Human Trajectory Prediction Using Similarity-Based Multi-Model Fusion

Golnaz Habibi and Jonathan P. How, Fellow, IEEE

Abstract—Understanding pedestrian behaviors is crucial for a safe navigation of self-driving vehicles. However, pedestrians exhibit a large variety in their motion behaviors that are affected by interaction with the environment and other users of the road/sidewalk. Most of current models are limited to batch (offline) settings which requires to learn from the entire datasets. This letter presents a similarity-based model fusion algorithm, called SimFuse, for improving the prediction accuracy which enables autonomous agents to incrementally update their knowledge by communicating with other vehicles (V2V) by infrastructures (V2I). In the proposed framework, knowledge is shared in a compact form of “learned model” instead of raw data which provides a scalable sharing paradigm between agents. This work extends our prior work SILA [1] by providing multi model fusion of \( n \geq 2 \) models at the same time. We evaluate our algorithm in both intersection and non intersection scenarios and compare it with other baselines. The results show our algorithm outperforms state of the art in terms of Average Displacement Error at intersection scenarios and it has comparable result for non intersection scenarios with 3% improvement over SILA in ADE. The results also show that SimFuse updating time is up to 12 times faster than SILA with similar performance.

Index Terms—Distributed Robot Systems, Human and Humanoid Motion Analysis and Synthesis, Computer Vision for Transportation.

I. INTRODUCTION

WITH advances in Connected Vehicle (CV) technologies, autonomous vehicles get access to more information, have better understanding of the environment, and therefore act more safely and effectively. For instance, the data collected from a fleet of vehicles helps the autonomous driver understand the traffic flow and congestion, hazardous location, or learn different motion behavior which leads to higher safety and mobility in intelligent transportation system. Despite leveraging V2V communication for tracking task [2] or distributed planning [3] in intelligent transportation system, there is no work known that has ever used shared information for updating the learned models and eventually improve prediction accuracy. Since most of the learning model of motion behaviors are limited to batch setting, they cannot be updated incrementally when more data/information is available from on board sensors/other agents.

Previously, we have presented SILA [1] with the ability of incremental learning of pedestrian motion behavior. However, it was limited to fuse only two models at a time. This limits the possibility for V2X application, where an agent – either an autonomous vehicle (V2V) or an infrastructure (V2I) – wants to update its knowledge by the fusion of multiple models shared at the same time. This letter extends SILA and proposes a Similarity-based Multi-Model Fusion (SimFuse) algorithm such that each agent is able to incrementally update its knowledge by communicating with other agents and fuses multiple models. In SimFuse framework, the human motion model is presented in terms of motion primitives and their transitions (see Section III). To fuse multiple models, their motion primitives are compared and similarly (matching) graph is created. Then edges with lower similarity are successively relaxed until a consistency between matched motion primitives is achieved. To update the model, matched nodes and their transitions in the consistent graph are finally fused.

List of Contributions: (1) a new consistency algorithm suitable for matching motion primitives is proposed; (2) a new scalable Gaussian Process (GP) fusion algorithm based on sparse GP is proposed, which can be applied to any problem that requires fusion of multiple GPs; (3) Conducted several experiments including intersection scenarios and non intersection scenarios to evaluate SimFuse. The results confirm SimFuse outperforms the state-of-the-art algorithm in terms of average displacement error (ADE) with 30% improvement in intersection scenarios, and it achieves a comparable performance for non-intersection scenarios with the best ADE in average by 3% improvement over SILA. (4) SimFuse can update the knowledge by fusing multiple models much faster than SILA. The results show SimFuse updating rate is up to 12 times faster than SILA for the fusion of up to \( n = 35 \) models, while both achieve similar prediction accuracy.

II. RELATED WORK

Pedestrian Motion Prediction: prediction of the pedestrian motion has been studied widely in robotics and autonomous driving communities [6]. Most recent work incorporate interaction between the pedestrians in their model to improve the accuracy [7]–[10]. However, they are typically based on batch learning, which requires each vehicle access to the entire dataset and process all the data at a time. This assumption may not

Manuscript received July 1, 2020; accepted December 6, 2020. Date of publication January 1, 2021; date of current version January 22, 2021. This letter was recommended for publication by Associate Editor Achim J. Lilienthal and Editor T. Asfour upon evaluation of the reviewers’ comments. This work was supported by Ford Motor Company. (Corresponding author: Golnaz Habibi.)

The authors are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: golnaz.habibi@gmail.com; jhow@mit.edu).

Digital Object Identifier 10.1109/LRA.2020.3048652

Authorized licensed use limited to: University of Oklahoma Libraries. Downloaded on January 18,2023 at 20:53:59 UTC from IEEE Xplore. Restrictions apply.
be feasible for vehicles with limited capacity and it limits the agent to use pre-trained model, without flexibility to update the model when the new data/information is available. Previously, we presented SILA [1], where each agent can incrementally update its model when more information is available from its own experience. However, SILA is limited to incrementally learn from one set of datasets. This letter presents SimFuse which provides an additional property for a group of agents where they can fuse any number of models.

**Distributed Machine learning and V2V communication:** although distributed learning approaches has been studied previously [11], most of the prior work in autonomous driving are either limited to multi-agent reinforcement learning [12]–[14] or solve different problems other than prediction of human/vehicle trajectories. For instance, Shorinwa et al. [2] proposed an algorithm for tracking of multiple vehicles by a fleet of autonomous vehicles. Their work focused on solving multi-tracking problem, for short-term prediction when the target car is partially observable by each tracker vehicle. This letter instead, focuses on improving the prediction model of pedestrian trajectory in each agent by fusion of models it receives from other agent. The vehicle-to-vehicle (V2V) communication has also been considered in prior work [15], [16], which shows advantages in different aspects of traffic surveillance and management, route optimization, prevention of collision, sharing hazards and accidents [17]. The work in [18] addresses collaborative perception in autonomous vehicles, where the sensor measurements are shared between vehicles to increase the automated driving systems confidence in detecting objects using a 3D sensor fusion algorithm. FusionEye [19] provided a connected vehicle system that enables vehicles to share their data, which is then merged into a more complete traffic scene. We previously presented SILA [1] as an incremental learning framework where an agent incrementally updates its pre-trained model when and where a new batch of data is available. SILA can be used as an efficient and scalable framework for multi-agent setting. In this framework, vehicles share their knowledge in the abstract form of “learned motion behaviors” instead of raw data. However, SILA was limited to fuse only two models. This letter presents SimFuse which is able to fuse any number of models. Additionally, the proposed framework provides a real-time distributed learning process which is scalable to the large number of agents. Recently, Kucner et al. [20] presented a framework where a large map of motion patterns is stored in one model. In our approach, we do not store all the motion patterns. Instead, we leverage the similarity between motion primitives and fuse the similar motion patterns. The benefit of fusion of motion primitives over storing all motion primitives is the ability to continually improve the knowledge in a scalable format without need for learning from the entire dataset, while the model complexity maintains reasonably small.

**III. BACKGROUND AND PROBLEM STATEMENT**

Assume there are $n$ Agents, denoted by $A_i$, $i = 1, \ldots, n$. Each agent gets access or collects a mini-batch of $k_i$ training trajectories, represented in grid with $P \times Q$ blocks and width of $\omega$ each.

Each trajectory $t^{iz} \in \mathbb{R}^{3PQ}$, $z = 1, \ldots, k_i$, and $i = 1, \ldots, n$ has three components of: x-y velocities ($v^{ix} \in \mathbb{R}^{PQ}$, $v^{iy} \in \mathbb{R}^{PQ}$), and activeness variable denoted by $a^{iz} \in \mathbb{R}^{PQ}$. If $t^{iz}$ goes into a cell $pq$, $a^{iz}_{pq} = 1$, otherwise $a^{iz}_{pq} = 0$ (see Fig. 1(a), the shaded cells have nonzero values). Using this terminology, the matrix of data $X^{3PQ\times k_i}$ is built from $k_i$ trajectories, i.e., each column is a trajectory vector.

**A. Learning Motion Primitives**

Given a dataset $X$, a set of $K^i_d$ motion primitives denoted by $D^{(i)}$, are learned using a sparse coding method [4]. This dictionary learning method is a multi-variable optimization problem which tries to present the data with minimum number of motion primitives while the residual error is minimized simultaneously. The output is a set of $K^i_d$ motion primitives in form of matrix $D^{(i)}$, each column is a motion primitive vector, denoted by $m^i_k \in \mathbb{R}^{3PQ}$, $k = 1, \ldots, K^i_d$. Fig. 1(b) shows an example of motion primitives learned from the trajectory data. Note that motion primitives can overlap each other. Similar to trajectories, motion primitives are represented in grid, but they usually have sparse non-zero values, as each motion primitive usually represents a part of the trajectory. More precisely, trajectories are segmented into clusters, each cluster is presented by a motion primitive (see Fig. 1(c)). As illustrated, A trajectory typically consists of more than one segment (motion primitive), such that a pedestrian transits from one primitive to another motion primitive in a walking scenario. To model the transition between motion primitives, we define $R^{(i)}_{K^i_d\times K^i_d}$ called transition matrix, such that each entity $r^{ij}_{K^i_d\times K^i_d}$ specifies the number of transitions from motion primitive $m^i_j$ to $m^i_k$ observed in the training data. Note that this matrix is asymmetric in general.

---

**Fig. 1.** (a) Trajectories are discretized in grid world. Note that the activation cells are expanded when the trajectory is close to the cell boundaries. (b) The model is trained as a set of motion primitives and their transitions learned by ASNSC-based algorithms [4], [5]. (c) Illustration of trajectories clustered in model $M^i$, each cluster is represented by motion primitives shown in (b), with the same color. The observed transitions between trajectory segments are considered as $r^{ij}_{fg}$. (d) The dual graph includes all non-zero transitions i.e., $r^{ij}_{fg} \neq 0$.
To represent motion primitives and their transitions, we use motion primitive dual graph, where the nodes are motion primitives and the edges are the transition between two motion primitives. This graph is a weighted directed graph, where the weights are the values of $R^i$ entities corresponding to that indices. Each transition $r_{fg}$ in $R^i$ are modeled by two independent Gaussian Processes (GPs): $GPy_{x_{fg}}^i, GPy_{y_{fg}}^i$. These GPs are used to learn the velocity components as a function of $(x, y)$ so that $v_x^i = GPy_{x_{fg}}^i(x, y)$ and $v_y^i = GPy_{y_{fg}}^i(x, y)$ where the trajectory indices are omitted for brevity.

**B. Problem Statement**

Given a set of agents $A^{(i)}, i = 1, \ldots, n$ and their learned models $M^{(i)} = \{D^{(i)}, R^{(i)}\}, i = 1, \ldots, n$. These agents can share their knowledge via V2X communication, i.e., Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) [21]. The goal is to design a fusion algorithm such that each (ego) agent $A^i$ compares its model $M^{(i)}$ with models received from other agents, $M^{(j)}, j = 1, \ldots, N^i$, and then update its model by fusion of the received models $M^{(j)}$. We previously presented SILA for fusion of two models when a single agent incrementally updates its model, which means only two models are fused. In multi-agent setting when multiple models from neighboring agents are available, i.e., $N^i \geq 2$, pairwise fusion may not be efficient as it requires iteratively fusion of the models until all the models are fused. This work extends the model fusion to any number of $N^i \geq 2$ models to improve both the accuracy and scalability.

**IV. APPROACH**

Let model in agent $A^i$ at time $t$ be denoted by $M^{(i)}(t)$ which compromises motion primitive vectors $D^{(i)}(t)$ and transition matrix $R^{(i)}(t)$, and assume $N^i(t)$ models are shared with $A^i$, denoted by $M^{(t)}(t) = \{M^{(j)}(t), R^{(j)}(t)\}_{j=1}^{N^i(t)}$. We propose a Similarity-based multi-model Fusion approach (SimFuse) to update the model $M^{(i)}(t)$ at time $t$ by fusing the models in $M^{(j(t))}$. SimFuse has three main steps: (1) Pairwise similarity scores between all motion primitives in $D^{(i)}$ with those in $D' = \{D^{(j)}\}_{j=1}^{N^i(t)}$ are computed; (2) The matching graph $G_s$ for the similarity larger than a given threshold is created and make $G_s$ consistent using edge relaxation; (3) Matched motion primitives and their corresponding transitions are fused and the non-matched motion primitives and transitions are added to model. Model is updated to $M^{(i)}(t+1)$. (See Fig. 2 for the block diagram and Algorithm 1 for the pseudo code). Note that SILA has similar steps. However, the main contribution of this work is step 2 and 3. In step 2, the new relaxing edges algorithm is proposed to make the graph consistent. In step 3, we propose a new scalable algorithm for the fusion of multiple GPs.

**A. Step 1: Computing Pairwise Similarity Score**

Inspired by the concept of coherence of dictionary atoms [22], [23] and similar to [1], the pairwise similarity between two motion primitive vectors $m_{i}^{(f)} \in D^{(i)}$ and $m_{g}^{(j)} \in D^{(j)}$ are computed as the normalized inner product of them, known as cosine similarity:

$$S(m_{i}^{(f)}, m_{g}^{(j)}) = \frac{\langle m_{i}^{(f)}, m_{g}^{(j)} \rangle}{|m_{i}^{(f)}||m_{g}^{(j)}|}$$

where, $\langle \cdot, \cdot \rangle$ is the inner product, $m_{i}^{(f)}$ and $m_{g}^{(j)}$ denote the $f$-th motion primitive in model $M^{(i)}$ and the $g$-th motion primitive in model $M^{(j)}$ respectively. Recall each motion primitive is defined in $P \times Q$ cells with two components of $(v_{x}^{k}, v_{y}^{k}), k \in P \times Q$. Cosine similarity is the average of cosine of the angles between motion primitive vectors in each grid cell. If a motion primitive is not present in a cell where another primitive presents, their similarity is zero in that cell. Thus, we can measure the similarity between motion primitives both in terms of the shape, i.e., cosine angle in common cells, and their locations, i.e., the number of cells they overlap.

**B. Step 2: Matching Graph and Edge Relaxation Algorithm**

Pairs of motion primitives $m_{i}^{(f)}$ and $m_{g}^{(j)}$ with similarity score $S(m_{i}^{(f)}, m_{g}^{(j)})$ greater than a predefined similarity threshold, $\beta$, are considered as matched motion primitives. The matched motion primitives build a matching graph $G_s$, where the edges represent the pairwise matching between motion
primitive in model $M^{(i)}$ and motion primitives in models in $M'$. Note this graph does not consider the matching between motion primitives within models $M'$. Also, $G_s$ by definition does not include unmatched nodes which implies that the unmatched nodes are not going to be fused (see Fig. 3(b), nodes with dashed circle). This figure shows an example of relaxing the edges using the proposed method.

Fig. 3 shows an example of inconsistent matching. In this case it is not clear which sets of motion primitives have to be fused. Consistency in matching graph has been studied for a while. However, in those approaches there exists a real “true label” [24] for matched nodes, that is, there is an optimal solution which achieves “correct” matching. In motion primitive application, however, there is not such a ground truth to optimize the matching. Instead, the matched nodes with highest similarity is suggested to be fused. We propose a relaxing edges algorithm in which edges with the least similarity scores are successively relaxed (removed) until only one edge remains between matching nodes from $u_j \in M^{(i)}$ and $l_g \in M'^{(j)}$. Fig. 3 shows an example of relaxing the edges using the proposed method.

C. Step 3.1: Fusion of Motion Primitives

Given a consistent $G_s$ with $N_c$ matched components, the fusion step consists of fusing matched nodes, and fusing the transitions (modeled as GPs) in each matching component $\{C_o\}_{o=1}^{N_c}$. Let $\text{mm}_o = \{m_\omega \in \mathbb{R}^{3PQ}\}_{\omega=1}^{[C_o]}$ be motion primitives in $C_o$, and the size of motion primitives in $C_o$ is $[C_o]$. $N_m = 2PQ$ is the first $2PQ$ dimension of each motion primitive vector (i.e., their (xy) velocity components). To effectively fuse motion primitives, we simply compute the element-wise average between the all vectors in $\text{mm}_o$, denoted by $\text{fuse}(\text{mm}_o)$:

$$\text{fuse}(\text{mm}_{o,k}) = \frac{1}{[C_o]} \sum_{\omega=1}^{[C_o]} m_{\omega,k}$$

(2)

Where, $m_{\omega,k}$ is the $k$-th dimension of $m_\omega \in \text{mm}_o$, $k = 1, \ldots, 2PQ$ (see Fig. 4(a)). The last $PQ$-th dimension of the fused primitive (i.e., the activeness variable and size of motion primitive in $C_o$) as a necessary condition for achieving consistency in $C_o$. That means, there is only one and only one edge between any motion primitive $u_j$ in $M^{(i)}$ and a motion primitive $l_g$ in $M'^{(j)}$ in a consistent graph.

Conjecture 1: To fuse similar motion primitives between $M^{(i)}$ and $M'^{(j)}$, each matched set should be consistent.

Fig. 3(a) shows an example of inconsistent matching. In this case it is not clear which sets of motion primitives have to be fused. Consistency in matching graph has been studied for a while. However, in those approaches there exists a real "true label" [24] for matched nodes, that is, there is an optimal solution which achieves "correct" matching. In motion primitive application, however, there is not such a ground truth to optimize the matching. Instead, the matched nodes with highest similarity is suggested to be fused. We propose a relaxing edges algorithm in which edges with the least similarity scores are successively relaxed (removed) until only one edge remains between matching nodes from $u_j \in M^{(i)}$ and $l_g \in M'^{(j)}$. Fig. 3 shows an example of relaxing the edges using the proposed method.

Fig. 4. (a) Fusion of motion primitives from three models (green, red, blue), the final fused primitive is shown in black. (b) Prediction distribution with three branches and their likelihood (figures are seen better in colors).
D. Step 3.2: Fusion of Transitions (GPs)

GP regression algorithms are powerful non-parametric approaches for predicting the functions with uncertainty. However, they usually do not scale well to large dataset [25]. One approach to improve GP scalability is to approximate them by a small set of inducing inputs. For example [26] introduced pseudo inputs, which reduces the complexity from $O(N_2^2)$ to $O(N_2^p N_d)$, where $N_d$ is the number of data points and $N_I$ is the number of inducing inputs, and $N_I \ll N_d$. In this letter, we propose a scalable framework for the fusion of multiple GPs. The approach taken here combines these two ideas to fuse $N_G$ GPs. We assume each $GP_g, g = 1, \ldots, N_G$ is represented by a set of pseudo inputs with size of $n_g^2$. We build a new dataset, called induced data $d_{ind}^t$ by augmenting pseudo inputs from each GPs, assuming the pseudo inputs are sufficiently representative of the original dataset. We also assume $n_g^2 = N_I, g = 1, \ldots, N_G$. Using this approximation, the size of data is reduced to $|d_{ind}^t| = N_G N_I$. Without this approximation, the original dataset size is $|d_{orig}^t| = \sum_{g=1}^{N_G} |d_{orig}^{gp_g}|$, where $|d_{orig}^{gp_g}|$ is the size of data points associated with $g$-th GP. Since $N_I \ll |d_{orig}^{gp_g}|$, our new approach significantly reduces the size of the dataset and it is scalable for large $N_G$.

Using this new dataset, we apply the method in [26] to learn sparse GPs in the form of pseudo inputs $I'$ along with the hyper-parameters $H'$. The proposed GP fusion method is summarized in Algorithm 2 (see [26] for more details of learning pseudo inputs). The advantage of this approach is to provide a scalable framework for fusion of GPs. Recall, each transition in a model is modeled by two GPs: $(GP_x, GP_y)$. To fuse a set of transitions, their corresponding GPs are fused using the proposed method explained above.

E. Prediction Phase

Updated model comprises of updated motion primitives and transitions. When a new trajectory is observed in the test environment, the closest motion primitive to the observed trajectory is selected as “current primitive”. Transition matrix $R^{t}$ and its GPs component are used to estimate next motion primitives branched from the current primitive and its associated likelihood respectively. Fig. 4(b) shows an example of prediction of a future trajectory. The estimation is a set of branches with their associated likelihood. This prediction procedure is common in all ANNSC-based algorithms [4] discussed in the Section V. For evaluation, the trajectories are sampled from the distribution of the prediction which will be explained in Section V.

V. RESULT

A. Evaluation Scenarios and Experiment Setup

Two sets of scenarios are considered for evaluation of SimFuse and baselines: intersection and non-intersection scenarios. In both scenarios trajectories are sampled with the frequency of 2.5 Hz. Although our algorithm is able to be trained and tested on trajectories with any length, we select only trajectories with the length at no shorter than length of 6 secs (20 frames) to have a fair comparison with other baselines. For testing, each trajectory is observed for 3.2 sec (8 frames) and is predicted for 4.8 sec (12 frames). We adopt the common “leave one out” method, where one scene from $N_s$ scenes is considered as test scene and the model is trained on remaining $N_s - 1$ scenes. For distributed learning approaches (including SimFuse, SILA, SimFuse-Naive baselines), $N_s - 1$ agents are trained in a particular assignment (e.g. agent 1 is trained from scene 1, agent 2 is learned from scene 2, and so on). The final fused model is tested on the remaining scene.

B. Evaluation Metrics

1) Average Displacement Error (ADE) [10]: defined as the average Euclidean distance (L2 norm) between the predicted and ground truth trajectory over a fixed time horizon $T$:

$$ADE = \frac{1}{T} \sum_{t=0}^{T} \|p_{t}^{pred} - p_{t}^{g}\|_2,$$  

(4)

where $p_{t}^{pred}$ and $p_{t}^{g}$ are points at time $t$ in the predicted and ground truth trajectories respectively. $t_{obs}$ is the time corresponding to the last observed point, before prediction starts.
2) Final Displacement Error (FDE): defined as the Euclidean distance between predicted and ground truth trajectories at horizon time $T$ [9]:

$$FDE = \left\| p_{\text{pred}}^{t_{obs}+T} - p_{\text{gt}}^{t_{obs}+T} \right\|_2.$$  

(5)

C. Baselines

Three groups of baselines are considered for the evaluation:

1) Linear model or constant velocity model (in red, Table I)

2) Interaction-aware approaches [7], [9], [27]–[30] (in blue, Table I), all of them are batch learning (when the model is learned from entire dataset) and they incorporate the interaction between pedestrian in prediction model.

3) Non-interaction aware, but context aware approaches (in magenta, Table I), which are the variations of ASNSC, including batch setting of SimFuse (TASNSC) [5]; SimFuse which fuses $N_e - 1$ models in each time step; Modified SILA by considering the consistency in matching graph, referred as SILA in all comparisons; SimFuse-naive baseline, where models are added without any fusion.

As discussed in Section IV, ASNSC-based baselines including SimFuse, output a distribution of future trajectories. To have a fair comparison with other baselines [7], $K = 20$ trajectories are sampled and the one with minimum ADE is selected for the evaluation.

D. Experiment 1: Non-Intersection Scenarios

Five publicly available datasets of ETH, HOTEL, UNIV, ZARA1, and ZARA2 have been used [10], [31]. Table I summarizes the results, where each column corresponds to each testing scene. The results confirm SimFuse generally has a comparable performance with other baselines in terms of ADE, but it achieves the best performance in terms of ADE in average, showing that SimFuse generalizes to unseen scenes better than other algorithms. Model fusion in SimFuse helps removing redundancy in learned motion behaviors and thus avoids over-fitting. Since SimFuse does not consider the interaction between the pedestrians, its FDE as a measure of long-term prediction is not as good as interaction-aware baselines. Interestingly, SimFuse outperforms its batch setting (TASNSC) in all scenes. In TASNSC, the predicted trajectories are computed as the result of a non-linear optimization problem (for learning the motion primitives) and non-parametric Gaussian regression (for modeling the transitions). This such non-linearity leads to get a sub optimal solution, which means the batch setting in TASNSC does not necessarily lead to a better performance. On the other hand, the fusion of sub-optimal results in SimFuse may lead to updated model with better performance. Moreover, TASNSC loses a level of accuracy as it compromises the prediction accuracy to optimize the number of motion primitives. As a result, learning from a small dataset in non-batch setting (i.e., SimFuse, SimFuse-Naive and SILA) could achieve a better accuracy, leading to higher performance, but in the cost of increasing the model size.

E. Experiment 2: Intersection Scenarios

We have collected pedestrian trajectories in four different intersections in Boston/Cambridge streets shown on top of the Table II denoted by $A$, $C$, $D$, $E$. data related to $A$, $E$ were collected by a Polaris GEM vehicle equipped with three Logitech C920 cameras and a SICK LMS151 LIDAR [32]. For collecting data at intersections $C$, $D$, we have used a 3D LiDAR sensor (Velodyne VLP-16) which was mounted on a tripod. Intersection $B$ is located at MCity, Michigan. This data was collected by tracking pedestrians when they were wearing RTK GPS hats. These five intersection dataset contain 1024 trajectories in total. In this experiment, the data is normalized respect to the intersection corner geometry similar to TASNSC [5]. We compare our algorithm with other ASNSC-based algorithms, as well as Social-STGCNN [7] which is known as the State-of-the-art among the interaction-aware baselines. We follow leave-one-out setting in the previous experiments. The results show ASNSC based algorithms (magenta) outperform Social-STGCNN in all-except-one scenarios. This result confirms that incorporating environment context (i.e., distance to the curbside) significantly improves the prediction in intersection scenarios, such that all TASNSC-based algorithm outperform Social-STGCNN which does not consider the environment context. The results show SimFuse performs the best in this experiment.

F. Experiment 3: Updating Time Analysis

In this section we analyze the updating time in SimFuse and then show how it is supported in an experiment. Assume the matching graph of $N^i + 1$ models has $N_c$ components. The fusion time is governed by the number of the edges/nodes to be fused in each time step, which is $|E_{\text{total}}| = (N^i + 1)N_c$ for SimFuse in the worst case scenario, when all models are similar to each other. In SILA this number is reduced to $2N_c$ since only two models are fused. However, the pairwise fusion repeats for $N^i$ times and the total edges to be fused is $|E_{\text{total}}| = 2 N^i N_c$.
TABLE II
PERFORMANCE COMPARISON IN THE FORMAT OF (ADE/FDE) AT FIVE INTERSECTION SCENARIOS, LOWER IS BETTER

<table>
<thead>
<tr>
<th>Algorithm/Domain</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social-STGCNN</td>
<td>0.96/1.62</td>
<td>1.63/2.71</td>
<td>0.87/1.36</td>
<td>1.39/2.24</td>
<td>1.57/2.95</td>
<td>1.27/2.17</td>
</tr>
<tr>
<td>TASNSC (batch)</td>
<td>0.68/1.30</td>
<td>0.78/1.65</td>
<td>1.00/2.13</td>
<td>1.00/2.18</td>
<td>0.96/2.02</td>
<td>0.88/1.86</td>
</tr>
<tr>
<td>SimFuse-naive (no fusion)</td>
<td>0.68/1.37</td>
<td>0.79/2.05</td>
<td>0.78/1.63</td>
<td>1.15/2.40</td>
<td>0.90/1.75</td>
<td>0.90/1.65</td>
</tr>
<tr>
<td>SILA [1]</td>
<td>0.70/1.39</td>
<td>0.86/1.80</td>
<td>0.86/1.82</td>
<td>1.05/2.31</td>
<td>0.99/2.07</td>
<td>0.89/1.87</td>
</tr>
<tr>
<td>SimFuse</td>
<td>0.67/1.37</td>
<td>0.96/2.04</td>
<td>0.77/1.64</td>
<td>1.13/2.41</td>
<td>0.88/1.76</td>
<td>0.88/1.84</td>
</tr>
</tbody>
</table>

In practice, the number of nodes in each matched component is much smaller than $N^4$ and and SILA typically requires more time compared to SimFuse for the fusion of $N^4 + 1$ models. To show this, we have conducted an experiment in which 848 trajectories collected in intersection A, and data are partitioned into 35 small batches, which are fed into 35 agents, implying each agent learns from 29 trajectories. Table III compares the updating time between SILA and SimFuse in each time step in average and total time required for an agent to update its model with 34 other agents model. We also compute the average number edges to be fused in each fusion step. The results confirm the updating time in each time step is almost proportional to the number of fused edges. e.g., in this example the ratio of number of matched edges in SimFuse to SILA is close to the ratio of their fusion time in each time step ($\approx 3.41$ vs. $\approx 3.33$), which confirms the correctness of the analysis. Note that SILA has to run the fusion step for all models which requires significantly longer time to reach the final updated model in total. To compare the performance, the final updated model from each algorithm is used to predict the trajectory set at intersection C, 167 trajectories in total. This results shows the agent improves its initial prediction up to 20%, when it fuses its model with other 34 agents. Although SimFuse-Naive achieves the best performance in this experiment, its model-size is growing so fast due to no fusion.

G. Experiment 4: Distributed Learning of Motion Primitives

In ad-hoc communications scenarios, the assumption of fully connectivity may violate. The agents share their knowledge via pairwise communication (communicating with one neighbor at a time) which limits to number of model fusion to two. This is essentially similar to SILA approach, with slight modification in relaxing edges, and the extension of SILA to perform as a distributed solution in ad-hoc communication environments.

Fig. 5(a) shows an experiment, where five agents $A - E$, each exploring the intersections $A - E$ respectively, but their communications are constrained by a connectivity graph shown in

H. Effect of Similarity Threshold

We study the effect of similarity threshold in SimFuse on prediction error (ADE) and model size, defined as the number of motion primitives and transitions. As the similarity threshold increases, the number of matched motion primitives between models decreases leading to less number of fusion and increasing the final model size. On the other hand, decreasing threshold increases the fusion, causing more fusion, which may create a model with higher prediction error. Fig. 6 shows model size versus prediction error (ADE) in an intersection scenario experiment described in Experiment 3, when similarity threshold $\beta$ is...
gets the best trade-off between model size and error.

\[
\beta = 0.5 \rightarrow \beta = 0.6 \rightarrow \beta = 0.7 \rightarrow \beta = 0.8 \rightarrow \beta = 0.9 \rightarrow \beta = 1
\]


β varies from β = 0.5 to β = 1 (i.e., almost no fusion). SimFuse with β = 0.7 gets the best trade-off between model size and error.

VI. CONCLUSION

We have proposed a multi-model fusion framework for a set of agents that enables them to update their prediction models by communicating with other agents and fuse their own model with models shared by multiple other agents. The results confirm SimFuse outperforms state of the art in terms of prediction error in intersection scenarios for pedestrian trajectory prediction, while the model size remains sufficiently small compared to Naïve approach. SimFuse is not limited to learn human trajectories, it can also be used for understanding other motion behaviors such as cyclists and cars, which is left for the future work. Another future study would be to combine SimFuse with SILA such that each agent updates its model by incrementally explore the environments and learning from the data when arrived.

REFERENCES


