coupled. Also, the function may not be differential, which makes many mathematical optimization algorithms (e.g., quasi-Newton BFGS algorithm [64] or conjugate gradient algorithm [65]) infeasible.

- *Multi-objective* When encountering real-world problem, we usually need to deal with multiple objectives, which need to be optimized simultaneously. For example, when designing a car, the engineers need to consider the shape, performance of the engines, and more importantly cost. As some of the objectives (such as performance and cost) are conflicting, the designer needs to find a Pareto optimal solution [66], in which none of the value of the objective functions can not be increased without considering decreasing the value of other objectives. Evolutionary multi-objective optimization is an efficient technique for generating such Pareto optimal solutions for multi-objective optimization problems. For a review, see [66].

2.2.3.2 Evolutionary Robotics

Another application of evolutionary computation is evolutionary robotics, in which the controller and/or morphology of the robots are automatically generated. The user considers the robot as a whole and only needs to specify the criterion which is represented by a fitness function. The fitness function could be single-objective or multi-objective. The aim is to optimize (minimize or maximize) the fitness using evolutionary algorithms. The normal procedure of evolutionary robotics is as follows. First, an initial population of random controllers are generated. Each controller are represented by a chromosome. If the controller is a neural network, the chromosome is a vector of real numbers. Each controller was imported into the robot(s), and the robot's performance when performing various tasks is measured and evaluated using the pre-defined fitness function. After some genetic operators (e.g., crossover, mutation), the 'fitter' robot controller has a higher chance of being selected and generating offspring in the next generation. This process iterates until a good solution is found.

In contract to evolutionary robotics, another method of designing robot controller and/or morphology is behavior-based approach, where the designer divides the whole system into several simple parts intuitively. These separate parts are then integrated all in once through a coordination mechanism—competitive or cooperative coordination [34]. In competitive coordination, only one part has an effect on the output of the robot,

while in cooperative coordination, several parts contribute to the output of the robot with different weights. The behavior-based method was shown to be very robust in many tasks such as gait control in locomotion and object transport. The challenge of designing robot controller and/or morphology using behavior-based method is it requires a lot of experience from the designer. Moreover, when the system is highly coupled, separating the whole design into different parts may not be a good strategy. However, evolutionary robotics provides the designer an alternative way of controlling the robot as a whole, rather than focuses on the details of each separate component. The synthesized control system is a result of self-organized process.

There are many control structures that can be adapted in evolutionary robotics. Artificial neural network is a popular choice for the robot's controller due to its simple representation and strong power of control. The tree-based structure like LISP is also very robust, and it is widely used in evolutionary programming [67]. The *building blocks* method was proposed by Brooks [68]. However, these blocks are described using high-level languages, which are not suitable for evolution in the low level. Comparing with other control structures, artificial neural network have many advantages [69, 70]. According to the types of behavior (e.g., simple perception or non-linear dynamic), we can choose different neural networks (forward neural networks and recurrent neural networks, etc.). Recent development reveals that we can even evolve the topology and weights of the neural networks [71].

There are two main research aims in evolutionary robotics: 1) engineering—developing the control strategy for robots; 2) biology—to understand the biological systems using simulated (or physical) evolution. In engineering, a range of work has been presented, ranging from simple behaviors, such as phototaxis behavior, self-charging behavior to complex behaviors such as navigation and locomotion of robots with multiple degrees of freedom. In biology, evolutionary robotics is used as a tool to understand the general principles of evolution. It provides much more efficient and faster way to validate or even create hypothesis based on evolution in simulation or real robots, compared with the slow evolutionary process in nature. AVIDA [72] and AEvol [73] are two computer software systems that are used for studying the evolution of bacterias. In hypothesis validation, evolutionary robotics was used as a tool to investigate some key issues in biology. For example, whether altruistic plays an important role in cooperation among species [74, 75], the conditions of emergence of communication during the evolution [76], and how morphology and control are coupled [77], coevolution of predator and prey [78, 79].

Two methodologies are adapted in evolutionary robotics. The first one is evolving the robot(s)'s solutions in simulation and then transferring the best solution into the real robot(s), the other is evolving the solution directly on the physical robot(s). A drawback of the first method is *reality gap*. That is, the simulation used for generating the solutions may not can not match the robot(s)' real operating environment, causing the decline of the robot(s)'s performance. Many works are done to reduce the *reality gap* [80, 81, 82]. In [80, 81], the author presented the transferability approach to improve the control quality of the robot. The controllers were generated in simulation, and the best controller (selected with multi-objective criteria) was transferred into the real robot, and the data collected were used for refining the simulator. This increased the quality of controllers generated in simulation, and also reduced the number of experiments to be conducted on the real robot.

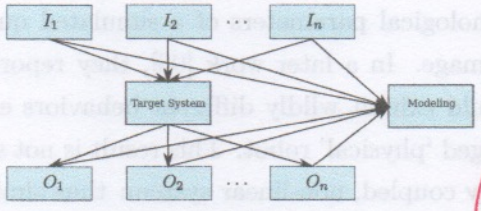
In [83], the evolution was even performed directly on the real (Khepera) robot to evolve the controller to perform navigation task. The major advantage of this is the *reality gap* has been completed avoided. However, it may take a significant amount of time to evolve a desirable solution. In [83], it took two weeks to evaluate only 100 generations. The battery is still a problem as it can only last a limit time. Although the authors in [83] used a wire to connect the power of the robot with a charging station, it is not desirable for outdoor experiments and multiple robots. Recently, a distributed online onboard evolutionary method (artificial embodied evolution) has received much attention [84, 85, 86]. In artificial embodied evolution, the population is distributed among different robots, and the gene exchange is done through *mating*. Each robot only exchanges its genes with its nearby neighbors. There is no central control over the group of robots. This approach is particularly suitable for the situation where the environment that the robots are operating on is not predictable or changing after deploying the robots. The robots need to evolve to adapt to the changing environment, while satisfying certain basic requirements inserted by the designers. A more challenging research area could be evolving both the morphology and controller of the robots in a distributed manner—*evolution of things* [87]. This forms an open-end evolution among different artifacts, which is similar to the process of how living creatures evolve in real world. The fast development of 3D printing technique makes this method appealing [88].

2.2.3.3 System Identification

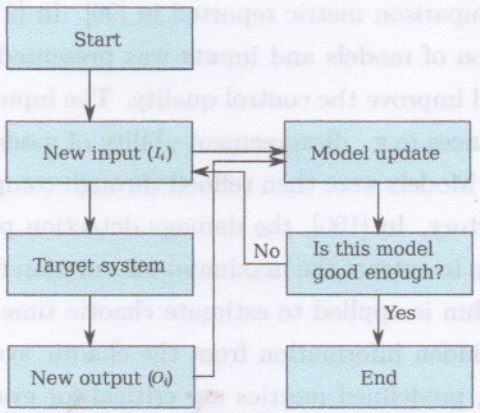
System identification which is about building the model of a hidden system through conducting a set of experiments is widely used in both academic and industrial areas [89]. The experiments are conducted by a series of inputs into the system and the outputs corresponding to the inputs are collected. The process of system identification is to find a model that fits the inputs and outputs. System identification process is composed of observed data, model structure, a criterion to evaluate different models according to the observed data, validation of the obtained model based on different data set, and revision of model if necessary.

There are two system identification approaches: the offline approach and online approach. In the offline approach, the observation data is collected first, and the aim of modeling is to generate a model that fits the observed data. This method is often used when the data is easy to collect or the parameters and operating environment of the system do not change too much in a short time. However, in some cases, the parameters of the system are always changing due to different operating environment. That means the obtained model based on the previous observation data can not be applied to the new situation any more. In this case, the model should be updated using the new observed experimental data and online system identification method. The two approaches of system identification are shown in Fig. 2.5. Note that for some systems, there are only outputs.

System identification can be divided into two main procedures: modeling and estimation. Modeling defines the order or general structure of the hidden system [90]. Estimation identifies the parameters associated with the given structure. There are many estimation algorithms to determine the parameters for a given structure. Typical estimation algorithms are recursive prediction error methods [91], which are based on gradient search. Gradient search method such as greedy algorithm can easily get trapped in local optima, especially when there are numerous local optimal points near the global optimum. An alternative search method is using evolutionary algorithms. As evolutionary algorithms use population-based search method, this makes it more likely to get rid of local optima. There are some system identification methods that combine the modeling and estimation process, e.g., NARMAX method [11]. Neural network is also good representation of the system under investigation, as its topology (structure) and weights (parameters) can be optimized simultaneously. The disadvantage of neural network is it is difficult to interpret the obtained model especially when there are multiple layers.



(a)



(b)

Figure 2.5: The diagram showing the two approaches ((a): offline; (b): online) for system identification. (I_i, O_i), where $i \in \{1, 2, \cdots, n\}$, represents a pair of input data and output data.

Genetic programming provides an alternative way for finding the structure and parameters of the system. The model represented by genetic programming is described in a tree-based structure. As the structure in genetic programming is evolving, the obtained model may have different structures in different runs [92]. Bloating [93] is a problem in genetic programming, where the growth of the tree increases the complexity of the model structure.

Coevolutionary algorithms provide an effective way for system identification [94], [95, 96, 97, 15, 98, 99]. A range of work have been performed on simulated agents. In [95], Bongard and Lipson proposed the *estimation-exploration algorithm*, a nonlinear system identification method to coevolve inputs and models in a way that minimizes the number of inputs to be tested on the system. In each generation, the input that leads to the highest disagreement between the models' predicted output in simulation was carried out on the real system. The quality of the models was evaluated through quantitatively comparing the output of the real system and the models' prediction. The method was

applied to evolve morphological parameters of a simulated quadrupedal robot after it undergoes 'physical' damage. In a later work [96], they reported that "in many cases the simulated robot would exhibit wildly different behaviors even when it very closely approximated the damaged 'physical' robot. This result is not surprising due to the fact that the robot is a highly coupled, non-linear system: thus similar initial conditions [...] are expected to rapidly diverge in behavior over time". They addressed this problem by using a more refined comparison metric reported in [96]. In [97], an algorithm which is also based on coevolution of models and inputs was presented to model the simulated quadrotor helicopter and improve the control quality. The inputs were selected based on multiobjective performances (e.g., disagreement ability of models as in [95] and control quality of a given task). Models were then refined through comparing their prediction to each selected test trajectory. In [100], the damage detection process is conducted by a coevolutionary algorithm to extract the maximum information from the system. In [101], a coevolutionary algorithm is applied to estimate chaotic time series, in which the test data that can extract hidden information from the chaotic system co-evolves with the models. In these works, predefined metrics are critical for evaluating the performance of models.

Many studies also investigated the implementation of evolution directly in physical environments, on either a single robot [102, 103, 104] or multiple robots [105]. In [102], a four-legged robot was built to study how it can infer its own morphology through a process of continuous self-modeling. The robot ran a coevolutionary algorithm onboard. One population evolved models for the robot's morphology, while the other evolved actions (inputs) to be conducted on the robot for gauging the quality of these models through comparing sensor data collected. In [105], a distributed coevolutionary approach was presented to coevolve on-board simulators and controllers of a swarm of ten robots to perform foraging behavior. Each robot has its own simulator which models the environment. The evolution of each robot's simulator was driven by comparing the real-world foraging efficiency (a pre-defined fitness metric) of its nearby neighbors each executing the best controller generated by their own simulators. Each robot has a population of controllers, which evolved according to the robot's on-board simulator. The best controller was chosen for performing real-world foraging. In all of the above approaches, the model optimization is based on pre-defined metrics (explicit or implicit), which are task dependent.

2.3 Combining AI/Robotics and Animal Behavior

The variety of animal behaviors in nature is immense, ranging from simple perception to complicated behaviors such as navigation and communication. The scientific study of animal behavior is pursued not only because it is a subject of interest in itself, but also because the knowledge gained from it has several practical applications. In AI/Robotics, there is a large interest of studying animal behavior, as the model/knowledge learned can be used to build a more intelligent machine. At the same time, building machine that mimics the animal helps researchers better understand its behavior. In this section, we review how AI/robotics and animal behavior study benefit each other. In Section 2.3.1, we introduce some interesting animal behaviors observed in nature. In Section 2.3.2, we detail how animal behavior can be used as inspiration for solving engineering tasks, especially in the area of AI/robotics. In Section 2.3.3, we show how AI/Robotics can contribute to the study of animal behavior, which is the theme of this thesis.

2.3.1 Animal Behavior in Nature

Comparing with the behaviors exhibited by complex animals such as mammals, insect behavior arises a large interest both by biologists and roboticists. One reason is due to the simple neural system of the insects, which provides a good inspiration for engineering purpose. This simplicity also makes it easy to replicate the intelligence exhibited by the insects.

A basic insect behavior is taxis, which is its intrinsic behavioral response to a specific stimulus. Taxis is divided into different types according to the stimulus which elicits the insect's response. These behaviors include phototaxis (light), chemotaxis (chemicals), thermotaxis (temperature), etc. For example, a lobster follows the salt-water plume—a kind of chemical signal to find the source. Another interesting taxis behavior of crickets is that the female crickets perform complex auditory orientation behavior towards the male crickets. Researchers have found this complex sound localization behavior emerges from simple reactive steering responses to specific sound pulses generated by male crickets [106]. Apart from taxis behaviors, some insects use the stimuli as cues for navigation or migration. For instance, the bee uses its vision system to navigate in the air and avoid obstacles. Another interesting behavior found in insects is the ball movement of dung beetles [107]. Once the dung beetles form the pieces of dung into a ball, they always roll

the dung-ball in a straight line using various stimuli (e.g., the moon, sun and polarised light [108, 109]) as visual cues to transport the food source. This behavior ensures that they keep away from the competitors as far as possible. The stimulus-response behaviors in insects mentioned above are investigated by biologists for centuries. Although the behavior exhibited by insects may be simple and easy to mimic, it is not trivial to be modeled and well understood [110]. When investigating such behaviors, biologists need to learn how to interact with the animal in a meaningful way to extract all of its behavioral repertoire.

Apart from the behavior of a single animal, swarm behaviors, which are emergent (collective) behaviors that arise from the interactions of a number of animals (especially social insects) in a group, have also been widely observed in nature. The individual behaviors in a swarm tend to be relatively simple [17]. The global behavior that is exhibited in a swarm is a result of self-organized process. Researchers found that individuals do not need the representation or complex knowledge to build a map of what the global behavior should be [111]. There are no leaders in the group. From the point of control, swarm behavior is a distributed control system which does not rely on central coordination.

Many swarm behaviors are observed in nature. For example, the flocking behavior of birds are of particular interest to humans. This is not only because of their beautiful shape formed in a group, but also how the birds coordinate with each other to maintain that shape. A simple mathematical model was proposed by [112] to describe the individual behavior of each bird in a flocking. The three rules are: attraction, repulsion and alignment. In the attraction rule, the birds will be attracted by their neighbors, and this would result in each bird moving towards to the 'center' of their neighbors. Repulsion means the bird needs to avoid colliding with each other. Alignment assumes that each bird moves in the same direction with its neighbors. Although there is no proof that flocking birds follow exactly the three rules, it is attractive that the boids (in simulation) following such simple rules can mimic the real flocking behavior very closely. There are many other swarm behaviors which are also studied extensively such as the aggregation of cockroaches [113], foraging in ants [114], flashing synchronization in fireflies [115], mound building in termites [116].

2.3.2 Swarm Optimization and Swarm Robotics

In the previous sections, we introduce some interesting animal behaviors observed in nature. Many algorithms are inspired from observation of animal behaviors. In this sec-

tion, we will review two bio-inspired algorithms—ant colony optimization algorithm [117] and particle swarm optimization algorithm [118]. Swarm robotics which uses the swarm behavioral rules of social insects as an design principles to solve complex tasks has been paid much attention in recent decades.

2.3.2.1 Swarm Optimization

Ant Colony Optimization

Ant Colony Optimization (ACO) is an optimization technique that gets inspiration from foraging behavior of ants. When ants go out to search for food, they will leave pheromone in the path. The other ants will be attracted by the pheromone, the strength of which represents the quality of the food source. Researchers have found that this indirection communication, which is know as *stigmergy* [119], leads them to found the shortest path along their nest and location of food source. The initial application of ACO is to find the optimal path in the combinational (discrete) problem. Note that nowadays the application of ACO algorithms ranges from network optimization (e.g., routing and load balance [120]) to continuous optimization [121].

The principle of ACO algorithms can be divided into the following two steps:

- Use the pheromone model to generate candidate solutions. From the aspect of mathematics, the pheromone model is a parameterized probability distribution in the search space.
- The candidate solutions are used as a bias for the future sampling to get better solutions.

For a complete implementation of ACO algorithms, see [122].

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is another an optimization algorithm which gets inspiration from flocking of birds or schooling of fish. It was first proposed by Kennedy and Eberhart [118]. The initial application is to optimize the weights of neural networks—a continuous optimization problem. It is also widely used in discrete optimization.

The basic component in PSO is called *particle*. A PSO algorithm consists of a finite set of particles. The movement of each particle is updated using *velocity*. The velocity of each particle in each time step is updated based on its current velocity, the deviation between the best position (it has found so far) and its current position, and deviation between the best position by its neighbors and its current position. This will result in the particles moving towards the high-quality solutions after certain iterations. The update of each particle can be written using two equations as follows:

$$\overrightarrow{v}_{i+1} = \overrightarrow{v_i} + c_1\overrightarrow{R_1} \otimes (\overrightarrow{p_i} - \overrightarrow{x_i}) + c_2\overrightarrow{R_2} \otimes (\overrightarrow{p_g} - \overrightarrow{x_i}) \tag{2.4}$$

$$\overrightarrow{x}_{i+1} = \overrightarrow{x_i} + \overrightarrow{v_i} \tag{2.5}$$

where $\overrightarrow{R_1}$ and $\overrightarrow{R_2}$ are independent random number generators that return a vector of random values in range $[0, 1]$. c_1 and c_2 are referred to as acceleration coefficients. $(\overrightarrow{p_i} - \overrightarrow{x_i})$ and $(\overrightarrow{p_g} - \overrightarrow{x_i})$ represent respectively the deviation between the best position (it has found so far) and its current position, and deviation between the best position by its neighbors and its current position. The first item in Eq. (2.4) keeps the particle moving in the previous direction; the second item makes the particle move towards the best position of its own; the third position forces the particle move towards to the best position that its neighbors have found. Eq. (2.5) updates the particle's position.

2.3.2.2 Swarm Robotics

In the previous sections, we talk about how swarm behaviors can be used as inspirations for algorithms design. In this section, we introduce how to use swarm intelligence techniques to multiple robots research, which is referred to as swarm robotics. Many social insects (e.g, ants, termites, wraps and bees) behavior can be used as inspirations for swarm robotics.

Swarm robotics investigates how multiple robots each with limited ability communicate, coordinate and self-organize to accomplish complex tasks. To finish the same task, using a single expensive robot with complex control structure may be feasible but it may have low efficiency and prone to failure. The advantages of swarm robotics are as follows:

- *Robustness* The robustness of a swarm robotic system can be explained in the following reasons: 1) if some robots failed, the other robots would replace the functions of the failed robots; 2) the control is distributed; 3) as the individual is simple, it is less likely to be damaged; 4) the perception from multiple robots would increase the system's robustness. Note that in some swarm robotic systems, there may be exceptions. That is, some individuals' failure would influence the whole self-organized process [123].

- *Scalability* In swarm robotics, the number of robots do not have significant difference on the global performance/behavior in the system, unless the number is sufficiently small. That is, increasing/decreasing certain number of robots, the system is still under control and the coordination is maintained. When investigating the performance of a swarm robotic system, scalability study is usually considered [124, 125].

- *Flexibility* The swarm could easily adapt to the changing tasks and generate relevant solutions [126]. The role of each robot in the swarm could be changed depending on the need for the task.

In order to cooperate, the robots need to interact with each other and environment. There are three kinds of interaction in swarm robotic systems.

- *interaction via environment* In this interaction method, the robots communicate with each through changing the environment. There is no explicit communication between each robot (i.e. they do not exchange message). In nature, the pheromone ants leave when foraging is an environmental stimulus for locating the food source.

- *interaction via perception* In this method, the robots can perceive each other in a limited range. This perception is local and there is no explicit communication between each robot. This requires the robots can distinguish between robots and objects in the environment. In nature, when ants need to collectively pull the food to the nest, they need to perceive each other (to avoid collision) and the food (object).

- *interaction via explicit communication* In this method, a network is required to communicate with a swarm of robots in real time. This could be done by broadcast (e.g., WiFi [127]) or a distributed sensing network [128]. How to build a reliable network when the number of robots is significantly large is still a hot topic

widely discussed. When the number of robots increases, the load of communication increases exponentially. A possible solution is combining the advantage of network communication and local communication using the robots' perception.

A range of tasks have been demonstrated in swarm robotics. The tasks range from aggregation [129, 21, 130, 113], dispersion [131, 132], pattern formation [133, 134], collective movement [135] to cooperative transport [136, 137, 138, 124], etc. Aggregation can be considered as the fundamental behavior of other more complex tasks. In [113], a group of robots mimic the cockroaches' aggregation behaviors, in which the robots gather into or leave the nest with a probability proportional to the size of the nest. The advantage of using stochastic algorithm is that they do not need to form a connected network in the initial configuration. In [21], the robots each with a binary sensor were reported to aggregate into a single cluster, validated using 40 e-puck robots. The robots do not need to perform algorithmic computation. This work was scaled well using 1000 robots in simulation. In [139], Werfel et al. designed a group of termite-inspired robots that work collectively to build several structures. The robots communicate with each other using *stigmergy*. In [135], a group of nine Kobot robots were reported to mimic the flocking behavior of birds. These robots followed some simple rules similar to those proposed by Reynolds [112]. In cooperative transport, Chen et al. [124] proposed a strategy in which the robots only push the object when the robots' vision of the goal is occluded by the object. This strategy was proved to push any convex object in a plenary environment.

2.3.3 Contribution of AI/Robotics to Ethology

In the previous sections, we reviewed how animal behavior study can be used as inspiration for designing algorithms and robotic systems. However, robotics can also benefit animal behavior study. In this section, we review two approaches in AI/robotics to contribute to the study of animal behavior: *learning from synthesis* and *robot-animal interaction*.

2.3.3.1 Learning from Synthesis

Ethologists have studied animal behavior over a century. There are some basic steps that ethologists follow in the study of animal behavior [140]. The first step is observation, and after that they formulate some scientific questions on the observed behavior, and

generate hypothesis to answer these questions. In order to verify the hypothesis, they actively conduct related experiments on the animals and collect data. After analyzing the data, the conclusion will be made to support or reject their hypothesis.

Robotics or artificial life can be used as an alternative methodology to investigate and understand animal behavior. Robots can be used as physical models of animal behaviors for testing hypotheses [141, 6]. For example, taxis behavior has often been implemented on mobile robotic systems for investigating steering and navigation [110]. In [142], Webb used a robot to model the phonotaxis behavior of crickets [143]. The robot can locate the position of a sound source and move towards it under different conditions. There was a good agreement between data collected from experiments on the robot and the animal. Another taxis behavior—chemotaxis in which animals follow a specific chemical trial has been used as a model for robots to find odour source based on artificial neural networks [144] and even Braitenberg vehicles [145]. Robots can be used as a validation for the models obtained from biologists and allow them to better understand the animal behavior from a synthetic point of view. Besides, roboticists can generate new hypotheses and test them using (simulated or physical) robots.

In social behavior study, Balch et al. [146] built executable models of the behaviors and ants and monkeys, which can be directly executed by multi-robot systems. The aim is to show how research into multi-robot systems can contribute to the study of collective animal behaviors. In [147], Chappell et al. argue that there are many ways in which biologists interested in natural intelligence can learn from AI and robotics, and they outline the many specific kinds of contributions that AI can make to biological study. They also give some suggestions on how AI and robotics researchers collaborate with biologists to combine the advantage of each other and solve some cutting-edge problems in animal behavior.

As opposed to the works mentioned above, the method proposed in this thesis aims to synthesize models of agent (animal) behaviors automatically, rather than manually. This could help to spare scientists from having to perform numerous laborious experiments, allowing them instead to focus on using the generated models to produce new hypotheses and conduct further experiments.

2.3.3.2 Robot–Animal Interaction

Besides pure robot-based or AI research, researchers also use robots to interact with real animals. They build and programme robots (i.e., replicas) that can be inserted into the group of social animals [148, 149, 150, 151, 152]. Robots can be created and systematically controlled in such a way that they are accepted as con- or hetero-specifics by the animals in the group [153]. In this case, one "animal" in a group is completely controlled and they can observe the behaviors of the mixed society [154]. The behavior of the inserted robot can be controlled and the model can also be embedded into the robot for verification [155]. The behavior of robots can be programmed in such a way that its behavior is not influenced by the other real animals in the group, and they can be used as demonstrators or leaders in the experiments. Furthermore, it is easier to verify a hypothesis through controlled interaction in social behaviors.

In [148], a replica fish which resembled the appearance (i.e., visual morphology) of sticklebacks was created to investigate two types of interaction: recruitment and leadership. In [150], autonomous robots which executed the derived model were mixed into a group of cockroaches to modulate their decision-making of selecting shelter in the aggregation behavior. The robots behaved in a similar way to the cockroaches. Although the robots' appearance was different to that of the cockroaches, the robots released a specific odor that the cockroaches could detect and regard the robots as conspecifics. In [156, 152], Vaughan et al. have built a mobile robot that can interact with the ducks in a circular arena and drive them to the safe place. Halloy et al. in [157], Gribovskiy et al. designed a robot which is capable of interacting with chicks to study how the behavior of chicks can be influenced by the others in a group. In [158], a robot-fish that can interact intelligently with live zebrafish to study their preference and locomotion behavior was designed. Although robots which are well designed can be mixed in social animals, building such kind of robot is a time-consuming process. It is also expensive to some extent and requires the collaboration of researchers from different disciplines. In these works, the models were manually derived and the robots were only used for model validation. We believe that this robot-animal interaction framework could be enhanced through the proposed system identification method, which autonomously infers the collective behavior.