Moving Mixtures of Active and Passive Elements with Robots That Do Not Compute

Gopesh Yadav Dosieah[1,4], Anıl Özdemir[2,5], Melvin Gauci[3], and Roderich Groß[1]

[1]Department of Automatic Control and Systems Engineering, The University of Sheffield, UK
[2]Department of Computer Science, The University of Sheffield, UK
[3]Amazon.com, Inc., USA
[4]Dyson Technology Limited, UK
[5]Zebra Technologies, London, UK
`r.gross@sheffield.ac.uk`

Abstract. This paper investigates the problem of moving a mixture of active and passive elements to a desired location using a swarm of wheeled robots that require only two bits of sensory input. It examines memory-less control strategies that map a robot's sensory input to the respective wheel velocities. Results from embodied simulations show that the problem can be solved without robots having (i) to discriminate between active and passive elements or (ii) sense other robots. Strategies optimized for moving passive elements, or mixtures of active and passive elements, performed robustly when changing the mixture of elements, or scaling up the number of robots (up to 25) or elements (up to 100). All strategies demonstrated to be fairly robust to noise and adaptable to active elements of different dynamics. Given the simplicity of the robot capabilities and strategies, our findings could be relevant in scenarios where microscopic swarm robots need to manipulate mixtures of elements of unknown dynamics, with potential applications in nanomedicine.

1 Introduction

Many studies examine the ability of swarms of robots to physically manipulate their environment. For example, this could concern the cooperative transport of an object that is too heavy to be effectively displaced by individual members of the swarm [11, 1, 28, 4, 25, 5]. In the following, we specifically focus on the ability of swarms of robots to manipulate numerous elements at the same time.

In some application scenarios, the elements to be manipulated would be entirely *passive*, as exerting no control over their movement. This would be the case, for example, when collecting plastic waste in water bodies [20]. Beckers et al. [2] study a group of robots equipped with C-shaped pushers. The latter enable the robots to push and retain multiple, smaller objects even during turns. Each robot moves in a straight line and rotates by a random angle when detecting an obstacle or when the resistance met by its pusher exceeds a threshold. The strategy is shown capable of clustering 81 objects in a bounded environment. Melhuish et al. [15] propose an extension of this strategy enabling a group of robots to spatially separate colored objects into distinct clusters. Kim and Shell [10, 21] studied a cluster task similar to [2]. As the robots are circular in shape, careful design is required to prevent the formation of (possibly separate) clusters along the boundary. The authors propose a strategy by which some robots are 'diggers', which follow walls and separate objects from the boundary, whereas others are 'twisters', which act on 'dug up' objects, and push them towards the center. This results in all objects ending up in a single cluster.

In other application scenarios, the elements to be manipulated may be *active*, as exerting control over their movements. This would be the case, for example, when shepherding groups of land mammals. The problem of *shepherding* a set of active elements to a goal region has been addressed using single-robot systems [3, 6, 22–24, 26, 27]. Vaughan et al. [26] propose a strategy to shepherd a flock of ducks towards a goal. An external system is used to determine the position of the flock, as well as the position and orientation of the robot. The robot is attracted towards the flock, the further the latter is away from the goal, the stronger the attraction. Moreover, it is repelled from the goal. This simple behavior succeeds in driving the flock towards the goal. A number of studies use distinct behaviors for (i) gathering the elements, and (ii) driving them towards the goal, which are executed either in alternation or simultaneously. Gathering maneuver include moving in arcs, zig-zags or orbiting. Driving maneuver include approaching the flock in a straight line from a position opposite to the goal, or performing gathering maneuver while gradually moving towards the goal [3, 6, 22, 24]. In [23], a robot shepherds a group of sheep agents. It relies only on local sensing. It moves repeatedly behind the sheep robot that is furthest away from the goal. Owing to a cohesion behavior however the sheep have no natural tendency to split into separate groups. Studies considering a group of shepherd robots include the work by Lien et al. [13] that demonstrated that a group of shepherds outperformed a single one. Other examples are Miki & Nakamura [16] and Lee & Kim [12] which study sets of simple rules to replicate common types of shepherding behaviors. They both demonstrated that the active elements could be herded by a swarm of robots without centralized coordination.

In our previous works [8], a computation-free paradigm for controlling swarms of simple robots was proposed. It was subsequently used to design computation-free controllers for swarms of robots to cluster passive elements [7], without specifying a desired goal region. Moreover, it was used to design computation-free controllers for swarms of robots to shepherd active elements towards a goal region [18].

This paper goes beyond prior work in swarm robotics by considering for the first time the problem of moving a loose mixture of active and passive elements towards a goal region. This problem is important for real-world applications, where the dynamics of the elements to be manipulated may not be known, or could vary among the elements. We hypothesize that a single set of rules exists that requires no run-time memory and yet solves the problem irrespective of

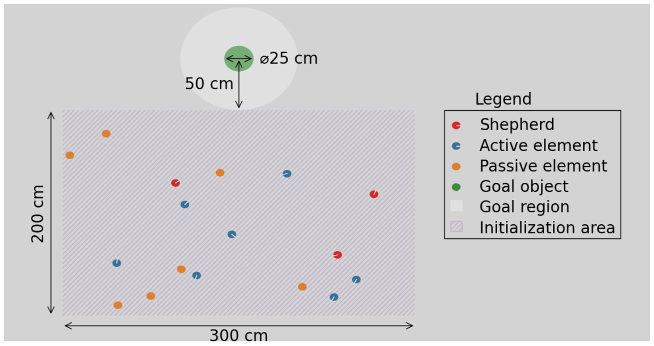

Fig. 1. A group of shepherd agents (red) is tasked to herd a mixture of elements of unknown dynamics towards a goal region (white) near a goal object (green). The elements can be actively moving (blue) or purely passive (orange).

whether the elements to be manipulated are (i) passive, (ii) active, or (iii) a mixture of active and passive elements. We examine to what extent the controllers trained for any one of these sub-problems generalize to the respective other sub-problems, and hence, how the sub-problems compare in terms of complexity.

2 Methods

This section presents the problem formulation, the simulation setup used during design and validation, the control strategies of the shepherd agents, and the optimization process used for obtaining the parameters of the strategies.

2.1 Problem Formulation

The environment is an unbounded, planar, continuous space (see Figure 1). It contains $m \geq 1$ shepherd agents, $n \geq 1$ elements, of which $n_a \geq 0$ are active and $n_p = n - n_a \geq 0$ are passive, as well as a goal object. The shepherd agents and all elements have cylindrical bodies of identical dimensions and mass. The goal object is stationary. It is also cylindrical, and assumed to be taller than the shepherd agents and elements.

Each shepherd agent has two wheels that are placed equidistant from its center. They can be controlled by setting a pair of normalized wheel speeds, $v_\ell, v_r \in [-1, 1]$, where -1 and 1 represent the maximum backward and forward speeds, respectively.

The shepherd agent has two line-of-sight sensors pointing forward, and assumed to have an infinite range.[1] The sensors are discrete; they only return the

[1] Throughout this work, we assume an unlimited sensing range; in practice, however, nearly-identical results can be obtained if the sensing range is limited to a reasonably high value. The effects of a limited sensing range in a similar setting were studied in [8].

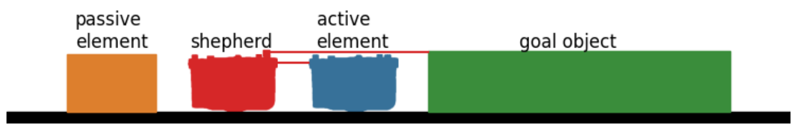

Fig. 2. Illustration of line-of-sight sensor implementation. Each shepherd agent obtains two bits of sensory information. The first indicates whether the goal object is in front of the robot (in the direct line of sight). The goal object is taller than any other object, which allows the shepherd to detect it even when some other agents or elements are placed in between. The second bit of sensory information indicates whether an element is in front of the robot (in the direct line of sight). This is only the case, if the nearest object is an element. The sensor does not distinguish between active or passive elements.

type of the first detected object in their direct line of sight (see Figure 2). The first sensor is used to detect the goal which is taller than the shepherd agents and elements. This allows the shepherd agent to detect the goal if it is oriented towards the goal. The second sensor is used to detect the active and passive elements without distinguishing between them. For the sake of simplicity, we assume that the shepherd agent obtains a single, combined sensor reading,

$$I = \begin{cases} 0, & \text{if neither goal nor any element is detected;} \\ 1, & \text{if only an active or passive element is detected;} \\ 2, & \text{if only the goal is detected;} \\ 3, & \text{if both the goal and an active or passive element are detected.} \end{cases} \quad (1)$$

The objective for the shepherds is to herd all elements toward the goal. We define a goal region around the goal object (see Fig. 1) and assume that an element has been successfully moved towards the goal, as long as its center resides within the goal region at the end of the evaluation period. Note that the goal region is not detectable by any of the agents, it merely serves for evaluation purposes.

We consider three variants of the problem, also referred to as scenarios:

1. *Active only* scenario: all elements are active ($n_a = n$);
2. *Passive only* scenario: all elements are passive ($n_p = n$);
3. *Combined* scenario: elements of both types are present ($n_a \geq 1, n_p \geq 1$).

2.2 Setup for Computational Experiments

Open-source robot simulator Enki [14] was used for all computational experiments. All bodies are rigid. Their dynamics and kinematics are updated every 0.01 s. The sensors, control cycle, and actuation are updated every 0.1 s. The goal object is a cylinder of 12.5 cm radius and 5 cm height. The goal region is a disk of radius 50 cm. The shepherd agents are modelled as e-puck robots [17], which have cylindrical bodies of 3.7 cm radius and 4.7 cm height, and weigh 152 g. The active elements are modelled as e-puck robots too. The passive elements are cylindrical bodies of identical dimensions and mass. Their friction coefficient with the ground is 2.5.

The dynamics model of active elements are loosely inspired by the boids model [19]. It comprises three behavioral components:

1. To weakly repel from nearby active and passive elements;
2. To strongly repel from any nearby shepherd agent;
3. To move randomly.

All components rely only on local sensing: Active element i has two neighborhoods. The first, denoted by $\mathcal{N}_i^{\mathrm{el}}$, comprises all other elements that are no more than $d^{\mathrm{el}} = 10\,\mathrm{cm}$ away. The second, denoted by $\mathcal{N}_i^{\mathrm{sh}}$, comprises all shepherds that are no more than $d^{\mathrm{sh}} = 50\,\mathrm{cm}$ away. The repulsion components can then be expressed as

$$\mathbf{F}_i = k^{\mathrm{el}} \sum_{j \in \mathcal{N}_i^{\mathrm{el}}} \frac{\hat{\mathbf{r}}_{ji}}{||\mathbf{x}_j - \mathbf{x}_i||^2} + k^{\mathrm{sh}} \sum_{j \in \mathcal{N}_i^{\mathrm{sh}}} \frac{\hat{\mathbf{r}}_{ji}}{||\mathbf{x}_j - \mathbf{x}_i||^2}, \tag{2}$$

where coefficients $k^{\mathrm{el}} = 100$ and $k^{\mathrm{sh}} = 500$ model the strength of repulsion from other elements and shepherds, respectively, $\mathbf{x}_i$ is the position of focal element i, $\mathbf{x}_j$ is the position of any other element/agent within the corresponding neighborhood, and $\hat{\mathbf{r}}_{ji}$ is the unit vector from element/agent j to element i. We assume that the elements are indexed $1, 2, \ldots, n$ and shepherd agents are indexed $n+1, n+2, \ldots, n+m$. The wheel speeds for the active element are then calculated as:

$$\begin{pmatrix} \upsilon_l \\ \upsilon_r \end{pmatrix} = \begin{pmatrix} C_1 & C_2 \\ C_1 & -C_2 \end{pmatrix} \begin{pmatrix} f_x \\ f_y \end{pmatrix}, \tag{3}$$

where $C_1 = 2.0$ is a linear coefficient, $C_2 = 1.3$ is an angular coefficient, and f_x and f_y are the force components of $\mathbf{F}_i$ along the x- and y-axis in the focal element's coordinate frame (with the x-axis pointing towards the front of the robot).

The final behavioral component (random walk) is realized by adding random variables, which follow normal distributions $X \sim \mathcal{N}(0,\ 1)$, to the speed values of each wheel of the active element. Before applying the value to the actuator, it is truncated to half the maximum speed of the e-puck robot (12.8cm/s). Therefore, in the default setup, the speed of the active elements are at most 50% of the maximally possible speed of the shepherd agents.

2.3 Control Strategies of the Shepherd Agents

Each shepherd uses the same controller. The controller is fully reactive, that is, it has no memory to store any values during run-time. It maps sensor reading I directly onto a pair of normalized wheel speeds $\upsilon_\ell, \upsilon_r \in [-1, 1]$.

The complete parameterized controller can be written as $\mathbf{v} = (\upsilon_{\ell_0}, \upsilon_{r_0}, \upsilon_{\ell_1}, \upsilon_{r_1}, \upsilon_{\ell_2}, \upsilon_{r_2}, \upsilon_{\ell_3}, \upsilon_{r_3}), \quad \mathbf{v} \in [-1, 1]^8$, where $(\upsilon_{\ell_0}, \upsilon_{r_0})$ is the left and right normalized wheel velocities when the combined sensor reading $I = 0$ and so on (for a definition of I, see Equation 1).

We design three control strategies—one for each variant of the problem. We refer to them as `Controller A` (active only scenario), `Controller P` (passive only scenario), and `Controller A+P` (combined scenario). The controller variants only differ in the choice of parameter values.

2.4 Optimization Process

To optimize the parameter values of the controller, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [9] is employed. CMA-ES is a stochastic method for optimization of non-linear, non-convex functions with continuous domains. It self-adapts the variance of decision variables and the co-variances between decision variables. In our case, the decision variables are the wheel speed pairs for every possible value of sensory input, that is, $\mathbf{v}$.

CMA-ES is conventionally unbounded, operating in the continuous space $\mathbb{R}^d$, where $d = 8$ is the problem dimension. However, as normalized wheel velocities are considered in the controller design, a way to map $\mathbb{R}^d \mapsto [-1, 1]^d$ is needed. This is achieved by using a sigmoid-based function on each wheel velocity, $sig(v) = \frac{1-e^{-v}}{1+e^{-v}}, \forall v \in \mathbb{R}$.

We set the initial solution to the zero vector $\mathbf{v}^{(0)} = \mathbf{0}$, population size to $\lambda = 20$, and the initial step size to $\sigma^{(0)} = 0.72$. These settings approximate a uniform distribution over $[-1, 1]^d$, as empirically demonstrated by Gauci et al. [7] using Monte Carlo simulations. Each evolution runs for 500 generations.

Fitness Function To evaluate the utility of candidate solutions, a fitness function is used. For the problem considered here, the fitness function has a dual purpose. First, it shall reward candidate solutions that gather the elements, thereby providing cohesion. Second, it shall reward candidate solutions for moving elements near to the goal. A corresponding metric is established, which at time t is given by

$$f(t) = \frac{1}{4nr^2} \sum_{j=1}^{n} ||\overline{\mathbf{x}}(t) - \mathbf{x}_j(t)||^2 \cdot ||\overline{\mathbf{x}}(t) - \mathbf{g}||^2, \tag{4}$$

where r is the radius of the element body, $\overline{\mathbf{x}}(t)$ is the centroid of all elements at time t, $\mathbf{x}_j(t)$ is the position of element j at time t, and $\mathbf{g}$ is the position of the goal object.

Each simulation trial is associated with a weighted sum of the fitness values at times t,

$$F(T) = \sum_{t=1}^{T} t \cdot f(t), \tag{5}$$

where $T = 600\,\mathrm{s}$ is the total evaluation period (in simulated time). The weighted sum rewards the speed at which the elements are gathered and driven towards the goal while also rewarding ‘convergence’ towards a stable configuration.

Table 1. Best controller for each scenario.

Controller		$I=0$	$I=1$	$I=2$	$I=3$
`Controller A` (active elements only)	v_ℓ	0.459	0.995	0.738	-0.163
	v_r	0.983	0.161	-0.958	0.948
`Controller P` (passive elements only)	v_ℓ	0.997	0.925	-0.996	0.995
	v_r	0.632	1.000	-1.000	0.703
`Controller A+P` (combined scenario)	v_ℓ	0.592	1.000	0.729	0.794
	v_r	0.939	0.917	-0.998	0.983

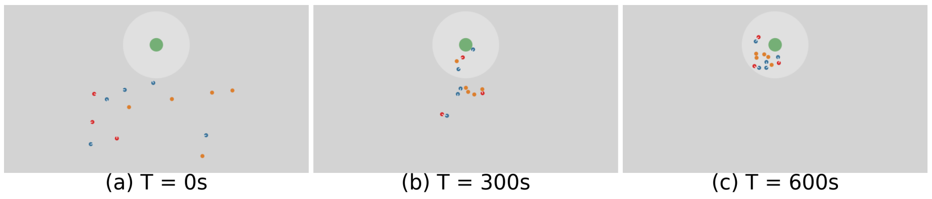

Fig. 3. Sequence of snapshots showing three shepherd agents (red) moving five active (blue) and five passive (orange) elements to the goal object (green). The shepherds use controller `A+P`, which was specifically optimized for this scenario.

The final fitness value is obtained as the mean $F(T)$ score across $N = 20$ independent simulation trials. In each trial, the starting locations of all agents and elements are sampled using a uniform distribution from within the initialization region denoted in Figure 1.

3 Results

We performed 30 evolutionary runs for each of the three problem variants. In all simulation trials, $m = 3$ shepherd agents and $n = 10$ elements were used. The number of active (n_a) and passive (n_a) elements were as follows:

1. $n_a = 10, n_p = 0$ in the active only scenario (to synthesize `Controller A`);
2. $n_a = 0, n_p = 10$ in the passive only scenario (to synthesize `Controller P`);
3. $n_a = n_p = 5$ in the combined scenario (to synthesize `Controller A+P`).

For each scenario and for each of the 30 evolutionary runs, we post-evaluated the highest rated control strategy 100 times with random starting configurations. The best-rated controller from these post-evaluations is considered as the final controller for that scenario.

The evolved control parameters are shown in Table 1. Figure 3 shows a sequence of snapshots taken from a typical trial with `Controller A+P`. The shepherds tend to orbit around the elements and the goal. This helps to gather the elements and move them towards the goal region. A video showing representative

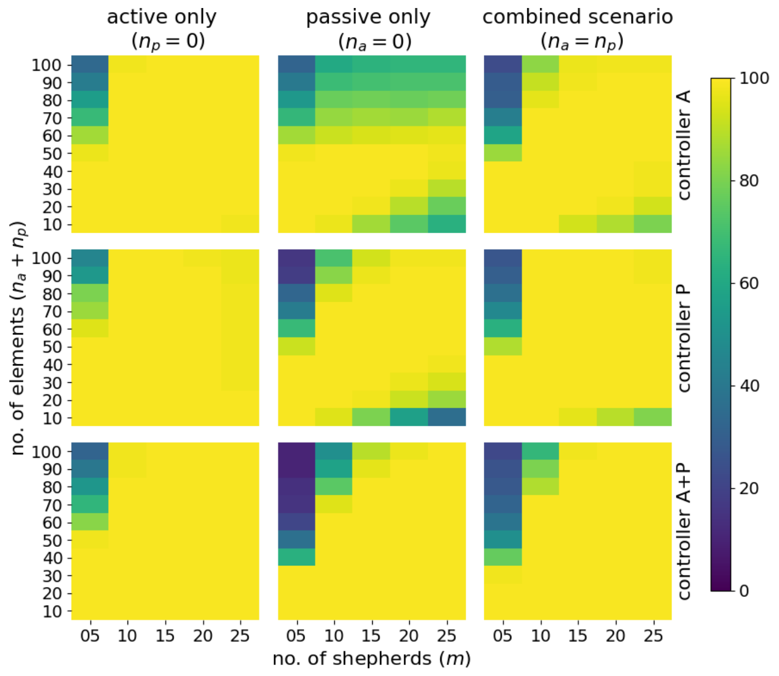

Fig. 4. Generalization and Scalability Analysis. Heat map showing the success rate grouped by controller type and scenario. Average rates over 100 trials in which m shepherd agents herd n_a active elements and n_p passive elements to the goal region.

trials for all controllers and scenarios is available on `https://www.sheffield.ac.uk/naturalrobotics/supp/2022-001`.

3.1 Generalization and Scalability Analysis

Each controller is examined in all three scenarios, thereby testing to what extent it generalizes beyond the specific scenario it was optimized for. For example, `Controller A`, which was trained in the active only scenario, is tested here in all three scenarios, including the passive only scenario and the combined scenario. Moreover, to test scalability of the controllers, a range of configurations is considered. Specifically, the number of shepherds is chosen as $m \in \{5, 10, 15, 20, 25\}$ and the number of elements is chosen as $n \in \{10, 20, 30, \ldots, 100\}$ (for the combined scenario, we use $n_a = n_p = \frac{n}{2}$). Each of these configurations is tested against each of the three controllers. For each setup, 100 independent trials are conducted. Each trial lasts 1500 s.

Figure 4 shows the performance. Reported is the average *success* rate which is defined as the percentage of elements inside the goal region at the end of the trial. When only $m = 5$ shepherd agents are available, the performance of all three controllers decreases as the number of elements $n_a + n_p$ goes beyond 50. This can

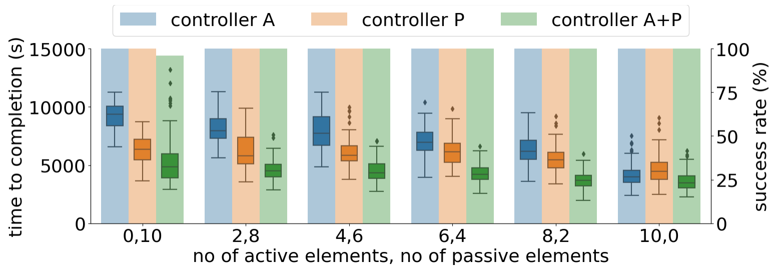

Fig. 5. Varying Ratio Analysis. Times to completion (box plots) and success rates (bar charts) for the three controllers as the ratio of active (n_a) to passive (n_p) elements is varied (100 trials per setup).

be attributed to the limited time available for five shepherds to move a relatively large herd, but possibly as well, though to a lesser extent, to the challenge of containing the herd, while elements move at random. The performance for `Controller A` and `Controller P` also drops when the number of shepherds m is similar to, or even exceeds, the number of elements $(n_a + n_p)$. Moreover, `Controller A` struggles in the passive only scenario when $n_p \geq 60$. This can be attributed to the behaviors being insufficiently optimized for handling passive elements: As the elements are no longer repelled by the shepherd, the latter has to push the elements for them to move. However, we found that this is not an issue in the combined scenario as the active elements help by pushing the passive elements and the passive elements prevent the active elements from dispersing. `Controller A+P` exhibited the best overall performance.

3.2 Varying Ratio Analysis

To analyse the controllers' performance with different ratio of active and passive elements, the latter is varied while keeping the total number of elements constant $(n_a + n_p = 10)$. The number of shepherds is set to $m = 3$. For each setup and controller, 100 independent trials are conducted.

Figure 5 shows the time to completion, that is, the time by which the last element enters the goal region, as well as the average success rates. As the fraction of active elements increases, the times to completion tend to become shorter, especially for `Controller A` and `Controller A+P`. For any pair of active and passive elements (n_a, n_p), `Controller A+P` outperforms the other two controllers in terms of completion times. However, for the $(n_a = 0, n_p = 10)$ pair, its success rate (96%) was slightly lower than that of the other two controllers (100%).

3.3 Varying Speed Analysis

To further examine to what extent the evolved controllers cope with elements of different dynamic properties, we consider the impact of the maximum speed of the active elements. We use $m = 3$ shepherds, $n_a = 10$ active elements and

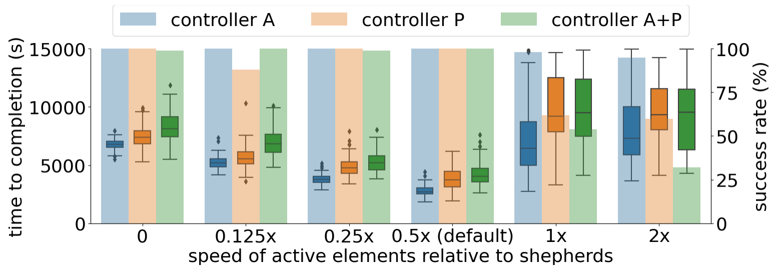

Fig. 6. Varying Speed Analysis. Times to completion (box plots) and success rates (bar charts) for the three controllers as the maximum speed of active elements is varied, expressed relative to the maximum speed of shepherd agents (100 trials per setup).

no passive elements ($n_p = 0$). We choose the maximum speed of the active elements as $0 \times s, 0.125 \times s, 0.25 \times s, 0.5 \times s, 1 \times s, 2 \times s$, where $s = 12.8\,\mathrm{cm/s}$ is the maximum speed of the e-puck robot, and hence the maximum possible speed that any shepherd agent could move. For each setup and controller, 100 independent trials are conducted.

Figure 6 shows the times to completion and the average success rates. The performance is reasonably robust with respect to variations in speed. The best performance both in terms of completion times and success rates is obtained when the active elements use the default maximum speed ($0.5 \times s$). This could be because the controllers were specifically optimized for this setup. However, it is also plausible that when the active elements are too slow, they require to be *pushed* which may prove slightly less effective, whereas when the active elements are too fast, they may disperse faster (using Eq. 2), and hence make it more challenging to be contained by the shepherd agents.

3.4 Noise Analysis

To examine the robustness with respect to sensor noise, we conducted nine sets of experiments, testing each controller on each scenario. Each of the two binary sensor readings is subjected to (i) false-positive noise with probability $p \in [0, 1]$ (i.e. the sensors detect an object even though the object is not there) and (ii) false-negative noise with probability $p \in [0, 1]$ (i.e. the sensors do not detect an object even though it is present).

Figure 7 shows the average success rates. All three controllers were particularly robust to false-negative noise on the goal sensor (solid blue line). To identify which conditions have the most adverse affect on performance, we ranked the conditions (per controller) by the lowest noise level that caused the success rate to drop to 50% (or below). `Controller A` and `Controller P` were most affected by false-positive noise on the goal sensor (irrespective of the type of noise affecting the element sensor). On the contrary, `Controller A+P` was most affected by false-negative noise on the element sensor (irrespective of the type of noise affecting the goal sensor).

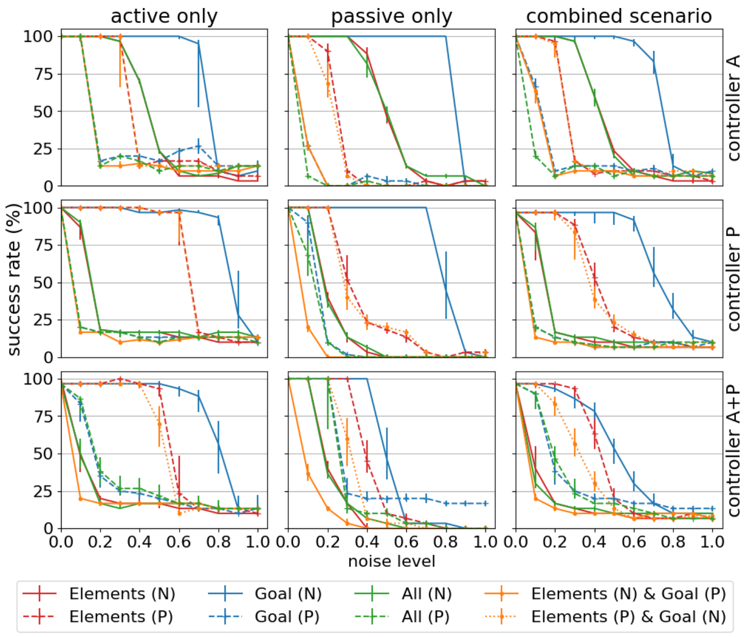

Fig. 7. Noise analysis. Success rates for the three controllers when subject to false-positive (P) and false-negative noise (N); 100 trials per setup.

4 Conclusions

This paper considered for the first time the problem of using a swarm of robots to move a mixture of active and passive elements to a goal region. It showed that this problem can be successfully addressed by robots that have only two binary sensors that detect the presence of the goal and of the elements in front of the robot, without having to distinguish between active and passive elements, and without needing to perceive the other robots in the swarm. Each robot has no run-time memory, and hence, on its own, is unable to learn the unknown dynamics of the elements during run-time.

We evolved three controllers, one for an active elements only scenario, one for a passive elements only scenario, and one for a combined scenario. The controllers generalized well between these scenarios, expect for the controller optimized for the active elements scenario, which did not perform well on the passive elements scenario. The controllers proved flexible, capable of dealing with elements of different dynamics, and reasonably robust to sensory noise. Moreover, their performance scaled well, as validated with up to 25 robots and 100 elements.

In the future, we intend to validate the controllers in physical experiments with e-puck2 robots, and possibly extending the work to 3D environments. Further studies could investigate the evolution of controllers for active elements with non-identical dynamics.

References

1. Becker, A., Habibi, G., Werfel, J., Rubenstein, M., McLurkin, J.: Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. pp. 520–527. IEEE (2013)
2. Beckers, R., Holland, O.E., Deneubourg, J.L.: From local actions to global tasks: Stigmergy and collective robotics. In: Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems. pp. 181–189. MIT Press (1994)
3. Bennett, B., Trafankowski, M.: A comparative investigation of herding algorithms. In: Proceedings of the Symposium on Understanding and Modelling Collective Phenomena (UMoCoP). pp. 33–38 (2012)
4. Chen, J., Gauci, M., Li, W., Kolling, A., Groß, R.: Occlusion-based cooperative transport with a swarm of miniature mobile robots. IEEE Transactions on Robotics **31**(2), 307–321 (2015)
5. Farivarnejad, H., Berman, S.: Multirobot control strategies for collective transport. Annual Review of Control, Robotics, and Autonomous Systems **5**, 205–219 (2021)
6. Fujioka, K., Hayashi, S.: Effective shepherding behaviours using multi-agent systems. In: 2016 IEEE Region 10 Conference (TENCON). pp. 3179–3182 (2016)
7. Gauci, M., Chen, J., Li, W., Dodd, T.J., Groß, R.: Clustering objects with robots that do not compute. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems. p. 421–428. International Foundation for Autonomous Agents and Multiagent Systems (2014)
8. Gauci, M., Chen, J., Li, W., Dodd, T.J., Groß, R.: Self-organized aggregation without computation. The International Journal of Robotics Research **33**(8), 1145–1161 (2014)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**, 159–195 (2001)
10. Kim, J.H., Shell, D.A.: A new model for self-organized robotic clustering: Understanding boundary induced densities and cluster compactness. In: Proceedings - IEEE International Conference on Robotics and Automation. pp. 5858–5863. IEEE (2015)
11. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robotics and autonomous systems **30**(1-2), 85–101 (2000)
12. Lee, W., Kim, D.: Autonomous Shepherding Behaviors of Multiple Target Steering Robots. Sensors (Basel, Switzerland) **17** (2017)
13. Lien, J.M., Rodríguez, S., Malric, J.P., Amato, N.M.: Shepherding behaviors with multiple shepherds. In: Proceedings - IEEE International Conference on Robotics and Automation. pp. 3402–3407 (2005)
14. Magnenat, S., Waibel, M., Beyeler, A.: Enki: An open source fast 2D robot simulator. `https://github.com/enki-community/enki` (2009)
15. Melhuish, C., Holland, O., Hoddell, S.: Collective sorting and segregation in robots with minimal sensing. In: From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior. pp. 465–470. MIT Press (1998)
16. Miki, T., Nakamura, T.: An Effective Simple Shepherding Algorithm Suitable for Implementation to a Multi-mobile Robot System. In: First International Conference on Innovative Computing, Information and Control. vol. 3, pp. 161–165 (2006)

17. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions. vol. 1, pp. 59–65 (2009)
18. Özdemir, A., Gauci, M., Groß, R.: Shepherding with robots that do not compute. ECAL 2017: The Fourteenth European Conference on Artificial Life pp. 332–339 (2017)
19. Reynolds, C.W.: Flocks, Herds and Schools: A Distributed Behavioral Model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. pp. 25–34. ACM (1987)
20. Rojas, J.: Plastic waste is exponentially filling our oceans, but where are the robots? In: 2018 IEEE Region 10 Humanitarian Technology Conference, R10-HTC. pp. 1–6 (2018)
21. Song, Y., Kim, J.H., Shell, D.A.: Self-organized clustering of square objects by multiple robots. Swarm Intelligence pp. 308–315 (2012)
22. Strömbom, D., Mann, R.P., Wilson, A.M., Hailes, S., Morton, A.J., Sumpter, D.J.T., King, A.J.: Solving the shepherding problem: heuristics for herding autonomous, interacting agents. Journal of The Royal Society Interface **11**, 20140719 (2014)
23. Sueoka, Y., Ishitani, M., Osuka, K.: Analysis of Sheepdog-Type Robot Navigation for Goal-Lost-Situation. Robotics **7**(2), 21 (2018)
24. Tsunoda, Y., Sueoka, Y., Sato, Y., Osuka, K.: Analysis of local-camera-based shepherding navigation. Advanced Robotics **32**(23), 1217–1228 (dec 2018)
25. Tuci, E., Alkilabi, M.H., Akanyeti, O.: Cooperative object transport in multi-robot systems: A review of the state-of-the-art. Frontiers in Robotics and AI **5**, 59 (2018)
26. Vaughan, R., Sumpter, N., Frost, A., Cameron, S.: Robot Sheepdog Project achieves automatic flock control. In Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior on From Animals to Animats 5 pp. 489–493 (1998)
27. Vaughan, R., Sumpter, N., Henderson, J., Frost, A., Cameron, S.: Robot control of animal flocks. In: Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell. pp. 277–282 (1998)
28. Wilson, S., Pavlic, T.P., Kumar, G.P., Buffin, A., Pratt, S.C., Berman, S.: Design of ant-inspired stochastic control policies for collective transport by robotic swarms. Swarm Intelligence **8**(4), 303–327 (2014)