Learning local behavioral sequences to better infer non-local properties in real multi-robot systems

Taeyeong Choi, Sehyeok Kang, and Theodore P. Pavlic, *Member, IEEE*

***Abstract*— We propose a more powerful machine learning approach, as an extension of [1], to tackle the Remote Teammate Localization (RTL) problem where a robot member in a multi-robot team is to predict positions of all other teammates only using the observations on its nearest neighbor without any communication between robots. In the previous work, we showed feasibility of a scalable method by which a predictor robot was trained in a modular 3-robot team but could extend the prediction to a larger team without additional training, also suggesting an application of such an inference in caging mission. In this work, however, we focus mainly on 1) achieving better performance to improve the applicability and 2) conducting evaluation in more realistic environments. To be specific, we adopt a Long-Short Term Memory (LSTM) [2] to learn possible evolution of behaviors in a modular team helping reduce the errors from regression outcomes. Furthermore, while the previous work relied only on computer simulations, all the experiments here are conducted on a physical two-wheeled robotic platform, *Thymio*, to demonstrate the performance gain under realistic constraints.**

I. Introduction

In multi-robot systems including swarms, every robot is usually allowed to observe only a subset of its team members and interact with them to determine the next action according to relatively simple motion rules. Such a property enables the entire system to perform in a distributed manner, and also the behavior of it can be driven easily by a few leader robots to eventually achieve the goal [1], [3]–[5]. This implies that if a robot has an ability to reognize useful property, e.g.) formation, of the whole team in real time using its local sensors, it could present adjustive actions accordingly to better promote a collective behavior for the sake of the team.

In [1], we suggested a machine learning method to solve the RTL problem where a robot, called *Tail* at one end of a line formation of a multi-robot team, is to predict positions of all other teammates only using local observations about a single nearby teammate. Since each robot has a limited sensor radius and a relatively simple motion rule with dependency on the position of its nearest neighbors, the *Tail* has to be able to learn the regularity of the observed motions of neighbor to finally infer the poses of all other robots. We introduced a repetitive prediction scheme to use predictions on nearer teammates to make predictions on farther ones until the prediction reached the *Head* robot at the other end in line formation. Also, computer simulations showed the feasibility of the method especially with caging scenario in which the *Tail* could recognize a caging action of *Head* in early stage and promote a proactive maneuver to better assist in team cooperation.

*This work was supported in part by NSF grant #1735579.

All authors are with Arizona State University, Tempe, AZ, 85281, USA. T. Choi, S. Kang, and T. P. Pavlic are with the School of Computing, Informatics, and Decision Systems Engineering. T. P. Pavlic is also with the School of Sustainability and the School of Life Sciences. {tchoi4, skang66, tpavlic}@asu.edu

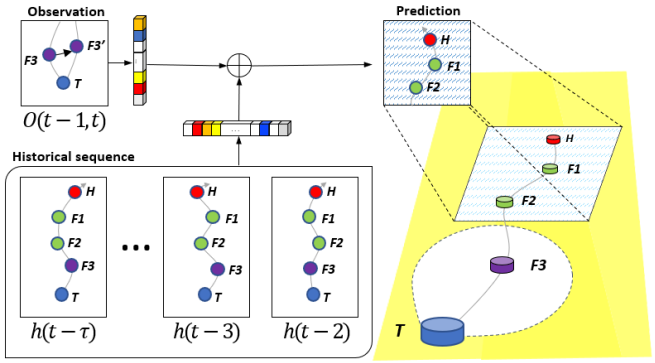

Fig. 1: Illustration of our proposed pipepline in a snapshot example of 5 robots at time t. Each robot has a limited view and a motion rule dependent on its neighbors except the *Head* robot leading the team at the front. *Tail* uses recent observations on its neighbor, *Follower 3*, which is denoted as $O(t-1,t)$. A sequence of historical poses, h, is also encoded for the model to make a final prediction on the unseen teammates.

RTL problem provides some unique characteristics compared to general localization problems. In RTL, the robot does not execute predictions on its own location but on its teammates using accessible information. Moreover, the robot is not allowed to communicate with other members during prediction phase to consider communication-free scenarios, which differs from cooperative localization problem. Hence, RTL usually assumes that robots behave with correlations with their neighbors so that the resulting behaviors contain cues about the state of their neighbors. In this sense, state observers in networked robotic system could be a more similar configuration to RTL, but RTL more emphasizes scalable applications to variable sizes of robot team as well as a general framework that would impose little constraint on dynamics of the system.

Figure 1 illustrates our proposed pipeline in which a deep neural network is to learn to synthesize the observations on its neighbor with knowledge about historical positions of all the teammates. As shown in Fig. 2, a LSTM layer is deployed to encode the historical sequence input, which could learn probable evolution of the team shape over time under physical constraints. The sequence encoding could help filter out impossible solution candidates that the model might produce if it only utilized the recent observations on the neighbor, as in [1],

Not only a more powerful model but also a more re-

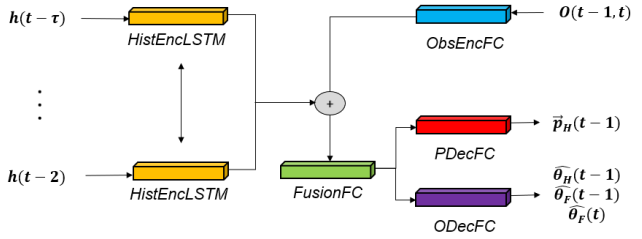

Fig. 2: Structure of our proposed deep neural network. This is an snapshot example when applied to a focused module of 3 robots, called *Tail, Follower, Head* within it, at time t. The Encoder-Decoder structure encodes 1) historical positions and orientations of *Follower* and *Head* until $t-2$ and 2) the observed positions of the *Follower* at $t-1$ and t. The decoder part learns to estimate 1) the position of *Head* at $t-1$ and 2) orientations of *Follower* at $t-1$ and t and *Head* at $t-1$.

alistic experiment environment are prepared. Previously, we conducted all demonstrations on computer simulations, but in this work, a physical two-wheeled robotic platform, *Thymio* [6], is mainly used to set up more realistic environments.

This paper is organized as follow. In Section II, we explore related literature and the distinction of our work. Section III explains more details about our setting of RTL problem. Then, we introduce our method, *IPY-Net*, in Section IV, and Section V explains details about experiments performed on real robots, including data collection, hyperparameters used for learning, and the results. Lastly, we summarize our research and discuss future directions in Section VI.

II. Related work

Although the RTL problem is to localize robots, as mentioned in Section I, it has unique properties compared to conventional localization problems including cooperative localization or tracking [7]–[10]. It is rather about robot behavior as a container to represent useful team-level state and learning to decode it. We discuss related literature below that introduced scenarios to use and decrypt robot behaviors in multi-robot system, followed by a comparison with a work that exploited deep learning to learn dynamics of robot on presentation of force input. Lastly, we explain similarities to our previous work as well as main contributions as an extension.

A. Group state recognition

Our approach is basically an attempt to infer positional state of the entire multi-robot team, using only locally obtainable information in the aspect of a robot member. This is because the knowledge about global configuration could help make a better decision for the sake of whole team. In this spirit, the authors in [11], [12] sampled a subset of robot agents from a swarm to monitor their local interactions and classify the swarm-level behavior. In fact, they implemented relatively simple robots run on a virtual simulator and predefined selections of global behaviors such as *flock* and *torus*. In contrast, our work is to estimate pose of real robots, which would require a more powerful model to regress on any value in a wide range of arena size.

B. Behavioral cue interpretation

One of the main assumptions in RTL is the simple motion rule of each robot which has dependency on the state of its neighbors. This implies that the behavior of neighbors could be a method to encrypt the state change of more remote teammates. Novitzky at el. [13] and Das et al. [14] employed a type of special action, which appeared like "waggle dance" of honeybees [15], in robot team so that the behavior could convey a meaning between robots. Our work also encourages a robot to decode behavioral cues, but it occurs without entering an explicit phase of signaling. In other words, the robot has to learn to gain useful information from a sequence of continuous usual interactions.

C. Robot dynamics learning

Byranvan and Fox [16] proposed a deep learning approach to predict the next visual frame given a current frame and the information of force that would affect one of the shown objects. One of their examples was to understand the dynamics of robotic arms and the relationship with control commands possibly executed. In the RTL problem also, the *Tail* robot may have known a global shape of robot team, and it has to be able to predict the future formation as a new observation on its neighbor is provided, which would encode the information of an unseen force driving the team. Such a similarity inspired the architecture of our neural network model, but they focused on learning motions of rigid objects, while a chain of robots in our work can present much flexibility in team shape. In addition, the force is inferred by movements of the neighbor, which would likely contain noise to some level, while *SE3-Nets* is provided with clear information about the applied force.

D. Scalable teammate localization

In our previous work [1], the detailed configurations for RTL problem were first introduced with realistic assumptions. To solve the problem particularly with little robot-to-robot communication, the *Tail* robot is designed to use a repetitive strategy of inference with which the predictions on a closer robot are used as input to prediction for more distant team members. Such an approach enables to scale up the estimation capability without further training even though more robots are added. Albeit we use the same prediction scheme with repetition, here we focus more on developing a more powerful regressor to broaden applicability and also on performing realistic evaluations in a multi-robot system realized on an off-the-shelf robotic platform.

III. RTL Problem & System Design

In our previous work [1] , we introduced a design of robotic system and the RTL problem to solve on it. For this work, we build the robot team based on the previous design, where the robot team is implemented to move in a line formation in which each robot has two adjacent neighbors except for the ones at the ends, each of which has only one neighbor. The robot on one end is called *Head*, which is the only robot that drives the team toward a destination

predetermined in an independent manner. The robot on the opposite end is named *Tail*, and all other robots in-between are *Follower* i where $i \in \{1, 2, ..., n-2\}$ starting from the robot closest to the *Tail*.

Every robot has the same capability of sensor and motion under the same kinematic constraints, although they may take different actions according to motion rules depending on their roles, as formulated in [1]. This catalyzes the *Follower* robots and the *Tail* to behave based on the current positions of their close neighbors. For simplicity, we assume that every robot has an ability to accelerate fast enough to always keep the neighbors within their sensory range.

The RTL is to localize all the teammates from the view of *Tail* only using locally visible information, which is the position of *Follower* 1 at every time step. We assume that until a specific time instant τ, all the information about positions and orientations of all robots have been shared especially with the *Tail* robot, possibly via global communication, but from $\tau+1$, such data becomes no longer available causing a need of the localization technique only utilizing local observations.

More formally, at the time instant τ, the following set of poses of all teammates is available:

$$\{\vec{p}_{r@t}, \theta_{r@t}\}$$

where $\vec{p}_{r@t} \triangleq (x_{r@t}, y_{r@t})$ is the position of robot r at time t where $r \in \{T, F_1, F_2, ..., F_{n-2}, H\}$ and $t \leq \tau$.

Also, at each time $t > \tau$, the position of F_1, $\vec{p}_{F_{1@t}}$ is observed. That is, the RTL problem is to use all the available information to predict the pose set above at $t' > \tau$.

IV. Method

First, we briefly revisit the scalable prediction approach, shown in [1], to train the model on a small team and to be able to produce predictions even on a larger team without additional training. Then, we present the regressor built for learning not only with observations on the neighbor but also with a sequence of past team formations.

A. Scalable prediction

The basic idea in [1] is that the *Tail* uses all the available information such as the observation on its neighbor at t and $t+1$ to predict $\vec{p}_{H@t}$ of *Head* in its subteam of 3 robots, which is actually *Follower 2* in the entire team. After twice applications of such a prediction until $t+2$, the prediction outputs, $\vec{p}_{H@t}$ and $\vec{p}_{H@t+1}$, enable to perform the same process in the subteam of the next 3 robots, which are *Follower 1, Follower 2*, and *Follower 3*, as if *Follower 1* and *Follower 3* were *Tail* and *Head* in that subteam, respectively.

Since the inference is applied in each of subteam in a repetitive manner, a predictor can be trained simply with a 3-robot team and still be used for a larger size of team without additional training for it. However, the extension of modular prediction brings about a time delay in making a prediction for a far robot. Specifically, a prediction for *Head* in a subteam requires a series of at least 2 observations on *Follower*, and in other words, the initial position of robot i can be estimated at time i for the first time where robot i refers to the robot that is i robots ahead of *Tail*.

In the next section, we discuss how to incorporate such a prediction strategy with the proposed model regressor in more technical details.

B. Regression model

Because the learned regressor is designed to work in a modular team of 3 robots, all the notations here for *Head, Follower*, and *Tail* only indicate them. In addition, all positions noted here are assumed to be relative positions to the *Tail* robot, since in practice, the *Tail* has to understand positions of others by projection onto the local coordinate system centered at itself. Though a similar conversion may be considered for orientation, we use absolute direction in this work because assuming robots equipped with an instrument such as a electric compass is acceptable.

Previously, in [1] we used a series of fully-connected (FC) layers to only take the observation input and estimate the position and orientation of interest:

$$Y = f(X_{obs}) \tag{1}$$

where f is a series of *FC* layers,
$X_{obs} = (\vec{p}_{F@t}, \theta_{F@t}, \vec{p}_{F@t+1})$, and $Y = (\hat{\vec{p}}_{H@t}, \hat{\theta}_{F@t+1})$

As shown in Fig. 2, however, our proposed model uses not only an observation input but also a historical sequence of poses. First of all, we use a slightly different observation encoding:

$$o = f_{obs}(X_{obs}) \tag{2}$$

where f_{obs} is a FC layer, $X_{obs} = (\vec{p}_{F@t}, \vec{p}_{F@t+1})$, and $o \in \mathbb{R}^k$ is the encoded observation feature where k is the size of FC layer.

A historical sequence of poses is encoded by a bi-directional LSTM layer [17]:

$$h = f_{hist}(X_{hist}) \tag{3}$$

where f_{hist} is a LSTM layer, $X_{hist} = (\vec{p}_{H@t-l+1:t}, \theta_{H@t-l+1:t}, \vec{p}_{F@t-l+1:t}, \theta_{F@t-l+1:t})$, and $h \in \mathbb{R}^{2\times m}$ is the encoded history feature where l is the length of historical sequence, and m is the size of LSTM layer.

Then, o and h are synthesized by a layer ϕ, which is passed as input to two separate final regressors, g_p and g_θ.

$$\begin{aligned} Y_p &= g_p(\phi(o,h)), \\ Y_\theta &= g_\theta(\phi(o,h)) \end{aligned} \tag{4}$$

where $Y_p = \hat{\vec{p}}_{H@t}$ and $Y_\theta = (\hat{\theta}_{F@t}, \hat{\theta}_{H@t}, \hat{\theta}_{F@t+1})$.

For fusion of o and h features, layer ϕ in Equation 4 could be implemented by any type of layer. During our experiments, we built a FC layer of size d to find a non-linear relationship between the input features and achieved a satisfactory performance.

	Duration	Num. of Samples	Num. of Instances
3 robots	100.6 minutes	6,975	465
5 robots	45.0 minutes	8,736	208

TABLE I: Description of data collected from executions of 3-robot and 5-robot teams.

Furthermore, $\hat{\theta}_{F@t+1}$ gained with $\hat{\theta}_{F@t}$ is estimated again when $\hat{\theta}_{F@t+2}$ is estimated at the next prediction step. Although our model keeps the later estimate only, we discovered that involving it in both steps can regulate the regressor during learning to produce a model that achieves a better score in validation.

To find a best combination of model parameters mentioned above, we performed an extensive random search with choices of other learning parameters and finally set $k = 80, m = 160$, and $d = 160$.

V. Experiments

To demonstrate the effectiveness of our method, we employ a physical robotic platform, *Thymio* [6], which allows to execute a team of small two-wheeled mobile robots. We use a central computer connected with a overhead camera to simulate better proximity sensors, a more powerful computing power, and a GPS system that would easily run on each robot in real scenario. Specifically, the central system is set up to detect the locations of robots in real time using off-the-shelf computer vision packages and communicate with a *Raspberry Pi* board [18] on each robot to send the next command relying on its neighbors. The command was essentially obtained based on the formulation in [1], but as the robot was asked to move backward, we just set it not to move for its neighbor behind to catch it up, which helped gain smoother trajectories of the team and eventually avoid possible collisions between robots.

Although the experiment setting involves some external computations and sensors due to limited capability of *Thymio*, most of realistic assumptions still hold from the physical constraints and disturbance during execution. For better understanding of readers on the prepared environment, we submit a supplementary video.

Finally, we collected the pose data from two robot teams, one of 3 robots and one of 5 robots, that ran separately in an arena of $2.5m \times 1.9m$. The location detection was performed at 4 frames per second at each of which a new command was received by each robot. Also, all pose data was collected at the rate of 2 frames per second, which was not necessarily synchronized with the command timing. We set the length of history to 5 seconds (10 time steps in data recording) and the time window for prediction to the next 8 seconds (16 time steps). Table I provides details about the collected data where a sample refers to a set of coordinates and orientations in a subteam with which a prediction can be performed, and an instance is set of all available samples for 13 seconds from the entire team. We clustered recordings so that an instance has at least 7-second time gap to another ensuring the motions between separate instances have little dependency.

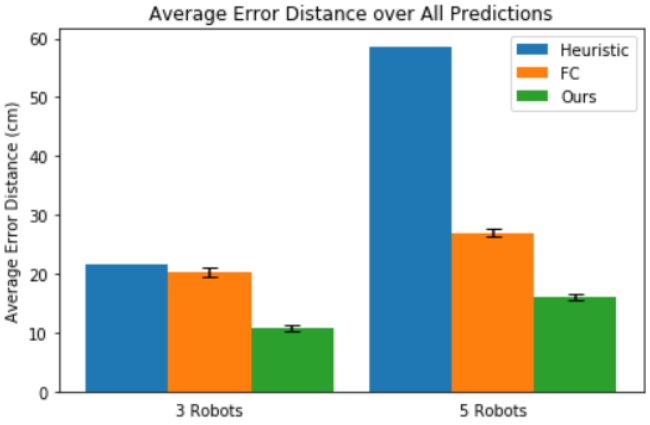

Fig. 3: Average accumulated error of each model in two different sizes of robot team. For each of machine learning methods, the mean performance of 5 separate training sessions is reported with a error bar to visualize the performance variation.

60% data from the 3-robot team is used to train our model, and another 10% was set aside for validation. At each epoch of training, a validation followed so that the learned weights that achieved the best validation performance were saved. The rest of 20% data and all the data from 5-robot were used to test the model.

Our model is implemented in *Tensorflow Python* library [1] to realize the entire pipeline, and it was trained to minimize loss functions such as Euclidean distance and mean absolute errors for position and orientation estimation, respectively.

We compare our proposed model to two different approaches such as:

- 2*X Heuristic*: The prediction on *Head* within a modular subteam is performed by doubling the vector $\vec{p}_F - \vec{p}_T$.
- *FC*: Two fully connected layers run in the predictor without historical information, which is based on [1].

A. Overall performance

First of all, we evaluate each model in a macroscopic view, as shown in Fig. 3, where each model is tested with the test data of 3-robot team and 5-robot team separately, and in each case, the averaged error for all position predictions is reported. Specifically, in the case of 3 robots, the average error is calculated only on the prediction for the *Head* robot, while in the 5-robot case, it involves all predictions for *Follower 2*, *Follower 3*, and *Head*. Moreover, for every model except the *2X Heuristic*, the mean performance of 5 separate learning sessions is reported with the standard deviation.

Figure 3 displays that the machine learning methods outperform the *2X Heuristic* in any case. In particular, the performance gap between *2X Heuristic* and *FC* becomes much larger in 5-robot case suggesting that during the data collection, we added much randomness to the shape of the robot team.

Also, our model clearly shows the performance improvement over the *FC* model, since the average error was reduced

[1] https://www.tensorflow.org/

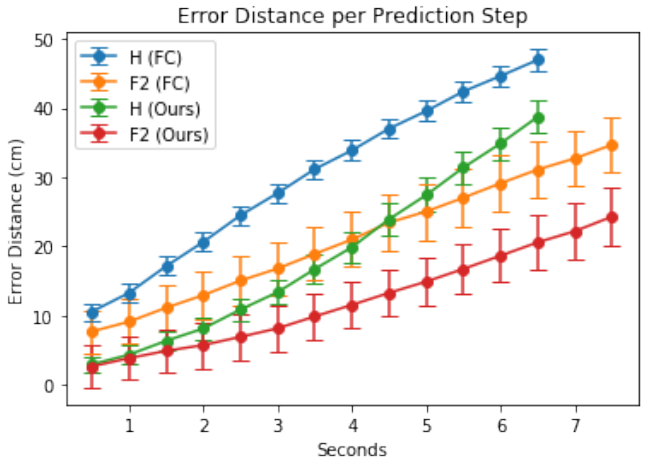

Fig. 4: Average step-wise error for different target robots in 5-robot team. For each model, all the prediction errors at different time steps are averaged for a specific robot. The error bars represent the standard deviation of 5 separate models in terms of the performance metric. For the sake of visualization, the error for *Follower 3* is omitted, but note that in any model, the error ranges between the two visualized errors at every step.

by 47% and 40% in the 3-robot and 5-robot case, respectively. In addition, considering the diameter of a *Thymio* robot is 12 cm, as only three robots are deployed, the average distance between the prediction and the true position is shorter than the length of the body. This overall result proves the effectiveness of encoding a sequence of historical behaviors as a feature input over the model fed only with very recent observation.

B. Microscopic analysis

In this section, we analyze the performance of the machine learning based models in more fine-grained perspective. For each model, Fig. 4 visualizes the step-wise error for different target robots (*Follower 2* and *Head*) in 5-robot case, which is calculated by averaging the errors of each step across instances. Such a way of evaluation can offer us a crucial insight to understand an approximate time frame within which a level of average error could be ensured for a specific robot. During this evaluation, we also had 5 separate training sessions to gain the mean and the variability of the performance.

Figure 4 shows that for any target robot, the error increases over time due to the design of the repetitive prediction method where previous errors would negatively impact the prediction power for the following steps. In a similar sense, each model brings about larger error on the *Head* prediction than on the *Follower 2*.

At every time step, our approach causes less error than the *FC* model for any robot, and the visualized error bars confirm that the performance gap is significant in any case. Especially, for first 4 seconds, the prediction on *Head* from our model appears more accurate than the prediction on *Follower 2* from the *FC* implying that until then, lower errors is produced on *Follower 3* as well.

1) Limitation: Figure 4 shows that using our regressor, we could trust the inference outcome until before first 3 seconds and 4.5 seconds depending on which robot to target between *Head* and *Follower 2*, since the body of a *Thymio* robot is around 12 cm. This actually relies on the acceptable degree of error and the diameter of robotic platform used in the problem to tackle. Still, our result demonstrates the potential of effectively alleviating prediction error by adopting a well-designed model, and also, the relatively short time period of guaranteed localization may be sufficient to save on the communication bandwidth within the robot team.

Also, the error distance when our model performs for *Head* increases relatively slow initially but after some point, the increase rate gradually becomes higher. This is contrast to the case of *FC*, which leads to a constant increase at each time step. This may be because as the historical features are extracted more from previous predictions instead of the truths, it could cause a more accumulated error in prediction outcome than using previous predictions only as observation input as in *FC*.

C. Qualitative results

Figure 5 displays three different instances of 5-robot team in which both the true positions of all robots and the predictions on them are represented. In overall, the predictions until 4 seconds are shown to be reliable, although there are some errors while the group of robots is turning with a high angle and the *Head* changes its destination. An observation is that the predictions tend to be located more inward the curve the team is moving on. Yet, even when the prediction for a nearer robot is away from the truth, the prediction for a farther robot appears correctly in some cases. Such a correction occurs probably because even if an observation input from a nearer robot was actually inaccurate, the past positions in input could alleviate the impact of it and still help generate a decent estimation result.

VI. Discussion & Future work

We have proposed a new design of regression model that can take into consideration the historical behavioral sequences to solve the RTL problem on a real robotic platform. Following the fundamental of scalable localization algorithm introduced in [1], our approach enables a robot at one end of string of robots to make repetitive predictions on poses of unseen teammates by taking prediction outcomes for nearer robots as input to prediction for farther ones.

We utilized a physical robotic system, *Thymio*, to collect datasets from 3-robot and 5-robot teams. Through empirical experiments, we explained that in overall, the proposed machine learning model offers more accurate estimation than other baselines. In addition, our analysis on timestep-wise errors helped explore the benefits from encoding historical behavior sequences as well as a drawback of it. Lastly, we visualized prediction outcomes from a set of samples to illustrate specific cases where the features from past behaviors could also reduce the estimation error passed from previous predictions for nearer robots.

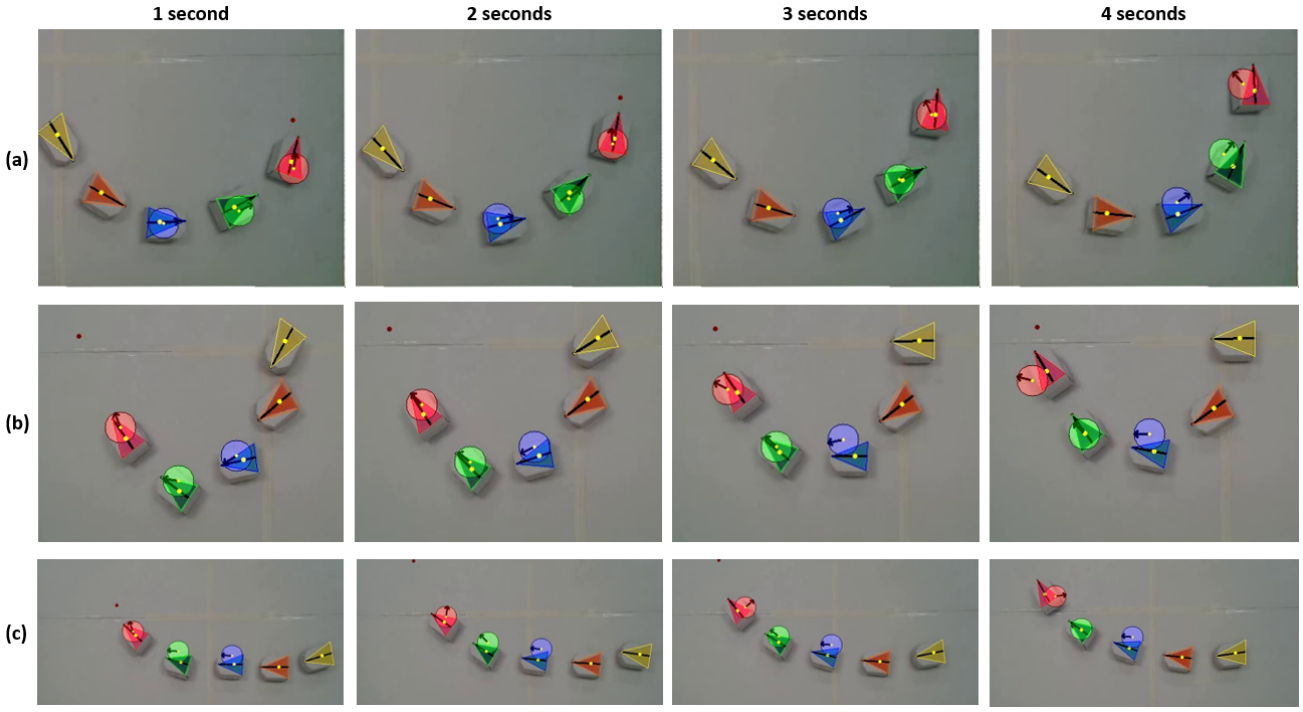

Fig. 5: Sample video frames with prediction results of our proposed model for three different instances of 5 robot team. Each row presents four frames of an instance for first 4 seconds. The yellow robot is *Tail*, the pink is *Head*, and in-between are *Follower* robots. The colored circles are predicted position outputs, in each of which a black arrow is drawn to indicate the predicted orientation as well. The colored triangles with a black line through the center are the results of localization detection used in data collection stage. A small red dot on the arena in each frame is the random destination the *Head* is moving toward, which is resampled at intervals.

In the future work, we could more deeply investigate the factors in model accuracy such as the length of history or types of team behavior that might favor the prediction scheme. Also, since the LSTM layer is exposed to various evolutions of team shape during training, the vector representation of it may be examined to characterize team states and finally detect abnormality of the whole system.

REFERENCES

[1] T. Choi, T. P. Pavlic, and A. W. Richa, "Automated synthesis of scalable algorithms for inferring non-local properties to assist in multi-robot teaming," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. IEEE, 2017, pp. 1522–1527.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "Improved leader election for self-organizing programmable matter," in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*. Springer, 2017, pp. 127–140.

[4] K. Elamvazhuthi and S. Berman, "Scalable formation control of multi-robot chain networks using a PDE abstraction," in *Proc. of the 12th International Symposium on Distributed Autonomous Robotic Systems*, 2016, pp. 357–369.

[5] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[6] J. Shin, R. Siegwart, and S. Magnenat, "Visual programming language for thymio ii robot," in *Conference on Interaction Design and Children (IDC'14)*. ETH Zürich, 2014.

[7] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized collaborative localization for sparsely communicating robots." in *Robotics: Science and Systems*. New York, NY, USA, 2016.

[8] A. Franchi, P. Stegagno, M. Di Rocco, and G. Oriolo, "Distributed target localization and encirclement with a multi-robot system," in *Proc. of the 7th IFAC Symposium on Intelligent Autonomous Vehicles*, 2010, pp. 151–156.

[9] Y. Cheng and D. Xie, "Distributed observer design for bounded tracking control of leader–follower multi-agent systems in a sampled-data setting," *International Journal of Control*, vol. 87, no. 1, pp. 41–51, 2014.

[10] O. De Silva, G. K. I. Mann, and R. G. Gosine, "Efficient distributed multi-robot localization: A target tracking inspired design," in *Proc. of the 2015 IEEE International Conference on Robotics and Automation*, 2015, pp. 434–439.

[11] D. S. Brown and M. A. Goodrich, "Limited bandwidth recognition of collective behaviors in bio-inspired swarms," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 405–412.

[12] M. Berger, L. M. Seversky, and D. S. Brown, "Classifying swarm behavior via compressive subspace learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5328–5335.

[13] M. Novitzky, C. Pippin, T. R. Collins, T. R. Balch, and M. E. West, "Bio-inspired multi-robot communication through behavior recognition," in *Proc. of the 2012 IEEE Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 771–776.

[14] B. Das, M. S. Couceiro, and P. A. Vargas, "MRoCS: A new multi-robot communication system based on passive action recognition," *Robotics and Autonomous Systems*, vol. 82, pp. 46–60, 2016.

[15] K. Von Frisch, *The Dance Language and Orientation of Bees*. Harvard University Press, 1967.

[16] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 173–180.

[17] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural

machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.
[18] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.