

Modeling Multimodal Dynamic Spatiotemporal Graphs

Boris Ivanovic

Marco Pavone

Abstract—Spatiotemporal graphs (STGs) are a powerful tool for modeling multi-agent interaction scenarios, used commonly in human trajectory prediction and proactive planning and decision making for safe human-robot interaction. However, many current STG-backed methods rely on a static graph assumption, i.e., that the underlying graphical structure maintains the same nodes and edges throughout the scenario. This assumption is frequently broken in real-world applications, especially in highly-dynamic problems such as human trajectory prediction in crowds. To remove the reliance on this assumption, we present a methodology for modeling and predicting agent behavior in both highly dynamic and multimodal scenarios (i.e., where the scene’s graphical structure is time-varying and there are many possible highly-distinct futures for each agent). Our approach to model dynamic STGs augments prior multimodal, multi-agent modeling methods with a gating function on edge models that smoothly adds and removes edge influence from a node. We demonstrate the performance of our approach on the ETH multi-human trajectory dataset and on NBA basketball player trajectories. Both are highly-dynamic, multimodal, and multi-agent interaction scenarios which serve as proxies for many robotic applications.

I. INTRODUCTION

Graphical structures are pervasive in real world multi-agent systems, where agents are frequently modeled as nodes and their interactions as edges. In particular, spatiotemporal graphs (STGs) are a popular graphical modeling tool as they naturally capture both spatial and temporal information. Like any graph, STGs also act as a general intermediate representation and provide an abstraction from domain-specific elements of problems, enabling developed methodologies to be deployed on a wide variety of applications. Further, they encourage model reuse as different parts of a graph may use the same underlying model, e.g., leading to super-linear parameter scaling [1]. As a result, there are many works across domains that demonstrate the benefits of introducing graphical structure to otherwise model-free approaches in areas such as physics simulation [2], [3], [4], robotic reinforcement learning [5], natural language processing [6], human trajectory forecasting [7], [1], and more [8].

One such problem domain is modeling human behavior for safe human-robot interaction (HRI), with specific motivating examples including a robot safely traversing through a dense crowd, negotiating traffic on a highway onramp, or packing items for delivery alongside humans. Modeling human behavior is a vital component of safe HRI systems as

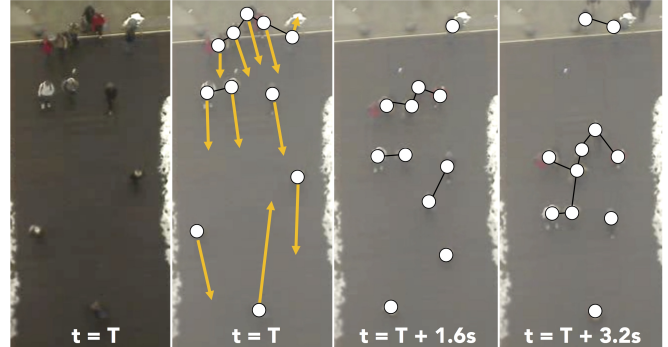


Fig. 1. A scene from the ETH multi-human trajectory dataset as it evolves over time. An undirected graph representation of the same scene is also visualized, illustrating how much a scene’s underlying graphical structure can vary through time. Nodes and edges are represented as white circles and solid black lines, respectively. Orange arrows depict the various agent velocities present in the scene, and accordingly show how much nodes in a graph shift through relatively small time periods. They are only shown once for clarity. This figure is best viewed in color.

it enables proactive planning, decision making, and trajectory generation [9]. While prior work has addressed one of the main challenges in human trajectory modeling (i.e., multimodality; where a human has many possible highly-distinct futures [1], [10], [11]), another core facet of human behavior that troubles graphical representations remains unaddressed: it is *dynamic*, i.e., time-varying. This may be an obvious statement, but it means that a scene’s graphical structure is also time-varying, with edges and nodes appearing and disappearing as agents move into and out of the scene, as well as between themselves. Many current graph modeling methods do not model scenes as dynamic graphs, instead making an implicit “static graph” assumption (i.e., that the nodes and edges on a scene’s graph are constant). This assumption holds in some applications, e.g., physical simulations of a d -link swimmer [12], but seldom in human trajectory modeling. An example of the time-varying nature of real world STGs is illustrated in Fig. 1.

Without considering the dynamic nature of real world STGs, autonomous systems can end up in undesired situations. For example, a self-driving car may be slowly surrounded by drivers and boxed into a lane far from where it needs to be. If the vehicle’s model accounted for this possibility, it could have changed lanes to escape.

Statement of Contribution: This paper presents a framework for modeling dynamic multimodal STGs, addressing the problem of modeling dynamic nodes and edges which was identified recently as an open question in graph network architectures [8]. Specifically, we present a method of augmenting prior multimodal STG models to incorporate time-

*This work was supported by the Toyota Research Institute (“TRI”). This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

Boris Ivanovic and Marco Pavone are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA {borisi, pavone}@stanford.edu

varying nodes and edges with minimal inference overhead. Our key technical contribution is the introduction of a modulating function on edge models that smoothly adds and removes edge influence from a node. Our approach inherently handles a variable number of nodes and edges, and does not add any assumptions about their longevity, outcome preference, or structure of interaction beliefs (e.g., as might be found in game theory). We demonstrate its performance on human trajectory modeling with two highly dynamic, multimodal datasets: the ETH multi-human tracking dataset and National Basketball Association (NBA) player trajectories.

Outline: In Section II, we review related literature and explain the reasoning guiding our model design decisions. Section III defines the notation we will use throughout this paper and rigorously formulates the problem we address. Section IV describes our approach to model multimodal dynamic STGs in detail. Quantitative and qualitative analyses follow in Section V. Finally, Section VI concludes the work and discusses future research avenues.

II. RELATED WORK

There are numerous prior works leveraging STG representations of problems. They can be roughly grouped into two classes of methods: one in which the graph represents some underlying mathematical truth in the problem, and the other in which the graph serves as a more general guide for algorithmic computations. Probabilistic Graphical Models, which encode conditional independencies in probabilistic formulations of the problem at hand, are emblematic of this first approach, while many recent Deep Learning Methods adopt the second.

A. Probabilistic Graphical Models

Probabilistic graphical models (PGMs) are a natural choice for modeling the evolution of STGs, and have successfully done so in prior work [13], [14]. Conditional Random Fields (CRFs; a specific form of PGM focused on directly modeling the conditional distribution of the output given the input), the most common form of PGM applied to such sequential modeling tasks, cannot handle dynamic edges in their original form [15], [16]. While a generalization of CRFs exists that enables it to be applied to dynamic-length sequences [17], the method essentially unrolls a fixed graph through time, applying it at each time step to observed data. While this is a form of modeling dynamic edges, it differs from our problem in that edges and nodes can change (appear and disappear) across both space and time, independently, rather than all at once across only time. The extension presented in [17] is an instance of a more general class of dynamic graphical models (DGMs). In general, DGMs involve fixed graphical structures unrolled through time [18], making them unsuitable for the problem at hand. Additionally, sampling from PGMs usually requires a sampling method such as Metropolis-Hastings [19] which is too slow for robotic use-cases.

B. Deep Learning Methods

Within deep learning methods for graph modeling, there is a delineation between models which explicitly mimic the input problem graph in their architecture (i.e., directly define the structure of a deep learning architecture) [20], [10], [1], [21], and methods which take a graph as input and provide n -step predictions as their output [4], [8], [2], [22].

Architectures Based on Graphs: The first class of methods generally represent agents as nodes and their interactions as agents, modeling both with deep sequence models such as Long Short-Term Memory Networks (LSTMs) [23], enabling the models to capture spatial relations through edge models and temporal relations through node models.

The Structural-RNN [21] explicitly formulates a CRF for STG modeling and then implements it as a graphical LSTM architecture over the agents considered, as described above. After defining the graph, it is unchanging throughout (as in a CRF). Similarly, the Social-LSTM [7] forms a similar graphical LSTM architecture, but combines neighboring agent information with a spatial pooling operation. This allows for the handling of dynamic edges, however the addition or removal of an edge would sharply affect future predictions since the value returned by pooling.

[1] presents a multi-agent modeling framework that explicitly considers the inherent multimodality in human behavior and combines a Conditional Variational Autoencoder (CVAE) with a similar graphical LSTM architecture as in the Structural-RNN. Notably, [1] attribute their nodes and edges with “types”, all instances of which share the same model weights. [10] is similar in model structure, but focuses on long-term predictions, different from [1] which chooses medium-length prediction horizons to bridge the gap between dynamics and planning (i.e., any shorter horizon and propagating dynamics should suffice for trajectory prediction, any longer horizon and planning methods are better for capturing global movement patterns). Both [10] and [1] do not consider dynamic edges or nodes. Finally, Vemula et al. compute a soft attention over neighboring nodes to take into account the most relevant edge information in the graph [20]. However, doing so requires maintaining a complete graph online in order to determine which edges are relevant, an $O(N^2)$ proposition which scales poorly with graph size.

Graphs as Inputs to Prediction Functions: Models in this class generally represent agents and their interactions in the same way, but usually assume a directed, attributed multi-graph instead of undirected graphs as in the previous group. Since these methods take in a graph G at each timestep and retain no global information (global information is instead retained with each graph instance G [8]), they are able to handle graphs which change in-between prediction steps. However, this is only an implicit ability to handle dynamic edges, and it is still unclear how to explicitly handle nodes and edges which dynamically appear and reappear over time [8]. Models in this class are also generally different instantiations of a “graph network” as defined in [8].

A graph network that generates next-step predictions of a graph representing a dynamic system (e.g., d -link swimmer

[12]) was presented in [4]. [2] is a similar graph network applied again to predicting the evolution of dynamic systems, but which also learns system identification. They even evaluate on real world data collected from a robot arm, however this is not multimodal data and the graph network framework in general has no multimodal modeling capabilities yet [8]. [22] presents a decoupled version of a graph network architecture, which takes in only observations of a scene known to be interacting and generates graph structures modeling the interaction. However, the implicit assumption in this work is that the observations are generated by a static graph throughout the scene (it is unknown if one could even reconstruct a dynamic graph from observations alone).

Overall, we decided to make our model part of the first class, closest to [1], as a result of their current ability to model multimodal data, stateful graph representation (leading to efficient iterative predictions online), and public code availability.

III. PROBLEM FORMULATION

Notation: We use superscripts, e.g., $x^{(t)}$, to denote the values of quantities at discrete time steps, and the notation $x^{(t_1:t_2)} = (x^{(t_1)}, x^{(t_1+1)}, \dots, x^{(t_2)})$ to denote a sequence of values at time steps in $[t_1, t_2]$ for $t_1 \leq t_2$.

A. Interaction Dynamics

Let the deterministic, time-invariant, discrete-time state space dynamics of N humans and a robotic agent in a scene be given by

$$x_{h_i}^{(t+1)} = f_{h_i}(x_{h_i}^{(t)}, u_{h_i}^{(t)}), \quad x_a^{(t+1)} = f_a(x_a^{(t)}, u_a^{(t)}) \quad (1)$$

respectively, where $x_{h_i}^{(t)} \in \mathbb{R}^{d_{x_h}}$, $x_a^{(t)} \in \mathbb{R}^{d_{x_a}}$ denote the states of human i and the robotic agent and $u_{h_i}^{(t)} \in \mathbb{R}^{d_{u_h}}$, $u_a^{(t)} \in \mathbb{R}^{d_{u_a}}$ denote their chosen control actions at time $t \in \mathbb{N}_{\geq 0}$. Let $x^{(t)} = (x_{h_1}^{(t)}, \dots, x_{h_N}^{(t)}, x_a^{(t)})$ and $u^{(t)} = (u_{h_1}^{(t)}, \dots, u_{h_N}^{(t)}, u_a^{(t)})$ denote the joint state and control of the $N + 1$ entities. Note that the inclusion of the robotic agent in this formulation is optional. We chose to explicitly include it in our problem formulation as it enables easier later integration with robotic planning and decision making frameworks for safe HRI, e.g., in the problem of navigating a robot through a dense crowd, one may evaluate human reactions to candidate future robot actions with this formulation [1], [11]. One would omit this and related terms for a pure multi-agent modeling framework.

A scenario ends when the joint state first reaches a terminal set $\mathcal{T} \subset \mathbb{R}^{N d_{x_h} + d_{x_a}}$ and let T denote the final time step, $x^{(T)} \in \mathcal{T}$. For each $t < T$, we assume that all humans' next actions $u_{h_1, \dots, h_N}^{(t+1)}$ are drawn from a distribution conditioned on the joint interaction history $(x^{(0:t)}, u^{(0:t)})$ and the agent's next action $u_a^{(t+1)}$. Thus, $U_{h_1, \dots, h_N}^{(t+1)} \sim P(x^{(0:t)}, u^{(0:t)}, u_a^{(t+1)})$, with drawn values denoted in lowercase. We further suppose that $U_{h_1, \dots, h_N}^{(t+1)}$ is distributed according to a probability density function (pdf) which we

denote as $p(u_{h_1, \dots, h_N}^{(t+1)} | x^{(0:t)}, u^{(0:t)}, u_a^{(t+1)})$. Finally, we also assume that all past states and actions by all entities are fully observable. This means that an agent may reason about $U_{h_1, \dots, h_N}^{(t+1:t+S)}$, representing all humans' response sequences to agent actions $u_a^{(t+1:t+S)}$ over a horizon of length S , by propagating (1) and sampling $U_{h_1, \dots, h_N}^{(t+1)}$. Our goal in this paper is to learn $p(u_{h_1, \dots, h_N}^{(t+1)} | x^{(0:t)}, u^{(0:t)}, u_a^{(t+1)})$, which acts as a model of human behavior that takes into account neighboring human behavior as well as candidate future robot actions.

B. Specific Scenarios

Although we have developed our approach generally for multiple human-agent and human-human interactions, we chose basketball playing [24] and pedestrian motion [25] as the scenarios to demonstrate our model's performance as they are both challenging multimodal multi-agent modeling scenarios with time-varying scene graphs.

For both of these scenarios, let (x, y) be the coordinate system defining agent positions in a scene. In basketball, this represents the longitudinal and lateral distance along the length and width of the court, respectively. For pedestrian walking, it is the position along a defined coordinate axes in a scene, usually dependent on the specific data collection method, a mounted camera facing a street in the case of the ETH dataset [25]. We consider the center of mass of human i to obey single-integrator dynamics (i.e., $u_{h_i} = [\dot{x}_{h_i}, \dot{y}_{h_i}]$). This is an intuitive choice as a human's movements are all position-changing, e.g., walking increases one's position along a direction, running does so faster. We enforce an upper bound of 12.42m/s on any human's speed, which is the current footspeed world record set by Usain Bolt [26].

IV. DYNAMIC MULTIMODAL MULTI-AGENT MODELING

We propose a solution that combines prior multimodal, multi-agent modeling methods with an output modulating function that smoothly accounts for edge addition and removal. The overall model produces high-quality multimodal trajectories that are predictions of the future behaviors of N humans, taking into account the time-varying structure of the underlying graph. Our full architecture is illustrated in Fig. 2.

Considering a fixed prediction time step t , let $\mathbf{x} = (x^{(0:t)}, u^{(0:t)}, u_a^{(t+1:t+S)})$ be the input to the model (neighboring human interaction history and candidate agent future) and $\mathbf{y} = u_{h_1, \dots, h_N}^{(t+1:t+S)}$ be the desired prediction output (all humans' futures). We wish to learn a pdf $p(\mathbf{y}|\mathbf{x})$. To do this, we leverage the CVAE framework and introduce a latent variable \mathbf{z} so that $p(\mathbf{y}|\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{y}|\mathbf{x}, \mathbf{z}) p(\mathbf{z}|\mathbf{x}) d\mathbf{z}$. \mathbf{z} 's purpose is to model latent structure in the interaction which both improves learning performance and enables interpretation of results [27], [11], [1]. In our work, $p(\mathbf{y}|\mathbf{x}, \mathbf{z})$ and $p(\mathbf{z}|\mathbf{x})$ are modeled using neural networks that are fit to maximize the likelihood of a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_i$ of observed interactions. This optimization is performed by maximizing the evidence-based lower bound (ELBO) of the log-likelihood $\log p(\mathbf{y}|\mathbf{x})$ by

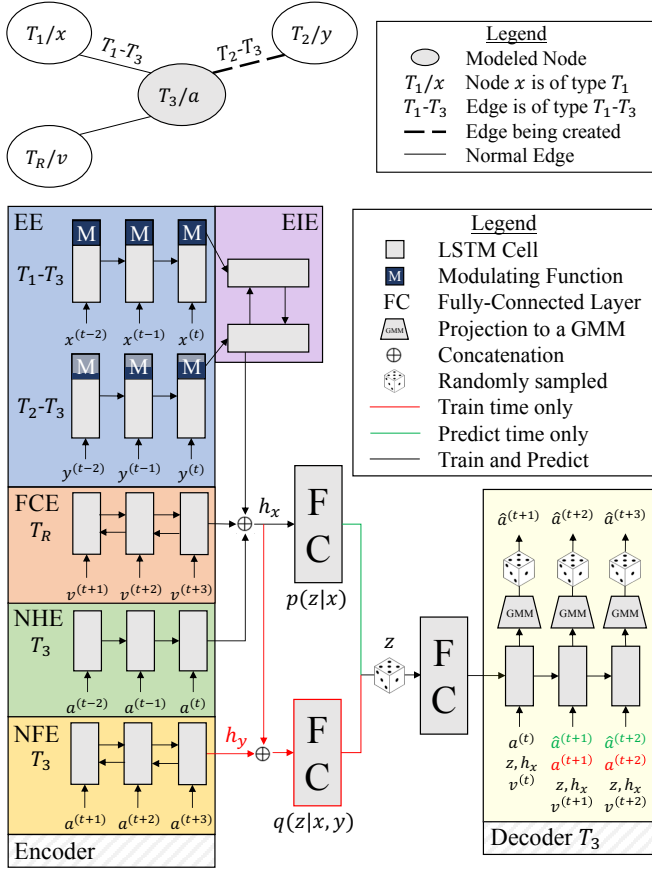


Fig. 2. **Top:** An example graph with four nodes. a is our modeled node and is of type T_3 . It has three neighbors: y of type T_2 ; x of type T_1 ; and the agent v . Further, y is about to connect with a . **Bottom:** Our corresponding architecture for node a . The EE and EIE (“Edge Encoder”) and “Edge Influence Encoder”) are responsible for encoding the effects neighbors have on the modeled node. On top of the EEs are our modulating functions M , which modulate the outputs of the EEs through scalar multiplication. Here, the outputs of T_2-T_3 are being scaled by a value between 0 and 1, indicated by how full of navy the modulation block is. The FCE (“Future Conditional Encoder”) models the high-level semantics of a candidate agent future. The NHE (“Node History Encoder”) represents the modeled node’s past. The NFE (“Node Future Encoder”) is only used during training and helps shape our latent representation. The Decoder samples a latent state z and generates trajectories for the modeled node. Inputs to the model are concatenated states (x) and actions (u). This figure is best viewed in color.

importance sampling z from a neural network approximation $q(z|x, y)$ of the true posterior $p(z|x, y)$ [28].

To model nodes and edges, we use LSTM networks. LSTMs are a type of recurrent neural network where each cell contains internal gating functions that enable a finer control of information flow [23]. Each cell’s internal computations are as in [29].

We proceed by describing our model and highlighting key components that enable it to model dynamic STGs.

Graphical Representation: When presented with an input problem, we first automatically create an undirected graph $G = (V, E)$ representing the scene (as in Fig. 1). In this work, we form edges based on entities’ spatial proximity, as in prior work [1], [7].

Encoding Dynamic Influence from Neighboring Humans: We use Edge Encoders (EEs) to incorporate influence from neighboring nodes. They are LSTMs with 8 hidden dimen-

sions. Note that we focus on edge *types* rather than individual edges. This allows for more efficient scaling and dataset efficiency as we reuse edge encoder weights across edges of the same type. We combine the states of all neighboring nodes of a specific edge type by summing them and feeding the result into the appropriate edge encoder, obtaining an edge influence representation. These representations are then passed through “modulation blocks”, i.e., scalar multiplications that modulate the outputs of EEs depending on the age of an edge. The actual values they are multiplied by are stored in a 4D edge modulation tensor M with shape $B \times T \times N \times N$ where $M[b, t, n_1, n_2]$ is the edge modulation factor between nodes n_1 and n_2 at time t within data batch b . This enables minimal training overhead as it reduces dynamic edge inclusion to a 4-tuple lookup and element-wise multiplication operation: $\hat{h}_t = h_t \odot M[b, t, n_1, n_2]$.

For training, we precompute all values in M by convolving specific filters (for the addition A and removal R of an edge) with a dataset-specific “edge mask” E (essentially a 3D binary adjacency matrix with time as the depth dimension), i.e., $M = \min\{A * E + R * E, 1\}$ where \min is applied element-wise. Computing M during test time is a simpler process as only one $N \times N$ slice needs to be computed per timestep. This is done by incrementing counters on the age of edges (essentially checking the adjacency matrix of the previous step) and computing the necessary edge modulation factor ($A(e_t)$ if edge e was created recently and $R(e_t)$ if e was removed recently).

As an example, if we wished to encourage gentle edge addition (e.g., over 5 timesteps) and sharp edge removal (e.g., over 1 timestep), we could define our filters as $A = 0.2e_t \forall 0 \leq t \leq 5$ and $R = 1 - e_t \forall 0 \leq t \leq 1$ where e_t is the age of edge e at global timestep t . The only condition we impose on A and R is that they start at 0 and end at 1. An example of why one might prefer smooth edge additions is that it rejects high-frequency switching, e.g., if an agent is constantly dithering at the edge radius it would not cause predictions to fluctuate sharply over time.

This modulated representation is then merged with other edge influences via an Edge Influence Encoder (EIE) to obtain a total edge influence encoding. The EIE is a bi-directional LSTM with 8 hidden dimensions and standard sigmoid/tanh nonlinearities. We feed encoded edges as inputs to the EIE and output the final concatenated forward and backward hidden and memory vectors. We choose to combine representations in this manner rather than via concatenation in order to handle a variable number of neighboring nodes with a fixed architecture while preserving information [4], [21], [1].

Other Sequence Models: We use a Node History Encoder (NHE) to represent a node’s state history. It is an LSTM network with 32 hidden dimensions. We use a Future Conditional Encoder (FCE) to represent a neighboring future-conditional agent’s effect on the modeled node. It is a bi-directional LSTM with 32 hidden dimensions and standard sigmoid/tanh nonlinearities. For our decoder, we use an LSTM with 128 hidden dimensions whose outputs are Gaus-

sian Mixture Model (GMM) parameters with $N_{GMM} = 16$ components (chosen according to [1]). These GMMs are over the human action space, from which we sample to output trajectories.

Additional Considerations: To handle multiple dynamic edges of the same type, we similarly sum the modulation functions of the edges and apply the same element-wise minimization such that the resulting combined modulation function is properly scaled and takes into account the proximity of approaching or receding nodes. Specifically the resulting modulation function is $N[b, t, n_i, k] = \min \left\{ \sum_{n_j \in N_k(n_i)} M[b, t, n_i, n_j], 1 \right\}$ where \min is applied element-wise, k is a specific edge type that n_i possesses, and $N_k(n_i)$ is the set of neighbors of n_i connected by an edge of type k . The resulting 4D tensor N has shape $B \times T \times N \times N_{ET}$ where N_{ET} is the number of unique edge types in the graph. $N[b, t, n_i, k]$ represents the combined modulation function for edge type k affecting node n_i at time t in batch b .

Depending on the graph, E , M , and N may be dense or sparse. We make no assumptions about adjacency structure in this work, however, this is a point where additional structure may be infused to make computation more efficient for a specific use-case.







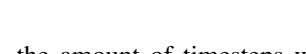
V. EXPERIMENTS

All of our source code, trained models, and data can be found at <https://github.com/StanfordASL/DynSTGModeling>. Training and experimentation was performed on a single Amazon Web Services p2.xlarge EC2 instance with TensorFlow [30]. All of the following results were obtained after only 290 training steps on their respective datasets. These values are very small when compared to traditional deep learning methods due to the model’s aggressive weight sharing scheme. Throughout this section, we use a model’s negative log-likelihood (NLL) prediction loss on the evaluation set as our test metric, meaning lower values are better.

NBA Dataset: The NBA dataset is a collection of 97 “plays” recorded from 26 distinct professional basketball players during the 2015-16 NBA season [24]. Position data was captured every 40ms, accurately tracking each of the 10 players during a game. This equates to $T \approx 600$ assuming a 24s play duration, which is the length of the shot clock during a single possession. We define a “play” as any contiguous amount of time-reducing game play (i.e., we ignore breaks, free throws, fouls, etc.). Taking a prediction horizon of $S = 15$ (600ms into the future), the total dataset contains approximately 51,000 $\mathbf{x} = (x^{(0:t)}, u^{(0:t)}, u_a^{(t+1:t+S)})$, $\mathbf{y} = u_h^{(t+1:t+S)}$ data points. We chose this time horizon because a professional basketball player can cover distances of ~ 4.5 m in 600ms [31]. We set aside 10 plays from an NBA game not present in the training dataset to form our validation set with which we evaluate and visualize our model’s performance.

For the NBA dataset, we constructed a model with a 2m edge radius (a good tradeoff between number of parameters, memory usage, compute time and model performance [1]) and evaluated the performance of edge modulation lengths

TABLE I
MODEL PERFORMANCE VS. EDGE MODULATION SCHEMES

Edge Modulation Scheme	NBA NLL	ETH NLL
None, as in [21], [2], [1], and others	−21.94	−41.50
	−26.51	−46.36
	−26.18	−47.99
	−26.10	−43.58
	−26.72	−47.23
	−27.72	−48.04
	−25.95	−45.86
	−27.86	−45.53

(i.e., the amount of timesteps when A and R are nonzero) of 1-10 timesteps (0.04-0.4s). We chose these modulation lengths as human reaction time averages 0.2s [32].

ETH Dataset: The ETH dataset is a pair of videos of busy locations recorded from a birds-eye view at 15Hz and annotated with the positions of the pedestrians that traverse through the scene [25]. Due to the relative sparsity of annotation data in the ETH dataset (annotations are at 2.5Hz), we fit¹ a B-spline to the provided points and created an up-sampled version of the annotations such that there was a point at every frame id. We trained on the first 12k and 17k timesteps of the up-sampled “Hotel” and “ETH” sequences, respectively, leaving the rest for evaluation.

For the ETH dataset, we constructed a model with a 1.5m edge radius (following commonly-used sidewalk width guidelines [33], [34]) and edge modulation lengths of 1-7 timesteps (0.07-0.47s). We chose these modulation lengths as before, keeping in mind human reaction time [32].

We don’t specifically model obstacles in this work, however one could incorporate obstacles by introducing a stationary node of an obstacle type e.g., “Obstacle” or “Tree”, as in prior methods [25].

A. Comparing Different Edge Modulation Schemes

We trained a model for each of the evaluated edge modulation schemes to compare the performance of models with and without edge modulation schemes, as well as ones with symmetric and asymmetric schemes. They are each visualized in Table I, where $A(e_t)$ and $R(e_t)$ are shown on the left and right side of the first column, respectively.

We first evaluated the performance of prior methods which rely on a static graph assumption. These form the baseline that we compare to.

We also evaluated the naïve dynamic edge addition strategy of simply factoring in edges as soon as they appear, which performed better as it is able to take into account the actual graph structure. However, this scheme is undesirable in practice as it creates a sudden change in predictions (an example is shown in Table II), which could cause sudden

¹We used the `scipy.interpolate` sub-package in Python.

jerk in robot behavior, a potentially unsafe outcome. This edge scheme is employed in existing methods like [7] which are able to handle dynamic edges but have them sharply affect the model. Further, this edge scheme on the ETH dataset is comparable to a GMM and CVAE-augmented form of [4], [8], [2] since all nodes and edges are of the same type and thus use the same node and edge model weights, respectively.

We then evaluated different smooth modulation functions, with ramp-up and down periods pictured in Table I. The ramp-up and down period is identical to the figures for the NBA dataset, i.e., we evaluated periods of length 2, 5, and 10 timesteps. For the ETH dataset, we evaluated ramp-up and downs of length 1, 4, and 7. The smaller periods are due to a smaller framerate in the ETH dataset.

NBA Results: We first compared symmetric and asymmetric ramp functions to determine if players are more affected by smooth edge addition, removal, or benefit from both (the second section of Table I). We obtained the best performance with a symmetric edge modulation function, which makes sense as it implies that basketball players are equally affected by other players regardless of the direction of interaction (i.e., approaching and moving away). This emphasizes that professional basketball is a highly-dynamic environment, as even after an opponent or teammate has left the immediate vicinity of a player, they are still kept in mind by the player (e.g., in an attempt to coordinate a pass, block, shooting opportunity, etc). This intuition is further reinforced by the results from the last section of Table I. The reason why a symmetric exponential modulation function performs the best can be explained by basketball being a reactive as much as proactive game, and an exponential edge modulation scheme implies having a slight knowledge of an impending player to come nearby or leave (usually called “game sense”, the modeled player is keeping track of players around him, but not immediately surrounding him) and as they approach within the radius the modeled player is reacting to it.

ETH Results: As opposed to the NBA dataset, our model performs better with an asymmetric linear ramp modulating function in the second section of Table I. This makes sense as pedestrians focus very little on others after they have passed by, naturally assuming their velocity and final destination are very different from their own. This also explains why the long symmetric linear ramp modulation scheme performs the best, it is likely taking advantage of the long linear edge addition modulating function rather than the long edge removal function. The choice for a linear ramp function is sensible as pedestrians gradually incorporate observations of others and change their trajectories accordingly. Further, the fact that the sigmoid and exponential functions perform worse indicates there is no preference for a node being affected suddenly when close or finalizing planning when they are farther away.

B. Observing Dynamic Edge Behavior

Table II shows predictions from our model and prior approaches on specific dynamic examples from both datasets.

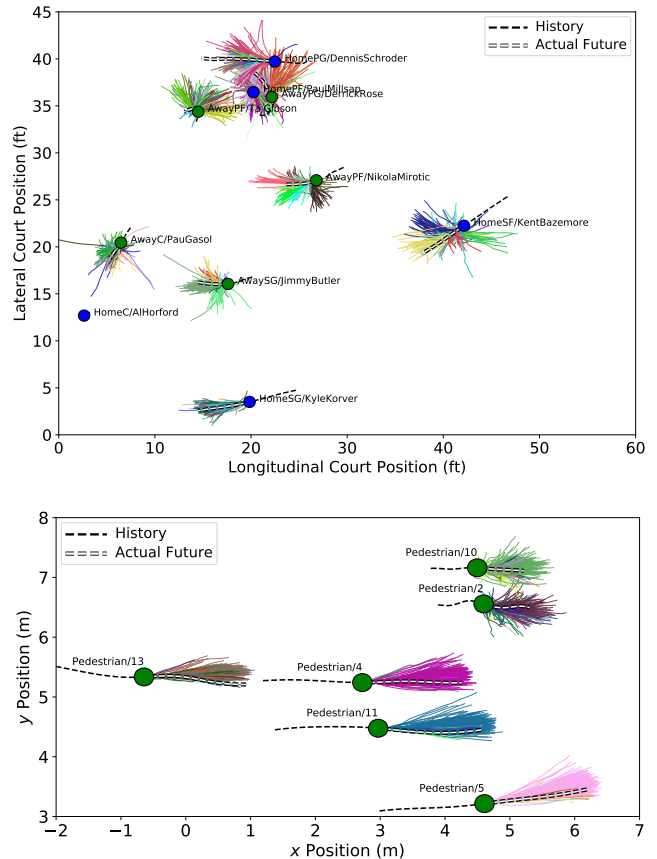


Fig. 3. **Top:** A full output from our model of a scene from the NBA dataset. **Bottom:** A full output from our model of a scene from the ETH dataset. In both cases, 200 future predictions are sampled and visualized for each node. Colored lines are sampled predictions where color indicates the value of the latent variable z . A node’s name and type are adjacent to the node. This figure is best viewed in color.

The first two rows show a situation from the ETH dataset where two pedestrians pass by each other. We can see that our model’s predictions smoothly change when the agents begin interacting as opposed to the static model whose predictions completely do not align with the actual future trajectory of the top right pedestrian. The bottom three rows show predictions from our model, those from a static graph, and one which takes into account dynamic edges but without any smooth modulation function (i.e., a “hard switch”). As can be seen, the static method’s predictions do not change as a result of the interaction. The hard switch model’s predictions account for the new player, but by suddenly predicting many trajectories in its direction. Our model smoothly shifts an existing mode’s trajectories towards the new agent, incorporating new information smoothly and exhibiting the desired gentle behavior.

VI. DISCUSSION AND CONCLUSION

In conclusion, this work presents a new method for modeling dynamic spatiotemporal graphs with minimal inference overhead. Specifically, we present a method of augmenting prior multimodal STG models to incorporate time-varying

TABLE II
MODEL PREDICTIONS VS. TIME IN DYNAMIC SITUATIONS WITHIN THE NBA AND ETH DATASETS

Model and Dataset	Before the Interaction	During the Interaction	During the Interaction	After the Interaction
Static [2] - ETH				
Our Model - ETH				
Static [1] - NBA				
Hard Switch - NBA				
Our Model - NBA				

nodes and edges with the introduction of a modulating function on edge models that smoothly adds and removes edge influence from a node. We have demonstrated its performance on two real world, highly-dynamic, multimodal trajectory modeling datasets.

An exciting future direction includes implementing this work in a deep learning framework based on dynamic computation graphs, such as PyTorch [35], and adding modulated model components as they appear in the graph in order to enable the method to handle streaming data.

REFERENCES

- [1] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, “Generative modeling of multimodal multi-human behavior,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2018, in Press.
- [2] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, “Graph networks as learnable physics engines for inference and control,” in *Int. Conf. on Machine Learning*, 2018, pp. 4470–4479.
- [3] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, “A compositional object-based approach to learning physical dynamics,” in *Int. Conf. on Learning Representations*, 2017.
- [4] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, “Interaction networks for learning about objects, relations and physics,” in *Conf. on Neural Information Processing Systems*, 2016.
- [5] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone, (2018) BaRC: backward reachability curriculum for robotic reinforcement learning. Available at <https://arxiv.org/abs/1806.06161>.
- [6] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proc. of Conf. on Empirical Methods in Natural Language Processing*, 2013.
- [7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human trajectory prediction in crowded spaces,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.
- [8] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, (2018) Relational inductive biases, deep learning, and graph networks. Available at <https://arxiv.org/abs/1806.01261>.
- [9] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, “Probabilistically safe robot planning with confidence-based human predictions,” in *Robotics: Science and Systems*, 2018.
- [10] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, “DESIRE: distant future prediction in dynamic scenes with interacting agents,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.
- [11] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, “Multimodal probabilistic model-based planning for human-robot interaction,” in *Proc. IEEE Conf. on Robotics and Automation*, 2018.
- [12] Y. Tassa, T. Erez, and B. Smart, “Receding horizon differential dynamic programming,” in *Conf. on Neural Information Processing Systems*, 2007.
- [13] M. Kolar, L. Song, A. Ahmed, and E. P. Xing, “Estimating time-varying networks,” *Annals of Applied Statistics*, vol. 4, no. 1, pp. 94–123, 2010.
- [14] D. F. Fouhey and C. L. Zitnick, “Predicting object dynamics in scenes,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.
- [15] S. Nowozin and C. H. Lampert, “Structured learning and prediction in computer vision,” *Foundations and Trends in Computer Graphics and Vision*, vol. 6, no. 3–4, pp. 185–365, 2011.
- [16] C. Sutton and A. McCallum, “An introduction to conditional random fields,” *Foundations and Trends in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.
- [17] C. Sutton, A. McCallum, and K. Rohanimanesh, “Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data,” *Journal of Machine Learning Research*, vol. 8, pp. 693–723, 2007.
- [18] J. Bilmes, “Dynamic graphical models,” *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 29–42, 2010.
- [19] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.
- [20] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling

- attention in human crowds,” in *Proc. IEEE Conf. on Robotics and Automation*, 2018.
- [21] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, “Structural-RNN: Deep learning on spatio-temporal graphs,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.
 - [22] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” in *Int. Conf. on Machine Learning*, 2018, pp. 2688–2697.
 - [23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, 1997.
 - [24] K. Linou. (2016) NBA-Player-Movements. Online. Available at <https://github.com/linouk23/NBA-Player-Movements>.
 - [25] S. Pellegrini, A. Ess, K. Schindler, and L. v. Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in *IEEE Int. Conf. on Computer Vision*, 2009.
 - [26] R. Graubner, R. Buckwitz, M. Landmann, and A. Starke. (2009) Biomechanical analysis. Online. International Association of Athletics Federations. Available at https://web.archive.org/web/20140514050117/http://berlin.iaaf.org:80/mm/document/competitions/competition/05/30/83/20090817081546_httppostedfile_wch09_m100_final_13529.pdf.
 - [27] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” in *Conf. on Neural Information Processing Systems*, 2015.
 - [28] C. Doersch. (2016) Tutorial on variational autoencoders. Available at <https://arxiv.org/abs/1606.05908>.
 - [29] N. Kalchbrenner, I. Danihelka, and A. Graves. (2015) Grid long short-term memory. Available at <https://arxiv.org/abs/1507.01526>.
 - [30] M. Abadi *et al.* (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from [tensorflow.org](https://www.tensorflow.org/). [Online]. Available: <https://www.tensorflow.org/>
 - [31] (2017) Draft combine strength and agility. National Basketball Association. Available at <http://stats.nba.com/draft/combine-strength-agility>.
 - [32] R. J. Kosinski. A literature review on reaction time. Clemson University. Available at <http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/biae.clemson.edu/bpc/bp/Lab/110/reaction.htm>.
 - [33] Walkways, sidewalks, and public spaces. Online. U.S. Federal Highway Administration. Available at https://safety.fhwa.dot.gov/ped_bike/univcourse/pdf/swless13.pdf.
 - [34] Pedestrian characteristics. Federal Highway Administration University Course on Bicycle and Pedestrian Transportation. U.S. Federal Highway Administration. Available at <https://www.fhwa.dot.gov/publications/research/safety/pedbike/05085/pdf/lesson8lo.pdf>.
 - [35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017.