Clustering Objects with Robots That Do Not Compute

Melvin Gauci, Jianing Chen, Wei Li, Tony J. Dodd and Roderich Groß
Natural Robotics Lab
Sheffield Centre for Robotics & Department of Automatic Control and Systems Engineering
The University of Sheffield, UK
{m.gauci, j.n.chen, wei.li11, t.j.dodd, r.gross}@sheffield.ac.uk

ABSTRACT

This paper presents a multi-robot solution to the task of object clustering, where the simplicity of the robots is pushed to the extreme that (i) each robot can only detect the presence of (but not the distance to) an object or another robot in its direct line of sight, and (ii) the robots are unable to store previous inputs and cannot perform arithmetic computations. Controllers for the robots were synthesized through an evolutionary robotics approach driven by physics-based simulations. The results show that the problem can be solved even if the robots cannot distinguish between objects and other robots; however, if they are able to make this distinction, the clustering performance is significantly improved. The controllers have been shown to scale well to large numbers of robots and objects and to be robust to noise. The sensor/controller solution was implemented on the e-puck robotic system. Across 10 systematic experiments with 5 robots and 20 objects, on average, 86.5% of the objects were in one cluster after 10 minutes. We believe that the sensor/controller simplicity paves the way for the implementation of multi-robot systems at very small scales, as required, for instance, in nanomedical applications.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence Robotics

General Terms

Algorithms, Experimentation

Keywords

Swarm Robotics, Evolutionary Robotics, Object Clustering, Minimal Information Processing, e-puck

1. INTRODUCTION

Multi-robot systems [21] have attracted much research attention in the last two decades, for a number of reasons. For instance, they are inherently robust to failures, because redundancy can be introduced into the system in the form of additional robots, and therefore the system can still achieve its task in the presence of a few failed units [4]. Moreover, there are many tasks that are inherently distributed, and can thus be performed faster by multi-robot system, which can easily exploit this feature through parallelization [5]. Another often-cited benefit of multi-robot systems is that they allow for the complexity of the individual robots to be reduced, as compared to traditional, monolithic robotic systems. This simplicity can, in some cases, help to reduce the cost of the whole system, but it can also bring about another advantage: allowing for the robots to be scaled down in size for applications such as nanomedicine [19].

While many researchers have successfully implemented multi-robot systems with relatively simple units, few of these systems are simple enough that the robots could conceivably be implemented at scales that are smaller than the currently available technologies. This work contributes to the understanding of what can still be achieved when the complexity of the robots is reduced to a minimal extremum, hence allowing them to be implemented in the future at scales where the space and the energy available for sensing and processing power are immensely scarce [22].

We show that the task of object clustering can be solved by robots that (i) can only detect the presence of (but not the distance to) an object or another robot in its direct line of sight, and (ii) are memory-less, and cannot perform arithmetic computations.

This simplicity comes at the cost of using a longer sensing range than is traditionally assumed in swarm robotic systems [2]; nevertheless the robots only need one sensor (for this task) and, as long as a suitable technology can be used to provide the necessary sensing range, the system has the potential to be truly scalable to small scales. Furthermore, using a minimal amount of long-range information also brings about other advantages, namely (i) the fact that only a minimal amount of features need to be extracted from the environment allows for system designs that can be transferred from simulation to reality with minimal effort [11]; (ii) using information that is not restricted in range allows the system to solve problems that are otherwise unsolvable (see [13], referred to in [12]).

A number of works have presented multi-robot solutions to tasks that involve physical interactions with objects using simple sensors/controllers. Beckers [1] used robots with grippers that are equipped with two infrared sensors and a micro switch that activates when the robot's gripper is pushing more than a certain number of objects. Maris and te Boekhorst [15] used robots with only two infrared sensors that provide the distance to perceived objects or robots. In

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

this work, the objects are constrained to be no larger than the distance between these two sensors, and no clustering behavior was observed with only one robot.

Melhuish et al. [16] used a similar mechanism to the one of Beckers et al. [1] to achieve two-object segregation. The robots in this work also have grippers, and have a micro switch that indicates when the force on the gripper is larger than some threshold value. Furthermore, they have four infrared sensors, as well as an optical sensor that is used to detect the different colors of the objects to be segregated. In a follow-up work, Melhuish et al. [17] showed that this behavior can also be extended to the scenario with more than two groups of objects.

In contrast to the above works, our system makes use of robots that have no grippers, and are only equipped with one sensor, which is an extension of the one presented by Gauci et al. [6], who used it to solve the task of robot aggregation. This sensor can distinguish between objects and other robots, but cannot provide distance information. The robots can therefore only perceive one object or other robot at a time, and, because they do not possess distance information or force sensors, they are unable to know whether they are in contact with the object, as opposed to the case of [1]. Because only one sensor is used, there is no (theoretical) restriction on the size of the objects, as opposed to the work of [15]. The robots are memory-less, and perform no arithmetic computations, which to our knowledge, makes this work the simplest sensor/controller solution to the object clustering problem that has been presented so far.

This paper is organized as follows. Section 2 describes the methods used, including the the objects and the robots, the fitness measure, and the evolutionary algorithm. Sec. 3 explains how controllers were synthesized using the evolutionary algorithm. Sec. 4 presents the results from simulation studies about the emergent behaviors of the synthesized controllers, their scalability with respect to the number of robots in the environment, and their robustness with respect to sensory noise. Sec. 5 explains how the sensor and one of the controllers were ported onto a physical robotic platform, and presents the results from 10 systematic trials. Sec. 6 concludes the paper.

2. METHODS

2.1 Problem Definition

Consider an environment containing $m \geq 2$ cylindrical objects and $n \geq 1$ differential-wheeled robots. The objective for the robots is to bring the objects in the environment together as quickly as possible, into one cluster that is as compact as possible.

2.1.1 Sensor

Each robot is equipped with a line-of-sight sensor I at its front, which indicates to the robot what it is pointing at: $I = 0$ if it is pointing at nothing (or the walls of the environment, if this is bounded), $I = 1$ if it is pointing at an object, and $I = 2$ if it is pointing at another robot. Note that the sensor does not provide the distance to a perceived object or robot.

2.1.2 Controller

The robots have no memory, i.e. they are not able to store previous inputs, and therefore, at each time step t, they must decide on their actions based solely on the current sensor reading, $I^{(t)}$. As $I^{(t)}$ is discrete-valued, only one form of controller is possible: a mapping from each of the three possible values onto a pair of angular velocities for the robot's wheels. Note that any other memory-less controller (e.g. a feed-forward neural network) can be reduced to this form.

Let $\bar{v}_\ell, \bar{v}_r \in [-1, 1]$ represent the normalized angular velocities of the robot's wheels, where -1 and 1 correspond, respectively, to the wheel rotating backwards and forwards with the maximum possible velocity. The controller can now be represented as a six-tuple:

$$\bar{\mathbf{v}} = (\bar{v}_{\ell,0}, \bar{v}_{r,0}, \bar{v}_{\ell,1}, \bar{v}_{r,1}, \bar{v}_{\ell,2}, \bar{v}_{r,2}), \quad \bar{\mathbf{v}} \in [-1, 1]^6, \qquad (1)$$

where $\bar{v}_{\ell,0}$ denotes the normalized angular velocity of the left wheel when $I = 0$, etc.

2.1.3 Fitness Measure

Following Graham and Sloane [8], we use the second moment of the objects as a measure of their dispersion, which we want to minimize. Let r_o represent the radius of one object. Let $\mathbf{p}_i^{(t)}$ represent the position of object i at time t, and let $\bar{\mathbf{p}}^{(t)} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{p}_i^{(t)}$ represent the centroid of the positions of the objects. Then, the second moment of the objects is given by:

$$u^{(t)} = \frac{1}{4r_o^2} \sum_{i=1}^{m} ||\mathbf{p}_i^{(t)} - \bar{\mathbf{p}}^{(t)}||^2. \qquad (2)$$

The $4r_o^2$ in the denominator serves to normalize $u^{(t)}$ such that it becomes independent of r_o for geometrically similar configurations. $u^{(t)}$ does not have an upper bound, because the objects can be arbitrarily dispersed. It has a positive lower bound, because of the physical constraint that the objects cannot overlap with each other, i.e. $||\mathbf{p}_j^{(t)} - \mathbf{p}_i^{(t)}|| \geq 2r_o$, $i \neq j$. Graham and Sloane [8] report lower bounds of $u^{(t)}$ for several values of n up to $n = 499$. Except for $n = 212$, the packings corresponding to the lower bounds of $u^{(t)}$ are optimal among hexagonal packings [3].

Eq. 2 measures the dispersion of the objects at one point in time. We use the following equation to measure the object clustering performance of a controller $\bar{\mathbf{v}}$ when employed on a number of robots for T time steps, $t \in \{0, 1, \ldots, T-1\}$:

$$U(\bar{\mathbf{v}}) = \sum_{t=0}^{T-1} t\, u^{(t)}. \qquad (3)$$

Eq. 3 is designed to reward both a low dispersion at the end of the time interval, as well as the speed with which the robots cluster the objects. It penalizes large values of $u^{(t)}$ at every time instant (except for the very first one), but gives increasing importance to small values of $u^{(t)}$ for increasing values of t.

2.2 Objects and Robots

For the objects, we used cylinders made of expanded polystyrene (EPS), shown in Fig. 1. These cylinders have a diameter and a height of 10 cm. Their mass is approximately 35 g, and their coefficient of static friction with the floor of our arena is approximately 0.58.

We use the e-puck robotic platform [20], which is shown in Fig. 1. The e-puck is a miniature, differential wheeled mobile

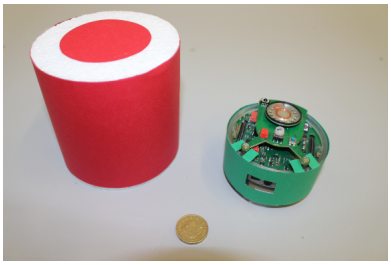


Figure 1: Left: A cylindrical object made of expanded polystyrene (EPS). The object's diameter and height are both 10 cm. The object is wrapped in red paper in order to make it visually distinguishable to the robots. A red marker is also attached to its top, in order to facilitate the tracking of its position by an overhead camera. Right: An e-puck robot fitted with a green 'skirt' that makes it visually distinguishable to the other robots.

robot. Its diameter and height are approximately 7.4 cm and 5.5 cm, respectively, and its weight is approximately 150 g.

The e-puck is equipped with a directional camera located at its front, which has been used in this study to realize the line-of-sight sensor in the physical implementation (see Sec. 5). The e-puck's processor is a Microchip dsPIC microcontroller with 8 KB of RAM and 144 KB of flash memory.

2.3 Simulation Platform

The simulations presented here were performed using the open-source Enki library [14], which is used by Webots™ [18] in 2-D mode. Enki is capable of modeling the kinematics and the dynamics of rigid bodies in two dimensions, and has a built-in model of the e-puck. In Enki, the body of an e-puck is modeled as a disk of diameter 7.4 cm and mass 152 g. The inter-wheel distance is 5.1 cm. The velocities of the left and right wheels along the ground[1] can be set independently in $[-12.8, 12.8]$ cm/s. The objects were modelled as disks of diameter 10 cm, mass 35 g, and a coefficient of friction with the ground of 0.58. The line-of-sight sensor of the robots was realized by projecting a line from the robot's front and checking whether it intersects with the body of an object or another robot (and if it intersects with both, checking with which it intersects first). The length of the control cycle was set to 0.1 s, and the physics were updated at a rate of 10 times per control cycle (i.e. 100 times per second).

2.4 Evolutionary Algorithm

In order to find controllers that optimize the fitness measure defined by Eq. 3, we use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10], which is an algorithm for optimization on real-valued decision vector spaces, $\mathbb{R}^d$. The main feature of CMA-ES is the non-random adaptation of the variance of each decision variable, as well as all the covariances among these variables. CMA-ES is quasi parameter free, because all its internal parameters can be set according to theoretically-justified formulae [9, 10]. The user only needs to specify three exogenous parameters: a starting point for the decision vector, $\mathbf{m}^{(0)}$; the initial step size, $\sigma^{(0)}$; and the population size, λ. CMA-ES is also scale-invariant, because in its selection mechanism, it does not take into account the actual fitnesses of the candidate solutions, but rather selects the μ best solutions from the λ candidate solutions. This is a desirable feature for evolutionary robotics applications, because it ensures that the outcome of the evolution is not overly sensitive to the precise choice of the fitness measure. For a detailed explanation of CMA-ES and all its features, we refer the interested reader to [9, 10]. Below, we will outline the details that are specific to our implementation.

2.4.1 Constraint Handling

In its standard form, CMA-ES operates across the entire real space, $\mathbb{R}^d$; however, in our case, the decision variables need to be constrained within the interval $[-1, 1]$, as explained in Sec. 2.1. In order to achieve this, we let CMA-ES operate in its unconstrained form; however, before evaluating a candidate solution, we map it to the feasible region by applying the following sigmoid-based function to each of the decision variables:

$$\mathrm{sig}(x) = \frac{1 - e^{-(x)}}{1 + e^{-(x)}} \qquad \forall x \in \mathbb{R}. \tag{4}$$

2.4.2 Parameter Settings

We set the starting point for the decision vector, $\mathbf{m}^{(0)}$, to the zero vector, and the initial step size, $\sigma^{(0)}$, to 0.72. Monte Carlo simulations show that with this setting, the initial population will be approximately uniformly distributed in $[-1, 1]^d$, when it is mapped onto this region by the function of Eq. 4.

For the population size, we use $\lambda = 10$, based on the 'default' setting suggested in [9, 10] of $\lambda \approx \lfloor 4 + 3 \ln d \rfloor$ (here, $d = 6$).

3. CONTROLLER SYNTHESIS

We wish to study the importance of the robots being able to detect the presence of other robots, and distinguish them from the objects in the environment. For this reason, we performed 3 sets of evolutions, with 25 evolutions per set, using the following settings:

1. In the first set of evolutions, the robots' sensors work as described in Sec. 2.1, i.e. they can distinguish between pointing at nothing, an object, or another robot[2]. We term this setting *Robots as Robots*.

2. In the second set of evolutions, the robots are unable to detect the presence of other robots. Therefore, when their line-of-sight sensor is pointing at another robot, it gives a reading of $I = 0$ rather than $I = 2$. Consequently, only the first four parameters of the controller of Eq. 1 are utilized. We term this setting *Robots as Nothing*.

3. In the third set of evolutions, the robots are able to detect the presence of other robots; however, they are

[1]This refers to the instantaneous linear velocity of the robot at the point at which the wheel makes contact with the ground. It is equal to the wheel's angular velocity multiplied by the wheel's radius.

[2]Note that in the simulations presented in this paper, no limit was imposed on the range of the sensors. However, the good results obtained in the physical experiments (Sec. 5.3) demonstrate that the system's performance is not undermined if the sensor has a finite, but reasonably long range.

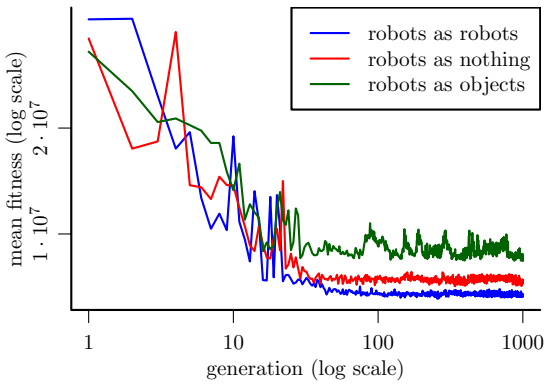


Figure 2: Evolution Fitness Dynamics.

not able to distinguish them from the objects. In other words, their line-of-sight sensor returns $I = 0$ if it is pointing at nothing, and $I = 1$ if it is pointing at an object or another robot. As in the previous case, only the first four parameters of the controller of Eq. 1 are utilized. We term this setting *Robots as Objects*.

Each evolution was run for 1000 generations. Note that for the sake of fairness, all the evolutions operated in a 6-dimensional space (i.e. $d = 6$). In the evolutions for the cases *Robots as Nothing* and *Robots as Objects*, the last two parameters of the controller of Eq. 1 were simply not used by the robots, and therefore had no impact on the selection of the candidate solutions.

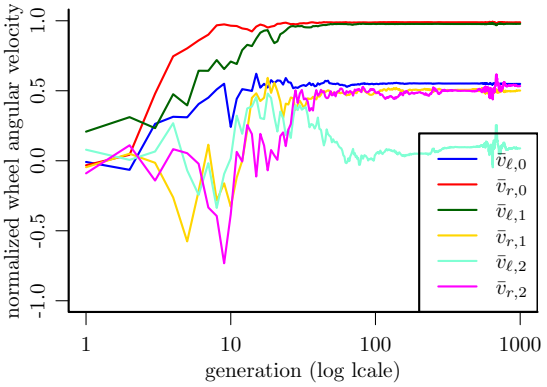
3.1 Evaluation of Candidate Solutions

In each generation, each of the $\lambda = 10$ candidate solutions (i.e. controllers) was evaluated by running it for 100 s on $n = 2$ robots in an environment containing $m = 5$ objects. The objects and the robots were initialized with a uniform distribution in a virtual square of sides 111.80 cm, such that on average, the area per object was 2500 cm^2. Additionally, the initial orientation of each robot was chosen randomly in $[-\pi, \pi]$. Each candidate solution was evaluated 10 times with different initial configurations of the objects and the robots, and the mean value of U (see Eq. 3) across these 10 runs was assigned as its fitness measure. Note that the set of initial configurations was identical for each candidate solution within each generation, but a new set was chosen in every new generation.

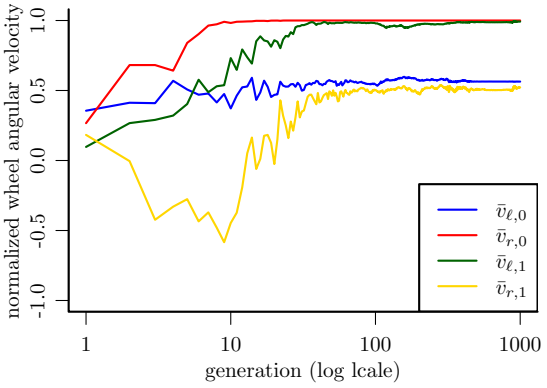
3.2 Post-Evaluations

The selection of the controller for the three cases *Robots as Robots*, *Robots as Nothing* and *Robots as Objects*, was performed as follows. Each of the controllers in the last generation of the respective 25 evolutions was post-evaluated using the same mechanism used within the evolutions (see Sec. 3.1), with the difference that each controller was evaluated 100 times (rather than 10 times) with different initial object/robot configurations. The controller with the highest mean fitness (see Eq. 3) across the 100 runs was selected to be used in the experiments presented in the rest of this paper.

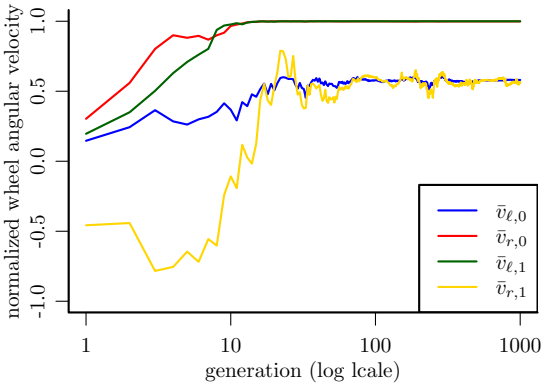
Fig. 2 shows the fitness dynamics of the three 'best' evolutions, i.e. the ones that led to the best controllers. For this plot, the controllers generated by these three evolutions in



(a) Robots as Robots



(b) Robots as Nothing



(c) Robots as Objects

Figure 3: Evolution Parameter Dynamics.

each generation were post-evaluated in the same way as the controllers of the last generation. Fig. 3 shows the dynamics of the parameters in these evolutions (the parameters $\bar{v}_{\ell,2}$ and $\bar{v}_{r,2}$ are not shown in Figs. 3b and 3c, as they are irrelevant). Note that in each of the three cases, the dynamics of the evolutions have reached steady state well before the last generation.

The evolution with *Robots as Robots* led to the best fitness at the last generation (see Fig. 2), followed by *Robots as Nothing* and *Robots as Objects*. From Fig. 3, we see that the first four parameters of the controller (see Eq. 1) in the last generations are very similar across the three evolutions.

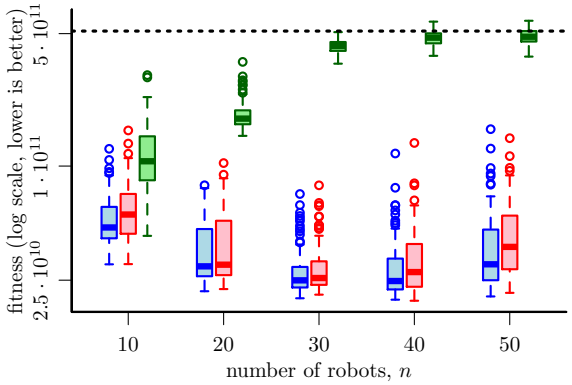


Figure 5: Scalability Study. The blue, red and green boxes correspond to the *Robots as Robots*, *Robots as Nothing*, and *Robots as Objects* controllers, respectively. The dotted black line shows the mean of the baseline fitness measure (i.e. the fitness if robots do not move throughout the run).

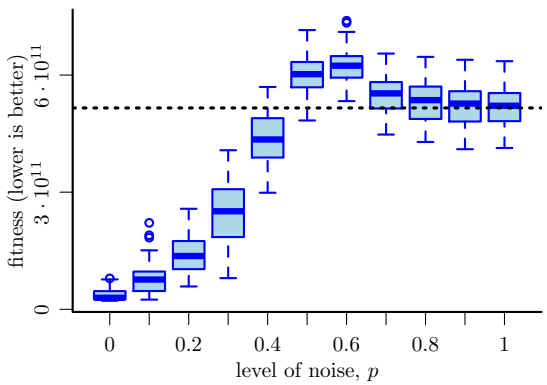


Figure 6: Noise Study. The dotted black line shows the mean of the baseline fitness measure (i.e. the fitness if robots do not move throughout the run).

4. SIMULATION EXPERIMENTS

4.1 Controller Analysis

It turns out that each of the three controllers leads, in a qualitative sense, to the same emergent behavior. Fig. 4 shows snapshots of 50 robots and 20 objects over 1000 seconds where the robots employ the *Robots as Robots* controller (these settings are scaled up by a factor of 10 from the ones used within the evolutions). The robots first find their way to the periphery of (most of) the objects. Following this, they move in a circular formation around the objects, with each robot pushing each object slightly inwards on each contact. Sometimes, some objects are initially not included within the robots' circle, but at some point, the robots branch out from their circular formation to encircle these stray objects as well.

This emergent behavior is robust, as we have observed with many different parameter settings (e.g. number of objects/robots, initial dispersion, and initial configuration). The robots always manage to find their way to the periphery of most of the objects, and after some time, most of the objects end up in one cluster (although sometimes, some objects that are initially very far from the rest are missed, potentially due to the discrete nature of the sensor/controller cycle). Interestingly, the behaviour also works if there is only one robot in the environment.

4.2 Scalability Study

In this section, we study the effect of the number of robots relative to the number of objects on the clustering performance, for each of the three controllers.

We ran simulations using $m = 50$ objects and lasting for 1000 seconds. With each controller, we ran 100 simulations with each number of robots in the set $n = \{10, 20, \ldots, 50\}$, and for each simulation, we recorded the fitness measure given by Eq. 3. Fig. 5 shows a box plot[3] of the results, where the blue, red, and green boxes correspond to the *Robots as Robots*, *Robots as Nothing* and *Robots as Objects* controllers, respectively. The dotted line indicates the expected value of the *baseline fitness measure*, i.e. the fitness measure if the robots do not move throughout the run (found by performing a Monte Carlo simulation with 100 evaluations; note that this does not depend on the number of robots).

With the *Robots as Objects* controller, the performance degrades as the number of robots is increased, until with 40 robots it is not significantly different from the baseline (paired Wilcoxon signed-ranked test, $p < 0.02$). With the *Robots as Robots* and *Robots as Nothing* controllers, the performance has a bowl-shaped profile with respect to the number of robots, with the optimum occurring at $n = 30$ (with the resolution used). However, for each value of n, the performance of the *Robots as Robots* controller is significantly better than that of the *Robots as Nothing* controller.

4.3 Noise Study

In this section, we investigate the effect of sensory noise using the *Robots as Robots* controller. We chose this controller as it is the best-performing one, as shown in the previous section, and as we also use it in our physical experiments (see Sec. 5).

We use the following sensory noise model, which we have found to be realistic from experiments with our physical setup. The sensor always gives the correct reading of $I = 0$ when it is pointing towards nothing (in our physical setup, the walls of the arena). When the sensor is pointing at either an object or another robot, its reading is corrupted with some probability p. In this case, with equal probability, the sensor either misses the object or robot, giving a reading of $I = 0$, or registers the wrong type of item (i.e. a robot if there is actually an object, and vice-versa).

We performed simulations with $p \in \{0, 0.1, \ldots, 1\}$. For each value of p, we performed 100 runs with 50 objects and 20 robots, with each run lasting for 1000 seconds. In each run, we recorded the fitness measure given by Eq. 3. Fig. 6 shows a box plot of the results, where the dotted line indicates the expected value of the *baseline fitness measure*,

[3]The box plots presented here are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles (25-th and 75-th percentiles) of the data, and the whiskers represent the lowest and the highest data points that are within 1.5 times the inter-quartile range from the lower and the upper quartiles, respectively. Circles represent outliers.

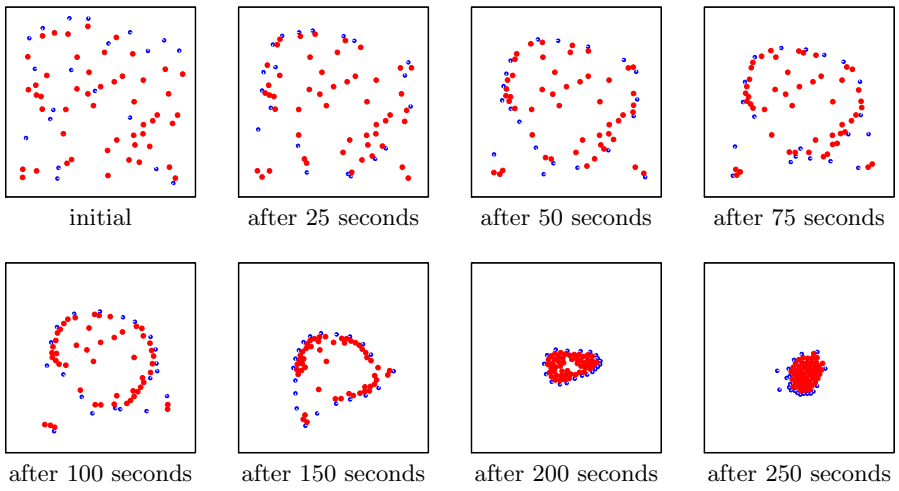


Figure 4: Emergent Behavior. The robots (blue) first find their way to the periphery of (most of) the objects (red). Following this, they move in a circular formation around the objects, with each robot pushing each object slightly inwards on each contact.

i.e. the fitness measure if the robots do not move throughout the run (found by performing a Monte Carlo simulation with 100 evaluations). Up until $p = 0.4$ the performance is significantly better than the baseline (paired Wilcoxon signed-rank test, $p < 0.02$). Interestingly, with $p = 0.5$, 0.6, and 0.7, the fitness is significantly worse than the baseline, meaning that the robots *disperse* the objects more than they were at the start. With $p = 0.8$ and $p = 0.9$, the fitness is still statistically significantly different from the baseline, but the difference is minimal. With $p = 1$, the fitness is not significantly different from the baseline.

5. PHYSICAL EXPERIMENTS

5.1 Sensor and Controller Implementation

The sensor was implemented using the e-puck's directional camera. For this reason, the objects were wrapped with red paper, and the robots were fitted with green 'skirts', in order to make them distinguishable from each other and from the white walls of the arena (see Fig. 1). The e-puck's camera is a CMOS RGB color camera with a resolution of 640 (horizontal) by 480 (vertical) pixels, with corresponding viewing angles of 56° and 42°, respectively. Note that the amount of RAM available on the e-puck's micro-controller is not large enough to even store a single raw color image from the camera, which comes to illustrate the importance of keeping the amount of information processing to a minimum on small-scale robotic systems.

In principle, using one pixel of the camera is sufficient for implementing the line-of-sight sensor. However, in order to account for misalignments among the robots' cameras, as well as the presence of noise in a real-world environment, we used the following method in order to achieve a more robust implementation of the sensor. Firstly, the image from the camera is sub-sampled to obtain a 40×15 pixel image, spanning the whole of the original image. Then, a 10×10 pixel window from the center of this sub-sampled image is used to implement the sensor, as follows. Each pixel is compared against a threshold to decide whether it is white, red or green. If there are less than 2 green pixels, and less than 2 red pixels, then the sensor gives a reading of $I = 0$, indicating that it is pointing at a wall. Otherwise, the majority of the colored pixels decides whether the robot gives a reading of $I = 1$ or $I = 2$, indicating that it is pointing at a red object or a green robot, respectively (if the number of green and red pixels is equal, green is given precedence). The implemented sensor has been found to provide reliable readings for up to a range of around 150 cm (with some misperceptions).

The *Robots as Robots* controller was implemented on the e-pucks without any modifications. As the sensor and the controller structure are extremely simple, the evolved controller is not very sensitive to the nuances of the environment. This helps to bridge the so-called "reality gap", which is often an issue when making use of evolutionary robotics approaches.

5.2 Experimental Setup and Procedure

The arena used for the experiments is a rectangle of size 400 cm × 225 cm. It has a light gray floor, and is surrounded by white walls that are 50 cm in height. Its floor is marked with a grid of $15 \times 8 = 120$ points, spaced 25 cm from each other and from the walls. For each trial, 25 of these points were chosen randomly to serve as the initial positions of the objects and the robots. With the objects and the robots positioned on these points, an infrared signal was issued to instruct each robot to turn on the spot through a randomly generated portion of a revolution, such that robots now faced in random directions. Another infrared signal was then issued to instruct the robots to start executing the controller. The robots were programmed to stop automatically after 600 s. We performed 10 trials with $m = 20$ objects and $n = 5$ robots. Each trial was recorded by an overhead camera. All the 10 videos are available in the online supplementary material [7].

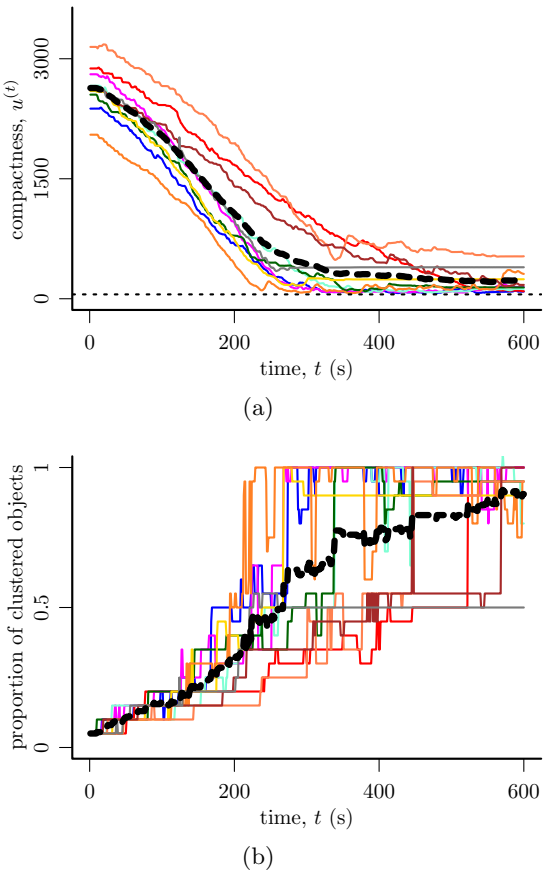


Figure 7: Physical Dynamics.

Figure 8: The Final Configurations of the Objects and the Robots in the 10 Physical Trials.

5.3 Results

Fig. 9 shows a sequence of snapshots from one of the trials. We observe the same emergent behavior as that in simulation (see Sec. 4.1), and in this particular trial, the robots have gathered the objects into one cluster after around 300 s. Fig. 8 shows the final configurations (i.e. after 600 s) of the the objects and the robots in the 10 trials. On average, at the end of the trials, the largest cluster of objects[4] contains 86.5% of the objects.

Fig. 7 shows plots of the clustering dynamics. Fig. 7a shows dynamics of the second moment of the objects (see Eq. 2), while Fig. 7b shows the dynamics of the proportion of objects in the largest cluster. In both plots, the different colored curves correspond to the 10 individual trials, while the dashed black curve represents the mean measure across the 10 trials. In Fig. 7a, the horizontal dotted black line shows the theoretical lower bound of the second moment for 20 objects, as given in [8].

6. CONCLUSION

This paper has presented the simplest solution so far to the problem of clustering objects with a swarm of robots. The robots are memory-less and are unable to perform arithmetic computations. They are only able to detect the presence of an object or another robot in their line of sight. As they do not possess any distance information about a perceived object (nor any force sensors), they are unable to tell whether they are manipulating it. Despite these limitations, we identified a controller that is capable of consistently gathering the objects into a single cluster. Simulation results have shown that the ability of the robots to distinguish between objects and other robots is beneficial; indeed, if the robots can only perceive other robots as objects, the behavior does not scale well with increasing numbers of robots. Simulations have also shown that the controller is fairly robust with respect to sensory noise. The sensor/controller solution was implemented on a physical system of 5 e-puck robots, and favourable results have been obtained in systematic experiments with 20 objects, with 86.5% of the objects being gathered into one cluster after 10 minutes (average across 10 trials). In the future, we intend to implement a solution of the object clustering task with robots at the sub-millimeter scale.

[4]A cluster is defined as a maximal connected subgraph of the graph defined by the objects' positions, where two objects are considered to be adjacent if another object cannot fit in between them (in other words, objects i and j are adjacent if $||\mathbf{p_j}^{(t)} - \mathbf{p_i}^{(t)}|| < 4r_o$).

Acknowledgments

M. Gauci acknowledges support by a Strategic Educational Pathways Scholarship (Malta). The scholarship is part financed by the European Union - European Social Fund (ESF) under Operational Programme II - Cohesion Policy 2007-2013, "Empowering People for More Jobs and a Better Quality of Life". R. Groß acknowledges support by a Marie Curie European Reintegration Grant within the 7-

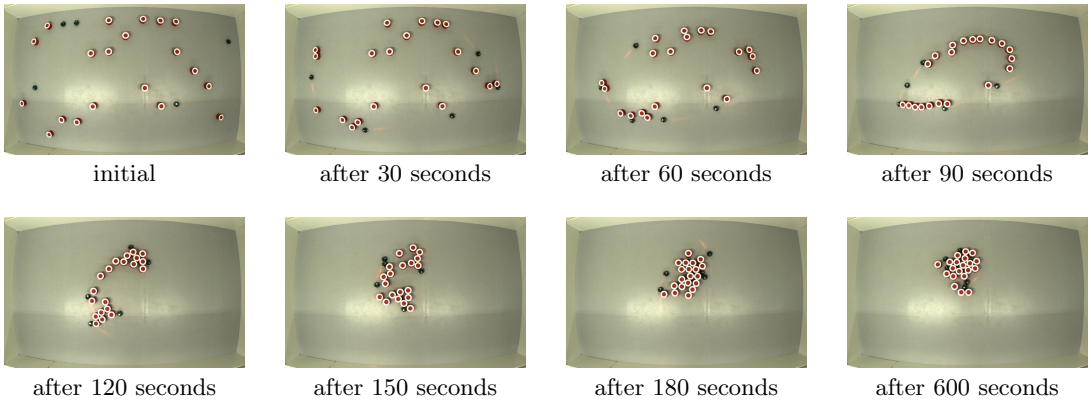


Figure 9: Snapshots From a Physical Trial at Several Time Instances.

th European Community Framework Programme (grant no. PERG07-GA-2010-267354).

7. REFERENCES

[1] R. Beckers, O. E. Holland, and J. L. Deneubourg. From local actions to global tasks: Stigmetry and collective robotics. *Artificial Life*, 4:181–189, 1994.

[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell.*, 7:1–41, March 2013.

[3] T. Y. Chow. Penny-packings with minimal second moments. *Combinatorica*, 15:151–158, June 1995.

[4] A. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Trans. on Evol. Comput.*, 13:754–766, August 2009.

[5] E. Şahin. Swarm robotics: From sources of inspiration to domains of application. In E. Şahin and W. M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20. Springer-Verlag, Berlin & Heidelberg, Germany, 2005.

[6] M. Gauci, J. Chen, T. J. Dodd, and R. Groß. Evolving aggregation behaviors in multi-robot systems with binary sensors. In *Proc. 2012 Int. Symp. Distributed Autonomous Robotic Syst.* In press.

[7] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß. Online supplementary material. `http://naturalrobotics.group.shef.ac.uk/supp/2014-003/`, 2014.

[8] R. L. Graham and N. J. A. Sloane. Penny-packing and two-dimensional codes. *Dicrete Computational Geometry*, 5:1–11, 1990.

[9] N. Hansen. The CMA-ES evolution strategy: A tutorial. `https://www.lri.fr/~hansen/cmatutorial.pdf`, 2011. Accessed 10 October 2013.

[10] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, June 2001.

[11] N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In *Proc. 4th European Conf. Artificial Life*, pages 348–357, Cambridge, MA, USA, 1997. The MIT Press.

[12] E. Klavins. Programmable self-assembly. *IEEE Cont. Syst. Mag.*, 27:43–56, August 2007.

[13] I. Litovsky, Y. Méivier, and W. Zielonka. The power and limitations of local computations on graphs. In *Graph-Theoretic Concepts in Comput. Sci.*, volume 657 of *Lecture Notes in Comput. Sci.*, pages 333–345. Springer-Verlag, Berlin & Heidelberg, Germany, 1993.

[14] S. Magnenat, M. Waibel, and A. Beyeler. Enki: The fast 2D robot simulator. `http://home.gna.org/enki/`, 2011. Accessed 10 October 2013.

[15] M. Maris and R. te Boekhorst. Exploting physical constraints: Heap formation through behavioural error in a group of robots. In *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, pages 1655–1660, 1996.

[16] C. Melhuish, O. Holland, and S. Hoddell. Collective sorting and segregation in robots with minimal sensing. In *Proc. 5th Int. Conf. Simulation of Adaptive Behaviour*, pages 465–470, Cambridge, MA, USA, 1998. MIT Press.

[17] C. Melhuish, M. Wilson, and A. Sendova-Franks. Patch sorting: Multi-object clustering using minimalist robots. In J. Kelemen and P. Sosík, editors, *Advances in Artificial Life*, volume 2159 of *Lecture Notes in Computer Science*, pages 543–552. Springer-Verlag, Berlin & Heidelberg, Germany, 2001.

[18] O. Michel. Webots: Professional mobile robot simulation. *Int. J. Advanced Robotic Syst.*, 1(1):39–42, 2008.

[19] S. M. Moghimi, A. C. Hunter, and J. C. Murray. Nanomedicine: current status and future prospects. *The FASEB Journal*, 19(3):311–330, March 2005.

[20] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Canci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proc. 9th Conf. Autonomous Robot Syst. and Competitions*, volume 1, pages 59–65, 2009.

[21] L. E. Parker. Multiple mobile robot systems. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, Springer Handbooks, pages 921–941. Springer-Verlag, Berlin & Heidelberg, Germany, 2008.

[22] A. Requicha. Nanorobots, NEMS, and nanoassembly. *Proc. IEEE*, 91:1922–1933, Nov 2003.