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*— When members of a multi-robot team follow regular motion rules sensitive to robots and other environmental factors within sensing range, the team itself may become an informational fabric for gaining situational awareness without explicit signalling among robots. In our previous work [1], we used machine learning to develop a scalable module, trained only on data from 3-robot teams, that could predict the positions of all robots in larger multi-robot teams based only on observations of the movement of a robot's nearest neighbor. Not only was this approach scalable from 3-to-many robots, but it did not require knowledge of the control laws of the robots under observation, as would a traditional observer-based approach. However, performance was only tested in simulation and could only be a substitute for explicit communication for short periods of time or in cases of very low sensing noise. In this work, we apply more sophisticated machine learning methods to data from a physically realized robotic team to develop Remote Teammate Localization (RTL) modules that can be used in realistic environments. To be specific, we adopt Long–Short-Term–Memory (LSTM) [2] to learn the evolution of behaviors in a modular team, which has the effect of greatly reducing errors from regression outcomes. In contrast with our previous work in simulation, all of the experiments conducted in this work were conducted on the *Thymio* physical, two-wheeled robotic platform.**

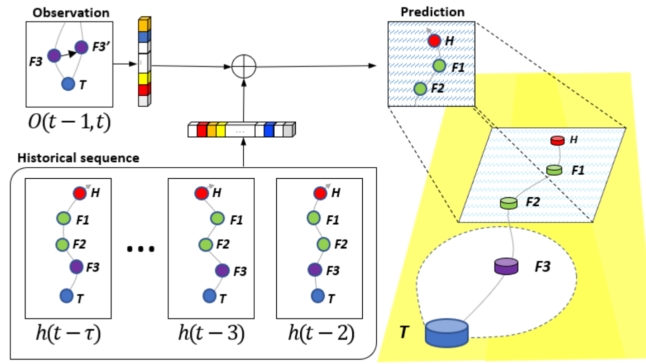

Fig. 1: Illustration of our proposed pipeline in a snapshot example of 5 robots at time t. Each robot has a limited view and a motion rule dependent on its neighbors except the *Head* robot leading the team at the front. *Tail* uses recent observations on its neighbor, *Follower 3*, which is denoted 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.

I. Introduction

In multi-robot systems including swarms, each robot typically has the capability to observe only a subset of its team members to determine its next action according to relatively simple motion rules. Consequently, these multi-robot systems are fundamentally distributed, having long-term outcomes that are all coupled together but with no simple whole-team coordination mechanism that can be taken for granted. One approach to facilitating coordination in such distributed systems is to establish a distinguished leader with a singular influence on all team members, e.g., [3]–[5]. However, if others in the team in potentially influential positions could indirectly infer the state and intentions of the leader, complementary actions could extend the leader's direct influence and improve the performance of the collective. For example, recognition by one robot at the end of a formation of a non-trivial and informative formation deviation by a robot at the other end could trigger motions that allow *both* robots to work together to appropriately move the collective [1].

In our previous work [1], we used a machine-learning method to solve the RTL problem where a robot (*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. Because each robot has a limited sensor radius and a relatively simple motion rule that depends on the position of its nearest neighbors, the *Tail* has to be able to learn the regularity of the observed motions of its neighbor to finally infer the poses of all other robots. We introduced a repetitive prediction scheme to use predictions about nearer teammates to make predictions on farther ones until the prediction reached the *Head* robot at the other end in line formation. In a multi-robot simulation, we showed the feasibility of using the method in an example caging scenario in which the *Tail* could recognize the early stages of a caging action of *Head* and promote a proactive maneuver to better assist in coordinating the team to quickly enclose an encountered object in the environment.

Figure 1 illustrates our proposed pipeline in which a deep neural network is to learn to synthesize the observations of its neighbor with knowledge about historical positions of all the teammates. As shown in Fig. 2, an LSTM layer is deployed to encode the historical sequence input, which could learn robot dynamics and the probable evolution of the team shape over time under physical constraints. Furthermore, the learned sequence encoding could help filter out impossible solution candidates that the model might produce if it only utilized the most recent observations on the neighbor, as in our previous work [1]. In addition to the more powerful deep-learning pipeline, we also make use of a more realistic robotic platform than in our previous work. Previously, we conducted all demonstrations on computer simulations, but

*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

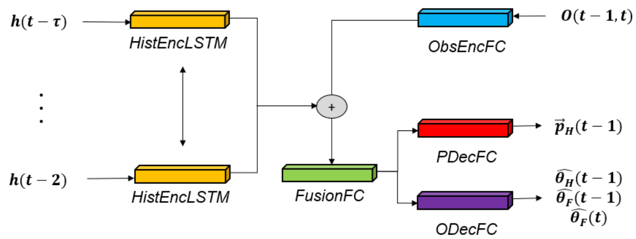

Fig. 2: Structure of our proposed deep neural network. This is an snapshot example when applied to a focused module of 3 robots, called *Tail, Follower, Head* within it, at time t. The Encoder-Decoder structure encodes 1) historical positions and orientations of *Follower* and *Head* until $t-2$ and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

in this work, we implement our approach in a realistic and reproducible physical environment realized by a widely available two-wheeled robotic platform, *Thymio* [6].

This paper is organized as follows. 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 *IPY-Net* method 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

In this section, we discuss results from the multi-robot systems literature similar to ours and elaborate on the distinct contribution of our work. We conclude this section with an elaboration of the differences between our results here and our previously presented work.

A. Cooperative localization and tracking

Although the RTL problem is superficially similar to other known cooperative localization and tracking problems [7]–[11], such as cooperative SLAM, it is a distinct from general robotic localization. In RTL, the robot does not execute predictions on its own location but on its teammates using accessible information. Moreover, in contrast with cooperative localization approaches, robots in RTL are assumed to be communication free and thus not allowed to communicate with other members during the prediction of positions. Hence, in lieu of direct signalling, an underlying assumption of the RTL problem is that the robot behaviors are correlated with the state of the environment around them and thus contain cues about the state of their neighbors. In this sense, state observers in a networked robotic system are more similar to RTL than robotic localization [12], [13], but RTL does not depend upon knowledge of the structure of the underlying robotic controllers and does not require that robots move according to simplistic, analytically tractable dynamical models. Furthermore, we emphasize RTL solutions focused on scaling from training with small teams to implementation in potentially much larger and possibly variably sized teams.

B. Group state recognition

The RTL problem aims to infers the positional state of an 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, Brown and Goodrich [14] as well as Berger et al. [15] show that local interactions of robots within a swarm can be used to classify swarm-level macroscopic structures, such as particular swarm-level shapes like *flock* and *torus*. In contrast, our work is to estimate the pose of robots themselves, which are the microscopic elements of the multi-robot team. Consequently, we make inferences on a space with far more degrees of freedom and require a more powerful regression model.

C. Behavioral cue interpretation

In the RTL problem, the pose of remote robots is to be inferred from the motions of nearby robots performing otherwise nominal behaviors. This approach allows information to flow around a multi-robot team without traditional communication modalities for explicit signalling, such as radio communication. Motivated by similar constraints on reducing the use of these modalities, Novitzky at el. [16] and Das et al. [17] showed how robots performing a special behavior, similar to a “waggle dance” of honeybees [18], could convey information visually or mechanically to remotely observing robots. Their approaches are different from ours in that we do not require robots to deviate from their normal behaviors for the purposes of explicit communication; we infer positions only from the latent information in nominal robot behavior and interactions.

D. Robot dynamics learning

Byranvan and Fox [19] proposed a deep learning approach to predict the next visual frame given both a current visual frame as well as knowledge of a force acting on an object within the frame. One of the motivations for this work was to understand the dynamics of robotic arms and the relationship with control commands possibly executed. In the RTL problem described in Section I, the *Tail* robot may have accumulated a global shape of robot team over time, and it has to be able to predict the future formation as a new observation on its neighbor is provided, which could be viewed as gaining knowledge of an applied force on the team. Such a similarity inspired the architecture of our neural network model, but Byranvan and Fox 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 positional information about the nearest neighbor is only loosely analogous to the perfect knowledge of force used in the frame-prediction example. Consequently, our approach is a significant deviation from the one proposed by Byranvan and Fox [19].

E. Contribution beyond past work

We first presented the RTL problem in our previous work [1]. To solve the problem 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 scaling the estimation capability from training on small teams to implementation in larger teams without further training. We use the same repetitive prediction scheme here; however, our focus is now on improving the regression engine to broaden its applicability from simulation to physically implemented robotic teams using commercially available, off-the-shelf robotic platforms.

III. RTL Problem & System Design

Here, we build on the problem and approach we introduced in our previous work [1]. In particular, we consider a team of $n \in \{3, 4, \dots\}$ robots designed to move in a line formation. Each robot maintains a programmed proximity with both the robot ahead of it and behind it, except for the *Head* robot that leads the convoy and the *Tail* that only follows its single neighbor ahead of it. Each robot in between *Head* and *Tail* are referred to as *Follower* i, where $i \in \{1, 2, \dots, n-2\}$ represents the position relative to the *Tail* (i.e., *Follower* 1 is closest to the *Tail*).

Every robot has the same sensory and motion capabilities and constraints, although they may take different actions according to motion rules depending on their roles. This leads to the *Follower* and *Tail* robots to behave differently 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 teammates from the view of *Tail* only using locally observable information of the position 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 reliably with the *Tail* robot, possibly via global communication, but continued information sharing is unavailable after time τ thus requiring *Tail* to use this localization technique to extrapolate from the previously known reliable positions.

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

$$\{\vec{p}_{r@t}, \theta_{r@t}\} \tag{1}$$

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 $t \leq \tau$. The RTL problem is thus, at each time $t > \tau$, to observe the position of F_1, $\vec{p}_{F_{1@t}}$, and use all available information to predict the pose set in Eq. (1) for time $t' > \tau$.

IV. Method

Here, we first briefly review our scalable 3-to-many prediction approach for training with a small team and implementation on a larger team without additional training [1]. Then in Section IV-B, we present our improved regression approach that uses not only current nearest-neighbor observations but also sequences of past team formations.

A. Scalable prediction approach

The scalable RTL implementation we previously introduced [1] iteratively applies predictions of farther and farther robots within focal 3-robot teams, starting with the nearest *Follower* robots and eventually leading to prediction of the *Head*. In particular, *Tail* initially uses all available information about *Follower* 1 at time τ and $\tau + 1$ to predict $\vec{p}_{F_2@\tau}$ of *Follower* 2. Then, at $\tau + 2$, *Tail* may repeat this approach centered on *Follower* 2 to predict $\vec{p}_{F_3@\tau}$ of *Follower 3*. Thus, after $n-1$ iterations of this approach, the *Tail* will eventually have an estimate of the position $\vec{p}_{H@\tau}$ of *Head* at time τ.

Because each inference is only applied in 3-robot teams (i.e., the *Tail*, a focal *Follower*, and the robot immediately ahead of that *Follower*), the predictor can be trained simply with a 3-robot team and used for a larger size time without additional training for the larger-team case. However, this iterative modular-prediction approach brings about a time delay in making a prediction for a far robot. Furthermore, any estimation errors tend to accumulate, making the use of this approach limited either to short chains of robots or short estimation time periods. Thus, in the next section, we describe a regressor significantly improved over our previous version [1] that increases the feasibility of using this approach in realistic multi-robot teaming scenarios.

B. Improved 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:

$$Y = f(X_{obs}) \tag{2}$$

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:

$$o = f_{obs}(X_{obs}) \tag{3}$$

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.

	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.

A historical sequence of poses is encoded by a bi-directional LSTM layer [20]:

$$h = f_{hist}(X_{hist}) \tag{4}$$

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.

$$\begin{aligned} Y_p &= g_p(\phi(o,h)), \\ Y_\theta &= g_\theta(\phi(o,h)) \end{aligned} \tag{5}$$

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 Eq. (5) could be implemented by any type of layer. During our experiments, we built a FC layer of size d to find a nonlinear relationship between the input features and achieved a satisfactory performance.

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 [21] 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.

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:

- *2X 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

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

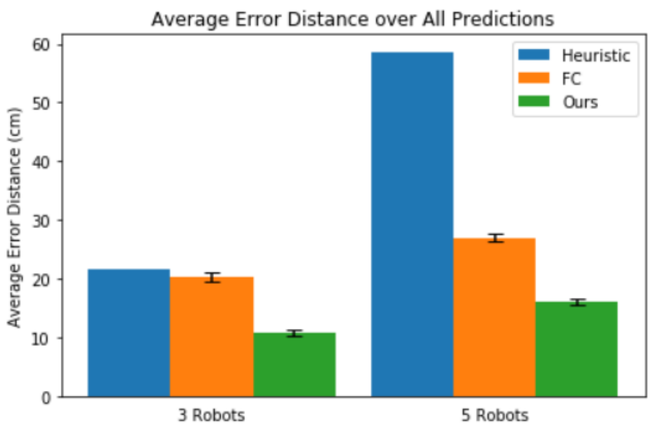

Fig. 3: Average accumulated error of each model in two different sizes of robot team. For each of machine learning methods, the mean performance of 5 separate training sessions is reported with a error bar to visualize the performance variation.

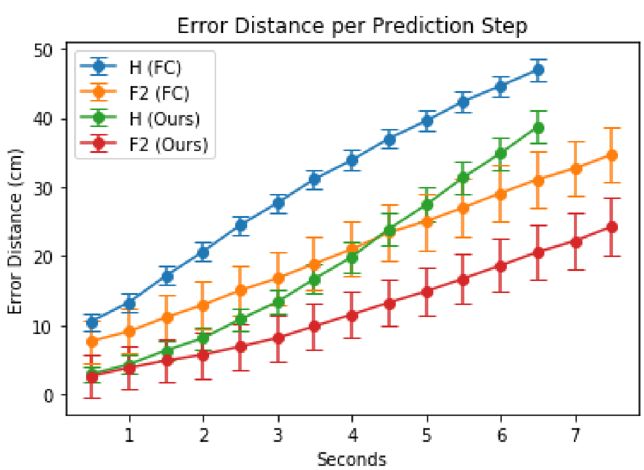

Fig. 4: Average step-wise error for different target robots in 5-robot team. For each model, all the prediction errors at different time steps are averaged for a specific robot. The error bars represent the standard deviation of 5 separate models in terms of the performance metric. For the sake of visualization,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

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 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.

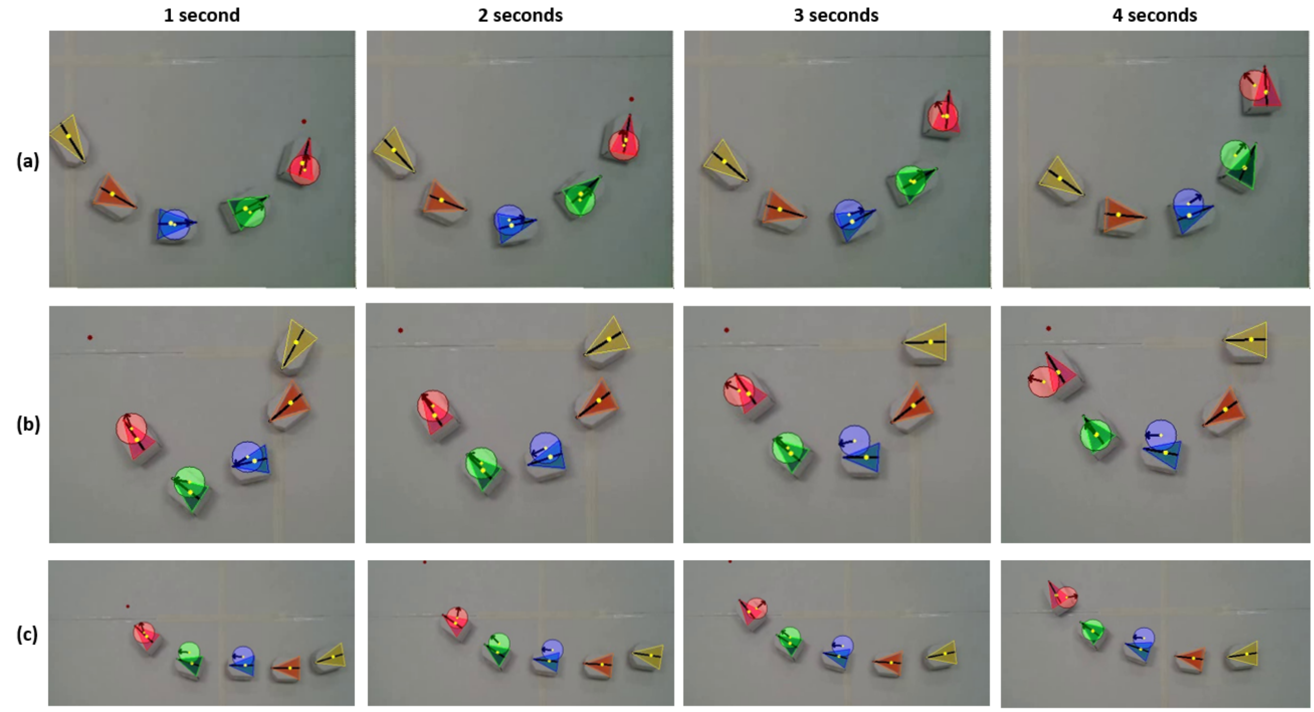

Fig. 5: Sample video frames with prediction results of our proposed model for three different instances of 5 robot team. Each row presents four frames of an instance for first 4 seconds. The yellow robot is *Tail*, the pink is *Head*, and in-between are *Follower* robots. The colored circles are predicted 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.

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.

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] 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.

[4] 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.

[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] B. Grocholsky, V. Kumar, and H. Durrant-Whyte, "Anonymous cooperation in robotic sensor networks," in *Proc. of the AAAI-04 Workshop on Sensor Networks*, 2004.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] L. Xiao-Yuan, H. Na-Ni, and G. Xin-Ping, "Leader-following formation control of multi-agent networks based on distributed observers," *Chinese Physics B*, vol. 19, no. 10, 2010.

[13] G. Antonelli, F. Arrichiello, F. Caccavale, and A. Marino, "A decentralized controller-observer scheme for multi-agent weighted centroid tracking," *IEEE Transactions on Automatic Control*, vol. 58, no. 5, pp. 1310–1316, 2012.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.