Enhancing Socially Aware Navigation of Robots in Healthcare Environments through Predicting Human Trajectories

Efstratios Mytaros[1], **Bob Schadenberg**[1], **and Gwenn Englebienne**[1]

[1]**University of Twente**

ABSTRACT

This research explores the development and evaluation of a GRU-based trajectory prediction model aimed at forecasting human walking paths in enclosed spaces, such as hospitals. Utilizing datasets THOR and MAGNI, the study involved meticulous preprocessing steps, including the transformation of Cartesian coordinates to polar coordinates and handling missing values. The model's performance was assessed using key metrics like Root Mean Squared Error (RMSE) and Euclidean distance, with a focus on both positional and angular accuracies. Results indicated that while the model performs well in familiar environments, it struggles with generalization to unseen scenarios. The study also highlights the need for more diverse training data and real-world validation to enhance model robustness. Future steps include refining preprocessing techniques, optimizing computational efficiency, and integrating multimodal data to improve trajectory prediction accuracy in dynamic indoor environments..

Keywords: Human prediction process, human-robot interactions, machine learning

1 INTRODUCTION

1.1 Background and Motivation

The integration of socially aware robots in healthcare environments is becoming increasingly important due to their potential to enhance patient care, improve operational efficiency, and reduce the workload on healthcare staff. These robots must be capable of navigating complex and dynamic hospital environments, where predicting human walking trajectories is essential for ensuring safe and efficient movement. Despite significant advancements, current trajectory forecasting solutions face several limitations, including the inability to accurately predict human movements in enclosed spaces and a lack of suitable datasets tailored for such tasks.

Healthcare environments can be complex and dynamic, with a constant flow of patients, medical staff, and equipment. Robots operating in these environments need to navigate safely and efficiently and interacting harmoniously with humans. Predicting human predictions in their navigational patterns becomes more relevant as it allows the robot to anticipate human behavior, understand their intentions, and respond appropriately. This predictive capability enables the robot to proactively adjust its navigation strategies, such as path planning and obstacle avoidance, to ensure smooth and collision-free interactions with humans.

1.2 Problem Statement

Efficient and socially aware navigation aids robots to seamlessly interact with people in closed spaces, such as hospital corridors. However, current robot navigation systems (1)(2)(3) face challenges in autonomously navigating through these environments and accurately predicting human actions and reactions. Among other things, one of the frequently encountered mentioned drawbacks of socially aware robot navigation is the lack of an internal model reflecting the intentions of the individuals encountered limits the robot's ability to engage harmoniously and safely with humans.

Existing datasets for human trajectories often lack the specificity and quality required for healthcare settings. The THOR(4) and MAGNI(5) datasets , however, provide a robust foundation for this research. The THOR dataset serves as an effective test dataset, while the MAGNI dataset is used for training purposes. Both datasets offer rich and detailed trajectory information that is critical for developing and testing predictive models in a hospital context.

A critical aspect of utilizing these datasets is the preprocessing of trajectory data to make it suitable for forecasting tasks within a robotics context. This involves handling missing data, normalizing trajectories, and transforming the data into formats that are compatible with machine learning models. Proper preprocessing ensures that the data accurately represents the movement patterns of people active in the scene, which is essential for training effective predictive models.

Machine learning (ML) models such as Multilayer Perceptrons (MLPs), Gated Recurrent Units (GRUs), and Transformer networks offer significant advantages for trajectory forecasting, as they also provide links to human behaviour (6). MLPs provide a straightforward approach with their fully connected layers, making them suitable for initial experiments. GRUs, with their ability to handle sequential data, are particularly effective for capturing temporal dependencies in human movement. Transformers, known for their attention mechanisms, offer a powerful means to model long-range dependencies and complex interactions within the data.

1.3 Objectives

The objective of this research is to structure the THOR and MAGNI datasets in a manner that enhances their suitability for trajectory forecasting tasks and to employ appropriate machine learning models to generate predictions that closely match the ground truth. By focusing on data preprocessing and leveraging advanced predictive models, this study aims to bridge the gap in current trajectory forecasting solutions and contribute to the development of socially aware robots in healthcare settings.

The main research questions this research aims to answer shall be:

- What are the necessary pre-processing steps in order to use the full potential of a trajectory dataset such as THOR and MAGNI?
- How do the resulting ML model predictions compare to the ground truth?
- How do different ML models compare in predicting human navigation trajectories?

1.4 Structure of the Thesis

This thesis is structured as follows: Chapter 2 reviews the relevant literature on trajectory forecasting and machine learning models. Chapter 3 describes the THOR and MAGNI datasets in detail. Chapter 4 outlines the preprocessing steps applied to the datasets. Chapter 5 discusses the model selection and implementation. Chapter 6 presents the experimental results and evaluation. Chapter 7 provides an analysis of the findings. Finally, Chapter 8 concludes the thesis with a summary of contributions, limitations, and suggestions for future work.

2 BACKGROUND LITERATURE

2.1 Human Prediction Process

Various theories have been developed that describe the way that the human mind generates predictions about their surroundings. Although none of them have been accepted as the global truth about the core functions of the brain, they all contain some degree of truth to them. The goal is to determine to what extent we can use aspects of these theories in our predictive models to fit the problem at hand. Throughout this paper, references to intentions and predictions will be made. Predictability refers to the ability to forecast or anticipate future events or outcomes with a degree of confidence based on available information. Intent is the consciously formed and purposeful mental state that drives an individual's planned actions or decisions. Depending on the predictability of some scenario, an agent's intent may change, hence these terms will often be used around each other.

In a unified theory, the FEP tries to establish a base theoretical framework for future model development (7). Modern predictive processing, theory of mind and cognitive function approaches share a common ground in focusing on the top-down error minimising prediction of information processing Radical predictive processing envisions a massively predictive brain, where neural encodings describe predictions or prediction errors in an inter-connected hierarchy. It can be seen as an emphasis on the global top-down cascade of predictions across neural hierarchies. Embodied predictive coding varies from extensions of comparator based approaches to treatments couched in dynamical systems theory and enactivism. The

FEP addresses the reason the brain must engage in embodied predictive processing in order to maintain its enactive integrity. Through the use of Bayesian models in describing decision making, the issue of prior beliefs can get stuck in an origin loop. Similarly, reinforcement learning's cost functions describe an optimal behaviour is defined by whatever an agent chooses to do. Through cell's self-creation, or autopoiesis, the grounds for 'visitable' state availabilities are set and restricted, thus fundamentally trying to preserve such a boundary through the so called Markov blanket, separating it from its surroundings.

Current implementations of such theories in robotic models (1)(2)(3) lack the focus and specificity that enable complex behaviours. Most models of human intentions and predicting human navigation are built around Bayesian models, or Boltzmann-type (optimum action-seeking) behaviours, which although closely linked to literature, don't always capture of the intricacies of human behaviour. Human behaviour is also influenced by internal motivations depending on their set goals and current environmental influences, such as dynamic obstacles, personal distractions and social factors (small-talk interactions, personal space, giving priorities to staff members etc.). Different explicit, or implicit models have been used, but none that have found a balance between socio-cognitive rules dictating human behaviour, and the generative, emergent and spontaneous decisions that humans are capable of.

Overall, the aim is to mimic, or at least closely replicate this human prediction process, by inclusing a hierarchical processing structure that results in predictions. Proper encodings of the world would need to be established, as well as a model architecture that can capture the complexities associated with human navigation. Next, we will be looking at what some common trajectory prediction approaches have been.

2.2 Trajectory prediction models

The models that have currently been in use to determine people's perspectives on a scene, or predict their behaviour through internal modelling of other's predictions have been varied, as already seen from section 2.1. In this section the algorithmic models of human prediction processes will be investigated and related to their corresponding contexts.

2.2.1 Cognitive science models

Given the background of predictive processing and the FEP, certain cognitive science models warrant further investigation. Initial research identifies key structures and algorithms related to the theories discussed in 2.1.

A study on cognitive science models (6) highlights deep learning's importance, linking neural networks and cognition. The free energy principle's variational Bayes approach relates to Variational Autoencoders, though deep learning's applications in cognitive science extend beyond this. Neural networks, originally not designed for cognition studies, evolved from Parallel Distributed Processing (PDP) introduced in the 80s, featuring layers and backpropagation. Deep CNNs excel in vision, LSTMs in natural language processing, and reinforcement learning in action selection. Bayesian analysis frequently appears in models guiding predictions and behavior (8).

Various models apply to different scenarios. For instance, human learning models (9) use a generative model for predictions, a reinforcement learning model for parameter adjustment, and an active inference process for behavior guidance, aligning with deep learning applications in cognitive science.

Deep learning models also aid in studying human strategic behavior, involving prediction and intention inference. While not directly related to navigation, these predictive approaches are valuable. Neural network models benefit from encoding domain knowledge, and feedforward networks combat overfitting through data augmentations. In (2), invariance is reinforced by model architecture, using "pooling units" for information sharing, similar to MLP convolution layers. Pooling illustrates the power of information sharing, even if not directly adapted.

Combining models can explore abstractions within selected architectures. Flexible yet interpretable models are ideal for capturing human behavior. In (10), an exploratory DNN model based on LSTM architecture was developed to model temporal behavior, incorporating feedback connections. A Q-learning model focused on reward maximization, and a reward-oblivious LSTM captured sequential action patterns. The exploratory DNN outperformed the reward-oriented model, showing predictable non-reward-driven action patterns in uncertain scenarios.

2.2.2 Contextual trajectory prediction models

A few studies have directly tried to model human trajectories in various contexts. Though the primary contextual focus of these studies is generally modelling the way people navigate around other people, without the explicit inclusion of robots in the scene, a lot can be extracted from their approach. Looking at the ways that current state-of-the-art methods have attempted to tackle this problem, a pattern of incorporating additional social-navigation information have been adapted.

One common attribute across the examined papers is that they all use a similar structure for the selected model, the variational autoencoder (VAE). One of the studies (11) adds a social pooling layer into their VAE model, along with the pedestrian's goals and trajectories. In this way, the forecasted trajectory is taking the people's neighbours into account, as well as their potential goals. Whereas in this case the neighbours were explicitly mapped in the scene and inferred some rules through social pooling.

Other studies also adapted a similar approach, where social aspects were taken into account, but were rather modelled as social physics (12) that would dictate the people's behaviours as 2D particles. In this case people's trajectories are still learned through a VAE, but are refined through a social physics model that follows Newton's second law of motion. Although this is a simplistic representation of social dynamics, it shows quite good results when guessing people's trajectories in a scene.

Lastly, another study that used VAEs for trajectory prediction has shown that no explicit social rules need to be present, but rather a social-implicit implementation (13). This study leverages the power of Average Mahalanobis Distance (AMD) and Average Maximum Eigenvalue (AMV) to generate accurate trajectories. Each of these techniques quantifies how close the whole generated samples are to the ground truth and the overall spread of the predictions, hence aiming for a balance between broad generations of trajectories and closely related paths to the ground truth.

2.2.3 Models used in pedestrian interactions

Exploring pedestrian interactions and trajectory prediction models reveals widely-used techniques in well-researched fields. Various models for pedestrian trajectory prediction are discussed in the literature (1), including those based on behavioral empirical science and machine learning. These predictions, which consider goals, obstacles, state variables, and road geometry, can benefit from combining empirical results with machine learning to avoid over-fitting.

Behavior prediction with unknown goals has several applications. Pedestrian motion can be modeled by MDPs with immediate rewards for chosen paths, and Bayesian methods can infer destinations by computing prior distributions from observed trajectories. Previous implementations used deep learning and inverse reinforcement learning to infer pedestrian goals. Multi-modal trajectory forecasting in structured environments employed CNNs for reward maps and RNNs combined with track history to predict future trajectories. GANs, which naturally handle uncertainty and multi-modality, have also been used to predict short-term pedestrian motion. Another approach involved using a conditional VAE to generate future trajectories, with an RNN scoring features in an inverse optimal control manner.

Pedestrian interaction models, focusing on individual behavior predictors, include proxemics (utility of proximity zones), physical models (social forces and desired velocities), cellular models (discrete, time-based modeling on cell grids), and queuing network models (evacuation dynamics and Monte Carlo simulations). When evaluating pedestrian intent through pose estimation, RNN-based models such as LSTMs, GRUs, and transformers were found most suitable, with GRUs and transformers performing best in accuracy and F1 scores (14).

2.2.4 Summary

The background of predictive processing and the FEP suggests the importance of cognitive science models, particularly those integrating deep learning techniques, for further investigation. Initial research highlights key structures and algorithms, such as neural networks, CNNs, LSTMs, and reinforcement learning models, in studying cognition and predicting behavior (6).

Neural networks, specifically evolved from Parallel Distributed Processing (PDP) models, excel in various applications: CNNs in vision, LSTMs in natural language processing, and reinforcement learning in action selection. Bayesian analysis frequently supports these models (8). Models for human learning and strategic behavior also demonstrate the value of deep learning applications in cognitive science (9; 2).

MLPs, LSTMs, GRUs, and transformers are particularly suitable for trajectory predictions due to their unique strengths:

- **MLPs (Multi-Layer Perceptrons):** Capable of learning complex patterns and relationships in data, MLPs are suitable for initial trajectory prediction tasks. They provide a foundational understanding of data structures and can serve as a baseline for more advanced models. Their simplicity and ease of implementation make them a practical choice for straightforward trajectory prediction scenarios.

- **LSTMs (Long Short-Term Memory networks):** Excellent for modeling temporal dependencies and long-range interactions in sequential data, LSTMs are crucial for accurate trajectory prediction. They effectively manage the vanishing gradient problem, making them ideal for capturing the temporal dynamics inherent in trajectory data, such as predicting a pedestrian's future path based on their movement history.

- **GRUs (Gated Recurrent Units):** Similar to LSTMs but with a more streamlined architecture, GRUs offer efficient training and are effective in capturing temporal dependencies in trajectory data. Their simpler gating mechanism reduces computational complexity while maintaining performance, making them a robust choice for real-time trajectory prediction applications.

- **Transformers:** Provide state-of-the-art performance in sequence modeling due to their self-attention mechanisms, making them highly effective for capturing complex dependencies in trajectory predictions. Transformers excel in handling large-scale data and parallel processing, which is beneficial for predicting trajectories in crowded environments where interactions between multiple agents must be considered.

Combining these models can leverage their individual strengths, enabling more accurate and interpretable predictions. For example, MLPs can be used for initial data processing and feature extraction, LSTMs and GRUs can handle temporal dependencies, and transformers can capture intricate dependencies and interactions.

The benefits and use cases of these models include:

- **MLPs:** Useful for baseline predictions and understanding basic trajectory patterns, especially in scenarios with less complex data.

- **LSTMs:** Ideal for predicting trajectories over time, such as pedestrian movement, where historical data significantly influences future paths.

- **GRUs:** Suitable for applications requiring efficient real-time predictions, such as autonomous vehicle navigation in dynamic environments.

- **Transformers:** Best for complex, large-scale trajectory prediction tasks, such as predicting the movement of multiple agents in dense urban settings, due to their ability to handle vast amounts of data and capture multifaceted interactions.

In summary, integrating MLPs, LSTMs, GRUs, and transformers allows for the development of sophisticated models that can accurately predict trajectories by leveraging the unique strengths of each approach. This combination enhances model performance and interpretability, making it possible to understand and anticipate human behavior in various dynamic contexts.

2.3 Data Collection

The initial step in developing a socially aware navigation system for robots in healthcare involves collecting human trajectory data. The THOR dataset (4) provides exclusive indoors footage of a fixed space. In some scenes, additional obstacles were placed in the environment, allowing for variations in the environmental conditions. There were also people placed in the scene, some of which had fixed paths and tasks, whereas others were more freely roaming through the scene. Both video and Bags files are included in the dataset and give plenty of data to work with. An additional remark is that the data was recorded through a small forklift Linde CitiTruck robot with a footprint of 1.56 x 0.55 meter and 1.17 meter high. The main focus is on using this dataset for the fine-tuning of the generative model to indoor HRI. This is the primary dataset that will be considered, since it provides enough interactions in an indoors environment, in combination with different scenarios where only humans are navigating the scene, and one where the humans are navigating the scene with a robot present as well. In case that this dataset is not sufficient for any reason, there are quite a few alternatives.

The OpenTraj (15) dataset contains comprehensive information regarding human trajectories across various outdoors spatial contexts. Each frame of the dataset's videos is annotated with labels identifying individuals, including interactions between individuals, individuals and their environment, as well as interactions between individuals and other dynamic agents within the environment. Most of this data comes from an outdoors context, which is not a direct fit to what we want to apply our model to. Such a contextual limitation can have an impact on the fidelity of the results, especially when applied to a different context. However, due to the variety in the data available and plethora of different behaviours, the samples are more than enough to get a good idea of how people navigate their environment.

Constructing a navigational dataset from scratch requires thorough preparation, setting up and permissions from the ethics committee. Suitable equipment for this purpose would be a robot that can serve the purpose of this experiment, such as the Teresa, or the Kuka-iDo robots. They are both able to navigate the environment in question and offer good integration with the ROS (16) platform, as well as be equipped with enough sensors to interact with their environment. Moreover, they are available to use and provided by the university of Twente. Some of the scenarios that would be interesting to explore are: door interactions, following/guiding individuals, corner turn surprises, driving close to people, overtaking, yielding priority, and lane keeping. Suitable environments can be found all around the campus, with corridors and indoor environments facilitating interactions that can simulate the different scenarios quite well. For all this to be possible, ethical recruitment of participants would be needed, where the setup of the experiments would abide by the ethical rules of the university. Participants would therefore need to be appropriately recruited and instructed on what to do. The setup would consist of a fixed area within the university buildings, where the robot would be remotely controlled by the researchers and carrying out some predetermined tasks within the environment. The participants would then have to also carry out their own personally assigned tasks, designed in such a way that they will be forced to come in contact with the robot in some way.

On the other hand, there is also the possibility of using an existing dataset, specifically designed for robot socially aware indoors navigation. One of the primary examples is the MuSoHu dataset (17), containing scenes from a first-person point of view video and lidar data, compatible with ROS. The data was recorded by a human navigating both outdoor and indoor interactions with humans. The main purpose of this dataset is to document human navigation scenarios and assist the development of robot socially aware navigation algorithms. Enough data has been collected that can be used for training of the model through either video, or lidar sensors. The downside of that is that the footage is recorded by a human, hence making the interactions less representative of the goal of this study.

More datasets of a similar nature exist, with the advantage of being recorded by a robot instead of a human. One of the more extensively used datasets is the SCAND collection (18). The footage was recorded by a remote-controlled Clearpath Jackal and a legged Boston Dynamics Spot. There is a plethora of different scenes where the robots are navigating through variable crowd densities in both indoor and outdoor environments. This dataset also offers ROS integration tools, for visualising the human movements in space, as well as interacting with them.

All these serve as potential datasets that could be used. However, the one that is selected as a starting point is the THOR dataset, as it provides the most suitable collection for our research purposes.

2.4 Evaluation Metrics

Evaluating trajectory prediction models is crucial to ensure their accuracy and reliability in various applications, such as autonomous driving and pedestrian movement analysis. Several metrics are commonly used to assess the performance of these models, each providing insights into different aspects of prediction quality.

2.4.1 Average Displacement Error (ADE)

Average Displacement Error (ADE) is a widely used metric that calculates the average Euclidean distance between predicted trajectory points and the ground truth points over the entire prediction horizon. It is defined as:

$$ADE = \frac{1}{T}\sum_{t=1}^{T} \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|$$

where T is the total number of time steps, $\mathbf{y}_t$ is the ground truth position at time t, and $\hat{\mathbf{y}}_t$ is the predicted position at time t. ADE provides a general measure of overall prediction accuracy (19; 20).

2.4.2 Final Displacement Error (FDE)

Final Displacement Error (FDE) measures the Euclidean distance between the predicted final position and the ground truth final position at the end of the prediction horizon. It is defined as:

$$FDE = \|\mathbf{y}_T - \hat{\mathbf{y}}_T\|$$

FDE focuses on the accuracy of the final predicted position, which is particularly important in applications where the end location is critical (19; 20).

2.4.3 Minimum Final Displacement Error (minFDE$_l$)

Minimum Final Displacement Error (minFDE$_l$) evaluates the best prediction among the top l generated trajectories at the final time step T. This metric accounts for the model's ability to generate multiple plausible futures and selects the one closest to the ground truth:

$$minFDE_l = \min_{i=1}^{l} \left\|\mathbf{y}_T - \hat{\mathbf{y}}_T^i\right\|$$

This approach is useful in multi-modal prediction scenarios where multiple future paths are considered (19; 21).

2.4.4 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is another commonly used metric that measures the square root of the average squared differences between predicted and actual trajectory points:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\mathbf{y}_t - \hat{\mathbf{y}}_t)^2}$$

RMSE is sensitive to large errors and provides a measure of the overall prediction accuracy, making it suitable for various trajectory prediction tasks (22).

2.4.5 Euclidean Distance

Euclidean Distance is often used in trajectory prediction to measure the direct spatial difference between predicted and ground truth points. It is a fundamental metric that underpins both ADE and FDE, providing a straightforward way to assess prediction accuracy (19).

2.4.6 Use Cases

Autonomous Driving: In autonomous driving, accurate trajectory predictions are essential for safe navigation and collision avoidance. Metrics like FDE and minFDE$_l$ are critical as they help in evaluating the end positions of predicted paths, which is crucial for planning safe maneuvers in dynamic environments (22).

Pedestrian Movement Analysis: For pedestrian trajectory prediction, metrics such as ADE and RMSE are commonly used to evaluate the accuracy of predictions over time. These metrics help in understanding how well the model can predict the continuous movement of pedestrians in various scenarios (19; 21).

Cross-dataset Evaluation: Evaluating models across different datasets ensures their robustness and generalizability. Metrics like ADE, FDE, and RMSE are used to compare model performance across various datasets, highlighting their effectiveness in different contexts (20).

These evaluation metrics are particularly suitable for human trajectory predictions in indoor hospital environments. Hospitals are dynamic and complex settings with frequent pedestrian movement, including staff, patients, and visitors. Metrics like ADE and FDE are crucial for ensuring that trajectory predictions are accurate and reliable, thereby facilitating smooth and safe navigation. In scenarios where multiple potential paths need to be considered, minFDE$_l$ helps in evaluating the most probable future trajectories. RMSE and Euclidean Distance provide additional layers of accuracy assessment, ensuring that models can handle the variability and unpredictability of indoor hospital environments effectively. Accurate trajectory predictions in hospitals can improve the efficiency of autonomous robots for delivery tasks, enhance patient monitoring systems, and contribute to overall safety and operational effectiveness.

3 DATASET DESCRIPTION

3.1 Introduction to THOR and MAGNI Datasets

The THOR and MAGNI datasets are pivotal resources for research in robotics and autonomous systems. These datasets provide comprehensive data essential for developing and testing algorithms in areas such as trajectory prediction, path planning, and autonomous navigation.

3.1.1 Overview of the Datasets and Their Sources

The **THOR** (The Humanoids at Örebro) dataset is curated by Örebro University and comprises extensive sensory data collected from humanoid robots. It includes diverse scenarios and environments, aiming to facilitate research in human-robot interaction, robotic perception, and autonomous decision-making. More information about the THOR dataset can be found at http://thor.oru.se/thor.html.

The **MAGNI** dataset, also from Örebro University, focuses on autonomous ground vehicles operating in various outdoor environments. This dataset includes high-resolution sensory inputs and accurate ground truth data, supporting research in autonomous driving, obstacle detection, and path optimization. Details about the MAGNI dataset are available at http://thor.oru.se/magni.html.

3.2 Dataset Characteristics

3.2.1 Size, Format, and Key Attributes of the Datasets

Both datasets are extensive and structured to support a wide range of research activities. However, when it comes to training and testing our models, we want to use a sample, where no robot is present in the scene.

THOR Dataset: Out of the different scenarios present in the THOR dataset, the one we are the most interested in, is the first experiment, where all human agents interact, without the interference of a robot agent. From now on, this will be considered the main source of data from the THOR dataset.

- **Size:** The THOR dataset includes several terabytes of data overall. There are 1068948 frames (0.1 seconds per frame) that are useful to us.

- **Format:** The data is provided in various formats, including ROS (Robot Operating System) bag files, CSV files, and image sequences.

- **Key Attributes:** The dataset contains synchronized sensory inputs from multiple sensors, including RGB-D cameras, LiDAR, inertial measurement units (IMUs), and microphones. Key attributes also include timestamps, sensor calibration data, and metadata describing the environmental context.

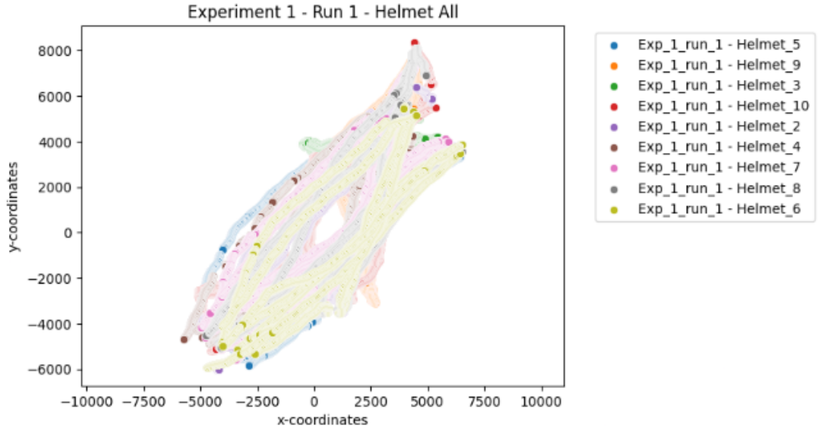

Figure 1. All THOR dataset trajectories visualised

MAGNI Dataset:

- **Size:** The MAGNI dataset also encompasses several terabytes of data. There are 2896285 frames (0.1 seconds per frame) that are useful to us.

- **Format:** Similar to the THOR dataset, MAGNI data is available in ROS bag files, CSV files, and image sequences.

- **Key Attributes:** This dataset features high-resolution sensory data from LiDAR, GPS, cameras, and IMUs. Key attributes include timestamps, ground truth position data, and environmental descriptors.

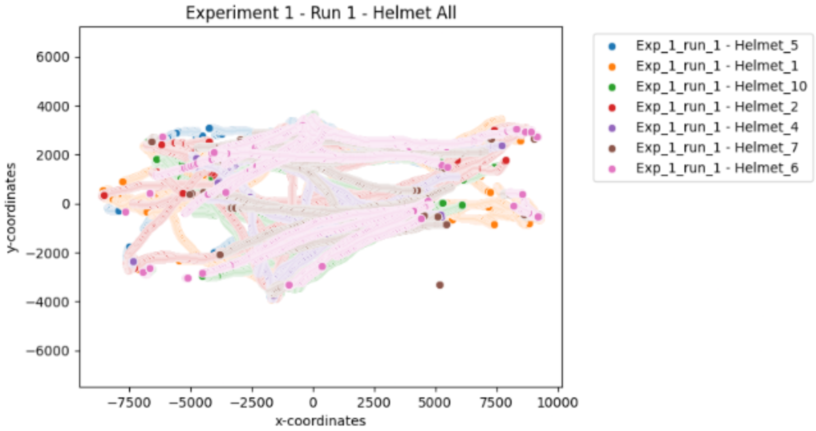

Figure 2. All MAGNI dataset trajectories visualised

3.3 Data Collection Methodology

3.3.1 How the Data was Collected and Any Preprocessing Steps Performed by the Dataset Providers

Both datasets were collected from the same university, ensuring consistency and alignment to the goals and purpose of their intented use.

THOR Dataset:

- **Data Collection:** Data for the THOR dataset was collected using humans equipped with a suite of sensors. The participants were deployed in various indoor environments, capturing interactions with each other and navigating through different scenarios. Data collection was conducted under controlled conditions to ensure consistency and accuracy.

- **Preprocessing:** The dataset providers performed initial preprocessing steps, such as sensor calibration, synchronization of multi-sensor data, and noise reduction. This preprocessing ensures that the dataset is ready for immediate use in research applications.

MAGNI Dataset:

- **Data Collection:** The MAGNI dataset was gathered using autonomous ground vehicles and humans equipped with tracking sensors operating in indoor environments. Data collection included various paths and trajectories of humans, based on assigned roles.

- **Preprocessing:** Preprocessing steps for the MAGNI dataset included sensor calibration, data synchronization, and filtering to remove erroneous readings. The dataset providers also ensured the alignment of sensory data with ground truth information, enhancing the reliability of the dataset for research purposes.

The detailed and carefully curated THOR and MAGNI datasets serve as invaluable resources for advancing research in robotics and autonomous systems. Their comprehensive nature and high-quality data support a wide range of applications, from developing new algorithms to testing and validating autonomous behaviors. For the purpose of our research, we are planning on using the MAGNI dataset for training and the THOR dataset for testing due to their respective sizes.

4 METHODOLOGY

4.1 Data pre-processing

4.1.1 Initial Data Exploration

The initial step in the data preprocessing phase involved a thorough exploration of the THOR and MAGNI datasets to understand their structure, contents, and characteristics. This exploration included examining the dimensions of the datasets, primarily the structure of the trajectories. Visualizing the data through the plotting of the cartesian coordinates helped in identifying distributions and trends. These visualizations were crucial in revealing initial patterns, and potential issues such as outliers or missing values, which informed subsequent cleaning steps.

The primary observation was that there were a plethora of cases where values from the trajectories were missing. This could be due to multiple reasons, the most common ones being two. One was the trajectory reached the end of the sensor's observing range, hence leading to no values being recorded. The second was due to interference, or faulty receiving within the observable area. Both these cases yield values that are either perceived as NaN, or 0.0 coordinates. This would lead to the resulting plotted trajectories to seemingly "teleport" to the 0.0 coordinates every so often, which was not ideal.

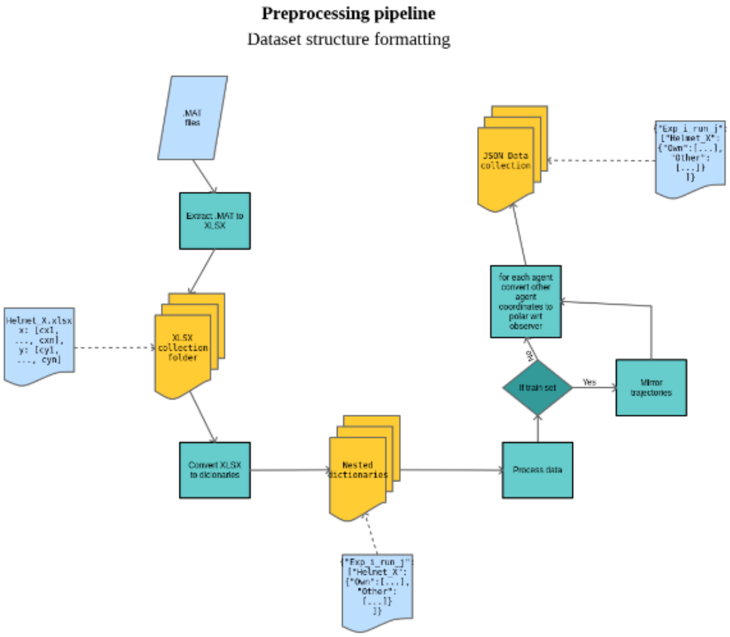
Preprocessing pipeline
Dataset structure formatting
.MAT files
Extract .MAT to XLSX
Helmet_X.xlsx x: [cx1, ..., cxn], y: [cy1, ..., cyn]
XLSX collection folder
Convert XLSX to dicionaries
Nested dictionaries
{"Exp_i_run_j": ["Helmet_X": {"Own":[...], "Other": [...]}]}
Process data
If train set
Yes
No
Mirror trajectories
for each agent convert other agent coordinates to polar wrt observer
JSON Data collection
{"Exp_i_run_j": ["Helmet_X": {"Own":[...], "Other": [...]}]}

Figure 3. A diagram of the initial preprocessing pipeline for transitioning from the MATLAB format to a more structured time-series forecasting form.

Figure 3 illustrates how the trajectories were extracted from their original MATLAB format (.mat) and converted into sheets for easier manual inspection and then into JSON format for exporting and labelling of individual experiment runs and agent trajectory titles.

After observing the different experiments and runs of each for both datasets, a few slight differences were noted between them. Although the THOR and MAGNI datasets are in essence carried out in the same environment for each experiment, the obstacles around the environment, as well as the goals (only for the MAGNI dataset) would shift across experiments. For this reason we decided to only use one of the ones where the environment stayed consistent, considering that the obstacles were not encoded and provided to the model in any way.

Moreover, the data needed to be represented in a way that's more suitable for trajectory predictions, more specifically in polar coordinates. The agents' trajectories were thus represented as polar with respect to the observer agent. These were then stacked next to the observer for the association to be clear. This leads us to tables of trajectories with the observer agent represented in cartesian coordinates and all other agents and goals around them in polar with respect to their position in each time-step.

Additionally, during the exploration of the data, and particularly the THOR dataset, it was observed that there were no goals provided for the agents in the scene. Therefore, the goals needed to be inferred, if they were to be used properly as inputs during inference. In order to extract the goals, detecting the most likely points of interest of the agents was required. Our approach involved extracting the end-points of the map, hence where the agents would exit the scene and subsequently reach their goals, as well as the points where the agents would remain stationary for longer periods of time. After ensuring that these sets of trajectories were collected, a K-means clustering method was applied to assign the locations of the extracted goals. The complete process is outlined in figure 4, where the structuring and extracting features from the data can be seen.

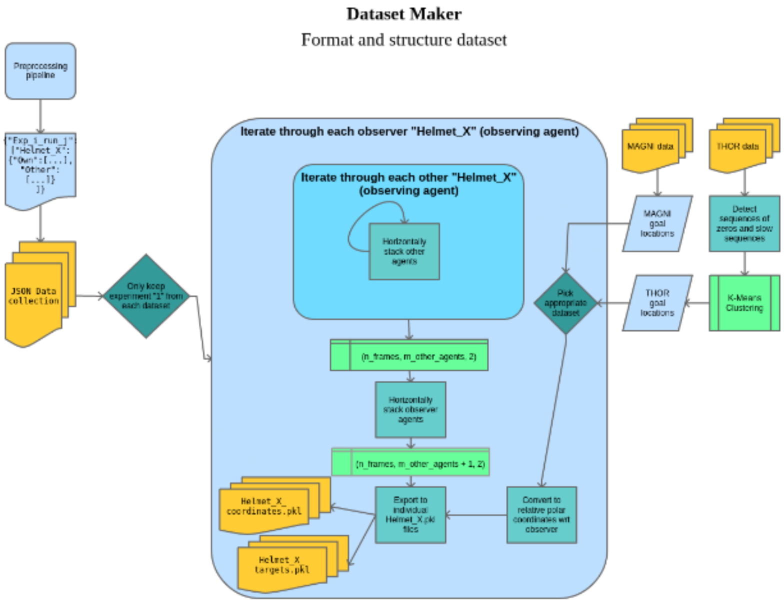
Dataset Maker
Format and structure dataset
Preprocessing pipeline
{"Exp_i_run_j":{"Helmet_X":{"Own":[...], "Other": [...]}]}
JSON Data collection
Only keep experiment "1" from each dataset
Iterate through each observer "Helmet_X" (observing agent)
Iterate through each other "Helmet_X" (observing agent)
Horizontally stack other agents
(n_frames, m_other_agents, 2)
Horizontally stack observer agents
(n_frames, m_other_agents + 1, 2)
Export to individual Helmet_X.pkl files
Helmet_X_coordinates.pkl
Helmet_X_targets.pkl
Convert to relative polar coordinates wrt observer
Pick appropriate dataset
MAGNI data
THOR data
MAGNI goal locations
Detect sequences of zeros and slow sequences
THOR goal locations
K-Means Clustering

Figure 4. A diagram outlining the process of extracting data from the scene and formatting it in a useful way.

4.1.2 Data Transformation

To ensure the quality and integrity of the datasets, the data cleaning process addressed errors and inconsistencies. Missing values were identified and managed through strategies such as interpolation, or by removing the incomplete entries, depending on the extent and impact of the missing data on the analysis. Radial basis function (RBF) interpolation is a technique used to estimate values at unknown points by leveraging the known values at surrounding points. This method assumes that the value at any given point can be interpolated based on a weighted average of nearby points, where the weights are determined by a radial basis function applied to the distance from the point of interest. RBF interpolation is particularly useful in spatial analysis and trajectory prediction, as it allows for smooth transitions and realistic approximations of paths or surfaces, ensuring continuity and consistency in the estimated data. The radial basis function, typically Gaussian, multiquadric, or thin-plate spline, transforms the distances into weights that influence the interpolation. By considering the distances between points, RBF interpolation provides a robust framework for predicting intermediate values in a spatial context, making it especially effective in scenarios where smooth and accurate spatial predictions are required. The complete process is shown in figure 5.

Preparing the data in suitable formats for machine learning models was a critical step, achieved through normalization, feature engineering, and coordinate transformation. Normalization involved scaling the features to a consistent range, between 0 and 1, ensuring that all features contributed equally during the model training process. This step was later omitted due to the small impact it had on the performance of the model. Feature engineering created new attributes or modified existing ones to enhance the predictive

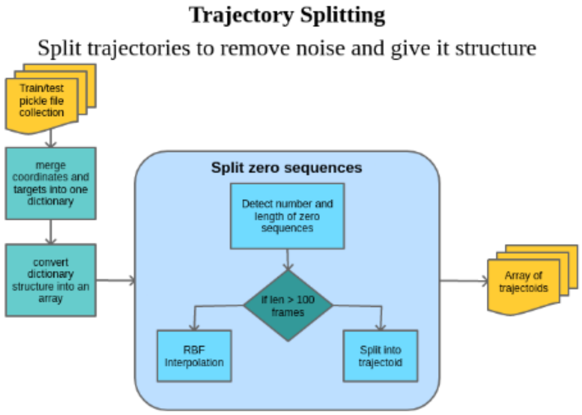
Trajectory Splitting
Split trajectories to remove noise and give it structure
Train/test pickle file collection
merge coordinates and targets into one dictionary
convert dictionary structure into an array
Split zero sequences
Detect number and length of zero sequences
If len > 100 frames
RBF Interpolation
Split into trajectoid
Array of trajectoids

Figure 5. A diagram outlining how the trajectories were split into trajectoids based on consecutive zero sequences and how the data was interpolated for suitability of use.

power of the models, such as the coordinate format. A significant transformation was converting cartesian coordinates to relative polar coordinates with respect to the observer agent. This was crucial as polar coordinates, which include radius and angle, offer a more intuitive representation of human movement by providing direct information on direction and distance. The transformation involved calculating the relative position of each data point to the observer and then converting these cartesian coordinates (x,y) into polar coordinates (r,θ) using the formulas $r = \sqrt{x^2+y^2}$ and $\theta = \tan^{-1}\left(\frac{y}{x}\right)$. This step was vital for normalizing the polar coordinates to ensure they were within a suitable range for model input.

Additionally, the polar coordinates were calculated with the orientation of the observer agent in mind. To determine the orientation of a point P relative to another point O, we calculate the angle between them in a 2D plane using polar coordinates. This orientation (angle) provides the direction from point O to point P, which is essential in trajectory prediction. Let O be the observer point with coordinates (x_O, y_O) and P be the point of interest with coordinates (x_P, y_P). The relative position is calculated as $\Delta x = x_P - x_O$ and $\Delta y = y_P - y_O$. The angle θ (orientation) from point O to point P can be determined using the arctangent function:

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

Since the $\tan^{-1}$ function can sometimes provide results that need adjustment based on the quadrant, the atan2 function is often used, which takes both Δy and Δx as arguments and returns the angle in the correct quadrant:

$$\theta = \operatorname{atan2}(\Delta y, \Delta x)$$

For example, if O is at $(2,3)$ and P is at $(5,7)$, then $\Delta x = 3$ and $\Delta y = 4$, resulting in $\theta = \operatorname{atan2}(4,3)$. This angle θ is crucial for accurately modeling human movement in enclosed spaces such as hospitals, as it provides direct information on the direction of movement relative to the observer. By transforming cartesian coordinates to relative polar coordinates, we improve the accuracy and reliability of trajectory predictions, which is a key focus of this thesis.

4.1.3 Data Splitting

Data splitting accounts for two separate operations. One is the splitting of the trajectories into input and target sequences. The other is about the splitting of the dataset into training, validation and testing sets. The whole process is visualised in figure 7. It should be noted that splitting into inputs and targets is also done in a sliding-window fashion, as shown in figure 6.

Due to the high frame rate of the original trajectory recordings (100 frames per second), all sequences, including the observer agent, other agents and environmental goals, have been condensed to skip every

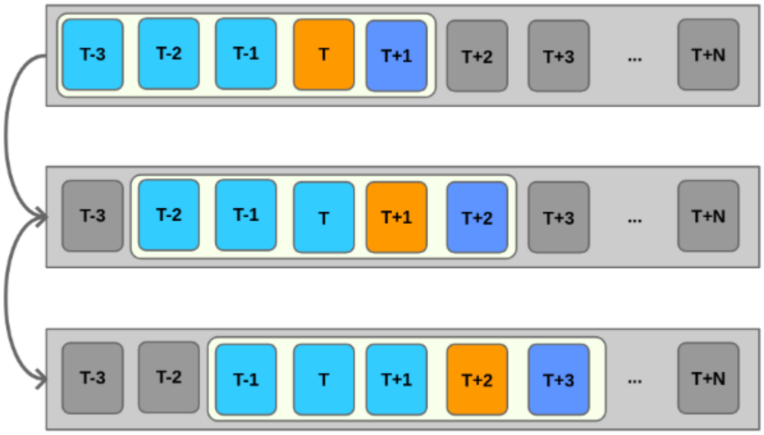

Figure 6. An example of a sliding window, where the frame at time-step T (orange) is taken as the reference frame, frames up to $T-3$ (light blue) are the input frames and then $T+1$ (dark blue) is the target frame. The window then slides through the frames, each time considering the corresponding sets of time-steps as input, reference and target.

50 frames. This results to all sequences being sampled at 2 fps. Input and output sequence sizes are controlled as well and typically held at 3 frames for the input (1.5 second sample), used as material to predict one frame into the future (0.5 seconds).

The agent coordinates were split into input and target sequences through various steps. Initially, the data that would be put into the complete input set was determined by means of relevance to the observer agent. Some agents were either too far away, or out of sight from the observer agent for them to be considered and were therefore filtered by means of proximity and presence in the observer's field of view.

Lastly, the transformation of the, so far, cartesian frames to polar coordinates with respect to either the past (for the target) or future (for the inputs) would take place. The frame used as a reference point for the transformation is called the reference point, which is not considered in neither the input, nor the target sequences. This practice is illustrated in figure 6, where the inputs are represented in light blue, the reference point in orange and the target in dark blue. This process would be repeated for N frames, indicating the end of the available frames from the dataset.

Trajectories in such settings are often better captured using polar coordinates relative to a fixed point (the observer agent). This transformation provided critical direction information through the angle (θ), indicating the movement direction relative to the observer, and distance information through the radius (r), showing how far the person was from the observer. Mathematically, polar coordinates naturally represent circular and radial movements, common in confined spaces. Additionally, normalizing these polar coordinates ensured they were in a suitable range for machine learning models, enhancing both learning efficiency and model performance. This step was fundamental in transforming the data into a format that improved the accuracy and reliability of trajectory predictions.

The final preprocessing step involved dividing the dataset into training and testing subsets to facilitate the evaluation of model performance. The train-test split typically allocated 80% of the data for training and 20% for testing. This split ensured that the model was trained on one subset and evaluated on another, unseen subset, which helped prevent overfitting and provided a realistic measure of the model's performance. The rationale behind this split was to allow for fine-tuning of model parameters using cross-validation techniques on the training set while reserving the test set for final evaluation, ensuring that the model's generalization capability was accurately assessed.

4.2 Model Training Method

In this chapter, we detail the model training methodology employed for the BiLSTM-based trajectory prediction model. The primary focus is on the training setup, hyperparameter settings, loss functions, and validation mechanisms. Additionally, we discuss the use of TensorBoard for visualization and model checkpointing.

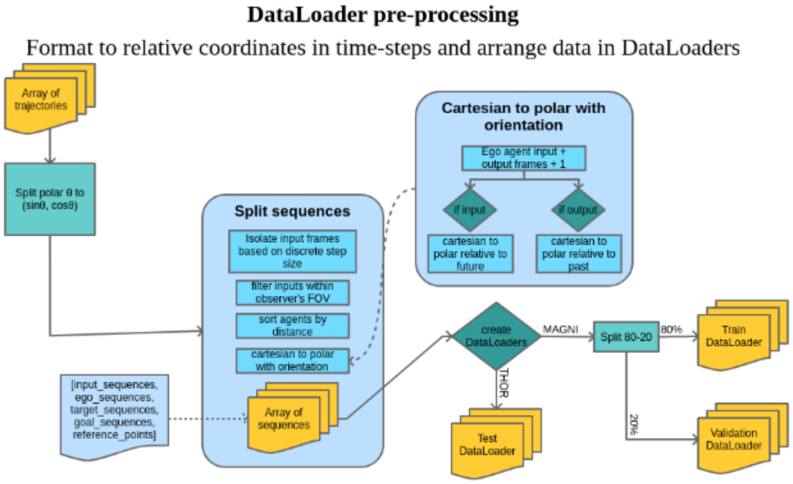
DataLoader pre-processing
Format to relative coordinates in time-steps and arrange data in DataLoaders
Array of trajectories
Split polar θ to (sinθ, cosθ)
Split sequences
Isolate input frames based on discrete step size
filter inputs within observer's FOV
sort agents by distance
cartesian to polar with orientation
Array of sequences
[input_sequences, ego_sequences, target_sequences, goal_sequences, reference_points]
Cartesian to polar with orientation
Ego agent input + output frames + 1
if input
if output
cartesian to polar relative to future
cartesian to polar relative to past
create DataLoaders
MAGNI
THOR
Split 80-20
80%
20%
Train DataLoader
Validation DataLoader
Test DataLoader

Figure 7. A diagram of the initial preprocessing pipeline for transitioning from the MATLAB format to a more structured time-series forecasting form

4.2.1 Model Description

MLP Model The MLP (Multi-Layer Perceptron) model utilized in this research handles sequential data processing tasks using fully connected layers. Each input frame is treated as a separate feature set. The core structure consists of the following components:

- **Fully Connected Layers**: Two fully connected layers (fc1 and fc2) map the flattened input features to the hidden layer size and add depth to the model. The equation for each layer is $y = Wx + b$, where W is the weight matrix, x is the input vector, and b is the bias vector.

- **Dropout Layer**: Applied after the first fully connected layer to prevent overfitting.

- **Activation Function**: The hyperbolic tangent (tanh) function introduces non-linearity: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

- **Output Layers**: Two separate fully connected layers predict distances and angles. The distance layer maps the hidden layer size to one-third of the output size, and the angles layer maps it to two-thirds of the output size.

- **Output**: The final output is a concatenation of distance and angle predictions, reshaped to match the required output frames and features.

During the forward pass, the input sequence is flattened and processed through the fully connected layers with tanh activations and dropout. The output is then passed through the final fully connected layers to obtain predictions for distances and angles.

BiLSTM Model The BiLSTM model predicts human walking trajectories using a bidirectional Long Short-Term Memory (LSTM) layer implemented with a Gated Recurrent Unit (GRU) for efficiency. The core structure consists of:

- **Bidirectional GRU Layer**: Captures temporal dependencies in both forward and backward directions, enhancing sequence context understanding. The GRU has:
  - **Reset Gate**: Determines how much of the past information to forget: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$.
  - **Update Gate**: Decides how much of the past information to retain and how much of the new information to incorporate: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$.
  - **Candidate Hidden State**: Uses the reset gate to control the amount of past information to include in the new candidate state: $\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$.

    - **Final Hidden State**: Combines the previous hidden state and the candidate hidden state, controlled by the update gate: $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$.

- **Dropout Layer**: A dropout layer with a rate of 0.2 is applied to the output of the GRU to prevent overfitting.
- **Activation Function**: The tanh function introduces non-linearity.
- **Linear Layers**: Two linear layers predict distances and angles separately, similar to the MLP model.
- **Output**: The final output is a concatenation of distance and angle predictions, reshaped to match the required output frames and features.

During the forward pass, the input sequence is processed through the GRU, followed by dropout and tanh activation. The output is then flattened and passed through the linear layers to obtain the final predictions.

Transformer Model The TransformerModel handles sequential data processing tasks by integrating several key components foundational to advancements in natural language processing (23). The core structure includes:

- **Positional Encoding**: Adds sequence information to the input embeddings, crucial for the transformer to understand the order of tokens:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right)$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right)$$

- **Transformer Encoder**: Composed of multiple layers, each consisting of:
    - **Multi-Head Self-Attention**: Allows the model to dynamically focus on different parts of the input sequence:

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

    - **Feed-Forward Networks**: Each position's output is passed through a feed-forward network:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- **Dropout Layer**: Applied within transformer layers to prevent overfitting.
- **Linear Layers**: Similar to the MLP model, linear layers are used to adjust the dimensions of the input data and the outputs from the transformer.
- **Output**: The final output is a concatenation of distance and angle predictions, reshaped to match the required output frames and features.

During the forward pass, the input sequence, embedded with positional encodings, is processed through the transformer layers, followed by dropout. The output is then flattened and passed through the linear layers to obtain the final predictions.

4.2.2 Directory Management and Experiment Setup

To organize experiments and ensure each run is uniquely identified, the script dynamically determines the next available run number. This process involves scanning existing run directories and finding the highest run number, incrementing it for the current experiment. This approach ensures that each experiment is logged separately, aiding in the systematic analysis of results and comparisons across different runs.

4.2.3 Hyperparameters and Model Initialization

The following hyperparameters are critical to the model's configuration:

- **Learning Rate**: Set to 0.0002, balancing between slow and rapid convergence.
- **Batch Size**: Fixed at 32, ensuring efficient use of memory while maintaining a good batch gradient estimate.
- **Hidden Layer Size**: Configured to 256, providing sufficient capacity for the BiLSTM layers to capture complex patterns.
- **Number of Layers**: The model consists of 4 BiLSTM layers, offering depth to capture temporal dependencies.
- **Input Dimensions**: Derived from the concatenation of input frames, ego information, and goals, adjusted to match the dataset's feature structure.
- **Output Dimensions**: Determined by the number of coordinates (3 per output frame) multiplied by the number of output frames, aligning with the prediction requirements.

The model, BiLSTM, is instantiated with these parameters and optimized using the Adam optimizer, which is chosen for its efficiency and adaptive learning rate capabilities. The loss functions used include Mean Squared Error (MSE) for distance predictions and L1 loss for angle predictions, combined to form a comprehensive loss metric.

4.2.4 Loss functions

The loss functions employed in the GRU-based prediction model are Mean Absolute Error (MAE) and Mean Squared Error (MSE). These functions are crucial for assessing the accuracy of the predicted trajectories in terms of both distances and angular values (represented by sine and cosine components).

Mean Absolute Error (MAE) MAE is used to measure the average magnitude of errors in a set of predictions, without considering their direction. It is particularly useful for distance predictions, as it provides a straightforward interpretation of the average prediction error. The MAE is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i is the true value and $\hat{y}_i$ is the predicted value.

In the context of this model, MAE is applied to the distance component of the predicted trajectories. This helps in quantifying how far, on average, the predicted positions are from the actual positions, providing a clear measure of positional accuracy.

Mean Squared Error (MSE) MSE is used to measure the average of the squares of the errors, giving more weight to larger errors. It is particularly useful for the angular predictions (sine and cosine values) as it penalizes larger deviations more severely. The MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In this model, MSE is applied to the sine and cosine components of the angular predictions. This helps in assessing the accuracy of the predicted orientations by evaluating how well the predicted sine and cosine values match the true values.

Combining MAE and MSE The combined use of MAE for distances and MSE for sine and cosine values ensures a balanced and comprehensive evaluation of the model's performance. By addressing both the positional accuracy (through MAE) and the angular accuracy (through MSE), the model can be effectively trained to produce reliable and precise trajectory predictions.

The total loss is a weighted sum of the distance loss (MAE) and the angle loss (MSE):

$$\text{Total Loss} = \alpha \cdot \text{MAE}_{\text{distance}} + \beta \cdot \text{MSE}_{\text{angles}}$$

where α and β are weights that balance the contributions of the distance and angle losses, respectively. This approach ensures that the model gives appropriate importance to both aspects of the trajectory predictions, leading to more accurate and realistic results.

4.2.5 Model Loading and Device Setup

For scenarios requiring continuation from a previous checkpoint, the script is capable of loading the latest saved model state. This is achieved by identifying the most recent checkpoint file and restoring the model parameters. The training is conducted on a CUDA-enabled GPU if available, falling back to the CPU if not. TensorBoard is initialized to log various training metrics, facilitating real-time monitoring and post-training analysis.

4.2.6 Training Procedure

The training loop involves iterating through the dataset in batches. Each batch undergoes several key steps:

- **Data Preparation**: Inputs, ego states, goals, and targets are loaded and transformed as necessary, ensuring they are in the correct shape and format for the model.
- **Forward Pass**: The data is passed through the BiLSTM model, which outputs reconstructed trajectories.
- **Loss Calculation**: The distance and angle losses are computed. The distance loss measures how close the predicted positions are to the actual positions, while the angle loss measures the accuracy of the predicted orientations. These are weighted and summed to obtain the total reconstruction loss.
- **Backward Pass and Optimization**: Gradients are computed via backpropagation, and the optimizer updates the model parameters to minimize the loss.
- **Metrics Logging**: Key metrics, including losses and predictions, are logged for each batch to monitor training progress.

4.2.7 Validation Process

Validation is performed at regular intervals to assess the model's performance on unseen data. This involves a similar process to training but excludes the backpropagation step. Key metrics from the validation phase include validation loss and reconstructed trajectory accuracy. These metrics are crucial for detecting overfitting and ensuring the model generalizes well to new data.

4.2.8 Checkpointing and Experiment Logging

To prevent loss of progress and facilitate model evaluation, checkpoints are saved periodically. This includes saving the model's state dictionary at predefined epochs. TensorBoard logs comprehensive metrics, including training and validation losses, parameter distributions, and hyperparameter settings, providing a detailed overview of the model's performance throughout the training process.

4.2.9 Summary

This chapter outlines the systematic approach to training the BiLSTM model for trajectory prediction. By carefully managing hyperparameters, utilizing efficient optimization techniques, and implementing robust validation and logging mechanisms, the training process ensures the development of an accurate and generalizable model. The detailed setup and continuous monitoring enable effective experimentation and refinement, leading to improved trajectory prediction capabilities in hospital environments.

5 RESULTS

5.1 Performance Evaluation

The performance of the GRU-based trajectory prediction model is evaluated using several key metrics that assess both the positional accuracy and the angular accuracy of the predictions. These metrics provide a comprehensive understanding of the model's effectiveness in predicting human walking trajectories in enclosed spaces.

5.1.1 Experiment 1 - Overall model evaluation on THOR and MAGNI

Root Mean Squared Error (RMSE) Root Mean Squared Error (RMSE) is used to measure the average magnitude of the errors in the positional predictions, with larger errors being more heavily penalized. RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

This metric provides a clear indication of the model's accuracy in predicting the distance between the true and predicted positions, with lower RMSE values indicating better performance.

A preliminary step involved the choice between the THOR and MAGNI sets for testing and training. This decision was made based on the performance on each dataset, when using the same model, which in this case was the GRU-based model.

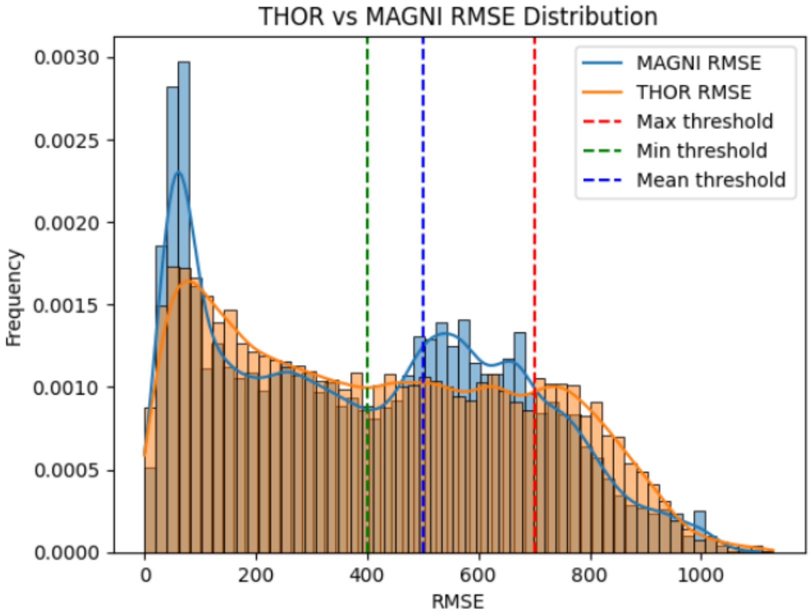

Figure 8. Comparing the RMSEs for each dataset, when trained on the same model.

The distribution shown in figure 8 and table 1 show that the MAGNI dataset would be more suitable for training, as it leads to RMSE values that are better distributed for the maximum, minimum and mean values established as acceptable thresholds.

Metric	MAGNI	THOR
RMSE below max threshold	85.17%	80.32%
RMSE below min threshold	51.00%	50.56%
RMSE below mean threshold	60.76%	60.73%

Table 1. RMSE Comparison between MAGNI and THOR

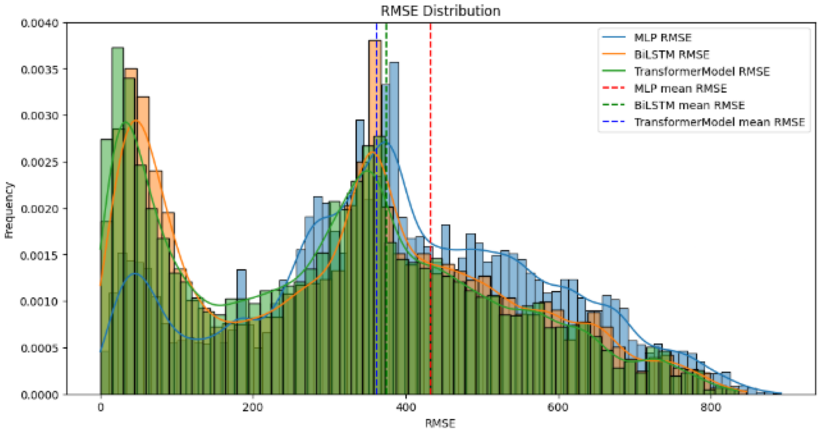

Figure 9. RMSE distributions for the three different models with mean and median locations

Model	RMSE
MLP	433.233
BiLSTM	374.456
TransformerModel	362.120

Table 2. Root Mean Squared Error (RMSE) Comparison of Different Models

Euclidean Distance Euclidean distance is another metric used to assess the positional accuracy of the predictions. It measures the straight-line distance between the true and predicted positions in the Cartesian coordinate system. The Euclidean distance for a pair of true and predicted coordinates (x_i, y_i) and $(\hat{x}_i, \hat{y}_i)$ is given by:

$$d_i = \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}$$

Averaging these distances over all predictions provides a robust measure of the overall positional accuracy of the model.

When comparing predicted trajectories to the ground truth, RMSE and Euclidean distance values are critical indicators of the model's performance. Low RMSE and Euclidean distance values signify that the predicted trajectories closely match the actual trajectories, indicating high accuracy. Conversely, high values suggest significant deviations between predictions and the ground truth, reflecting lower accuracy.

The RMSE metric provides a measure of the average magnitude of prediction errors, with larger errors having a disproportionately higher impact due to the squaring term. This makes RMSE particularly sensitive to outliers. The Euclidean distance, on the other hand, directly measures the straight-line distance between predicted and actual positions, offering a clear and intuitive understanding of positional accuracy. However, due to the resulting distributions looking very similar, there are not many new conclusions and insights that can be drawn from the separate analysis of these graphs. For the sake of consistency, it is preferable to use the RMSE, since the loss function is also more closely related to that.

Walking speed varies significantly between normal and crowded environments. In typical indoor settings, the average walking speed for healthy adults ranges between 1.2 and 1.4 m/s (approximately 4.3 to 5.0 km/h) (24; 25). However, in crowded environments, the walking speed can decrease considerably due to higher pedestrian density, which forces individuals to navigate more cautiously and avoid collisions. Studies indicate that in high-density environments, the average walking speed can drop to as low as 0.8 to 1.2 m/s (approximately 2.9 to 4.3 km/h) (26; 27). This reduction is influenced by factors such as the need to frequently change direction, the physical layout of the space, and psychological factors like stress and spatial awareness.

Given that frames are sampled every half second, the error metrics must reflect the real-world impact of prediction inaccuracies over this time interval. The thresholds for RMSE are derived from the average walking speeds reported in the literature. For healthy adults walking at an average speed of 1.4 m/s, the distance covered in half a second (the frame interval) is:

$$\text{maxDistance} = \text{Speed} \times \text{Time} = 1.4\,\text{m/s} \times 0.5\,\text{s} = 0.7\,\text{meters}$$

In crowded environments, where the average speed may drop to around 0.8 m/s, the distance covered in half a second is:

$$\text{minDistance} = 0.8\,\text{m/s} \times 0.5\,\text{s} = 0.4\,\text{meters}$$

A reasonable threshold for RMSE might be set around 0.5 meters. This threshold ensures that, on average, the predicted position is within 0.5 meters of the actual position after half a second. This value is justified as it accommodates the natural variability in human walking speed and direction changes, especially in dynamic environments like the one we are treating in this scenario. This range allows for minor deviations while still maintaining a high level of accuracy in trajectory predictions. Ensuring that the majority of predictions fall within these thresholds helps in maintaining reliable and realistic predictions, crucial for applications such as navigation aids in crowded indoor environments.

These thresholds balance the need for precision with the inherent variability in human movement, providing a robust framework for evaluating the performance of trajectory prediction models. Based on these thresholds, we can set the bounds and evaluate how each model performs with regards to the minimum, mean and maximum.

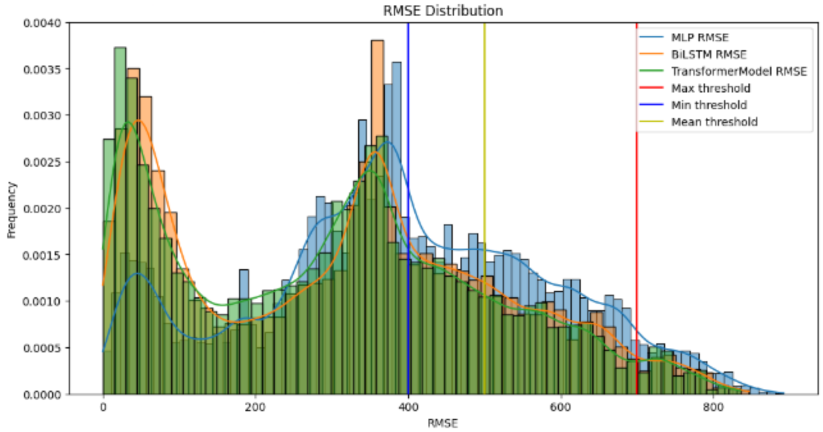

Figure 10. RMSE distributions for the three different models with the min max and mean thresholds

Model	Max threshold (%)	Min threshold (%)	Mean threshold (%)
MLP	94.674	55.080	71.023
BiLSTM	96.608	66.597	80.073
TransformerModel	**96.860**	**69.257**	**82.294**

Table 3. Threshold Comparison of Different Models

Similar to the previous results, table 3 and figure 10 display how the models perform with respect to the thresholds defined above. These results display the percentage of predictions whose RMSE falls below the min, max and mean. The TransformerModel shows the highest values of them all.

5.1.2 Experiment 2 - Distance, Sine and Cosine inspection

The results from the performance evaluation metrics are interpreted to understand the model's behavior and accuracy. Key insights include:

- **Positional Accuracy**: The RMSE provides insights into the positional accuracy of the model. Lower values indicate that the model predictions are closer to the actual positions.

- **Angular Accuracy**: The RMSE for polar coordinates reveals the model's effectiveness in predicting the direction of movement. Lower RMSE values suggest better angular accuracy.

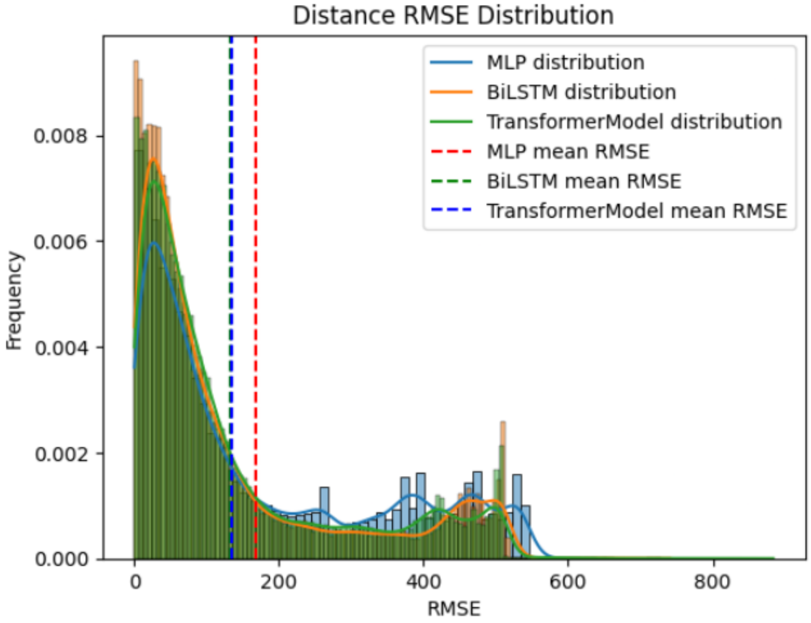

Figure 11. RMSE distribution for distance predictions.

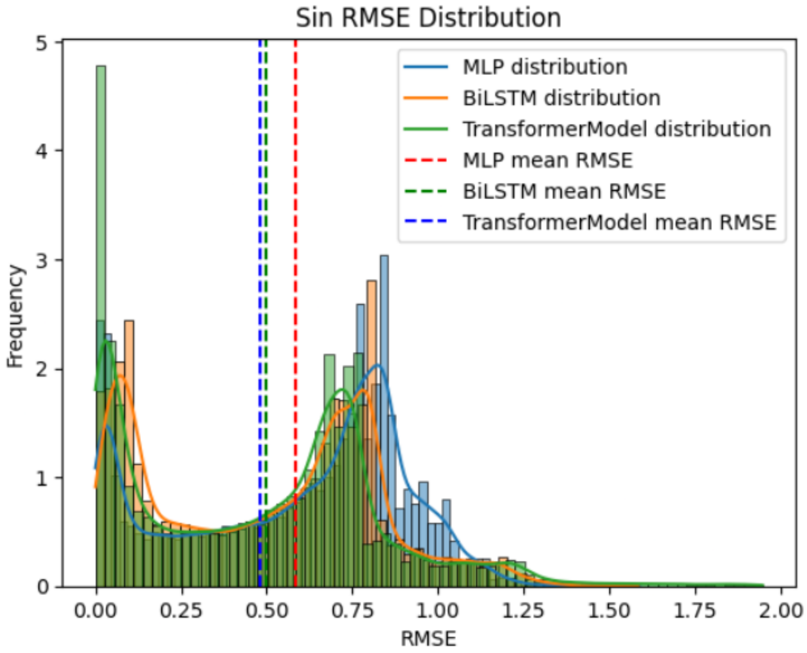

Figure 12. RMSE distribution for sin predictions.

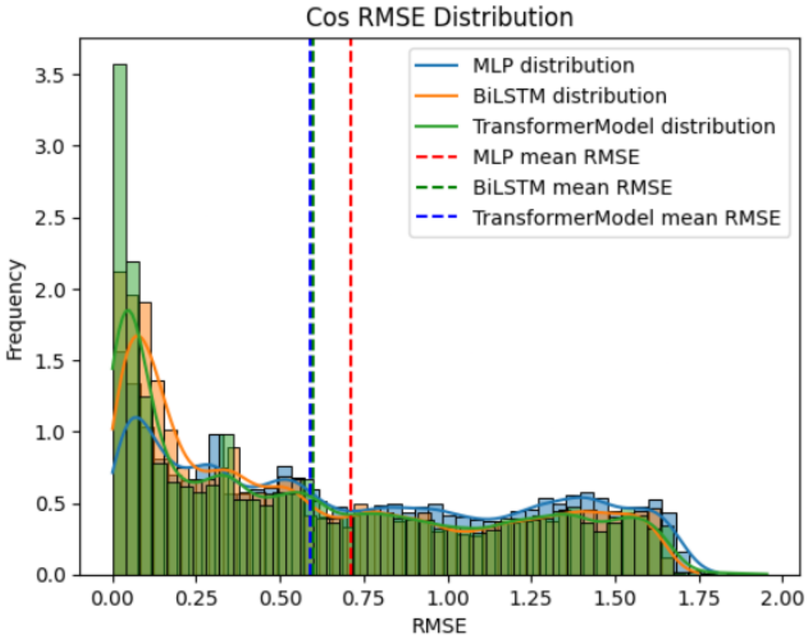

Figure 13. RMSE distribution for cos predictions.

The figures displaying the RMSE for distance, sin and cos are each determined based on the test set. Based on the higher means present in the sin and cos predictions (Table 4), we can see that the model struggles with predicting the correct orientation of the agent in their subsequent steps. Although figures 11 and 13 show a more acceptable distribution of the predicted distances, it is clear that the skewed results we see in figure 9 and 12 for the test set are very much alike.

Model	Distance Mean RMSE	Sine Mean RMSE	Cosine Mean RMSE
MLP	168.122	0.584	0.714
BiLSTM	**132.216**	0.497	0.597
TransformerModel	133.889	**0.479**	**0.591**

Table 4. Mean RMSE Comparison of Different Models for Distance, Sine, and Cosine

5.2 Discussion

In trajectory prediction models, polar coordinates are often used to describe positions relative to a reference point. The angle in polar coordinates can be represented using sine and cosine components. Observations in the sines and cosines of the predicted polar coordinates' angle reveal critical insights into the accuracy and reliability of the trajectory predictions.

Understanding Sine and Cosine Components in 2D Predictions In 2D predictions of human trajectories, the position of an individual at any point in time can be described using polar coordinates, which consist of a radius (distance from a reference point) and an angle (direction from the reference point). The angle can be decomposed into its sine and cosine components, which correspond to the y and x coordinates, respectively.

The sine component ($\sin\theta$) of the angle describes the vertical displacement relative to the reference point. It represents how much an individual moves up or down along the y-axis. For example, when an individual is moving vertically upward, the sine component will have a higher value, approaching 1. Conversely, when moving downward, the sine component will approach -1.

The cosine component ($\cos\theta$) of the angle describes the horizontal displacement relative to the reference point. It indicates how much an individual moves left or right along the x-axis. For instance, when an individual is moving horizontally to the right, the cosine component will be high, nearing 1. Conversely, when moving to the left, the cosine component will approach -1.

Implications of Errors in Sine of Angle The sine function in polar coordinates corresponds to the y-component of the direction vector. Larger errors in the sine value can lead to several specific implications:

- **Vertical Displacement Errors**: Since sine corresponds to the vertical component of the direction, larger errors will result in more pronounced inaccuracies in the predicted vertical movement. This can lead to significant deviations when predicting how much an individual or robot moves up (north) or down (south) relative to the reference point.

- **Direction Misinterpretation**: Errors in the sine component can lead to incorrect interpretations of the direction, particularly in quadrants where the sine value has a higher absolute magnitude (i.e., near 90° and 270°). This means that movements in these directions could be more inaccurately predicted.

- **Trajectory Curvature**: Trajectories that involve significant vertical changes or curves are more likely to be misrepresented if the sine component is inaccurate. This could affect the predicted path, making it less reliable for applications that require precise directional changes.

Comparison with Cosine Errors The cosine function corresponds to the x-component of the direction vector. While errors in both sine and cosine components can impact the predicted trajectory, the specific impacts differ:

- **Horizontal Displacement**: Errors in the cosine value affect the horizontal component of the direction. Larger errors in cosine imply inaccuracies in predicting the left-right (west-east) movement relative to the reference point.

- **Directional Consistency**: The cosine function reaches its maximum absolute values at 0° and 180°, meaning errors in these directions can significantly impact the perceived horizontal displacement. However, the sensitivity to directional changes may differ compared to sine, especially in the quadrants where cosine values are less dominant (near 90° and 270°).

Impacts of Sine Prediction Errors Observing that sine predictions exhibit two peaks in RMSE, one near 0 and another at 0.75, reveals specific challenges in trajectory prediction. The spike near RMSE of 0 indicates that while some predictions are highly accurate, others deviate significantly. This bimodal distribution suggests that the model may perform well under certain conditions but fail under others, leading to inconsistencies. The second peak at 0.75 highlights scenarios where the model struggles to predict the sine component accurately, potentially corresponding to specific angles where errors are more pronounced.

Cosine Prediction Distribution In contrast, the RMSE distribution for cosine predictions shows a peak at 0 with a gradual degradation towards 1.75, where the frequencies are very low. This distribution suggests that while most cosine predictions are accurate, the accuracy decreases steadily without sharp spikes. This gradual degradation indicates a more consistent performance across different scenarios, making cosine predictions generally more reliable for horizontal displacement.

5.2.1 Experiment 3 - What leads to error variations

Heatmaps are a valuable visualization tool for localizing errors in predicted trajectories by highlighting areas with high RMSE. By plotting RMSE values on a spatial grid, heatmaps can effectively show where the model's predictions deviate most from the actual trajectories. In this context, the heatmaps have been scaled by 200, which corresponds to 0.2 meters, allowing for a detailed and intuitive understanding of error distribution. This scaling helps in clearly identifying regions where the model's performance needs improvement, thus facilitating targeted refinements in the prediction algorithms. The use of heatmaps provides a clear and immediate visual representation of error hotspots, making it easier for us to diagnose and address specific issues in trajectory predictions.

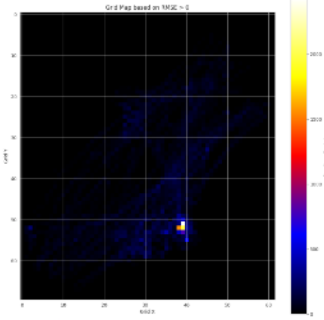

Figure 14. THOR heatmap for RMSE > 0

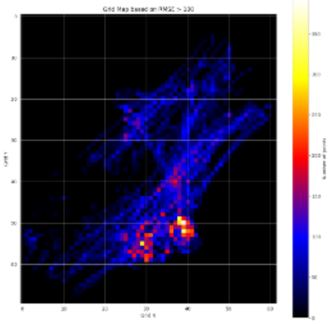

Figure 15. THOR heatmap for RMSE > 100

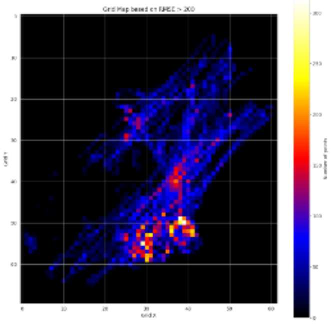

Figure 16. THOR heatmap for RMSE > 200

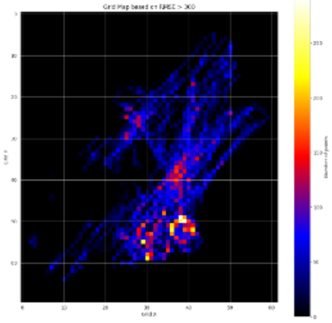

Figure 17. THOR heatmap for RMSE > 300

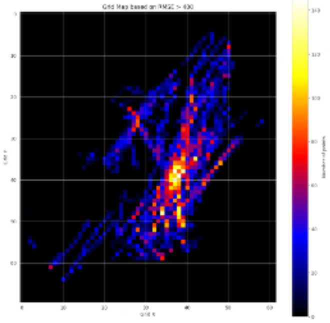

Figure 18. THOR heatmap for RMSE > 400

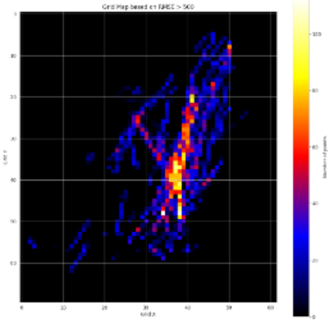

Figure 19. THOR heatmap for RMSE > 500

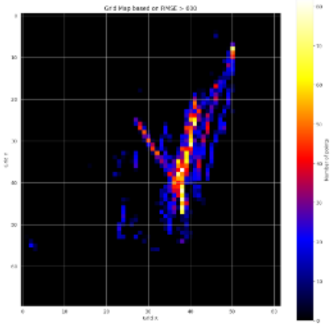

Figure 20. THOR heatmap for RMSE > 600

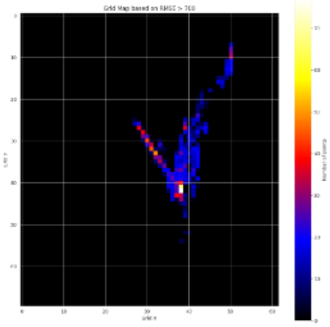

Figure 21. THOR heatmap for RMSE > 700

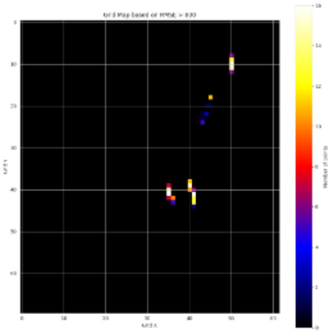

Figure 22. THOR heatmap for RMSE > 800

Figure 23. Heatmaps of THOR test set RMSE locations

Although the errors are approximately equally distributed across the lower thresholds, there are some more noteworthy patterns in the 400, 500 and 600 thresholds. The highest concentration occurs in the south-east part of the map. Inspecting the highest values of 700 and 800 show a more refined segment of where most of the errors come up, which happen to be in the turning points of the trajectories.

Errors up to 300 seem to be spread more or less equally between the different trajectory segments, although after that, there is a much more focused segment that is popping up. There are patterns in the south-east part of the map that are increasingly highlighted. This implies there is a particular set of tracks in which the model struggles.

The particular trajectory that could be problematic is shown in figure 24. The pattern of this ”triangle” trajectory is quite unique and consists of multiple twists and turns with changes to the orientation and focus of the agent. Such variation is not particularly common in the training data, as most of the trajectories are long straights with one sharp turn once their goal is reached.

An attempt was made to correct this, by introducing the mirroring that was discussed during the

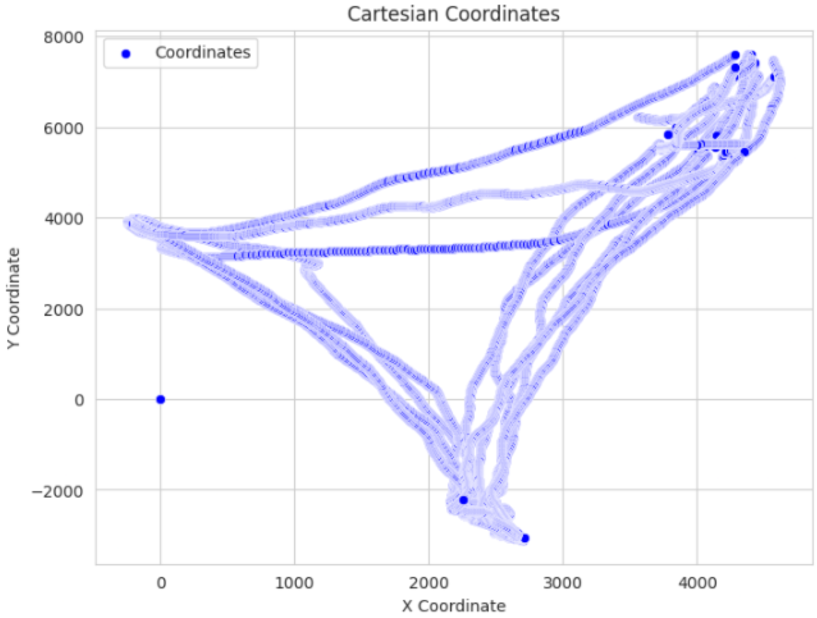

Figure 24. Specific complete trajectory from an agent in the THOR dataset

pre-processing steps. Although in principle, the behaviours should be able to be detected from a trajectory similar to the one shown in figure 25, the model still struggles to generalise to this scenario.

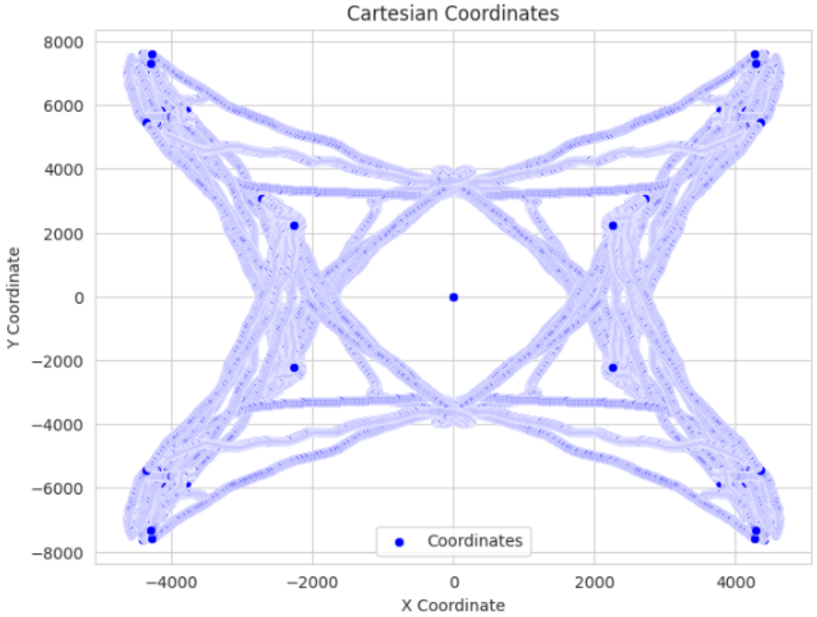

Figure 25. Sample mirrored training trajectory from MAGNI dataset

6 CONCLUSIONS

The analysis of the MLP, BiLSTM, and TransformerModel demonstrates significant capabilities and differences in predicting human walking trajectories in enclosed spaces, with a particular emphasis on both positional and angular accuracies. The evaluation metrics, specifically RMSE and Euclidean distance, provide clear insights into each model's performance. Lower RMSE and Euclidean distance values indicate high prediction accuracy, signifying that the model's predicted trajectories closely match the actual trajectories.

Key findings from the performance evaluation include:

- **Positional Accuracy**: The RMSE and Euclidean distance metrics indicate that the TransformerModel achieves the lowest overall RMSE (362.120), outperforming both the BiLSTM (374.456) and the MLP (433.233). This highlights the TransformerModel's superior ability to generalize across different environments. However, the BiLSTM also demonstrates strong performance, suggesting robustness in familiar settings but facing challenges in unseen scenarios. The MLP, while effective, shows the highest error rates, indicating room for improvement in its ability to handle positional predictions.

- **Angular Accuracy**: The RMSE for polar coordinates, particularly for sine and cosine components, reveals significant differences among the models. The BiLSTM model performs well with low sine RMSE (0.497) and cosine RMSE (0.597), indicating reliable angular predictions. The TransformerModel shows the best overall performance with the lowest sine RMSE (0.479) and cosine RMSE (0.591), making it highly reliable for predicting the direction of movement. The MLP, with higher errors (sine RMSE 0.584, cosine RMSE 0.714), suggests greater difficulty in accurately predicting angular movements.

- **Impact of Environmental Variability**: The analysis shows that increasing the diversity of training environments can enhance each model's generalization capabilities. Introducing a portion of the test environment into the training set improved performance, evidenced by better-distributed error metrics and higher percentages of trajectories within acceptable thresholds. The TransformerModel demonstrates exceptional performance with the highest max (96.860%), min (69.257%), and mean (82.294%) threshold values, indicating its robustness in varying environmental conditions. The BiLSTM also shows strong generalization but slightly lower than the TransformerModel, while the MLP benefits significantly from environmental variability but still lags behind the other two models.

- **Implications for Robot Navigation in Healthcare Settings**: The observed inaccuracies in sine predictions have critical implications for robot navigation in indoor healthcare settings, such as hospitals. Larger errors in vertical displacement could result in robots misjudging vertical movements required to navigate through different floors or reach specific vertical positions, such as adjusting to different heights of patient beds or medical equipment. This could hinder the robot's ability to perform tasks like delivering supplies or assisting with telemedicine applications effectively. In contrast, the more consistent performance of cosine predictions suggests that horizontal navigation tasks, such as moving through hallways or delivering items between rooms, might be more reliably handled. However, the gradual degradation in cosine accuracy still implies that the robot's performance might deteriorate over long distances or complex navigation routes, requiring ongoing monitoring and adjustment.

Overall, the results underscore the importance of diverse training environments and detailed error analysis in developing robust trajectory prediction models. By addressing the identified issues and continuing to refine the models, it is possible to enhance their predictive accuracy, making them valuable tools for applications such as navigation aids in crowded indoor environments. The study demonstrates that with further improvements, the MLP, BiLSTM, and particularly the TransformerModel, can achieve reliable and accurate trajectory predictions, crucial for real-world applications in dynamic and complex settings. The TransformerModel, in particular, shows great promise due to its superior performance metrics, making it an excellent candidate for further development and deployment in such environments.

6.1 Limitations

While the MLP, BiLSTM, and Transformer models show promising results, there are several limitations that need to be addressed:

- **Error in Angular Predictions**: The models exhibit larger errors in predicting the sine and cosine components of the angles, leading to inaccuracies in vertical displacement and direction changes.

This issue is particularly pronounced in the MLP model, while the TransformerModel and BiLSTM perform better but still encounter challenges in scenarios involving significant vertical movements or complex directional changes.

- **Impact of Preprocessing Steps**: The preprocessing steps, including data cleaning, handling missing values, and transforming coordinates, introduce potential sources of error. For instance, the transformation from Cartesian to polar coordinates may lead to inaccuracies if not handled carefully. Additionally, the mirroring technique used to augment the training data may not fully capture the complexity of real-world trajectories.

- **Dependency on Walking Speed Variability**: The thresholds for RMSE are derived based on average walking speeds, which may not account for the full range of walking speeds encountered in different indoor environments. Variability in walking speed due to factors like pedestrian density, obstacles, and individual differences can affect the accuracy of the predictions, impacting all models.

- **Heatmap Resolution**: The heatmaps used for error localization are scaled by 200 (0.2 meters), providing a detailed view of error distribution. However, this resolution may not capture finer nuances in error patterns, especially in highly dynamic environments. Further refinement of heatmap resolution may be necessary for more precise error analysis.

- **Computational Complexity**: The BiLSTM and Transformer models, while effective, are computationally intensive. Training these models on larger and more diverse datasets may require significant computational resources. This limitation could affect the feasibility of deploying these models in real-time applications. The MLP, being less complex, is computationally more efficient but shows higher error rates, indicating a trade-off between complexity and accuracy.

- **Limited Real-world Validation**: Although the models have been evaluated using representative datasets, such as THOR and MAGNI, there is still a need for extensive validation in actual indoor environments. Field testing and validation are crucial to ensure the models' robustness and applicability in practical settings, particularly in healthcare environments where precision and reliability are paramount.

Addressing these limitations will be essential for improving the models' performance and reliability. Future work should focus on expanding the diversity of training environments, enhancing the preprocessing techniques, and conducting extensive real-world validations to ensure the models' applicability in various indoor navigation scenarios.

6.2 Future Steps

Building on the limitations and results of the current study, several future steps are proposed to enhance the performance and applicability of the different models presented in this research:

- **Diversify Training Data**: One of the key limitations identified is the models' difficulty in generalizing to unseen environments. To mitigate this issue, future work should focus on diversifying the training data to encompass a wider range of indoor environments. This can be achieved by collecting data from various indoor settings, such as hospitals, shopping malls, airports, offices, and schools, each with different layouts, pedestrian densities, and obstacle configurations. For instance, data collected from a hospital may include narrow corridors, crowded waiting areas, and frequent interruptions by medical staff, whereas data from a shopping mall might feature wide open spaces, escalators, and varying foot traffic.

  In addition to real-world data collection, synthetic data augmentation techniques can be employed. These techniques involve creating virtual environments using simulation tools where pedestrian movements can be simulated under different conditions. For example, a virtual environment can simulate an airport scenario with varying passenger volumes, different times of day, and the presence of static obstacles like seating areas and dynamic obstacles like moving trolleys. By training the models on both real and synthetic data, we can ensure that they are exposed to a wide range of scenarios, thereby enhancing their generalization capabilities.

Furthermore, leveraging techniques such as Generative Adversarial Networks (GANs) to create realistic synthetic data can provide additional variability and richness to the training datasets. This approach not only expands the diversity of scenarios but also introduces controlled variations in pedestrian behaviors and environmental changes, allowing the models to learn and adapt more effectively.

- **Refine Preprocessing Techniques**: The preprocessing steps, particularly the transformation from Cartesian to polar coordinates, need further refinement. Current methods of converting coordinates can introduce errors, particularly if the data is noisy or contains outliers. To address this, future research should explore alternative methods for coordinate transformation that can maintain data integrity and reduce inaccuracies.

  One approach is to directly predict polar coordinates instead of converting from Cartesian coordinates. This can be achieved by designing models that are trained to output polar coordinates directly, thus bypassing the need for transformation and the associated errors. Alternatively, hybrid approaches that combine both Cartesian and polar representations could be explored. For example, a model could use Cartesian coordinates for initial processing and then convert to polar coordinates for final predictions, combining the strengths of both representations.

  Developing more robust preprocessing pipelines is also essential. This includes implementing advanced data cleaning techniques to handle missing values and noise. For instance, using interpolation methods to estimate missing data points or applying smoothing techniques to reduce noise can improve data quality. Outlier detection methods, such as statistical tests or machine learning algorithms, can identify and handle anomalous data points that could skew the results.

  Incorporating domain-specific knowledge into the preprocessing steps can further enhance the accuracy of the models. For example, in a healthcare setting, understanding the typical walking patterns and behaviors of different patient groups can inform the preprocessing techniques used, leading to more accurate and reliable trajectory predictions.

- **Account for Walking Speed Variability**: To ensure the models' accuracy across different walking speeds, future work should consider incorporating dynamic thresholds for RMSE and Euclidean distance that adjust based on the observed walking speed. Walking speeds can vary significantly based on the individual's age, physical condition, and environmental context. For example, an elderly person in a hospital may walk more slowly than a young adult in a shopping mall.

  To account for this variability, the models can be trained to recognize different walking speeds and adjust their error thresholds accordingly. This involves analyzing the distribution of walking speeds in the training data and setting adaptive thresholds that reflect this variability. For instance, the model can apply a lower threshold for slower walking speeds and a higher threshold for faster speeds, ensuring more accurate predictions across different conditions.

  Collecting data from a broader demographic is also crucial. This includes gathering data from various age groups, physical conditions, and environments to capture a wide range of walking speeds and behaviors. For example, data can be collected from children, adults, and the elderly, as well as from people with different mobility aids, such as wheelchairs or walkers. This diversity in the training data helps the model generalize better to different scenarios and populations.

  Furthermore, incorporating dynamic modeling techniques can enhance the model's adaptability. For example, using techniques such as Kalman filtering or particle filtering can help the model adjust its predictions based on the observed walking speed in real-time. These methods allow the model to continuously update its state and predictions, improving its accuracy and robustness in dynamic environments.

- **Conduct Extensive Real-world Validation**: To ensure the models' robustness and applicability in practical settings, extensive real-world validation is essential. Field testing in various indoor environments, such as hospitals, shopping malls, and airports, can provide valuable feedback on the models' performance and highlight areas for further improvement.

  Real-world validation should involve testing under different conditions, including varying crowd densities, and the presence of dynamic obstacles. For instance, in a hospital setting, validation can be performed during different times of the day to capture variations in lighting and pedestrian traffic. Testing in different wards with varying layouts and obstacles, such as medical equipment and furniture, can provide insights into the model's ability to navigate complex environments.

Additionally, it is crucial to involve end-users, such as hospital staff and patients, in the validation process. Their feedback can help identify practical challenges and areas where the models need improvement. For example, staff can provide insights into the typical walking patterns and behaviors observed in the hospital, which can inform further refinements to the models.

Incorporating real-time monitoring and feedback mechanisms during validation can also enhance the process. By deploying the models in a controlled environment and continuously monitoring their performance, any deviations or errors can be promptly identified and addressed. This iterative process of validation and refinement ensures that the models are robust, reliable, and ready for deployment in real-world scenarios.

Finally, collaboration with interdisciplinary teams, including engineers, healthcare professionals, and human factors experts, can enrich the validation process. Their diverse perspectives can help address any shortcomings in the models and ensure that they are optimized for practical use in dynamic and complex environments, such as hospitals and other indoor settings.

- **Incorporate Real-time Feedback Mechanisms**: Future iterations of the models could benefit from incorporating real-time feedback mechanisms that allow for continuous learning and adaptation based on new data. Implementing online learning techniques, where the model updates its parameters incrementally as new data becomes available, can help the models stay updated with changing environments and pedestrian behaviors.

  Real-time feedback mechanisms can involve the use of sensor data to dynamically adjust predictions and improve model performance. For example, real-time data from motion sensors, cameras, and wearable devices can be used to monitor individual's movements and update the model's parameters accordingly. This allows the model to adapt to changes in the environment and pedestrian behavior in real-time, improving its accuracy and reliability.

  Additionally, incorporating user interactions into the feedback loop can enhance the model's learning process. For example, healthcare staff can provide real-time feedback on the model's predictions, highlighting any inaccuracies or areas for improvement. This feedback can be used to fine-tune the model and ensure that it remains accurate and reliable over time.

  Implementing real-time feedback mechanisms also requires robust data processing and integration frameworks. This includes developing systems to collect, process, and analyze real-time data from various sources, such as sensors and user inputs. By integrating these frameworks with the trajectory prediction models, we can create a dynamic and adaptive system that continuously learns and improves based on new data.

  Moreover, real-time feedback mechanisms can enhance the model's ability to handle unexpected events and anomalies. For example, if the model encounters a sudden change in pedestrian behavior or an unexpected obstacle, real-time feedback can help it adjust its predictions and navigate the situation effectively. This adaptability is crucial for ensuring the model's robustness and reliability in dynamic and unpredictable environments.

- **Explore Multimodal Data Integration**: Integrating additional data modalities, such as visual inputs from cameras or sensor data from wearable devices, can provide a richer context for trajectory prediction. Combining modalities through techniques like sensor fusion or using multimodal neural networks can enhance the models' ability to understand and predict complex human movements in indoor environments.

  For instance, visual data from cameras can be used to detect and track obstacles, understand the layout of the environment, and identify other moving agents. This information can be combined with trajectory data to improve the model's predictions. In a hospital setting, cameras can monitor the movement of patients and staff, detect the presence of medical equipment, and identify potential obstacles, such as unattended trolleys or temporary barriers.

  Sensor data from wearable devices, such as accelerometers, gyroscopes, and heart rate monitors, can provide additional insights into the pedestrian's physical state and movement patterns. For example, accelerometer data can be used to detect sudden changes in speed or direction, while heart rate data can indicate the pedestrian's level of exertion. By integrating these data sources, the model can gain a more comprehensive understanding of the pedestrian's behavior and environment, leading to more accurate and reliable trajectory predictions.

Implementing multimodal data integration requires the development of sophisticated data fusion techniques that can combine and process information from different sources. Techniques such as Kalman filtering, particle filtering, and Bayesian networks can be used to integrate sensor data and improve the model's accuracy. Additionally, neural network architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be adapted to handle multimodal inputs and learn complex relationships between different data sources.

Furthermore, incorporating multimodal data can enhance the model's ability to handle dynamic and unpredictable environments. For example, in a crowded hospital corridor, the model can use visual data to detect and avoid obstacles, while sensor data can provide real-time feedback on the pedestrian's movements. This multimodal approach can improve the model's robustness and reliability, ensuring that it can effectively navigate complex indoor environments.

Finally, integrating multimodal data can also enable the development of more advanced features and functionalities. For example, the model can be designed to recognize and respond to specific events, such as a patient falling or a staff member calling for assistance. By leveraging the rich context provided by multimodal data, the model can become a more effective and versatile tool for indoor navigation and trajectory prediction.

- **Modeling Environmental Obstacles**: To enhance the accuracy and robustness of human trajectory prediction models, it is crucial to incorporate detailed representations of environmental obstacles. These obstacles can significantly influence pedestrian movement patterns, particularly in complex indoor environments.

  Firstly, the model should integrate spatial data about the environment, including the layout of walls, furniture, and other static objects. This can be achieved through the use of floor plans and architectural models, which provide precise information about the spatial constraints within a given area. By embedding this spatial data into the model, we can ensure that trajectory predictions account for the physical limitations imposed by the environment.

  Moreover, incorporating dynamic obstacles, such as other pedestrians and movable objects, is essential for realistic trajectory predictions. To model these dynamic elements, the model can utilize data from real-time sensors, such as LIDAR, cameras, and motion detectors, which provide continuous updates on the positions and movements of obstacles. By integrating real-time sensor data, the model can dynamically adjust predictions based on the current state of the environment.

  Advanced techniques such as Simultaneous Localization and Mapping (SLAM) can be employed to create and update a real-time map of the environment. SLAM algorithms allow the model to construct a detailed map while simultaneously tracking the movement of the agent within it. This approach is particularly useful in environments where the layout frequently changes, such as hospitals where equipment and furniture are often moved.

  In addition to static and dynamic obstacles, the model should consider temporary barriers and changes in the environment, such as construction areas or temporary installations. Incorporating this information requires a flexible data integration framework that can quickly adapt to new inputs and update the model accordingly. For example, integrating real-time data feeds from facility management systems can provide up-to-date information on temporary changes in the environment.

  Finally, combining environmental obstacle modeling with predictive algorithms allows the model to simulate potential future states of the environment. This predictive capability is essential for anticipating and planning for changes, such as predicting pedestrian flow during peak hours or anticipating the impact of a temporary barrier on movement patterns. By simulating various scenarios, the model can provide more robust and reliable trajectory predictions.

  Implementing these enhancements will significantly improve the model's ability to accurately predict human trajectories in complex and dynamic indoor environments. This, in turn, will make the models more effective for practical applications, such as guiding robots in hospitals, optimizing pedestrian flow in shopping malls, and enhancing safety and efficiency in various other indoor settings.

- **Modeling of Human Attributes and Behaviors**: To further enhance the accuracy and applicability of the models, future work should focus on incorporating detailed human attributes and behaviors into the prediction process. Attributes such as gaze direction, height, body pose estimation, and individual walking patterns can provide valuable context for trajectory predictions.

  For example, gaze direction can indicate a pedestrian's intended direction of movement or areas of interest, which can be crucial for predicting sudden changes in trajectory. Height and body pose estimation can provide insights into a pedestrian's physical state and potential obstacles they may face. Incorporating these attributes requires the integration of advanced computer vision techniques and sensor data.

  One approach to model these attributes is to use pose estimation algorithms, such as OpenPose, to extract key body landmarks from visual data. These landmarks can then be used to determine the pedestrian's posture, gait, and other physical attributes. Combining this information with trajectory data can enhance the model's understanding of human movements and improve prediction accuracy.

  Additionally, modeling individual walking patterns can provide personalized predictions for different pedestrians. For example, elderly individuals or people with mobility impairments may have distinct walking patterns that need to be accounted for in the prediction model. By incorporating detailed human attributes and behaviors, the models can become more adaptive and accurate in various real-world scenarios.

- **Modeling Relative Perceptions of Other Agents and Goals**: Incorporating the relative perceptions of other agents and their goals is crucial for improving the accuracy of human trajectory prediction models, especially in dynamic and interactive environments. This approach allows the model to better understand and anticipate the movements of individuals in relation to others and their intended destinations.

  Firstly, the model should incorporate information about the positions and movements of other agents within the environment. This can be achieved through multi-agent tracking systems that utilize data from cameras, LIDAR, and other sensors to continuously monitor the locations and trajectories of individuals. By integrating this data, the model can consider the relative positions and velocities of other agents when making predictions.

  In addition to tracking physical movements, the model should infer the goals and intentions of other agents. Understanding where individuals are likely to go and their intended paths can significantly enhance prediction accuracy. For example, in a hospital setting, the model could use contextual information, such as the locations of patient rooms, nurses' stations, and common areas, to predict the likely destinations of hospital staff and visitors. Machine learning techniques, such as goal inference algorithms, can be employed to estimate these intentions based on historical movement patterns and current context.

  Social behaviors and interactions between agents also play a critical role in trajectory prediction. Agents often adjust their paths to avoid collisions, maintain personal space, and follow social norms. The model should incorporate these social dynamics by using algorithms that simulate human behavior, such as social force models or pedestrian interaction models. These models help predict how individuals will navigate around each other and respond to the presence of others in their environment.

  Moreover, the model can benefit from incorporating communication cues and signals. For instance, observing gestures, eye contact, and body orientation can provide additional insights into the intentions and future movements of other agents. Advanced computer vision techniques can be used to detect and interpret these cues, enhancing the model's ability to predict trajectories in interactive scenarios.

  Integrating multi-agent simulation capabilities can further improve the model's predictive accuracy. Multi-agent simulations allow the model to test and validate predictions in virtual environments, where the interactions and behaviors of multiple agents can be observed and analyzed. This approach helps refine the model's algorithms and ensures that it can handle complex scenarios involving numerous interacting agents.

Finally, combining the perception of other agents with the understanding of environmental goals creates a holistic model that accurately predicts human trajectories. For example, in an airport, the model can consider the locations of gates, shops, and restrooms, along with the movements and intentions of other travelers, to provide more accurate predictions. This integrated approach ensures that the model accounts for both individual behaviors and collective dynamics within the environment.

Implementing these enhancements will significantly improve the model's ability to predict human trajectories in interactive and dynamic settings. This, in turn, will make the models more effective for applications such as autonomous navigation in crowded environments, enhancing pedestrian safety, and optimizing the flow of people in various public spaces.

Implementing these future steps will be crucial for advancing the capabilities of the MLP, BiLSTM, and TransformerModel, ensuring they are robust, accurate, and applicable to a wide range of real-world indoor navigation scenarios.

REFERENCES

[1] F. Camara, N. Bellotto, S. Cosar, F. Weber, D. Nathanael, M. Althoff, J. Wu, J. Ruenz, A. Dietrich, G. Markkula, A. Schieben, F. Tango, N. Merat, and C. Fox, "Pedestrian models for autonomous driving part ii: High-level models of human behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5453–5472, 2021.

[2] J. S. Hartford, J. R. Wright, and K. Leyton-Brown, "Deep learning for predicting human strategic behavior," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[3] A. D. Dragan and S. S. Srinivasa, "Familiarization to robot motion," in *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 366–373, 2014.

[4] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal, "Thör: Human-robot navigation data collection and accurate motion trajectories dataset," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 676–682, 2020.

[5] T. Schreiter, T. R. de Almeida, Y. Zhu, E. G. Maestro, L. Morillo-Mendez, A. Rudenko, T. P. Kucner, O. M. Mozos, M. Magnusson, L. Palmieri, *et al.*, "The magni human motion dataset: Accurate, complex, multi-modal, natural, semantically-rich and contextualized," *arXiv preprint arXiv:2208.14925*, 2022.

[6] P. Perconti and A. Plebe, "Deep learning and cognitive science," *Cognition*, vol. 203, p. 104365, 2020.

[7] M. Allen and K. J. Friston, "From cognitivism to autopoiesis: Towards a computational framework for the embodied mind," *Synthese*, vol. 195, no. 6, p. 2459–2482, 2016.

[8] M. Bar, "The proactive brain: using analogies and associations to generate predictions," *Trends in Cognitive Sciences*, vol. 11, no. 7, pp. 280–289, 2007.

[9] R. Tian, M. Tomizuka, A. Dragan, and A. Bajcsy, "Towards modeling and influencing the dynamics of human learning," 2023.

[10] M. Fintz, M. Osadchy, and U. Hertz, "Using deep learning to predict human decisions, and cognitive models to explain deep learning models," *bioRxiv*, 2021.

[11] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," 2020.

[12] J. Yue, D. Manocha, and H. Wang, "Human trajectory prediction via neural social physics," 2023.

[13] A. Mohamed, D. Zhu, W. Vu, M. Elhoseiny, and C. Claudel, "Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation," 2022.

[14] U.-H. Kim, D. Ka, H. Yeo, and J.-H. Kim, "A real-time predictive pedestrian collision warning service for cooperative intelligent transportation systems using 3d pose estimation," 2022.

[15] J. Amirian, B. Zhang, F. V. Castro, J. J. Baldelomar, J.-B. Hayet, and J. Pettre, "Opentraj: Assessing prediction complexity in human trajectories datasets," in *Asian Conference on Computer Vision (ACCV)*, no. CONF, Springer, 2020.

[16] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot operating system 2: Design, architecture, and uses in the wild,” *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[17] D. M. Nguyen, M. Nazeri, A. Payandeh, A. Datar, and X. Xiao, “Toward human-like social robot navigation: A large-scale, multi-modal, social human navigation dataset,” 2023.

[18] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone, “Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation,” 2022.

[19] E. Weng, H. Hoshino, D. Ramanan, and K. Kitani, “Joint metrics matter: A better standard for trajectory forecasting,” 2023.

[20] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, “Uncertainty estimation for cross-dataset performance in trajectory prediction,” 2022.

[21] J. Lu, C. Cui, Y. Ma, A. Bera, and Z. Wang, “Quantifying uncertainty in motion prediction with variational bayesian mixture,” 2024.

[22] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, “Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps,” 2019.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[24] Y. Kobayashi, M. Sudo, H. Miwa, H. Hobara, S. Hashizume, K. Nakajima, N. Takayanagi, T. Ueda, Y. Niki, and M. Mochimaru, “Estimation accuracy of average walking speed by acceleration signals: Comparison among three different sensor locations,” in *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018)* (S. Bagnara, R. Tartaglia, S. Albolino, T. Alexander, and Y. Fujita, eds.), (Cham), pp. 346–351, Springer International Publishing, 2019.

[25] C. Willen, K. Lehmann, and K. Sunnerhagen, “Walking speed indoors and outdoors in healthy persons and in persons with late effects of polio,” *Journal of Neurology Research*, vol. 3, no. 2, 2013.

[26] N. A. Sen, P. Carreno-Medrano, and D. Kulić, “Pedestrian walking speed analysis: A systematic review,” *Scientific Reports*, vol. 13, no. 1, pp. 1–10, 2023.

[27] N. Ah Sen, P. Carreno-Medrano, and D. Kulić, “Human-aware subgoal generation in crowded indoor environments,” in *Social Robotics* (F. Cavallo, J.-J. Cabibihan, L. Fiorini, A. Sorrentino, H. He, X. Liu, Y. Matsumoto, and S. S. Ge, eds.), (Cham), pp. 50–60, Springer Nature Switzerland, 2022.