MS-TIP: Imputation Aware Pedestrian Trajectory Prediction

Pranav Singh Chib [*1] **Achintya Nath** [*1] **Paritosh Kabra** [*1] **Ishu Gupta** [*1] **Pravendra Singh** [*1]

Abstract

Pedestrian trajectory prediction aims to predict future trajectories based on observed trajectories. Current state-of-the-art methods often assume that the observed sequences of agents are complete, which is a strong assumption that overlooks inherent uncertainties. Understanding pedestrian behavior when dealing with missing values in the observed sequence is crucial for enhancing the performance of predictive models. In this work, we propose the MultiScale hypergraph for Trajectory Imputation and Prediction (MS-TIP), a novel approach that simultaneously addresses the imputation of missing observations and the prediction of future trajectories. Specifically, we leverage transformers with diagonal masked self-attention to impute incomplete observations. Further, our approach promotes complex interaction modeling through multi-scale hypergraphs, optimizing our trajectory prediction module to capture different types of interactions. With the inclusion of scenic attention, we learn contextual scene information, instead of sole reliance on coordinates. Additionally, our approach utilizes an intermediate control point and refinement module to infer future trajectories accurately. Extensive experiments validate the efficacy of MS-TIP in precisely predicting pedestrian future trajectories. Code is publicly available at `https://github.com/Pranav-chib/MS-TIP`.

1. Introduction

Predicting future trajectories of agents based on their previous motions is the primary objective of multi-agent trajectory prediction. This is not only crucial for various applications, including self-driving cars and understanding human motion patterns, but it also serves as a representation learning, bridging historical knowledge and future behaviors. Given the observed past trajectories, trajectory prediction requires modeling and predicting future trajectories. Capturing the social interaction among pedestrians is a critical task, and several efforts have been made to model these interactions. However, existing methods (Shi et al., 2021; Xu et al., 2022b;a) exhibit significant inaccuracies in predicting final goals due to the accumulation of errors from recursive predictions. To prevent the accumulation of errors caused by recursive trajectory prediction, endpoint prediction approaches have emerged in which the potential endpoints of trajectories are determined first (Wang et al., 2023; Mangalam et al., 2021; Chiara et al., 2022; Bae & Jeon, 2023), followed by interpolation. Although these solutions showed promising performance increases, difficulties still persisted. These methods are unable to handle missing values in past observed sequences. Additionally, these approaches often assume that the observed sequences of agents are complete, which is too stringent a condition for real-world scenarios. Sensor malfunctions, occlusion, and other issues frequently result in missing data in real-world observed trajectories (Xu et al., 2023). Predicting future trajectories with missing values in the observed sequence undoubtedly impacts prediction accuracy significantly. Trajectory prediction models must simultaneously handle trajectory imputation (filling missing values) and prediction with an accurate understanding of pedestrian interactions.

In prior time-series imputation methods, consideration is given to the following methods: the RNN-based methods (Yoon et al., 2019), which imputes missing values using an RNN graph, gated recurrent unit (GRU) variant, and bidirectional RNN. These methods are time-consuming and memory-constrained, making it challenging to address long-term dependence in time series. GAN-based (Liu et al., 2019) imputation methods utilize a generative adversarial network (GAN) structure to model temporal information of incomplete time series. However, due to their loss formulation, GANs are prone to non-convergence and collapse. On the other hand, VAE-based methods (Ramchandran et al., 2021) employ latent variables in sampling and imputation, which may not align with data distributions, making imputation challenging. Additionally, many of the aforementioned methods are autoregressive, leading to compounding errors.

[*]Equal contribution [1]Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee, India. Correspondence to: Pranav Singh Chib <pranavs_chib@cs.iitr.ac.in>, Pravendra Singh <pravendra.singh@cs.iitr.ac.in>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

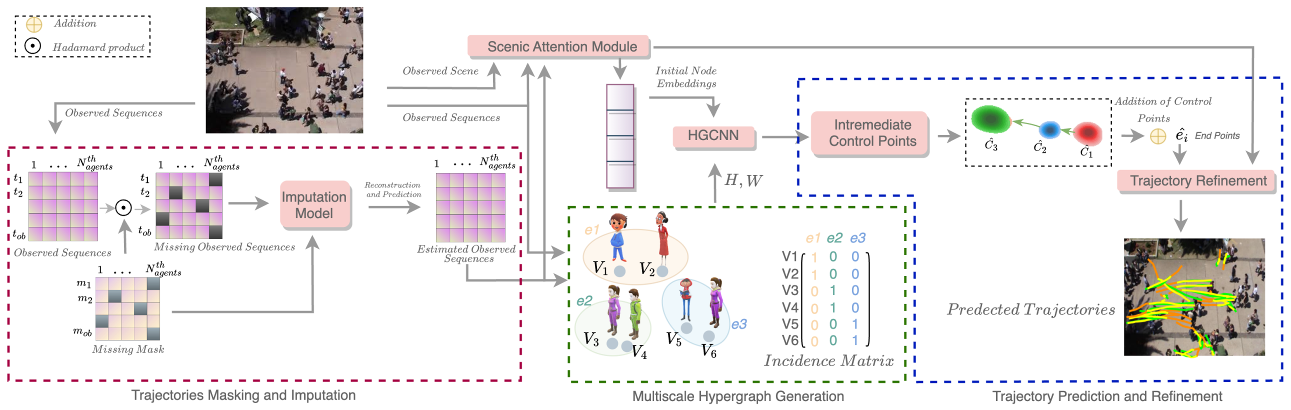


Figure 1. Visualization of the Three Core Modules of MS-TIP: the Imputation Module (indicated by a purple dotted line rectangle), Multiscale Hypergraph Module (teal dotted line rectangle), and Trajectory Prediction and Refinement Module (blue dotted line rectangle). Initially, given the observed sequences, the Imputation Module identifies the missing values in the past temporal sequences based on *NaN* values. A corresponding missing mask *(0,1 values)* is generated for the observed sequences. This 'missing mask' is then input into the Imputation Module during the imputation process. Once the estimated observed sequences are generated, the Multiscale Hypergraph Module produces hyperedges e_1, e_2, e_3, corresponding to the imputed sequence and interaction strengths at different scales (pairwise interaction denoted by $S = 2$ or group interaction by $S \geq 3$). The initial node embeddings from the scenic attention module, the incident matrix (H), and the diagonal weight matrix (W) from the hypergraph are input into the hypergraph convolution neural network (HGCNN), where hypergraph convolution is computed. The resultant embeddings are utilized for the prediction of control points. Incorporating the intermediate control points, we determine the endpoint, and subsequently, the Refinement Module refines the trajectory, resulting in the final plausible future trajectory predictions.

Self-attention (Shan & Oliva, 2021) emerges as a promising technique for imputation tasks. Its non-autoregressive nature overcomes RNNs' slow speed and memory limits and can reduce compounding errors, thereby improving imputation. While these approaches have demonstrated good imputation performance, they do not predict trajectories.

In this paper, we propose a novel approach, **M**ulti**S**cale hypergraph for **T**rajectory **I**mputation and **P**rediction (MS-TIP), as shown in Fig. 1. MS-TIP is an endpoint prediction approach that mitigates the accumulation of errors caused by recursive trajectory prediction. Our proposed approach is an end-to-end process that is capable of simultaneously handling trajectory imputation (finding missing values) and trajectory forecasting. Specifically, we leverage self-attention to learn missing values. We artificially introduce missing values in the observed coordinates using missing masks. Subsequently, we train the transformer-based model with diagonal masked self-attention to impute the missing values based on the missing mask. The Imputation module aims to impart knowledge about missing patterns in partial observations. For trajectory prediction, we propose a multiscale hypergraph with scenic attention. The multi-scale hypergraph captures group-wise interactions with varied group sizes. Furthermore, for incorporating physical scene information, we utilize scene attention, calculating the influence of a scene on social interactions among pedestrians. Our trajectory prediction model initially predicts the endpoint based on intermediate control points and then refines the interpolated trajectory. Extensive experimentation on pedestrian benchmarks consistently validates the efficacy of our proposed approach (MS-TIP).

2. Related Work

2.1. Trajectory Prediction

As predicting the future trajectory of an agent involves inherent uncertainty and often results in multiple possible outcomes, recent advancements have adopted the utilization of deep generative models (Mao et al., 2023; Xu et al., 2022a; Gu et al., 2022; Maeda & Ukita, 2023). These models encompass various techniques, including conditional variational autoencoders (CVAEs) (Xu et al., 2022a; Lee et al., 2022; Mangalam et al., 2020), generative adversarial networks (GANs) (Sadeghian et al., 2019; Hu et al., 2020; Gupta et al., 2018; Sadeghian et al., 2019), and diffusion models (Mao et al., 2023). Although notable progress, these stochastic prediction approaches have inherent drawbacks, such as unsteady training or the production of abnormal trajectories. Recently transformer-based (Girgis et al., 2022; Yuan et al., 2021; Yu et al., 2020; Sekhon & Fleming, 2021) models excel at capturing long-term dependencies. They also capture temporal and social elements simultaneously via the attention mechanism. Our primary focus is on endpoint-conditioned trajectory prediction (Bae & Jeon, 2023; Wang et al., 2023; Mangalam et al., 2020), which aims to forecast the final destination of an agent. Specifically, our approach emphasizes predicting the endpoint while conditioning on the intermediate points.

2.2. Graph-based Trajectory Prediction

Graph-based methods (Xu et al., 2022a; Yuan et al., 2021; Hu et al., 2020) have been specifically employed to model social interactions among agents in a scene through relational reasoning. Attention-based graph models (Chen et al., 2023; Huang et al., 2019; Kosaraju et al., 2019; Sekhon & Fleming, 2021) simulate agent-to-agent interactions during trajectory prediction. NMMP (Hu et al., 2020) explores message passing within the fully connected graph, calculating interaction weights for enhanced weighted social interaction modeling. SGCN (Shi et al., 2021) leverages self-attention to create a sparse graph for interaction learning. Graph convolution is another prominent approach in graph neural networks, as shown by Social-STGCNN (Mohamed et al., 2020). This method predicts paths using a Graph Convolutional Network (GCN) (Mendieta & Tabkhi, 2021), grouping spatial information of agents and incorporating physical constraints based on distances between pedestrians. However, its reliance on distance-related connections limits its effectiveness. The aforementioned methods only focus on the single interaction with the agent, ignoring the group and multiple interactions; we focus on modeling the different scales of interaction using the multiscale hypergraph.

2.3. Trajectory Imputation

Imputation is a nontrivial task because of the challenge to determine the missing values. For time-series imputation, Recurrent Neural Networks (RNNs) have been extensively utilized, as evidenced in works like BRITS (Cao et al., 2018) and M-RNN (Yoon et al., 2019), which employ bidirectional RNNs for missing value imputation. Despite their effectiveness, the recurrent network structure of RNNs leads to significant time and memory consumption, making them less suitable for long-term dependencies. Alternatively, GAN-based (Liu et al., 2019) approaches leverage both a generator and a discriminator to address missing values. Variational Auto-Encoder (VAE) models, such as GP-VAE (Fortuin et al., 2020), utilize a Gaussian Process (GP) prior in the latent space in conjunction with a variational auto-encoder for imputation. Although both VAE and GAN methodologies mitigate the compounding error issue inherent in RNNs, they are challenging to train and often suffer from non-convergence. Self-Attention-based (Shan & Oliva, 2021) approaches have emerged as a promising solution. The non-autoregressive nature and faster inference capabilities of self-attention make them particularly suitable for trajectory imputation tasks. However, only a few works, such as NAOMI (Liu et al., 2019) and GMAT (Zhan et al., 2018), have explored the trajectory imputation problem. These studies focus exclusively on trajectory imputation, neglecting the prediction task. INAM (Qi et al., 2020) and GC-VRNN (Xu et al., 2023) have addressed this limitation. INAM conducts imputation and prediction in an asynchronous manner, whereas GC-VRNN facilitates end-to-end imputation. Our approach employs a non-autoregressive transformer for imputing missing values across the temporal domain.

3. Methodology

3.1. Problem Formulation

For the problem formulation, we consider the trajectory sequence as a set of past and future sequences. Here, $X_i^{\leq t_{ob}} = \{X_i^{t_1}, X_i^{t_2}, ..., X_i^{t_{ob}}\}$ represents the past observed trajectory of agent i, ranging from t_1 to t_{ob}. The $X_i^{t_{ob}} \in \mathbb{R}^2$ denotes the 2D coordinates of agent i at time step t_{ob}. Similarly, the future trajectory sequence (ground truth) can be represented as $Y_i^{t_{ob+1} \leq t \leq t_{pred}}$ over the duration from t_{ob+1} to t_{pred}. The problem of trajectory prediction aims to forecast the future trajectory $\widehat{Y}_i^{t_{ob+1} \leq t \leq t_{pred}}$ based on observed past trajectory $X_i^{\leq t_{ob}}$.

3.2. Trajectory Imputation and Prediction

We use Self Attention (Du et al., 2023) for the trajectory imputation to increase the model robustness by imputing missing coordinates. The inputs to the imputation model are the input coordinates and the mask, as depicted in Fig. 2.

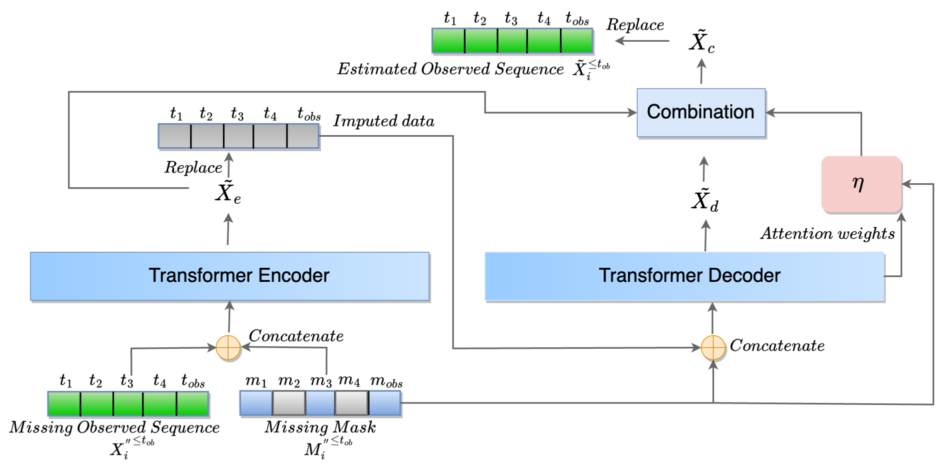
3.2.1. Trajectories Masking

Imputation is the process whereby the model fills in missing coordinates in the given input. These missing coordinates are introduced as null values ($NaNs$) in the original input coordinates $X_i^{\leq t_{ob}}$ to simulate missing coordinates in our dataset. The resulting sequence of coordinates after introducing null values is denoted by $X'^{\leq t_{ob}}_i$.

We get $X_i^{''\leq t_{ob}}$ by artificially masking (making those values as $NaNs$) some % of the observed coordinates in $X_i^{'\leq t_{ob}}$. $X_i^{''\leq t_{ob}}$ is a result of both, original missing values and artificially masked values. $I_i^{\leq t_{ob}}$ denotes the indicating mask, expressed as $\{I_i^{t_1}, ..., I_i^{t_{ob}}\}$.

$$I_i^t = \begin{cases} 1 & \text{if } X_i^{'t} \text{ is observed \& } X_i^{''t} \text{ is artificially masked} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We then obtain missing mask $M_i^{''\leq t_{ob}}$, which takes both originally missing values and artificially masked values into



t_1 t_2 t_3 t_4 t_{obs}
Replace
$\tilde{X}_c$
Estimated Observed Sequence $\tilde{X}_i^{\le t_{ob}}$
Combination
Imputed data
Replace
$\tilde{X}_e$
η
$\tilde{X}_d$
Attention weights
Transformer Encoder
Transformer Decoder
Concatenate
Concatenate
m_1 m_2 m_3 m_4 m_{obs}
Missing Observed Sequence $X_i^{''\le t_{ob}}$
Missing Mask $M_i^{''\le t_{ob}}$

Figure 2. Illustration of the imputation model, comprising three main blocks, each dedicated to a distinct operation. The 'encode block' predicts the imputed data, which is then passed to the 'decoder block.' Concurrently, the 'η block' processes the attention weights and the mask. A weighted combination ($\tilde{X}_c$) of representations from the encoder and decoder blocks ($\tilde{X}_e$ and $\tilde{X}_d$) is combined by the 'combination block.'

account.

$$M_i^{''t} = \begin{cases} 1 & \text{if } X_i^{''t} \text{ is observed} \\ 0 & \text{if } X_i^{''t} \text{ is missing} \end{cases} \tag{2}$$

The imputation model Θ_{imp} does its imputation using the missing observed sequence ($X_i^{''\le t_{ob}}$) and missing mask ($M_i^{''\le t_{ob}}$), defined above. Indicating mask $I_i^{\le t_{ob}}$ is used for the Masked Imputation Task (MIT), while the Observed Reconstruction Task (ORT) uses $M_i^{''\le t_{ob}}$ for ensuring that the distribution of the predicted values is close to the actual observed values (see Eqs. 6, 7).

3.2.2. Trajectories Imputation

We employ a transformer architecture comprising an encoder block, a decoder block, and a linear combination of both for imputing missing values, as illustrated in Fig. 2. Our approach includes a modification of the self-attention mechanism into a diagonal-masked self-attention format, where the diagonal entities of the attention map are masked (set to infinity). This modification aims to enhance feature correlation across different timestamp values and improve the learning of temporal representations.

The input coordinates (missing observed sequence) and the missing mask are concatenated to form the input to the encoder block, after which the missing values are replaced with $\tilde{X}_e$ as shown in Fig. 2. The decoder block processes the output from the encoder block, concatenated with the missing mask. The η block, as defined in Eq. 3, utilizes the attention weight (A) and missing mask ($M_i^{''\le t_{ob}}$) to produce combining weights η. The combination of $\tilde{X}_e$ and $\tilde{X}_d$ with η yields $\tilde{X}_c$, as depicted in Eq. 4. The corresponding values of $\tilde{X}_c$ are then utilized to replace those in the missing observed sequence.

$$\eta = \text{Sigmoid}\left(\text{Concat}\left(A, M_i^{''\le t_{ob}}\right) W_\eta + b_\eta\right) \tag{3}$$

$$\tilde{X}_c = (1-\eta) \odot \tilde{X}_e + \eta \odot \tilde{X}_d \tag{4}$$

$$\tilde{X}_i^{\le t_{ob}} = M_i^{''\le t_{ob}} \odot X_i^{''\le t_{ob}} + \left(1 - M_i^{''\le t_{ob}}\right) \odot \tilde{X}_c \tag{5}$$

Here, $\tilde{X}_e$ denotes the representation learned from the encoder block, $\tilde{X}_d$ is the representation learned from the decoder block, and $\tilde{X}_c$ is a linear combination of both, as defined by Eq. 4. The learnable parameters W_η and b_η are used in this computation. The symbol $\odot$ denotes an element-wise product. The estimated observed sequence $\tilde{X}_i^{\le t_{ob}}$ is obtained via Eq. 5, in which the missing values in $X_i^{''\le t_{ob}}$ are replaced with the corresponding values from $\tilde{X}_c$. This estimated observed sequence for each agent is then fed into the multiscale hypergraph module for capturing interactions, as depicted in Fig. 1.

$$\mathcal{L}_{ORT} = MAE((||\tilde{X}_i^{\le t_{ob}} - X_i^{'\le t_{ob}}||) \odot (M_i^{''\le t_{ob}})) \tag{6}$$

$$\mathcal{L}_{MIT} = MAE((||\tilde{X}_i^{\le t_{ob}} - X_i^{'\le t_{ob}}||) \odot (I_i^{\le t_{ob}})) \tag{7}$$

$$\tilde{X}_i^{\le t_{ob}} = \Theta_{imp}(X_i^{''\le t_{ob}}) \tag{8}$$

$$\mathbf{\Theta^\star_{imp}} = \underset{\boldsymbol{\Theta_{imp}}}{\arg\min}(\mathcal{L}_{\text{ORT}} + \mathcal{L}_{\text{MIT}}) \tag{9}$$

Where MAE refers to Mean Absolute Error, $\mathcal{L}_{ORT}$ calculates loss between the estimated observed sequence and the original observed sequence using the missing mask and $\mathcal{L}_{MIT}$ calculates the loss using the indicating mask.

We pre-train the imputation model Θ_{imp} using Eq. 9. The pre-trained imputation model obtained thereafter is fine-tuned during the end-to-end training.

3.3. HyperGraph Topology and Generation

In our proposed approach, we train our model using both the original observed sequence $X_i^{\leq t_{ob}}$ and the estimated observed sequence $\tilde{X}_i^{\leq t_{ob}}$. The initial step involves generating a hypergraph from $X_i^{\leq t_{ob}}$ and subsequently training the model end-to-end on this input. We then repeat the process by passing the estimated observed sequence, $\tilde{X}_i^{\leq t_{ob}}$, and again training the model end-to-end. This is performed for every batch of the given dataset to effectively capture the impact of imputation on the predicted final coordinates. Throughout the remainder of the paper, we exclusively present formulations using $X_i^{\leq t_{ob}}$ for the sake of simplicity. However, it is important to note that the same formulation is applicable to $\tilde{X}_i^{\leq t_{ob}}$. We construct a multi-scale hypergraph to effectively capture interactions among various agents.

3.3.1. Hyper Graph Topology

Our objective is to create a multi-scale hypergraph that shows how different groups of agents interact in various ways. Each scale of the hypergraph will represent a different kind of interaction with different numbers of agents/nodes involved. Formally, our aim is to generate a set of hypergraphs $\mathcal{G}^{(t)} = \{\mathcal{G}^{(t,S_1)}, \mathcal{G}^{(t,S_2)}, \ldots, \mathcal{G}^{(t,S_{|S|})} \mid t \in \mathbb{N}, 1 \leq t \leq t_{ob}, S_j \in S, 1 \leq j \leq |S|\}$, where S is a set of predefined scales, illustrating the connections between different nodes (agents). Let $\mathcal{G}^{(t,S_j)} = (\mathcal{V}, \mathcal{E}^{(S_j)})$ denote the hypergraph generated at time t and scale S_j, with the hyperedge set $\mathcal{E}^{(t,S_j)} = \{e_1^{(t,S_j)}, e_2^{(t,S_j)}, \ldots, e_{M_s}^{(t,S_j)}\}$ and the vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. The scale parameter S_j indicates the number of agents involved in the interactions, with a larger S_j value representing a larger group of agents. As the number of agents (and consequently, the number of nodes) in the hypergraph is variable, a flexible scaling mechanism is implemented. This involves using a predefined set of scales S = { 2, 3, 5, 7, 9 }. However, if the number of agents exceeds the largest scale in this set, the scaling stops. For instance, if there are 6 agents, the scales used would be 2, 3, and 5; whereas, if there are 10 agents, the scales would be 2, 3, 5, 7, and 9.

3.3.2. Hyper Graph Generation

To infer a multiscale hypergraph, we utilize the K-Nearest Neighbors (KNN) algorithm to generate the hypergraph, employing the Euclidean distance as the metric. In the given trajectory, each agent is represented as a node within the hypergraph. The number of edges in the hypergraph $\mathcal{G}^{(t,S_j)}$ is N, leading to a total of $|S| \times N$ hyperedges in $\mathcal{G}^{(t)}$, where $|S|$ is the total number of scales. In the hypergraph $\mathcal{G}^{(t,S_j)}$, the i^{th} hyperedge encompasses S_j nodes, including the i^{th} node and the $S_j - 1$ nearest neighbors to the i^{th} node. For instance, in a hypergraph generated at $scale = 3$, i.e., $\mathcal{G}^{(t,3)}$, there will be N hyperedges; the i^{th} hyperedge will consist of the i^{th} node and the two nearest nodes among the remaining $N - 1$ nodes. Subsequently, we obtain an incidence matrix $\mathbf{H}^{(t,S_j)} \in \mathbb{R}^{|\mathcal{V}| \times |N|}$ of $\mathcal{G}^{(t,S_j)}$. The incidence matrix $\mathbf{H}^{(t)} \in \mathbb{R}^{|\mathcal{V}| \times |SN|}$ of $\mathcal{G}^{(t)}$ can be calculated by $\mathbf{H}^{(t)} = \left[\mathbf{H}^{(t,S_1)} \middle| \mathbf{H}^{(t,S_2)} \middle| \ldots \mid \mathbf{H}^{(t,S_{|S|})}\right]$, where $\mid$ denotes the horizontal concatenation operation. Two distinct approaches are used for assigning weights to the hyperedges. The first approach assigns a uniform weight of 1 to every hyperedge, ensuring equal weightage in the convolution process. The second approach, in contrast, involves assigning varying weights to the hyperedges given by Eq. 10.

$$w(e_i^{(t,s)}) = \sum_{u,v \in e_i^{(t,s)}} \exp\left(-\frac{d(X_u^t, X_v^t)^2}{\sigma^2}\right) \tag{10}$$

Where u and v are any two vertices in the hyperedge $e_i^{(t,s)}$. $d(X_u^t, X_v^t)$ represents the Euclidean distance between vertices u and v, with σ denoting the mean value of distances between all vertex pairs. Subsequently, $\mathbf{w}^{(t)} = \left[\mathbf{w}^{(t,S_1)} \middle| \mathbf{w}^{(t,S_2)} \middle| \ldots \mid \mathbf{w}^{(t,S_{|S|})}\right]$, where $\mid$ denotes the horizontal concatenation operation. The ablation study for these two methods of weight assignment is detailed in the ablation study Section 4.4.3. Utilizing the hypergraph, we obtain the incidence matrix $\mathbf{H}^t$ and weight $\mathbf{w}^t$, which is passed to HGCNN along with the initial node embeddings (refer to Section 3.4), as shown in Fig. 1.

3.4. Scenic Attention

In order to effectively capture the physical scene information, we employ a pre-trained convolutions neural network. This is done to extract features from scene images. The chosen backbone network is the ImageNet pre-trained VGG-19 model, and the resultant features are denoted as V_k

$$\begin{aligned} V_k &= VGG19\left(Image_{\text{scene}}\,; W_{VGG19}\right) \\ S_i^t &= \text{SceneAttention}\left(V_k, X_i^t; W_{att}\right) \end{aligned} \tag{11}$$

Here, W_{VGG19} and W_{att} represent the weights of the VGG-19 model and trainable attention weights, $Image_{\text{scene}}$ is the image from the dataset, X_i^t is the position of the agent i at time step t. Then we concatenate scene embedding S_i^t with the coordinate of the agents (nodes) to form initial node embeddings $z_i^{(t,0)}$.

$$z_i^{t,0} = Concate\left(X_i^t, S_i^t\right) \tag{12}$$

$$Z^{t,0} = \{z_1^{t,0}, z_2^{t,0}, \dots, z_N^{t,0}\} \tag{13}$$

Here, the term $z_i^{t,0}$ denotes the initial node embeddings of the i^{th} node obtained after concatenating them with the scene embeddings at time t, where the superscript 0 represents the initial node embeddings and $Z^{t,0}$ represents the initial node embeddings of all the nodes. When these initial embeddings are fed as input to the HGCNN module, denoted by Eq. 14, we get the final node embeddings. The inclusion of scene information significantly impacts the accuracy of pedestrian prediction. The motivation behind this inclusion is to ensure that the model doesn't solely depend on coordinates but also considers the contextual information provided by the scene.

3.5. HyperGraph Convolution Neural Network

Node embeddings corresponding to the convolution operator are obtained by passing the incidence matrix H^t, initial node embedding $Z^{t,0}$, and edge weight W^t to the HGCNN module. The hypergraph convolution operation is represented in Eq. 14.

$$z_i^{t,(l+1)} = \sigma\left(\sum_{j=1}^{N}\sum_{e=1}^{M} H_{ie}^t H_{je}^t W_{ee}^t z_j^{(t,l)} \mathbf{P}\right) \tag{14}$$

Here, N is the total number of nodes, $M = |S| \times N$ is the total number of hyperedges, $\{z_i^{(t,l+1)} \mid 1 \le i \le N\}$ denotes the node embeddings of the i^{th} node at t^{th} time after $(l+1)$ convolution operations, H^t is the incidence matrix (analogous to the adjacency matrix in a normal graph), $W^t \in \mathbb{R}^{M\times M} = diag(\mathbf{w^t})$ is a diagonal matrix where $\mathbf{w^t} \in \mathbb{R}^{1\times M}$, and $\mathbf{P}$ is the trainable weight matrix. $\sigma(\cdot)$ represents a nonlinear activation function. In simple terms, the equation calculates the new embedding for the i^{th} node by considering all edges connected to it. For each edge, if both i^{th} and $j^{th}\{1 \le j \le N\}$ is present, it multiplies the edge weight and the current embedding of the connected j^{th} node. The sigmoid activation is then applied to the sum of these products. The resulting embeddings $\mathcal{Z} = \left[Z^{1,l+1}, Z^{2,l+1}, \dots, Z^{t_{ob},l+1}\right]$ are then passed to the trajectory prediction and refinement module for future trajectory predictions. Here, $Z^{t,l+1}$ denotes the node embeddings of all agents at t^{th} time and after $(l+1)^{th}$ convolution operation. $\mathcal{Z}$ represents the node embeddings of all agents at all time stamps after the final convolution layer.

3.6. End Point Prediction and Refinement

Our model adopts an endpoint-conditioned approach (Bae & Jeon, 2023; Mangalam et al., 2020; 2021) for predicting future pedestrian trajectories. Initially, we segment a pedestrian's trajectory into intermediate control points. These points outline the path a pedestrian is likely to take a route to their final destination. The locations of these control points are inferred as probability densities using mixture density networks (MDNs). Subsequently, the final endpoint, representing the pedestrian's destination, is determined by linking these control points, as illustrated in Eq. 17.

$$\begin{gathered} C_i = \left\{c_i^k = Y_i^{t_{ob}+\tau\times k} - Y_i^{t_{ob}+\tau\times(k-1)}\right\} \\ e_i = X_i^{t_{ob}} + \sum_{k=1}^{K} c_i^k \\ \text{for } \forall k \in \{1,\dots,K\}, \\ \text{for } \forall i \in \{1,\dots,N\}, \quad \tau = \frac{t_{pred}-t_{ob}}{K} \end{gathered} \tag{15}$$

Where K represents the total number of control points. We can obtain the ground truth control points C_i for the training dataset, since we know the t_{pred} coordinates for every pedestrian, and we denote them as $X_i^{t_{ob}+\tau\times k}$ as the pedestrian coordinates for K control points between t_{ob} to t_{pred}. We now have to predict the control points. Following the previous study (Gupta et al., 2018), we sample $\Omega = 20$ times to obtain $\hat{C}^\omega$, where $1 \le \omega \le \Omega$. We sample $\hat{C}_i^\omega = \{\hat{c}_i^{k,\omega}\}$ from the Gaussian Mixture Models (GMM) parameters predicted from the endpoint prediction module as shown below.

$$\hat{C}^\omega \sim GMM(\mathcal{Z}) \tag{16}$$

$$\hat{e}_i^\omega = X_i^{t_{ob}} + \sum_{k=1}^{K} \hat{c}_i^{k,\omega} \tag{17}$$

Here, $\hat{e}_i^\omega$ denotes the final endpoint of the i^{th} agent, and $\hat{c}_i^{k,\omega}$ represents the k^{th} intermediate control point of i^{th} agent in the ω^{th} sample time. $\hat{E}_i = \{\hat{e}_i^\omega \mid i,\omega \in \mathbb{N}, 1 \le \omega \le \Omega, 1 \le i \le N\}$. We divide the predicted endpoints into two sets: a positive set $\Upsilon^+$and a negative set Υ^- given below:

$$\begin{gathered} \Upsilon_i^+ = \{\hat{e}_i^\omega \mid \|\hat{e}_i^\omega - e_i\| \le \Gamma\} \\ \Upsilon_i^- = \{\hat{e}_i^\omega \mid \|\hat{e}_i^\omega - e_i\| > \Gamma\} \\ \text{for } \forall \omega \in \{1,\dots,\Omega\}, \Gamma = \frac{\left\|X_i^{t_{ob}} - X_i^{t_1}\right\|}{t_{ob}\times\gamma_i}, \end{gathered} \tag{18}$$

where γ_i is a scale indicator that adaptively adjusts the averaged displacement of the i^{th} agent. We calculate the loss and backpropagate gradients for only the positive sets of endpoints using the valid mask given below:

$$\psi_i^\omega = \begin{cases} 1 & \text{if } \hat{e}_i^\omega \in \Upsilon_i^+ \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

In early training, we will have very few accurate predictions and, hence, very few positive endpoints. We address this by additionally sampling a set of Ω positive endpoints, called "guided endpoints," within a specified range Γ around the

Table 1. Details of the ETH, HOTEL, UNIV, ZARA1, and ZARA2 datasets. 'Obs frames', 'Pred frames', 'Obs time', and 'Pred time' denote observed frames, predicted frames, observed time horizon, and predicted time horizon, respectively.

Datasets	Pedestrians	Obs frames	Pre frames	Obs time	Pred time
ETH	750	8	12	3.2 sec	4.8 sec
HOTEL		8	12	3.2 sec	4.8 sec
UNIV	786	8	12	3.2 sec	4.8 sec
ZARA1		8	12	3.2 sec	4.8 sec
ZARA2		8	12	3.2 sec	4.8 sec

actual values. Hence total number of control points is 2Ω. This supplements the limited number of positive examples during the initial phases. Unlike existing methodologies, our approach integrates a multiscale hypergraph to enhance the prediction of both endpoints and control points, effectively capturing dynamic interactions among pedestrians. After predicting control points and generating endpoints, our trajectory refinement module introduces correction vector fields $f_i^{t,\omega}$ for the initial trajectory, which is added to the trajectory obtained from linear interpolation. This integration enables our model to predict accurate future trajectories.

$$\widehat{Y}_i^{t,\omega} = X_i^{t_{ob}} + \frac{\hat{e}_i^{\omega} - X_i^{t_{ob}}}{t_{pred} - t_{ob}} \times (t - t_{ob}) \quad \forall t \in \{t_{\text{obs}} + 1, \ldots, t_{\text{pred}}\}, \; \forall n \in \{1, \ldots, N\}, \; \forall \omega \in \{1, \ldots, \Omega\} \tag{20}$$

$$f_i^{t,\omega} = \Theta_{tr}(Concat(\widehat{Y}_i^{t,\omega}, S_i^t)) \quad \widehat{Y}_i^{t,\omega} = \widehat{Y}_i^{t,\omega} + f_i^{t,\omega} \quad \forall t \in \{t_{\text{ob}} + 1, \ldots, t_{\text{pred}}\}, \; \forall n \in \{1, \ldots, N\} \; \forall \omega \in \{1, \ldots, \Omega\} \tag{21}$$

where Θ_{tr} is trajectory refinement model, S_i^t is the scenic attention (refer 3.4).

Our approach incorporates the control point prediction and, lastly, the refinement loss. For the control point loss (Eq. 22), we optimize an expectation to train the control point prediction module. We sum the probability density functions for all predicted control point distributions and pedestrians. Furthermore, we minimize the trajectory refinement loss as shown in Eq. 23. The loss is based on the mean square error (MSE) of the average displacement between a refined and

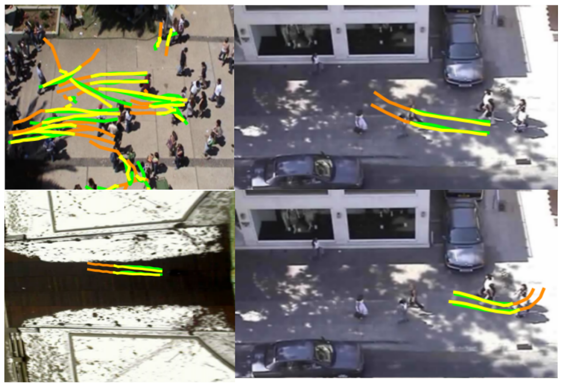


Figure 3. Visualization of Predicted Trajectories in the images of UNIV (top-left), ETH (bottom-left), and ZARA (both on the right) datasets. Predicted pedestrian trajectories are depicted in yellow. The observed trajectories are shown in orange, while the ground truth trajectories are represented in green. Our approach accurately predicts future trajectories, closely aligning with the ground truth.

ground truth trajectory.

$$\mathcal{L}_c = \sum_{i=1}^{N}\sum_{k=1}^{K} -\log\left[\sum_{m=1}^{\#_M} \hat{\pi}_i^{m,k} \frac{\exp\left(-\frac{\left(c_i^k - \hat{\boldsymbol{\mu}}_i^{m,k}\right)^2}{2\left(\hat{\boldsymbol{\sigma}}_i^{m,k}\right)^2}\right)}{\sqrt{2\pi}\hat{\boldsymbol{\sigma}}_i^{m,k}}\right] \tag{22}$$

Here, $\hat{\boldsymbol{\mu}}_i^{m,k}$ is the mean, $\hat{\boldsymbol{\sigma}}_i^{m,k}$ is the standard deviation, and $\hat{\pi}_i^{m,k}$ is a mixing coefficient obtained from the multivariate GMM (Bae & Jeon, 2023), N is the number of agents, and K is the number of control points. c_i^k is the ground truth control point. $\#_M$ is the number of selected mixture models, which are used to sample the predicted control point $\hat{c}_i^k$.

$$\mathcal{L}_r = \sum_{i=1}^{N}\sum_{\omega=1}^{2\Omega}\sum_{t=t_{ob+1}}^{t_{pred}} \psi_i^{\omega} \left|\left|Y_i^{t,\omega} - \widehat{Y}_i^{t,\omega}\right|\right|_2^2 \tag{23}$$

Here, ψ_i^{ω} is the mask used while backpropagating gradients. $Y_i^{t,\omega}$ are the coordinates of the ground truth trajectory and $\widehat{Y}_i^{t,\omega}$ are the predicted coordinates (calculated from Eq. 21). Finally, the trajectory prediction loss function is given by $\mathcal{L}_{total} = \mathcal{L}_c + \mathcal{L}_r$, where $\mathcal{L}_c$ is control point prediction loss and $\mathcal{L}_r$ is the trajectory refinement loss.

4. Experiments

In this section, we present the quantitative and qualitative results of our approach. Details regarding the implementation are provided in the Appendix.

4.1. Dataset and Metrics

We evaluated our method on the widely used publicly available human trajectory prediction benchmarks ETH, HOTEL,

Table 2. Comparison of MS-TIP (Our) with other approaches on ETH, HOTEL, UNIV, ZARA1, and ZARA2 datasets in terms of ADE/FDE (lower values are better). All approaches use the observed 8-time steps and produce the future 12-time steps. The top performance is highlighted in **bold**, and the second-best performance is indicated with <u>underline</u>.

Model	STGAT	CARPE	SGCN	GroupNet	STT	S-Implicit	MemoNet	BCDiff	G-Tern	FlowChain	Our
Venue	AAAI-21	AAAI-21	CVPR-21	CVPR-22	CVPR-22	ECCV-22	CVPR-22	NIPS-23	AAAI-23	ICCV-23	-
ETH	0.56/1.10	0.80/1.40	0.52/1.03	0.46/0.73	0.54/1.10	0.66/1.44	1.00/2.08	0.53/0.91	<u>0.42/0.58</u>	0.55/0.99	**0.39/0.57**
HOTEL	0.27/0.50	0.52/1.00	0.32/0.55	0.15/0.25	0.24/0.46	0.20/0.36	0.35/0.67	0.17/0.27	<u>0.14/0.23</u>	0.20/0.35	**0.13/0.22**
UNIV	0.32/0.66	0.61/1.23	0.37/0.70	0.26/0.49	0.57/1.15	0.31/0.60	0.55/1.19	<u>0.24/0.40</u>	0.26/0.45	0.29/0.54	**0.24/0.40**
ZARA1	0.21/0.42	0.42/0.84	0.29/0.53	0.21/0.39	0.45/0.94	0.25/0.50	0.46/1.00	<u>0.21/0.37</u>	<u>0.21/0.37</u>	0.22/0.40	**0.20/0.34**
ZARA2	0.20/0.40	0.34/0.74	0.25/0.45	0.17/0.33	0.36/0.77	0.22/0.43	0.32/0.71	**0.16/0.26**	0.17/0.29	0.20/0.34	<u>0.17/0.29</u>
AVG	0.31/0.62	0.46/0.89	0.37/0.65	0.25/0.44	0.43/0.88	0.33/0.67	0.55/1.15	0.26/0.44	<u>0.24/0.38</u>	0.29/0.52	**0.22/0.36**

Table 3. Ablation experiments using different missing mask values on ETH, HOTEL, UNIV, ZARA1, and ZARA2 datasets. The best results were achieved when missing coordinates were generated by introducing 5% and 15% null values ($NaNs$) in the total observed trajectories. **Bold** represents the best, and <u>underline</u> represents the second-best ADE/FDE values.

Imputation %	5 %	10 %	15 %	20 %
ETH	0.43/0.60	<u>0.40/0.61</u>	**0.39/0.57**	0.46/0.65
HOTEL	**0.13/0.22**	0.14/0.24	<u>0.14/0.23</u>	0.15/0.24
UNIV	**0.24/0.40**	<u>0.24/0.40</u>	0.24/0.41	0.25/0.40
ZARA1	**0.20/0.34**	0.21/0.35	0.21/0.35	<u>0.21/0.34</u>
ZARA2	0.18/0.30	<u>0.17/0.29</u>	**0.17/0.29**	0.20/0.35
AVG	<u>0.23/0.37</u>	0.23/0.38	**0.23/0.37**	0.25/0.39

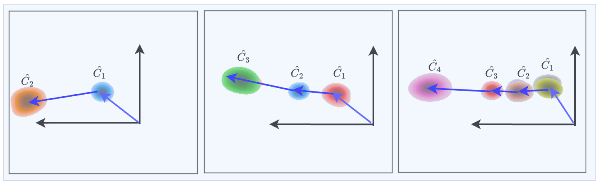


Figure 4. Visualization of predicted control points. The control points are represented by areas of varying color density, indicating the probability density of each intermediate step a pedestrian is likely to take toward the endpoint. The control points are connected by a blue line, depicting the intermediate straight curve.

Table 4. Ablation experiments evaluating the contributions of imputation (IMP), scenic attention (SCA), and the multiscale hypergraph (MHG) in our approach (MS-TIP). Empirical evaluation shows that all three components (combined) are necessary for the effectiveness of MS-TIP.

Components	Baseline	IMP	IMP+SCA	IMP+SCA+MHG
ETH	0.47/0.88	0.43/0.69	0.40/0.61	0.39/0.57
HOTEL	0.15/0.26	0.15/0.24	0.14/0.24	0.13/0.22
UNIV	0.25/0.41	0.24/0.40	0.24/0.40	0.24/0.40
ZARA1	0.21/0.35	0.21/0.34	0.20/0.34	0.20/0.34
ZARA2	0.18/0.30	0.18/0.31	0.18/0.30	0.17/0.29
AVG	0.25/0.44	0.24/0.39	0.23/0.37	**0.22/0.36**

UNIV, ZARA1, and ZARA2. More details are provided in Table 1. We use popular assessment measures for trajectory prediction, such as Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE denotes the average L2 distance between anticipated and ground truth trajectories over all time steps, whereas FDE measures the L2 distance at the final time step or endpoint.

4.2. Quantitative Results

The quantitative results on ETH, HOTEL, UNIV, ZARA1, and ZARA2 are presented in Table 2. We have compared our approach with STGAT (Sekhon & Fleming, 2021), CARPE (Mendieta & Tabkhi, 2021), SGCN (Shi et al., 2021), GroupNet (Xu et al., 2022a), STT (Monti et al., 2022), S-Implicit (Mohamed et al., 2022), MemoNet (Xu et al., 2022b), BCDiff (Li et al., 2023), G-Tern (Bae & Jeon, 2023), and FlowChain (Maeda & Ukita, 2023). In all five datasets, our approach has demonstrated the best performance over ETH, HOTEL, UNIV, and ZARA1 datasets and the second-best result over ZARA2. Our approach outperforms the state-of-the-art endpoint prediction method Graph-TERN (Bae & Jeon, 2023), achieving an average ADE of 0.22 and an average FDE of 0.36. The results presented in Table 2 include training MS-TIP on missing coordinates set at 15% for the ETH and ZARA2 datasets, and 5% for the HOTEL, UNIV, and ZARA1 datasets.

4.3. Qualitative Results

As demonstrated in Fig. 3, our method accurately predicts future trajectories close to the ground truth trajectories. The model effectively learns various patterns in the scene, capturing the interactions and movement of pedestrians. Additionally, we have illustrated the predicted control points enhancing goal prediction. The inference of intermediate control points of various sizes (2, 3, 4) is depicted in Fig. 4.

4.4. Ablation Study

4.4.1. Different Values of Imputation

Table 3 shows the different variations of missing values (indicating how many values are to be masked). Missing co-

Table 5. Ablation experiments involving different hyperedge weights. We investigate the result using Eq. 10, which assigns varying weights (VAR) to the hyperedges. Results illustrate that prediction accuracy is highest when a uniform weight of 1 is assigned to every hyperedge (UNI). **Bold** represents the best, and <u>underline</u> represents the second-best ADE/FDE values.

Impuation/	5%		10%		15%		20%	
Weights	VAR	UNI	VAR	UNI	VAR	UNI	VAR	UNI
ETH	0.44/0.61	0.43/0.60	0.43/0.60	0.41/0.61	0.44/0.62	0.39/0.57	0.43/0.60	0.46/0.65
HOTEL	0.14/0.24	0.14/0.23	0.14/0.24	0.15/0.25	0.14/0.24	0.13/0.22	0.14/0.24	0.15/0.24
UNIV	0.24/0.44	0.25/0.40	0.24/0.41	0.25/0.46	0.24/0.41	0.24/0.40	0.25/0.40	0.25/0.40
ZARA1	0.20/0.33	0.21/0.35	0.20/0.33	0.21/0.35	0.20/0.34	0.20/0.34	0.21/0.35	0.21/0.40
ZARA2	0.17/0.29	0.18/0.30	0.17/0.30	0.18/0.29	0.17/0.29	0.17/0.29	0.17/0.29	0.21/0.35
AVG	0.24/0.38	0.24/0.38	<u>0.23/0.38</u>	0.24/0.39	0.24/0.38	**0.22/0.36**	0.24/0.38	0.25/0.39

Table 6. Ablation experiments to investigate the impact of different intermediate predicted control points on the final trajectory prediction in terms of ADE/FDE. The optimal result is obtained when predicting three intermediate control points. **Bold** represents the best, and <u>underline</u> represents the second-best ADE/FDE values.

Control points	K=1	K=2	K=3	K=4	K=6
ETH	0.44/0.76	0.47/0.70	0.39/0.57	0.42/0.69	0.47/0.85
HOTEL	0.19/0.32	0.15/.025	0.13/0.22	0.14/0.24	0.13/0.23
UNIV	0.25/0.43	0.25/0.42	0.24/0.40	0.25/0.43	0.25/0.44
ZARA1	0.24/0.41	0.21/0.36	0.20/0.34	0.21/0.35	0.21/0.36
ZARA2	0.21/0.37	0.19/0.33	0.17/0.29	0.19/0.33	0.18/0.32
AVG	0.26/0.46	0.21/0.41	0.22/0.36	0.24/0.411	0.25/0.44

Table 7. The variation of ADE/FDE with the scale parameter for MS-TIP on the ETH dataset.

Dataset	Scale	ADE	FDE
ETH	2,3	0.41	0.65
	2,3,5	0.40	0.59
	2,3,5,7,9	**0.39**	**0.57**
	2,3,4,5,6,7,8,9	0.40	0.57

ordinates are generated by introducing null values ($NaNs$) in the total observed trajectories. The results demonstrate that the best performance in the trajectory prediction task is achieved by training the imputation model to reconstruct and impute 5% and 15% $NaNs$ values.

4.4.2. Contribution of Different Components

Table 4 presents the contributions of imputation (IMP), scenic attention (SCA), and the multiscale hypergraph (MHG) in our approach (MS-TIP) compared to the baseline that uses a simple GCN (graph convolutional network). Imputation enhances model robustness to incomplete sequences, while the hypergraph and scenic attention facilitate interaction learning in dynamic scenarios. Notably, the incorporation of a hypergraph yields a more substantial relative improvement in model performance.

4.4.3. HyperEdge Weight

Table 5 presents an investigation of various ADE/FDE values when the hyperedges weights are assigned using Eq. 10 under different values of the imputation. It is evident from the table that the highest prediction accuracy is achieved when the weight of every hyperedge is set to 1.

4.4.4. Variation in Control Points

We evaluate the model's performance using different numbers of intermediate control points, and our findings are presented in Table 6. The optimal results are achieved when the value of K is set to 3, indicating the prediction of three intermediate control points.

4.4.5. Variation in Scale Parameter

Based on the results presented in Table 7, we observe that when the scale is small (scale = [2, 3]), the ADE/FDE metrics tend to be relatively higher. However, as the scale increases, MS-TIP demonstrates improved capture of social interactions, resulting in a decrease in both ADE/FDE. Interestingly, the ADE/FDE values for scales (2, 3, 5, 7, 9) and (2, 3, 4, 5, 6, 7, 8, 9) appear to be nearly identical. It's important to note that each scale generates a hypergraph, and employing more scales introduces a greater number of hyper-edges. Increasing the number of scales further does not significantly enhance the ADE/FDE metrics. Moreover, a large number of scales incurs greater convolutional complexity and raises concerns about capturing false interactions.

5. Conclusion

In this paper, we present a novel approach capable of simultaneously addressing trajectory sequence imputation and future trajectory prediction tasks. Our proposed framework utilizes a non-autoregressive transformer model to impute missing coordinates, employing a dual approach of missing value reconstruction and prediction. By integrating a multi-scale hypergraph, our model efficiently captures complex interactions at different scales. Furthermore, the introduction of scenic attention facilitates contextual learning among pedestrians in various environments. Extensive experiments demonstrate the effectiveness of our approach.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bae, I. and Jeon, H.-G. A set of control points conditioned pedestrian trajectory prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6155–6165, 2023.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.

Chen, X., Zhang, H., Hu, Y., Liang, J., and Wang, H. Vnagt: Variational non-autoregressive graph transformer network for multi-agent trajectory prediction. *IEEE Transactions on Vehicular Technology*, 2023.

Chiara, L. F., Coscia, P., Das, S., Calderara, S., Cucchiara, R., and Ballan, L. Goal-driven self-attentive recurrent networks for trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2518–2527, 2022.

Du, W., Côté, D., and Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Systems with Applications*, 219:119619, 2023.

Fortuin, V., Baranchuk, D., Raetsch, G., and Mandt, S. GP-VAE: Deep probabilistic time series imputation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1651–1661. PMLR, 26–28 Aug 2020.

Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J. A., Kahou, S. E., Heide, F., and Pal, C. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Dup_dDqkZC5.

Gu, T., Chen, G., Li, J., Lin, C., Rao, Y., Zhou, J., and Lu, J. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17113–17122, 2022.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2255–2264, 2018.

Hu, Y., Chen, S., Zhang, Y., and Gu, X. Collaborative motion prediction via neural motion message passing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6319–6328, 2020.

Huang, Y., Bi, H., Li, Z., Mao, T., and Wang, Z. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6272–6281, 2019.

Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, H., and Savarese, S. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Lee, M., Sohn, S. S., Moon, S., Yoon, S., Kapadia, M., and Pavlovic, V. Muse-vae: multi-scale vae for environment-aware long term trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2221–2230, 2022.

Li, R., Li, C., Ren, D., Chen, G., Yuan, Y., and Wang, G. Bcdiff: Bidirectional consistent diffusion for instantaneous trajectory prediction. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Liu, Y., Yu, R., Zheng, S., Zhan, E., and Yue, Y. NAOMI: Non-autoregressive multiresolution sequence imputation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Maeda, T. and Ukita, N. Fast inference and update of probabilistic density estimation on trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9795–9805, 2023.

Mangalam, K., Girase, H., Agarwal, S., Lee, K.-H., Adeli, E., Malik, J., and Gaidon, A. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 759–776. Springer, 2020.

Mangalam, K., An, Y., Girase, H., and Malik, J. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15233–15242, 2021.

Mao, W., Xu, C., Zhu, Q., Chen, S., and Wang, Y. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5517–5526, 2023.

Mendieta, M. and Tabkhi, H. Carpe posterum: A convolutional approach for real-time pedestrian path prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2346–2354, 2021.

Mohamed, A., Qian, K., Elhoseiny, M., and Claudel, C. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14424–14432, 2020.

Mohamed, A., Zhu, D., Vu, W., Elhoseiny, M., and Claudel, C. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. In *European Conference on Computer Vision*, pp. 463–479. Springer, 2022.

Monti, A., Porrello, A., Calderara, S., Coscia, P., Ballan, L., and Cucchiara, R. How many observations are enough? knowledge distillation for trajectory forecasting. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6543–6552, 2022. doi: 10.1109/CVPR52688.2022.00644.

Qi, M., Qin, J., Wu, Y., and Yang, Y. Imitative non-autoregressive modeling for trajectory forecasting and imputation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12736–12745, 2020.

Ramchandran, S., Tikhonov, G., Kujanpää, K., Koskinen, M., and Lähdesmäki, H. Longitudinal variational autoencoder. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3898–3906. PMLR, 13–15 Apr 2021.

Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., and Savarese, S. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF, 2019.

Sekhon, J. and Fleming, C. Scan: A spatial context attentive network for joint multi-agent intent prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6119–6127, 2021.

Shan, S. and Oliva, J. B. NRTSI: Non-recurrent time series imputation. *ArXiv*, abs/2102.03340, 2021.

Shi, L., Wang, L., Long, C., Zhou, S., Zhou, M., Niu, Z., and Hua, G. Sgcn: Sparse graph convolution network for pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8994–9003, 2021.

Wang, M., Zhu, X., Yu, C., Li, W., Ma, Y., Jin, R., Ren, X., Ren, D., Wang, M., and Yang, W. Ganet: Goal area network for motion forecasting. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1609–1615. IEEE, 2023.

Xu, C., Li, M., Ni, Z., Zhang, Y., and Chen, S. Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6498–6507, June 2022a.

Xu, C., Mao, W., Zhang, W., and Chen, S. Remember intentions: Retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6488–6497, 2022b.

Xu, Y., Bazarjani, A., Chi, H.-g., Choi, C., and Fu, Y. Uncovering the missing pattern: Unified framework towards trajectory imputation and prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9632–9643, 2023.

Yoon, J., Zame, W. R., and van der Schaar, M. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2019. doi: 10.1109/TBME.2018.2874712.

Yu, C., Ma, X., Ren, J., Zhao, H., and Yi, S. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pp. 507–523. Springer, 2020.

Yuan, Y., Weng, X., Ou, Y., and Kitani, K. M. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9813–9823, 2021.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. Generating multi-agent trajectories using programmatic weak supervision. *arXiv preprint arXiv:1803.07612*, 2018.

A. Implementation Details

A.1. Imputation Module

We first pretrain the imputation model over the entire dataset for making a reasonable initial imputation. We use `n_steps=8`, the number of steps after which the features of the next pedestrian begin. We use `n_features=4`, the number of features of each pedestrian (x, y, v_x, v_y). We use Adam optimizer with a learning rate of 1×10^{-4} and weight decay of $\left(1 \times 10^{-5}\right)/2$ for this pretraining.

A.2. HyperGraph Convolution Neural Network

The initial node embeddings are fed as input to hypergraph convolution. The initial embeddings for a particular agent comprise a 6-dimensional vector resulting from the concatenation of coordinate features (2 dimensions) and scenic embeddings of dimension 4. Hence, the number of input channels of hypergraph convolution is 6. The output channel of the hypergraph convolution is 16 (which corresponds to the dimensionality of hidden features of a node). The output of the hypergraph convolution is $(batch, obs\_len, num\_of\_peds, out\_channels)$.

A.3. Scenic Attention Module

The model takes the position and scenic features, V_k, obtained using a pre-trained VGG-19 on ImageNet. It ensure that trajectory predictions consider not only pedestrian coordinates but also the contextual scene. Given that dataset images are captured by fixed cameras, we perform a one-time calculation for each dataset as follows:

$$V_k = VGG19\left(Image_{\text{scene}}\,; W_{VGG19}\right) \tag{24}$$

The calculation for producing V_k remains consistent regardless of historical time steps. It's worth noting that S_i^t is influenced by historical time frames, as it incorporates both V_k and X_i^t as inputs.

$$S_i^t = \text{SceneAttention}\left(V_k, X_i^t; W_{att}\right) \tag{25}$$

It calculates attention weights using scenic features and coordinates. It first processes the input positions and spatially embeds them. The VGG features are projected using a multi-layer perceptron (MLP). The resulting features are concatenated with the spatial embeddings and processed by another MLP to obtain attention weights. These weights are normalized using softmax, multiplied with the spatial embeddings to produce the final sequential scene attention embeddings (S_i^t), for individual agents at a specific time. These scenic embeddings of a particular node are of dimension 4.

A.4. Trajectory Refinement Module

Trajectory refinement model Θ_{tr} comprises of two parts:

- The initial phase consists of a combination of a Graph Convolution Network (GCN) and a Temporal Convolution Network (TCN). The entire trajectory is formed by concatenating both the observed and linearly predicted trajectories. A multi-relational graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, is employed. Here, $\mathcal{R}$ represents a set of relations, specifically $\mathcal{R} = \{\text{Distance}, \text{Displacement}, \frac{1}{\text{Distance}}, \frac{1}{\text{Displacement}}\}$. Furthermore, $\mathcal{V}$ encompasses the set of pedestrians, and $\mathcal{E}$ signifies the edges connecting these pedestrians. The scenic embeddings, generated through the Scenic Attention Module (refer A.3), are concatenated with the coordinates to produce the initial node embeddings (of a node at a particular time-stamp) with a dimensionality of 6. The initial node embeddings and the adjacency matrix of the multi-relational graph are passed into the GCN to get the final node embeddings of dimension 16. These final node embeddings are then passed into a Temporal Convolution Network (TCN), a CNN layer within channels = 16, out channel = 16, and kernel size = (3,1). The output of this model is the form of (batch, total_len, num_of_peds, out_channels).
- The second part contains two different CNNs. The first CNN is dedicated to time-wise convolution, with an input channel of 16 ($t_{total} = t_{ob} + t_{pred}$) and an output channel of 12 (t_{pred}) and kernel_size = 3. The second CNN is used for channel-wise convolution, where the input channel is set to 16, and the output channel is set to 2. The kernel size is set to 3. The output of this model gives the correction vector field. This is added to the linearly interpolated trajectory to get the final trajectory.

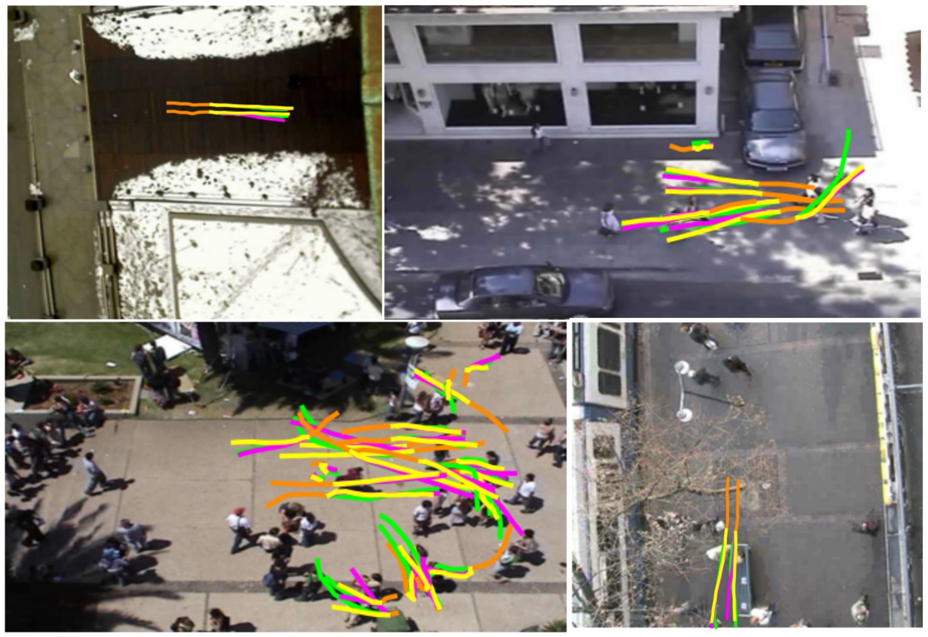


Figure 5. Visual comparison of predicted trajectories in images from ETH (top-left), UNIV (bottom-left), HOTEL (bottom-right), and ZARA (top-right) datasets. Predicted pedestrian trajectories from our approach are depicted in yellow, observed trajectories in orange, and ground truth trajectories in green. Graph-TERN predictions are represented in purple. Our approach accurately predicts future trajectories, closely aligning with the ground truth compared to Graph-TERN in most cases.

Calculation details for γ:

It is calculated by averaging the set of coordinates of each pedestrian for a given timestamp and then calculating the L^2 norm of the average X and Y coordinate thus obtained. This is normalized with the obs_seq_len by dividing the result with the same.

A.5. Overall Implementation detail

Following the standard evaluation strategy (Gupta et al., 2018), the observation frames, denoted as t_{ob}, consist of 8 frames, equivalent to 3.2 seconds, while the prediction frames are determined as $t_{pred} - t_{ob}$, amounting to 12 frames or 4.8 seconds. The number of control points K (refer 3.6) is set to 3. The number of endpoints sampled from the GMM, Ω is set to 20. We employ a batch size of 128 during the training process, with the number of training epochs set to 512. The learning rate for the entire model is specified as 1×10^{-4}. The optimiser used is SGD. We use Python 3.8.13 and PyTorch version 1.13.1+cu117.

Initially, we pretrain the imputation model on the complete dataset to establish a sensible initial imputation. The parameters used in pre-training are described in section A.1. We apply the pre-trained imputation model to fill in the randomly occurring NaNs in the input, as discussed in Section 3.2. Subsequently, we fine-tune the imputation model using a learning rate of 1×10^{-5}.

The imputed coordinates are fed as input to the hypergraph generation module (refer 3.3) to generate the incidence matrix $\mathbf{H}$ and hyperedge weights $\mathbf{w}$. The scenic embeddings, generated through the Scenic Attention Module (refer A.3), are concatenated with the coordinates to produce the initial node embeddings (of a node at a particular time-stamp) with a dimensionality of 6. The initial node embeddings are passed into the HGCNN module (refer A.2) to get the final node embeddings of dimension 16. These final node embeddings are then passed into Temporal Convolution Network (TCN), a CNN layer with in_channels = 16, out_channel = 16, and kernel_size = (3,1). The output of this model is the form of $(batch, obs\_len, num\_of\_peds, out\_channels)$.

To calculate the mean (μ_x, μ_y), standard deviation (σ_x, σ_y), and the mixing coefficient (π) for the Gaussian Mixture Model (GMM), we employ two distinct Convolutional Neural Networks (CNNs). The first CNN is dedicated to time-wise convolution, with an input channel of 8 (t_{ob}) and an output channel of 8 (representing the number of GMMs used, denoted as $\#_M$) and kernel_size = 3. The second CNN is used for channel-wise convolution, where the input channel is set to 16, and the output channel is calculated as $output_{features}(\mu_x, \mu_y, \sigma_x, \sigma_y, \pi) \times K$, where K represents the number of control points. In this case, it results in 5×3 equal to 15. The kernel size is set to 3.

The output features generated are used to generate a GMM, which is used for sampling $K = 3$ control points. These control points are added to get the final predicted end_point. We sample $\Omega = 20$ such endpoints. Next, we sample 20 additional guided end-points (refer 3.6).

The endpoint and the coordinates at time t_{ob} are combined in a linear manner to create a linearly interpolated trajectory. This linear trajectory is then passed to the trajectory refinement module (refer A.4) to get the correction vector. The correction vector is added to a linear interpolated trajectory to get the final trajectory. The losses $\mathcal{L}_r$ and $\mathcal{L}_c$ are calculated and back-propagated.

B. Additional Visualization

In ETH and HOTEL, our model effectively predicts pedestrian trajectories compared to Graph-TERN in most cases. Our multi-scale hypergraph GCN, considering social interactions, performs better in predicting trajectories for densely populated scenes in the UNIV dataset. Similarly, in ZARA, our model accurately predicts future movements, as depicted in Figure 5.