach state-of-the-art results, establishing the efficacy of environmental and social components. In our research, we thoroughly examine the influence of social interactions on estimating the wearer's egomesh. We establish when this conditioning strategy has the highest impact. We use proxies for social interaction that can be extracted from our setup. We demonstrate how proximity and eye contact between the wearer and the interactee bring the most from our conditioning. Moreover, we study the effect of future knowledge about the interactee's motion, as we humans use the experience to predict and react to the future movements of the people around us. Our framework, exposed to this information, provides a significant performance boost. To corroborate the strength of our approach, we evaluate our model on the dataset GIMO [372], which is egocentric and includes the environment but is not multi-person.

To summarize our contributions:

- We propose the task of social egomesh estimation. Emphasizing the role of other actors in 3D scenes to account for the lack of information about the camera wearer.

- We introduce SEE-ME, a monolithic framework based on latent diffusion capable of predicting both the wearer and interactee poses with SOTA performances
- We perform an in-depth ablation study to highlight the scenarios in which modeling social interactions brings the most benefits.

7.2 Related Work

Pose Estimation from third-person cameras. 3D pose estimation from images and videos in a third-person perspective has seen significant research efforts in recent years. It can be broadly categorized into two main approaches. The first approach aims to predict the positions of joints from images and videos directly [159, 206, 219, 246, 310, 375]. The second approach employs a parametric human body model [199] to estimate the parameters of the body model based on images or videos [60, 147, 159, 160, 206, 344]. Recent approaches have advanced the realism of human motion modeling by incorporating human dynamics. This is achieved through the utilization of learned priors [268] or physics-based priors [96, 251, 269, 289, 334, 349]. These priors are inherently formulated within the human coordinate frame.

However, a notable challenge arises when dealing with egocentric videos, which we consider in this study, where the whole body is often not visible because body joints are mostly hidden from view. By considering the physical and social aspects, we aim to provide we aim to overcome this lack of information.

Motion Estimation from Egocentric Video. There is a growing emphasis on pose estimation from egocentric videos, where various hardware setups are utilized, including fisheye cameras [6, 133, 196, 304, 317], outward-facing cameras [144, 235, 290, 345, 361, 362, 372], and additional inputs such as controllers and synchronized headsets [146, 371], all aimed at estimating a person's pose. None of them consider social interactions. However, there has been a growing interest in incorporating interactions between people in a given scene, as evidenced by numerous studies [148, 235, 361, 362]. Notably, [361, 362] both focus on estimating the interactee without relying on any additional cues from the camera wearer. In contrast, You2Me [235] predicts full-body wearer motions by observing the interaction poses of a second person in the camera view. While this approach is effective for keypoint estimation, it lacks critical information necessary for our task, such as body meshes, 3D scene details, and global rotation and orientation. Conversely, our approach involves probabilistic human mesh estimation, incorporating scene information as well.

In [181], a distinctive method is proposed for multi-hypothesis human mesh estimation. This approach decouples the problem by estimating head movement, employing it as a conditioning variable for a DDPM [130] model to generate plausible poses. In contrast, our approach eliminates the reliance on a three-block process involving SLAM [302], Gravitynet, and Headnet [181]; tackling the problem as a unified mesh estimation task. We employ a Latent Diffusion Model to generate the entire body without preprocessing steps such as optical flow estimation and camera localization. Furthermore, our approach incorporates the social component as a conditioning factor.

Probabilistic models for human pose estimation. Due to limited image or video observations and inherent depth ambiguity, estimating a 3D human pose from a single image can give rise to numerous potential solutions, mainly when body truncation is a factor. Recent research endeavors have approached this challenge by framing it as a generative process or predicting multiple hypothetical poses. In various studies, a discrete set of hypotheses has been generated to address this issue, as seen in [26, 141, 161, 178, 181, 196,

236, 326, 361]. We extend the advancements made in recent motion diffusion models [46, 52, 67, 303, 359]. Our approach further integrates conditioning on bystanders or individuals within the scene. It is crucial to highlight that only [361] considers the 3D scene constraint. Our methodology uses the latent diffusion process to capture the inherent ambiguity in pose estimation, incorporating flexible scene and social conditioning techniques.

7.3 Methodology

Modeling social interaction requires solving a series of challenging tasks. Note, as done in [181, 362], we pose the origin of our coordinate system as the camera of the wearer. First, we extract a 3D representation of the interactee's human body from a video stream. Subsequently, we need to position this representation into a depiction of the 3D environment where the interaction occurs. Hence, this is the information that SEE-ME exploits to recover the camera wearer's mesh using only the egocentric video stream, where the wearer itself is not visible. We showcase our pipeline in Figure 7.2. Before using social interaction and scene description to drive the generation process, we train part of our model to encode human motion in a latent space (Sec. 7.3.1). This dramatically reduces the dimensionality required to represent a person. In this space, we learn to synthesize human motion from pure noise using latent diffusion processes (Sec. 7.3.2).

Problem formalization. Our objective is to generate a plausible representation of the camera wearer $\mathbf{P^w}$, conditioned on the other person in the scene $\mathbf{P^i}$ and the 3D scene $\mathbf{S}$. We define the sequence of poses of the camera wearer as $\mathbf{P^w} = \{\mathbf{p^w}_k\}_{k=1}^{F} \in \mathbb{R}^{F\times V}$, where F and V are the number of frames and the parameters of the pose vector. Similarly, $\mathbf{P^i} = \{\mathbf{p^i}_k\}_{k=1}^{F} \in \mathbb{R}^{F\times V}$. Following literature [52, 201, 358, 359], our poses vectors $\mathbf{P^w}$ and $\mathbf{P^i}$ contain pose, translation and rotation and represents the SMPL [199] body representation. Finally, $\mathbf{S}$ is defined as a 3D scene point cloud, denoted as $\mathbf{S} \in \mathbb{R}^{N\times 3}$, where N is the number of points.

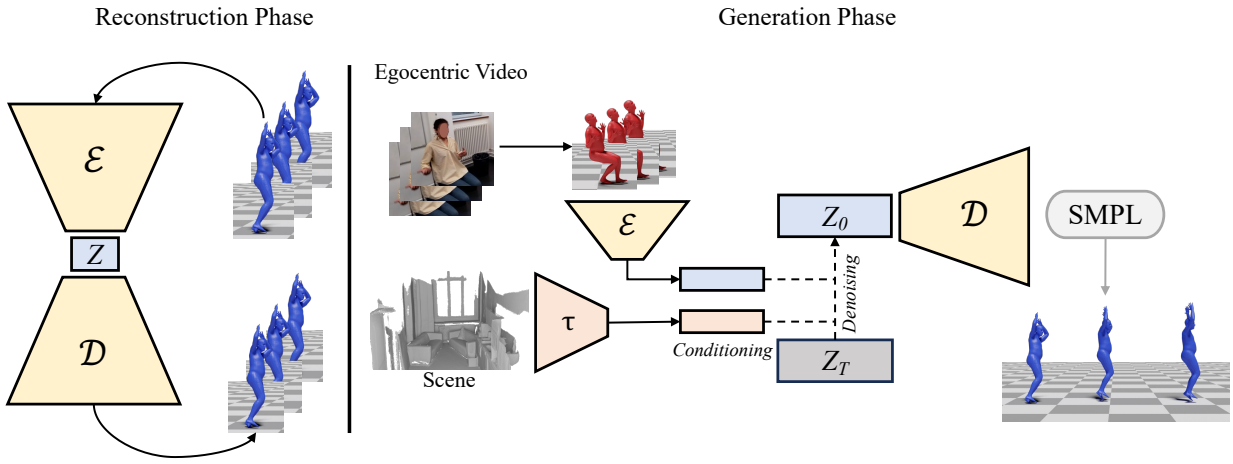


Figure 7.2: SEE-ME framework. On the left, we present VAE's training to learn a meaningful latent space by solving reconstruction tasks. On the right, we extract and process our conditioning strategies. Corresponding to the 3D point cloud representation of the scene and the interactee's pose extracted from the video sequence. After the conditional denoising process, we can output a SMPL representation of the wearer's pose.

7.3.1 Latent Human Representation

VAEs consist of an encoder-decoder generative architecture trained to minimize the reconstruction error. We employ it to reduce the pose vector V dimensionality, projecting onto a manifold of feasible poses. In this framework, the encoder network $\mathcal{E}$, parameterized by ϕ, generates lower-dimensional embeddings $\mathbf{z} \in \mathbb{R}^{n \times D}$ as outputs based on the input poses $\mathbf{P} = \{\mathbf{p}_k\}_{k=1}^{F}$. Following VAE literature [151], $q_\phi(\mathbf{z}|\mathbf{p})$ approximates the true posterior distribution of the latent space with a multivariate Gaussian with a diagonal covariance structure:

$$q_\phi(\mathbf{z}|\mathbf{p}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{p}), \sigma_\phi^2(\mathbf{p})\mathbf{I}), \tag{7.1}$$

where $\mu_\phi(\cdot)$ and $\sigma_\phi^2(\cdot)$ are the encoder's outputs. We sample from the approximated posterior $\mathbf{z}_i \sim q_\phi(\mathbf{z}|\mathbf{p}_i)$ using:

$$\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i^2 \odot \boldsymbol{\rho}, \tag{7.2}$$

where $\mathbf{z}_i$ is a one-dimensional vector of size D, and $\boldsymbol{\rho}$ is sampled from a standard multivariate Gaussian distribution. The decoder network $\mathcal{D}$ parametrized by θ maps the sampled values back to body poses $p_\theta(\mathbf{p}|\mathbf{z})$ mapped into body meshes with the differentiable SMPL model. The network parameters are obtained by optimizing the Evidence Lower Bound (ELBO) objective, as described in [151]. The motion decoder $\mathcal{D}$ relies on a transformer decoder architecture [52, 252] with a cross-attention mechanism, taking f^* zero motion tokens as queries, where f^* is the target sequence length, and a latent $\mathbf{z} \in \mathbb{R}^{1 \times D}$ as memory, ultimately generating a human motion sequence $\hat{\mathbf{p}}_{1:f^*}$.

7.3.2 Ego-Mesh Estimation via Latent DDPMs

We utilize the latent-DDPM framework introduced in [271] and adapted for human motion synthesis in [52] where the diffusion process occurs on a condensed, low-dimensional motion latent space. Latent DDPM have a different approximated posterior $g(\mathbf{z}_t|\mathbf{z}_{t-1})$, denoted *diffusion process*, which gradually gradually converts latent representations $\mathbf{z}_0 = \mathbf{z}$ into random noise $\mathbf{z}_T$ in T timesteps:

$$g(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \sqrt{\bar{\alpha}_t}\mathbf{z}_{t-1}, (1-\bar{\alpha}_t)\mathbf{I})\,, \tag{7.3}$$

where $\bar{\alpha}_t$ is a scaling factor specific to timestep t. Then, the reverse process dubbed *denoising* gradually refines the noised vector to a suitable latent representation $\mathbf{z}_0$. Following [52, 72, 130, 271], we use the notation $\{\mathbf{z}_t\}_{t=0}^{T}$ to denote the sequence of noised latent vectors, with $\mathbf{z}_{t-1} = \epsilon_\psi(\mathbf{z}_t, t)$ representing the denoising operation at time step t. Here, ϵ_ψ refers to a denoising autoencoder trained to predict the denoised version of its input.

$$Loss := \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\mathbf{I}),t}[\|\epsilon - \epsilon(\mathbf{z}_t, t)\|_2^2]\,. \tag{7.4}$$

During the denoiser's training, the encoder and decoder remain frozen. In the subsequent diffusion reverse stage, $\epsilon_\psi(\mathbf{z}_t, t)$ predicts $\hat{\mathbf{z}}_0$ through a series of T iterative denoising steps. Following this, the decoder $\mathcal{D}$ translates $\hat{\mathbf{z}}_0$ into poses $\mathbf{P^i}$ and meshes in a single forward pass.

7.3.3 Social Conditioning

We introduce a novel conditioning strategy to embed the knowledge about social interaction into the generation of the wearer's mesh. Part of the DDPM's great success is the ability to incorporate signals $\mathbf{c}$ to drive the trajectories of the (conditional) denoising process $\epsilon_\psi(\mathbf{z}_t, t, \mathbf{c})$. We add a new domain encoder and exploit

Table 7.1: Egocentric Pose Estimation Results. We activate and deactivate Scene and interactee conditioning to assess their contribution. In each version, our framework improves SoA performances by a large margin. We obtain the best results when the scene and interactee conditioning are present.

Models	**Conditioning**		**EgoBody [362] Dataset**			
	Scene	Interactee	MPJPE (mm)	Orientation Error	Translation Error (mm)	Acceleration Error (mm/s^2)
EgoEgo [181]			268	**0.40**	207	10.8
SEE-ME w/o Scene		✓	138	0.51	174	3.07
SEE-ME w/o Int.ee	✓		130	0.51	171	2.69
SEE-ME	✓	✓	**126**	0.48	**164**	**2.67**

our VAE to accommodate our two conditions. The first one compresses the scene's point-cloud representation $\mathbf{S}$ using an MLP network τ_η consisting of several residual blocks to encode the global input scene into a scene feature vector. To encode interactee's poses $\mathbf{P^i}$, we exploit our pre-trained VAE encoder. At training time, we use ground truth poses; at test, we rely on SOTA egocentric mesh recovery algorithm EgoHMR [361]. To inject these embedded conditions into the transformer-based denoiser, we apply cross-attention. Then, the conditional objective is defined as follows:

$$\mathcal{L} = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}),t,\mathbf{P_i},\mathbf{S}}[\|\epsilon - \epsilon_\psi(\mathbf{z}_t, t, \mathcal{E}_\phi(\mathbf{P_i}), \tau_\eta(\mathbf{S})\|_2^2]\,, \tag{7.5}$$

where $\mathcal{E}_\phi$ and τ_η stay frozen.

7.4 Experiments

In this section, we validate our model against the state of the art and showcase the qualitative results of our approach. Additionally, we perform several ablation studies highlighting the impact of social interactions. Below, we define the dataset and the reference metrics.

Dataset. We employ the EgoBody [362] dataset to assess our technique, the sole available egocentric and social dataset featuring an environment and body meshes defined by parametric body models [199] for both the wearer and the interactee. The dataset is recorded using Microsoft Hololens2 [1] and contains 125 sequences at 30 fps. The total number of frames is 220k, but, as in [362], we selectively use frames where the interactee is within the wearer's field of view. The training set comprises 90000 images, the validation set consists of 23000 images, and the test set includes 62000 images. These recordings occur in 15 indoor scenes, each accompanied by 3D representations, which are recorded using an iPhone 12 Pro Max running a 3D Scanner App (for additional information, refer to [181]).

To enhance comparability, we also include the GIMO dataset [372] and benchmark SEE-Me without Int.ee—meaning the variant of SEE-ME based solely on the scene, as GIMO contains only single-person videos. Although this specific setup excludes social cues, we conduct this evaluation to further assess our overall framework. This dataset features egocentric views in single-body environments, and we adapt our model accordingly. It has been acquired from 11 different subjects performing actions in 19 3D scenes, scanned using an Apple iPhone 13 Pro Max. The dataset includes up to 125,400 egocentric images captured at 30 fps with a Hololens2 and 217 trajectories captured by an Inertial Measurement Units (IMU) system. In total, GIMO contains approximately 129,000 frames

Baselines. To estimate the wearer's pose from a first-person perspective, we evaluate our model against EgoEgo [181], the existing state-of-the-art model designed for estimating the wearer's pose from an egocentric video. EgoEgo is a probabilistic model based on DDPM [130] that leverages head poses as conditioning to generate a plausible pose. The model consists of two modules. Firstly, it estimates the head pose by using two sub-modules. The first module estimates the global orientation and translation of the camera based on optical flow. The second module involves head pose estimation using DROID-SLAM [302]. By combining these estimates, the output is the global position of the head, comprised of pose, orientation, and trajectory. During the second phase, the estimated head is employed as a conditioning factor for a DDPM model, facilitating the generation of realistic poses. We employ EgoEgo's [181] configuration, where body poses are represented relative to the initial pose. Concerning the scene alignment, we adopt an approach in line with EgoHMR[361].

Metrics. We consider the evaluation metrics commonly employed in the current literature on egocentric human pose estimation [181, 361, 362].
Specifically, we evaluate the accuracy of our model by computing the error on the keypoints extracted from the SMPL body representation, ignoring the mesh form factor. We use SMPL due to its widespread adoption for representing virtual humans. The added realism brought by meshes allows for augmenting the fidelity of interactions, for example, by modeling collisions. The Mean Per Joint Position Error (MPJPE) measures the average Euclidean distance between the predicted and ground truth 3D positions of individual joints across a sequence of frames:

$$\text{MPJPE} = \frac{1}{J \times T} \sum_{j=1}^{J} \sum_{t=1}^{T} \|\mathbf{p}_{j,t} - \mathbf{g}_{j,t}\|^2. \tag{7.6}$$

We measure the Orientation Error using the Frobenius norm of the 3x3 rotation matrix of the reference joint $\mathbf{A}_{pred}$ predicted and ground truth $\mathbf{A}_{GT}$, expressed as follows:

$$\text{Orientation Error} = ||\mathbf{A}_{pred} - \mathbf{A}_{GT}^{-1} - \mathbf{I}||_2. \tag{7.7}$$

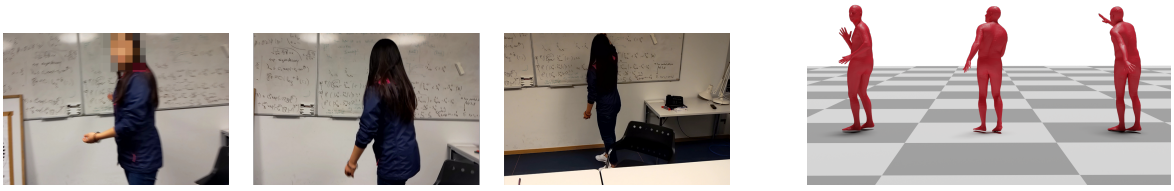
To evaluate the error of the generated motion translation, we use the Euclidean distance between the predicted trajectory $\mathbf{r}^{pred}$ and the ground truth $\mathbf{r}^{GT}$:

$$\text{Translation Error} = \frac{1}{T} \sum_{t=1}^{T} ||\mathbf{r}_t^{pred} - \mathbf{r}_t^{GT}||^2. \tag{7.8}$$

We then compute the acceleration for predicted poses $\mathbf{a}^{pred}$ and ground truth $\mathbf{a}^{GT}$, and we define the Acceleration Error as their Euclidean distance. We employ millimeters (mm) for MPJPE and Translation, millimeters per second squared (mm/s^2) for Acceleration, and the Frobenius norm between the rotation matrices for Orientation.

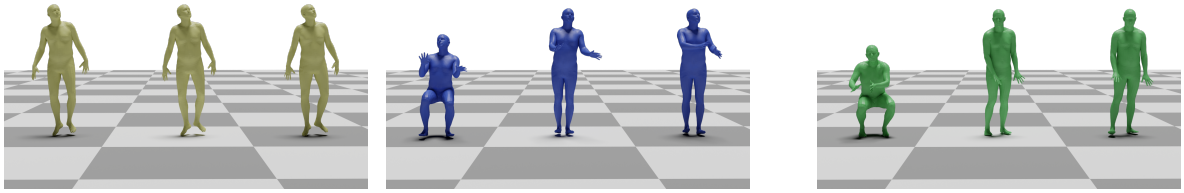
7.4.1 Comparison with SOTA

We quantitatively assess the camera wearer's pose using the results from the EgoBody [362] dataset. As the GIMO [372] dataset lacks any interactee, we benchmark only the SEE-ME variant, focusing solely on the scene. Furthermore, we conduct several studies to explore the impact of social relation proxies and their effects.



(a) Video stream of the wearer's egocentric view, not used in the model.

(b) Wearer's view of the interactee, extracted from the video stream.



(c) Prediction of the wearer's pose by EgoEgo.

(d) Prediction of the wearer's pose by SEE-ME.

(e) Ground truth wearer pose.

Figure 7.3: Front view of 3 frames extracted from an egocentric sequence. We compare SEE-ME (blue) with EgoEgo (yellow), and the ground truth (green). In red we have the interactee's poses, extracted from the egocentric video sequence next to it, but not used in our model.

Social egocentric human pose estimation. Table 7.1 compares our model quantitatively against the current leading technique [181]. We retrained the HeadNet module of their model on the EgoBody [362] dataset, specifically targeting head rotation and translation distance estimation. The rest of the model's modules, initially trained on a large-scale motion capture dataset like AMASS [210], were frozen during this process.

We improve the performance of MPJPE from 268mm to 126mm over the state-of-the-art [181] yielding a 53% enhancement in MPJPE, 21% in Translation, and a significant 75% in Acceleration. The impact of the interactee is evident, even when the wearer is not engaged in active interaction with it. Furthermore, compared to [181], conditioning solely on the interactee enhances the performance from 268mm to 138mm in MPJPE, giving a 49% increase and a substantial 72% improvement in the Acceleration error between predicted and ground truth joints. Predicting the wearer's pose conditioning on the scene alone allows a direct comparison with EgoEgo, and our model proves again to boost performances. Going from 268mm to 130mm, we get a 51% MPJPE improvement, predicting the wearer's pose from the scene alone We get 17% and 75% improvements in translation and acceleration errors, respectively.

Models	MPJPE (mm)	Orientation Error	Translation Error (mm)
PoseReg [348]	189	1.51	1528
Kinpoly-OF [207]	404	1.52	1739
EgoEgo [181]	152	0.67	**356**
SEE-ME w/o Int.ee	**141**	**0.61**	843

Table 7.2: Quantitative results on GIMO [372] dataset.

Egocentric human pose estimation. To reinforce the effectiveness of our approach, we further assess SEE-ME using the GIMO dataset, as there is a scarcity of 3D social egocentric environmental datasets aside from EgoBody. Thus we set to assess the quality of SEE-ME by the sole consideration of the scene. The results shown in Table 7.2 are comparable to EgoEgo and they should be paralleled with SEE-ME w/o Int.ee in the prior experiment, as the interactee is absent in GIMO. SEE-ME w/o Int.ee outperforms EgoEgo on the orientation estimation but it yields a larger translation error as EgoEgo leverages information from SLAM. Overall, on the general MPJPE performance, SEE-ME w/o Int.ee outperforms EgoEgo by 7.2%,

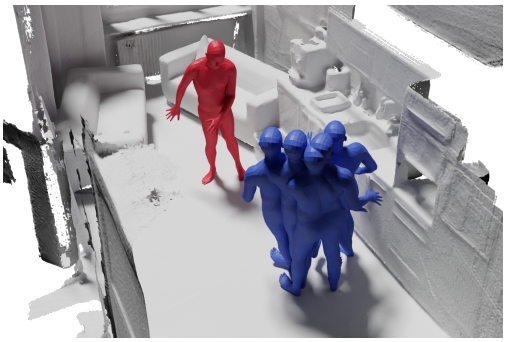


Figure 7.4: The interactee (red) influences the wearer's motion (blue).

which reasserts the quality of the proposed model, across different settings.

7.4.2 Qualitative results

We show the results obtained from SEE-ME qualitatively, comparing them to the state-of-the-art [181]. In Figure 7.3, the input footage (a) is processed to extract the interactee's pose by EgoHMR [361] (b), which is then ingested by SEE-ME, alongside the encoding of the scene point cloud. The qualitative reconstructions match the quantitative evaluation of Table 7.1 as SEE-ME (d) adheres better to the ground truth, than EgoEgo (c) does. E.g. the actor stands up in (d) to attend to the interactee.

Figure 7.4 depicts a different qualitative study case: the interactee's pose (red) conditions the generation of the ego actor (blue), which SEE-ME successfully positions in front of it, taking plausible poses while conversing.

7.4.3 Ablation studies

Social Interaction ablation. We explore the influence of social relations through ablation studies to gain a deeper understanding of the interactee's impact. To quantify interaction, we initially employ a straightforward proxy: the distance between individuals. This is achieved by categorizing distances into three ranges based on the root joint. As illustrated in Table 7.3, the proximity of the interactee and wearer correlates with more pronounced effects on the final results. Notably, the most substantial improvements are observed in MPJPE (by 6%), Translation Error (by 9%), and Acceleration (by 23%) when considering proximate interactions. The diminished performance in Translation and Acceleration beyond the 2-meter range is attributed to the wearer moving toward the interactee. As individuals draw closer, there is a heightened relative acceleration and translation, making it more susceptible to errors.

Wearer-Interactee Distance Threshold (m)	MPJPE (mm)	Orientation Error	Translation Error (mm)	Acceleration Error (mm/s^2)
-	126	**0.48**	164	2.67
$d > 2$	132	**0.48**	183	2.76
$1 < d < 2$	128	0.49	161	2.73
$d < 1$	**119**	**0.48**	**156**	**2.06**

Table 7.3: Ablation study on the interpersonal distance between wearer and interactee. Conditioning on the interactee's pose works best when in close proximity.

The investigation in Table 7.4 aims to ascertain mutual gaze fixation between two individuals using information derived from their respective head rotation matrices. Mutual gaze, a critical component of non-verbal communication, can indicate interpersonal engagement. The proposed method relies on converting

rotation matrices to Euler angles, extracting gaze directions, and evaluating the angular deviation between these vectors. We employ 30° and 60° thresholds to determine whether two individuals are making eye contact. We look at two data subsets for each threshold: one satisfying the condition and the other not. Remarkably, both thresholds yield improved results when satisfied, affirming the initial intuition. The most substantial improvement occurs when individuals are directly looking at each other (30°), leading to enhancements in MPJPE from 127mm to 117mm (by 8%), from 175mm to 137mm in Translation Error (by 22%), and also enhancing the Orientation Error (14%), and Acceleration (by 9%) for when the condition not being met.

Wearer-Interactee looking at each other	FOV	MPJPE	Orientation Error	Translation Error	Acceleration Error
-	-	126	0.48	164	2.67
No	60	127	0.48	173	2.71
Yes		**123**	**0.48**	**157**	**2.57**
No	30	127	0.50	175	2.79
Yes		**117**	**0.43**	**137**	**2.54**

Table 7.4: Ablation study on gaze directions. By considering an angle of 60 and 30 degrees, we asses if the wearer and the interactee are looking at each other. If this is the case, the conditioning boots improve the performance even more.

Finally, we examine a scenario in which the wearer possesses knowledge of the interactee's future movements. This is achieved by conditioning on future frames rather than on the present. As indicated in Table 7.5, the MPJPE and the Translation Error exhibit improvements of 2% and 22%, respectively. The latter's enhancement can be attributed to the wearer's anticipation of the interactee's movements, reducing Translation error.

Wearer's input = present Interactee's input = x	MPJPE	Orientation Error	Translation Error	Acceleration Error
x = present	126	**0.48**	164	**2.67**
x = future	**123**	**0.48**	**128**	3.40

Table 7.5: Ablation study on wearer poses conditioned on the interactee's present and future ones. Even a little glance into the future reduces the MPJPE.

Implementation details. Our framework consists of three main components: the encoder, the decoder, and the denoiser. The reconstruction phase, which has 24 million parameters, is handled by the encoder and decoder, while the generation phase involves the denoiser (9 million parameters) and the frozen decoder. Each component has 9 layers 4 heads, and a latent space whose dimensionality is 256. The encoder and the decoder both use standard transformer layers. Based on [359], the denoiser combines classical self-attention on the latent vector with linear cross-attention between the latent vector and textual input. The batch size is 64 for the first phase and 128 for the second one, and the AdamW optimizer is used with a learning rate of 10^{-4}. The diffusion steps are set to 1000 and 20 during training and inference. We train on 8 Tesla V100 GPUs for 3k epochs for both phases.

Limitations and Future Work. We do not inject any type of social relations knowledge externally but leave the model free of learning from the dataset. This means that the quality of the predictions is highly dependent on the training data distribution. While we recognize the presence of social relations in the utilized dataset, EgoBody[361], we acknowledge that these relationships could be more effectively leveraged

in more diverse and socially engaging datasets, such as EgoHumans [148] for which currently both mesh recovery and 3D point cloud are not available. Additionally, we recognize the potential to enhance our model by exploiting large-motion datasets such as AMASS, and possibly others such as KIT, during the VAE training for the reconstruction phase. This would allow us to capture movements beyond the constraints of a single dataset. Currently, our model works without taking any wearer's input, but considering its egocentric observed body parts as additional data could be a research path to explore, and which could further improve our performances.

7.5 Conclusions

In conclusion, accurately determining the 3D pose of the camera wearer in egocentric video sequences is pivotal for advancing human behavior modeling in virtual and augmented reality applications. Despite the challenges posed by limited visibility, SEE-ME has demonstrated that the pose of the wearer can be reconstructed to an improved level of accuracy.

By solely utilizing a latent probabilistic diffusion model, our approach integrates conditioning techniques that effectively capture both social interactions and the surrounding environment. Moreover, it is straightforward and does not require any extra preprocessing or overhead compared to other methods that utilize localization techniques. This development shows great potential for improving the realism and precision of egocentric video-based human behavior modeling, particularly for applications in augmented reality/virtual reality (AR/VR) and embodied AI.

Chapter 8

PREGO: online mistake detection in PRocedural EGOcentric videos

8.1 Introduction

Egocentric procedure learning is gaining attention due to advancements in Robotics and Augmented Reality (AR) technologies. These technologies are pivotal to enhancing online[1] monitoring systems, offering real-time feedback, and improving operator efficiency in various fields. Recent works have produced numerous datasets [22, 74, 80, 100, 143, 221, 260, 279, 279, 284, 301, 323, 376], methodologies aimed at advancing procedure learning [100, 143, 203, 301, 324, 373] and error detection models [74, 279, 323]. Despite these advancements, as outlined in Table 9.1, state-of-the-art methods typically focus on supervised and offline mistake detection. They are unsuitable for situations requiring dynamic decision-making, specifically within an *online* setting, or when errors occur unpredictably, thus defining these instances as open-set conditions.

In this work, we propose the first model to detect PRocedural errors in EGOcentric videos (PREGO), which operates online, thus causal, and can recognize unseen procedural mistakes, fitting for open-set scenarios. We prioritize egocentric videos due to their highly detailed perspective, essential for accurately identifying steps within procedures. Additionally, the widespread use of egocentric cameras in industries [253] necessitates the development of online error detection techniques to improve the safety and efficiency of workers. The online attribute is achieved by analyzing input videos sequentially up to a given frame t, ensuring that no future actions influence the current step recognition. On the other hand, open-set learning is performed by exclusively exposing PREGO to correct procedural sequences when predicting mistakes, following the One-Class Classification (OCC) paradigm [91, 354]. Any step within a procedure that significantly diverges from the expected correct patterns is identified as an error, allowing PREGO to recognize a wide range of procedural mistakes without being confined to a restricted set of predefined ones.

PREGO's architecture is dual-branched, as depicted in Fig. 9.1. The first branch, the *step-recognition branch*, analyzes frames in a procedural video up to a current time t, aiming to classify the action being undertaken by the operator. This branch can exploit the current state-of-the-art video-based online step recognition model, [10, 324]. Concurrently, the second branch is in charge of *step-anticipation*, tasked to predict the action at time t, based solely on the steps up to $t-1$. We propose using a pre-trained

[1]Most workflows can be aided by *online* monitoring algorithms, which provide feedback to the operator in due course. However, they may lag due to processing or connectivity delays. We distinguish online from real-time, whereby the second has strict requirements of instantaneous response.

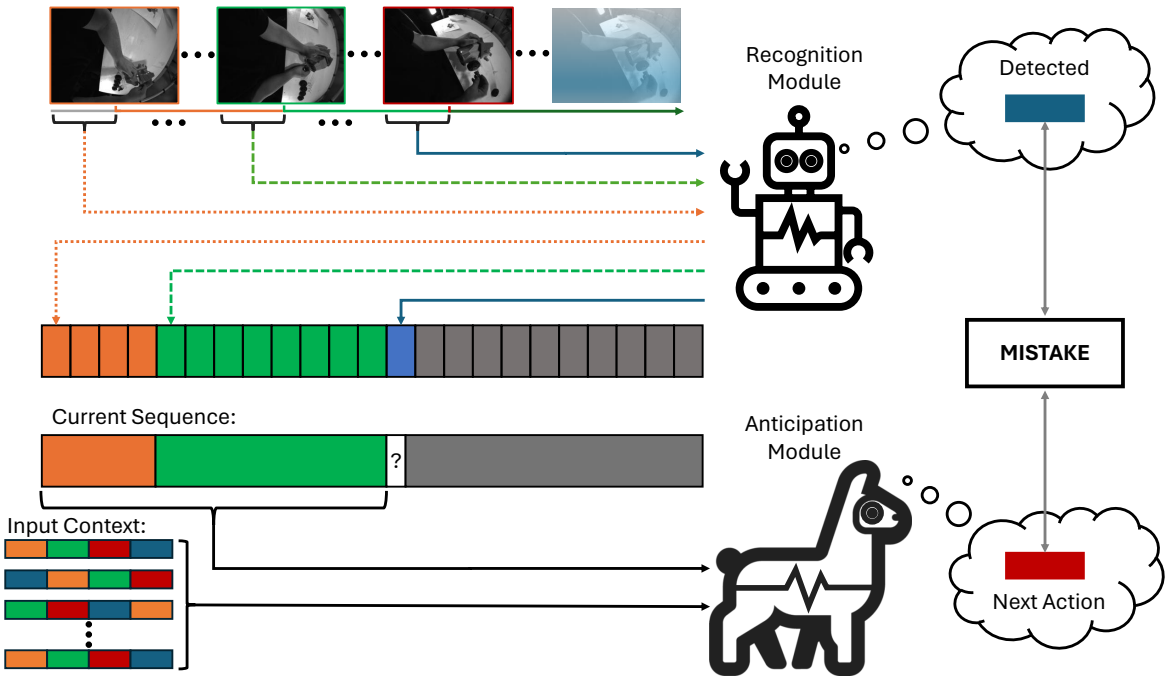


Figure 8.1: PREGO is based on two main components: The recognition module (top) processes the input video in an online fashion and predicts actions observed at each timestep; the anticipation module (bottom) reasons symbolically via a Large Language Model to predict the future action based on past action history and a brief context, such as instances of other action sequences. Mistakes are identified when the current action detected by the step recognition method differs from the one forecasted by the step anticipation module (right).

Table 8.1: Comparison among relevant models. In the modalities column, *RGB* stands for RGB images, *H* for hand poses, *E* for eye gaze, *K* for keystep labels. Differently from previous works, we are the first to consider an egocentric one class and online approach to mistake detection.

	Ego	OCC	Online	Modalities	Task	Datasets
Ding et al. [74] - *Arxiv '23*				*K*	Mistake Detection	Assembly101 [284]
Wang et al. [323] - *ICCV '23*	✓			*RGB+H+E*	Mistake Detection	HoloAssist [323]
Ghoddoosian et al. [100] - *ICCV '23*				*RGB*	Unknown sequence detection	ATA [100] and CSV [258]
Schoonbeek et al. [279] - *WACV '24*	✓		✓	Multi	Procedure Step Recognition	IndustReal[279]
PREGO	✓	✓	✓	*RGB*	Mistake Detection	*Assembly101-O, Epic-tent-O*

Large Language Model (LLM) [307] for zero-shot symbolic reasoning through contextual analysis [115, 222, 298]. An error is detected upon a misalignment between the currently recognized action and the anticipated one, thereby signaling a deviation from the expected procedure. Utilizing correctly executed procedures as instances in the query prompt obviates the necessity for additional model fine-tuning and leverages the pattern-completion abilities of LLMs. Our proposed approach is an abstraction from the video content. Using labels allows for longer-term reasoning, as a label summarizes several frames. Also, this approach is an alternative to the carefully constructed action inter-dependency graphs [16]. We demonstrate that symbolic reasoning subsumes understanding lengthy procedures and the action inter-dependencies, suggesting repositioning from semantic-based expressions of procedures to an implicit representation, where only patterns of symbols have to be recognized and predicted. By representing procedures as sequences and their steps as symbols, we let the predictor focus on the patterns that characterize the correct procedures.

To support the evaluation of PREGO, we adapt the procedural benchmarks of Assembly101 [284] and Epic-tent [143], formalizing the novel task of online procedural mistake detection. In the adapted online mistake detection benchmarks, which we dub Assembly101-O and Epic-tent-O, the model is tasked with

detecting when a procedural mistake is made, thus compromising the procedure. The compromising mistake may be a wrong action or a relevant action performed in such an order that the action dependencies are not respected.

We summarize our contributions as follows:

- We present PREGO, the first method designed for online and open-set detection of procedural errors in egocentric videos. PREGO's online feature ensures causal analysis by sequentially processing input videos up to a given frame, preventing future actions from influencing current step recognition.
- PREGO achieves open-setness by exclusively relying on correct procedural sequences at training time, following the One-Class Classification (OCC) paradigm. This allows PREGO to identify a wide range of procedural mistakes, avoiding confinement to a predefined set of errors and avoiding the need for fine-grained mistake annotations.
- We propose using a pre-trained LLM for zero-shot symbolic reasoning through contextual analysis to predict the next action.
- To evaluate PREGO, we introduce the novel task of online procedural mistake detection and re-arrange existing datasets to provide two new benchmarks, referred to as Assembly101-O and Epic-tent-O.

8.2 Related Work

8.2.1 Procedural Mistake Detection

Procedural learning has seen significant advancements with the creation of diverse datasets [80, 221, 260, 301, 376] that provide insights into both structured [22, 259, 323] and unstructured [68, 143] tasks, covering a spectrum from industrial assembly [259, 260, 279, 284] to daily cooking activities [68, 170, 295]. Despite the increased focus on this area, there is a notable lack of a unified methodology for mistake detection, resulting in fragmented literature and scarce evaluations.

Datasets. ATA [100] is a procedural dataset designed for offline mistake detection in assembling activities. It only reports video-level mistakes annotations, making it impractical for frame-based applications. Assembly101 [284] is a large-scale video dataset that annotates frame-level mistakes. The videos represent actors assembling toys, and the dataset offers synchronized Ego-Exo views and hand-positions data. Another recent assembling dataset with frame-level annotations is IndustReal [279]. However, the authors consider a single toy, which results in a single procedure to be learned. Epic-tent [143] is a dataset with a different domain, as it reports actors building up a tent in an outdoor scenario. The participants have different degrees of expertise, and they naturally commit mistakes that have been annotated in Epic-tent. Holoassist [323] is a recent dataset that presents egocentric videos of people performing several manipulating tasks instructed by an expert. In this study, we employ [143, 284] datasets since they give insights into errors happening during procedures in two different contexts, i.e., controlled industrial and outdoor environments[2].

Methods. In Table 9.1, we report the main features of the recent approaches to Mistake Detection in procedural videos. In [100], the authors train an action recognizer model and consider error detection a semantic way of evaluating the segmentation results. Their method is thus explicitly offline, while PREGO

[2]At the time of writing [279, 323] were unavailable publicly.

aims to promptly detect procedure mistakes as soon as they occur. By contrast, Assembly101 [284] and Holoassist [323] apply the same error detection baselines on varying granularity but also operate offline, requiring video segmentation. Ding et al. [74] use knowledge graphs for error identification, bypassing video analysis and extracting procedural steps from transcripts, presenting a distinct methodology within the procedural learning field. PREGO diverges from these works as it leverages the video frames to detect the steps of the procedure online and leverages symbolic reasoning for an online assessment of the procedure's correctness. Moreover, acknowledging that the mistake detection task shares many aspects with the established field of Video Anomaly Detection, we design PREGO to work in an OCC framework. As motivated in [91, 354], this choice ensures that PREGO is not constrained to detect only specific kinds of errors, as it is trained on sequences that do not contain mistakes.

8.2.2 Steps recognition and anticipation

Step recognition is the task of identifying actions within a procedure. Indeed, a procedure is an ordered sequence of steps that bring to the completion of a task. Step recognition is crucial in areas such as autonomous robotics and educational technology. Recent contributions in this domain include [285], which uses a novel loss for self-supervised learning and a clustering algorithm to identify key steps in unlabeled procedural videos. [203] introduces an action segmentation model using an attention-based structure with a Pairwise Ordering Consistency loss to learn the regular order of the steps in a procedure. They devise a weakly supervised approach, using only the set of actions occurring in the procedure as labels, avoiding frame-level annotations. [373] approaches the task by leveraging online instructional videos to learn actions and sequences without manual annotations, blending step recognition with a deep probabilistic model to cater to step order and timing variability. Notably, An et al. [10] proposed miniROAD explicitly targeting online action detection. They leverage an RNN architecture and regulate the importance of the losses during training to perform active action recognition.

On the other hand, step anticipation focuses on predicting forthcoming actions in a sequence crucial for real-time AI decision-making. [2] addresses this by generating multiple potential natural language outcomes, pretraining on a text corpus to overcome the challenge of diverse future realizations. Additionally, the framework of [257] proposes solutions to future activity anticipation in egocentric videos, using contrastive loss to highlight novel information and a dynamic reweighing mechanism to focus on informative past content, thereby enhancing video representation for accurate future activity prediction. Unlike prior works, PREGO is the first model that anticipates actions via LLM symbolic reasoning in the label space.

8.2.3 Large Language Modelling and Symbolic Reasoning

LLMs are trained on large datasets and have many parameters, giving them novel capabilities compared to previous language models [327]. LLMs have shown remarkable abilities in modeling many natural language-related [307] and unrelated tasks [32, 115, 327]. Their next-token prediction mechanism aligns with our action anticipation branch, where both systems aim to infer future actions based on collected data.

Recent research [85, 115, 189, 222, 239] has explored LLMs' ability to operate as *In-Context Learners* (ICLs), which means they can solve novel and unseen tasks. Given a query prompt with a context of input-output examples, LLMs can comprehend and address the problems in this setting without further fine-tuning. LLMs as ICLs have been used for a variety of tasks, including planning [239], programming [115, 189], logical solvers [85], and symbolic reasoning [222].

Some work has shown that LLMs can generate semantically significant patterns [222], while [329] has explored LLMs' in-context capabilities on semantically unrelated labels, where there is no relationship between a token and its meaning. Recent works [237, 239] studied the opportunity to employ LLMs for devising plans to accomplish tasks. In our mistake detection pipeline, we leverage ICL using an LLM as our action anticipation branch. Given examples of similar procedures, such LLM continues sequences of steps in a procedure, represented as symbols. The LLM acts as a symbolic pattern machine, continuing the pattern of actions given a context of sequences performed goal-oriented, even if the sequences do not follow a semantic scheme. This combines the challenges of predicting future actions and of having no semantics.

8.3 Methodology

PREGO exploits a dual-branch architecture that integrates procedural step recognition with anticipation modeling, as depicted in Fig. 9.1. In the following sections, we elaborate on the problem formalization (Sec. 9.3.1), present the branches for step-recognition (Sec. 8.3.2) and step-anticipation (Sec. 9.3.3), and finally we illustrate the mistake detection procedure (Sec. 9.3.4).

8.3.1 Problem Formalization

We consider a finite set of N procedures $\{p_i\}_{i=1}^{N}$ that encodes the sequence of actions as $p_i = \{a_k\}_{k=1}^{K_i}$ where K varies depending on the specific procedure i and $a_k \in \mathcal{A} = \{a | a \text{ is a possible action}\}$. Each procedure is also represented by a set of videos $\{v_i\}_{i=1}^{N}$ that are composed of frames $v_i = \{f_\tau\}_{\tau=1}^{M_i}$ where M_i is the total number of frames in the video i.
Fixed a frame f_τ from a given video v_i, PREGO's task is double-folded: it has to (1) recognize the action a_τ corresponding to the frame f_τ in the video and (2) predict the action a_τ that will take place at time τ considering only past observations until time $\tau - 1$.
The step recognition task is performed by a module ρ that takes as input the encoded frames of v_i up to τ and returns an action a_τ^ρ. We then feed the module ξ, responsible for the anticipation task, with all the $a_1^\rho, ..., a_{\tau-1}^\rho$ actions to have a prediction a_τ^ξ for the next action in the obtained sequence.
Finally, we compare a_τ^ρ with a_τ^ξ and we deem as mistaken the actions where a misalignment between the outputs of the two branches occurs. For clarity, in the remainder of this section, we consider a single procedure p associated with a video v.

8.3.2 Step Recognition

The step recognition module, denoted as ρ, receives encoded frames from v_i up to τ as input and generates the action a_τ^ρ. This module can be designed in a modular fashion under the condition that the model operates online, meaning it lacks knowledge of future events. In our approach, we leverage MiniRoad [10], renowned for its state-of-the-art performance in online action detection, its efficiency in computational complexity (measured in GFlops), and parameter count.

Within this framework, with w representing the size of window W, the model forecasts the action a_τ by considering frames $f_{\tau-w}, .., f_\tau$. However, this approach yields redundant outcomes as the model frequently predicts the same action for consecutive frames. We adopt a simple procedure to ensure consistency: we only consider unique actions whenever the model predicts the same action for consecutive frames. The loss

for this step recognition module is calculated through a Cross Entropy Loss, comparing the actual action a_τ with the predicted action a^ρ_τ.

8.3.3 Step Anticipation

We introduce a novel approach for step forecasting in procedural learning by harnessing the power of symbolic reasoning [222] via a Language Model (LM). Specifically, we employ a Large Language Model (LLM) as our ξ model for next-step prediction, feeding it with prompts from procedural video transcripts. These prompts are structured in two parts: the first part comprises contextual transcripts C, *Input Context* in Fig. 8.2, extracted from similar procedures to inform the LLM about typical step sequences and order. The second part, *Sequence* in the Figure, includes the current sequence of actions up to a specific frame, f_τ, detected by our module ρ, i.e.,

$$s_\tau = [a^\rho_1, ..., a^\rho_{\tau-1}] \tag{8.1}$$

This approach enables the LLM to utilize in-context learning, eliciting its ability to anticipate subsequent actions. Our framework operates in a zero-shot fashion, relying on the LLM's ability to retrieve the correct sequence continuation without specific training or fine-tuning but only leveraging the positive examples within the input prompts. Additionally, our method employs symbolic representations of the steps, converting the set of actions $\mathcal{A}$ into a symbolic alphabet Ω through an invertible mapping γ. Therefore, we can express the symbolic predicted sequence as:

$$\gamma(s_\tau) = [\gamma(a^\rho_1), ..., \gamma(a^\rho_{\tau-1})] = [\omega_1, ..., \omega_{\tau-1}] \tag{8.2}$$

This conversion abstracts the actions from their semantic content, allowing the LLM to focus on pure symbols and sequences, thus simplifying the complexity of predicting the following action.
Finally, the ξ module, given the examples C and the current symbolic transcript $\gamma(s_\tau)$ described in its prompt, is required to output the most probable symbol ω_τ to continue the sequence (see Figure 8.2). At this point, we apply the inverse function of γ to retrieve the underlying step label, i.e., $a^\xi_\tau = \gamma^{-1}(\omega_\tau)$.

8.3.4 Mistake Detection

We finally compare the outputs of the two modules to detect procedural mistakes. Precisely, we consider as correct all the steps where the outputs of the two modules align with each other, while we deem as an error the cases for which the two outputs diverge. That is:

$$\begin{cases} a^\rho_\tau \neq a^\xi_\tau & \text{MISTAKE} \\ a^\rho_\tau = a^\xi_\tau & \text{CORRECT} \end{cases} \tag{8.3}$$

8.4 Benchmarking online open-set procedural mistakes

This section presents the benchmark datasets and the evaluation metrics used in our experiments. First, we introduce the reviewed online variants of Assembly101 and Epic-tent (Sec. 8.4.1), and then we define the proposed online metrics in Sec. 8.4.2.

8.4.1 Datasets

We propose *Assembly101-O* and *Epic-tent-O* as a refactoring of the original datasets [143, 284], detailing the selected labeling for online benchmarking, and the novel arrangement of training and test splits, to account for open-set procedural mistakes.

Assembly101-O

Assembly101 [284] is a large-scale video dataset that enables the study of procedural video understanding. The dataset consists of 362 procedures of people performing assembly and disassembly tasks on 101 different types of toy vehicles. Each procedure is recorded from static (8) and egocentric (4) cameras and annotated with multiple levels of granularity, such as more than 100K coarse and 1M fine-grained action segments and 18M 3D hand poses. The dataset covers various challenges, including action anticipation and segmentation, mistake detection, and 3D pose-based action recognition.

Assembly101 for online and open-set mistake detection (*Proposed*) We introduce a novel split of the dataset [284] that enables online, open-set mistake detection by design. Assembly101-O mainly encompasses two edits on [284], namely, a new train/test split and a revision of the length of the procedures. The novel split encloses all the correct procedures in the train set, leaving the videos with mistakes for the test and validation set. This modification is needed to allow models to learn the sequences of steps that characterize correct procedures in a one-class classification fashion. In this way, models do not undergo the bias of learning specific kinds of mistakes during training; instead, as they are exposed exclusively to correct processes, they adhere to the OCC protocol and consider mistakes all actions that diverge from the learned normalcy. As a further advantage, this saves all mistaken annotated videos for the test set, granting better balanced correct/mistaken validation and test sets and a more comprehensive evaluation of mistake detection. The second revision involves evaluating each video for benchmarking until the procedure is compromised, meaning until a mistake occurs due to incorrect action dependencies. Indeed, coherently with the OCC protocol, models are tasked with learning the correct flows of steps that allow procedures to be efficiently completed and considering sub-process after a mistake occurs creates a gap between the actions in the train set and those in the test, which prevents the models from recognizing or correctly anticipating the procedure steps. Moreover, this work proposes to focus on egocentric videos to be consistent with real-world applications. Hence, we only leverage a single egocentric video from the four views available for each video in [284].

Epic-tent-O

Epic-tent is a dataset of egocentric videos that capture the assembly of a camping tent outdoors. The dataset was collected from 24 participants who wore two head-mounted cameras (GoPro and SMI eye tracker) while performing the task. The dataset contains 5.4 hours of video recordings and provides annotations for the action labels, the task errors, the self-rated uncertainty, and the gaze position of the participants. The dataset also reflects the variability and complexity of the task, as the participants interacted with non-rigid objects (such as the tent, the guylines, the instructions, and the tent bag) and exhibited different levels of proficiency and uncertainty in completing the task.

Epic-tent for online and open-set mistake detection (*Proposed*) This section introduces a novel split for the Epic-tent dataset [143], designed to be adapted for the open-set mistake detection task. It is labeled with nine distinct mistake types. However, among these, "*slow*", "*search*", "*misuse*", "*motor*", and "*failure*" do

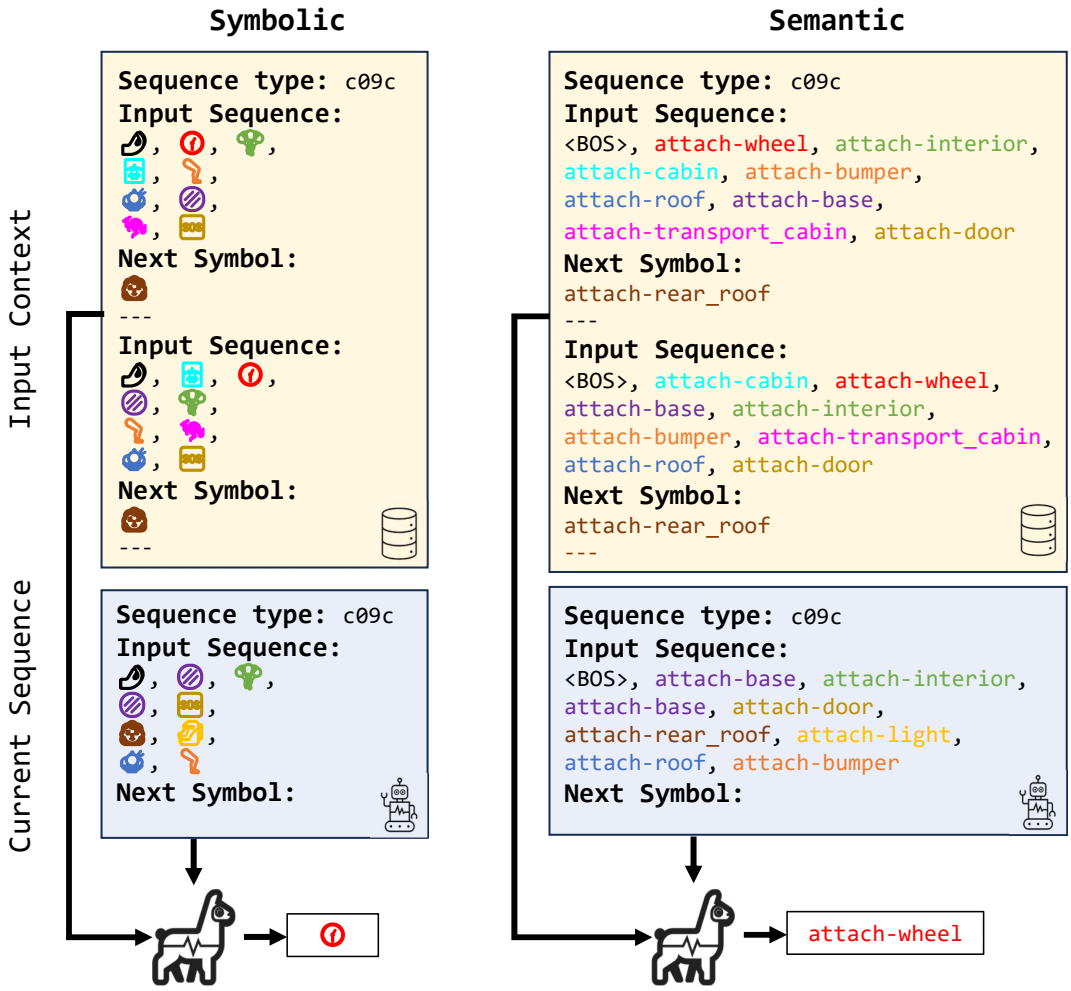


Figure 8.2: Two different representations of the actions in the prompt for the LLM model. On the left, the prompt is represented using symbolic labels. On the right, the prompt encompasses the names of the actions in the transcript. The context part of the prompt is fixed and retrieved from the dataset, while the recognition module extracts the current sequence.

not represent procedural errors, since, when they occur, the procedure is not tainted. On the other hand, the categories "*order*", "*omit*", "*correction*", and "*repeat*" are procedural mistakes, which we consider for our task. Epic-tent is designed for the supervised error detection task and, differently from [284], every reported procedure includes some mistakes, hampering the reproduction of the split procedure proposed for Assembly101-O. Nonetheless, this dataset provides the confidence scores assigned to each frame by the performer, indicating their self-assessed uncertainty during the task. Thus, we define a strategy for splitting , reported in Sec. C of the supplementary materials, in which videos featuring the most confident performers form the train set, while those showing higher uncertainty (and thus potentially more prone to errors) populate the test set. This partitioning strategy holds encouraging promise, especially in real-world scenarios where the accurate labeling of erroneous frames is hard to achieve or where the training of a mistake detector can initiate immediately post-recording without necessitating the completion of the entire annotation process. The resulting split comprises 14 videos for the training set and 15 for the test set.

The Epic-tent dataset showcases only egocentric videos recorded through Go-Pro cameras. This further highlights the practicality and relevance of the proposed novel benchmark in open-scene contexts. The videos in the test set are also trimmed up to the last frame of the first mistake occurring in the video, while

Table 8.2: A comparative assessment between PREGO and the chosen baseline methods is conducted to detect procedural mistakes using the Assembly101-O and Epic-tent-O datasets.

			Assembly101-O			Epic-tent-O		
	Step Recog.	**Step Antic.**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
One-step memory	*Oracle*		16.3	30.7	21.3	6.6	26.6	10.6
BERT [71]	*Oracle*		78.2	20.0	31.8	75.0	5.6	10.4
PREGO	*Oracle*	*GPT-3.5*	29.2	75.8	42.1	9.9	73.3	17.4
PREGO	*Oracle*	*LLAMA*	30.7	94.0	46.3	10.7	86.7	19.1
OadTR for MD [324]	*OadTR [324]*	*OadTR [324]*	24.3	18.1	20.7	6.7	21.7	10.2
PREGO	*OadTR [324]*	*LLAMA*	22.1	**94.2**	35.8	9.5	**93.3**	**17.2**
PREGO	*MiniRoad [10]*	*GPT-3.5*	16.2	87.5	27.3	4.3	66.6	8.0
PREGO	*MiniRoad [10]*	*LLAMA*	**27.8**	84.1	**41.8**	**8.6**	20.0	12.0

those representing correct procedures are maintained unaltered.

8.4.2 Metrics

To assess the performance of our procedural mistake detection model, we use True Positives as a measure of the model correctly identifying errors and True Negatives as a measure of accurately labeling steps that are not errors. Thus, we rely on the Precision, Recall, and F1 score metrics to evaluate the performance of our model. These metrics offer valuable insights into the model's capability to identify and classify mistakes within procedural sequences. More specifically, precision quantifies the accuracy when predicting mistakes, minimizing false positives. Recall assesses the model's capability to retrieve all mistakes, reducing the number of false negatives. Finally, the F1 score is the harmonic mean of precision and recall, and it balances failures due to missing mistakes and reporting false alarms.

8.5 Experiments

In this section, we present the results of our experiments on online and open-set mistake detection in procedural videos. We contrast PREGO with several baselines that employ different mistake detector techniques or use the ground truth as an oracle. The oracular scenario represents an upper bound for a given anticipation method since the recognition branch does entirely rely on the ground truth. All the baselines are assessed on the Assembly101-O and Epic-tent-O datasets, detailed in section 8.4.1. Evaluation metrics include precision, recall, and F1 score, as outlined in 8.4.2. Baselines are introduced in section 9.7.1, and the primary results are analyzed in 8.5.2. Furthermore, we explore the influence of different prompt types in 8.6.2 and the context in 8.5.4. Lastly, implementation specifics are discussed in 8.5.5, along with addressing certain limitations.

8.5.1 Baselines

To estimate the effectiveness of PREGO, we evaluate its performance by comparing it against the following baseline models based on the metrics presented in Sec. 8.4.2:

One-step memory We define a *transition matrix* considering only the correct procedures. Specifically, given the set of the actions $\mathcal{A}$ in the training set with $|\mathcal{A}| = C$, we define a transition matrix $M \in \mathbb{R}^{C \times C}$ which stores in position (l, m) the occurences that action m follows action l. We then label as *mistake* the actions occurring in the test split that do not correspond to transitions recorded in the training set.

OadTR for mistake detection The work [324] proposes a framework for online action detection called OadTR that employs a Vision Transformer to capture the temporal structure and context of the video clips.

The framework consists of an encoder-decoder architecture. The encoder takes the historical observations as input and outputs a task token representing the current action. The decoder takes the encoder output and the anticipated future clips as input and outputs a refined task token incorporating the future context. In the context of procedural error detection, a mistake is identified when the output from the encoder does not align with the one from the decoder.

BERT [71] We leverage the capability of BERT utilizing its specific [CLS] token to predict the correct or erroneous sequence of action. More specifically, we fine-tune BERT using the next-sentence-prediction task, where the model is trained to predict whether one sentence logically follows another within a given text. In our context, we apply this to determine whether step B can follow another step A within a procedure. Here, steps are defined as sets of two words, such as *attach wheel*, representing coarse actions. To perform this, BERT is presented with pairs of sentences corresponding to actions A and B, tasking it with predicting the sequential relationship between them. BERT's advantage lies in pre-training on a vast text corpus, followed by fine-tuning for our specific scenario. This process enables BERT to grasp contextual connections between sentences, rendering it effective for tasks like classifying procedures and comprehending the logical flow of information in text.

8.5.2 Results

We evaluate PREGO's performance on two datasets, Assembly101-O and Epic-tent-O, and detail the results in Table 9.6. We replaced the step recognition branch's predictions with ground truth action labels to assess the upper bound on performance without step detection bias defining the *Oracle* setting. This approach simulates a scenario where the video branch perfectly recognizes actions in the videos. The One-step memory method considers only the previous action, while BERT reasons at a higher level of abstraction and leverages past actions more effectively. This reduces false alarms but introduces a conservative bias in the form of missing mistakes. PREGO outperformed all baselines by leveraging symbolic reasoning for richer context modeling. PREGO$_{LLama}$ achieved the highest F1-score with a 45.6% improvement over BERT, demonstrating the effectiveness of symbolic reasoning. Among PREGO configurations, PREGO$_{LLAMA}$ performed 9% better than PREGO$_{GPT-3.5}$ on Assembly101-O, due to its more powerful symbolic representation. Similar trends are observed on Epic-tent-O with metric values influenced by dataset characteristics (Epic-tent-O allows for more diverse assembly procedures compared to Assembly101-O).

We move beyond oracle methods that rely on ground truth information and compare PREGO's performance against the established method OadTR [324] per-frame action detection and forecasting. PREGO$_{Llama}$, using the same method for step recognition, significantly outperforms OadTR for MD achieving a 102% improvement in F1-score (refer to Table 9.6 for detailed results). OadTR is restricted to processing fixed-size video segments with a default window of 64 frames, resulting in the smallest F1-score. Indeed, it is insufficient for capturing the context of long procedures lasting an average of 7 minutes in Assembly101. The improvement can also be attributed to PREGO's symbolic step anticipation branch. Symbolic reasoning allows PREGO to operate at a higher level of abstraction than video-based methods like OadTR. This advantage mitigates video-based approaches' challenges with occlusion and forecasting fine-grained actions.

PREGO$_{LLama}$ can better learn the normal patterns of the procedures and detect deviations from them, achieving the best results in terms of F1-score. In addition, PREGO$_{GPT-3.5}$ incurs costs that scale with the number of processed tokens, hindering its suitability for large-scale studies. LLAMA, being open-source, facilitates cost-effective exploration of PREGO at scale. Compared to their oracle counterparts, PREGO$_{GPT-3.5}$ and PREGO$_{LLama}$ could potentially gain 54% and 11% improvement in F1-score, respec-

tively. This suggests that the video branch's accuracy bottlenecks overall performance. However, the oracle recognition experiment also highlights the potential for improvement within PREGO itself. Other factors influencing performance include the quality of symbolic inputs, semantic prompts, and the underlying LLM architecture.

8.5.3 Performance of Different Prompt Types

We investigate the effect of different action representations in the prompt for the Step Anticipation task. Following [222], we consider three ways of representing an action: numerical, semantic, or random symbols. Numerical representation means that an action label is replaced with an index in the range $[0, \mathcal{A}]$, where $\mathcal{A}$ is the total number of actions. Semantic representation implies that the action is represented by its action label. Random symbol indicates that each action is assigned to a different symbol, such as a set of emojis. This allows us to examine how the LLM can manage different levels of abstraction and expressiveness of the input prompt. Fig. 8.2 illustrates an example of the same prompt in two representations, symbolic and semantic.

Table 8.5 shows the experiment results using the described representations. We observe that all the different representations achieve close performance, with the random representation achieving the highest F1 score, 41.8, followed by the semantic and numerical representations, with 41.4 and 39.9, respectively. We hypothesize that employing a numerical system to represent different actions might inadvertently introduce a form of bias related to ordering. This type of bias occurs because the relationship between specific actions and their corresponding numerical values is inherently arbitrary, lacking a natural or logical sequence. As a result, the numerical mapping can obscure the characteristics of the actions being represented, leading to potential challenges in accurately anticipating or predicting future actions based on these numerical representations. Remarkably, the semantic representation achieves a comparable performance even though words can introduce bias or ambiguity into the model. This indicates that PREGO can handle the natural language input and extract the relevant information for the step anticipation task. Surprisingly, the random symbol representation has the highest performance amongst the other representations, even though the model does not have any semantic or numerical association with them. This suggests that the model effectively learns the temporal structure of the actions from the input history, regardless of the symbol representation.

8.5.4 Performance of Different Prompt Context

We examine two alternative ways of writing a prompt (Table 8.4) for the PREGO method: prompting with a less representative context Vs. a more elaborate one. The less informative prompt, labeled as "Unreferenced-Context" in Table, requests PREGO to produce the next step without providing the model with the information that the contexts are sequences and that the output required is a symbol. The context is simply given as "Context", the current sequence is given as "Input", and the next step is requested as "Output". The more elaborate prompt, labeled "Elaborate" in Table, has a more complex prompt for both the context and the output. The context is given with the sentence "Given the sequences of the following type:", the sequence to be completed as "Complete the following sequence", and the output "Sequence is completed with". The three prompts are shown in Fig. 1 of the supplementary materials.

The results show that the referenced-context prompt achieves the best F1 score (41.8). The other two alternatives perform similarly, reaching an F1 score of 41.4 and 40.5. The detailed prompt structure is the most effective way of writing a prompt for the PREGO method, as it clearly conveys the essential

Table 8.3: Performance of PREGO with different prompt representations for Procedural Mistake Detection evaluated via F1 score, precision and recall on the Assembly101-O dataset.

	Precision	**Recall**	**F1 score**
Numerical	26.7	78.6	39.9
Semantic	**27.8**	81.3	41.4
Random	**27.8**	**84.1**	**41.8**

Table 8.4: Impact of prompt variations on PREGO - Unreferenced-Context, Elaborate, and Referenced-Context prompts. Evaluated via F1 score, precision and recall on the Assembly101-O dataset.

	Precision	**Recall**	**F1 score**
Elaborate	26.9	82.4	40.5
Unreferenced-Context	27.3	**85.2**	41.4
Referenced-Context (*PREGO*)	**27.8**	84.1	**41.8**

information and the objective of the task.

8.5.5 Implementation Details

PREGO is trained on two P6000 GPUs using the Adam optimizer, a batch size of 128, a learning rate of $1e^{-5}$, and a weight decay of $1e^{-4}$. For Assembly-101-O, we use the pre-extracted TSN frame level features from [284]. For Epic-Tent-O, we extract the features using the same method. The training process takes approximately 4 hours. PREGO achieves 0.02 fps on an NVIDIA Quadro P6000, meeting our needs without real-time constraints.

Limitations Across the currently available procedural datasets with annotated mistakes, the number of procedures only ranges up to hundreds, which is a limitation for current deep learning techniques. The original Assembly101 [284] dataset encompasses 330 procedures; our proposed Assembly101-O inherits only the procedures without mistakes as the learning set, namely 190 procedures; similarly, both Epic-tent [143] and Epic-tent-O only include 29 videos depicting the same task. We acknowledge the need for a large-scale dataset for online mistake detection and leave it as a future work. Indeed, more procedures will likely let the models generalize better, improving their capability to deal with multiple plausible procedures.

8.6 Ablation Study

Here, we present the results on step anticipation using two different Large Language Models (LLMs) (c.f. Sec. 8.6.1) and compare them with the case of oracular step recognition (c.f. Sec. 8.6.1). We also investigate the effect of different semantic prompts on the performance of the LLMs (c.f. Sec. 8.6.2).

Table 8.5: Performance of PREGO with different prompt representations for Procedural Mistake Detection evaluated via F1 score, precision and recall on the Assembly101-O dataset.

	Precision	**Recall**	**F1 score**
Numerical	26.7	78.6	39.9
Semantic	**27.8**	81.3	41.4
Random	**27.8**	**84.1**	**41.8**

8.6.1 Step Anticipation

Llama Vs GPT-3.5 In this section, we evaluate the performance of different LLMs for procedural mistake detection with symbolic inputs. We follow the Step Anticipation setting described in Sec. 9.3.3 and compare the LLAMA-2 model [307] with the OpenAI GPT-3.5 method. We report the results in Table 9.6, using F1 score, precision and recall as the evaluation metrics. The Table shows that the OpenAI GPT-3.5 method achieves the highest F1 score on the Assembly101-O dataset. We conclude that the OpenAI GPT-3.5 method is the most effective LLM for procedural mistake detection, as it can better learn the normal patterns of the procedures and detect deviations from them. We chose LLAMA as the LLM for the experiments on Epictent and the ablation studies due to its open source, accessible nature, and performance, which is almost on par with GPT-3.5.

Oracular Evaluation To evaluate the action anticipation model's performance, we replace the step recognition branch's predictions with the ground truth action labels. This experiment mimics a situation where the video branch can perfectly recognize the actions performed in the videos. We refer to the last two rows in Table 9.6, comparing them with PREGO and the other baseline methods. As expected, the oracle recognition experiment outperforms the PREGO model, achieving an F1 score of 46.3 and 44.1 with LLAMA and GPT-3.5, respectively, compared to 35.8 and 31.2 of PREGO, and indicating that the accuracy of the video branch is a bottleneck for the overall performance. However, the oracle recognition experiment also reveals our model's potential for improvement. Other factors influence the model's performance, such as the quality of the symbolic inputs, the semantic prompts, and the LLM architecture.

8.6.2 Performance of different prompt types

We investigate the effect of different action representations in the prompt for the Step Anticipation prediction. Following [222], we consider three ways of representing an action: numerical, semantic, or random symbols. Numerical representation means that an action label is replaced with an index in the range $[0, \mathcal{A}]$, where $\mathcal{A}$ is the total number of actions. Semantic representation implies that the action is represented by its action label. Random symbol indicates that each action is assigned to a different symbol, such as a set of emojis. This allows us to examine how the LLM can manage different levels of abstraction and expressiveness of the input prompt. Fig. 8.2 illustrates an example of a prompt in the three different representations.

Table 8.5 shows the experiment results using the described representations. We observe that all the different representations achieve close performance, with the numerical representation achieving the highest F1 score, with 33.4, followed by the random representation, 33.2. We hypothesize that the numerical representation is easier for the model to understand and predict the next step, as it reduces the number of tokens

to be generated. Indeed, an action index requires only four tokens, whereas the emojis and the action labels require at least eight tokens. Remarkably, the semantic representation achieves a comparable performance even though words can introduce bias or ambiguity into the model. This indicates that the model can handle the natural language input and extract the relevant information for the step anticipation task. Surprisingly, the random symbol representation has a comparable performance to the other representations, even though the model does not have any semantic or numerical association with them. This suggests that the model can learn the temporal structure of the actions from the input history, regardless of the symbol representation.

8.7 Conclusion

We have introduced PREGO, a one-class, online approach for mistake detection in procedural egocentric video. PREGO predicts mistakes by comparing the current action predicted by an online step recognition model with the next action, anticipated through symbolic reasoning performed via LLMs. To evaluate PREGO, we adapt two datasets of procedural egocentric videos for the proposed task, thus defining the Assembly101-O and Epic-tent-O datasets. Comparisons against different baselines show the feasibility of the proposed approach to one-class online mistake detection. We hope that our investigation and the proposed benchmark and model will support future research in this field.

Chapter 9

Leveraging LLMs with Chain of Thought and In-Context Learning for Mistake Detection in Procedural Egocentric Videos.

9.1 Introduction

Detecting procedural errors from videos has recently gained increasing interest due to its potential to yield substantial benefits across several fields. The capability to detect mistakes in videos of individuals performing tasks like recipe execution, object assembly or handling complex workflows is promising, as it greatly enhances the training and learning experience by providing real-time feedback. This immediate feedback has the potential to allow for timely corrections, enabling faster skill development and acquisition and a safer learning environment in hazardous sectors such as surgery or aviation.

Advanced mistake detection models will become a fundamental asset to ensure precision, safety and effectiveness in several procedural applications. These characteristics bolster the generation of new datasets [22, 74, 80, 100, 143, 221, 260, 279, 284, 301, 323, 376] and methodologies, aimed at pushing forward the evolution of procedure learning [100, 143, 203, 301, 324, 373] and error detection models [74, 279, 323]. Currently, existing approaches to mistake detection vary widely. Some methods focus on action detection, aiming to identify specific types of errors, such as missing steps or incorrect step orders [233]. These approaches emphasize tracking the sequence of actions and evaluating whether each action adheres to the correct procedure flow. In contrast, other models avoid action detection entirely and instead monitor modifications made to the assembled object to verify procedural completeness [39, 50].

An ideal Procedural Mistake Detection (PMD) model should possess two key properties: robustness to diverse types of mistakes and the ability to provide online feedback in a timely manner. First, a mistake detection system should be capable of identifying any deviation from the correct workflow, regardless of the nature of the error. While there is a finite set of correct ways to perform a procedure, the number of possible mistakes is virtually limitless. To achieve this comprehensive capability, we use the One-Class Classification (OCC) framework which involves training the model exclusively on videos of correct executions and then testing it on a mixed set of correct and incorrect procedures — similar to anomaly detection methods [91, 354]. Second, the model should have online capabilities to deliver feedback quickly enough

to help users correct mistakes promptly, minimizing the reinforcement of incorrect actions [1]. Additionally, the model should be suitable for real-world uses. By adopting an egocentric perspective, we offer a learning experience that mirrors how humans naturally perceive and interact with tasks, enhancing usability and deployment feasibility. This perspective aligns well with real-world scenarios, making the model more suitable for practical applications during task execution.

PREGO [92] introduces a dual-branch system designed to detect procedural mistakes by identifying mismatches between the outputs of two distinct modules. The first branch, a step recognition module, processes the input video and identifies the current action being performed. The second branch, an step anticipation module, predicts the following action based on the sequence of previous actions through a language model. PREGO identifies a mistake by flagging frames where the recognized action diverges from the anticipated action, highlighting inconsistencies between the predicted and expected outcomes.

This work builds on the contributions of PREGO [92], expanding its scope and refining key components to enhance procedural mistake detection. While PREGO's evaluation of Large Language Models (LLMs) focused on a single model and a fixed approach, laying the groundwork for future developments, we conduct a comprehensive evaluation of LLMs as step anticipators to address these limitations, achieving state-of-the-art results on Assembly101-O and Epic-tent-O [92]. Specifically, we explore a broader range of LLMs, fine-tune them using Low-Rank Adaptation (LLoRA), and experiment with various prompting strategies, including Zero-Shot, Few-Shot, and Automatic Chain of Thought (Auto-CoT) [368], with the latter notably enhancing the predictive capabilities of the anticipation branch.

Furthermore, the recognizer's frame-by-frame predictions are prone to noise which disrupts the continuity of action sequences. To address this, we improve PREGO's original approach by moving beyond fixed window aggregation and investigating alternative strategies to mitigate noise and improve prediction accuracy. Additionally, we introduce frame-level metrics to provide a more precise evaluation, aligning the action recognition module's output with the anticipation module's input. Given the online nature of the task, we emphasize the importance of accurate and timely per-frame evaluations in dynamic scenarios to ensure effective mistake detection and minimize delays.

Our contributions include: (1) the exploration and benchmarking of multiple LLMs to improve upon the foundations laid by PREGO's approach, (2) finetuning the LLM anticipator with LLoRA for better adaptation to procedural tasks, (3) experimenting with various prompting methods, including Auto-CoT which improves the overall performances, (4) evaluating frame-level metrics for more accurate system performance compared to the original fixed window approach, and (5) investigating the impact of different aggregation techniques on the robustness and cohesion of mistake detection systems. Through extensive experimentation, we demonstrate both the challenges and opportunities of dual-branch architectures and LLM-based step anticipation for open-set mistake detection. Our results highlight the potential of integrating step recognition and anticipation within a unified framework for online procedural error detection, providing a more adaptable and effective solution for real-world applications.

[1] We differentiate this online setup from a real-time setup, which aims for immediate responses by allowing slight delays in processing while still providing effective feedback.

Table 9.1: Comparison of relevant models in procedural mistake detection. In the modalities column, *RGB* refers to RGB images, *H* stands for hand poses, *E* represents eye gaze, and *K* indicates keystep labels. Our approach is the first to adopt an egocentric, one-class and online method for detecting procedural mistakes.

Model	Egocentric (Ego)	One-Class (OCC)	Online	Modalities	Task	Datasets
Ding et al. [74] - *ArXiv '23*				*K*	Mistake Detection	Assembly101 [284]
Wang et al. [323] - *ICCV '23*	✓			*RGB+H+E*	Mistake Detection	HoloAssist [323]
Ghoddoosian et al. [100] - *ICCV '23*				*RGB*	Unknown Sequence Detection	ATA [100], CSV [258]
Schoonbeek et al. [279] - *WACV '24*	✓		✓	Multi	Procedure Step Recognition	IndustReal [279]
Lee et al. [176] - *CVPR '24*	✓	✓		RGB	Mistake Detection	EgoPER [176]
PREGO [92]	✓	✓	✓	*RGB*	Mistake Detection	*Assembly101-O*, *Epic-tent-O*
Proposed	✓	✓	✓	*RGB*	Mistake Detection	*Assembly101-O*, *Epic-tent-O*

9.2 Related Works

9.2.1 Procedural Mistake Detection

Procedural learning has made notable progress with the development of diverse datasets [80, 221, 260, 301, 376], offering valuable insights into both structured [22, 259, 323] and unstructured [68, 143] tasks. These datasets span a wide range of applications, from industrial assembly [259, 260, 279, 284] to everyday cooking activities [68, 170, 295]. Despite the growing interest in this field, the lack of a standardized method for mistake detection has resulted in limited evaluation and fragmented literature.

9.2.2 Dataset

Assembly101 [284] is a large-scale video dataset that provides frame-level mistake annotations. It features videos of actors assembling toys, with synchronized Ego-Exo views and annotated hand positions. IndustReal [279] focuses on a single toy, which leads to a single procedure being learned. Epic-tent [143] is a dataset that covers a different domain of unscripted actions that capture actors assembling a tent outdoors. The participants exhibit varying levels of expertise and naturally make mistakes, which have been annotated. ATA [100] is a procedural dataset created for offline mistake detection in assembling activities. However, it provides only video-level mistake annotations, limiting its usefulness for frame-based applications. HoloAssist [323] provides egocentric videos of individuals performing multiple manipulation tasks following expert instructions. CaptainCook4D [247] is tailored for evaluating procedural activities, specifically focusing on the culinary domain. While comprehensive, [247] incorporates both procedural errors and attentional lapses. For our purposes, the latter category falls outside the scope of our investigation, yielding such dataset unsuitable. Notably, [176] introduced a novel egocentric procedural error dataset consisting of videos depicting various errors within the cooking domain. In this study, we propose a new benchmark building upon PREGO tilize the datasets from [143, 284] as they provide insights into procedural errors in two distinct settings: controlled industrial and outdoor environments.

9.2.3 Methods

Table 9.1 presents an overview of the core characteristics of recent methods for Mistake Detection in procedural videos. Ding et al. [74] propose a method for procedural learning that utilizes knowledge graphs to identify errors. Their approach is based on textual transcripts, ignoring the use of visual cues coming from videos.

In [100], the authors train an action recognition model and treat error detection as a semantic evaluation of the segmentation results. On the other hand, Assembly101 [284] and HoloAssist [323] apply the same

error detection baselines on varying granularity but also run offline, requiring video segmentation and labels during training, thereby limiting their ability to detect previously unseen errors. In [279], the authors address the problem of Procedure Step Recognition, which focuses on recognizing the correct completion of steps rather than the partial execution of activities. EgoPER [176] proposes a framework composed of an action segmentation module and a constrastive module for detecting errors unseen during training. PREGO [92] leverages video frames to detect procedural steps and incorporates symbolic reasoning for evaluating the procedure's correctness. Its objective is to identify mistakes online as soon as they occur, ensuring prompt detection. Additionally, it is designed to function within an OCC framework, which allows it to detect a wide range of errors since it is trained on sequences without mistakes, without any constraint on specific errors [91, 354].

9.2.4 Step recognition

Step recognition, the process of identifying discrete actions within a structured procedural sequence, is fundamental to progress in domains such as autonomous robotics and educational technology.

Zhuang *et al.* [203] introduce an action segmentation model that employs an attention-based architecture coupled with a Pairwise Ordering Consistency (POC) loss function. This approach effectively captures the correct sequence of procedural steps. Their innovation lies in developing a weakly-supervised method that requires only the set of actions in a procedure as input, eliminating the need for labor-intensive frame-level annotations. In Shah *et al.*[285], a novel loss for self-supervised learning is combined with a clustering algorithm to detect key steps in procedural videos without using any labels. [373] tackles the task by utilizing online instructional videos to learn actions and sequences without manual annotations, combining step recognition with a deep probabilistic model to account for step order and timing variability. An et al. [10] introduced MiniROAD, the state-of-the-art online step recognizer, which utilizes an RNN architecture and adjusts loss importance during training to perform online action recognition. However, existing methods like MiniROAD perform detection on a frame-by-frame basis, which can be limiting in practical scenarios that require recognizing sequences of actions rather than isolated frames. This limitation is particularly apparent in real-world situations where understanding the broader context of sequential actions is crucial. To overcome this, we introduce aggregation techniques to improve the coherence of action recognition, addressing the frame-level limitation inherent in current methods.

Moreover, the approach proposed in Shen et al. [287] also highlights the importance of understanding actions in a more holistic manner rather than treating each frame in isolation. Yet, while PREGO focuses on detecting procedural mistakes using an OCC-based approach with LLMs for action anticipation, [287] targets action segmentation and progress estimation with task graphs.

9.2.5 Step Anticipation

Conversely, step anticipation predicts upcoming actions in a sequence, which is important for real-time AI decision-making. [2] generates multiple possible natural language outcomes, leveraging pretraining on a text corpus to handle the variability in future actions. [218] employs a two-level hierarchical approach, integrating both high-level human intentions and low-level action sequences, improving action anticipation in the long term. The pipeline consists of two models: the first extracts intentions and classifies actions, and the second generates future actions, conditioning human intentions to narrow down the uncertainty set of future actions.

Furthermore, the framework presented in [257] addresses future activity anticipation in egocentric videos by employing a contrastive loss to emphasize novel information and a dynamic reweighting mechanism that prioritizes informative past content. This approach improves video representation, leading to more accurate predictions of future activities. While we adopt a similar dual-branch architecture as in PREGO [92]. However, we omit symbolic reasoning, instead leveraging the actual semantic labels to harness the reasoning capabilities of LLMs through an Automatic Chain of Thought.

9.2.6 Reasoning tasks with Large Language Models

Large Language Models (LLMs), trained on vast datasets and equipped with numerous parameters, exhibit advanced capabilities beyond those of earlier language models [327]. They have demonstrated exceptional performance in both natural language processing tasks [307] and non-language tasks [32, 115, 327]. The next-token prediction mechanism of LLMs closely parallels our action anticipation framework, as both aim to predict future outcomes based on accumulated data.

Recent research [5, 85, 115, 149, 189, 222, 239] has investigated the ability of LLMs to function as *In-Context Learners* (ICLs), meaning they can solve novel and unseen tasks without requiring additional fine-tuning. When provided with a query prompt that includes a context of input-output examples, LLMs can understand the problem and generate appropriate solutions within this framework. LLMs as ICLs have been applied to a wide range of tasks, including planning [5, 239], programming [115, 189], logical problem-solving [85], and symbolic reasoning [222]. In addition to ICL, other paradigms, such as Chain-of-Thought (CoT) [328] and Automatic Chain-of-Thought (ACoT) [368], have been employed to enable LLMs to reason through their responses step-by-step, improving their ability to plan and articulate their thought process explicitly.

Previous research has shown that LLMs can generate semantically meaningful patterns [222, 369], while other work [329] has explored their in-context learning abilities with semantically unrelated labels, where there is no direct relationship between a token and its meaning. Recent studies [149, 237, 239] have investigated using LLMs for creating task-oriented plans

The approach in [149] is the most similar to ours, as it employs a Socratic method [355] that uses video features to predict subsequent actions with an LLM. Socratic Models (SMs) are a framework that utilizes structured dialogue between pre-existing foundation models, each leveraging its unique capabilities based on its training data distribution. However, while PALM integrates an Action Recognition Model (ARM) and a Vision-Language Model (VLM) to generate text prompts for long-term action anticipation, we diverge by using a dual-branch architecture: one branch dedicated to video recognition for action detection and the other utilizing an LLM for action anticipation.

In our mistake detection pipeline, we integrate In-Context Learning (ICL) and Automatic Chain of Thought (ACoT), utilizing an LLM as part of our action anticipation branch. ACoT enables the model to automatically generate intermediate reasoning steps, breaking the problem-solving process into smaller, logical steps. The approach helps the LLM to externalize its reasoning, enhancing its effectiveness in managing complex, sequential tasks.

9.3 Methodology

The proposed system leverages a dual-branch framework that integrates the recognition of procedural steps with anticipation modeling, as shown in Fig. 9.1. In the following sections, we elaborate on the problem

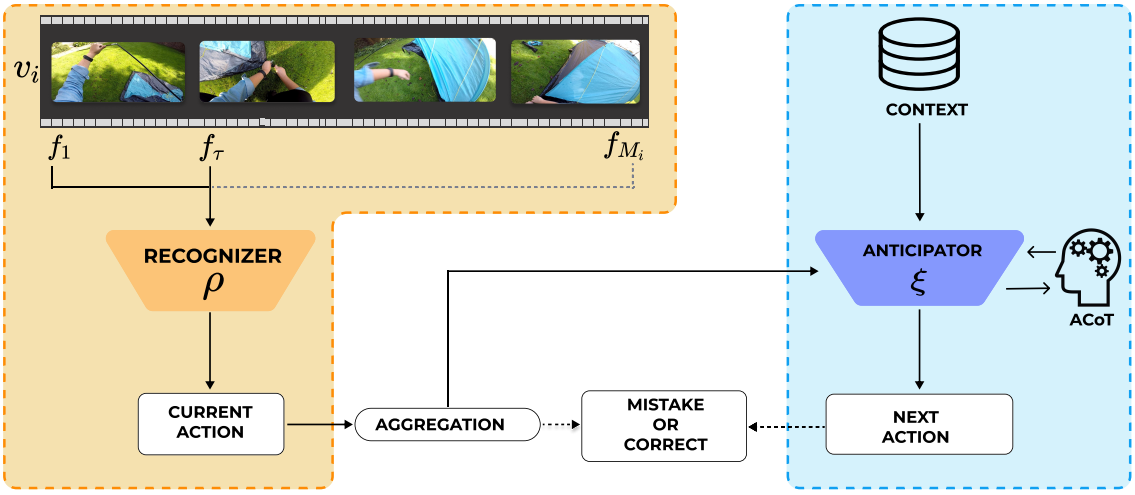


Figure 9.1: Our proposed model is based on two main components: The recognition module (orange) processes the input video in an online fashion and predicts actions observed at each timestep; the anticipation module (blue) reasons symbolically via a Large Language Model, utilizing automatic Chain of Thought (ACoT) reasoning to predict the future action based on past action history and a brief context, such as instances of other action sequences. Mistakes are identified when the current action detected by the step recognition method differs from the one forecasted by the step anticipation module.

formalization (Sec. 9.3.1), present the branches for step-recognition (Sec. 9.3.2) and step-anticipation (Sec. 9.3.3) and finally we illustrate the mistake detection procedure (Sec. 9.3.4).

9.3.1 Background

Let us consider a finite collection of N procedures $\{p_i\}_{i=1}^{N}$, where each procedure encodes a sequence of actions as $p_i = \{a_k\}_{k=1}^{K_i}$. Here, K_i varies depending on the specific procedure, and each action a_k belongs to the set $a_k \in \mathcal{A} = \{a | a \text{ is a possible action}\}$. Additionally, each procedure is represented by a set of videos $\{v_i\}_{i=1}^{N}$, which consist of frames $v_i = \{f_\tau\}_{\tau=1}^{M_i}$, with M_i indicating the total number of frames in video i. Given a specific frame f_τ from video v_i, our task is two-fold: we must (1) identify the action a_τ corresponding to the frame f_τ and (2) predict the action a_τ that will occur at time τ, relying solely on past observations up to time $\tau - 1$.

The step recognition task is handled by a module ρ, which takes the encoded frames of v_i from the start up to frame τ as input and outputs an action a_τ^ρ (Sec. 9.3.2). Subsequently, we input all the actions $a_1^\rho, ..., a_{\tau-1}^\rho$ into module ξ, which is responsible for the anticipation task, allowing it to predict the next action in the sequence as a_τ^ξ (Sec. 9.3.3).

Finally, we compare a_τ^ρ with a_τ^ξ designating actions as mistaken when there is a misalignment between the outputs from the two branches (Sec. 9.3.4). For clarity, the rest of this section will concentrate on a single procedure p associated with a video v.

9.3.2 Step Recognition

The step recognition module referred to as ρ, takes encoded frames from v_i up to frame τ as input and produces the action a_τ^ρ. In our approach, we utilize MiniRoad [10], which is well-known for its exceptional performance in online action detection, as well as its computational efficiency in terms of GFlops and parameter count.

In this setup, with w indicating the size of the window W, the model makes predictions for the action a_τ by examining the frames $f_{\tau-w}, .., f_\tau$. This approach often results in redundant predictions, with the model frequently assigning the same action to consecutive frames. To address this issue, we explore methods for aggregating redundant and similar actions (Sec. 9.6).

The loss for this step recognition module is calculated through a Cross Entropy Loss, comparing the actual action a_τ with the predicted action a^ρ_τ.

9.3.3 Step Anticipation

We employ an LLM as our ξ model for next-step prediction, feeding it with prompts from procedural video transcripts and exploiting its reasoning ability thanks to *Automatic Chain of Thought (ACoT)* [368]. Our framework operates without specific training or fine-tuning, leveraging only the reasoning from ACoT (See Fig. 9.1) and few-shot prompts. These prompts consist of two components: the first one, contextual transcripts C, is derived from similar procedures (i.e., different procedures for the same toy) to provide the LLM with information about typical step sequences and their order. The second part, *Sequence* s_τ, includes the current sequence of actions up to a specific frame, f_τ, detected by our module ρ, i.e., the sequence s_τ up to frame f_τ is defined as $s_\tau = [a^\rho_1, ..., a^\rho_{\tau-1}]$. Where a^ρ_i represents the action detected by module ρ at frame i.

The prompting occurs in two stages. First, we employ an ACoT mechanism of ξ as an intermediate step ϕ_τ. This ACoT process uses C, s_τ, and an additional prompt to explicitly stimulate reasoning. Then, we use the output of ϕ_τ along with C and s_τ to predict the next most probable action. By breaking down the task into smaller logical steps, the LLM can leverage the contextual transcript C and the current action sequence s_τ to generate intermediate reasoning steps, bridging the gap between observed and future actions.

9.3.4 Mistake Detection

Ultimately, we analyze the outputs of the two modules to identify procedural errors. Specifically, we classify steps as correct when the outputs of both modules match, while we label cases as errors when the outputs differ. That is:

$$\begin{cases} a^\rho_\tau \neq a^\xi_\tau & \text{MISTAKE} \\ a^\rho_\tau = a^\xi_\tau & \text{CORRECT} \end{cases} \tag{9.1}$$

We compare two different evaluation settings: aggregated and frame-level. In both cases, the input data to the recognizer is aggregated; this is done by compressing multiple window frames' actions in a single one by considering the most frequent action and then merging adjacent actions. While in the first case, we directly compare the anticipator's output to that aggregated input, in the frame-level case, we have to expand it to the original dimensionality.

9.4 Experimental Setup

9.4.1 Dataset

In our experiments, we relied on Assembly101-O and Epic-tent-O, novel splits of the original Assembly101 [284] and Epic-tent [143] datasets proposed in PREGO [92].

Assembly101-O includes two major modifications: a revised train/test split and an adjustment to the length of procedures. In the novel split, all correct procedures form the training set (adhering to the OCC protocol), while videos containing mistakes are reserved for the test set.

Additionally, this approach allocates all annotated mistake videos to the test set, resulting in a more balanced evaluation with both correct and mistaken procedures for testing, leading to a more thorough and realistic assessment of mistake detection.

The second modification involves benchmarking each video only until the procedure is compromised by a mistake caused by incorrect action dependencies. This ensures that the knowledge the model learns from the correct procedures during training can be fully exploited since evaluating sub-processes after a mistake would create a disconnection between the actions in the training and test sets, which could prevent models from accurately recognizing or anticipating procedure steps. Furthermore, the authors focus exclusively on egocentric videos to align with real-world applications, leveraging only one egocentric view of the four available in the original dataset [284].

Epic-tent is labeled with nine distinct types of mistakes, but not all are relevant to procedural errors. For example, categories such as "*slow*", "*search*", "*misuse*", "*motor*", and "*failure*" do not represent procedural errors since they do not compromise the procedure itself. Conversely, the categories "*order*", "*omit*", "*correction*", and "*repeat*" do represent procedural mistakes, and these are considered in our task. Unlike [284], which allows for a clean split between correct and mistaken procedures, every reported procedure in Epic-tent includes some mistakes. Despite this, the Epic-tent dataset provides performer confidence scores for each frame, representing their self-assessed uncertainty during the task. PREGO [92] devises a strategy to form Epic-tent-O: videos featuring the most confident performers from the training set, while those with higher uncertainty—likely to contain more errors—are allocated into the test set. This methodology offers considerable potential for real-world applications, particularly in contexts where precise frame-level error labeling is challenging. A key advantage is that mistake detection systems can initiate training immediately post-recording without the delay typically associated with annotation processes. Epic-tent-O was partitioned into 14 training videos and 15 testing videos. The Epic-tent dataset consists exclusively of egocentric videos captured through Go-Pro cameras, underscoring the practicality and relevance of this benchmark in open-scene environments. Additionally, the videos in the test set are trimmed to the last frame before the first detected mistake, while videos depicting correct procedures remain unaltered.

9.4.2 Metrics Definition

We work with a sequence of frames, each representing a procedure, where all actions are correct except for the last one, which is a mistake. A mismatch between the prediction of the step recognition and the ones of the step anticipation indicates the presence of a mistake, while no mismatch means the procedure is correct. To assess the performance of our procedural mistake detection model, we follow the evaluation proposed in PREGO: we use True Positives as a measure of the model correctly identifying errors and True Negatives as a measure of accurately labeling steps that are not errors. Thus, we rely on the Precision, Recall, and F1 score metrics to evaluate the performance of our model. These metrics offer valuable insights into the model's capability to identify and classify mistakes within procedural sequences. Precision quantifies the accuracy when predicting mistakes, minimizing false positives. Recall assesses the model's capability to retrieve all mistakes, reducing the number of false negatives. Finally, the F1 score is the harmonic mean of precision and recall, and it balances failures due to missing mistakes and reporting false alarms.

To establish a method-agnostic evaluation metric, we focus solely on the model's ability to classify ac-

tions as correct or erroneous. While our approach incorporates an online action detection component, we do not directly assess its general step recognition capabilities—instead, our evaluation centers on the accuracy of distinguishing between correctly and incorrectly performed actions. This decision stems from the limitations of metrics that blend recognizer and anticipator module performances, which can lead to unfair comparisons with approaches employing alternative prediction strategies. Given the expected class imbalance in ego-centric mistake detection tasks and the critical importance of identifying errors, the mistakes are considered as mistakes as the positive class in our evaluation framework.

When evaluating frame-level metrics (see Sec. 9.7.3), we consider corner cases such as the best, worst, and random scenarios (see Sec. 9.7.1) for clarity. To ensure a fair comparison, we use balanced accuracy as the evaluation metric, the average of true positive and true negative rates. This is useful to mitigate the impact of dataset imbalance, which would otherwise lead to an unfair comparison.

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right). \tag{9.2}$$

See Tab. 9.7 for experiments' results.

9.5 Large Language Models as Step Anticipators

This section comprehensively analyzes how an action anticipator operates using various techniques and methodologies. We begin by evaluating the performance of several large language models (LLMs) using in-context learning with few-shot examples (Sec. 9.5.1), which allows us to determine which model performs best in this setting. Once the most effective LLM is identified, we investigate different prompting strategies, starting from a system prompt's impact, including zero-shot, few-shot, and Automatic-Chain-of-Though prompting (Sec 9.5.2). In advance, we explore the potential improvements by fine-tuning each prompting method to enhance the model's performance (Sec. 9.5.3). Lastly, we perform a speed analysis, comparing different prompting methods (Sec. 9.5.4).

All experiments use ground truth aggregated labels of the Assembly101-O dataset as the output of the predictor's branches. This approach ensures that our performance analysis remains objective and untainted by errors from the step recognition branch, allowing for a more accurate evaluation of the step anticipator's capabilities.

9.5.1 Model Selection

In this analysis, we evaluate the performance of various LLMs, as shown in Table 9.2 when using few-shot context. The results demonstrate a clear performance hierarchy among the tested models, with LLAMA 3.1 8B emerging as the superior performer across all metrics. LLAMA 3.1 8B achieved the highest accuracy at 47.40% and the best F1 score of 48.40%, significantly outperforming its counterparts. The performance trend correlates with model size and recency, suggesting that larger, more recently developed models benefit from architectural improvements and more extensive pre-training. This is exemplified by the LLAMA 3.1 8B model, which leverages its 8 billion parameters to capture more complex patterns and better generalize the assembly task. The superior performance of LLAMA 3.1 8B could be attributed to several factors, including potential advancements in its training methodology, the quality and diversity of its training data, and possible optimizations for general task comprehension. An exciting pattern observed across all models is the substantial disparity between Precision and Recall scores. The consistently high Recall scores, peaking

at 91.30% for LLAMA 3.1 8B, indicate that these models identify relevant assembly steps. However, the markedly lower Precision scores suggest a tendency towards overprediction, resulting in false positives. This discrepancy highlights the inherent complexity of the assembly prediction task, where models must balance comprehensive step identification with selective precision.

Table 9.2: Results of different LLMs on Assembly101-O.

Model	Accuracy	Precision	Recall	F1
Mistral 7B	35.20	28.60	88.90	45.60
Gemma 9B	31.90	27.20	85.60	43.50
Phi 3 medium 4k instruct	39.60	29.50	88.70	46.20
LLAMA 2 7B	36.3	30.7	**94.0**	46.3
LLAMA 3 7B	36.40	29.80	87.30	45.60
LLAMA 3.1 8B	**47.40**	**33.80**	91.30	**48.40**

9.5.2 Prompt Analysis

System prompt. The results in Table 9.3 highlight the positive impact of combining Automatic-Chain-of-Though (ACoT) reasoning with Few-Shot (FS) examples on the performance of LLAMA 3.1 8B across different modalities in the Assembly101-O dataset. In all prompting scenarios, the system prompt is written as:

> *System:* I am going to provide an input sequence that represents a sequence of actions. Your task is to predict the next action of the last sequence based on the patterns observed in the provided input. Limit yourself to only answer with the predicted sequence, and follow the same format given as input.

This prompt consistently offers essential context about the task, improving the model's performance, particularly in accuracy and precision.

In Zero-Shot (ZS) settings, the instruction prompt is presented as:

> *System:* Below is an instruction that describes the task, paired with an input. Write a response that appropriately completes the request.

In Few-Shot prompting, this is enhanced by adding examples of input sequences and corresponding responses to guide the model.

We explore two different input representations: a textual and a numerical one: in the first, we represent an action by its name and natural language, e.g., `attach-wheel`; in the second one, we use the index associated with the action triplet. For instance, in the textual modality, accuracy improves by 6.8%, and precision increases from 31.80% to 33.80%, reflecting a 6.3% gain. Similarly, in the numerical modality, including the system prompt boosts accuracy by 8.2% (from 40.10% to 43.40%) and precision by 4.6% (from 30.40% to 31.80%).

The improvements can be attributed to the system prompt and FS examples, which provide explicit context and structure, enabling the model to better understand the task's nuances and generate more accurate and relevant predictions.

Table 9.3: Results of LLAMA 3.1 8B on Assembly101-O adding a system prompt.

System Prompt	Modality	Accuracy	Precision	Recall	F1
✓	Textual	**47.40**	**33.80**	91.30	**48.40**
	Numerical	43.40	31.80	92.30	46.20
X	Textual	44.40	31.80	87.30	46.10
	Numerical	40.10	30.40	**93.30**	46.60

Prompting Methods. When comparing ZS, FS, and ACoT prompting methods, ACoT emerges as the clear winner (Table 9.4), particularly when using the pre-trained model (*Base*), across both textual and numerical tasks. In the ACoT approach, automatic ACoT reasoning is employed, where the LLM is queried twice: first, to generate an internal reasoning step for the answer, and then to use this reasoning, combined with a few examples (as in FS), to derive the final answer. This two-step process allows the model to not only provide an explanation for its decision but also leverage the reasoning and examples to produce a more accurate and robust output. This superiority can be quantified: for example, in the textual modality, the ACoT base achieves a 51.40% accuracy, which is approximately 81% higher than the ZS base (28.40%) and 8.4% higher than the FS base (47.40%). In terms of the F1 score, the ACoT base (50.40%) outperforms the ZS base (42.40%) by 18.9% and the FS base (48.40%) by about 4.1%. This difference is less pronounced in numerical tasks, where the ACoT base achieves an accuracy of 40.40%, 47.4% higher than the ZS base (27.40%), and 8% higher than the FS base (37.40%).

These differences can be attributed to how these methods prompt the model. ZS and FS methods direct the model to generate answers straight from the input without guiding it through intermediate reasoning. While this may suffice for more straightforward tasks, it often falls short when dealing with more complex tasks that require deeper understanding.

ACoT, by contrast, excels in these scenarios because it prompts the model to break down the task into a series of logical, intermediate steps before concluding. This approach mirrors the "self-explanation" cognitive strategy, where breaking down and explaining each step improves problem-solving abilities. This explains why ACoT significantly outperforms ZS and FS, especially in tasks that demand nuanced understanding and intermediate reasoning.

9.5.3 Finetuning Analysis

We analyze the performance of LLAMA 3.1 8B when fine-tuned on Assembly101-O (Table 9.4). Contrary to the common expectation that fine-tuning enhances performance by adapting the model to the task's specific characteristics and distribution, this is not consistently observed. In both the ZS and FS methods, fine-tuning offers no significant improvement across metrics, with some cases showing similar or even lower precision and recall values. Notably, in the ACoT method, fine-tuning consistently results in a decrease in performance, particularly in terms of precision, suggesting that fine-tuning may not always provide the anticipated benefits.

This phenomenon can be explained by considering the dataset's nature and ACoT prompting's capabilities. Fine-tuning typically helps when there is sufficient data to prevent overfitting, allowing the model to generalize better. However, when the dataset is small, as in this scenario, fine-tuning may cause the model to overfit the specific examples seen during fine-tuning, capturing noise rather than functional general patterns.

For ACoT, the base model's ability to handle complex reasoning through its intermediate steps seems

Table 9.4: Results of LLAMA 3.1 8B on Assembly-101-O with different prompt methods, modalities, and inference modes.

Prompt	Modality	Mode	Accuracy	Precision	Recall	F1
ZS	Textual	Base	28.40	27.30	**98.30**	42.40
	Textual	Finetuned	28.40	27.30	99.30	42.40
	Numerical	Base	27.40	27.10	98.70	43.60
	Numerical	Finetuned	27.20	27.80	98.40	42.60
FS	Textual	Base	47.40	33.80	91.30	48.40
	Textual	Finetuned	46.40	30.80	97.30	45.40
	Numerical	Base	43.40	31.80	92.30	46.20
	Numerical	Finetuned	45.40	32.10	87.30	46.60
ACoT	Textual	Base	**51.40**	**34.80**	90.30	**50.40**
	Textual	Finetuned	46.40	31.80	85.30	47.60
	Numerical	Base	40.40	31.20	93.30	41.60
	Numerical	Finetuned	40.10	32.80	84.30	40.60

sufficiently robust, and it does not benefit from the additional adjustments that fine-tuning provides. Fine-tuning might disturb these established reasoning pathways, leading to a decline in performance. This suggests that ACoT is inherently well-suited to tasks requiring deep reasoning, making it less dependent on the benefits of fine-tuning, especially in scenarios with limited data.

9.5.4 Speed Analysis

Table 9.5 presents a speed test comparison of different prompting methods—ZS, FS, ACoT—using LLAMA 3.1 on the Assembly101-O dataset. The results indicate that ZS is the fastest method, with an average processing speed of 0.208 seconds per sample. FS is slightly slower at 0.216 seconds per sample, representing a marginal 3.8% rise in processing time compared to ZS. ACoT, while offering superior accuracy and reasoning ability, is the slowest, taking 0.315 seconds per sample, which is a 51.4% increase in speed compared to ZS. These differences in speed can be attributed to the inherent complexity of each method. ZS and FS prompt the model to generate answers directly, with FS introducing a minimal overhead due to the additional context provided by the few examples. On the other hand, ACoT requires the model to engage in more complex, step-by-step reasoning, which naturally demands more computational resources and time. While ACoT's performance benefits are clear, these come at the cost of slower processing, which could be a consideration in time-sensitive applications. Thus, the choice of prompting method should balance the trade-off between speed and output quality based on the task's specific requirements.

Table 9.5: Speed Test with LLama 3.1 on Assembly-101-O

Model	Speed (s/sample)
ZS	**0.208**
FS	0.216
ACoT	0.315

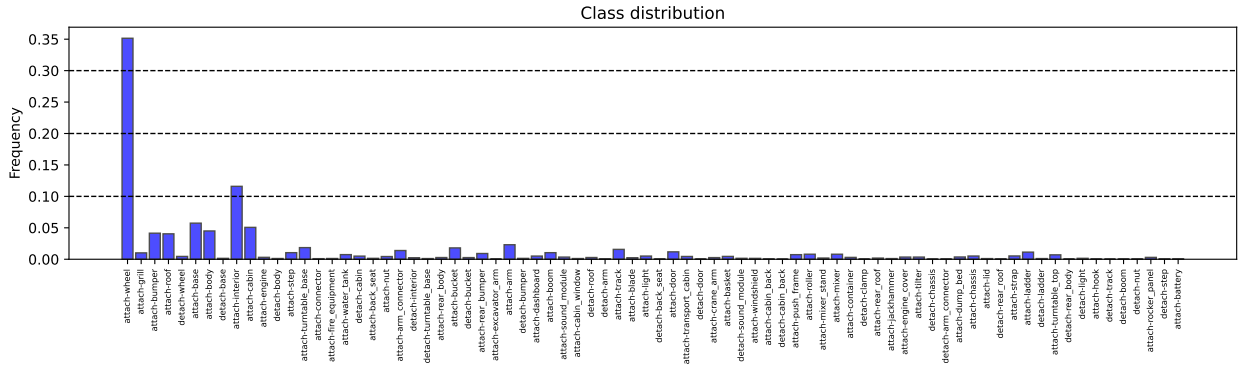


Figure 9.2: Class cardinality in Assembly101-O.

9.6 Aggregation Strategy

9.6.1 Step Recognizer Analysis

The action recognizer is the architecture module responsible for online action detection, a task extensively explored in recent research due to its various possible applications. Several models have been proposed using datasets such as Thumos [137], TVSeries [97], and Fine Action [194], with the current leading model being MiniROAD [10]. The performance of MiniROAD on procedural datasets has been suboptimal, largely due to two key factors: the shift in action distribution between the training and testing phases, and the inherent class imbalance within the dataset. The distribution shift occurs because the actions seen during training often differ from those encountered during testing, leading to poorer generalization. Additionally, Assembly101 contains a significant imbalance in class frequencies (see Figure 9.2), where common actions dominate the dataset while less frequent yet crucial procedural actions are underrepresented. These challenges hinder MiniROAD's ability to effectively capture and recognize the full range of procedural actions in real-world assembly tasks.

9.6.2 Frame Aggregation

These performance issues further degrade the results in subsequent stages. Specifically, they introduce significant noise into the input sequence for the anticipator module. For example, if the model struggles to differentiate between two actions, it may oscillate between them, predicting one and then the other over small intervals. To provide a cleaner input for the aggregation module, we propose implementing aggregation strategies that reduce this noise and stabilize the action predictions. Our approach tackles this challenge by introducing three distinct aggregation methods, as shown in Figure 9.3.

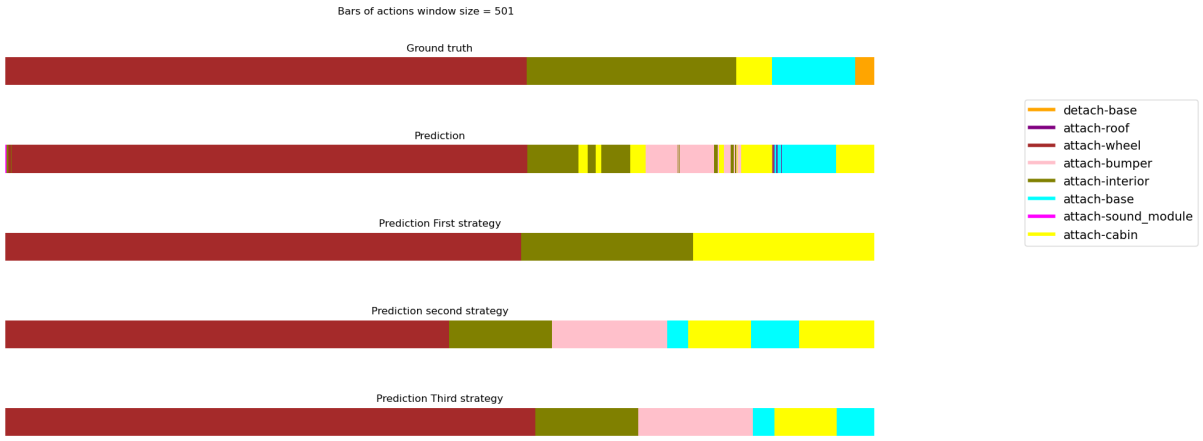


Figure 9.3: Predictions bars: (a) Ground truth predictions, (b) Predictions using the recognizer,(c) First aggregation strategy:, (d) Second aggregation strategy, (e) Third aggregation strategy

Each aggregation strategy's performance will be evaluated using the Levenshtein similarity, which derives from the Levenshtein distance. The similarity between two procedures A and B is defined as:

$$\text{Levenshtein Similarity}(A, B) = 1 - \frac{d_{dev}(A, B)}{\max(len(A), len(B))} \tag{9.3}$$

where: $d_{dev}(A, B)$ is the Levenshtein distance between procedure A and B, which is the minimum number of single-action insertions, deletions, or substitutions required to transform procedure A into procedure B, and len(A) and len(B) represent the length of procedure A and B, respectively. Note that the term max(len(A),len(B)) normalizes the score by the length of the longer procedure, ensuring that the ratio is between 0 and 1.

First strategy

The first aggregation strategy (Figure 9.3.c) we propose consists of a two-step process. In the first step, we calculate the mode for each non-overlapping sliding window of frames of a given length l. Because the windows are non-overlapping, this approach introduces a slight delay in processing, which is a trade-off for improved prediction stability. The mode, representing the most frequent action label within that window, replaces all predictions in that window, ensuring consistency across the frames and removing the noise of smaller portions wrongly predicted. In the second step, we apply the elimination of successive duplicates to remove consecutive repeated action labels, which prevents redundant predictions. This cleaned output serves as the input to the anticipation module, ultimately aiming to produce a more stable and accurate sequence of action predictions.

Second strategy

The second aggregation strategy (Fig. 9.3.d) we propose employs a sliding window approach with a window size of 1, allowing for overlapping portions of frames. In this method, we evaluate the mode within the overlapping window that includes the last frame and its neighboring context. The prediction of the last frame is then substituted with this mode, which captures the most frequent action label in that localized

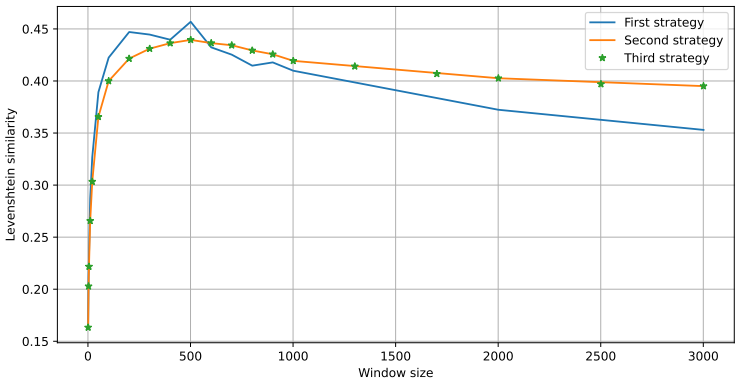


Figure 9.4: Levenshtein similarity for the proposed aggregation strategies. The maximum similarity is achieved by the first strategy when using a window size of 500 frames.

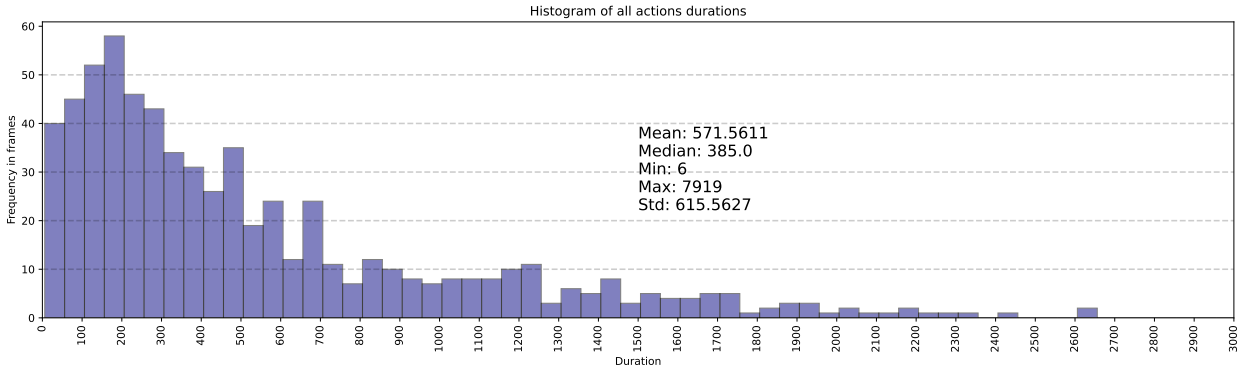


Figure 9.5: Histogram of the durations of the action in Assembly101-O

area. This method is designed with the intent to reduce delay, as it processes frames more dynamically compared to non-overlapping windows.

Third Strategy

The third aggregation strategy (Fig. 9.3.e)uses a sliding window with a size of 1, similar to the previous approach. To address the potential issue of the mode being incorrectly positioned within the window, we opt to substitute the prediction of the central frame rather than the final frame, with some corrections for the initial and final frames in the video. This adjustment ensures that the most frequent action label is applied to a frame that is more representative of the overall context, thus enhancing the accuracy of the prediction.

Figure 9.4 illustrates that the first approach is the most effective in maximizing similarity.

Considering that datasets like Assembly101 are recorded at 30 fps and the average action lasts about 570 frames (Fig. 9.5), we selected 500 frames as a sensible trade-off between achieving high similarity and minimizing the computational burden.

9.7 Experiments

9.7.1 Baselines

To estimate the effectiveness of our model, we evaluate its performance by comparing it against several baseline models based on the metrics presented in Sec. 9.4.2. This evaluation framework builds upon the baselines initially introduced in PREGO [92], as detailed below:

One-step memory In PREGO [92], we defined a *transition matrix* considering only the correct procedures. Specifically, given the set of actions $\mathcal{A}$ in the training set with $|\mathcal{A}| = C$, we define a transition matrix $M \in \mathbb{R}^{C \times C}$, which records, in position (l, m), the number of occurrences where action m follows action l. During evaluation, an action in the test split that does not correspond to a transition recorded in this training set matrix is labeled as a *mistake*. This baseline provides a simple method for recognizing deviations from standard procedural transitions.

OadTR for mistake detection The work in [324] presents a framework for online action detection called OadTR, which uses a Vision Transformer to capture video clips' temporal structure and context. The framework consists of an encoder-decoder architecture: the encoder processes historical observations to output a task token that represents the current action, while the decoder refines this task token by incorporating information from anticipated future clips. In the context of procedural error detection, we consider a mistake to have occurred if the output from the encoder does not match the output from the decoder, indicating a divergence between observed and expected actions.

BERT-based Mistake Detection We also leverage the capabilities of BERT [71], using its specific [CLS] token to predict whether an action sequence is correct or erroneous. More specifically, we fine-tune BERT with a next-sentence-prediction task, training the model to determine whether step B logically follows step A within a given procedure. In our setting, each step is represented as a set of two words, such as *attach wheel*, which describes coarse-level actions. During evaluation, BERT is presented with pairs of actions and predicts whether the sequence of those actions is correct. The advantage of BERT lies in its pre-training on a vast corpus of text, which helps it understand contextual relationships between procedural steps, making it effective for classifying sequences and understanding the logical flow of actions.

In addition to these baselines, we introduce several new baselines to explore corner-case scenarios, offering a broader perspective for comparison:

Best-Case Scenario In the best-case scenario, we assume that the anticipation branch always predicts the correct action. This means that when the procedural context indicates a mistake is expected, the model will predict anything other than the incorrect action, effectively identifying the mistake. Conversely, for correct actions, the model will consistently make accurate predictions. This ensures that all mistakes are flagged correctly, and all correct actions are anticipated without error, serving as an upper bound for the system's performance.

Worst-Case Scenario In the worst-case scenario, the anticipation branch always predicts the incorrect action. Specifically, if a mistake is expected, the model fails to detect it, and if a correct action is expected, the model still predicts an incorrect action. This serves as a lower bound, helping us understand the limitations of our approach under adversarial conditions.

Random Anticipation In the random scenario, the anticipation branch provides random guesses for each action, regardless of the procedural context. This baseline highlights the importance of a well-informed anticipation module by comparing it against a model without contextual awareness of the sequence.

These baselines allow us to comprehensively evaluate our model's performance across a range of con-

Table 9.6: A comparative assessment between Ours and the chosen baseline methods is conducted to detect procedural mistakes using the Assembly101-O and Epic-tent-O datasets.

			Assembly101-O			Epic-tent-O		
	Step Recog.	**Step Antic.**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Best	Oracle		0	0	0	-	-	-
Worst	Oracle		0	0	0	-	-	-
Random	Oracle		45	91	25	-	-	-
One-step memory	*Oracle*		16.3	30.7	21.3	6.6	26.6	10.6
BERT [71]	*Oracle*		78.2	20.0	31.8	75.0	5.6	10.4
PREGO [92]	*Oracle*	*Llama 2*	30.7	94.0	46.3	10.7	86.7	19.1
Ours	*Oracle*	*Llama 3.1*	34.80	90.30	50.40	10.6	93.5	22.4
OadTR for MD [324]	*OadTR [324]*	*OadTR [324]*	24.3	18.1	20.7	6.7	21.7	10.2
PREGO [92]	*OadTR [324]*	*Llama 2*	22.1	**94.2**	35.8	9.5	93.3	17.2
PREGO [92]	*MiniRoad [10]*	*Llama 2*	27.8	84.1	41.8	8.6	20.0	12.0
Ours	*MiniRoad [10]*	*Llama 3.1*	**30.30**	76.8	**43**	**9.8**	**100**	**19.2**

ditions, from ideal to highly challenging scenarios, providing a fair and robust comparison to gauge the effectiveness of our proposed approach.

9.7.2 Sequence-Level Results

We evaluate our model, which utilizes Automatic Chain of Thought for intermediate reasoning, alongside various baseline methods on two datasets, Assembly101-O and Epic-tent-O, as shown in Table 9.2. For this evaluation, we replaced the step recognition predictions with ground truth action labels (Oracle setting) to establish an upper bound on performance, simulating perfect action recognition by the video branch.

The comparison includes several baselines. The One-step memory method only considers the previous action, while BERT takes a more abstract reasoning approach, leveraging past actions more effectively. However, this leads to a conservative bias, reducing false alarms and missing some mistakes. PREGO leverages symbolic reasoning to model richer contextual information, with PREGO$_{Llama}$ demonstrating the strongest performance across most metrics.

In the Assembly101-O dataset, our model achieves a balanced performance, particularly in the F1-score, outperforming many baseline methods. Similarly, on the Epic-tent-O dataset, our model attains competitive precision and recall, highlighting its effectiveness in detecting procedural mistakes across different tasks. Specifically, our model outperforms OadTR, which suffers from the limitations of fixed video segment processing and benefits from leveraging semantic labels directly in reasoning processes, resulting in a 102% improvement in the F1-score.

The results indicate that incorporating the Automatic Chain of Thought enhances intermediate reasoning, allowing our model to anticipate actions more effectively and detect deviations from normal procedure patterns. This positions our model as a powerful alternative, with better procedural mistake detection performance than symbolic reasoning-based approaches like PREGO and other baselines.

9.7.3 Frame-Level Results

During preprocessing, we split the action sequences into windows of fixed size n, selecting the most frequent action within each window as the representative action. Then, we aggregate the actions to remove any repetitions. We further experiment with frame-level metrics by computing the previously defined metrics at the frame level. To achieve this, we reverse the agglomeration process, expanding the output of the

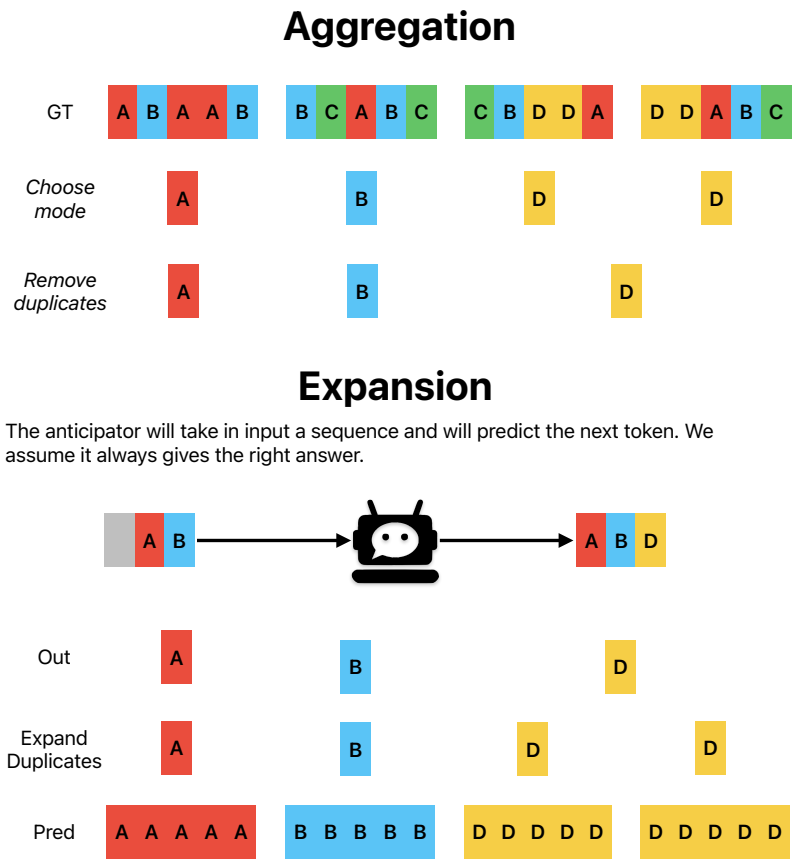


Figure 9.6: With a window frame of size $n > 1$ there can be a mismatch between the ground truth window, whose distribution is often not constant, and the expanded prediction, whose is.

anticipator back to the frame level. This can be done by simply keeping track of the original number of frames in each window during aggregation. Due to the aggregation process and frame windowing, even the best-case scenario does not achieve optimal performance, and the worst-case scenario does not result in an F1 score of zero. See Fig. 9.6 for clarification.

In Table 9.7, we present the results on the Assembly101-O dataset for varying frame window sizes. To provide a clearer comparison of our model, we report on corner cases in the table. Given that our anticipator model is a Large Language Model (LLM) in this setting, we define three corner cases: the best case is when the anticipator always predicts the correct next action, the worst case is when it consistently predicts the wrong action, and the random case is when it predicts a random action from the set of possible actions. See Sec. 9.7.1 for a deeper explanation of corner cases.

All predictions are made in an online manner similar to PREGO, the only difference is that the input is not aggregated and the anticipator works at the frame level, meaning we predict the action associated with each frame individually. We would expect the best and worst case to correspond to a 1 and 0 F1-score, but that happens only when considering each frame individually, and no aggregation is performed. In the other cases aggregation creates issues as explained before; to get an idea: on a total of 380110 frames, when applying a window of size 200 and considering an oracular anticipator, we get 0.05% true positives, 6% false positives, 94% true negatives, and 0 false negatives.

Table 9.7: Performance metrics for different frame windows on the Assembly-101-O dataset.

	Frame Window	Accuracy	Precision	Recall	F1	Balanced Accuracy
Best	1	100.00	100.00	100.00	100.00	100.0
Worst	1	0.00	0.00	0.00	0.00	0.0
Random	1	0.25	0.05	100.00	0.10	50.1
Ours	1	51.50	0.06	62.64	0.12	57.1
Best	50	98.42	2.95	100.00	5.73	99.2
Worst	50	0.00	0.00	0.00	0.00	0.0
Random	50	0.35	0.05	100.00	0.10	50.1
Ours	50	83.93	0.08	25.27	0.15	54.6
Best	200	93.93	0.78	100.00	1.55	97.0
Worst	200	0.01	0.00	0.00	0.00	0.0
Random	200	0.44	0.05	98.35	0.09	49.4
Ours	200	83.07	0.06	20.33	0.11	51.7
Best	400	88.08	0.40	100.00	0.80	94.0
Worst	400	0.03	0.00	0.00	0.00	0.0
Random	400	0.40	0.05	100.00	0.10	50.2
Ours	400	83.87	0.06	21.43	0.13	52.7

9.8 Limitations

While our dual-branch architecture for online procedural mistake detection demonstrates promising results, several limitations must be acknowledged.

Although our model achieves strong performance on the Assembly101-O and Epic-tent-O datasets, its generalizability to other procedural tasks is yet to be fully established. Different procedures or action categories may require further framework adaptation to maintain effectiveness across diverse contexts. The open-set nature of procedural mistakes presents additional complexities, as our model, while designed to handle novel errors, may struggle with unforeseen or atypical actions that fall outside the trained action set.

Real-time processing constraints also pose a challenge. Our dual-branch architecture, particularly when incorporating Large Language Models (LLMs), can introduce latency in inference, which is critical for online applications. Optimizing the model for faster processing while maintaining accuracy remains an ongoing endeavor.

Furthermore, relying on symbolic reasoning simplifies certain aspects of action prediction but may not adequately capture the intricate semantic relationships between actions. Future investigations should explore hybrid approaches that combine symbolic reasoning with richer semantic representations to enhance the model's predictive capabilities.

Finally, our evaluation metrics primarily focus on action recognition and anticipation accuracy. However, it is essential to consider other factors, such as the model's robustness to noise, its ability to navigate ambiguous scenarios, and the interpretability of its predictions. Addressing these limitations will be crucial for advancing procedural mistake detection systems and ensuring their efficacy in dynamic, varied environments.

9.9 Conclusion

This paper explores the challenging task of detecting procedural errors in egocentric videos online, focusing on a dual-branch architecture that combines action recognition and anticipation. By leveraging Large Language Models (LLMs) within the action anticipation branch, we use in-context learning and Automatic-Cahin-of-Thought to demonstrate their ability to predict future steps based on prior actions. Our experiments comprehensively analyze various prompting schemes and aggregation strategies to optimize performance in online action recognition, highlighting the importance of effective, prompt formulation for LLM-based anticipators.

Our results emphasize the challenges of per-frame evaluation in real-time action recognition systems and show how symbolic reasoning, combined with predictive models, enhances procedural mistake detection.

Through extensive experimentation, we demonstrate the effectiveness of our dual-branch architecture, achieving new state-of-the-art performances for online mistake detection. This underscores the potential of integrating action recognition and anticipation in a unified framework for detecting procedural errors as they unfold online. Future work could explore further refinements in symbolic representation and expand the range of procedural tasks and environments in which our framework can be applied.

Chapter 10

Conclusion

This thesis explores the multifaceted landscape of human dynamics, focusing on three interconnected areas: Motion Forecasting, Social Navigation, and Egocentric Perception. Through innovative approaches integrating deep learning, role-based interaction models, and social dynamics, we make significant strides in advancing the prediction and understanding of human behavior within complex environments.

In Motion Forecasting, we go beyond traditional single-person techniques to address multi-person scenarios and the influence of environmental contexts. Our work on two-body pose forecasting [261] and contact-aware global motion forecasting [282] underscores the value of accounting for interpersonal interactions and environmental factors, significantly improving the accuracy of human trajectory and pose predictions. The staged framework further exemplifies how modeling contact points and motion stages can enhance performance in challenging prediction tasks.

The research into Social Navigation highlights the critical role of latent variables and social interaction cues in predicting human behavior, particularly in team-based scenarios and human-robot collaboration. We present a role-based approach [281] that captures these latent dynamics to improve trajectory forecasting in multi-agent systems. Additionally, models like the Social Dynamics Adaptation (SDA) [280] system contribute to real-time interaction in collaborative navigation tasks, establishing a new benchmark in human-robot interaction. Moreover, developing topological tools, such as TopoModelx [119], expands our capacity to represent complex, multi-agent dynamics through topological deep learning, enabling more abstract representations of social interactions.

In Egocentric Perception and Mistake Detection, we present PREGO [93], the first real-time system for detecting procedural errors from egocentric video streams. This model and its extended version integrate action recognition and future action anticipation, aided by the Automatic Chain of Thought, to improve mistake detection in dynamic environments. Our contributions also include the SEE-ME framework, which incorporates social interaction cues for more precise mesh estimation from egocentric videos, marking a significant step forward in first-person video analysis.

These contributions hold broad implications across fields such as robotics, virtual reality, sports analytics, and human-computer interaction. By enhancing the ability to predict and model human behavior in diverse and dynamic contexts, this work paves the way for more intuitive human-robot interactions, safer shared environments, and improved analytics in team-based activities.

There are exciting opportunities to expand upon this work. Future directions involve the integration of Vision-Language Models and Large Language Models to enhance egocentric video understanding further, potentially revolutionizing applications in augmented and virtual reality. Developing more advanced graph-

based models for multi-agent interactions and incorporating reinforcement learning into social navigation could lead to more adaptive and context-aware systems in motion forecasting and social dynamics.

In conclusion, this thesis contributes to theoretical models of human behavior and practical applications in areas where understanding and predicting human motion is critical. The methodologies and tools introduced here contribute to the future of intelligent systems capable of anticipating, understanding, and seamlessly interacting with human behavior in real-world environments.

Chapter 11

Future Work

Upon completing the research work of this thesis, several avenues remain open for further exploration. One such avenue is enhancing the understanding of egocentric video data using Vision-Language Models (VLMs). While the current work has laid the foundation for real-time mistake detection and pose estimation, integrating models like CLIP or GPT-4Vision can bridge the gap between visual and semantic understanding in complex, dynamic environments. Embedding VLMs into existing egocentric video frameworks makes it feasible to process human actions and contextual and object-level understanding. This would allow systems to align visual inputs with natural language descriptions, thereby improving the recognition of routine and more abstract tasks. Such advancements could revolutionize applications in augmented reality, virtual assistants, and healthcare by fostering systems that interpret human behavior and environments with more nuance, leading to adaptive, context-aware technologies.

Additionally, future research could advance the modeling of social interactions within multi-agent systems. Although significant progress has been made in understanding latent social roles and dynamics, especially in contexts like team-based collaboration and social navigation, there remains much to explore in terms of more sophisticated graph-based models. These models could capture the evolving complexity of interactions among agents, particularly in dynamic environments involving human-robot collaborations or group activities like sports. Incorporating topological and reinforcement learning methods could help develop systems that adapt more fluidly to social contexts, predicting human trajectories and higher-order agent interactions. These developments would pave the way for intelligent, interaction-aware agents capable of seamless collaboration in real-world applications, from factory automation and autonomous driving to household robotics.

Bibliography

[1] Microsoft hololens2. https://www.microsoft.com/en-us/hololens.

[2] M. A. Abdelsalam, S. B. Rangrej, I. Hadji, N. Dvornik, K. G. Derpanis, and A. Fazly. Gepsan: Generative procedure step anticipation in cooking videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2988–2997, October 2023.

[3] V. Adeli, E. Adeli, I. D. Reid, J. C. Niebles, and H. Rezatofighi. Socially and contextually aware human motion and pose forecasting. *IEEE Robotics and Automation Letters*, 5:6033–6040, 2020.

[4] M. Ahmed and N. Qin. Surrogate-based aerodynamic design optimization: Use of surrogates in aerodynamic design optimization. *International Conference on Aerospace Sciences and Aviation Technology*, 13(AEROSPACE SCIENCES and AVIATION TECHNOLOGY, ASAT- 13, May 26 - 28, 2009):1–26, 2009.

[5] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. M. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[6] H. Akada, J. Wang, S. Shimada, M. Takahashi, C. Theobalt, and V. Golyanik. Unrealego: A new dataset for robust egocentric 3d human motion capture. In *European Conference on Computer Vision*, 2022.

[7] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Nonrigid structure from motion in trajectory space. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

[8] E. Aksan, M. Kaufmann, P. Cao, and O. Hilliges. A spatio-temporal transformer for 3d human motion prediction. In *2021 International Conference on 3D Vision (3DV)*, 2021.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[10] J. An, H. Kang, S. H. Han, M.-H. Yang, and S. J. Kim. Miniroad: Minimal rnn framework for online action detection. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10307–10316, 2023.

[11] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

[12] M. Andriluka, U. Iqbal, A. Milan, E. Insafutdinov, L. Pishchulin, J. Gall, and B. Schiele. Posetrack: A benchmark for human pose estimation and tracking. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018.

[13] A. Antonucci, G. P. R. Papini, P. Bevilacqua, L. Palopoli, and D. Fontanelli. Efficient prediction of human motion for real-time robotics applications with physics-inspired neural networks. *IEEE Access*, 10:144–157, 2022.

[14] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[15] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv preprint arXiv:2010.04558*, 2020.

[16] K. Ashutosh, S. K. Ramakrishnan, T. Afouras, and K. Grauman. Video-mined task graphs for keystep recognition in instructional videos. *arXiv preprint arXiv:2307.08763*, 2023.

[17] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016.

[18] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[19] S. Barbarossa and S. Sardellitti. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020.

[20] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans. ObjectNav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[21] S. Becker, R. Hug, W. Hubner, and M. Arens. Red: A simple but effective baseline predictor for the trajnet benchmark. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[22] Y. Ben-Shabat, X. Yu, F. Saleh, D. Campbell, C. Rodriguez-Opazo, H. Li, and S. Gould. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 847–859, 2021.

[23] A. Benzine, B. Luvison, Q. C. Pham, and C. Achard. Deep, robust and single shot 3d multiperson human pose estimation from monocular images. In *2019 IEEE International Conference on Image Processing (ICIP)*, 2019.

[24] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19. Springer, 2011.

[25] A. Bhattacharyya, M. Fritz, and B. Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4194–4202, 2018.

[26] B. Biggs, S. Ehrhardt, H. Joo, B. Graham, A. Vedaldi, and D. Novotný. 3d multi-bodies: Fitting sets of plausible 3d human models to ambiguous image data. *ArXiv*, abs/2011.00980, 2020.

[27] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga. Fast differentiable sorting and ranking. In *International Conference on Leadership and Management (ICLM)*, 2020.

[28] C. Bodnar. *Topological Deep Learning: Graphs, Complexes, Sheaves*. PhD thesis, Apollo - University of Cambridge Repository, 2022.

[29] C. Bodnar, F. Frasca, N. Otter, Y. Wang, P. Lio, G. F. Montufar, and M. Bronstein. Weisfeiler and Lehman Go Cellular: CW Networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

[30] A. Bouazizi, A. Holzbock, U. Kressel, K. Dietmayer, and V. Belagiannis. Motionmixer: Mlp-based 3d human body pose forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022.

[31] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[32] T. Brooks, A. Holynski, and A. A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[33] E. Bunch, Q. You, G. Fung, and V. Singh. Simplicial 2-complex convolutional neural networks. In *TDA & Beyond*, 2020.

[34] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[35] Y. Cai, L. Huang, Y. Wang, T.-J. Cham, J. Cai, J. Yuan, J. Liu, X. Yang, Y. Zhu, X. Shen, D. Liu, J. Liu, and N. M. Thalmann. Learning progressive joint propagation for human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[36] T. Campari, P. Eccher, L. Serafini, and L. Ballan. Exploiting scene-specific features for object goal navigation. In *European Conference on Computer Vision*, pages 406–421. Springer, 2020.

[37] T. Campari, L. Lamanna, P. Traverso, L. Serafini, and L. Ballan. Online learning of reusable abstract models for object goal navigation. In *CVPR*, 2022.

[38] E. Cancelli, T. Campari, L. Serafini, A. X. Chang, and L. Ballan. Exploiting proximity-aware tasks for embodied social navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10957–10967, October 2023.

[39] D. Candido de Oliveira, B. T. Nassu, and M. A. Wehrmeister. Image-based detection of modifications in assembled pcbs with deep convolutional autoencoders. *Sensors*, 23(3):1353, 2023.

[40] W. Cao, Z. Yan, Z. He, and Z. He. A comprehensive survey on geometric deep learning. *IEEE Access*, 8:35929–35949, 2020.

[41] Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, and J. Malik. Long-term human motion prediction with scene context. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 387–404. Springer, 2020.

[42] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[43] C. Carling, J. Bloomfield, L. Nelsen, and T. Reilly. The role of motion analysis in elite soccer. *Sports medicine*, 38(10):839–862, 2008.

[44] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

[45] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019.

[46] Z. Chang, E. J. C. Findlay, H. Zhang, and H. P. H. Shum. Unifying human motion synthesis and style transfer with denoising diffusion probabilistic models. In *VISIGRAPP*, 2022.

[47] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020.

[48] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020.

[49] C. Chen, Y. Liu, S. Kreiss, and A. Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[50] C. Chen, C. Zhang, T. Wang, D. Li, Y. Guo, Z. Zhao, and J. Hong. Monitoring of assembly process using deep learning technology. *Sensors*, 20(15):4208, 2020.

[51] G. Chen, Z. Chen, S. Fan, and K. Zhang. Unsupervised sampling promoting for stochastic human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[52] X. Chen, B. Jiang, W. Liu, Z. Huang, B. Fu, T. Chen, J. Yu, and G. Yu. Executing your commands via motion diffusion in latent space. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18000–18010, 2022.

[53] Y. Chen, B. Ivanovic, and M. Pavone. Scept: Scene-consistent, policy-based trajectory predictions for planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17103–17112, June 2022.

[54] Y. F. Chen, M. Everett, M. Liu, and J. P. How. Socially aware motion planning with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[55] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[56] M. Chessa, G. Maiello, L. K. Klein, V. C. Paulun, and F. Solari. Grasping objects in immersive virtual reality. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1749–1754, 2019.

[57] E. Chien, C. Pan, J. Peng, and O. Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022.

[58] H.-K. Chiu, E. Adeli, B. Wang, D.-A. Huang, and J. C. Niebles. Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[59] M.-Y. Cho and Y. Jeong. Human gesture recognition performance evaluation for service robots. *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 847–851, 2017.

[60] H. Choi, G. Moon, and K. M. Lee. Beyond static features for temporally consistent 3d human pose and shape from a video. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1973, 2020.

[61] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015.

[62] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. page 215–230, 2012.

[63] CMU Graphics Lab. CMU Graphics Lab Motion Capture Database. `http://mocap.cs.cmu.edu`.

[64] E. Corona, A. Pumarola, G. Alenya, and F. Moreno-Noguer. Context-aware human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[65] M. Cuturi, O. Teboul, and J.-P. Vert. Differentiable ranking and sorting using optimal transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[66] R. Dabral, N. Gundavarapu, R. Mitra, A. Sharma, G. Ramakrishnan, and A. Jain. Dynamic multiscale graph neural networks for 3d skeleton-based human motion prediction. In *Multi-person 3d human pose estimation from monocular images. In 2019 International Conference on 3D Vision (3DV)*, 2019.

[67] R. Dabral, M. H. Mughal, V. Golyanik, and C. Theobalt. Mofusion: A framework for denoising-diffusion-based motion synthesis. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9760–9770, 2022.

[68] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[69] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[70] M. Deitke, D. Batra, Y. Bisk, T. Campari, A. X. Chang, D. S. Chaplot, C. Chen, C. P. D'Arpino, K. Ehsani, A. Farhadi, et al. Retrospectives on the embodied ai workshop. *arXiv preprint arXiv:2210.06849*, 2022.

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[72] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233, 2021.

[73] M. Di Luca and D. Rhodes. Optimal perceived timing: Integrating sensory information with dynamically updated expectations. *Sci. Rep.*, 6:28563, July 2016.

[74] G. Ding, F. Sener, S. Ma, and A. Yao. Every mistake counts in assembly. *arXiv preprint arXiv:2307.16453*, 2023.

[75] K. Ding, J. Wang, J. Li, D. Li, and H. Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4927–4936, Online, Nov. 2020. Association for Computational Linguistics.

[76] Y. Dong, W. Sawin, and Y. Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *ICML Graph Representation Learning and Beyond Workshop*, 2020.

[77] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[78] N. F. Duarte, M. Raković, J. Tasevski, M. I. Coco, A. Billard, and J. Santos-Victor. Action anticipation: Reading the intentions of humans and robots. *IEEE Robotics and Automation Letters*, 3(4):4132–4139, 2018.

[79] G. Edwards, P. Vetter, F. McGruer, L. S. Petro, and L. Muckli. Predictive feedback to V1 dynamically updates with sensory input. *Sci. Rep.*, 7(1):16538, Nov. 2017.

[80] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. 2019.

[81] D. Ellis, E. Sommerlade, and I. Reid. Modelling pedestrian trajectory patterns with gaussian processes. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1229–1234, 2009.

[82] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision – ECCV 2014*, 2014.

[83] A. Ess, B. Leibe, and L. Van Gool. Depth and appearance for mobile scene analysis. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[84] P. Felsen, P. Lucey, and S. Ganguly. Where will they go? predicting finegrained adversarial multi-agent motion using conditional variational autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 732–747, 2018.

[85] J. Feng, R. Xu, J. Hao, H. Sharma, Y. Shen, D. Zhao, and W. Chen. Language models can be logical solvers. *ArXiv*, abs/2311.06158, 2023.

[86] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[87] G. Ferrer, A. Garrell, and A. Sanfeliu. Social-aware robot navigation in urban environments. In *Proc. of thr European Conference on Mobile Robots*, 2013.

[88] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv*, abs/1903.02428, 2019.

[89] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[90] M. Fieraru, M. Zanfir, E. Oneata, A.-I. Popa, V. Olaru, and C. Sminchisescu. Three-dimensional reconstruction of human interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[91] A. Flaborea, L. Collorone, G. M. D. di Melendugno, S. D'Arrigo, B. Prenkaj, and F. Galasso. Multimodal motion conditioned diffusion model for skeleton-based video anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10318–10329, October 2023.

[92] A. Flaborea, G. M. D. di Melendugno, L. Plini, L. Scofano, E. De Matteis, A. Furnari, G. M. Farinella, and F. Galasso. Prego: Online mistake detection in procedural egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18483–18492, June 2024.

[93] A. Flaborea, G. M. D. di Melendugno, L. Plini, L. Scofano, E. D. Matteis, A. Furnari, G. M. Farinella, and F. Galasso. Prego: Online mistake detection in procedural egocentric videos. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18483–18492, 2024.

[94] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[95] E. Gärtner, M. Andriluka, H. Xu, and C. Sminchisescu. Trajectory optimization for physics-based reconstruction of 3d human pose from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13106–13115, June 2022.

[96] E. Gärtner, M. Andriluka, H. Xu, and C. Sminchisescu. Trajectory optimization for physics-based reconstruction of 3d human pose from monocular video. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13096–13105, 2022.

[97] R. D. Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars. Online action detection, 2016.

[98] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

[99] T. Gervet, S. Chintala, D. Batra, J. Malik, Devendra, and S. Chaplot. Navigating to objects in the real. *Science Robotics*, 2023.

[100] R. Ghoddoosian, I. Dwivedi, N. Agarwal, and B. Dariush. Weakly-supervised action segmentation and unseen error detection in anomalous instructional videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10128–10138, October 2023.

[101] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso. Transformer networks for trajectory forecasting. In *In The International Conference on Pattern Recognition (ICPR)*, 2020.

[102] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso. Transformer networks for trajectory forecasting. In *2020 25th international conference on pattern recognition (ICPR)*, pages 10335–10342. IEEE, 2021.

[103] L. Giusti, C. Battiloro, P. Di Lorenzo, S. Sardellitti, and S. Barbarossa. Simplicial attention networks. *arXiv preprint arXiv:2203.07485*, 2022.

[104] L. Giusti, C. Battiloro, L. Testa, P. Di Lorenzo, S. Sardellitti, and S. Barbarossa. Cell attention networks. *arXiv preprint arXiv:2209.08179*, 2022.

[105] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9. PMLR, 2010.

[106] D. S. González, J. Pérez, V. M. Montero, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17:1135–1145, 2016.

[107] C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022.

[108] W. Guo, X. Bie, X. Alameda-Pineda, and F. Moreno-Noguer. Multi-person extreme motion prediction. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[109] W. Guo, X. Bie, X. Alameda-Pineda, and F. Moreno-Noguer. Multi-person extreme motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13053–13064, 2022.

[110] W. Guo, E. Corona, F. Moreno-Noguer, and X. Alameda-Pineda. Pi-net: Pose interacting network for multi-person monocular 3d pose estimation. In *In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[111] W. Guo, Y. Du, X. Shen, V. Lepetit, X. Alameda-Pineda, and F. Moreno-Noguer. Back to mlp: A simple baseline for human motion prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4809–4819, 2023.

[112] W. Guo, Y. Du, X. Shen, V. Lepetit, A.-P. Xavier, and M.-N. Francesc. Back to mlp: A simple baseline for human motion prediction. *arXiv preprint arXiv:2207.01567*, 2022.

[113] X. Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery.

[114] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[115] T. Gupta and A. Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.

[116] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro. Human-friendly robot navigation in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[117] A. A. Hagberg, D. A. Schult, P. Swart, and J. Hagberg. Exploring network structure, dynamics, and function using networkx. *Proceedings of the Python in Science Conference*, 2008.

[118] M. Hajij, K. Istvan, and G. Zamzmi. Cell complex neural networks. *ArXiv*, abs/2010.00743, 2020.

[119] M. Hajij, M. Papillon, F. Frantzen, J. Agerberg, I. AlJabea, R. Ballester, C. Battiloro, G. Bern'ardez, T. Birdal, A. Brent, P. Chin, S. Escalera, O. H. Gardaa, G. Gopalakrishnan, D. Govil, J. Hoppe, M. R. Karri, J. Khouja, M. Lecha, N. Livesay, J. Meissner, S. Mukherjee, A. Nikitin, T. Papamarkou, J. Pr'ilepok, K. N. Ramamurthy, P. Rosen, A. Guzm'an-S'aenz, A. Salatiello, S. N. Samaga, M. T. Schaub, L. Scofano, I. Spinelli, L. Telyatnikov, Q. Truong, R. Walters, M. Yang, O. Zaghen, G. Zamzmi, A. Zia, and N. Miolane. Topox: A suite of python packages for machine learning on topological domains. *ArXiv*, abs/2402.02441, 2024.

[120] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzmán-Sáenz, and K. N. Ramamurthy. Higher-order attention networks. *arXiv preprint arXiv:2206.00606*, 2022.

[121] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzm'an-S'aenz, K. N. Ramamurthy, T. Birdal, T. K. Dey, S. Mukherjee, S. N. Samaga, N. Livesay, R. Walters, P. Rosen, and M. T. Schaub. Topological deep learning: Going beyond graph data. 2022.

[122] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzmán-Sáenz, K. N. Ramamurthy, T. Birdal, T. K. Dey, S. Mukherjee, S. N. Samaga, N. Livesay, R. Walters, P. Rosen, and M. T. Schaub. Topological deep learning: Going beyond graph data, 2023.

[123] M. Hassan, D. Ceylan, R. Villegas, J. Saito, J. Yang, Y. Zhou, and M. J. Black. Stochastic scene-aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11374–11384, 2021.

[124] M. Hassan, V. Choutas, D. Tzionas, and M. J. Black. Resolving 3d human pose ambiguities with 3d scene constraints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2282–2292, 2019.

[125] S. Hauri, N. Djuric, V. Radosavljevic, and S. Vucetic. Multi-modal trajectory prediction of nba players. In *Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[126] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[127] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[128] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, may 1995.

[129] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37:388–427, jan 2021.

[130] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[131] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural Computation*, pages 1735—-1780, 1997.

[132] D. Holden, J. Saito, T. Komura, and T. Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia*, 2015.

[133] R. Hori, R. Hachiuma, M. Isogawa, D. Mikami, and H. Saito. Silhouette-based 3d human pose estimation using a single wrist-mounted 360° camera. *IEEE Access*, PP:1–1, 2022.

[134] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[135] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[136] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, 2019.

[137] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, Feb. 2017.

[138] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[139] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[140] B. Ivanovic and M. Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[141] E. Jahangiri and A. L. Yuille. Generating multiple diverse hypotheses for human 3d pose consistent with 2d joint detections. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 805–814, 2017.

[142] A. Jain, A. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graph. In *The IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[143] Y. Jang, B. Sullivan, C. Ludwig, I. Gilchrist, D. Damen, and W. Mayol-Cuevas. Epic-tent: An egocentric video dataset for camping tent assembly. In *Int. Conf. Comput. Vis.*, Oct 2019.

[144] H. Jiang and K. Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3509, 2016.

[145] H. Jiang and V. K. Ithapu. Egocentric pose estimation from human vision span. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10986–10994, 2021.

[146] J. Jiang, P. Streli, H. Qiu, A. R. Fender, L. Laich, P. Snape, and C. Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. In *European Conference on Computer Vision*, 2022.

[147] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2017.

[148] R. Khirodkar, A. Bansal, L. Ma, R. A. Newcombe, M. Vo, and K. Kitani. Egohumans: An egocentric 3d multi-human benchmark. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19750–19762, 2023.

[149] S. Kim, D. Huang, Y. Xian, O. Hilliges, L. V. Gool, and X. Wang. Palm: Predicting actions through language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[150] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[151] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[152] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[153] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

[154] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[155] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[156] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[157] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *In European Conference on Computer Vision ECCV*, pages 201–214, 2012.

[158] H. Kivrak, F. Çakmak, H. Kose, and S. Yavuz. Social navigation framework for assistive robots in human inhabited unknown environments. *Engineering Science and Technology, an International Journal*, 2021.

[159] M. Kocabas, N. Athanasiou, and M. J. Black. Vibe: Video inference for human body pose and shape estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5252–5262, 2019.

[160] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2252–2261, 2019.

[161] N. Kolotouros, G. Pavlakos, D. Jayaraman, and K. Daniilidis. Probabilistic modeling for human mesh recovery. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11585–11594, 2021.

[162] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[163] H. S. Koppula and A. Saxena. Anticipating human activities for reactive robotic response. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2071–2071, 2013.

[164] H. S. Koppula and A. Saxena. Anticipating human activities for reactive robotic response. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2071–2071, 2013.

[165] P. Kothari, S. Kreiss, and A. Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7386–7400, 2022.

[166] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *CoRR*, abs/1511.06856, 2016.

[167] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, 2020.

[168] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[169] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[170] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*, 2014.

[171] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

[172] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik. Adapting rapid motor adaptation for bipedal robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[173] T. Lan, Y. Wang, W. Yang, S. N. Robinovitch, and G. Mori. Discriminative latent models for recognizing contextual group activities. volume 34, pages 1549–1562, 2012.

[174] N. W. Landry, M. Lucas, I. Iacopini, G. Petri, A. C. Schwarze, A. Patania, and L. Torres. Xgi: A python package for higher-order interaction networks. *J. Open Source Softw.*, 8:5162, 2023.

[175] M. Lee, S. S. Sohn, S. Moon, S. Yoon, M. Kapadia, and V. Pavlovic. Muse-vae: Multi-scale vae for environment-aware long term trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2221–2230, June 2022.

[176] S.-P. Lee, Z. Lu, Z. Zhang, M. Hoai, and E. Elhamifar. Error detection in egocentric procedural task videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18655–18666, 2024.

[177] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[178] C. Li and G. H. Lee. Generating multiple hypotheses for 3d human pose estimation with mixture density network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9879–9887, 2019.

[179] C. Li, F. Xia, R. Mart'in-Mart'in, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *ArXiv*, abs/2108.03272, 2021.

[180] C. Li, Z. Zhang, W. S. Lee, and G. H. Lee. Convolutional sequence to sequence model for human dynamics. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[181] J. Li, C. K. Liu, and J. Wu. Ego-body pose estimation via ego-head pose estimation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17142–17151, 2022.

[182] J. Li, F. Yang, M. Tomizuka, and C. Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *arXiv: Computer Vision and Pattern Recognition*, 2020.

[183] J. Li, F. Yang, M. Tomizuka, and C. Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. 10 2020.

[184] L. Li, J. Yao, L. Wenliang, T. He, T. Xiao, J. Yan, D. Wipf, and Z. Zhang. Grin: Generative relation and intention network for multi-agent trajectory prediction. *Advances in Neural Information Processing Systems*, 34, 2021.

[185] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:3316–3333, 2022.

[186] M. Li, S. Chen, Y. Zhao, Y. Zhang, Y. Wang, and Q. Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[187] M. Li, S. Chen, Y. Zhao, Y. Zhang, Y. Wang, and Q. Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 214–223, 2020.

[188] Q. Li, Z. Han, and X.-m. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[189] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. R. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2022.

[190] J. Liang, L. Jiang, and A. Hauptmann. Simaug: Learning robust representations from simulation for trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 275–292. Springer, 2020.

[191] J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann. The garden of forking paths: Towards multi-future trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10508–10518, 2020.

[192] Y. Liang, K. Ellis, and J. Henriques. Rapid motor adaptation for robotic manipulator arms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[193] X. T. Liu, J. S. Firoz, A. Lumsdaine, C. A. Joslyn, S. G. Aksoy, B. Praggastis, and A. H. Gebremedhin. Parallel algorithms for efficient computation of high-order line graphs of hypergraphs. *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 312–321, 2021.

[194] Y. Liu, L. Wang, Y. Wang, X. Ma, and Y. Qiao. Fineaction: A fine-grained video dataset for temporal action localization. *IEEE Transactions on Image Processing*, 31:6937–6950, 2022.

[195] Y. Liu, Q. Yan, and A. Alahi. Social nce: Contrastive learning of socially-aware motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15118–15129, 2021.

[196] Y. Liu, J. Yang, X. Gu, Y. Guo, and G. zhong Yang. Egohmr: Egocentric human mesh recovery via hierarchical latent diffusion model. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9807–9813, 2023.

[197] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-voxel cnn for efficient 3d deep learning. *Neural Information Processing Systems (NeurIPS)*, abs/1907.03739, 2019.

[198] P. Long, W. Liu, and J. Pan. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2):656–663, 2017.

[199] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2015.

[200] A. Loquercio, A. Kumar, and J. Malik. Learning visual locomotion with cross-modal supervision. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7295–7302. IEEE, 2023.

[201] Y. Lou, L. Zhu, Y. Wang, X. Wang, and Y. Yang. Diversemotion: Towards diverse human motion generation via discrete diffusion. *ArXiv*, abs/2309.01372, 2023.

[202] Y. Lu, X. Ruan, and J. Huang. Deep reinforcement learning based on social spatial–temporal graph convolution network for crowd navigation. *Machines*, 10(8):703, 2022.

[203] Z. Lu and E. Elhamifar. Set-supervised action learning in procedural task videos via pairwise order consistency. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19903–19913, June 2022.

[204] L. Luo, S. Zhou, W. Cai, M. Y. H. Low, F. Tian, Y. Wang, X. Xiao, and D. Chen. Agent-based human behavior modeling for crowd simulation. *Comput. Animat. Virtual Worlds*, 19(3–4):271–281, sep 2008.

[205] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[206] Z. Luo, S. A. Golestaneh, and K. M. Kitani. 3d human motion estimation via motion compression and refinement. In *Asian Conference on Computer Vision*, 2020.

[207] Z. Luo, R. Hachiuma, Y. Yuan, and K. Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *Advances in Neural Information Processing Systems*, 34:25019–25032, 2021.

[208] T. Ma, Y. Nie, C. Long, Q. Zhang, and G. Li. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[209] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[210] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. Amass: Archive of motion capture as surface shapes. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019.

[211] W. Mao, R. I. Hartley, M. Salzmann, et al. Contact-aware human motion forecasting. *Advances in Neural Information Processing Systems*, 35:7356–7367, 2022.

[212] W. Mao, M. Liu, and M. Salzmann. History repeats itself: Human motion prediction via motion attention. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[213] W. Mao, M. Liu, and M. Salzmann. History repeats itself: Human motion prediction via motion attention. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 474–489. Springer, 2020.

[214] W. Mao, M. Liu, M. Salzmann, and H. Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[215] W. Mao, M. Liu, M. Salzmann, and H. Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9489–9497, 2019.

[216] W. Mao, M. Liu, M. Salzmann, and H. Li. Multi-level motion attention for human motion prediction. *International Journal of Computer Vision*, 129(9):2513–2535, 2021.

[217] W. Mao, M. Liu, M. Salzmann, and H. Li. Multi-level motion attention for human motion prediction. *International journal of computer vision*, 129(9):2513–2535, 2021.

[218] E. V. Mascaró, H. Ahn, and D. Lee. Intention-conditioned long-term human egocentric action anticipation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6048–6057, 2023.

[219] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Trans. Graph.*, 36:44:1–44:14, 2017.

[220] C. Merhej, R. J. Beal, T. Matthews, and S. Ramchurn. What happened next? using deep learning to value defensive actions in football event-data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3394–3403, 2021.

[221] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*, 2019.

[222] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng. Large language models as general pattern machines. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, 2023.

[223] D. Mishkin and J. Matas. All you need is a good init. In *4th International Conference on Learning Representations, ICLR*, 2016.

[224] X. Mo, Y. Xing, and C. Lv. Graph and recurrent neural network-based vehicle trajectory prediction for highway driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1934–1939, 2021.

[225] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14424–14432, 2020.

[226] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *CVPR*, 2020.

[227] A. Mohamed, D. Zhu, W. Vu, M. Elhoseiny, and C. Claudel. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 463–479. Springer, 2022.

[228] A. A. Mohamed, D. Zhu, W. Vu, M. Elhoseiny, and C. G. Claudel. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[229] R. Möller, A. Furnari, S. Battiato, A. Härmä, and G. M. Farinella. A survey on human-aware robot navigation. *Robotics and Autonomous Systems*, 145:103837, 2021.

[230] A. Monti, A. Bertugli, S. Calderara, and R. Cucchiara. Dag-net: Double attentive graph neural network for trajectory forecasting. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2551–2558, 2021.

[231] E. Morgulev, O. H. Azar, and R. Lidor. Sports analytics and the big-data era. *International Journal of Data Science and Analytics*, 5(4):213–222, 2018.

[232] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardòs. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[233] M. Narasimhan, L. Yu, S. Bell, N. Zhang, and T. Darrell. Learning and verification of task structure in instructional videos. *arXiv preprint arXiv:2303.13519*, 2023.

[234] E. Ng, H. Joo, L. Hu, H. Li, T. Darrell, A. Kanazawa, and S. Ginosar. Learning to listen: Modeling non-deterministic dyadic facial motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20395–20405, 2022.

[235] E. Ng, D. Xiang, H. Joo, and K. Grauman. You2me: Inferring body pose in egocentric video via first and second person interactions. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9887–9897, 2019.

[236] T. P. Oikarinen, D. C. Hannah, and S. Kazerounian. Graphmdn: Leveraging graph structure and deep learning to solve inverse problems. *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020.

[237] A. Olmo, S. Sreedharan, and S. Kambhampati. Gpt3-to-plan: Extracting plans from text using gpt-3. *FinPlan 2021*, page 24, 2021.

[238] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1:33–55, 2016.

[239] V. Pallagani, B. Muppasani, K. Murugesan, F. Rossi, L. Horesh, B. Srivastava, F. Fabiano, and A. Loreggia. Plansformer: Generating symbolic plans using transformers. *ArXiv*, abs/2212.08681, 2022.

[240] M. Papillon, M. Hajij, F. Frantzen, J. Hoppe, H. Jenne, J. Mathe, A. Myers, T. Papamarkou, M. T. Schaub, G. Zamzmi, T. Birdal, T. K. Dey, T. Doster, T. H. Emerson, G. Gopalakrishnan, D. Govil, V. P. Grande, A. Guzm'an-S'aenz, H. Kvinge, N. Livesay, J. Meisner, S. Mukherjee, S. N. Samaga, K. N. Ramamurthy, M. R. Karri, P. Rosen, S. Sanborn, M. Scholkemper, R. Walters, J. Agerberg, G. Bokman, S. Barikbin, C. Battiloro, G. Bazhenov, G. Bernardez, A. Brent, S. Escalera, S. Fiorellino, D. Gavrilev, M. Hassanin, P. Hausner, O. H. Gardaa, A. Khamis, M. Lecha, G. Magai, T. Malygina, P. Melnyk, R. Ballester, K. V. Nadimpalli, A. Nikitin, A. Rabinowitz, A. Salatiello, S. Scardapane, L. Scofano, S. Singh, J. Sjolund, P. Snopov, I. Spinelli, L. Telyatnikov, L. Testa, M. Yang, Y. Yue, O. Zaghen, A. Zia, and N. Miolane. Icml 2023 topological deep learning challenge : Design and results. In *TAG-ML*, 2023.

[241] M. Papillon, S. Sanborn, M. Hajij, and N. Miolane. Architectures of topological deep learning: A survey on topological neural networks. *arXiv preprint arXiv:2304.10031*, 2023.

[242] R. Partsey, E. Wijmans, N. Yokoyama, O. Dobosevych, D. Batra, and O. Maksymets. Is mapping necessary for realistic pointgoal navigation? In *CVPR*, 2022.

[243] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, 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. *ArXiv*, abs/1912.01703, 2019.

[244] B. K. Patle, G. B. L., A. Pandey, D. R. Parhi, and A. Jagadeesh. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 2019.

[245] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[246] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1263–1272, 2016.

[247] R. Peddi, S. Arya, B. Challa, L. Pallapothula, A. Vyas, J. Wang, Q. Zhang, V. Komaragiri, E. Ragan, N. Ruozzi, Y. Xiang, and V. Gogate. CaptainCook4D: A dataset for understanding errors in procedural activities, 2023.

[248] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, R. J. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *ArXiv*, abs/1201.0490, 2011.

[249] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *SCA '07*, 2007.

[250] S. Pellegrini, A. Ess, and L. V. Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In *ECCV*, 2010.

[251] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph.*, 37:178, 2018.

[252] M. Petrovich, M. J. Black, and G. Varol. Temos: Generating diverse human motions from textual descriptions. *ArXiv*, abs/2204.14109, 2022.

[253] C. Plizzari, G. Goletto, A. Furnari, S. Bansal, F. Ragusa, G. M. Farinella, D. Damen, and T. Tommasi. An outlook into the future of egocentric vision. *ArXiv*, abs/2308.07123, 2023.

[254] X. Puig, K. K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. Virtualhome: Simulating household activities via programs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[255] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Berges, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi. Habitat 3.0: A co-habitat for humans, avatars, and robots. In *The Twelfth International Conference on Learning Representations*, 2024.

[256] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In *Proceedings of The 6th Conference on Robot Learning (CoRL), PMLR*, 2023.

[257] Z. Qi, S. Wang, C. Su, L. Su, Q. Huang, and Q. Tian. Self-regulated learning for egocentric video activity anticipation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6715–6730, 2023.

[258] Y. Qian, W. Luo, D. Lian, X. Tang, P. Zhao, and S. Gao. Svip: Sequence verification for procedures in videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19890–19902, June 2022.

[259] F. Ragusa, A. Furnari, S. Livatino, and G. M. Farinella. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1569–1578, January 2021.

[260] F. Ragusa, R. Leonardi, M. Mazzamuto, C. Bonanno, R. Scavo, A. Furnari, and G. M. Farinella. Enigma-51: Towards a fine-grained understanding of human-object interactions in industrial scenarios. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[261] M. R. U. Rahman, L. Scofano, E. D. Matteis, A. Flaborea, A. Sampieri, and F. Galasso. Best practices for 2-body pose forecasting. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3614–3624, 2023.

[262] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *CVPR*, 2022.

[263] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, et al. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. *arXiv preprint arXiv:2109.08238*, 2021.

[264] V. Ramanishka, Y.-T. Chen, T. Misu, and K. Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7699–7707, 2018.

[265] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 206–213, 2017.

[266] S. Raychaudhuri, T. Campari, U. Jain, M. Savva, and A. X. Chang. Mopa: Modular object navigation with pointgoal agents. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5763–5773, January 2024.

[267] R. Rein and D. Memmert. Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. *SpringerPlus*, 5(1):1–13, 2016.

[268] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas. Humor: 3d human motion model for robust pose estimation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11468–11479, 2021.

[269] D. Rempe, L. J. Guibas, A. Hertzmann, B. C. Russell, R. Villegas, and J. Yang. Contact and human dynamics from monocular video. In *Symposium on Computer Animation*, 2020.

[270] T. M. Roddenberry, N. Glaze, and S. Segarra. Principled simplicial neural networks for trajectory prediction. In *International Conference on Machine Learning*, pages 9020–9029. PMLR, 2021.

[271] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021.

[272] B. Rozemberczki, O. Kiss, and R. Sarkar. Karate club: An api oriented open-source python framework for unsupervised learning on graphs. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020.

[273] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1349–1358, 2019.

[274] A. Sampieri, G. M. D'Amely di Melendugno, A. Avogaro, F. Cunico, F. Setti, G. Skenderi, M. Cristani, and F. Galasso. Pose forecasting in industrial human-robot collaboration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[275] A. Sampieri, G. M. D. di Melendugno, A. Avogaro, F. Cunico, F. Setti, G. Skenderi, M. Cristani, and F. Galasso. Pose forecasting in industrial human-robot collaboration. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*, pages 51–69. Springer, 2022.

[276] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied AI research. In *ICCV*, 2019.

[277] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.

[278] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra. Signal processing on higher-order networks: Livin'on the edge... and beyond. *Signal Processing*, 187:108149, 2021.

[279] T. J. Schoonbeek, T. Houben, H. Onvlee, F. van der Sommen, et al. Industreal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4365–4374, 2024.

[280] L. Scofano, A. Sampieri, T. Campari, V. Sacco, I. Spinelli, L. Ballan, and F. Galasso. Following the human thread in social navigation. *ArXiv*, abs/2404.11327, 2024.

[281] L. Scofano, A. Sampieri, G. Re, M. Almanza, A. Panconesi, and F. Galasso. About latent roles in forecasting players in team sports. *AI4ABM@ICLR Workshop*, 2023.

[282] L. Scofano, A. Sampieri, E. Schiele, E. D. Matteis, L. Leal-Taix'e, and F. Galasso. Staged contact-aware global human motion forecasting. In *British Machine Vision Conference*, 2023.

[283] T. Seidl, A. Cherukumudi, A. T. Hartnett, P. Carr, and P. Lucey. Bhostgusters : Realtime interactive play sketching with synthesized nba defenses. In *MIT Sloan Sports Analytics COnference*, 2010.

[284] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[285] A. Shah, B. Lundell, H. Sawhney, and R. Chellappa. Steps: Self-supervised key step extraction and localization from unlabeled procedural videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10375–10387, October 2023.

[286] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, et al. iGibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[287] Y. Shen and E. Elhamifar. Progress-aware online action segmentation for egocentric procedural task videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18186–18197, 2024.

[288] L. Shi, L. Wang, C. Long, S. Zhou, M. Zhou, Z. Niu, and G. Hua. Sgcn:sparse graph convolution network for pedestrian trajectory prediction. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[289] S. Shimada, V. Golyanik, W. Xu, P. P'erez, and C. Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics (TOG)*, 40:1 – 15, 2021.

[290] T. Shiratori, H. S. Park, L. Sigal, Y. Sheikh, and J. K. Hodgins. Motion capture from body-mounted cameras. *ACM SIGGRAPH 2011 papers*, 2011.

[291] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4584–4589, 2010.

[292] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso. Space-time-separable graph convolutional network for pose forecasting. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[293] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso. Space-time-separable graph convolutional network for pose forecasting. In *International Conference on Computer Vision (ICCV)*, 2021.

[294] S. Starke, H. Zhang, T. Komura, and J. Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6):209–1, 2019.

[295] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13. Association for Computing Machinery, 2013.

[296] S. Sumikura, M. Shibuya, and K. Sakurada. Openvslam: A versatile visual slam framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 2292–2295, New York, NY, USA, 2019. Association for Computing Machinery.

[297] C. Sun, P. Karlsson, J. Wu, J. Tenenbaum, and K. Murphy. Stochastic prediction of multi-agent interactions from partial observations. 09 2018.

[298] D. Surís, S. Menon, and C. Vondrick. Vipergpt: Visual inference via python execution for reasoning. *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2023.

[299] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *ArXiv*, abs/1409.3215, 2014.

[300] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondru, S. Dharur, F. Meier, W. Galuba, A. X. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *ArXiv*, abs/2106.14405, 2021.

[301] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1207–1216, 2019.

[302] Z. Teed and J. Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021.

[303] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

[304] D. Tomè, T. Alldieck, P. Peluse, G. Pons-Moll, L. de Agapito, H. Badino, and F. de la Torre. Selfpose: 3d egocentric pose estimation from a headset mounted camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:6794–6806, 2020.

[305] D. Tome, P. Peluse, L. Agapito, and H. Badino. xr-egopose: Egocentric 3d human pose from an hmd camera. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7727–7737, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[306] Z. Tong, Y. Liang, C. Sun, D. S. Rosenblum, and A. Lim. Directed graph convolutional network, 2020.

[307] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[308] P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[309] N. Tsoi, A. Xiang, P. Yu, S. S. Sohn, G. Schwartz, S. Ramesh, M. Hussein, A. W. Gupta, M. Kapadia, and M. Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, 7:11047–11054, 2022.

[310] H.-Y. F. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *Neural Information Processing Systems*, 2017.

[311] J. van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935, 2008.

[312] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[313] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[314] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[315] T. Von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European conference on computer vision (ECCV)*, pages 601–617, 2018.

[316] B. Wang, E. Adeli, H.-K. Chiu, D.-A. Huang, and J. C. Niebles. Imitation learning for human pose prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[317] J. Wang, L. Liu, W. Xu, K. Sarkar, D. C. Luvizon, and C. Theobalt. Scene-aware egocentric 3d human pose estimation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13031–13040, 2022.

[318] J. Wang, Y. Rong, J. Liu, S. Yan, D. Lin, and B. Dai. Towards diverse and natural scene-aware 3d human motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20460–20469, 2022.

[319] J. Wang, H. Xu, M. Narasimhan, and X. Wang. Multi-person 3d motion prediction with multi-range transformers, 2021.

[320] J. Wang, H. Xu, M. G. Narasimhan, and X. Wang. Multi-person 3d motion prediction with multi-range transformers. In *Neural Information Processing Systems*, 2021.

[321] J. Wang, H. Xu, J. Xu, S. Liu, and X. Wang. Synthesizing long-term 3d human motion and interaction in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9401–9411, June 2021.

[322] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv: Learning*, 2019.

[323] X. Wang, T. Kwon, M. Rad, B. Pan, I. Chakraborty, S. Andrist, D. Bohus, A. Feniello, B. Tekin, F. V. Frujeri, N. Joshi, and M. Pollefeys. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. In *Int. Conf. Comput. Vis.*, pages 20270–20281, October 2023.

[324] X. Wang, S. Zhang, Z. Qing, Y. Shao, Z. Zuo, C. Gao, and N. Sang. Oadtr: Online action detection with transformers. In *Int. Conf. Comput. Vis.*, pages 7565–7575, 2021.

[325] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva. MultiON: Benchmarking semantic map memory using multi-object navigation. In *NeurIPS*, 2020.

[326] T. Wehrbein, M. Rudolph, B. Rosenhahn, and B. Wandt. Probabilistic monocular 3d human pose estimation with normalizing flows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11179–11188, 2021.

[327] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. hsin Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

[328] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[329] J. W. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou, and T. Ma. Larger language models do in-context learning differently. *ArXiv*, abs/2303.03846, 2023.

[330] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra. Embodied Question Answering in Photorealistic Environments with Point Cloud Perception. In *CVPR*, 2019.

[331] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2019.

[332] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[333] D. Xie, T. Shu, S. Todorovic, and S.-C. Zhu. Learning and inferring "dark matter" and predicting human intents and trajectories in videos. volume 40, pages 1639–1652, 2018.

[334] K. Xie, T. Wang, U. Iqbal, Y. Guo, S. Fidler, and F. Shkurti. Physics-based human motion estimation and synthesis from videos. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11512–11521, 2021.

[335] S. Xu, Y.-X. Wang, and L.-Y. Gui. Diverse human motion prediction guided by multi-level spatial-temporal anchors. In *European Conference on Computer Vision (ECCV)*, 2022.

[336] J. Xue, J. Fang, T. Li, B. Zhang, P. Zhang, Z. Ye, and J. Dou. Blvd: Building a large-scale 5d semantics benchmark for autonomous driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6685–6691. IEEE, 2019.

[337] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *CVPR 2011*, pages 1345–1352, 2011.

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

[339] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

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

[341] M. Yang and E. Isufi. Convolutional learning on simplicial complexes. *arXiv preprint arXiv:2301.11163*, 2023.

[342] J. Ye, D. Batra, A. Das, and E. Wijmans. Auxiliary Tasks and Exploration Enable ObjectNav. In *ICCV*, 2021.

[343] J. Ye, D. Batra, E. Wijmans, and A. Das. Auxiliary Tasks Speed Up Learning PointGoal Navigation. In *Conference on Robot Learning*, 2020.

[344] V. Ye, G. Pavlakos, J. Malik, and A. Kanazawa. Decoupling human and camera motion from videos in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[345] X. Yi, Y. Zhou, M. Habermann, V. Golyanik, S. Pan, C. Theobalt, and F. Xu. Egolocate: Real-time motion capture, localization, and mapping with sparse body-mounted sensors. *ACM Transactions on Graphics (TOG)*, 42:1 – 17, 2023.

[346] N. Yokoyama, Q. Luo, D. Batra, and S. Ha. Learning Robust Agents for Visual Navigation in Dynamic Environments: The Winning Entry of iGibson Challenge 2021. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[347] H. P. Young. Condorcet's theory of voting. *American Political science review*, 82(4):1231–1244, 1988.

[348] Y. Yuan and K. Kitani. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10082–10092, 2019.

[349] Y. Yuan, S.-E. Wei, T. Simon, K. Kitani, and J. M. Saragih. Simpoe: Simulated character control for 3d human pose estimation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7155–7165, 2021.

[350] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[351] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9793–9803, 2021.

[352] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021.

[353] J. Yue, D. Manocha, and H. Wang. Human trajectory prediction via neural social physics. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 376–394. Springer, 2022.

[354] M. Z. Zaheer, A. Mahmood, M. H. Khan, M. Segu, F. Yu, and S.-I. Lee. Generative cooperative learning for unsupervised video anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14744–14754, 2022.

[355] A. Zeng, A. S. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. S. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. R. Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *ArXiv*, abs/2204.00598, 2022.

[356] E. Zhan, S. Zheng, Y. Yue, L. Sha, and P. Lucey. Generating multi-agent trajectories using programmatic weak supervision. 03 2018.

[357] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[358] J. Zhang, Y. Zhang, X. Cun, S. Huang, Y. Zhang, H. Zhao, H. Lu, and X. Shen. Generating human motion from textual descriptions with discrete representations. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14730–14740, 2023.

[359] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *ArXiv*, abs/2208.15001, 2022.

[360] S. Zhang, X. Liu, and J. Xiao. On geometric features for skeleton-based action recognition using multilayer lstm networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 148–157, 2017.

[361] S. Zhang, Q. Ma, Y. Zhang, S. Aliakbarian, D. P. Cosker, and S. Tang. Probabilistic human mesh recovery in 3d scenes from egocentric views. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7955–7966, 2023.

[362] S. Zhang, Q. Ma, Y. Zhang, Z. Qian, T. Kwon, M. Pollefeys, F. Bogo, and S. Tang. Egobody: Human body shape and motion of interacting people from head-mounted devices. In *European Conference on Computer Vision*, 2021.

[363] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn. Magnet: A neural network for directed graphs. In *Advances in Neural Information Processing Systems*, 2021.

[364] X. Zhang, Y. Wu, H. Wang, F. Iida, and L. Wang. Adaptive locomotion learning for quadruped robots by combining drl with a cosine oscillator based rhythm controller. *Applied Sciences*, 13(19), 2023.

[365] Y. Zhang, M. J. Black, and S. Tang. We are more than our joints: Predicting how 3d bodies move. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3372–3382, 2021.

[366] Y. Zhang, H. Chen, S. L. Waslander, J. wei Gong, G. ming Xiong, T. Yang, and K. Liu. Hybrid trajectory planning for autonomous driving in highly constrained environments. *IEEE Access*, 6:32800–32819, 2018.

[367] Y. Zhang and S. Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20481–20491, 2022.

[368] Z. Zhang, A. Zhang, M. Li, and A. J. Smola. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493, 2022.

[369] Q. Zhao, C. Zhang, S. Wang, C. Fu, N. Agarwal, K. Lee, and C. Sun. Antgpt: Can large language models help long-term action anticipation from videos? *ArXiv*, abs/2307.16368, 2023.

[370] S. Zheng, Y. Yue, and P. Lucey. Generating long-term trajectories using deep hierarchical networks. 06 2017.

[371] X. Zheng, Z. Su, C. Wen, Z. Xue, and X. Jin. Realistic full-body tracking from sparse observations via joint-level modeling. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14632–14642, 2023.

[372] Y. Zheng, Y. Yang, K. Mo, J. Li, T. Yu, Y. Liu, K. Liu, and L. J. Guibas. Gimo: Gaze-informed human motion prediction in context. In *European Conference on Computer Vision*, 2022.

[373] Y. Zhong, L. Yu, Y. Bai, S. Li, X. Yan, and Y. Li. Learning procedure-aware video representation from instructional videos and their narrations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2023.

[374] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[375] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4966–4975, 2015.

[376] D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. In *CVPR*, 2019.

Chapter 12

Appendix

A.1 Internation Conferences and Workshops

- *"Staged Contact-Aware Global Human Motion Forecasting."*
  Scofano, Luca*, Alessio Sampieri*, Elisabeth Schiele, Edoardo De Matteis, Laura Leal-Taix'e and Fabio Galasso. British Machine Vision Conference (BMVC) 2023.

- *"Best Practices for 2-Body Pose Forecasting."*
  Rahman Muhammad Rameez Ur*, **Luca Scofano***, Edoardo De Matteis, Alessandro Flaborea, Alessio Sampieri and Fabio Galasso. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2023): 3614-3624.

- *"PREGO: Online Mistake Detection in PRocedural EGOcentric Videos."*
  Flaborea, Alessandro*, Guido Maria D'Amely di Melendugno*, Leonardo Plini, **Luca Scofano**, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella and Fabio Galasso. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024): 18483-18492.

A.2 Journals

- *"ICML 2023 Topological Deep Learning Challenge: Design and Results."*
  Papillon, M., Hajij, M., Frantzen, F., Hoppe, J., Jenne, H., Mathe, J., Myers, A., Papamarkou, T., Schaub, M.T., Zamzmi, G., Birdal, T., Dey, T.K., Doster, T., Emerson, T.H., Gopalakrishnan, G., Govil, D., Grande, V.P., Guzm'an-S'aenz, A., Kvinge, H., Livesay, N., Meisner, J., Mukherjee, S., Samaga, S.N., Ramamurthy, K.N., Karri, M.R., Rosen, P., Sanborn, S., Scholkemper, M., Walters, R., Agerberg, J., Bokman, G., Barikbin, S., Battiloro, C., Bazhenov, G., Bernardez, G., Brent, A., Escalera, S., Fiorellino, S., Gavrilev, D., Hassanin, M., Hausner, P., Gardaa, O.H., Khamis, A., Lecha, M., Magai, G., Malygina, T., Melnyk, P., Ballester, R., Nadimpalli, K.V., Nikitin, A., Rabinowitz, A., Salatiello, A., Scardapane, S., **Scofano, L.**, Singh, S., Sjolund, J., Snopov, P., Spinelli, I., Telyatnikov, L., Testa, L., Yang, M., Yue, Y., Zaghen, O., Zia, A., & Miolane, N. (2023). TAG-ML.

- *"About latent roles in forecasting players in team sports."*
  Scofano, Luca, Alessio Sampieri, Giuseppe Re, Matteo Almanza, Alessandro Panconesi and Fabio Galasso. Neural Processing Letters 56 (2), 1-12

A.3 Under Submission

- *"Following the Human Thread in Social Navigation."*
  Scofano, Luca*, Alessio Sampieri*, Tommaso Campari, Valentino Sacco, Indro Spinelli, Lamberto Ballan and Fabio Galasso. ArXiv abs/2404.11327 (2024): n. pag.

- *"Leveraging LLMs with Chain of Thought and In-Context Learning for Mistake Detection in Procedural Egocentric Videos."*
  Leonardo Plini*, **Luca Scofano***, Edoardo De Matteis*, Andrea Sanchietti, Flaborea, Alessandro, Guido Maria D'Amely di Melendugno, Antonino Furnari, Giovanni Maria Farinella and Fabio Galasso.

- *"Social EgoMesh Estimation."*
  Scofano, Luca*, Alessio Sampieri*, Edoardo De Matteis, Indro Spinelli and Fabio Galasso.

- *"TopoX: A Suite of Python Packages for Machine Learning on Topological Domains."*
  Hajij, Mustafa, Mathilde Papillon, Florian Frantzen, Jens Agerberg, Ibrahem AlJabea, Rubén Ballester, Claudio Battiloro, Guillermo Bern'ardez, Tolga Birdal, Aiden Brent, Peter Chin, Sergio Escalera, Odin Hoff Gardaa, Gurusankar Gopalakrishnan, Devendra Govil, Josef Hoppe, Maneel Reddy Karri, Jude Khouja, Manuel Lecha, Neal Livesay, Jan Meissner, Soham Mukherjee, Alexander Nikitin, Theodore Papamarkou, Jaro Pr'ilepok, Karthikeyan Natesan Ramamurthy, Paul Rosen, Aldo Guzm'an-S'aenz, Alessandro Salatiello, Shreyas N. Samaga, Michael T. Schaub, **Luca Scofano**, Indro Spinelli, Lev Telyatnikov, Quang Truong, Robin Walters, Maosheng Yang, Olga Zaghen, Ghada Zamzmi, Ali Zia and Nina Miolane. ArXiv abs/2402.02441 (2024): n. pag.