Human-Pose Prediction for Low-Latency Mixed-Presence Collaboration

Divyendu Dutta

December 01, 2024

Technische Universität Dresden

Interactive Media Lab Dresden
Institute of Software and Multimedia Technology
Faculty of Computer Science

Master’s Thesis

Human-Pose Prediction for Low-Latency Mixed-Presence Collaboration

Divyendu Dutta

Supervisor Dr.-Ing. Wolfgang Büschel
Interactive Media Lab Dresden
Technische Universität Dresden

First Reviewer Prof. Dr.-Ing. Raimund Dachselt
Interactive Media Lab Dresden
Technische Universität Dresden

Second Reviewer Dr.-Ing. Annett Mitschick
Interactive Media Lab Dresden
Technische Universität Dresden

December 01, 2024

Divyendu Dutta
Human-Pose Prediction for Low-Latency Mixed-Presence Collaboration
Master's Thesis, December 01, 2024
First Reviewer: Prof. Dr.-Ing. Raimund Dachselt
Second Reviewer: Dr.-Ing. Annett Mitschick
Supervisor: Dr.-Ing. Wolfgang Büschel

Technische Universität Dresden
Faculty of Computer Science
Institute of Software and Multimedia Technology
Interactive Media Lab Dresden
Nöthnitzer Str. 46
D-01062 and Dresden

Abstract

Mixed Presence Collaboration (MPC) systems facilitate seamless interaction between co-present and remote participants but face challenges such as presence disparity, where collaborators' sense of each other's presence varies. Immersive displays like AR and VR address this by offering embodied representations of remote collaborators, enhancing shared space and co-presence. A key enabler of this is real-time human motion tracking, which captures gestures and movements for lifelike interaction. However, tracking technologies often introduce latency, which combined with the inherent networking delays in distributed systems like MPC systems, disrupts natural interactions and diminishes collaboration effectiveness.

This master's thesis aims to address the aforementioned latency issue in MPC systems by integrating a 3D human pose prediction model, specifically the STS-GCN model. It begins with an exploration of the literature and the related work. The focus then shifts to the design, architecture and technical implementation of integrating the pose prediction model into an existing MPC system. This is followed by an evaluation of the prediction accuracy of the integrated pose prediction model through both qualitative visualizations and quantitative metrics. The model's performance is compared against the zero-velocity baseline model. The results indicate that while the model currently lags behind the zero-velocity baseline, it shows promise. With further fine-tuning, it could achieve better accuracy, particularly for dynamic and complex movements. Although the reduction in latency was not empirically tested, the design provides a theoretical contribution by offering a framework for future testing and validation of latency improvements in real-time MPC systems.

Finally, based on the work done in this master's thesis to utilize a pose prediction model with live tracking data for real-time applications, valuable insights are provided that serve as a foundation for future research on using pose prediction to mitigate latency in MPC systems.

Acknowledgement

First and foremost, I express my gratitude to my supervisor and advisor, Dr.-Ing. Wolfgang Büschel, who has spent valuable hours from his workday helping me grow from a novice student to a capable researcher. His focus on detail and accuracy has in turn helped me hone these qualities for myself. When discussing ideas and observations, his advice and comments helped me figure out and correct many of my mistakes. I am forever grateful for his time and knowledge.

I am deeply thankful to Prof. Dr.-Ing. Raimund Dachselt for providing me with the opportunity to undertake this exceptional research work and for granting me access to the resources of the Interactive Media Lab, which was instrumental in completing this master's thesis. I also extend my sincere gratitude to Prof. Dr.-Ing. Raimund Dachselt and Dr.-Ing. Annett Mitschick for kindly agreeing to review this thesis. Furthermore, I am immensely appreciative of Dipl.-Ing. Ricardo Langner's "Clean Thesis," which served as an invaluable resource in the preparation of this manuscript. Finally, I highly appreciate Mr. Saeed Saadatnejad's and Mr. Alessio Sampieri's clarification and insight about their research work.

Contents

1 Introduction 1

1.1 Background and Context 1

1.2 Problem Description and Motivation 2

1.3 Masters Thesis Structure 3

2 Background and Related Work 5

2.1 Mixed Presence Collaboration (MPC) 5

2.1.1 Role of Live Human Tracking for Embodied Representations in Collaboration 5

2.1.2 Role of Augmented Reality (AR) in MPC systems 8

2.2 Human Computer Interaction (HCI) and Mixed Presence Collaboration (MPC) 8

2.3 Short-Term Human Pose Prediction 9

2.3.1 Overview of Various Human Pose Prediction Models 10

2.4 Live Human Motion Tracking and Human Pose Prediction 11

2.4.1 Latency challenge in Live Human Motion Tracking 12

2.4.2 Potential Solutions to Mitigate Latency in Motion Tracking . . 12

2.5 Selection of Appropriate Human Pose Prediction Model 13

2.5.1 Latency and Accuracy 13

2.5.2 Autoregressive vs Non Autoregressive Models 13

3 Architectural Design and Implementation Details 15

3.1 Overview of the Existing Mixed Presence Collaboration (MPC) System 15

3.2 Pose Data Recording and Replay for Study Replicability 17

3.3 Integration of Pose Prediction Model in the Existing Mixed Presence Collaboration (MPC) System 19

3.3.1 Overview of the Selected Pose Prediction Model 20

3.3.2 Design Considerations 21

3.3.3 End-to-End Architecture 22

3.3.4 Implementation of the Inference Server for Pose Prediction . . 24

3.3.5 Detailed Workflow 29

3.4 Visualization and Evaluation 30

3.4.1 Implementation of Visualization 31

3.4.2 Implementation of Model Comparison and Precision Evaluation 33
3.5 Challenges Faced and Solutions 35

4 Results 39
4.1 Visualization Results 39
4.1.1 Visualization of Predicted Poses from Live Tracked Data . . . 40
4.1.2 Visualization of Predicted Poses from Saved Pose Data 41
4.2 Model Comparison and Precision Evaluation Results 42
4.2.1 Comparison of Models using Live Tracked Data 42
4.2.2 Comparison of Models and Precision Evaluation using Saved Pose Data 43
4.3 Limitations of the Integrated System 48
4.3.1 Limitation in Predicting Moving Poses in Space 48
4.3.2 Multiple "HololensClient" and "KinectServer" Support Challenges 51
4.3.3 Proposed Latency Testing Approach for the Integrated System 51

5 Conclusion 53
5.1 Summary of Contributions 53
5.2 Key Findings and Insights 55
5.3 Future Work 57

Bibliography 59

List of Figures 65

List of Tables 67

Declaration 71

Introduction 1

This chapter lays the foundation for the work presented in this master's thesis. It begins by providing the necessary background and context, situating the work within its broader academic and practical landscape. Next, it outlines the key challenge and the driving force behind this work, emphasizing its significance. Finally, an overview is presented, offering a roadmap to guide readers through the subsequent chapters.

1.1 Background and Context

With the growing accessibility of advanced technologies, Mixed Presence Collaboration (MPC) is emerging as a transformative tool for enabling seamless teamwork across the world. MPC bridges the gap between co-present and remote participants, creating hybrid environments that allow collaboration in ways traditional systems cannot.

The core strength and therefore challenge of MPC systems lies in their ability to integrate co-located and remote interactions effectively. These systems are central to the field of Human-Computer Interaction (HCI), where they serve as a valuable platform for studying how people collaborate and how technology influences group dynamics. Researchers in HCI use MPC to investigate interaction patterns, design principles, and interface innovations that support fluid communication and coordination across different physical and virtual settings.

MPC systems are implemented through various technologies, such as video conferencing, display-sharing tools, telepresence systems, and virtual environments. However, they often encounter significant hurdles, including:

1. **Display disparity**, where different devices or screen orientations lead to fragmented perspectives among collaborators.
2. **Presence disparity**, where co-located participants experience a stronger sense of presence compared to their remote counterparts.

The use of immersive displays, such as augmented reality (AR) and virtual reality (VR), offers a promising solution to the presence disparity challenge. AR/VR can create a shared virtual space, providing embodied representations of remote participants and fostering a deeper sense of presence and engagement. These advancements pave the way for MPC systems that feel more natural and unified, enhancing collaboration for all participants.

One critical technology that enables this immersive experience is live human motion tracking, which captures the gestures, movements, and physical presence of users. By integrating real-time motion data, MPC systems can provide lifelike, synchronized representations of participants, significantly improving the sense of co-presence and enhancing the collaborative experience.

1.2 Problem Description and Motivation

Despite the promise of MPC systems, a critical challenge lies in achieving accurate and responsive motion tracking to support real-time interaction. Technologies such as Microsoft Kinect are commonly used to track users' movements, capturing data for creating embodied representations within shared virtual spaces. However, these systems often suffer from inherent latency, where delays occur between the moment a user's motion is tracked and the point at which the data is processed and displayed. This latency issue is further amplified in MPC systems, where network delays exacerbate the problem. The result is a noticeable disconnect: motions appear out of sync, gestures are misinterpreted, and the natural flow of interaction is disrupted. Such challenges diminish the sense of presence and embodiment, which are critical for effective collaboration in hybrid environments. Misaligned or delayed motion data can hinder the interpretation of nonverbal cues, obstruct coordinated activities, and ultimately reduce the overall effectiveness of the collaborative experience.

The aspiration to address this latency problem in MPC systems forms the core motivation for this master's thesis. To mitigate latency and enhance real-time interaction, this work explores the integration of a human pose prediction model capable of predicting future poses based on live-tracked motion data. While advancements have been made in both MPC systems and 3D human pose prediction models independently, their integration remains largely unexplored. Bridging this gap presents a unique opportunity to enhance the responsiveness and usability of MPC systems, ensuring that they support lifelike, dynamic collaboration across diverse settings.

This work aims to fill this void and hence pave the way for more seamless and immersive collaborative experiences.

1.3 Masters Thesis Structure

This master's thesis is organized as follows:

Chapter 2

This chapter provides a comprehensive exploration of Mixed Presence Collaboration (MPC), focusing on the importance of live human tracking and augmented reality (AR) in enhancing remote and co-located interactions. It reviews existing MPC systems within Human-Computer Interaction (HCI) research and examines short-term human pose prediction models for real-time applications. Challenges in live human motion tracking, such as latency, are discussed alongside solutions for improving such latency. The chapter concludes by discussing the basis for selecting an appropriate human motion prediction model for use in a real-time setting.

Chapter 3

This chapter details the important elements and practical execution of a 3D human pose prediction model integrated into a Mixed Presence Collaboration (MPC) system. It begins by providing an overview of an existing MPC system. Next, it explores a feature implemented to enhance study replicability by allowing pose data to be recorded and replayed. The focus then shifts to the core of this work, describing how a 3D human pose prediction model was seamlessly incorporated into the MPC system to enhance it. This is followed by a description of how pose data was rendered and visualized followed by assessments done to determine the effectiveness of the integrated model. Finally, the chapter concludes by reflecting on obstacles encountered during development and the strategies employed to overcome them.

Chapter 4

This chapter describes the results derived from the integration and evaluation of the pose prediction model within the existing Mixed Presence Collaboration (MPC) system. It aims to provide insights into the integrated system's performance and limitations through visualization, precision evaluation, and model comparison. It begins by describing the qualitative results obtained through the visual representation of pose data. Then it describes the quantitative results based on metrics such as displacement error. These results compare the accuracy of the pose prediction model against the zero-velocity baseline, offering an evaluation of the model's predictive capabilities. Finally, the chapter concludes by exploring the boundaries of the current implementation, identifying areas where the system's performance may fall short and provides insights into potential future improvements.

Chapter 5

This chapter concludes the master's thesis by reflecting on its core contributions, key findings, and potential future directions. It begins by providing an overview of the primary contributions made through this research. Then it highlights the significant results and lessons learned, offering a deeper understanding of the implications of the work. Finally, the chapter concludes by outlining promising directions for extending and enhancing this work, paving the way for further exploration and innovation.

2 Background and Related Work

This chapter explores Mixed Presence Collaboration (MPC) and its significance in Human-Computer Interaction (HCI), highlighting how MPC bridges remote and co-located interactions. It examines short-term human pose prediction and live human motion tracking, addressing challenges like latency in motion prediction. Finally, it discusses the criteria for selecting an optimal human pose prediction model, balancing latency and accuracy for effective real-time collaboration in MPC systems.

2.1 Mixed Presence Collaboration (MPC)

This section introduces Mixed Presence Collaboration (MPC), emphasizing its role in facilitating interaction between remote and co-located participants. It highlights live human motion tracking as a key component, enabling realistic and dynamic representations of participants' movements, which enhance the sense of presence and engagement. Additionally, the role of augmented reality (AR) is discussed, showcasing how immersive displays can bridge physical and virtual spaces, making collaboration more intuitive and effective for all users.

2.1.1 Role of Live Human Tracking for Embodied Representations in Collaboration

Below, an overview of motion capture technology is provided, followed by an exploration of its specific applications for creating rich, embodied representations of users in collaborative environments.

Motion Capture (MoCap) in the Context of Live Human Tracking

Full-body motion capture (MoCap) involves monitoring and recording the movements of humans and objects using various technologies. These technologies are generally classified into optical and non-optical systems. Non-optical systems commonly utilize Inertial Measurement Units (IMUs), while optical systems rely on cameras, including depth cameras, for tracking. Optical systems can further be divided into marker-based and marker-less categories, depending on whether the tracked individual needs to wear specialized devices for motion tracking [ZL16].

MoCap sees a wide range of use cases such as in robotics [YH09], computer animation [@Rap14], and medicine [AN04] among others. When this is applied specifically to capture the motion of humans and is achieved in real-time, we get live human tracking. For it to be considered "live" or "real-time", such a method or system needs to "respond in a reasonable time or before a deadline, in a reliable manner" [ZL16].

Overview of Live Human Tracking used in MPC systems

Mixed presence collaboration combines physical and virtual environments to enable remote and co-located users to interact as though they are present in the same space. For effective collaboration in MPC systems, it's important to represent each user accurately, reflecting their gestures, pose, and orientation in real-time. Recent research efforts that incorporate live human motion tracking within collaborative systems are explored below. While some of these systems may not implement or fully support every aspect of mixed presence collaboration, they effectively capture its core principles or can be extended to do so. They are included here because the primary focus is to investigate how live human motion tracking contributes to creating embodied representations of participants' movements, thereby improving the sense of presence and engagement in collaborative environments.

[NYW15] introduce a system for mixed reality collaboration, which allows a local user to interact and collaborate with a remote user using natural hand motion. The remote user appears as a virtual avatar to the local user ie, avatar-mediated remote collaboration. They establish a structured design framework that includes the configuration of both hardware and software across multiple devices. Furthermore, they compare avatar representation with and without full body tracking. They use a Kinect for full body tracking but mention that due to the limited hand-tracking of Kinect, use LeapMotion for full articulated hand-tracking. The tracking data from the two sources is then fused so that the final result looks natural when viewed from the local or remote side. On the flip side, [HBY22] propose a VR remote collaboration system which enhances immersion via motion capture. They note that VR technology can enhance the effects of collaboration via information sharing, a sense of presence that is more immersive, and natural interactions. But existing VR systems employ an inverse kinematic method resulting in the user and character not being accurately matched. Hence the authors propose a method to improve the IK method to synchronize the user and the character via full-body motion capture. Specifically, they utilize inertial measurement units (IMUs) for capturing indoor and outdoor motions. This provides precise tracking information but has a high setup cost.

In the BEAMING (Being in Augmented Multimodal Naturally Networked Gatherings) project by [Oye+13], the authors allow a single user (referred to as the visitor) to be virtually transported to a remote real-world location. This is achieved via an advanced virtual reality system, known as the transporter. The visitor's presence is tracked through various methods, including both motion capture and emotion tracking technologies. They comment that their goal is to create a sense of co-presence for the local participants, so that visitors are represented through either a virtual or physical presence at the destination. [Cop+24] explore large interactive surfaces, specifically wall-sized displays for collocated and remote collaboration. They approach the problem of transmitting awareness information of remote collaborators using a multimodal pipeline and visualizing such non-verbal information through what they refer to as workspace awareness cues, across wall-sized displays placed at distant locations. Their system performs body tracking via the Kinect sensor. While their system does not create a full-body embodied representation of the remote user, they select certain joints of interest from the body tracking data to compute awareness information.

[CBL24] present a collaborative MR system called CoMR-MoCap which allow actors to easily rehearse scenes. Through the use of a Video See-Through Head Mounted Display (VST-HMD), actors can view digital avatars of other performers wearing motion-capture suits alongside real-time digital scene elements. This system facilitates collaboration by allowing multiple actors to wear both mocap suits for character animation and VST-HMDs for a synchronized view of digital content in the shared space. The results showed that their system enhanced usability, spatial and social presence, and embodiment over other methods.

2.1.2 Role of Augmented Reality (AR) in MPC systems

Augmented reality (AR) technology offers a broader range of applications compared to traditional virtual reality (VR) and as such there are a wide range of application fields in which AR can be utilized. AR has two unique features. First is that the virtual space is seamlessly merged with the physical one. The other is that the virtual space as well as the physical one reacts to the physical actions of physical participants. Moreover, prior research shows that AR headsets can be used to provide enhanced face-to-face collaboration or remote collaboration [Ohs+98; Kim+14]. This makes their usage suitable for MPC systems which require both face-to-face and remote collaboration.

[Nor+19] present an MPC system with two local users and one remote user using AR headsets. Their primary aim was to explore AR based MPC systems due to the lack of exploration of MPC with AR interfaces. Their system uses the Microsoft Hololens as the AR headset and was developed using Unity with the Vuforia plugin and Microsoft's Mixed Reality Tool Kit. Additionally, the system was designed to facilitate collaborative room layout planning, enabling users to discuss and experiment with various furniture arrangements on a virtual floor plan. They concluded that users found the system easy to learn, but noted issues with consistency and accuracy.

2.2 Human Computer Interaction (HCI) and Mixed Presence Collaboration (MPC)

This section explores some of the important research in the field of Human-Computer Interaction within the context of mixed presence collaboration, highlighting critical aspects such as awareness and the resolution of disparities in collaborative environments. The selected studies provide information about how HCI principles can be used to enhance user experience and effectiveness in these settings. By examining real-time interactions and spatial awareness challenges, these works highlight the importance of user-centered designs that account for diverse technical setups. Together, they showcase the role of HCI in bridging remote and co-located participants within mixed presence collaboration.

[SBR19] explore the effect of different levels of view congruence ie, the alignment of visual perspectives between remote and co-located participants in a mixed presence collaboration setup where collocated collaborators work around a tabletop display and remote collaborators wear a VR HMD to interact with a connected virtual environment giving them a 3D perspective. They found that when users share a more congruent view of a collaborative environment, it enhances awareness, helping them maintain a better sense of other collaborator's locations and actions in the space. Furthermore, this congruence strengthens co-presence as users are able to anticipate each other's actions and communicate more effectively. They conclude that designers of mixed presence collaboration environments have flexibility in mapping remote and collocated experiences but also need to consider tradeoffs when creating mixed reality configurations for mixed presence collaboration.

[TBG05] study mixed presence collaboration systems and note two problems such systems face. First is display disparity, where connecting different display types causes problems in the seating arrangement of collaborators. And the second is presence disparity, where co-located and remote users percieve the presence of other collaborators differently. They posit that presence disparity is caused by incomplete effective communication between remote participants which then lead to problems in group collaborative and communication dynamics. Furthermore, the authors create their own implementation of a mixed presence collaboration system which tries to solve the previously mentioned two issues. Their implementation uses techniques such as adding digital arm shadows as a rich embodiment of presence. [MRM07] also investigate the issue of presence disparity by developing a model to structure and represent their understanding of awareness in mixed presence collaboration systems. Their end goal is to enhance this model to help designers create awareness tools for mixed presence scenarios.

2.3 Short-Term Human Pose Prediction

A fundamental human skill that enables us to navigate and interact effectively with our environment is the ability to anticipate the actions of others in the immediate future. For machines, this capability is captured through 3D human pose prediction, which involves forecasting the most likely future movement poses of a person based on their past motion data. In this context, the previously observed poses, used as input, are referred to as the "seed sequence," while the predicted future poses are termed the "target sequence." This process is essential for developing systems that can interpret and respond to human behavior dynamically.

This task of pose prediction can further be classified into two categories, short-term or long-term, based on the length of the target sequence being aimed for. Furthermore, such a task is typically conceived as a generative modeling task. Generative modeling is a subfield of machine learning that focuses on enabling machines to imagine and synthesize new substances [Lam21]. This in itself is a difficult task, more so for pose prediction since it requires combining spatial and temporal information from the seed sequence to predict the future poses.

Human pose prediction has a wide range of applications such as in robotics [Che+18; Wag+18; Wen+16], human-robot interaction [BKK18; LS17; KSS16], autonomous navigation [Man+20], computer graphics [Lev+12; KGP08] and 3D human/object tracking [Gon+11; UFF06].

The following provides a concise summary of current advancements in the field of human pose prediction. This area of research focuses on forecasting human movements by leveraging past motion data and enabling applications in robotics, mixed reality, gaming, and human-computer interaction. Both, autoregressive and non-autoregressive approaches for 3D human pose prediction are reviewed.

2.3.1 Overview of Various Human Pose Prediction Models

Traditionally, recurrent neural networks (RNNs) have been widely preferred for the task of human motion prediction, aiming to forecast 3D human movement due to their capability to capture temporal dependencies in sequential data [Tor+19; AKH19; PGA18]. [MBR17] show via empirical evidence that prevailing RNNs have difficulty obtaining good performance for long-term and short-term predictions. They suggest improvements to the existing RNN architectures but conclude that their new architecture is still unable to perform well.

[Mao+19; Sof+21; Dan+21] utilize a graph convolutional network (GCN) to accomplish the task of human motion prediction. In a similar vein, [Ma+22] propose a two-network system comprising of a spatial dense GCN and a temporal dense GCN which operate alternatively. [Wan+23] take GCNs for human pose prediction to the next step. They propose "Universal Graph Convolution (UniGC)", which unifies various graph convolution designs under a single framework, treating them as special cases. Their aim with UniGC is to allow the discoverability of other new graph convolutions. [Aks+21; Saa+24; MVO21] utilize the transformer architecture to forecast human poses due to the recent success of transformer models for text generation. These works illustrate the versatility of transformer models, showcasing their ability to be applied in either autoregressive or non-autoregressive modes. Interestingly, [Li+23] propose a hybrid framework fusing autoregressive and non-autoregressive methods for human pose prediction.

[Saa+23] propose a diffusion model that frames the prediction of future human poses as a denoising problem. The main idea here is to address the challenge of occlusions during 3D human pose prediction. Furthermore, they show that their model can be used as a black box in combination with other pose prediction models. However, a major disadvantage of diffusion models is their slow generative process. To enable diffusion models to be used in real-time for pose prediction, [TZL24] propose a two step process. First, they simplify a pretrained diffusion-based motion predictor. Then to further speed up inference, they streamline the original denoiser by removing resource-intensive components. They conclude that compared to prior diffusion based motion prediction methods, their model significantly reduces inference time. This allows them to achieve real-time prediction without a noticeable reduction in accuracy.

Looking at more recent work done in this field, [Zha+24] realize that changes in the human skeletal size affect the data distribution and can therefore lead to poor performance during inference. The authors' proposed framework to solve this problem combines geometric encoding and neural networks to achieve size-invariant, high-quality motion prediction. [Liu+24] focus on short-term human motion prediction ie, within timeframes of less than 400 ms, along with improving its efficiency. They propose a new transformer-based method called "Joint-Aware Transformer (JAT)" which captures inter-joint correlations from both dynamic and static information. [Zai+24] propose a unique perspective that stands in contrast to other approaches for human pose prediction. They encode joint constraints through Lie algebra representation. And they chose to represent joint rotations this way to avoid the issues associated with other representations such as Euler angles or quaternions. [Wei+25] explicitly decouple the spatial and temporal contexts allowing them to learn from both complete and incomplete observed past motions. Hence their model is able to complete the incomplete motion sequences along with predicting future poses.

2.4 Live Human Motion Tracking and Human Pose Prediction

This section touches on the latency challenge inherent in live human motion tracking, particularly those introduced by motion tracking sensors when used in Mixed Presence Collaboration systems. Additionally, it emphasizes the critical role of human pose prediction models in mitigating this challenge, offering solutions that enhance real-time motion visualization.

2.4.1 Latency challenge in Live Human Motion Tracking

Motion tracking sensors introduce a notable delay between capturing a user's movements and making that data available for use in applications, such as animating a humanoid avatar or visualizing its motion on screen. Typically, this data arrives to us in the form of orientations and positions of various joints of the human body. In Mixed Presence Collaboration (MPC) systems, which rely on distributed architectures, network latency further compounds this delay as data must be transmitted between the server and clients for synchronization and rendering.

This delay presents a tradeoff: either frames are dropped to ensure the motion visualized on screen aligns with the real-time tracking, or all frames are retained, resulting in a noticeable lag in the displayed motion. Both these approaches are not ideal since most visual analysis applications would require visualizing motion without any dropped frames or much lag.

2.4.2 Potential Solutions to Mitigate Latency in Motion Tracking

To address the latency associated with Kinect sensors, one potential solution involves extracting raw RGBD data directly from the Kinect and processing it on the GPU to compute orientations and positions ourselves [@AA]. However, this approach is highly complex, requiring many pipeline steps to retrieve usable tracking data. Additionally, implementing such custom solutions gives up the ease and compatibility provided by the Kinect Body Tracking SDK, which seamlessly integrates with popular platforms like Unity.

Another solution is to utilize the previously mentioned 3D human pose prediction models, given that these models can predict future poses at a fast enough rate to combat the latency in motion tracking.

2.5 Selection of Appropriate Human Pose Prediction Model

Although significant advancements have been made in 3D human pose prediction, its application to live tracking data, such as that from motion tracking sensors, has seen limited exploration. There appears to be a gap in utilizing real-time tracked data within 3D human pose prediction, despite recent progress in predictive modeling techniques.

This section focuses on identifying a suitable model for integrating human pose prediction with live motion tracking systems. It emphasizes the importance of balancing latency and accuracy to ensure seamless, real-time performance. Additionally, it explores the differences between autoregressive and non-autoregressive models, analyzing their strengths and limitations.

2.5.1 Latency and Accuracy

Before utilizing the results of a pose prediction model, there are various factors of the model to be considered but the two major ones are latency and accuracy.

Latency is the amount of time required by the pose prediction model to predict one frame of the motion. This gives us valuable insight into selecting a suitable model depending on the amount of latency of the system that we are trying to combat. On the other hand, accuracy is one or more error metrics that inform us about how close the predicted frame is to the ground truth. Latency is measured in the millisecond range whereas different error metrics can have differing units. An appropriate model for this task should exhibit acceptable values of both latency and accuracy, since a model which predicts a single frame quickly but is not accurate is not of much use in a real-world scenario.

2.5.2 Autoregressive vs Non Autoregressive Models

Like most models that work on temporal data and predict future data from past data, 3D human pose prediction models can be either autoregressive or non-autoregressive in nature.

Autoregressive models break down the prediction of future frames into steps, where each step depends on the previous predictions [Aks+21]. Due to this, such models tend to accumulate errors over time until eventually, the predictions crash to an unlikely result. The implication is that these models are not very good at long-term predictions. Another disadvantage of these models is that due to their step-like prediction behavior, it is not possible to parallelize the prediction of the next N steps since they are generated sequentially one at a time.

Non-autoregressive models predict a specific number of frames in one go, ie. in parallel [Hig+21; MVO21]. But with these models, we have less control over the upper limit of the number of frames being predicted. For example, if such a model is trained to predict 25 frames, we can always use just 5 frames from the prediction and throw away the rest but we would be unable to predict 30 frames using this model unless we train another variant of the model which is specifically tailored to predict 30 frames.

Previous work done to evaluate various 3D human pose prediction models in terms of their latency and accuracy show that the STS-GCN model [Sof+21], which is a non-autoregressive model had the quickest inference time and competitive accuracy [DB24]. This makes it suitable for real-time applications.

3 Architectural Design and Implementation Details

This chapter details the important elements and practical execution of a 3D human pose prediction model integrated into a Mixed Presence Collaboration (MPC) system. It begins by providing an overview of an existing MPC system. Next, it explores a feature implemented to enhance study replicability by allowing pose data to be recorded and replayed. The focus then shifts to the core of this work, describing how a 3D human pose prediction model was seamlessly incorporated into the MPC system to enhance it. This is followed by a description of how pose data was rendered and visualized followed by assessments done to determine the effectiveness of the integrated model. Finally, the chapter concludes by reflecting on obstacles encountered during development and the strategies employed to overcome them.

3.1 Overview of the Existing Mixed Presence Collaboration (MPC) System

The existing mixed presence collaboration system focuses on enabling co-located and remote participants to collaborate in a shared space. The system aims to serve as a research platform that can be easily extended to accommodate different use cases, enabling quick prototyping of custom MPC systems tailored to specific user studies [Büs+24]. This approach is particularly advantageous, as developing a complete MPC system from scratch is often a complex and time-consuming process. This section highlights specific aspects and architectural details of the MPC system relevant to this master's thesis.

The system uses the Unity game engine and integrates Microsoft's Mixed Reality Toolkit (MRTK) to support augmented reality (AR) interactions using HoloLens 2 headsets. Furthermore, it uses the Microsoft Kinect to track and provide an embodied representation of the remote collaborators to the co-located participants of the system [Büs+24].

In the most minimalistic sense for our purpose, there are three components necessary to get the system running. They are the "KinectServer" component and two "HololensClient" components. The "KinectServer" component encapsulates the server role of the MPC system and has various important responsibilities in the system such as managing the state of the shared space used by the MPC system, including user sessions, room configurations, and avatar types. It is also responsible for accessing the Kinect to obtain body tracking data of one or more users. On the other hand, the "HololensClient" component encapsulates the client role of the MPC system and handles various responsibilities related to user management, room updates, and avatar synchronization. One of its key functionalities in the system is to adjust the position and orientation of remote users in the shared environment by utilizing the body tracking data received from the "KinectServer." This allows the system to accurately represent the movements and gestures of users, ensuring their avatars' actions align with their physical movements.

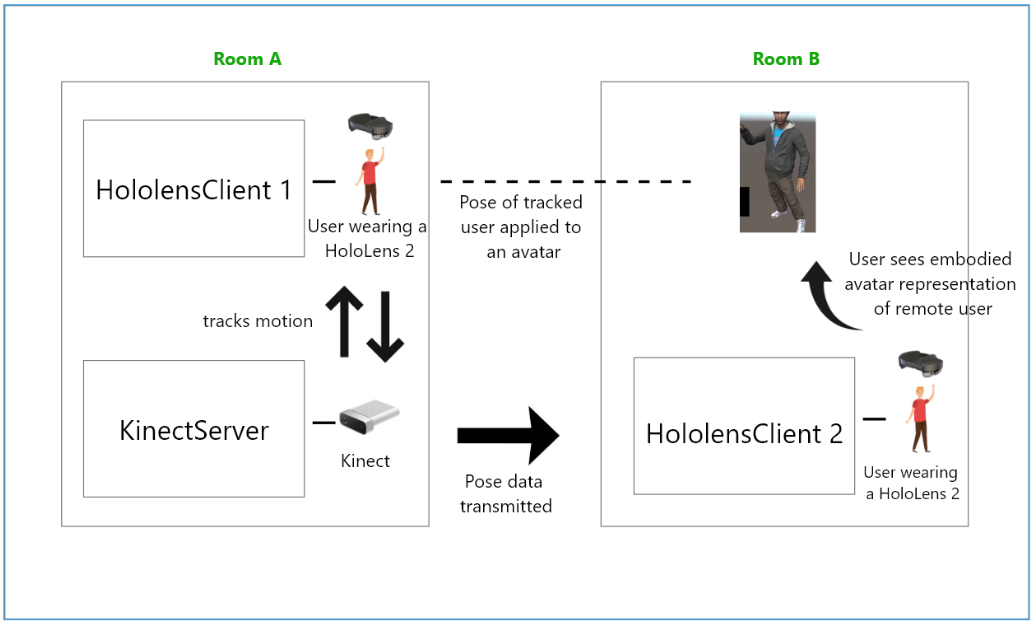

Room A
Room B
HololensClient 1
User wearing a HoloLens 2
Pose of tracked user applied to an avatar
User sees embodied avatar representation of remote user
tracks motion
KinectServer
Kinect
Pose data transmitted
HololensClient 2
User wearing a HoloLens 2

Fig. 3.1: Minimal depiction of the existing MPC system. One "KinectServer" component and two "HololensClient" components.

As shown in Figure 3.1, the KinectServer tracks the motion of a user (HololensClient 1) in "Room A" using the Kinect. The pose data from this user is then sent to a second user (HololensClient 2) located in a separate room, "Room B." The remote user in "Room B" views an embodied avatar of the user from "Room A" through the Hololens. This setup facilitates real-time remote collaboration, allowing the remote participant to experience the movement and actions of the local user as if they were physically present in a shared environment.

3.2 Pose Data Recording and Replay for Study Replicability

One of the key challenges of working with motion-tracking sensors, such as the Kinect, is the time-consuming process of development and debugging. This involves repeatedly moving in front of the sensor to track the user, obtaining pose data, and troubleshooting issues. The process becomes even more complex in mixed presence collaboration systems, which are distributed and involve multiple components. Additionally, since the Kinect's pose data can vary slightly each time it tracks motion, it further complicates debugging and development.

Hence, to enhance the testing and study replicability of the system, a mechanism for saving the tracked pose data from the Kinect is implemented. This data is then used in a custom class designed to mimic the Kinect sensor, allowing for the playback of recorded motions. By reusing this saved data, the system can simulate user movements for repeatable testing scenarios, enabling consistent evaluation across different stages of development and ensuring reliable study outcomes in MPC environments.

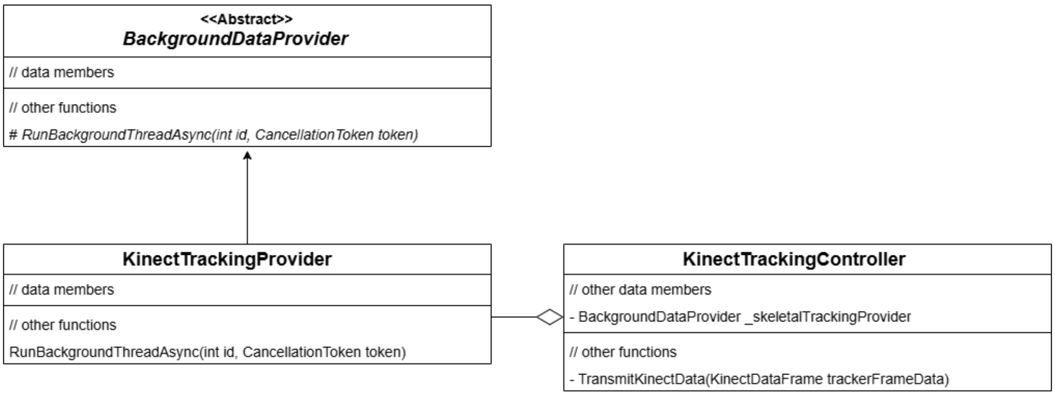

<<Abstract>>
BackgroundDataProvider
// data members
// other functions
# RunBackgroundThreadAsync(int id, CancellationToken token)
KinectTrackingProvider
// data members
// other functions
RunBackgroundThreadAsync(int id, CancellationToken token)
KinectTrackingController
// other data members
- BackgroundDataProvider _skeletalTrackingProvider
// other functions
- TransmitKinectData(KinectDataFrame trackerFrameData)

Fig. 3.2: UML diagram showing existing class relationships for Kinect data handling.

Figure 3.2 shows the existing relevant classes and their relationships for handling tracked pose data from the Kinect. All of these classes are part of the "KinectServer" component. The `RunBackgroundThreadAsync` method of the `KinectTrackingProvider` class is responsible for extracting the pose of the tracked user on a frame by frame basis from an actual Kinect. Now the `KinectTrackingController` class references the `KinectTrackingProvider` and transmits the pose data from the Kinect via the `TransmitKinectData` method to the relevant "HololensClient" component.

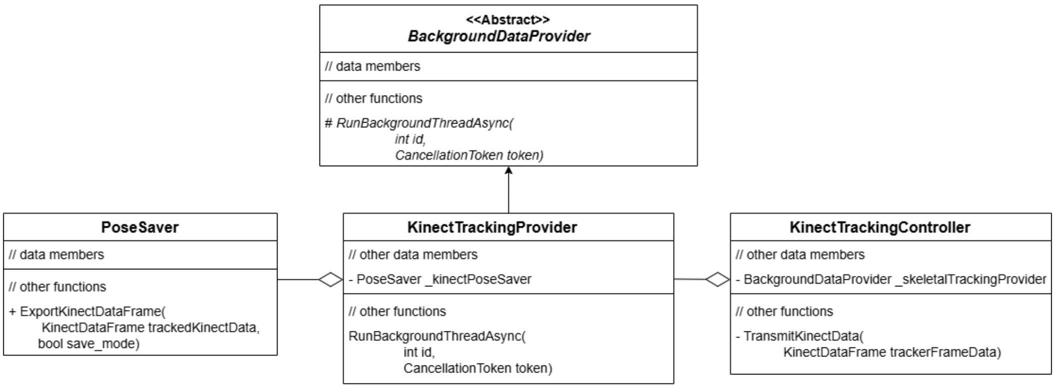


Fig. 3.3: UML diagram showing added pose data saving functionality.

The `ExportKinectDataFrame` method saves the tracked poses of a user from the Kinect to a JSON file or can print out joint information for quick debugging. Specifically, the positions of the 32 Kinect joints of all tracked bodies are saved for a specific number of frames. The exact number of frames to save is configurable within the script. For example, if 200 frames are saved, and since the Kinect operates at 30 FPS, this corresponds to approximately (1000/30) * 200 ms 6.7s of motion data. And the structure of data present in the JSON is `frames[bodies[joints[xyz position vector]]]`.

In saving mode, the method constructs this data structure to store pose data if at least one body is tracked. It tracks the number of frames processed and builds a record of pose data with each frame. As can be seen from figure 3.3, this method is then used in the `RunBackgroundThreadAsync` method of the `KinectTrackingProvider` class. So as the pose data per frame is being read from the Kinect, it can optionally also be saved to a JSON. The method ensures data integrity by carefully managing initialization, saving, and cleanup processes.

Figure 3.4 shows how the added class, `KinectJsonProvider` integrates with the existing MPC system. The `RunBackgroundThreadAsync` method is reimplemented to read the saved human pose data from a JSON. It does so frame by frame for all bodies so as to mimic the tracking behaviour of a Kinect. A 33ms delay is introduced between frames to simulate the Kinect frame rate. Enabling the feature to use replay pose data through the `KinectJsonProvider` eliminates the need for an actual Kinect during development. It is highly suitable for testing and simulation, as it replicates Kinect tracking behavior in a controlled manner using saved pose data. It includes robust error handling, metadata validation, and real-time emulation features, making it a versatile tool for Kinect applications. Overall, it simplifies the

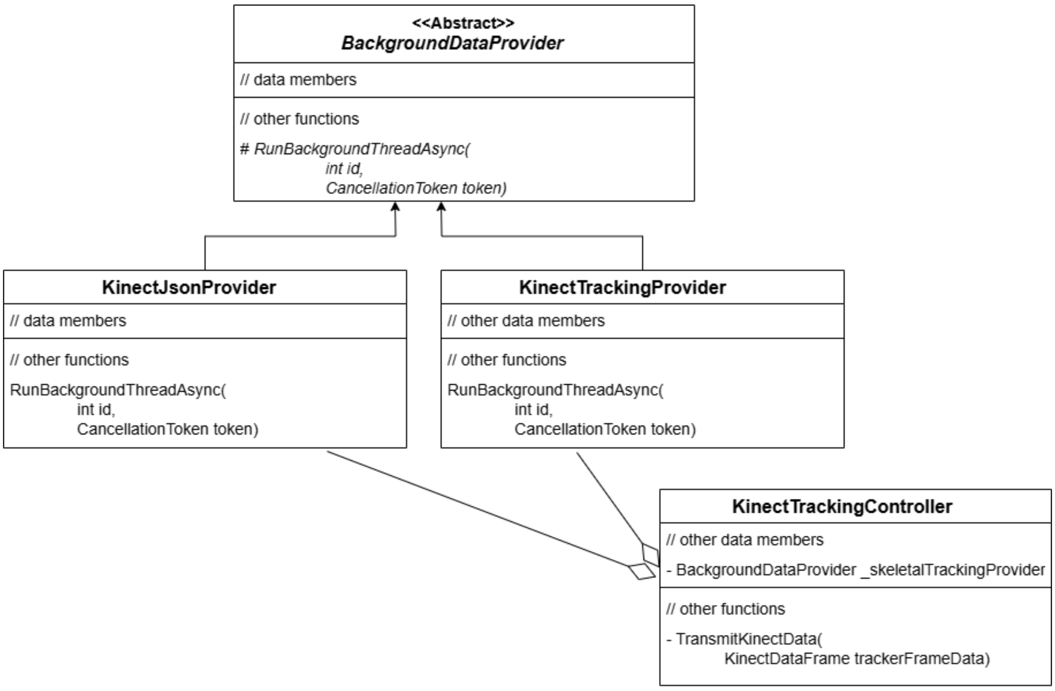


Fig. 3.4: UML diagram showing showing added pose replay integration into existing system.

process, making debugging and testing more efficient and less reliant on real-time motion tracking.

3.3 Integration of Pose Prediction Model in the Existing Mixed Presence Collaboration (MPC) System

This section describes the process of integrating a deep learning-based 3D human pose prediction model into an existing MPC system. The core concept is that the existing MPC system interfaces with the pose prediction model through a Python-based inference server. While this architecture is compatible with various pose prediction models with minimal changes, the first part of this section briefly covers key aspects of the specific model chosen for the implementation carried out in this master's thesis. Then it describes the general system design and architecture. The focus then shifts to the python based inference server's role in coordinating between the "KinectServer" and "HololensClient" components. Finally, it details how various components interact to achieve seamless integration.

3.3.1 Overview of the Selected Pose Prediction Model

Based on prior work, the **STS-GCN** 3D human pose prediction model was chosen due to it having the quickest inference time and competitive accuracy amongst various evaluated models [DB24]. It integrates both the temporal evolution of body pose and spatial joint interactions into a unified framework. Unlike traditional methods that treat these aspects separately, STS-GCN combines both into a single graph convolutional network (GCN) structure. This facilitates the exchange of information between motion and spatial correlations. It is a smaller model in terms of the number of parameters allowing it to train in a short period of time [Sof+21].

Furthermore, UnPOSed's implementation of the model was utilized [@Saa+]. UnPOSed provides an open-source framework for training and evaluating various human pose prediction models in Pytorch [Saa+24]. It supports various human pose datasets and also leverages the outcomes of prior open-source research efforts. Essentially, UnPOSed streamlines their training and evaluation processes within its framework.

The model was trained using the Archive of Motion Capture As Surface Shapes (AMASS) dataset which unifies multiple MoCap datasets. Furthermore, the AMASS dataset uses the SMPL (Skinned Multi-Person Linear) model to provide a unified representation of human motion. Motion capture data is retargeted to the SMPL body model, which represents the human body as a 3D mesh driven by pose and shape parameters [Mah+19]. Here the human pose is represented by the xyz positions of 18 joints in millimeters. A list of these joints can be found in the SMPL FAQ provided by the MPI for Intelligent Systems [@Int].

The following table outlines the key aspects of the STS-GCN model relevant to this master's thesis:

Model Property	**Value**	**Description**
Seed Sequence Size	10	No. of frames given to the model as input
Target Sequence Size	25	No. of future predicted frames given by the model
Type	Non-Autoregressive	Predicts future frames step-by-step, with each step relying on prior outputs

Tab. 3.1: Relevant key aspects of the STS-GCN model.

3.3.2 Design Considerations

Outlined below are the design considerations that guide the architecture and implementation of the integration, focusing on compatibility, performance, and scalability.

The pose prediction model could be executed locally on the same machine as either the "KinectServer" or the "HololensClient", but this approach introduces significant limitations in terms of flexibility and performance. Given that the pose prediction model is based on deep learning, it demands a lot of GPU resources for computation. Running this resource intensive model on the same machine as the "KinectServer" or the "HololensClient" would result in competition for machine resources, which is undesirable. The machine running the "KinectServer" is already responsible for managing the Kinect device itself, while the machine hosting the "HololensClient" is tethered to the HoloLens, adding another layer of complexity. This creates a situation where both components are competing for the same limited computational resources, which is particularly problematic for a system whose primary goal is to minimize latency. Resource contention between these components could lead to delays, reducing the system's overall responsiveness and affecting the user experience.

To avoid these issues and maintain optimal performance, the decision was made to implement a dedicated inference server for the pose prediction model. This inference server would communicate with both the "KinectServer" and the "HololensClient" over a network, allowing each component to operate independently while offloading the computational load of the pose prediction model. By separating the inference model into its own server, we gain several key advantages. First, it provides flexibility, as the inference server can either run on the same machine as the other components or be allocated its own dedicated resources. If dedicated resources are available, this can help to reduce the load on the machines running the "KinectServer" and "HololensClient", further improving performance and ensuring that each component can focus on its core responsibilities. Additionally, since the MPC system is inherently distributed, adding the pose prediction model as a separate server fits naturally within the existing system architecture, enhancing scalability and maintainability.

With the decision to implement an inference server made, there are several methods available for enabling communication between the inference server and the other components. Popular options include REST APIs, remote procedure calls (RPCs), and sockets (TCP/UDP). While REST APIs and RPCs are commonly used for such purposes, they can introduce overhead due to their higher-level nature. In order to keep system latency as low as possible and maintain the responsiveness critical to the MPC system, it was decided to utilize low-level sockets for communication. This choice ensures that the communication between the inference server and the other components is as direct and efficient as possible. High-level frameworks, while convenient and simplifying development, often introduce internal processing that can add hidden delays. By using low-level sockets, we avoid such overhead and minimize the potential for latency, aligning with the system's overall performance goals. This approach provides a more streamlined communication channel, helping to preserve the real-time capabilities of the existing MPC system while integrating the pose prediction model.

In Section 2, we explored the differences between Autoregressive and Non Autoregressive pose prediction models, along with their respective drawbacks. As previously mentioned, the pose prediction model chosen for integration into this system is the STS-GCN model, a Non-Autoregressive model. This decision was based on prior research, which highlighted significant latency issues with the Autoregressive model evaluated in that work, making it unsuitable for integration into this system [DB24]. Additionally, attempting to optimize the auto-regressive model was deemed unnecessary, as it generates only one frame per inference call, rather than a series of predicted frames. This behavior did not align well with the system's workflow design.

3.3.3 End-to-End Architecture

The complete architecture of the MPC system integrated with the pose prediction deep learning model is shown in Figure 3.5. The complete source code can be found in the Interactive Media Lab Dresden (IMLD) gitlab repo [@Dut24].

As shown in figure 3.5, the inference server acts as an intermediary between the "KinectServer" and "HololensClient 2" (hereafter referred to simply as "HololensClient"), ensuring seamless coordination between them. Both the "KinectServer" and "HololensClient" act as clients for the inference server.

The inference server announces its presence and identity to both the "KinectServer" and the "HololensClient" via its "Identity Broadcaster" element. On their end, these components are equipped with the ability to detect the broadcast message. This functionality is implemented using an `InferenceServerDiscoverer` script, which is attached to a game object within the Unity scenes associated with the "KinectServer" and "HololensClient" components as shown in figure 3.6.

At a high level, the system operates as follows:

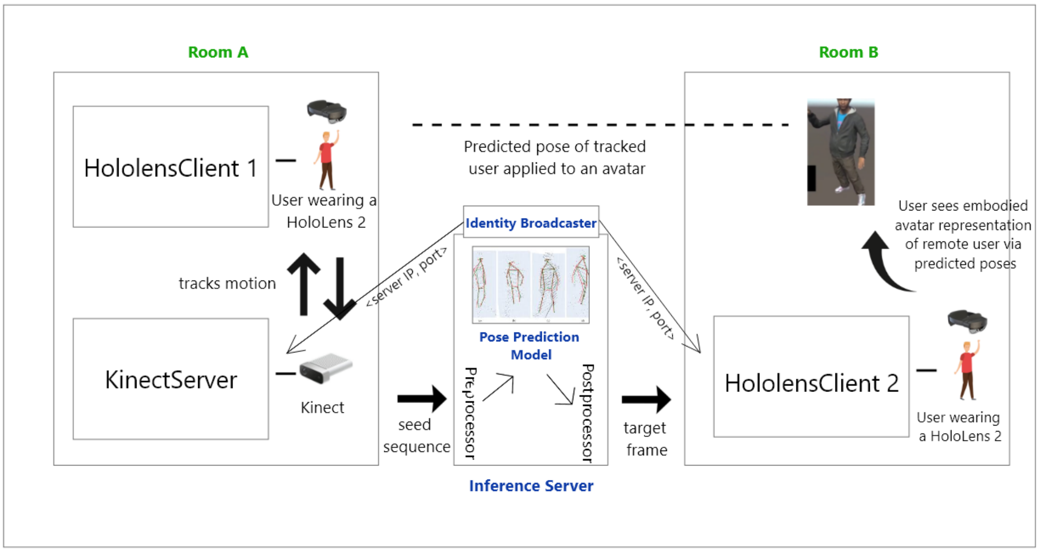
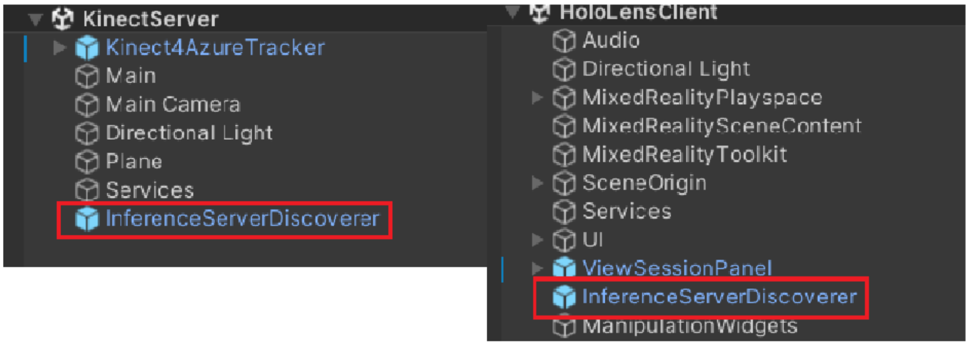


Fig. 3.5: End-to-End System Architecture : Integration of pose prediction model into the existing MPC system.



KinectServer
Kinect4AzureTracker
Main
Main Camera
Directional Light
Plane
Services
InferenceServerDiscoverer

HoloLensClient
Audio
Directional Light
MixedRealityPlayspace
MixedRealitySceneContent
MixedRealityToolkit
SceneOrigin
Services
UI
ViewSessionPanel
InferenceServerDiscoverer
ManipulationWidgets

Fig. 3.6: KinectServer and HololensClient equipped with InferenceServerDiscoverer.

1. The "KinectServer" tracks the user's motion, collects frames for the seed sequence, and sends them to the inference server.

2. The inference server preprocesses these frames for the pose prediction model and passes them to it.

3. The pose prediction model generates multiple future frames.

4. Based on overall system latency, an appropriate frame is selected, postprocessed, and sent to the "HololensClient".

5. The "HololensClient" is able to utilize the received frame.

A more detailed workflow will be provided in a later section. The "KinectServer" was chosen to transmit the seed sequence to the inference server rather than each "HololensClient" to avoid redundant data transmission over the network. Additionally, since each "HololensClient" may have to deal with different levels of latency, managing these differences would significantly complicate the implementation of the inference server. This approach is efficient also because the pose prediction model can handle predictions for multiple clients simultaneously by processing them in batches.

3.3.4 Implementation of the Inference Server for Pose Prediction

To ensure that the inference server can efficiently handle multiple clients concurrently, the decision was made to leverage asynchronous programming with Python's asyncio library. Asynchronous operations are particularly well-suited for I/O-bound tasks, such as networking and communication between the server and clients, making them the ideal choice for this use case. Unlike traditional multithreading, where Python's Global Interpreter Lock (GIL) can cause inefficiencies due to its limitation on executing code in multiple threads concurrently, asyncio enables multitasking. This means that the server can handle multiple tasks concurrently without the overhead of creating and managing multiple threads or processes. The core advantage of using asyncio is its ability to allow the server to handle many connections efficiently in a single thread. This is achieved through the use of coroutines ie, special functions that can pause and resume their execution, allowing the server to switch between tasks without blocking. Additionally, as would be the case with multiprocessing, by avoiding the need for creating separate instances of the pose prediction model for each client, the server can maintain a lean, efficient resource usage, and minimize overhead.

Since the workload of the inference server is I/O-bound with regard to communicating with the clients, asyncio provides an optimal and resource-efficient way to handle multiple client connections. By using asyncio's event loop and asynchronous tasks, the server can serve multiple clients concurrently without consuming excessive computational resources. This makes it a scalable and flexible solution, ensuring that the server can handle increased client load efficiently without significant performance degradation. Furthermore, currently the inference server handles two clients simultaneously: the "KinectServer" and one "HololensClient" from the MPC system. However, the use of asyncio allows for straightforward scalability, meaning the server can be easily modified to handle multiple "HololensClient" components concurrently. This design not only ensures optimal performance with the current setup but also provides the flexibility to support multiple users in the MPC system, thus enabling a more scalable system in the future.

Since the data from the "KinectServer" component is large due to the multiple frames in the seed sequence, it is read in 1KB chunks. The server distinguishes data sent between different server calls via a delimiter (newline character) in the last arriving chunk. The "KinectServer" component ensures that the delimiter is appended before sending pose data to the inference server, enabling seamless and efficient parsing of incoming data.

The inference server has the following elements:

1. Identity Broadcaster
2. Pose Prediction Model
3. Preprocessor
4. Postprocessor

The design of the inference server is structured to be modular. This modularization is achieved through a clear separation of the different elements responsible for processing. By decoupling these elements, the server can accommodate new pose prediction models without disrupting the overall system. This means that new or different models can be integrated into the server with minimal changes to its other parts. For example, if a more advanced pose prediction model, such as a newer variant of a Graph Convolutional Network or a model based on a different architecture like Transformers, is deemed feasible, it can be swapped in. The flexibility of the inference server allows it to adapt to such changes, ensuring the system remains scalable and future-proof.

In practice, this approach simplifies maintenance and upgrades. As new models are developed, they can be tested and integrated into the inference server easily, allowing it to continue evolving without significant downtime. By promoting easy interchangeability and scalability, the modular structure of the inference server enhances its robustness, making it suitable for integration into a range of environments such as research settings with high demands for performance and reliability.

Next, the inference server's different elements are explored in more detail.

Identity Broadcaster

As mentioned previously, the inference server's identity broadcaster broadcasts the server's IP address and port to all devices on the network via UDP packets. This allows the "KinectServer" and the "HololensClient" to determine the identity of the inference server thereby setting things up for future communication between them.

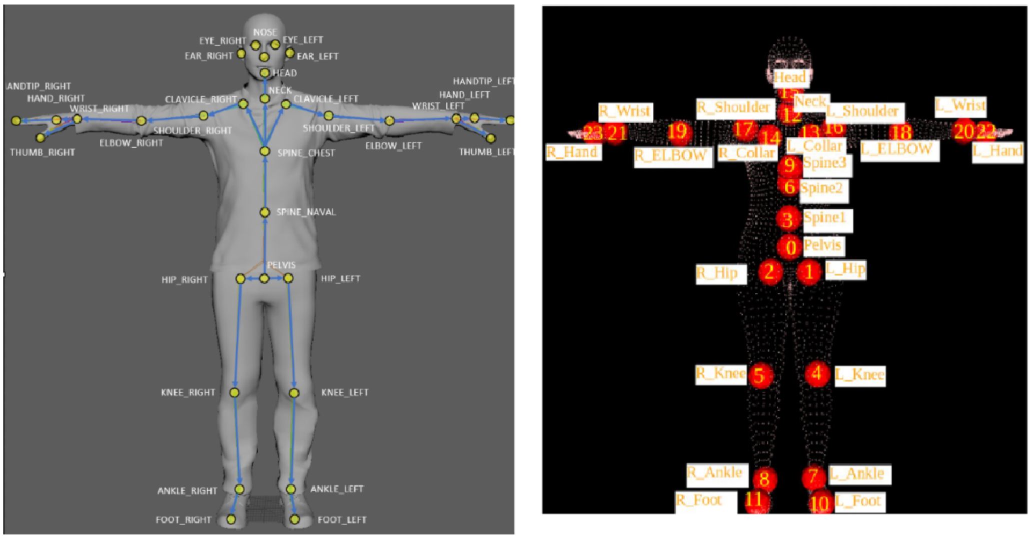
Pose Prediction Model

The 3D human pose prediction model serves as the core computational component of the inference server. It processes the input seed sequence - captured motion frames of a tracked user(s) - and generates predictions for future poses. The model is designed to handle high-throughput scenarios, allowing it to accommodate multiple simultaneous users by efficiently managing batched inputs. Its integration ensures reduced latency, a critical factor in maintaining real-time synchronization in MPC environments. More details about the specific pose prediction model that the inference server uses can be found in an earlier section.

The implementation of the inference server in this master's thesis leveraged the codebase of UnPosed, which was further modified to integrate seamlessly with the system. Enhancements included optimizing performance by initializing and loading the pose prediction model only once during the server's first call, rather than reloading it for each request. Furthermore, UnPosed's original inference code was initially developed to work with a test dataset, and it included several components that were not essential for the current use case. To integrate it into the inference server, the code underwent significant modifications. Unnecessary code was removed to streamline the process, ensuring that the system would be more efficient and tailored to the specific needs of the pose prediction model integration. These changes not only improved the code's performance but also helped reduce potential sources of latency and complexity. The goal was to make the inference process more efficient and better suited to the real-time requirements of the MPC system while ensuring smooth communication between the inference server and other components.

Preprocessor and Postprocessor

The pose prediction model chosen is trained on the AMASS dataset which uses the joint hierarchy as per SMPL and is also how the model expects input. But the pose data (seed sequence) arriving from the "KinectServer" originates from a Kinect and has a different joint hierarchy. The differences are shown in figure 3.7 [@Mic19; @Int].



a) Unity Joint Heirarchy

b) SMPL/AMASS Joint Heirarchy

Fig. 3.7: Kinect vs SMPL/AMASS joint hierarchy.

The pose prediction model requires only 18 joints from the input data, excluding the 4 static joints - pelvis, left and right hips, and spine 1 - as well as the 2 hand joints. In contrast, the Kinect tracks 32 joints, requiring a mapping process. The preprocessor bridges this gap by converting the Kinect's pose data format to align with the SMPL format required by the pose prediction model, ensuring compatibility and accurate predictions. Furthermore, the Kinect's XYZ axes are oriented in the opposite direction compared to Unity's coordinate system. As a result, the pose data from the Kinect was multiplied by -1. And, joint mappings required lateral inversion, meaning that the joints on the left side of the body were mapped to their corresponding joints on the right side.

The mapping between these two formats including lateral inversion is shown in the below code snippet.

```
_SMPL_TO_KINECT_JOINT_ID_MAPPING = {
  #SMPL : Kinect
  # 4 static joints
  #0: 0,        # pelvis
  #1: 18,       # left hip
  #2: 22,       # right hip
  #3: 1,        # spine 1 <-> spine_navel

  # 18 non static body joints
  4: 23,      # left knee     <->  right knee
  5: 19,      # right knee    <->  left knee
  6: -1,      # spine 2        =   missing in kinect joints
  7: 24,      # left ankle    <->  right ankle
  8: 20,      # right ankle   <->  left ankle
  9: 2,       # spine 3        =   spine chest
  10: 25,     # left foot     <->  right foot
  11: 21,     # right foot    <->  left foot
  12: 3,      # neck
  13: 11,     # left collar   <->  clavicle right
  14: 4,      # right collar  <->  clavicle left
  15: 26,     # head
  16: 12,     # left shoulder <-> right shoulder
  17: 5,      # right shoulder<-> left shoulder
  18: 13,     # left elbow    <-> right elbow
  19: 6,      # right elbow   <-> left elbow
  20: 14,     # left wrist    <-> right wrist
  21: 7,      # right wrist   <-> left wrist

  # 2 hand joints
  #22: 8,      # left hand
  #23: 15      # right hand
}
```

Listing 3.1: Mapping from SMPL joint ids to kinect joint ids

Additionally, to handle the missing spine 2 SMPL joint in the Kinect joints, it is taken as the average of spine 1 and spine 3 which correspond to the "spine navel" and the "spine chest" of the Kinect joints [Yao+18].

The preprocessor also converts the pose data received from the "KinectServer" into pelvis-relative joint positions. This adjustment centers the pelvis at the origin of the coordinate system, ensuring that all other joints are relative to it. This transformation is necessary because the Kinect provides absolute joint positions relative to the origin of the Kinect coordinate system ie, the Kinect sensor, which can vary depending on the user's location. The conversion is achieved by subtracting the pelvis's position from every other joint's position in the data. This approach standardizes the pose data, making it consistent and compatible with the requirements of the pose prediction model, which assumes a pelvis-relative coordinate system.

The postprocessor converts the predicted future frame's pose data from SMPL format back to Kinect format. For any Kinect joints not included in the pose prediction model's output, their x,y and z coordinates are assigned a value of 0.

Additionally, conversions of joint positions are carried out during both, preprocessing and postprocessing. During preprocessing, the joint positions are converted from meters (m) to millimeters (mm). This is necessary because the Kinect pose data provides joint positions in meters, but the pose prediction model was trained using joint positions in millimeters. Without this conversion, the pose prediction model would not be able to make accurate predictions, as it expects inputs in the unit it was trained on. Similarly, during postprocessing, joint positions are converted back from millimeters to meters. This step is needed because the pose prediction model predicts future frames using joint positions in millimeters. However, Unity operates with joint positions in meters, so the data must be transformed accordingly to maintain consistency and ensure that the predicted poses are correctly interpreted and displayed in the Unity environment.

These conversions are critical for ensuring that the data remains compatible with both the pose prediction model and Unity. The preprocessing and postprocessing steps act as bridges, allowing seamless communication between the Kinect's data format and the requirements of the deep learning model, as well as Unity's coordinate system. These conversions, although straightforward, play an important role in maintaining the accuracy and functionality of the system, ensuring the predictions from the pose prediction model are reliable and usable within the existing MPC system.

3.3.5 Detailed Workflow

To understand how the complete system functions and how its various parts interact with each other, let's examine an example scenario within the MPC system. The Kinect tracks motion at a framerate of 30fps, equating to a roughly 33ms interval between frames tracked by the "KinectServer". For the sake of simplicity let's assume the MPC system has a total latency of 33 x 4 = 132ms. This includes the latency introduced by the Unity body tracking SDK and the networking latency (which is considered minimal here). So each frame tracked by the "KinectServer" effectively reaches the "HololensClient" after 132ms.

The inference server, "KinectServer" and "HololensClient" are started up. The inference server broadcasts its IP address and port, which the "KinectServer" and "HololensClient" receive and store for future communication. The "KinectServer" tracks the motion of a user via a Kinect and composes a batch of the 10 newest frames as the seed sequence. Then it establishes a TCP/IP connection with the inference server, using the server's broadcasted IP and port. This connection is established once and reused for subsequent calls.

After transmitting the seed sequence, the "KinectServer" temporarily pauses data transmission, allowing the inference server to load, initialize, and warm up the pose prediction model. The inference server preprocesses the data and sends it to the pose prediction model for predicting future frames. A specific frame, selected based on the overall MPC system's latency considerations (now also considering the computation time of the pose prediction model and the data transmission time required to send the seed sequence from the "KinectServer" to the inference server, and subsequently from the inference server to the "HololensClient"), is postprocessed and transmitted to the "HololensClient".

The "HololensClient" can then utilize the provided frame to animate the remote avatar without noticeable lag. As new frames are tracked by the "KinectServer," the oldest frame in the seed sequence is replaced with the latest one, creating a new seed sequence for the next prediction cycle. This process repeats, allowing the "HololensClient" to continually receive animation frames. Once running, the inference server effectively acts as an intermediary, seamlessly supplying frames to the "HololensClient,".

3.4 Visualization and Evaluation

This section focuses on the implementation of pose data visualization and the evaluation of the pose prediction model's performance in terms of precision on the "HololensClient". Visualization serves as a crucial tool for verifying the correctness of the integrated system, allowing real-time observation of predicted poses within the MPC system. By rendering the predicted poses on the "HololensClient", users can assess the coherence and accuracy of the predictions, ensuring that they align with the intended motion sequences.

Additionally, this section evaluates the pose prediction model against two baselines: zero-velocity model and ground truth data. Zero-velocity predictions serve as a simple baseline where the last frame of the seed sequence is provided as the future predicted frame, while ground truth data provides an accurate representation of actual joint movements captured by the Kinect. By comparing the model's predictions to these benchmarks, its effectiveness in predicting future frames is evaluated through both visual inspection and the use of established performance metrics for quantitative analysis.

Both visualization and evaluation play an important role in assessing the effectiveness of the selected pose prediction model and its integration into the existing MPC system, which was one of the goals of this master's thesis. Additionally, both are implemented for real-time live tracked data from a Kinect, as well as for a saved motion sequence via the pose save and replay feature discussed earlier.

3.4.1 Implementation of Visualization

Initially, the predicted future frames generated by the model, which contained positions for the joints, were used to attempt the animation of a humanoid avatar with a skinned mesh renderer. However, this approach encountered significant challenges. One major issue was possibly the mismatch between the bone lengths of the predicted pose and those of the humanoid avatar, which resulted in an unrealistic and distorted animation. Additionally, since the model provided position data rather than joint orientations, the resulting movements lacked the natural alignment expected from real motion capture data, such as that recorded by the Kinect.

So to be able to more effectively visualize and observe the predicted poses, a simpler and more adaptable solution was implemented. A stick-figure avatar was chosen as the visualization medium. This decision was driven by the need for a clear and unambiguous representation of the pose data, where joint positions could be observed without the complications introduced by discrepancies in bone structure or orientation. The stick-figure avatar proved to be highly effective for this purpose. Its minimalistic design allowed for simple visualization of joint positions and the overall structure of the predicted poses. This simplification also made it easier to identify and analyze discrepancies between the predicted poses, zero-velocity baseline, and ground truth data.

Now onto how the actual pose data visualization works. Once the "HololensClient" receives the pose data from the inference server, it connects joint positions with lines to create a visual representation of a body skeleton. It draws lines between pairs of joints, as defined in the dictionary, to form the skeletal structure as shown in figure 3.8.

The code snippet shows the C-sharp dictionary that defines the joint hierarchy used to render a stick figure avatar. This dictionary outlines the parent-child relationships between joints, forming the skeletal structure of the stick figure. Each key-value pair in the dictionary represents a connection between two joints, where the key corresponds to the child joint and the value corresponds to its parent joint. These mappings allow for the visualization of the body structure as a connected series of line segments.

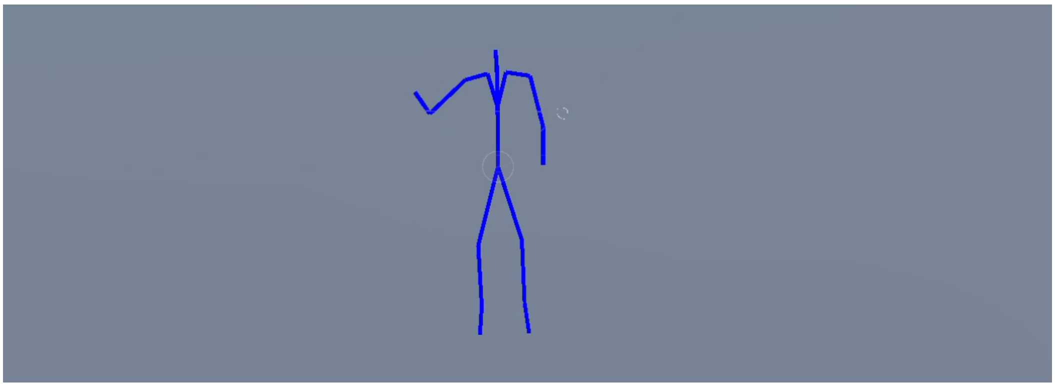


Fig. 3.8: Stick figure visualization of pose data on "HololensClient".

```
private readonly Dictionary<int, int> predJointHierarchy =
  new Dictionary<int, int>{
    /*{ 1, 0 }, { 2, 1 },*/ { 3, 2 }, { 4, 2 },
    { 5, 4 }, { 6, 5 }, { 7, 6 },
    /*{ 8, 7 }, { 9, 8 }, { 10, 7 },*/ { 11, 2 },
    {12, 11 }, { 13, 12 }, { 14, 13 },
    /*{ 15, 14 }, { 16, 15 }, { 17, 14 },*/
    { 19, 2 }, { 20, 19 }, { 21, 20 }, { 23, 2 },
    { 24, 23 }, { 25, 24 }, { 26, 3 }
};
```

Listing 3.2: The joint heirarchy dictionary used to draw the skeleton stick-figure avatar.

The joint identifiers in the dictionary are Kinect joint IDs, which correspond to specific body parts as tracked by the Kinect sensor. However, not all Kinect joints are included in the visualization. Joints that are not part of the pose prediction model's output are commented out and excluded from the visualization. This ensures that the skeleton visualization aligns with the available pose data while avoiding inconsistencies caused by missing joints. For example, the dictionary entry {3, 2} indicates that the neck joint (3) is connected to the spine joint (2) and is treated as a child of it. This relationship helps construct the skeletal hierarchy in a way that mirrors the actual human body structure.

By carefully defining the joint hierarchy, the visualized skeleton is both efficient and reflective of the pose prediction model's capabilities, making it an effective tool for observing pose data. This structure plays a critical role in maintaining the clarity and functionality of the stick-figure representation during visualization.

3.4.2 Implementation of Model Comparison and Precision Evaluation

Model comparison was implemented to enable the "HololensClient" to visually render three types of poses: pose prediction model poses, zero-velocity model poses, and ground truth poses. Initially, the simplest approach considered was to have both the pose prediction model and the zero-velocity baseline model generate their predictions, save them to a file, and later visualize them using a separate script. This would address the challenge of comparing predicted poses alongside ground truth, as the ground truth poses are available later than the predicted frames on the "HololensClient." However, this approach was discarded because it would prevent real-time visualization and comparison of the models with the benchmarks.

Instead, a more complex yet effective solution was adopted to allow real-time comparison. In this approach, three buffers are maintained: one on the inference server for the ground truth, and two on the "HololensClient"'s side to store the predicted poses - one for the pose prediction model and one for the zero-velocity model. On the inference server side, every time a batch of seed sequence frames is received, the last frame from the batch is appended to the ground truth buffer. If the number of frames in the buffer exceeds the set limit (which is the number of the future frame sent to the "HololensClient"), the oldest frame is removed to make space for the new frame. This ensures that only relevant ground truth frames are kept in the buffer. Once both the pose prediction model and zero-velocity model generate their predicted future frames, these frames are prepared for transmission to the "HololensClient." Along with the predicted frames, the most recent frame from the ground truth buffer is sent to the "HololensClient," but only if the buffer contains more frames than the set limit.

On the "HololensClient" side, the same buffering logic is used. Two buffers are maintained - one for the pose prediction model and one for the zero-velocity model. As the "HololensClient" receives the predicted frames from the inference server, they are added to the corresponding buffers. If the buffer size exceeds the set limit, the oldest frame is removed. When the "HololensClient" receives a batch of three frames - one ground truth frame and the corresponding predicted frames from both models - it can visualize all three poses at the same time. This process ensures that the "HololensClient" always has synchronized ground truth and predicted poses available for comparison. This approach to visually comparing different pose data involves observing pose information from the past, as the ground truth becomes available later than the predicted poses. This delay is specific to the model comparison visualization. In contrast, when visualizing only the pose prediction model's data, the process occurs in real-time.

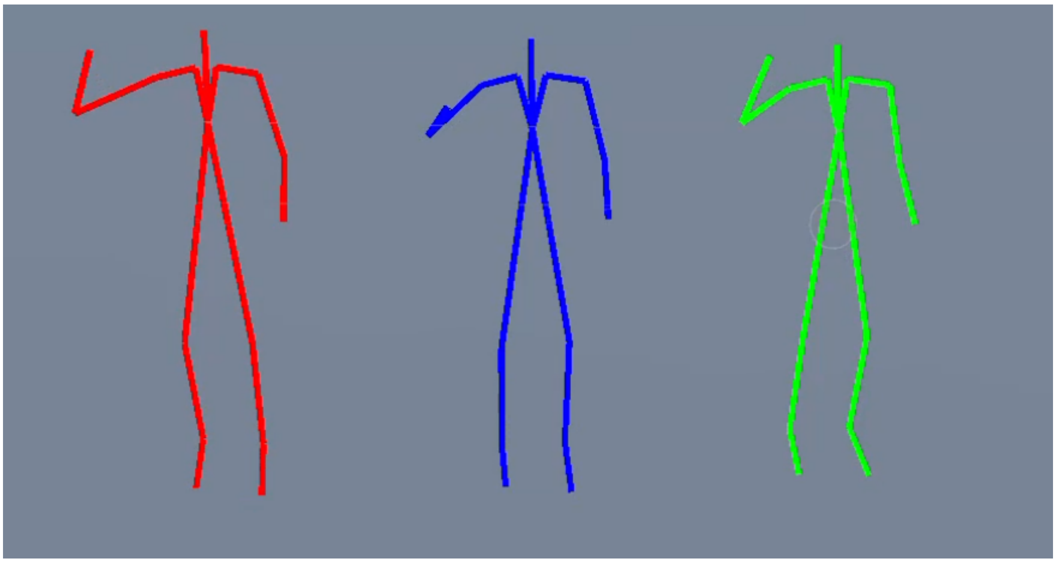


Fig. 3.9: Model comparison: Ground truth in Green, Pose prediction model in blue, Zero-velocity model in Red.

Once the three frames are available, they are visualized side by side on the "HololensClient" using a stick-figure avatar. The ground truth pose is displayed in green, the pose predicted by the model is shown in blue, and the zero-velocity model's pose is rendered in red, as shown in Figure 3.9.

Furthermore, to better visualize the relative error between the pose predictions of the model and the zero-velocity baseline compared to the ground truth, their predicted poses are superimposed on the ground truth with reduced opacity. This approach enhances clarity and enables a more intuitive comparison.

By using this approach, the system ensures that visualizations are continuously updated and aligned, providing a clear and accurate comparison of the pose prediction models' effectiveness in real-world scenarios.

Next onto precision evaluation. Visually comparing the ground truth and the pose prediction model's predictions are great but it's not quantifiable and hence it's measured quantitatively. Based on the existing literature, the most commonly used evaluation metric for pose precision is displacement error. It is the average euclidean distance over all the joints. And it was decided to report the displacement error averaged over a number of frames since it goes up and down depending on the specific pose in that frame. As the different pose data are being compared, at the same time, the average displacement error is computed between the ground truth and pose prediction model's prediction and between the ground truth and zero-velocity model's prediction. Also, only the joints available from the pose prediction model are considered for the displacement error computation.

3.5 Challenges Faced and Solutions

The process of integrating the pose prediction model into the existing MPC system had several challenges that required thoughtful solutions.

One of the first challenges encountered during the integration of the pose prediction model into the existing MPC system required an exploration and understanding of the existing system's codebase. This task was particularly challenging due to the complexity and comprehensiveness of the codebase, which had multiple interconnected components and was inherently distributed in nature. The distributed architecture meant dealing with communication between various parts of the system, each with its own dependencies and functionality. Familiarizing myself with such a system demanded significant effort, as it involved not just reading and understanding the code but also determining how to integrate new functionalities without disrupting existing operations.

One of the important steps during the initial stages of this integration was developing the pose saving and replay feature for study replicability. This feature served as both a practical tool and a learning opportunity. By implementing the pose saving and replay feature, the codebase could be navigated in a practical manner, which helped with understanding the system's structure and workflow. Moreover, the feature proved invaluable throughout the development process for several reasons. First, the pose saving and replay functionality reduced reliance on live Kinect data during debugging and testing. This simplification minimized hardware dependency, making it easier to troubleshoot issues in a controlled and reproducible environment. Second, it provided a mechanism to record motion sequences directly from the Kinect, which became crucial for visualizing and evaluating the integrated pose prediction model. With recorded data easily present, it was possible to iterate on the integration without the need for constant access to live input, enabling more efficient experimentation and testing.

Looking back, the initial challenges posed by the unfamiliar codebase and the distributed nature of the system were opportunities for deeper engagement with the project. The process of integrating the pose prediction model and building supporting features like pose saving and replay emphasized the importance of having modular, reusable tools within a complex system. It not only helped with the immediate development needs but also added long term value by enhancing the system's flexibility and usability for future research and development efforts.

Up next, the UnPOSed codebase uses Hydra, a powerful Python library designed to streamline the management of configuration files and parameters in software projects. Hydra is particularly used in domains like deep learning and machine learning, where workflows typically involve complex, hierarchical configurations. Rather than hardcoding parameters or relying on a single, big configuration file, Hydra allows for the dynamic composition of configurations by combining modular components. This modular approach ensures that configurations are not only reusable and maintainable but also adaptable to different scenarios and use cases. But, despite these advantages, it became apparent that Hydra was causing performance bottlenecks, particularly in terms of slowing down the execution of functions. The dynamic configuration process, while flexible, introduced overhead that hindered the real-time performance of the pose prediction model. This issue was particularly relevant in the context of inference server integration, where speed and efficiency were very important. To figure out that it was an issue with Hydra, various sections of the inference server integration code were timed. The resolution involved removing Hydra from the necessary parts of the inference server, using instead a direct configuration approach that would not cause unnecessary overhead.

The development of the integrated system faced delays due to slow testing processes, primarily because it required running multiple Unity instances simultaneously and coordinating their interactions in parallel. This approach was not only time consuming but also highly inefficient, as it placed a heavy load on system resources, slowing down the entire development and testing cycle. The need for three Unity instances - one running the "KinectServer" and two running the "HololensClient" - added to the complexity and time required for each test iteration.

To address this challenge, a solution was implemented by modifying the integration on the "HololensClient" side to directly consume requests from the inference server. This change allowed for a reduction in the number of Unity instances required, as it was no longer necessary to have multiple instances of the "HololensClient" running. Instead, testing could now be performed with just one instance of the "KinectServer" and one "HololensClient", streamlining the workflow. This simplification not only improved performance but also reduced the complexity of managing multiple Unity environments simultaneously. Furthermore, to facilitate more efficient development and testing, relevant Unity project folders were symlinked to create two synchronized versions of the project. This allowed both the "KinectServer" and the "HololensClient" to be run on the same machine without the need for separate machines. By using symbolic links, the project's resources could be shared seamlessly between the two instances, ensuring that changes made in one Unity instance were immediately reflected in the other.

Finally, the integration of the pose prediction model into the existing MPC system required using knowledge and technologies from multiple, different domains, including Human-Computer Interaction (HCI), deep learning, and networking concepts. Each of these areas brought its own set of challenges, which had to be carefully understood in order to successfully combine them. The task of integrating these technologies wasn't straightforward, as it required synthesizing knowledge from these different fields in a way that ensured compatibility and efficiency.

While proficiency in all these fields was not initially present, the process required significant learning and adaptation. In some areas, existing knowledge was sufficient, while other areas demanded deeper exploration and acquiring new concepts and skills. This experience highlighted the importance of cross disciplinary learning. The integration was not simply a technical challenge but also a learning journey that involved filling in gaps between domains. By combining these different technologies and applying them in a real-world context, the master's thesis provided a deeper understanding of how different technologies can work together to create a single, functional system.

Results

4

This chapter describes the results derived from the integration and evaluation of the pose prediction model within the existing Mixed Presence Collaboration (MPC) system. It aims to provide insights into the integrated system's performance and limitations through visualization, precision evaluation, and model comparison.

The first section talks about the qualitative results obtained through the visual representation of pose data. This visualization serves as a critical tool for verifying the functionality of the integrated system. The second section describes the quantitative results based on metrics such as displacement error. These results compare the accuracy of the pose prediction model against the zero-velocity baseline, offering an evaluation of the model's predictive capabilities. Finally, the last section explores the boundaries of the current implementation, identifying areas where the system's performance may fall short and provides insights into potential future improvements.

4.1 Visualization Results

Once the pose prediction model has been fully integrated into the existing Mixed Presence Collaboration (MPC) system, it is essential to evaluate whether the entire integrated system functions correctly. This involves verifying not only the transmission of predicted pose data from the inference server to the "HololensClient" but also ensuring that the data is accurately interpreted and utilized. Simply observing the raw numerical pose data arriving at the "HololensClient" is neither intuitive nor practical for identifying or diagnosing potential issues within the system.

To address this, a visual representation of the pose data proves highly effective. Translating the positional data into an animated avatar - specifically, a stick-figure avatar in our implementation - provides a clear and intuitive way to observe motion sequences. These sequences, whether derived from live tracking with a Kinect or replayed from a saved pose data, allow for real-time validation. By animating the avatar, one can visually verify whether the predicted motions align with expectations, enabling a straightforward way to detect inconsistencies or errors.

This visualization approach plays a crucial role in troubleshooting and ensuring system functionality. For instance, it can help uncover joint-mapping issues that might exist. It also provides a means to validate the correctness of preprocessing and postprocessing steps applied to the pose data before and after prediction. If any discrepancies or errors occur during these stages, the visual representation will make them more apparent than the raw data alone.

Overall, visualizing the predicted pose data serves as a powerful tool for verifying the functionality of the integrated system. It ensures that all components are working as expected, and it helps with the process of identifying and resolving potential issues.

4.1.1 Visualization of Predicted Poses from Live Tracked Data

One of the goals of the integrated system was to seamlessly process live tracked data from a Kinect, transmit it to the inference server, generate predicted poses using the pose prediction model, and then send these predictions back to the "HololensClient" for visualization. The system successfully accomplishes each of these steps, enabling real-time interaction and validation. The predicted poses are not just processed but are also represented visually through an animated stick figure avatar, as illustrated in Figure 4.1. This visual representation offers an intuitive and simple way to observe and analyze the motion data, providing immediate feedback. The use of a stick figure avatar simplifies the visualization process while effectively conveying the dynamics of the predicted motion sequences.

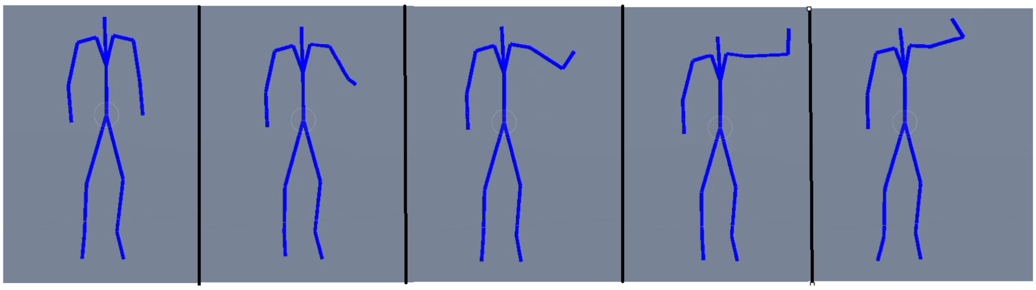


Fig. 4.1: Stick figure avatar depicting the predicted hand-waving motion from live-tracked Kinect data.

One instance of the direct advantage of visually rendering the predicted poses was the ability to quickly identify issues with the system. Specifically, when the predicted poses from the pose prediction model arrived at the "HololensClient," they did not appear as expected. The head was positioned near the pelvis, and overall, the joint placements were incorrect, with no clear correlation to the expected body pose. This issue could have been due to a problem with the pose prediction model itself or an error in the preprocessing or postprocessing stages of the integrated system.

The real-time visualization of the predicted poses revealed a potential issue with the transformation of the pose data before being fed into the model. It was hypothesized that the pose data required lateral inversion, as the Kinect and Unity coordinate systems operate on opposite axes. This observation led to a review of the input seed sequence, where a lateral inversion was applied to the pose data, which led to the predicted poses being correct. Through the real-time visualization of the poses, this subtle yet critical problem was identified, which might have otherwise gone unnoticed. The issue was resolved by adjusting the data preprocessing accordingly.

This example highlights the importance of visualizing pose data in diagnosing problems that might not be immediately apparent from raw data alone, ensuring that the integrated system functions as intended.

4.1.2 Visualization of Predicted Poses from Saved Pose Data

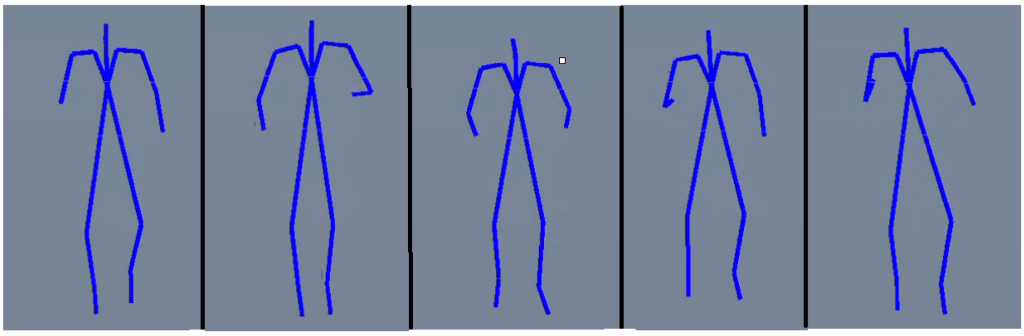


Fig. 4.2: Stick figure avatar depicting the predicted running motion from saved pose Kinect data.

In addition to visualizing live tracked data, the integrated system is also capable of rendering predicted poses from a pre-saved motion sequence captured by the Kinect, as shown in figure 4.2.

4.2 Model Comparison and Precision Evaluation Results

To evaluate the performance of the integrated system in terms of precision, the pose prediction model that was utilized was evaluated against both the ground truth poses and a zero-velocity baseline model. This evaluation was conducted through both visually and calculation of displacement error, a common metric for measuring pose accuracy. The comparison was carried out for two different data sources: live tracked data captured in real-time from the Kinect, and motion sequences that were specifically selected and pre-recorded for this purpose. By comparing the predictions of the pose prediction model with these benchmarks, it was possible to quantify and visually verify the pose prediction model's ability to predict accurate poses.

In the following subsections, the ground truth poses are represented in green, the predicted poses from the pose prediction model are shown in blue, and the predictions from the zero-velocity model are visualized in red.

4.2.1 Comparison of Models using Live Tracked Data

Figure 4.3 shows the visual comparison of the pose prediction model's predicted poses with respect to the actual ground truth and the zero-velocity baseline model's predictions.

The displacement error is not reported for live tracked data due to its non-reproducible nature. Live-tracked motions are inherently dynamic and unpredictable, meaning that the sequences captured during different instances are unlikely to be identical. This variability makes it impractical to compute and compare displacement error metrics reliably, as the ground truth itself is constantly changing with each new motion sequence. Instead, for live tracked data, the focus is shifted to real-time qualitative evaluation. Visualizing the predicted poses alongside the ground truth and the zero-velocity model allows for immediate, intuitive assessments of how well the integrated system performs. Any discrepancies, such as misaligned joints or unnatural motions, can be directly observed.

An immediate and noticeable distinction between the predicted poses from the pose prediction model, the actual ground truth, and the zero-velocity baseline model is the jitteriness observed in the visualization of the pose prediction model's predicted poses. This jittery behavior can be due to small inconsistencies in the model's predicted joint positions across frames which can cause a jittery appearance. And

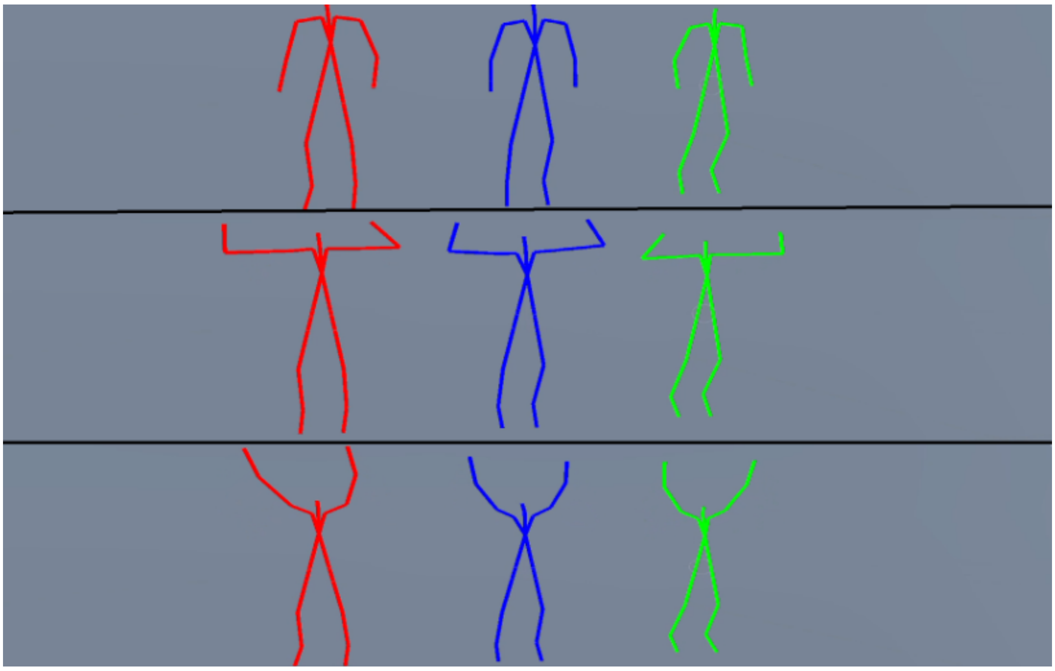


Fig. 4.3: Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames.
Source : **Live-tracked Kinect data of a person raising both arms.**
Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model.

these inconsistencies itself may result from overfitting to training data or the model's inability to generalize well to real-world scenarios ie, out of distribution data such as live tracked data from a Kinect. It might be possible to mitigate such jitteriness by applying smoothing techniques to the predicted pose data.

4.2.2 Comparison of Models and Precision Evaluation using Saved Pose Data

While visualizing live tracked data provides valuable real-time feedback, it does not lead to reproducible results due to the variability present in user generated movements. Each session may involve slight differences in the poses performed, making it challenging to evaluate the system's performance consistently or compare results across different trials. To address this limitation, the pose save and replay feature, previously described in an earlier chapter, was used to capture and store predefined motion sequences from the Kinect.

Three distinct motion sequences were recorded, carefully selected to encompass a diverse range of movements. These sequences were designed to include variations in motion intensity, ranging from slow, deliberate movements to rapid, dynamic actions. Additionally, the sequences incorporated motions involving different levels of joint activity, from those engaging only a few joints to those requiring coordinated movement across the entire body.

The three motion sequences recorded are as follows:

1. Slowly waving one hand.

2. Quickly waving both hands.

3. Quickly running. This engages most of the joints of the body.

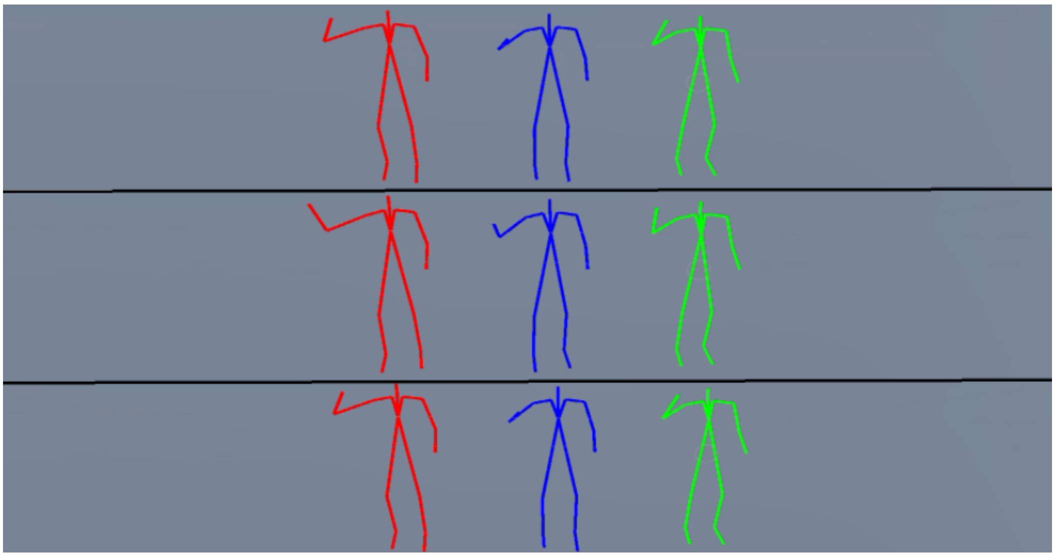


Fig. 4.4: Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames.
Source : **Slowly waving one hand saved pose data.**
Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model.

Figures 4.4, 4.5, 4.6 show the visual comparison of the pose prediction model's predicted poses with respect to the actual ground truth and the zero-velocity baseline model's predictions for all of the three recorded motion sequences.

Additionally, figure 4.7 illustrates the predicted poses from both the pose prediction model and the zero-velocity baseline, with each model's predictions superimposed on the ground truth for all three recorded motion sequences. This visualization clearly highlights the differences in performance between the two models. Upon closer inspection, it is evident that the pose prediction model consistently falls short when compared to the baseline. While both models aim to approximate the true pose, the baseline appears to maintain a closer alignment with the ground truth

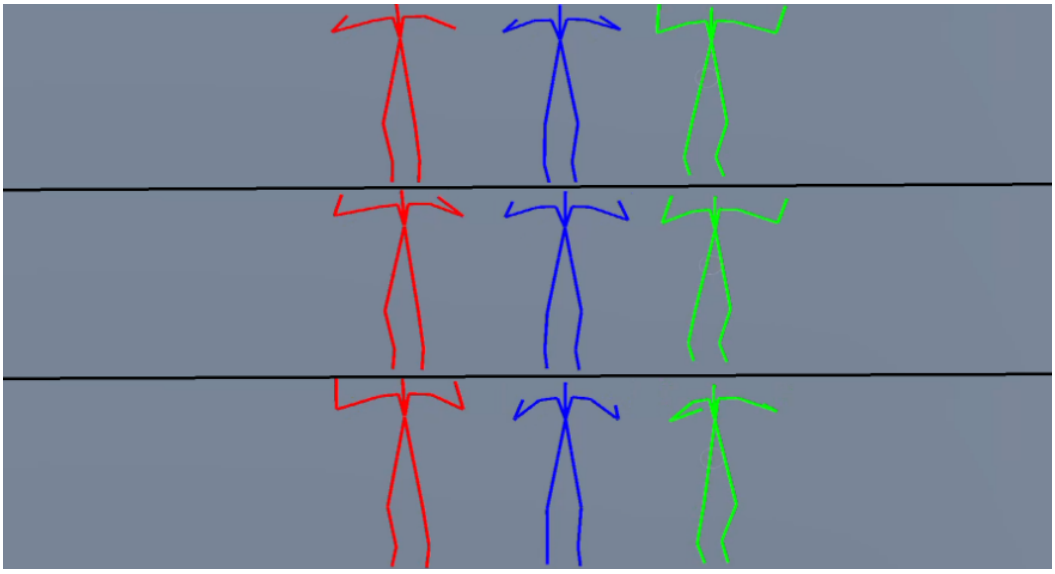


Fig. 4.5: Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames.
Source : **Quickly waving both hands saved pose data.**
Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model.

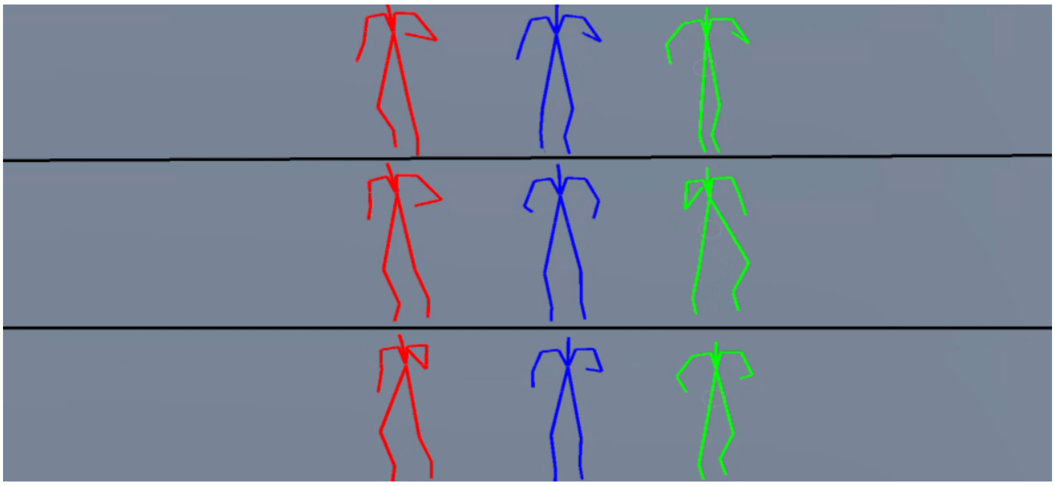


Fig. 4.6: Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames.
Source : **Quickly running saved pose data.**
Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model.

across all sequences. This suggests that the pose prediction model may be struggling with accuracy or robustness in certain scenarios, requiring further refinement or adjustments.

Although visual comparison is a valuable tool for an intuitive understanding of the differences in pose data, it has limitations. Subtle variations between poses may

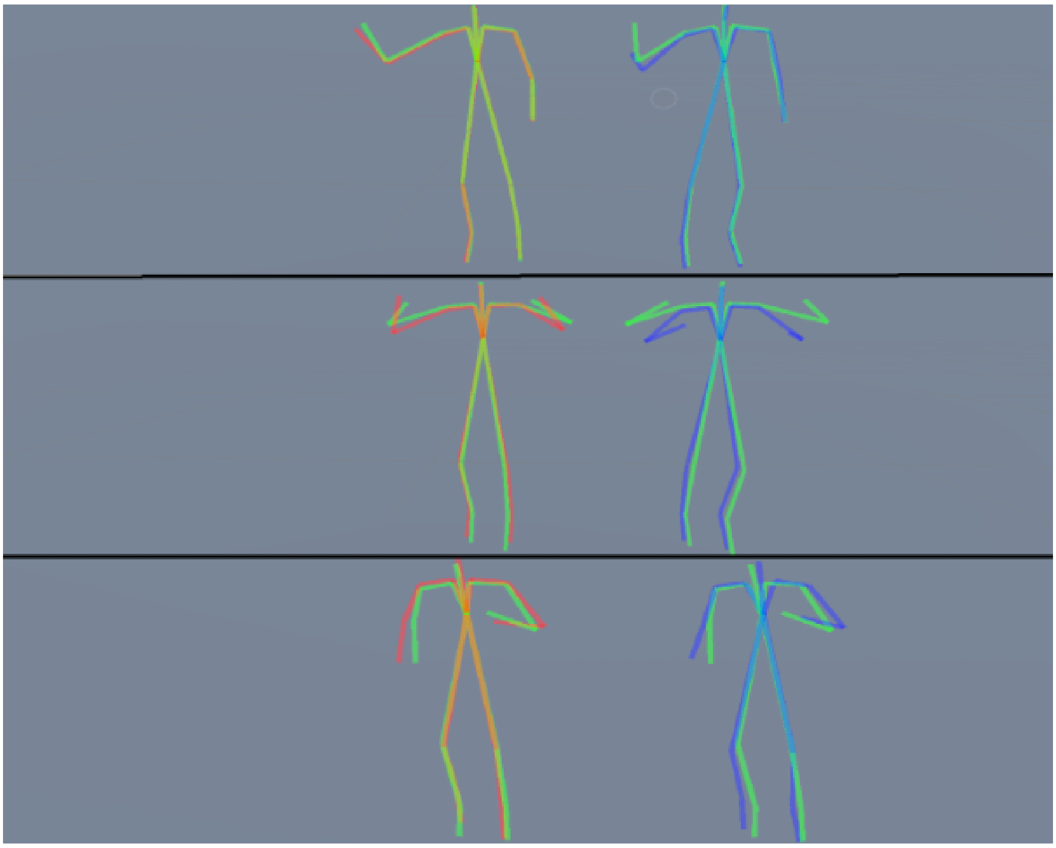


Fig. 4.7: Superimposed model comparison: Ground truth in Green, Pose prediction model in blue, Zero-velocity model in Red.
Top: **Slowly waving one hand saved pose data.**
Middle: **Quickly waving both hands saved pose data.**
Bottom: **Quickly running saved pose data.**

not be immediately apparent to the naked eye, and the process does not provide a measurable way to assess performance. To address these issues, the displacement error was computed.

For this analysis, the displacement errors of the pose prediction model's predicted poses and the zero-velocity baseline model's predicted poses were computed relative to the ground truth for all three pre-recorded motion sequences averaged over 200 frames. The 5th frame from the pose prediction model's predicted target sequence was selected as a representative case for evaluation. Consequently, the reported errors may vary if an earlier or later frame is chosen. However, this serves as a suitable general evaluation. This provides a more objective and detailed evaluation of the model's precision in replicating human motion. The results, summarized in Table 4.1, highlight the comparative performance of the models and make it easier to discern patterns or shortcomings that may not be visible through simple visual inspection.

Motion Sequence	**Predicted**	**Zero-Vel**
Slowly waving one hand	0.0166	0.0064
Quickly waving both hands	0.0359	0.0280
Quickly running	0.0551	0.0484

Tab. 4.1: Average displacement error over 200 frames in meters (m).
Predicted is the Integrated Pose Prediction Model vs Ground Truth.
Zero-Vel is the Zero-Velocity Baseline Model vs Ground Truth.

The average displacement errors reveal that the chosen pose prediction model does not perform as well as the zero-velocity baseline model across all three recorded motion sequences. [Zha+24] mention that this could be because the actor's body size differs from the norm or from the size used in the model's training. In any case, an interesting trend emerges when comparing the two models. Although the pose prediction model consistently performs worse than the zero-velocity baseline model, the baseline model's error rate increases dramatically as the complexity and speed of the motion grows. In particular, when the motion involves multiple joints moving quickly, the zero-velocity model, which assumes no movement, struggles to maintain accuracy. This is because the baseline model performs well in situations where motion is minimal, remaining close to the ground truth, but it fails to adapt to more complex, dynamic movements, leading to a larger error.

This suggests that there is likely a threshold of complexity in the motion sequence where the zero-velocity baseline model's error might become worse than the pose prediction model's. In simple and slow motions, the zero-velocity model is much more effective, but as the speed and complexity of the movement increases, the pose prediction model, despite its relatively higher error begins to catch up to the baseline model. This indicates that the pose prediction model has the potential to handle more dynamic motions, although it needs further optimization to better match the accuracy of the zero-velocity model.

It is certainly possible to enhance the performance of the pose prediction model by fine-tuning it with more specific Kinect data or by training it for additional epochs. This could help improve the model's accuracy. When that is achieved, exploring a hybrid approach could further offer an efficient solution. For example, a simpler model such as the zero-velocity model or a basic joint velocity-based model could be used when the motion is expected to be simple and minimal. This would save computational resources, especially when the system running the inference server is under heavy load. When more complex motion sequences arise and system load permits, the pose prediction model could then take over. By combining both models in this manner, it is possible to balance system performance and accuracy, optimizing both computational efficiency and the quality of the predicted poses.

4.3 Limitations of the Integrated System

This section addresses the current constraints and areas for improvement within the integrated system. While the system effectively integrates live human motion tracking with pose prediction and supports real-time visualization, it represents a foundational prototype and has its limitations. These limitations highlight the system's developmental stage and detail the necessary extensions for achieving a fully functional and robust Mixed Presence Collaboration (MPC) system.

4.3.1 Limitation in Predicting Moving Poses in Space

The pose prediction model selected for integration into the existing MPC system ie, STS-GCN model can predict future poses. However, a significant limitation of the model is that it can only predict poses in a local context. Specifically, it does not account for spatial movement, meaning the model can only predict poses for a stationary individual rather than someone moving through space. This limitation arises from the constraints of its training data. When trying to use the model for inference, it became evident that the STS-GCN model does not include "stationary joints" such as the pelvis, left hip, right hip, and navel spine in its input or output. Without the pelvis joint in particular, the predicted poses cannot guide an animated avatar to move through space.

A deeper analysis of the model's training dataset, the AMASS dataset, revealed the root cause of this constraint. The AMASS dataset is preprocessed to remove global translations from the motion data, keeping the pelvis at the origin and making all joint positions relative to the pelvis. This preprocessing step effectively removes global movement information, leading to the model's local prediction design. This is also the reason why during the preprocessing stage in the inference server, the same is done.

This issue was discovered late in the integration process, leaving insufficient time to fully address the challenge or adapt the system to account for global pose forecasting.

As a result, the development effort prioritized completing the integration of a functional system. This approach ensures a foundation is in place to support future improvements, such as swapping in a model that can handle global motion prediction. With the current integrated system in place, replacing or enhancing the model to allow global pose forecasting is a feasible next step in its development.

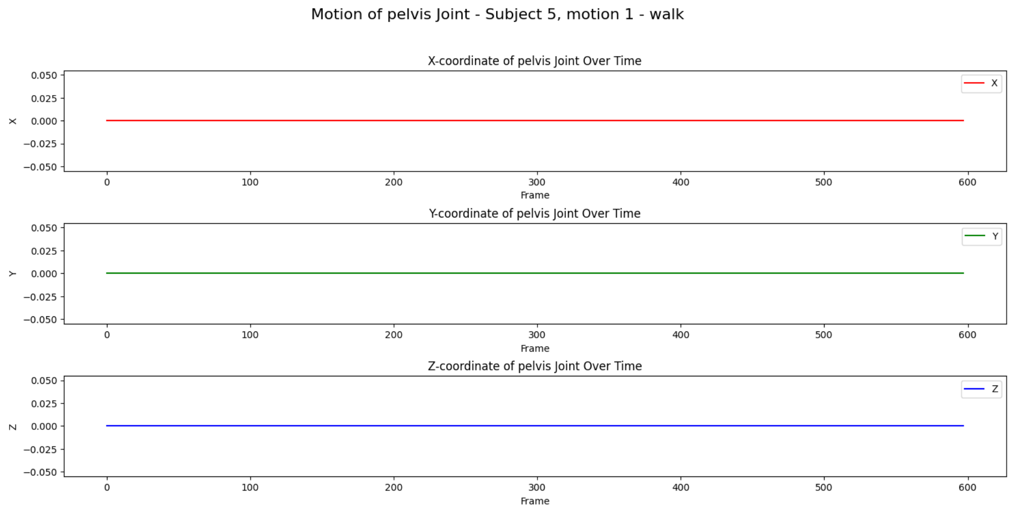


Fig. 4.8: Motion analysis of pelvis joint of a subject from AMASS dataset.

Figures 4.8 and 4.9 illustrate the X, Y, and Z coordinates of a subject's pelvis and left foot joints during a motion sequence taken from the AMASS dataset, which serves as the training data for the STS-GCN model. A close examination reveals that the pelvis remains fixed at the origin (0,0,0) throughout the sequence, while the XYZ coordinates of the left foot show periodic movement. This suggests that the motion occurs "in place". Because if global translation did occur, the coordinates would show a progressive increase or decrease along at least one axis, reflecting the subject's spatial movement. Similar analyses of other datasets, such as Human3.6M and 3DPW, confirm the use of similar preprocessing. The global translation is removed from the motion data, anchoring all joint positions relative to the pelvis. This pattern suggests an intentional decision during dataset preparation, effectively training models like STS-GCN to predict poses under the assumption of no global movement.

Further clarification was obtained by consulting the original researchers behind the STS-GCN model and the UnPOSed framework. One of the researchers confirmed that the field of pose prediction traditionally prioritizes local motion prediction, excluding global translations. This decision is largely influenced by the primary aim of capturing fine-grained joint dynamics over short time spans (typically under 1 second), where global movement is often negligible. By simplifying the prediction

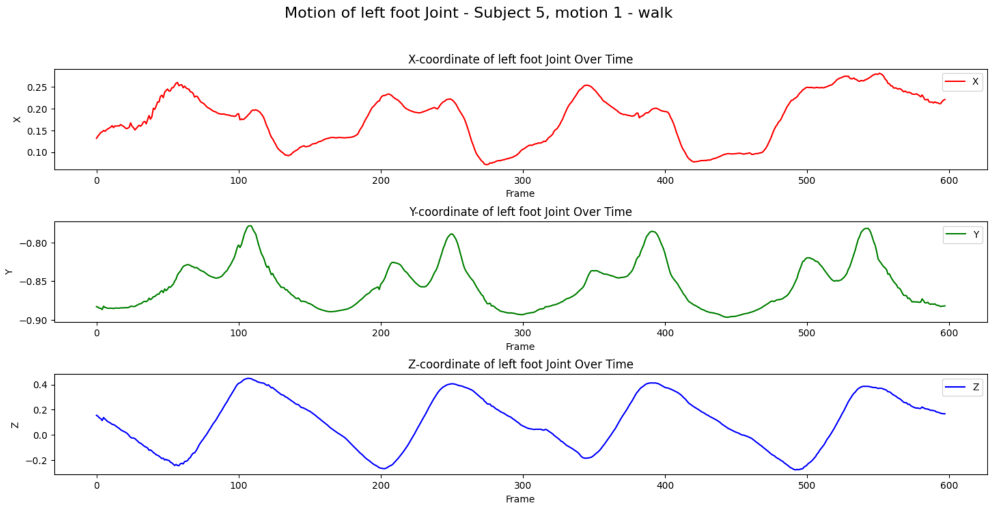


Fig. 4.9: Motion analysis of left foot joint of a subject from AMASS dataset.

problem, researchers can focus on refining the precision of localized joint behavior, a challenge that already involves significant complexity.

While this constraint simplifies model training, it presents a notable limitation for practical applications. The inability to predict global movement implies that the model cannot predict poses for individuals moving through space but only for those performing actions while stationary. This limitation affects the broader applicability of such models, particularly in scenarios where global pose prediction is essential, such as real-world motion tracking or mixed-reality applications.

While global pose prediction has not received as much attention as traditional pose prediction, some work has been done in this area. [JPY24] combine local pose prediction with multi-modal global trajectory prediction. In a similar vein, [Mah+22] developed a non-autoregressive transformer model to predict future pose. [Sco+23] propose a three stage process. First, they consider the scene and human interactions at contact points. Next, they predict the overall human trajectory within the scene. Finally, they refine the motion by matching detailed joint movements to the trajectory, accounting for the estimated contacts. [XML24] represent the interaction between the human body and the scene where movement occurs using the mutual distance between them. These mutual distances constrain both local and global human motion, leading to a prediction that accounts for whole-body motion constraints. They design a pipeline with two sequential steps: first predicting future mutual distances, then forecasting future human motion in the scene.

4.3.2 Multiple "HololensClient" and "KinectServer" Support Challenges

The current version of the integrated system serves as a foundational prototype designed to implement a minimal configuration of the existing MPC system. This prototype setup involves one "KinectServer" and one "HololensClient." While it provides a strong proof-of-concept, there is room for enhancement to bring it closer to the full capabilities of the existing MPC system. Currently, the inference server supports a single "HololensClient." However, thanks to its asynchronous programming architecture, it can be easily scaled to manage multiple "HololensClient" connections concurrently.

Extending support to multiple "KinectServer" components, on the other hand, introduces additional complexities. The existing MPC system differentiates between multiple "KinectServer" instances through the concept of rooms, which differentiates them. For the integrated system to support this feature, the inference server might also need to incorporate a room-based association mechanism, ensuring it correctly links data from each "KinectServer" to the corresponding "HololensClient." Implementing this room-based association could be achieved by using a single inference server for all "KinectServer" components in the MPC system. However, this approach risks overloading the inference server and complicating the architecture of the inference server, as the server would need to precisely manage which "HololensClient" corresponds to which "KinectServer." A more scalable and efficient alternative might involve deploying multiple inference servers, each dedicated to a specific "KinectServer." This architecture would distribute the computational load and simplify the overall architecture.

By expanding to accommodate these features, the integrated system can evolve from a basic prototype to a robust solution capable of mirroring the functionality of the MPC system in its entirety, supporting multiple users and devices simultaneously.

4.3.3 Proposed Latency Testing Approach for the Integrated System

The current design of the integrated system theoretically offers the potential to reduce or eliminate latency within the MPC system. However, due to time constraints, this capability has not been empirically tested with a fully operational MPC system setup. Consequently, the extent of latency reduction achieved by the integrated system remains unverified.

To evaluate the latency performance comprehensively, a complete MPC system configuration would need to be established. This setup would include one "KinectServer" and two "HololensClient" components, potentially running on different machines. Additionally, the inference server would operate on its dedicated machine, resulting in a total of four interconnected systems. The evaluation process would involve recording the live motion tracking of a person using a high-framerate camera while simultaneously capturing the motion rendered on the "HololensClient" both with and without the inference server integrated into the MPC system. The time lag observed between the live physical motion and the rendered motion on the "HololensClient" would then be analyzed to calculate the system's latency. By comparing these latency measurements in both scenarios, it would be possible to quantify the integrated system's impact on latency reduction.

Such a rigorous testing approach would not only validate the theoretical advantages of the system but also provide critical insights into its real-world performance and its readiness for broader applications.

Conclusion 5

This thesis focuses on the integration of a 3D human pose prediction model into an existing Mixed Presence Collaboration (MPC) system. The primary goal of this work is to address the challenges posed by latency in MPC systems, which can significantly disrupt the natural flow of interactions in collaborative environments.

There is an increasing reliance on MPC systems in fields such as telepresence, remote collaboration, and augmented reality applications, where accurate and real-time representation of human motion is critical. By integrating a pose prediction model capable of predicting future poses, the system can compensate for delays in motion capture and data transmission, effectively bridging the gap between real-time motion and rendered poses. This work explores not only the technical implementation of such integration but also provides practical insights into the challenges and considerations involved in adapting pose prediction models for real-time applications.

Through this integration, the thesis aims to contribute to the broader development of MPC systems and paving the way for future advancements in mixed-reality collaboration technologies. The work undertaken here is both exploratory and foundational, offering a stepping stone for further research into utilizing pose prediction models to enhance the capabilities of MPC systems.

This chapter concludes the master's thesis by reflecting on its core contributions, key findings, and potential future directions. The first section provides an overview of the primary contributions made through this research. The next section highlights the significant results and lessons learned, offering a deeper understanding of the implications of the work. Finally, the last section outlines promising directions for extending and enhancing this work, paving the way for further exploration and innovation.

5.1 Summary of Contributions

The key contributions of this work span system design and architecture, pose prediction model integration, visualization, and precision evaluation, providing insights and a practical implementation for enhancing MPC environments. The work addresses both the theoretical and implementation challenges of integrating pose prediction models into an MPC system, focusing on optimizing performance and reducing latency. Through relevant visualization and evaluation, this thesis lays a foundation for future advancements in real-time usage of pose prediction models in MPC systems.

System Design and Architecture : Careful consideration went into the design and integration of the pose prediction model within the existing MPC system. At its core, the integrated system leverages a modular architecture that accommodates different components of the existing MPC system, such as the "KinectServer", "HololensClient", and adds an inference server responsible for processing pose data as illustrated in figure 3.5.

The "KinectServer" captures the user's motion, generates pose data, and sends it to the inference server. This server then processes the data using the pose prediction model to predict future poses. A suitable future frame can be selected based on the overall latency of the MPC system and is subsequently provided to the "HololensClient" for visualization.

The integrated system was designed with scalability in mind, employing asynchronous programming in the inference server. Although only a single "KinectServer" and "HololensClient" is supported, the design provides a clear path for extending the system to handle multiple instances of these components in a more complex MPC setup.

Overall, this work bridges the gap between pose prediction models and Mixed Presence Collaboration systems.

Development of Real-Time Visualization Features : The implementation of real-time visualization capabilities enabled the animated rendering of predicted poses using a stick figure avatar, illustrated in figure 4.1. This feature proved critical for debugging, evaluation, and qualitative assessment, providing an intuitive way to observe system behavior and identify potential issues, such as incorrect joint mapping or preprocessing errors.

Pose Data Saving and Replay Functionality : To support replicable studies, systematic testing, and easier development, the pose saving and replay feature was added to the existing MPC system, illustrated in figures 3.3 and 3.4. This allowed for the recording of motion sequences from the Kinect, enabling evaluation with pre-recorded data alongside live data. This feature facilitated detailed comparisons between the pose prediction model, the zero-velocity baseline, and the ground truth.

Quantitative and Qualitative Model Precision Evaluation : This work also evaluated the integrated pose prediction model's prediction accuracy through both qualitative visualization and quantitative metrics. Average displacement error was computed to compare the model's predictions against the ground truth and the zero-velocity baseline model, providing a clear and measurable way to assess performance. The evaluation also highlighted the model's limitations, such as its inability to be as good as the zero-velocity baseline model as can be seen in table 4.1.

Theoretical Contributions to Latency Reduction : The system was designed and architected to reduce latency by predicting the appropriate future frame based on the MPC system's latency to be combatted. While this latency reduction was not empirically tested, the design itself represents a theoretical contribution, offering a framework for future testing and validation of latency improvements in real-time MPC systems.

Impact on the Existing MPC System : This work creates a foundational prototype of integrating a pose prediction model into an MPC system. The modularity of the design ensures that it can be expanded and adapted to meet the requirements of a fully operational MPC system, supporting more complex interactions and broader use cases. New pose prediction models can be easily swapped in and evaluated for their potential to reduce latency. The integrated system's visualization and precision evaluation features support this process.

5.2 Key Findings and Insights

Visualizing the pose predictions of the STS-GCN model and comparing them with the zero-velocity baseline model provided critical insights into the model's precision and limitations. While the STS-GCN model was functional, its precision fell short when compared to the zero-velocity baseline. This highlighted the need for further finetuning in the pose prediction model to better handle dynamic and complex motion sequences. However, the comparisons also revealed the potential of the

STS-GCN model ie, with proper fine-tuning, it could outperform the baseline in scenarios involving rapid or complex movements, offering greater adaptability for real-time systems.

The integration of pose prediction models with MPC systems remains a relatively underexplored area. Much of the existing research focuses on evaluating pose prediction models using static datasets, leaving the challenges of real-time integration largely unaddressed. This work highlighted several important considerations for successfully integrating a pose prediction model with live-tracked motion data in a real-time environment.

One major insight is the distinction between local and global pose prediction. While local pose prediction focuses on the relative movements of joints without accounting for spatial translation, global pose prediction extends this to include movement through space. The STS-GCN model, like many others, was designed for local pose prediction, which became evident from its exclusion of the pelvis and hip joints in both input and output. Additionally, the model's training data was preprocessed to remove global translations, reinforcing its local pose prediction design. Identifying this distinction is critical when selecting a model, as global pose prediction is essential for applications requiring spatial motion tracking.

Another consideration is the compatibility of the model's joint outputs for animating avatars. It is important to verify whether the predicted joints are sufficient for driving an avatar's animation or if additional computation is needed to fill in missing joints. Furthermore, the ability to map the model's joints to those provided by motion capture devices, such as the Kinect, ensures seamless integration. These mappings must be robust and account for any potential discrepancies in joint definitions. Preprocessing also plays a key role in ensuring compatibility between the motion capture device and the pose prediction model. Differences in coordinate systems, such as axis orientations, require adjustments like lateral inversion or alignment to ensure that the input data is correctly interpreted by the pose prediction model. These preprocessing steps are essential for maintaining consistency between the real world data and the model's predictions, particularly in real time systems.

Finally, the type of input and output representation used by the model has significant implications for system design. Many pose prediction models, including the STS-GCN, rely on joint positions as input and output. However, using joint orientations instead could simplify the animation pipeline. Orientation-based models eliminate the need to compute orientations from positions manually and avoid issues like bone length inconsistencies, which can arise when animating avatars with skinned mesh renderers using positional data.

Overall, using pose prediction models in real-time scenarios requires careful consideration of factors such as prediction type, joint compatibility, preprocessing needs, and data representation. These insights not only inform the selection and integration of models but also point to opportunities for improving the system's accuracy and efficiency. By addressing these challenges, future systems can deliver more robust real-time performance and better support for Mixed Presence Collaboration.

5.3 Future Work

This version of the integrated system incorporates the STS-GCN model as its core pose prediction component. However, the model is limited to predicting local poses, which excludes spatial translations. Future work could explore benchmarking models capable of global pose prediction. An alternative approach would involve retraining the STS-GCN model using datasets that retain global positions, thereby enabling it to predict spatial translations. This would also likely require modifying the model to include the pelvis and other joints, but such changes would require modifications to the model's architecture, particularly its input and output feature space at the very least.

Quantitative comparisons between the STS-GCN model and the zero-velocity baseline revealed that the STS-GCN model struggles with real-time pose data captured from the Kinect. To address this, fine-tuning the model with Kinect specific pose data could improve its accuracy. Alternatively, selecting a different pose prediction model that offers better accuracy while maintaining low latency could be a more effective solution. Given the modular design of the integrated system, replacing the pose prediction model requires minimal changes, making it simple to test and deploy new models.

Expanding the capabilities of the inference server is another area for future development. By supporting multiple "HololensClient" and "KinectServer" components, the system could align with the full-scale requirements of the existing MPC system.

The integrated system is designed to theoretically reduce latency in the MPC system. However, this claim has not been empirically validated. Establishing and testing the system with a complete configuration would provide concrete evidence of its latency reduction potential.

Lastly, the current system's use of joint positions from the output of the pose prediction model limits the quality of avatar animations, particularly for humanoid skinned mesh rendered avatars. Computing joint orientations from the predicted joint positions and using these orientations to drive avatar animations would likely result in smoother and more realistic movements.

Bibliography

[Aks+21] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. *A Spatio-temporal Transformer for 3D Human Motion Prediction*. 2021 International Conference on 3D Vision (3DV), Dec. 2021, pp. 565–574 (cit. on pp. 10, 13).

[AKH19] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. *Structured Prediction Helps 3D Human Motion Modelling*. The IEEE International Conference on Computer Vision (ICCV), Oct. 2019 (cit. on p. 10).

[AN04] Kamiar Aminian and Bijan Najafi. *Capturing human motion using body-fixed sensors: outdoor measurement and clinical applications*. Computer Animation and Virtual Worlds, 2004, pp. 79–94 (cit. on p. 6).

[Büs+24] Wolfgang Büschel, Katja Krug, Marc Satkowski, Stefan Gumhold, and Raimund Dachselt. *A Research Platform for Studying Mixed-Presence Collaboration*. IEEE International Symposium on Mixed and Augmented Reality Workshops (ISMARW), Oct. 2024 (cit. on p. 15).

[BKK18] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. *Anticipating many futures: Online human motion prediction and generation for human-robot interaction*. IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 4563–4570 (cit. on p. 10).

[CBL24] Alberto Cannavò, Francesco Bottino, and Fabrizio Lamberti. "Supporting motion-capture acting with collaborative Mixed Reality". In: *Computers and Graphics* 124 (2024), p. 104090 (cit. on p. 7).

[Che+18] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. *Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning*. 2019 International Conference on Robotics and Automation (ICRA), 2018, pp. 6015–6022 (cit. on p. 10).

[Cop+24] Adrien Coppens, Johannes Hermen, Lou Schwartz, Christian Moll, and Valérie Maquil. "Supporting Mixed-Presence Awareness across Wall-Sized Displays Using a Tracking Pipeline based on Depth Cameras". In: *Proc. ACM Hum.-Comput. Interact.* 8.EICS (June 2024) (cit. on p. 7).

[Dan+21] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. *MSR-GCN: Multi-Scale Residual Graph Convolution Networks for Human Motion Prediction*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2021, pp. 11467–11476 (cit. on p. 10).

[DB24] Divyendu Dutta and Wolfgang Büschel. *Short-term Prediction of User Motion from Live Tracking Data*. 2024 (cit. on pp. 14, 20, 22).

[Gon+11] Haifeng Gong, Jack Sim, Maxim Likhachev, and Jianbo Shi. *Multi-hypothesis motion planning for visual object tracking*. 2011 International Conference on Computer Vision, 2011, pp. 619–626 (cit. on p. 10).

[HBY22] Eunchong Ha, Gongkyu Byeon, and Sunjin Yu. "Full-Body Motion Capture-Based Virtual Reality Multi-Remote Collaboration System". In: *Applied Sciences* 12.12 (2022) (cit. on p. 7).

[Hig+21] Yosuke Higuchi, Nanxin Chen, Yuya Fujita, et al. *A Comparative Study on Non-Autoregressive Modelings for Speech-to-Text Generation*. 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2021, pp. 47–54 (cit. on p. 14).

[JPY24] Jaewoo Jeong, Daehee Park, and Kuk-Jin Yoon. *Multi-agent Long-term 3D Human Pose Forecasting via Interaction-aware Trajectory Conditioning*. 2024. arXiv: `2404.05218 [cs.CV]` (cit. on p. 50).

[Kim+14] Seungwon Kim, Gun Lee, Nobuchika Sakata, and Mark Billinghurst. "Improving co-presence with augmented visual communication cues for sharing experience through video conference". In: *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 83–92 (cit. on p. 8).

[KSS16] Koppula, Hema S., and Ashutosh Saxena. *Anticipating Human Activities Using Object Affordances for Reactive Robotic Response*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016, pp. 14–29 (cit. on p. 10).

[KGP08] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. *Motion graphs*. ACM SIGGRAPH 2008 Classes, 2008 (cit. on p. 10).

[Lam21] Alex Lamb. *A Brief Introduction to Generative Models*. 2021 (cit. on p. 10).

[LS17] Przemyslaw A Lasota and Julie A Shah. *A multiple-predictor approach to human motion prediction*. IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 2300–2307 (cit. on p. 10).

[Lev+12] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popovic, and Vladlen Koltun. *Continuous character control with low-dimensional embeddings*. ACM Transactions on Graphics (TOG), 2012, pp. 1–10 (cit. on p. 10).

[Li+23] Xuanqi Li, Xianqin Liu, Yijun Zhang, and Jianfang Hu. "Human Motion Prediction via Adaptive Fusing Autoregressive and Non-Autoregressive Attention Networks". In: *2023 6th International Conference on Software Engineering and Computer Science (CSECS)*. 2023, pp. 01–07 (cit. on p. 10).

[Liu+24] Chang Liu, Satoshi Yagi, Satoshi Yamamori, and Jun Morimoto. "Joint-Aware Transformer: An Inter-Joint Correlation Encoding Transformer for Short-Term 3D Human Motion Prediction". In: *IEEE Access* 12 (2024), pp. 156683–156693 (cit. on p. 11).

[Ma+22] 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*. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2022, pp. 6427–6436 (cit. on p. 10).

[Mah+22] Mohammad Mahdavian, Payam Nikdel, Mahdi TaherAhmadi, and Mo Chen. *STPOTR: Simultaneous Human Trajectory and Pose Prediction Using a Non-Autoregressive Transformer for Robot Following Ahead*. 2022. arXiv: `2209.07600 [cs.RO]` (cit. on p. 50).

[Mah+19] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. "AMASS: Archive of Motion Capture as Surface Shapes". In: *International Conference on Computer Vision*. Oct. 2019, pp. 5442–5451 (cit. on p. 20).

[Man+20] Karttikeya Mangalam, Ehsan Adeli, Kuan-Hui Lee, Adrien Gaidon, and Juan Carlos Niebles. *Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision*. IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 2784–2793 (cit. on p. 10).

[Mao+19] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. *Learning Trajectory Dependencies for Human Motion Prediction*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019 (cit. on p. 10).

[MBR17] Julieta Martinez, Michael J. Black, and Javier Romero. *On human motion prediction using recurrent neural networks*. Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017, July 2017, pp. 4674–4683 (cit. on p. 10).

[MVO21] Angel Martínez-González, Michael Villamizar, and Jean-Marc Odobez. *Pose Transformers (POTR): Human Motion Prediction With Non-Autoregressive Transformers*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct. 2021, pp. 2276–2284 (cit. on pp. 10, 14).

[MRM07] Gregor Mcewan, Markus Rittenbruch, and Tim Mansfield. "Understanding awareness in mixed presence collaboration". In: *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*. OZCHI '07. Adelaide, Australia: Association for Computing Machinery, 2007, pp. 171–174 (cit. on p. 9).

[NYW15] Seungtak Noh, Hui-Shyong Yeo, and Woontack Woo. "An HMD-based Mixed Reality System for Avatar-Mediated Remote Collaboration with Bare-hand Interaction". In: *ICAT-EGVE*. 2015 (cit. on p. 6).

[Nor+19] Mitchell Norman, Gun Lee, Ross T. Smith, and Mark Billinghurst. "A Mixed Presence Collaborative Mixed Reality System". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 1106–1107 (cit. on p. 8).

[Ohs+98] T. Ohshima, K. Satoh, H. Yamamoto, and H. Tamura. "AR/sup 2/Hockey: a case study of collaborative augmented reality". In: *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180)*. 1998, pp. 268–275 (cit. on p. 8).

[Oye+13] Oyewole Oyekoya, Ran Stone, William Steptoe, et al. “Supporting interoperability and presence awareness in collaborative mixed reality environments”. In: *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*. VRST ’13. Singapore: Association for Computing Machinery, 2013, pp. 165–174 (cit. on p. 7).

[PGA18] Dario Pavllo, David Grangier, and Michael Auli. *QuaterNet: A Quaternion-based Recurrent Model for Human Motion*. May 2018 (cit. on p. 10).

[Saa+24] Saeed Saadatnejad, Mehrshad Mirmohammadi, Matin Daghyani, et al. “Toward reliable human pose forecasting with uncertainty”. In: *IEEE Robotics and Automation Letters* (2024) (cit. on pp. 10, 20).

[Saa+23] Saeed Saadatnejad, Ali Rasekh, Mohammadreza Mofayezi, et al. *A generic diffusion-based approach for 3D human pose prediction in the wild*. International Conference on Robotics and Automation (ICRA), 2023 (cit. on p. 11).

[SBR19] Hossein Salimian, Stephen Brooks, and Derek Reilly. “MP Remix: Relaxed WYSIWIS Immersive Interfaces for Mixed Presence Collaboration With 3D Content”. In: *Proc. ACM Hum.-Comput. Interact.* 3.CSCW (Nov. 2019) (cit. on p. 8).

[Sco+23] Luca Scofano, Alessio Sampieri, Elisabeth Schiele, et al. *Staged Contact-Aware Global Human Motion Forecasting*. 2023. arXiv: `2309.08947 [cs.CV]` (cit. on p. 50).

[Sof+21] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. *Space-Time-Separable Graph Convolutional Network for Pose Forecasting*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021 (cit. on pp. 10, 14, 20).

[TBG05] Anthony Tang, Michael Boyle, and Saul Greenberg. “Understanding and Mitigating Display and Presence Disparity in Mixed Presence Groupware”. In: *Journal of Research and Practice in Information Technology - ACJ* 37 (Jan. 2005) (cit. on p. 9).

[TZL24] Sibo Tian, Minghui Zheng, and Xiao Liang. *Bayesian-Optimized One-Step Diffusion Model with Knowledge Distillation for Real-Time 3D Human Motion Prediction*. 2024. arXiv: `2409.12456 [cs.CV]` (cit. on p. 11).

[Tor+19] Chris Torkar, Saeed Yahyanejad, Horst Pichler, Michael W. Hofbaur, and Bernhard Rinner. *RNN-based Human Pose Prediction for Human-Robot Interaction*. Proceedings of the Joint ARW & OAGM Workshop, May 2019, pp. 76–80 (cit. on p. 10).

[UFF06] Raquel Urtasun, David J. Fleet, and Pascal V. Fua. *3D People Tracking with Gaussian Process Dynamical Models*. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), 2006, pp. 238–245 (cit. on p. 10).

[Wag+18] Fabien B Wagner, Jean-Baptiste Mignardot, Camille G Le Goff-Mignardot, et al. *Targeted neurotechnology restores walking in humans with spinal cord injury*. Nature - 563(7729), 2018, pp. 65–71 (cit. on p. 10).

[Wan+23] Xinshun Wang, Qiongjie Cui, Chen Chen, and Mengyuan Liu. *GCNext: Towards the Unity of Graph Convolutions for Human Motion Prediction*. 2023. arXiv: `2312.11850 [cs.CV]` (cit. on p. 10).

[Wei+25] Dong Wei, Huaijiang Sun, Xiaoning Sun, and Shengxiang Hu. "NeRMo: Learning Implicit Neural Representations for 3D Human Motion Prediction". In: *Computer Vision – ECCV 2024*. Cham: Springer Nature Switzerland, 2025, pp. 409–427 (cit. on p. 11).

[Wen+16] Nikolaus Wenger, Eduardo Martin Moraud, Jerome Gandar, et al. *Spatiotemporal neuromodulation therapies engaging muscle synergies improve motor control after spinal cord injury*. Nature medicine - 22(2), 2016, pp. 138–145 (cit. on p. 10).

[XML24] Chaoyue Xing, Wei Mao, and Miaomiao Liu. *Scene-aware Human Motion Forecasting via Mutual Distance Prediction*. 2024. arXiv: `2310.00615 [cs.CV]` (cit. on p. 50).

[YH09] Katsu Yamane and Jessica K. Hodgins. *Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data*. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 2510–2517 (cit. on p. 6).

[Yao+18] Li Yao, Xiaodong Peng, Yining Guo, et al. "A Data-Driven Approach for 3D Human Body Pose Reconstruction from a Kinect Sensor". In: *Journal of Physics: Conference Series* 1098.1 (Sept. 2018), p. 012024 (cit. on p. 28).

[Zai+24] Mayssa Zaier, Hazem Wannous, Hassen Drira, and Jacques Boonaert. "Motion-Lie Transformer: Geometric Attention For 3D Human Pose Motion Prediction". In: Oct. 2024, pp. 2515–2521 (cit. on p. 11).

[Zha+24] Haichuan Zhao, Xudong Ru, Peng Du, et al. "High-Quality Human Motion Prediction Using Size Invariant Motion Space". In: Oct. 2024 (cit. on pp. 11, 47).

[ZL16] Xudong Zhu and Kin Fun Li. *Real-Time Motion Capture: An Overview*. 2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), 2016, pp. 522–525 (cit. on p. 6).

Webpages

[@AA] Abdenour Amamra and Nabil Aouf. *Real-Time Robust Tracking of Moving Robots with Multiple RGBD Consumer Cameras*. URL: `https://arxiv.org/ftp/arxiv/papers/2110/2110.15815.pdf` (cit. on p. 12).

[@Dut24] Divyendu Dutta. *Masters Thesis: Human-Pose Prediction for Low-Latency Mixed-Presence Collaboration*. 2024. URL: `https://git.imld.de/teaching/theses/ma-divyendu-dutta-code` (cit. on p. 22).

[@Int] Perceiving Systems Department MPI for Intelligent Systems. *SMPL made simple FAQs*. URL: https://files.is.tue.mpg.de/black/talks/SMPL-made-simple-FAQs.pdf (cit. on pp. 20, 27).

[@Mic19] Microsoft. *Azure Kinect body tracking joints*. 2019. URL: https://learn.microsoft.com/en-us/previous-versions/azure/kinect-dk/body-joints (cit. on p. 27).

[@Rap14] Ilana Rapp. *Motion Capture Actors: Body Movement Tells the Story*. 2014. URL: https://web.archive.org/web/20140703113656/http://www.nycastings.com/dmxreadyv2/blogmanager/v3_blogmanager.asp?post=motioncaptureactors (visited on July 3, 2014) (cit. on p. 6).

[@Saa+] Saeed Saadatnejad, Mehrshad Mirmohammadi, Matin Daghyani, et al. *UnPOSed*. URL: https://github.com/vita-epfl/unposed (cit. on p. 20).

List of Figures

3.1 Minimal depiction of the existing MPC system. One "KinectServer" component and two "HololensClient" components. 16

3.2 UML diagram showing existing class relationships for Kinect data handling. 17

3.3 UML diagram showing added pose data saving functionality. 18

3.4 UML diagram showing showing added pose replay integration into existing system. 19

3.5 End-to-End System Architecture : Integration of pose prediction model into the existing MPC system. 23

3.6 KinectServer and HololensClient equipped with InferenceServerDiscoverer. 23

3.7 Kinect vs SMPL/AMASS joint hierarchy. 27

3.8 Stick figure visualization of pose data on "HololensClient". 32

3.9 Model comparison: Ground truth in Green, Pose prediction model in blue, Zero-velocity model in Red. 34

4.1 Stick figure avatar depicting the predicted hand-waving motion from live-tracked Kinect data. 40

4.2 Stick figure avatar depicting the predicted running motion from saved pose Kinect data. 41

4.3 Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames. Source : **Live-tracked Kinect data of a person raising both arms.** Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model. 43

4.4 Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames. Source : **Slowly waving one hand saved pose data.** Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model. 44

4.5 Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames. Source : **Quickly waving both hands saved pose data.** Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model. 45

4.6 Comparison of the pose prediction model with ground truth and zero-velocity baseline model across three frames. Source : **Quickly running saved pose data.** Green: Ground truth, Blue: Pose prediction model, Red: Zero-velocity model. 45

4.7 Superimposed model comparison: Ground truth in Green, Pose prediction model in blue, Zero-velocity model in Red. Top: **Slowly waving one hand saved pose data.** Middle: **Quickly waving both hands saved pose data.** Bottom: **Quickly running saved pose data.** 46

4.8 Motion analysis of pelvis joint of a subject from AMASS dataset. 49

4.9 Motion analysis of left foot joint of a subject from AMASS dataset. 50

List of Tables

3.1 Relevant key aspects of the STS-GCN model. 20

4.1 Average displacement error over 200 frames in meters (m). **Predicted** is the Integrated Pose Prediction Model vs Ground Truth. **Zero-Vel** is the Zero-Velocity Baseline Model vs Ground Truth. 47

Colophon

This thesis was typeset with $\LaTeX\,2_{\varepsilon}$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I hereby certify that I have authored this document entitled "Human-Pose Prediction for Low-Latency Mixed-Presence Collaboration" independently and without undue assistance from third parties. No other than the resources and references indicated in this document have been used. There were no additional persons involved in the intellectual preparation of the present document other than mentioned in the acknowledgment section. I am aware that violations of this declaration may lead to subsequent withdrawal of the master's thesis.

Dresden, December 01, 2024

Divyendu Dutta

Digitally signed by Divyendu Dutta
Date: 2024.12.01
23:09:48 +01'00'

Divyendu Dutta