A Neuro-Evolution Approach to Shepherding Swarm Guidance in the Face of Uncertainty

Essam Debie*, Hemant Singh*, Saber Elsayed*, Anthony Perry†, Robert Hunjet† and Hussein Abbass*
*School of Engineering and Information Technology, University of New South Wales, Canberra ACT, Australia.
Emails: {e.debie, h.singh, s.elsayed, h.abbass}@unsw.edu.au
†Defence Science and Technology, Australian Department of Defence, Edinburgh SA, Australia.
Emails: {anthony.perry, robert.hunjet}@dst.defence.gov.au

***Abstract*—Controlling a large swarm of agents is a challenging task. *Shepherding* refers to an active field of research that seeks to address this challenge by using a control agent (sheepdog), which guides a swarm (sheep) towards a goal. Traditional shepherding involves switching between two main behaviours: *driving* the swarm towards the goal, and *collecting* stray sheep back to the flock. Evidently, the movement of the agents are dependent on their sensed information. Therefore, effectively controlling a swarm is even more challenging when sensor information or communication channels are unreliable. In this paper, we propose a shepherding methodology to achieve efficient swarm control in the presence of noise in the sensed information. The proposed approach consists of a new resting behaviour and a neural network-based reinforcement learning model. The neural network is used to learn shepherding policies using the new resting behaviour, where the objective is to optimise the frequency of sheep-to-dog interactions with varying levels of noise. The proposed approach is validated through simulations. Numerical experiments show that the proposed approach results in a more effective and stable performance compared to some conventional shepherding models from the literature.**

I. Introduction

Swarm control has a wide variety of applications in many fields including, but not limited to, search and rescue, patrolling, and exploration [1], [2]. Shepherding, in the context of this work, refers to a swarm control approach inspired by the actual herding of sheep by sheepdogs in farms. The particular interest in shepherding approach stems from the ability to manipulate and influence a large number of individuals (sheep) using a small number of control agents, referred to as sheepdogs. Sheep are reactive agents that respond to influences (manifested as force vectors); they do not require knowledge about the goal state in advance. To guide the herd to the intended location, the sheepdog agent moves using a set of predefined rules. A number of shepherding models have been proposed in the literature to date. Lien et al. [3] proposed one of the earliest shepherding approaches using a dynamic road-map searching problem. Strombom et al. [4] proposed a shepherding model wherein the sheepdog dynamically switches between driving and collecting behaviours based on the state of the sheep. If the sheep are close to each other, the sheepdog moves to a driving position behind the global centre of mass of the sheep, and then it drives them towards the goal position. On the other hand, if the sheep are scattered, the sheepdog first attempts to collect the furthest sheep and bring it back to the flock; and then drives them to the goal. Bat-Erdene and Mandakh [5] proposed a boids-based approach for shepherding. Four simple rules adapted from Reynolds's boids rules [6] were used to control the flocking behaviour.

A. Motivation and Contributions

Traditional shepherding models, such as the one proposed by Strombom et al.[1] use a deterministic approach to switch between the dog's behaviours (driving and collecting), while assuming perfect communication among the agents (sheep-to-sheep and sheepdog-to-sheep). As a result, the positions of all sheep are assumed to be accurately known, which in turn, implies assumed accurate estimation of the centre of gravity of the flock and the furthest sheep to the flock. However, noise is inherent in most real-world scenarios. A few studies in the literature analysed the impact of noise on shepherding models. For example, Hung et al. [7] analysed the performance of shepherding models under two types of noise: actuation noise and perception noise. In actuation noise, the disruptive forces generated during actuation adversely impact the influence of the sheepdog in herding the flock to the target position. On the other hand, sensory noise affects the quality of the information received by the sheepdog about the location of sheep. The study analysed the shepherding performance with at various noise levels. The shepherding performance within the widely used Strombom model was found to be very sensitive to the noise environments. More specifically, with increasing levels of actuation noise, the sheepdog spends extraneous time in switching between driving the flock and collecting the furthest sheep. Moreover, different levels of actuation noise also required different parameter setups of the model to mitigate the impacts of noise.

Such sources of uncertainty, in turn, lead to difficulty in determining which behaviours are required to achieve effective shepherding and when they should be used. In this paper, we propose a reinforcement learning approach to address the shepherding problem under model uncertainty. In particular, we investigate the uncertainties resulting in the

[1]For brevity, we refer to it as Strombom model subsequently

system from actuation noise. This manifests as the swarm agents (sheep) not being influenced by the shepherd in the manner expected. The proposed approach addresses the drawbacks of traditional shepherding models by introducing a new behaviour called *resting*, whereby the sheepdog does not perform driving/collecting actions normally and instead waits for the subsequent time-step(s). A reinforcement learning approach is used to learn high-level shepherding policies (i.e. optimise when and how to interact with the sheep) under model uncertainty constraints. The hypothesis behind introducing the new behaviour is that it can reduce the frequency of the unnecessary switching pattern between collecting/driving observed in traditional approaches [7]. This can affect model performance in two ways: first, frequent switches between driving/collecting behaviours imposes unnecessary interactions with the sheep forcing them to move in undesired directions. Reducing shepherd-sheep interaction provides the sheep enough time to position themselves properly under model uncertainty. This in turn can affect task success rates. Second, reducing the switches between driving/collecting behaviours can lead to a reduction in the total time required to complete the task (i.e. better task efficiency).

Thus, the key intended contributions of this paper are:

- A new resting behaviour is introduced for improving the efficiency of shepherding task.
- A neural network-based reinforcement learning model is presented for learning effective shepherding policies in noisy environments.
- A neuro-evolutionary approach is presented for evolving neural network model weights.

The remainder of this paper is organised as follows. The shepherding problem and its related work are discussed in Section II. Section III introduces the proposed method of swarm shepherding. Section IV presents the experimental setup, followed by the results and discussions in Section V. Conclusions and possible future work are discussed in Section VI.

II. Background and Related Work

A. Swarm Shepherding

In this section, we describe two of the currently used models for shepherding, namely, the Strombom model [4] and the recently proposed CAD-SHEEP model [8].

1) Strombom Model: The Strombom model [4] defines the interactions between two types of agents: sheep and sheepdog. The dynamics of this interaction is controlled using two unique behaviours: driving and collecting. The movement of each sheep is determined using a weighted linear combination of influences (force vectors) of various entities in the environment, including other sheep and the sheepdog. The set of sheep agents are denoted by $\Pi = \{\pi_1, \ldots, \pi_i, \ldots, \pi_N\}$, while the sheepdog agents denoted as $B = \{\beta_1, \ldots, \beta_j, \ldots, \beta_M\}$. For consistency, we utilise the a weighted sum of force vectors to describe how a sheepdog positions itself in the environment dependant on the state of the sheep.

TABLE I
Agent parameters.

Parameter	Description	Typical values
N	total number of sheep	10 – 150
r_s	sheepdog detection distance	65 m
$R^a_{\pi\pi}$	agent to agent interaction distance	2 m
ρ_a	relative strength of repulsion from other sheep	2
c	relative strength of attraction to the n nearest neighbours	1.05
ρ_s	relative strength of repulsion from the sheepdog	1
h	relative strength of proceeding in the previous direction	0.5
$W_{e\beta_j}$	relative strength of angular noise	0.3
S_{β_j}	default sheep speed per time step	1 m/ts
p	probability of moving per time step while grazing	0.05

Driving behaviour: This is the primary behaviour in Strombom executed by the sheepdog when all sheep are considered clustered. The sheepdog evaluates if the sheep are clustered according to Equation 1. It does so by checking if all sheep that it operates on are within $R1$ distance from their Global Centre of Mass (GCM), using Equation 2.

$$Clustered(\Omega^t_{\beta_j}) = \begin{cases} 1 & dist\,(\pi_i, GCM) \leq R1, \forall \pi_i \in \Omega^t_{\beta_j} \\ 0 & otherwise \end{cases} \tag{1}$$

$$R1 = fn(R^a_{\pi\pi}, N) \tag{2}$$

The sheepdog moves towards a driving position situated behind the sheep on the direction to the goal. The sheepdog moves towards the driving position with a normalised force vector $F^t_{\beta_j cd}$ calculated as shown in Equation 3.

$$F^t_{\beta_j cd} = \frac{P^t_{\beta_j \sigma_1} - P^t_{\beta_j}}{\| P^t_{\beta_j \sigma_1} - P^t_{\beta_j} \|} \tag{3}$$

Collecting behaviour: If any sheep are at a distance greater than $R1$ to the GCM, then $Clustered(\Omega^t_{\beta_j}) = 0$, and the sheepdog will go to collect the furthest sheep π_f from the GCM. The collecting point behind the furthest sheep toward the GCM is calculated using Equation 4.

$$R^a_{\pi\pi} N^{2/3} \tag{4}$$

The sheepdog moves towards the collecting position with a normalised force vector $F^t_{\beta_j cd}$:

$$F^t_{\beta_j cd} = \frac{P^t_{\beta_j \sigma_2} - P^t_{\beta_j}}{\| P^t_{\beta_j \sigma_2} - P^t_{\beta_j} \|} \tag{5}$$

A random force, $F^t_{\beta_j \epsilon}$ is applied to the sheepdog movement to help resolve deadlocks among different forces. The

strength of this angular noise is denoted by $W_{e\beta_j}$. The combined force vector for sheepdog movement is therefore given by:

$$F^t_{\beta_j} = F^t_{\beta_j cd} + W_{e\beta_j} F^t_{\beta_j \epsilon} \tag{6}$$

The total force of the sheepdog is used to update the agent position as described in Equation 7.

$$P^{t+1}_{\beta_j} = P^t_{\beta_j} + S^t_{\beta_j} F^t_{\beta_j} \tag{7}$$

Sheep movement is defined by a set of repulsion and attraction forces. Sheep π_i is repulsed from sheepdog β_j with force $F^t_{\pi_i \beta}$ and repulsed from other sheep $\pi_{i1}, i_1 \neq i$ with force $F^t_{\pi_i \pi_{i1}}$. It is attracted to the centre of mass of its closest neighbours $\Lambda^t_{\pi_i}$ with force $F^t_{\pi_i \Lambda^t_{\pi_i}}$. Including angular noise with force $F^t_{\pi_i \epsilon}$ gives the force vector for movement of sheep π_i as:

$$F^t_{\pi_i} = W_{\pi_\upsilon} F^{t-1}_{\pi_i} + W_{\pi\Lambda} F^t_{\pi_i \Lambda^t_{\pi_i}} + W_{\pi\beta} F^t_{\pi_i \beta_j} + W_{\pi\pi} F^t_{\pi_i \pi_{-i}} + W_{e\pi_i} F^t_{\pi_i \epsilon}, \tag{8}$$

where each W represents the weight of the corresponding force vector.

The total force of each sheep is used to update the agent position as depicted in Equation 9. If there is a sheep within three times the sheep-to-sheep interaction radius, the sheep stops; thus, it will set its speed to zero: $S^t_{\beta_j} = 0$, otherwise it will use its default speed, $S^t_{\beta_j} = S_{\beta_j}$. The speed of a sheep is assumed constant; that is, $S^t_{\pi_i} = S_{\pi_i}$

$$P^{t+1}_{\pi_i} = P^t_{\pi_i} + S^t_{\pi_i} F^t_{\pi_i} \tag{9}$$

Table I summarises the set of parameters and their typical settings for the Strombom model.

2) CAD-SHEEP Model: CAD-SHEEP [8] was proposed to overcome the inefficiencies encountered in the collecting behaviour in the original Strombom's model. In particular, always collecting the furthest sheep was shown to be inefficient when the furthest sheep lies along the path to the goal. Experimental results showed that CAD-SHEEP is more efficient than the Strombom model in terms of the number of steps required to guide the sheep. This is due to the lower frequency of collecting and the quality of the decision as to which sheep to collect.

B. Reinforcement Learning

In reinforcement learning (RL), the decision process is modelled as a Markov Decision Process (MDP) [9]. MDP models sequential decision making of a learning agent that interacts with an environment on a given discrete time scale. Formally, an MDP is described by tuple $M = (S, A, T, R, \gamma)$ where S is the set of states of the environment, A is a set of actions, T is a state-transition function that measures the probability of obtaining a next state s_{t+1} given a current state-action pair (s_t, a_t), R is a reward function that defines an immediate reward achieved at each state-action pair (s, a), and $\gamma \in [0, 1)$ denotes a discount factor. To determine the reward R for each state action pair, a common approach is to learn a function $Q(s, a) : S, A \rightarrow R$ over the set of states S and actions A that can be taken from each state. This state-action function reflects the long term quality of taking an action in a given state. Q-learning [10] is one method to estimate the quality of state-action pairs. In Q-learning, the update rule is given as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \gamma max_a Q(s_{t+1}, a_{t+1}) \tag{10}$$

According to Equation 2, the quality of state-action pair is the sum of the immediate reward and the discounted return obtained by following the optimal policy in subsequent states.

A policy ψ of RL algorithm defines which action should be taken in each environment state where the goal of RL algorithm is to find an optimal policy ψ^* in order to maximise the expected sum of discounted future reward as follows:

$$G(\psi) = E\left[\sum_{t=0}^{T} \gamma_t R(s_t, a_t)\right] \tag{11}$$

C. Partial Observability

In many real-world application domains, one often encounters incomplete and noisy state information resulting from partial observability of the environment. In such cases, autonomous agent estimates the states given its current observation of the environment (i.e. the output of the agent's sensors). Partially Observable Markov Decision Process (POMDP) [11] is an extension to the MDP model that was proposed to capture such dynamics by taking into account that the state information is only partial information of the underlying environment states. Formally, POMDP can be described by a 7-tuple as follows:

$$(S, A, T, R, \gamma, \Omega, P) \tag{12}$$

where S, A, T, R, γ are the states, actions, transition, rewards and discount factor as in the standard MDP model. Instead of relying on true state information, the agent now receives an observation $o \in \Omega$ according to the probability distribution P. At each discrete time step t, the agent executes an action $a \in A$, which causes the environment to change into a next state $s_{t+1} \in S$ with the transition function $T(s_{t+1}|s_t, a_t)$. The agent then receives an observation $o_{t+1} \in \Omega$ about the next state of the environment with probability $P(o_{t+1}|s_{t+1}, a_t)$. Finally, the agent receives a reward $R(s_t, a_t)$. Then the process repeats. The agent uses all received observations to form a hidden state refereed to as the belief state $b_{t+1}(s_{t+1}) = p(s_{t+1}|O_{t+1}, a_t, b_t)$ (ie. the probability of being in a given state s_{t+1} given current observation $O_t + 1$ and previous action a_t and previous belief b_t at state s_t.

D. Q-Learning Approximators

In environments with large state spaces, it may not be feasible to calculate the quality of every possible state-action pair. This may be addressed by learning an approximation function that in turn learns this value from a small subset of state-action value pairs and generalises it over unseen state-action pairs. Neural networks (NNs) are traditionally used to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output.

III. PRNEURO-EVOLUTIONARY APPROACH FOR SHEPHERDING UNDER UNCERTAINTY

In this paper, we propose a reinforcement learning approach to solve the shepherding problem under model uncertainty. The proposed RL acts on observed states of the environment and produces an action that minimises the desired objective (number of steps to goal). Our aim is to control the shepherd, attempting to guide a flock of sheep robots in an unknown environment. The problem is formally defined in terms of environment states $x_i \in X$, shepherding actions $a \in A$ and observations $o \in O$ such that $x_{t+1} = f(x_t, a_t)$. That is, the environment state at time $t+1$ is determined by the state of the environment and the action taken by the sheepdog at time t. The objective of the sheepdog is to minimise the number of steps taken to guide all the sheep to the desired goal location.

The set of states of the environment is $X = \{d_{gcmgoal}, d_{farthestsheep}\}$, where $d_{gcmgoal}$ is the distance between the global centre of mass to the goal, and $d_{farthestsheep}$ is the distance between the farthest sheep and GCM. The set of actions $A = \{driving, collecting, resting\}$.

Our methodology for the shepherding problem under uncertainty is based on an RL approach where an agent with a neural network (NN) function approximator determined by its weights θ, is used to control the sheepdog. At any time t, the agent performs a single decision for the best action (behaviour) to take. The greedy single policy for the agent is used to find the best action:

$$a_t^* = \underset{a_t \in A}{argmax} Q(s_t, a_t) \tag{13}$$

Among the NN architectures investigated in this study, the reported architecture provided the best performance in our experiments. The one which provided best results is a fully connected feed forward NN with two hidden layers with 12 and 6 neurons respectively. This architecture is used throughout the remaining sections of this paper.

The inputs to the NN are the observation variables of the shepherding environment at a given state X. The NN outputs the estimated Q-value for each possible action (behaviour the sheepdog may take) for the agent in order to maximise the overall cumulative reward.

Algorithm 1: Genetic algorithm

Input: mutation variance V, population size M, truncation threshold T, elite threshold E fitness function F, number of generations G

$P \leftarrow$ Randomly initialised population of NN agents
for *g = 1, 2..., G generations* **do**
  for *n = 1, 2..., N individual in the population* **do**
    Evaluate fitness $F_n = F(\theta_n^g)$
  Sort P^g according to fitness with descending order
  for $i \in (1, N-E)$ **do**
    $t \leftarrow Random(1, T)$
    offspring $o_i^g = \theta_t^g + \sigma$
  Elite Candidates $C \leftarrow \theta_{1..E}^g$
  $P^{g+1} \leftarrow C \cup o_{i \in (1,N-E)}^g$

A. Evolving Agent Weights

Gradient-based back-propagation is the prevalent method for training neural networks in supervised learning domains [12]. It is an efficient algorithm used to gradually modify neural network weights to minimise a pre-defined loss function over labelled training data. On the contrary to supervised learning problems in which ground-truth labels are given during training, RL involves learning by trial and error without direct supervision. In RL, different action sequences are executed and sparse rewards are received for those actions. An RL model must learn how to maximise future rewards from this sparse feedback. Providing labelled data for RL problems to train an NN is thus a difficult task [13].

To overcome this challenge, the standard back propagation step is omitted from the NN model. Instead, we adopt a neuro-evolution approach to evolve the best network parameters.

We evolve the weights of NN Q-value approximators using a standard genetic algorithm (GA), including only mutation and selection, as described in Algorithm 1.The GA evolves networks, each of which is an agent that can solve the shepherding problem. The GA has a population of M individuals. Each individual chromosome is a set of all the parameters of an NN, represented as a vector $\theta_{m \in M}$. At every generation $g \in G$, each θ_m is evaluated by running the environment with that NN agent to produce a reward value which is then used as our fitness score $F(\theta_m)$. With the fitness scores calculated and solutions sorted, an elite set of the top T fittest individuals are selected as parents for the next generation.

Each offspring is generated from a randomly selected parent by adding a vector of random noise σ to generate the offspringâs parameters vector, where σ is sampled from a normal distribution with mean zero and variance V. We only evolve the weights and bias parameters of the neural network in our GA and keep the network structure fixed.

Moreover, the fittest solution from the current generation is preserved in the subsequent generation.

An indirect encoding is used where each individual network is represented in the population by its id rather than encoding all of its parameters. This indirect representation achieves modularity and minimal memory footprint.

A reward function is used to evaluate the fitness of individual chromosomes based on herding performance. Particularly, a reward inversely proportionate to the average distance from the sheep to the goal location combined with average number of steps taken to reach the goal as shown in Equation 14.

$$F\left(\theta_m^t\right) = 2 - \frac{D_t}{MaxD} + \frac{t}{MaxE}, \tag{14}$$

where $MaxD$ is the maximum distance in the environment, D_t is the average distance between the GCM and the target location at iteration t, and $MaxE$ is the maximum episode length (max number of steps) allowed in the experimental setup. The lower the distance from the sheep to the target location and the lower the number of steps taken so far the higher the reward (fitness) value of NN. Once the evolution phase is completed, the best performing agent is chosen.

IV. Experimental Setup

In this paper, we validate the proposed approach under actuation noise. Experiments are conducted using a similar parameter setup as proposed in [7], where sheep move randomly around the location they are supposed to move to. The level of actuation noise controls the range of this random movement.

The position of sheep π_i at time $t+1$ under the actuation noise is calculated as follows:

$$ActP_{\pi_i}^{t+1} = P_{\pi_i}^t + S_{\pi_i}^t \times \left(F_{\pi_i}^t + \lambda \times standardNormal()\right) \tag{15}$$

where $ActP_{\pi_i}^{t+1}$ is the position of sheep π_i at time step $t+1$, $S_{\pi_i}^t$ is the speed of sheep, $F_{\pi_i}^t$ is the total force of the sheep π_i calculated as a weighted sum of individual force vectors calculated using Equation 8. λ is an upper bound of noise calculated as in Equation 16. $standardNormal()$ is a standard normal distribution, with a mean of zero and standard deviation of 1.

$$\lambda = Act^{nl} \times (3 \times R_1) \tag{16}$$

Here, Act^{nl} is the level of actuation noise. In our experiments, Act^{nl} vary in the range $[0, 1]$ with step size of 0.1.

A simulated environment is created with a square of side length $L = 500$ and the number of sheep $N = 20$. In this environment, the goal area is located at the left bottom corner with coordinates $(0, 0)$. In each simulation, sheep are randomly initialised at the centre of the paddock with their coordinates ranging between $1/4$ and $3/4$ of the paddock length. The sheepdog is initialised at the lower left corner of the paddock. As with the evolution, the objective is to guide all sheep to the goal location with the minimum number of steps. Three actions are available for the sheepdog: collect, drive, and rest. The task is considered completed when all of the sheep are at the goal or a maximum episode length is reached.

The GA parameters are set as follows: the population size is 20, following our preliminary analysis we set 20 generations provided reasonable performance. Top performing k individuals are selected for reproduction after each generation where $k = 50\%$ of the population size. Mutation is applied with variance set to 0.1. NN agents play the simulation with a maximum episode length set to 2000.

V. Results and Discussion

In this section, we discuss and summarise the results obtained in our experiments. Results are displayed graphically with standard deviation shown by the shaded region.

A. Convergence

Firstly, we investigate the performance of the neuro-evolution approach for generating an NN model capable of capturing shepherding behaviours. Figure 1 summarises the average convergence for 10 independent evolutionary runs for each noise level. These results show that the GA can reliably learn parameter configuration for controlling the sheepdog in shepherding tasks under an increasing level of actuation noise. The evolutionary search is able to find highly-performing agents (learning to guide cluttered sheep to the goal area effectively with few errors) after three to five generations. The results also validate the system stability where performance plateaued over GA generations.

In each generation, the GA tends to make small changes (controlled by the mutation rate) to the parameter vector, suggesting that sampling in the region around good solutions is often sufficient to find better solutions. As such, a sequence of such local searches is sufficient to find high-quality shepherding policies under varying levels of actuation noise.

It is observable that the evolutionary search is not affected by actuation noise levels below 0.4, where it achieved relatively similar performance (average reward between 1.3 and 1.4). However, as the actuation noise level increases above 0.4, lower rewards (1.2 on average) are achieved. As expected, in high noise levels (0.9 and 1.0), more generations are required before a plateau is reached.

B. Comparison with Strombom Model

To validate the performance of NN based shepherding, we compare the Strombom model against the proposed NN learned model in an obstacle free environment. We use the same environment setup (environment of size 500×500 with 20 sheep and varying actuation noise levels ranging in $[0, 1]$) with a step size of 0.1. The two methods are compared in terms of their effectiveness (success rate) and efficiency (number of steps) to complete the shepherding task. Results summarised in Figure 3 show that Strombom's model is severely affected by actuation noise. We observed that that its success is inversely proportional to actuation noise. This

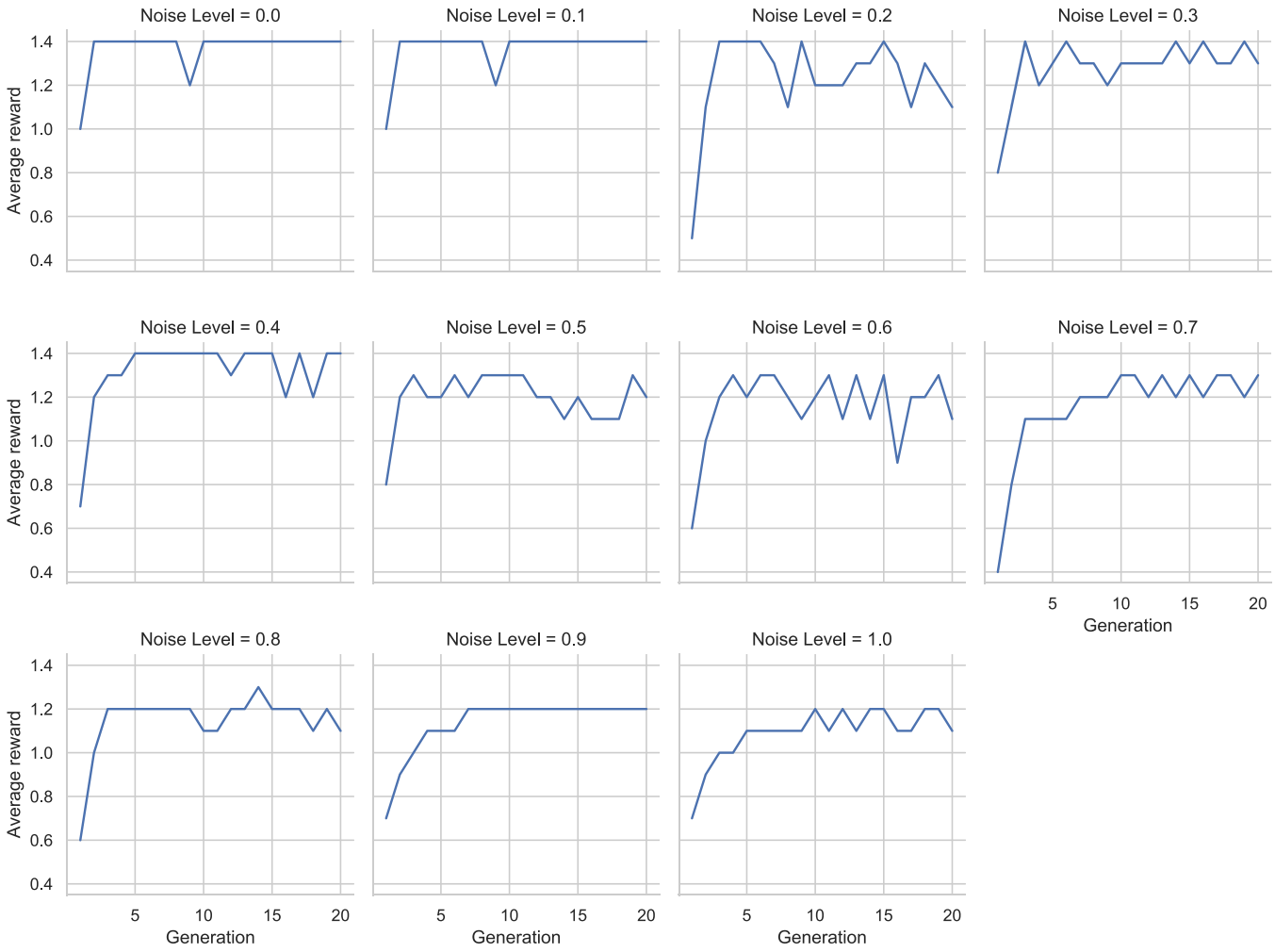

Fig. 1. GA learning performance on simulated shepherding task.

coincides with the results of Nguyen et al. [7] in their study. Encouragingly, the results suggest that our proposed NN model is resilient to varying levels of noise, achieving 100% success at most noise levels.

Furthermore, the NN model outperformed the standard Strombom method in terms of the number of steps required to guide the sheep to the designated goal area across all noise levels tested outperforming Strombom's model by 50-70% on average. The strength of our approach is especially evident at higher levels of actuation noise where the NN model can achieve robust performance.

To understand how the proposed NN model addresses the shepherding under actuation noise, we plotted the frequency of the resting behaviour executed by the NN in different levels of actuation noise. Our hypothesis was that the sheepdog should not act on every iteration in the task. Waiting for the sheep to move and take corrective actions to position themselves appropriately according to the applied sheepdog forces can help improve task performance in noisy environments. This can be observed in Figure 4 where the frequency of applying the resting behaviour is increasing with the level of noise. This validates our original hypothesis that having a resting action for the sheepdog is helpful to achieve better performance.

C. Comparison with CAD-SHEEP Model

We compared the performance of CAD-SHEEP to our proposed NN model using the same setup as introduced in Section IV. Both models achieved a 100% success rate. However, the models vary significantly in terms of efficiency as displayed in Figure 5. It can be seen that our NN approach outperforms CAD-SHEEP across all noise levels and with an increasing margin as the actuation noise level rises beyond ($\geq$ 0.7). At an actuation noise level of ($\geq$ 0.9) our NN approach outperforms CAD-SHEEP by 27%. These results emphasise the efficiency of the proposed NN model in guiding the flock to the goal.

VI. Conclusions and Future Work

In this paper, we presented an approach for shepherding a swarm of agents in the presence of noise within the expected

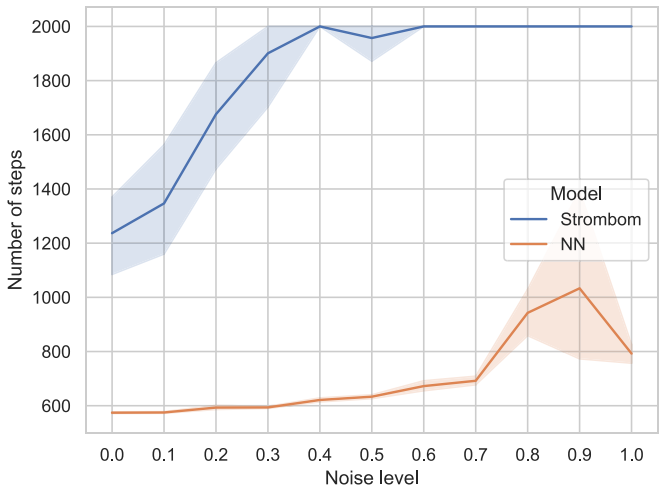

Fig. 2. Efficiency comparison between Strombom's model and NN model with different actuation noise levels.

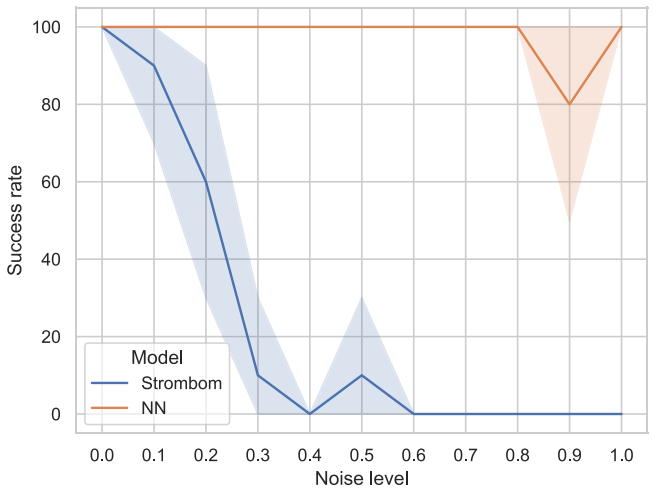

Fig. 3. Effectiveness in completing the task using Strombom's model and the proposed NN model.

swarm agent movement. The proposed approach consists of a neural network-based reinforcement learning model and a new resting behaviour to control the sheepdog. To alleviate the problems associated with providing labelled training data, a neuro-evolution approach was employed to evolve the neural network parameters. The efficacy of the proposed approach was validated through simulations outperforming traditional shepherding models such as Strombom and CAD-SHEEP. While in this paper, we only analysed sensory (acutation) noise, in the future we plan to expand this study to other sources of noise including perception noise (noise in the sheep measurements as sensed by the sheepdog). Moreover, a unified framework for shepherding under different types of noise is another potential direction for the work.

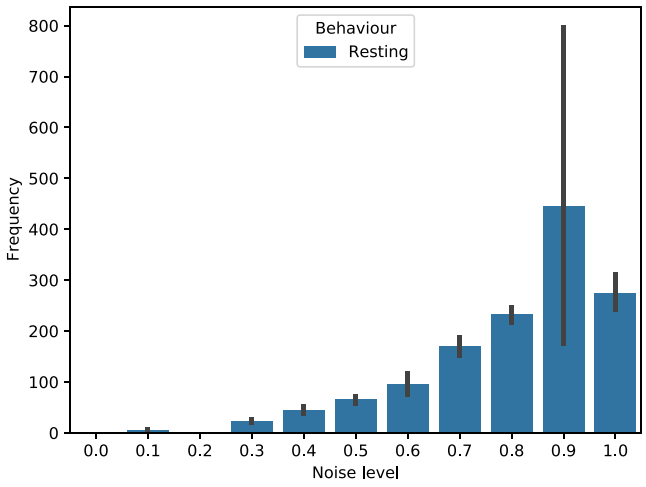

Fig. 4. Frequency of applying each shepherding behaviour in different noise levels.

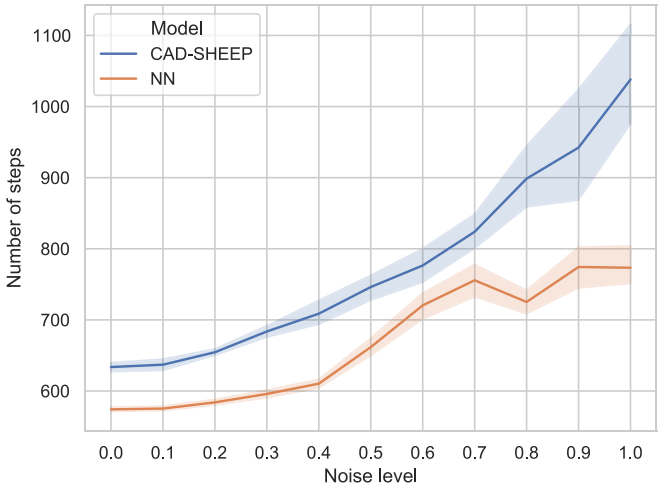

Fig. 5. Model efficiency with different actuation noise levels.

ACKNOWLEDGEMENT

This publication was made possible by Grant No N62909-18-1-2140 from the Office of Naval Research Global (ONRG). Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the ONRG.

REFERENCES

[1] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2015.

[2] B. Zhu, L. Xie, D. Han, X. Meng, and R. Teo, "A survey on recent progress in control of swarm systems," *Science China Information Sciences*, vol. 60, no. 7, p. 070201, 2017.

[3] J.-M. Lien, O. Bayazit, R. Sowell, S. Rodriguez, and N. Amato, "Shepherding behaviors," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, vol. 4, Jan 2004, pp. 4159 – 4164 Vol.4.

[4] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. T. Sumpter, and A. J. King, "Solving the shepherding problem: heuristics for herding autonomous, interacting agents," *Journal of The Royal Society Interface*, vol. 11, no. 100, p. 20140719, 2014.

[5] B. Bat-Erdene and O.-E. Mandakh, "Shepherding algorithm of multi-mobile robot system," in *Robotic Computing (IRC), IEEE International Conference on*. IEEE, 2017, pp. 358–361.

[6] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[7] H. T. Nguyen, M. Garratt, L. T. Bui, and H. Abbass, "Disturbances in influence of a shepherding agent is more impactful than sensorial noise during swarm guidance," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 2257–2264.

[8] S. Elsayed, H. Singh, E. Debie, A. Perry, B. Campbell, R. Hunjel, and H. Abbass, "Path planning for shepherding a swarm in a cluttered environment using differential evolution," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 2194–2201.

[9] R. A. Howard, "Dynamic programming and markov processes." 1960.

[10] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[11] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *Aaai*, vol. 94, 1994, pp. 1023–1028.

[12] T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," *Optica*, vol. 5, no. 7, pp. 864–871, 2018.

[13] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.