I Picture You: Imaginative localization on remote teammates

Taeyeong Choi, Sehyeok Kang, and Theodore P. Pavlic

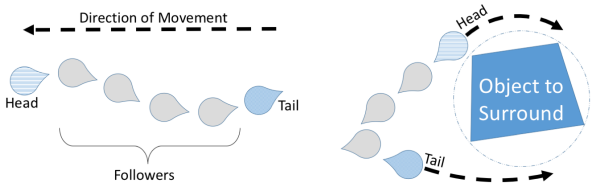
***Abstract*— We propose *IPY*, as an extension of [1], to tackle the Remote Teammate Localization (RTL) problem where a robot embedded in a multi-robot team is to predict positions of all other teammates when each robot has a limited sensor radius and a relatively simple motion rule with dependency on the position of its nearest neighbors. The previous work presented feasibility of a scalable machine learning approach for the predicting robot to proactively assist a team behavior in caging scenario. In this work, *IPY* is designed to improve scalability by taking into account physical constraints on the team formation as well as producing probabilistic predictions in the form of image through deep learning pipeline. Furthermore, while we previously relied only on computer simulations, all experiments in this work are performed in a physical robotic platform, *Thymio*, to demonstrate the performance gain in more realistic environments.**

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 [x]. This implies that if a robot has an ability to recognize 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 embedded in a multi-robot team is to predict positions of all other teammates using only 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 in line formation, the predicting robot has to be able to learn the regularity of the observed motions. Through simulated experiments, we showed the feasibility of the idea especially in caging scenario in which the predicting robot could recognize the team behavior and use a proactive maneuver accordingly.

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



(a) Multi-robot chain (b) Caging scenario

Fig. 1: Motivating examples. In (a), heterogeneous robots move in chain formation from right to left. In (b), the *Tail* robot breaks out of chain formation to more rapidly encircle an object encountered first by *Head*.

In this work, we propose *IPY* to significantly reduce prediction errors solving the RTL problem, which would allow to gain stable performance even with more robots added. *IPY* runs a deep neural network as a backbone, first to consider physical constraints on the sequential change of team formation and, second to use a probabilistic representation in the form of image for obtained observations and new predictions. Specifically, every observed or predicted position is encoded as an image in which the summation of all values must be 1, which can be viewed as a probability distribution of position of the robot. Hence, *IPY* learns how to synthesize inputs with some uncertainty to produce a probabilistic prediction outcome in end-to-end fashion. Furthermore, a recurrent layer is deployed in *IPY*, which enables the model to take into account possible evolution of team formation based on recent history and reduce the searching space for predicted solution. With all combined, historical probabilistic (soft) formations are used as input per prediction providing more flexibility to the predictor in selecting salient features and finally leading to more robust performance even when the input is noisy to some level.

We conduct all experiments on a physical robotic platform, *Thymio* [2], to demonstrate the improvement in more realistic environments in contrast to the previous work that relied only

*This work was not supported by any organization

T. Choi, S. Kang, and T. P. Pavlic are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA {tchoi4, tpavlic, aricha}@asu.edu

on computer simulations. All codes are available online[1], and supplementary videos are submitted as well.

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*, in Section IV, and Section X 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

III. Problem Description

IV. Method

V. Experiments

A. Data collection

B. Results

VI. Discussion & Future work

In networked multi-robot teams, coordinated behaviors typically emerge from interactions among adjacent robots with limited sensor range, simple motion rules, and no persistent connection to any centralized command and control authority. In principle, such a decentralized approach naturally scales well to real-world problems where a large number of robots must be deployed over relatively large areas where constant global communication is impractical. Significant effort has been focused on the design of decentralized control algorithms for multi-robot systems [3]–[15]. However, approaches that make use of high levels of communication among agents present scalability challenges, and approaches that eschew communication often achieve less than ideal coordination as a consequence.

If a single robot could, with little explicit communication, gain awareness of the status of other very remote agents in the team or awareness of some macroscopic state of the group, then a new class of highly scalable coordinated behaviors would be possible. Robots in one location could perform behaviors that complement the actions of robots in another location in a way that is currently only possible using levels of signalling that are particularly burdensome in large-scale groups. It is now possible to shift from explicit to implicit communication, from what behavioral ecologists call "signals" to "cues" [16], because of the recent increase in computational power available on modern small robotic platforms. Whereas the platforms of the past compensated for little local computational power with communication to coordinate with other agents, robotic platforms of the future can rely on high levels of local computational power to reduce the need for communication that may be energetically costly or physically infeasible.

Toward this end, we propose a machine-learning approach for an individual robot in a multi-robot team to estimate the position of its team members using only local information about the movements of a single nearby robot. This approach capitalizes on the regularity of the simple behavioral rules used by members of the multi-robot team; that is, the variability in observed behaviors of nearby team members tends to be a strong predictor of changing environmental conditions elsewhere in the team because the robots operate in a very predictable way under nominal conditions.

To demonstrate our approach in this paper, we consider a simple scenario where a heterogeneous multi-agent system like the one in Fig. 1a maintains a chain formation and consists of three types of nonholonomic mobile robots: *Head*, *Follower*, and *Tail*.

Head is a robot at one end of the chain formation, *Tail* is another at the distal end, and between them is a string of *Follower* robots. Our goal is to train artificial neural networks (ANNs) [17] on *Tail* to learn how to utilize locally observable information – the position of its nearest neighbor – to estimate the position of every robot in the system even including the *Head* when it is outside of the sensor range of the *Tail*. The major strength of our approach is that training of the ANNs on *Tail* requires only a 3-robot chain, and the resulting ANN can be applied by *Tail* to fix the positions of robots in longer chains simply by applying the ANN repeatedly in a recursive fashion. We validate our approach in a simulation of robots with realistic dynamics in the presence of noise and little-or-no communication availability. We show that the approach can facilitate coordinated behaviors like the one in Fig. 1b.

This paper is organized as follows. In Section II, we review related work and our contribution beyond it. Next, we formally define our system and its motion rules in Section VIII. Then, in Section IX, we describe our approach to training ANNs for the estimation tasks. Our approach to both estimation and the encirclement solution is validated in Section X on a simulation framework developed for a relatively new multi-robot testbed, the *Robotarium* [18]. Finally, we give concluding remarks in Section XI.

In the context of the example scenario presented in this work, our goal is to enable an individual robot, *Tail*, to infer the positions of its teammates in real time within its frame of reference. This task is different from traditional localization problems where a robot fixes its position given its action and measurement models [19]. Moreover, our problem differs from scenarios with multiple robots cooperating to track a target's state in a distributed way [20]–[22] because we impose a hard constraint on information sharing between robots, and all robots other than *Tail* do not perform any localization inferences of the pose of others in the team.

Our work is superficially similar to work by Novitzky et al. [23] and Das et al. [24] where robots visually signal to others like the "waggle dance" of honeybees [25], [26]. Strictly speaking, these behavioral communication channels still represent explicit signalling between agents. In the study of animal communication, such signalling evolved when the benefit of the explicit message passing outweighed the implementation costs [16]. Thus, in robotic scenarios where explicit signalling is notably costly, it may be useful to train robots to be responsive to more subtle cues that covary with

[1]http://www.github.com/ctyeong

environmental information; in that perspective, we are not building a motion-based signalling system between robots but achieving useful inferences about the group based on behavioral cues and not costly, explicit communication.

There are stronger similarities between our work and methods from human–robot interaction where robots learn to interpret and respond to *cues* (as opposed to explicit *signals*) from human teammates. For example, Maeda et al. [27] designed a framework with which a robot can infer how to appropriately assist a human partner assembling an item after learning from data collected from two humans completing the task. Thus, the robot can infer the normally cryptic intention of the human through non-verbal cues and can then respond appropriately. Our approach is similar, but our agent learns how to respond appropriately to cues from a complex system of other robots.

In our running example of a moving chain formation, the formation can be driven by two heterogeneous leaders, *Head* and *Tail*. A similarly structured robotic system was designed by Elamvazhuthi and Berman [14] consisting of a long string of holonomic point robots that are terminated by two *leader* agents. While the interior robots in the chain perform simple, wave-like motion rules based on position information from neighbors only, the two leader robots have pre-programmed trajectories that are guaranteed to drive the interior robots to desired locations along a formation. In contrast, our goal is to infer remote properties of the multi-robot team based on local information even when the individual-robot motion models are too complex to be analyzed mathematically. Thus, our methods generalize to more realistic robotic motion rules – such as nonholonomic robots limited to two-dimensional planar motion with collision avoidance.

Our motivating application example, caging, is a popular scenario in the robotics literature. Wilson et al. [9] developed a stochastic robotics method for boundary coverage meant to mimic the collective transport behavior of ants, and Derakhshandeh et al. [10] developed an amoeba-inspired distributed algorithm for coating objects on a specially defined grid. Each of these applications makes use of identically controlled agents that come to equilibrium in some desired formation, often with the assumption of a simplistic motion model. In contrast, we focus on heterogeneous multi-agent systems where some agents must gain global situational awareness in order to make decisions about whether and when to switch from certain behavioral modes to certain other modes. In principle, our method could be used in combination with these other techniques so that agents can determine when to switch to other behaviors after the desired configuration has been reached.

VII. Problem Description

VIII. System Design

We consider a multi-agent system consisting of n nonholonomic mobile robots that move within a chain formation in the plane of $\mathbb{R}^2$. At all times, each one has a pose denoted by a three dimensional vector (x, y, θ) where x and y specify its position and θ its orientation. Every robot is controlled by a decentralized controller and can only detect the distance and angle to another robot or other object within a limited sensor radius. Due to this limitation, each robot cannot localize in the global coordinate system but can use their own frame of reference to represent observed objects. The frame of reference will be explained in detail in Section IX.

Each robot type from Fig. 1a has a distinct motion rule.

- The *Head* moves independently in the given space.
- The $n-2$ *Follower* robots continuously update their position to maintain a constant distance between their two nearest neighbors.
- The *Tail* either follows its single neighbor at a constant distance or moves independently.

So as the *Head* moves, it pulls a nominally rigid chain of robots behind it, but *Tail* can switch to moving independently of the others, causing the chain to be pulled at both ends.

Formally, we define the set of robots $\mathcal{R} \triangleq \{1, 2, \ldots, n\}$ where robot n is the *Head*, robot 1 is the *Tail*, and robot $i \in \{2, \ldots, n-1\}$ is a *Follower*. Every robot $i \in \mathcal{R}$ is assumed to abide by nonholonomic unicycle kinematics of the form

$$\dot{x}_i = v_i \sin(\theta_i), \qquad \dot{y}_i = v_i \cos(\theta_i), \qquad \dot{\theta}_i = \omega_i$$

where vector $\vec{p}_i \triangleq (x_i, y_i)$ is the robot's Cartesian position, θ_i is the robot's orientation, v_i is the linear velocity of the robot, and ω_i is the angular velocity of the robot. The *pose* $\vec{r}_i \triangleq (x_i, y_i, \theta_i)$ of a robot $i \in \mathcal{R}$ is a vector containing its position in the plane and its heading orientation. Thus, by adjusting ω_i and v_i, the nonholonomic robot can be driven like a unicycle over the plane. We use this model for simplicity. However, the data-oriented machine learning approach described in Section IX should be applicable to a wide range of other kinematic and dynamic models as well.

To implement the chain-following formation shown in Fig. 1a, we use fully actuated, holonomic kinematics in the Cartesian plane to generate *reference trajectories* for ω_i and v_i to track within the more realistic nonholonomic unicycle kinematics. As described above, the underlying robots attempting to track these reference behaviors will do so with some error because they are nonholonomic. So the actual behavioral rules are a composition of simplistic holonomic kinematics with the more realistic nonholonomic kinematics of the simulated robots. The particular coordinate transformation and nonholonomic constraints are implemented within the controller in the simulator for the multi-robot testbed that we use to train and test the system (see Section X), and so we only present the fully actuated, target motion rules here. In particular:

- The *Head* robot approximates the motion rule:

  $$\dot{\vec{p}}_n = \vec{T}_h - \vec{p}_n$$

  where $\vec{T}_h$ is some target Cartesian position in the space. Thus, the *Head* robot is always moving toward some target point set elsewhere in its control hierarchy.
- Each *Follower* robot $i \in \{2, \ldots, n-1\}$ approximates:

  $$\dot{\vec{p}}_i = k_f((\|\vec{p}_i - \vec{p}_{i-1}\| - d)(\vec{p}_{i-1} - \vec{p}_i) + (\|\vec{p}_i - \vec{p}_{i+1}\| - d)(\vec{p}_{i+1} - \vec{p}_i))$$

where k_f is a scalar that scales the difference between the desired spacing around robot i to a desired velocity that should, if geometrically possible, return the *Follower* to a distance of d away from its two neighbors.

- The *Follower* normally behaves like a *Follower* with one neighbor, as in:

$$\dot{\vec{p}}_1 = k_t((\|\vec{p}_1 - \vec{p}_2\| - d)(\vec{p}_2 - \vec{p}_1)).$$

However, it will switch to the *Head* motion rule when it detects that *Head* has detected some object to encircle.

Thus, the challenge to be addressed in Section IX is how the *Tail* can use regularities in the patterns of interactions that emerge from such simple motion rules to predict the position of *Head* using only knowledge about the position of robot 2, the *Follower* closest to *Tail*.

IX. Estimation

Our goal is to design an essentially communication-free algorithm that enables *Tail* to estimate the position of each distant robot $i \in \{3, \ldots, n\}$ in the team using only locally sensed information about robot 2, the *Follower* immediately ahead of it. Furthermore, the same algorithm should be able to be used for a wide range of team sizes. The scalability of our approach comes from our use of modular artificial neural networks that are applied recursively as opposed to a monolithic ANN tailored for a particular team size.

A. Supervised Learning of Artificial Neural Networks

Toward accomplishing our goal, we first consider the simple 3-robot case ($n = 3$) where robot 3 is the *Head*. On *Tail*, we use backpropagation [28] to train a set of ANNs, $\{ANN_x, ANN_y, ANN_\theta\}$, as regression models that can predict the Cartesian coordinates of robot 3 as well as the orientation of robot 2 using only the sensed coordinates of robot 2. They are trained with pose data collected at discrete time instants during interactions among the three robots in simulations where robot 3 takes arbitrary motions.

From this point forward, we make use of subscripts form $i@\tau \rightarrow j@t$ to denote that a coordinate element of robot i at time τ is represented in the local coordinate frame of robot j at time t. Each ANN is a multi-layer perceptron employing three layers – one input layer, one hidden layer, and one output layer. The hidden layer utilizes 12 nodes, each of which performs computations that take values from all nodes of the input. Each hidden node uses the logistic, sigmoidal activation function whose output ranges continuously from 0 to 1. For regression purposes, one node set in the output layer has a linear activation function taking outputs of all hidden nodes as input. The major difference in the structure of the three ANNs is in their input layer. The ANN_x, which is used to predict $\vec{x}_{3@t\rightarrow 1@t}$, takes six inputs consisting of:

- $\vec{r}_{2@t\rightarrow 1@t}$, the relative pose of robot 2 in frame of robot 1 at time t
- $\vec{p}_{2@t+1\rightarrow 1@t}$, the relative coordinates of robot 2 at time $t+1$ in frame of robot 1 at time t
- b, a scalar bias

In training, the ANN_y that estimates $\vec{y}_{3@t\rightarrow 1@t}$ uses these same six inputs plus $\vec{x}_{3@t\rightarrow 1@t}$. Similarly, the ANN_θ for $\vec{\theta}_{2@t+1\rightarrow 1@t}$ takes the seven inputs of ANN_y as well as $\vec{y}_{3@t\rightarrow 1@t}$. For the $n > 3$ case, we explain how coordinate transformations allow these ANNs to be applied recursively to predict all robot positions in Section IX-C.

B. Estimation Process

For estimation with $n = 3$, the ANNs on *Tail* can estimate $\vec{p}_{3@t-1\rightarrow 1@t-1}$ and $\theta_{2@t\rightarrow 1@t}$ at any time t using only the locally sensed position of robot 2 (assuming the initial orientation of all robots are known). Throughout the estimation process, the $\vec{x}_{3@t\rightarrow 1@t}$ input to ANN_y and ANN_θ will be estimated by ANN_x, and the $\vec{y}_{3@t\rightarrow 1@t}$ input to ANN_θ will be by ANN_y. For example, at the second time instant $t = 1$, the *Tail* starts to estimate the position information of robot 3 for time 0 and thus also the orientation of robot 2 for time 1. In other words, by utilizing the sequential observations of $\vec{p}_{2@0\rightarrow 1@0}$ and $\vec{p}_{2@1\rightarrow 1@0}$ and the already known initial orientation, *Tail* can estimate $\vec{p}_{3@0\rightarrow 1@0}$ as well as $\theta_{2@1\rightarrow 1@0}$, which can be transformed to $\theta_{2@1\rightarrow 1@1}$ for future computation. Similarly, for every time instant t, *Tail* can estimate $\vec{p}_{3@t-1\rightarrow 1@t-1}$ and $\theta_{2@t\rightarrow 1@t}$.

C. Scalable Estimation

As described in Section IX-A, our approach trains three ANNs for the $n = 3$ case and applies them recursively to estimate the positions of robots in the $n > 3$ case. An alternative approach would be to apply ANNs trained on data for cases where $n \geq 3$. However, training under that approach would be complicated by a relatively large state space, and a new training phase would be required whenever the number of deployed robots changed.

For scalability and to simplify the training process, we take advantage of a coordinate transformation. Assume that $n > 3$ and *Tail* has processed the estimation until $t = 2$ as described above. *Tail* can transform the estimates for robots 2 and 3 into the reference frame of robot 2 to predict $\vec{p}_{4@0\rightarrow 2@0}$; that is, the *Tail* can act as if it is robot 2 predicting the position of robot 4. The predicted position also can be transformed back to the natural reference frame of *Tail* for it to gain the global awareness. By such a repeated computation, *Tail* can finally obtain estimates for the full pose information up to robot $i \in \{3, \ldots, n-1\}$ and thus also the position information up to robot $(i+1)$ in finite time. Estimates of a far robot $i \in \{3, \ldots, n\}$ will be delayed, but this delay scales only linearly with the distance from the *Tail*, which is no worse than would be expected with direct communication of position in a distributed, ad hoc wireless network. Thus, the ANNs trained in the 3-robot case are a general purpose tool for inferring pose information for arbitrary length chains.

X. Experiments

Experiments were conducted in the simulation platform designed for use with the *Robotarium* [18], a remotely accessible swarm-robotics testbed consisting of large numbers of *GRITSBot* robots [29]. The *GRITSBot* is a nonholonomic,

differential drive robot that is modeled explicitly within the Robotarium simulator, available in both MATLAB and Python. In principle, robotic implementations tested within the simulator can easily be ported to the actual *Robotarium* multi-robot testbed environment and executed remotely. For our simulated case, the rectangle-shaped arena has width 1.2 and length 0.7, and the diameter and sensor radius of each robot are 0.03 and 0.12, respectively.

A. Training Procedure

In the learning phase, only three robots, one *Head*, one *Follower*, and one *Tail*, are involved as explained in Section IX-A. They are allowed to interact with each other according to the built-in motion rules described in Section VIII. During the interactions, all pose information is recorded every time step to later train the three ANNs on *Tail*. For efficient learning, *Head* is designed to choose its actions from a predetermined set of linear and angular velocity values, leading the team to move forward or make different curves.

In addition to the velocity of *Head*, it is important to vary the motion-rule parameter k_t on *Tail* as well. The ANN is trained in a 3-robot scenario where the motion of *Tail* is not constrained by any robots that follow its own motion; however, when applying the ANN recursively in the $n > 3$ case as described in Section IX-C, the fictitious *Tail* robots in each recursion step will be more constrained than a true *Tail*. By training with different values of k_t, the ANN is able to anticipate this reduced flexibility.

Our training data is generated from the 20 combinations of the five *Head* velocities with four k_t parameter values. Each of the 20 treatments was allowed to run for 800 time steps, and the resulting trajectories were replicated five times so that Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.001^2)$ could be added to each datum to ensure stable performance of the ANNs even with noise. All ANNs were trained for 500 epochs with a learning rate of 0.05 and momentum 0.2.

B. Estimation Phase

To examine the feasibility and accuracy of our estimation method in various settings, we first show position estimation results from experiments where the *Tail* only has access to local information about its immediate neighbor. Next, a different scenario is introduced where the ANN is used to interpolate between slow updates from periodic communication of other robot positions. Also, we test the effect of noise on the estimation performance.

As a measure of estimation accuracy, the *Euclidean distance* between the actual position of robot $i \in \{3, ..., n\}$ and the corresponding estimate in *Tail*'s perspective is calculated every time step. Estimates with an error below the robot diameter of 0.03 are considered to be accurate. Moreover, the testing duration $1,600$ is twice the training duration, which ensures that the ANN is tested with novel challenges.

To differentiate between followers during the experiments, we use the naming convention that *Follower 1* is adjacent to *Tail*, *Follower 2* is adjacent to *Follower 1*, *Follower 3* is adjacent to *Follower 2*, and so on. In other words, *Follower* k refers to the follower that is k robots ahead of *Tail*.

1) Estimation without Communication: Figure **??**a plots the accuracy in the simplest case where the *Head* leads the other five robots to travel straight. The estimation error is larger for farther robots as they inherit some error from estimations of nearer robots. Yet, every error is below 0.03 and thus within our accuracy tolerance through $1,600$ steps.

We also tested our model with a curve formation that is more complex than a straight line. Figure **??**b shows the performance for the case where the *Head* leads a team of five robots to make a curve. The overall error is larger than the previous case, but every error remained in the acceptable level. The error peaked near 400 time steps when the curve's angle was at the maximum during the execution. However, the error decreased after that peak as robots recovered from overshooting the turn and returned to their straight trajectory.

Then, we explored a case of six robots, one more than the previous case, while making a turn. As shown in Fig. **??**c, the overall performance was worse than the two previous cases, as the estimate for *Head* was inaccurate between 350 and 750 steps. Though the estimation for the other robots (*Follower 2*, *3*, and *4*) was accurate, the accumulated error caused unacceptable estimation error for *Head*. Nevertheless, the performance pattern looks similar as the previous experiment in that the peak error occurs near 500 steps and declines to below the 0.03 accuracy threshold as the robots return to a more predictable straight-line trajectory after 750 time steps.

2) Intermittent Communication: Although we have focused on scenarios where there is no robot-to-robot communication, we also tested the effect of intermittent communication providing limited information at a relatively slow rate. In reality, we could expect that robots can, in principle, broadcast their own pose data but do not frequently do so due to technological limitations or to save on communication cost. For instance, an aerial robot may only be able to share its pose periodically with a single base station that can relay it later to other robots that periodically fly by.

In our case, every 150 time steps, *Tail* is allowed to access to actual position and orientation information for estimation process. Between messages, *Tail* must use the ANN to infer the pose of others, but actual pose is known intermittently.

Figure **??**d illustrates the improvement by the periodic communication, when six robots are turning a curve as previous setting. Just after the communication update, the error rapidly decreases toward zero due to the introduced information from the communication burst. These periodic decreases in error allow for longer periods of time when the total error is kept under the accuracy threshold.

3) Reliability Demonstration: In this set of experiments, we tested the case where measurements were corrupted with noise. Specifically, we added noise $\epsilon \sim \mathcal{N}(0, 0.001^2)$ to every data instance that is fed to the ANNs in order to examine if the scenarios that already showed successful results would lead to the similar performance in this situation as well. The distribution of noise was chosen to simulate that 99% of measurements would have an error off the true value by ≤ 0.003, 10% of the diameter of simulated robots.

Figure **??**e shows a noise-corrupted version of the scenario

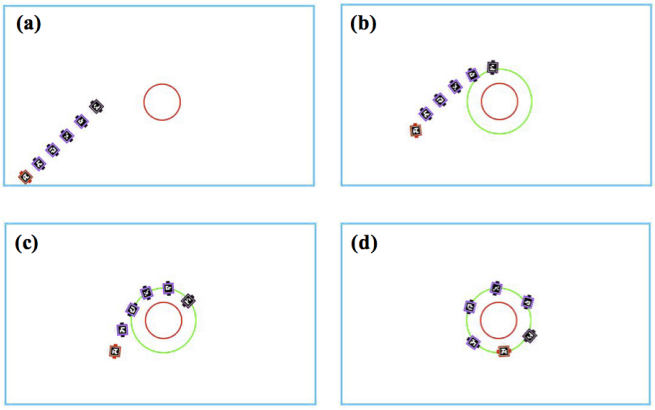


Fig. 2: Applicable caging scenario for a team with a *Tail* that can infer changes in the behavior of the *Head* from remote.

evaluated earlier in Fig. **??**b. Despite the additive noise, the ANNs still perform reliably well; every estimate is within the accuracy threshold over time. On average, the error has a similar shape to Fig. **??**b.

The case in Fig. **??**f was configured as in the periodic communication case of Fig. **??**d with additive noise. The noise-corrupted results vary only slightly from the noise-free results, and so our approach can perform well even in a noisy environment.

4) Application Scenario: Finally, we return to the motivating multi-robot team application that we introduced in Fig. 1b. The multi-robot team is to surround an object after being detected by *Head*. Empirically, we found that caging cannot be accomplished in a team of only *Head* and *Follower* robots because the *Follower* robots have no attraction to the object and thus will not circle it under their simple motion rules. By estimating the behavior of *Head*, the *Tail* can determine when the *Head* has detected an object and roughly where the object is. At that point, the *Tail* can move in a direction that ensures that the *Follower* robots (which are influenced by both their forward and rearward neighbors) are pulled toward the object.

In this scenario, we assume that *Head* and *Tail* can follow along with the surface of an object that they locally encounter in the environment. We also assume that *Follower* uses a potential-field-based controller [30] to avoid collisions when in close proximity to the object. Because *Head* will detect the object before *Tail*, we will study the case where *Tail* detects that *Head* has begun a caging maneuver that can be assisted by *Tail*. In that condition, *Tail* will begin to encircle the object in the reverse direction of *Head*. In the end, when *Head* and *Tail* are facing each other, they will switch to a *Follower* motion rule so that the caging formation converges to equilibrium "net" around the object, in which all the robots maintain the same distance to neighbors over time.

Figure 2 shows our implementation for 6 robots deployed for the caging mission on the simulator. We placed a virtual, circular object for the mission at the center with a circular boundary drawn around it indicating the desired final positions of the robots. The robot team starts searching from bottom left as shown in Fig. 2a. In Fig. 2b, *Head* has sensed the object and begins to move around it, and this motion is detected by *Tail* that starts to deviate from simply following its nearest neighbor. Figure 2c illustrates how the complementary actions of *Head* and *Tail* lead to an encircling maneuver, and Fig. 2d shows the final formation.

XI. Summary, Conclusion, and Future Work

We have shown *IPY*, a powerful approach to tackle the remote localization problem where a robot with limited sensors is to predict positions of all others only using observations about its nearest neighbor. *IPY* follow the fundamental spirit of [1] to apply a predictor trained for a small robot team to a chain of them in a larger team, since it can provide a scalable feature without re-training. The distinction in *IPY* is , however, to use a sequence of historical knowledge and make predictions based on imagination of robot states including position and orientation. We expected learning with history would help regulate the model on physical constraints on the team formation, and also the imaginative representation could contain certainty about the prediction, which would bring about more robust performance.

The experiment results were obtained using a realistic robotic platform to prove that our method outperforms not only the regressor introduced in [1] but also a general state-of-the-art neural network model that is able to use historical data. Especially, our model can present stable performance even when the input contains some errors which have occurred in previous prediction.

In future work,

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] 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.

[3] Z. Wang and V. Kumar, "Object closure and manipulation by multiple cooperating mobile robots," in *Proc. of the 2002 IEEE International Conference on Robotics and Automation*, 2002, pp. 394–399.

[4] N. Correll and A. Martinoli, "Modeling and optimization of a swarm-intelligent inspection system," in *Proc. of the Seventh International Symposium on Distributed Autonomous Robotics Systems (DARS 2004)*, Toulouse, France, 2004, pp. 369–378.

[5] L. U. Odhner and H. Asada, "Stochastic recruitment control of large ensemble systems with limited feedback," *Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 4, 2010.

[6] N. Napp and E. Klavins, "A compositional framework for programming stochastically interacting robots," *International Journal of Robotics Research*, vol. 30, no. 6, pp. 713–729, 2011.

[7] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[8] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[9] S. Wilson, T. P. Pavlic, G. P. Kumar, A. Buffin, S. C. Pratt, and S. Berman, "Design of ant-inspired stochastic control policies for collective transport by robotic swarms," *Swarm Intelligence*, vol. 8, no. 4, pp. 303–327, 2014.

[10] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, T. Strothmann, and S. Tzur-David, "Infinite object coating in the amoebot model," *arXiv preprint arXiv:1411.2356*, 2014.

[11] B. Yang, Y. Ding, Y. Jin, and K. Hao, "Self-organized swarm robot for target search and trapping inspired by bacterial chemotaxis," *Robotics and Autonomous Systems*, vol. 72, pp. 83–92, 2015.

[12] G. Valentini, H. Hamann, and M. Dorigo, "Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy tradeoff," in *Proc. of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, Istanbul, Turkey, May 4–8, 2015, pp. 1305–1314.

[13] G. Valentini and H. Hamann, "Time-variant feedback processes in colletive decision-making systems: influence and effect of dynamic neighborhood sizes," *Swarm Intelligence*, vol. 9, no. 2, pp. 153–176, 2015.

[14] 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.

[15] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: speed versus accuracy in binary discrimination problems," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 3, pp. 553–580, 2016.

[16] J. W. Bradbury and S. L. Vehrencamp, *Principles of Animal Communication*, 2nd ed. Sinauer Associates, Inc., 2011.

[17] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, 2015.

[18] D. Pickem, L. Wang, P. Glotfelter, Y. Diaz-Mercado, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "Safe, remote-access swarm robotics research on the robotarium," *arXiv preprint arXiv:1604.00640*, 2016.

[19] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] A. Dornhaus and L. Chittka, "Why do honey bees dance?" *Behavioral Ecology and Sociobiology*, vol. 55, no. 4, pp. 395–401, 2004.

[27] G. Maeda, M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in *14th IEEE-RAS International Conference on HUmanoid Robots*, 2014, pp. 527–534.

[28] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[29] D. Pickem, M. Lee, and M. Egerstedt, "The GRITSBot in its natural habitat – a multi-robot testbed," in *Proc. of the 2015 IEEE International Conference on Robotics and Automation*, 2015, pp. 4062–4067.

[30] R. C. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.