2012 Ninth Conference on Computer and Robot Vision

Accelerated Patch Sorting by a Robotic Swarm

Andrew Vardy
Department of Computer Science
Faculty of Engineering & Applied Science
Memorial University of Newfoundland
St. John's, Canada
av@mun.ca

***Abstract*—We introduce a new method for distributed object sorting by a swarm of robots. The patch sorting task involves pushing randomly distributed objects into homogeneous clusters. Most existing methods do not make use of vision and are therefore restricted to sensing the objects that lie immediately in front of the robot. We utilize vision both to sense the presence of a cluster and judge its homogeneity, and to seek out distant clusters or isolated objects to pick up. The objects to be sorted are coloured pucks. We present results using a realistic simulation which shows that a simple guidance strategy based on the size of distant clusters can dramatically accelerate the sorting process.**

***Keywords*-swarm robotics; biologically-inspired robotics; patch sorting; clustering**

I. Introduction

A common task in many industrial processes is sorting different materials. Minerals can be sorted by running crushed rock through a conveyor and ejecting the undesired varieties (e.g. separating feldspar from granite). Sorting is central to the recycling industry where magnetism, reaction to forced air, and filtering by size and weight are widely-used techniques. The machines used for these sorting tasks are large, costly to run, and generally inflexible once installed. Insects such as bees and ants also sort materials into groups. Honeybees distribute honey, pollen, and brood in a concentric pattern with brood cells surrounded by pollen, and pollen cells surrounded by honey-filled cells [1]. Ants also organize their brood in a concentric pattern according to size [2]. They also appear to cluster waste material (e.g. dead ants) into groups [3]. We are interested in borrowing the strategies used by social insects to sort physical assets into segregated clusters. The long-term goal is to apply these strategies in applications such as recycling and perhaps inherit some of the robustness and adaptability inherent in insect sorting.

This work is directly inspired by Deneubourg et al who proposed a model for the general capability of insects to gather like materials into larger and larger clusters [3]. The individual agents in this model pick up and drop objects with probabilities governed by the local object density, which is assessed by determining the frequency of object appearance in short-term memory. Isolated objects have a high probability of being picked up by unladen agents, while agents already carrying objects have a high probability of depositing them in high density areas. Over time this process aggregates the objects into larger and larger clusters. Deneubourg et al also studied the application of this process to multiple object types. In this case, local object density is determined separately for each object type (e.g. a sole red object amidst a green cluster would exhibit low red-density—prompting it to be picked up).

A number of researchers have implemented variants of Deneubourg et al's model, but most of these works have focused on a single object type [4], [5], [6], [7]. Research on applying Deneubourg et al's model to multiple object types has been conducted by Melhuish and colleagues [8], [9], [10] as well as Zhang and colleagues [11], [12]. Melhuish et al introduce some useful terminology to describe the task at hand [8]. *Clustering* is the process of grouping objects together, while *segregation* implies physical separation between differing object types. *Patch sorting* includes both clustering and segregation where the clusters are separate from each other and contain only one object type. *Annular sorting* also creates segregated clusters, only in this case they are arranged in roughly concentric rings. Our focus is on patch sorting using a swarm of distributed robots that do not explicitly communicate.

Whether the task is clustering or sorting, it can be decomposed into proximal and distal responses. The proximal response is the response of the agent to its immediate local environment. In Deneubourg et al's model, the proximal response is a probabilistic function of local object density and the type of object currently carried, if any. The agents in this model also react to collisions with other robots or the boundary of the environment by randomized turns. This collision response can also be considered as part of the proximal response. The distal response governs the reaction of the agent to more distant objects and aspects of the environment. Deneubourg et al's model effectively lacks a distal response. The only effect of distant objects is that they may have influenced the behaviour of the agent *in the past*. The only 'response' evoked is to continue moving ahead in the previously determined direction. The lack of a distal response is common to all of the research described above, save that of Verret et al [12]. Verret et al investigated varying

978-0-7695-4683-4/12 $26.00 © 2012 IEEE
DOI 10.1109/CRV.2012.48

their robots' perceptual range, either directly or by allowing a type of communication between agents where the view of one agent is augmented to include the view of another. Unladen agents in this model are drawn towards pucks that are nearby and appear to be isolated, while robots carrying pucks make a probabilistic decision to drop them near other pucks of the same type. In both cases, a robot's response is governed by objects at a distance. However, Verret et al's work relied upon an overhead camera to identify objects and transform them into the coordinate frames of the robots. Thus, it remains unclear whether their model could be applied using only on-board sensors.

Our work is similar in approach to Verret et al in that an unladen agent should seek an isolated object to pick up, and an agent already carrying an object should seek an existing cluster of the same type at which to deposit. However, we employ only on-board sensing (albeit simulated) using a forward-facing camera. The size of distant clusters (where 'cluster' can also imply a single puck) are determined by grouping adjacent image pixels together and calculating the corresponding ground area. The distal response of our method is governed by this notion of cluster size. The simple rule applied is that an unladen agent should seek the smallest cluster in view, while an agent carrying a puck should seek the largest matching-type cluster.

In the next section we present the algorithms tested. Then in section III we describe our experiments which were conducted in a simulation of our targeted robot platform. Section IV provides the results of these experiments. Sections V and VI provide discussion of the results, future work, and conclusions.

II. Algorithms

We describe two robot control algorithms below, the first of which elicits only a proximal response. This algorithm is therefore referred to as PROXIMAL. The other algorithm generates both a proximal and a distal response and will be referred to as SEEKER. Both consist of an ordered list of behaviours organized in a simplified subsumption architecture [13]. Each behaviour is considered in sequence and has an opportunity to examine the input image and to determine whether it should be activated or not. Once activated a behaviour can maintain control, although we have engineered all behaviours so that they deactivate after a short period.

Both algorithms share a common foundation of three behaviours. The common behaviours as well as those particular to each algorithm are outlined in table I and described in more detail below. Note that if none of these behaviours are active the robot will simply travel forwards at constant speed.

A. Common Behaviours

The common behaviours are *escape*, *colour-back-up*, and *steer-away*.

1) Escape: The *escape* behaviour is intended to help avoid deadlock situations where two or more robots are stuck in some fixed configuration, perhaps with their grippers intertwined. The *steer-away* behaviour described below was to designed to avoid this situation, but gaps in sensing and unforeseen combinations of movements can still allow robots to become intertwined. Our technique is to trigger *escape* probabilistically by drawing a random number on each iteration and comparing it to a threshold τ_e. The threshold itself varies based on the content of the image. We determine p_r, the proportion of active image pixels that correspond to other robots. We then multiply by some constant k to yield the threshold $\tau_e = kp_r$. A value of $k = 0.1$ appears to work well. In this case, if other robots fill 50% of the image, there is a 5% chance on each time step of triggering *escape*. When the behaviour is triggered a random movement is selected and applied for $t_e = 60$ time steps.

Note that this behaviour was not included in the original implementation by Beckers et al, however we have found it to improve performance here.

2) Colour-back-up: The *Colour-back-up* behaviour forms the proximal response to objects. A region of the image, referred to as BZ is monitored and the proportion p_i of pixels of each puck type i is determined. If BZ contains a mix of different coloured puck types then the behaviour will not be triggered. That is, if there is more than one puck type i for which $p_i > 0$, indicating a heterogeneous (i.e. undesirable) cluster. Since no action is taken in response to this heterogeneous cluster lying directly in front of the robot, the default action of moving straight ahead will often have the result of breaking the cluster up as the robot sails straight through it. An example of a heterogeneous cluster visible in BZ is given in figure 1.

The other factor which controls the response of *colour-back-up* is the state of the robot's gripper. In our case, the gripper can contain only one puck. The colour of this puck c is determined by sampling pixel values in the centre of the gripper. If the backup zone, BZ, contains pixel values which match that of the carried puck (i.e. $p_c > 0$) then we compare the proportion of coloured pixels in BZ with a threshold τ_{match}. If $p_c > \tau_{match}$ then a short sequence of actions is executed to deposit the puck at this cluster. The robot moves closer to the cluster, then backs away, leaving the puck behind. Finally, it turns away by a random angle in the range $[90^o, 270^o]$. We refer to this as a *deposit cycle*.

If the colour of the carried puck does not match the pucks that appear in the backup zone (i.e. $p_i > 0 \wedge i \neq c$) then we compare the proportion of coloured pixels in BZ with another threshold $\tau_{no\ match}$. If this threshold is reached then the robot immediately makes a random turn away from the cluster (a *turning cycle*). This is intended to preserve the

Common

Behaviour	Trigger	Action
Escape	Other robots loom large enough to trigger a probabilistic threshold	Random movement for t_e time steps
Colour-back-up	High percentage of same-coloured pucks detected in image region BZ	If carrying a matching puck, the robot pushes it into the cluster, then turns away; Otherwise, the robot turns away from the cluster
Steer-away	Wall or other robots detected within avoidance zones	Steer away from obstacles with a small random chance to stop and make a large random turn

SEEKER

Colour-seek	Some pucks are visible and lie a threshold distance away from other robots	If not carrying a puck, steer towards the most isolated puck visible; If carrying, steer towards the largest matching cluster

Table I
BEHAVIOURS OF THE TWO ALGORITHMS PROXIMAL AND SEEKER. THE PRIORITY OF BEHAVIOURS DESCENDS FROM TOP-TO-BOTTOM.

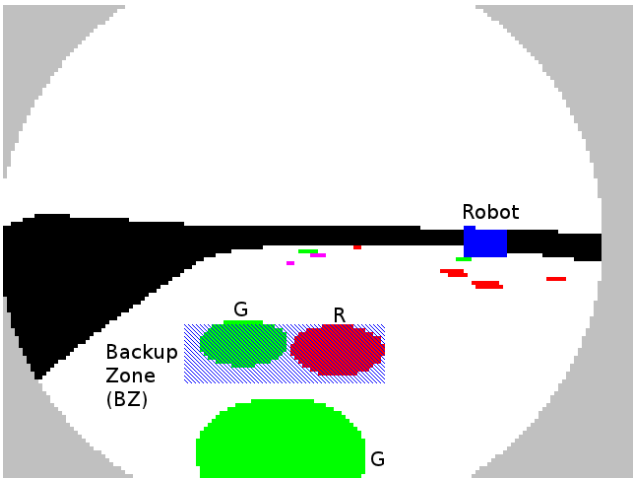


Figure 1. The robot's view of its environment showing pucks in red, green, and purple, another robot in blue, and the boundary of the environment in black. The diagonal hashed lines indicate the image region BZ which is used to trigger the *colour-back-up* behaviour. In this case there are both red (R) and green (G) pixels present in image region BZ indicating that *colour-back-up* should not be triggered. Note that the robot is carrying a green puck. The robot's gripper does not appear in the image.

cluster intact, without modifying it in any way. This aspect of the behaviour is similar to the "pull-back" mechanism used by Melhuish et al [8], [9]. However, since we have freedom in defining BZ we place it such that we can evaluate the cluster without being in direct contact. Therefore, we can turn and leave it intact if the cluster does not match the colour of the carried puck and is sufficiently large ($p_i > \tau_{no\ match}$).

The value of the thresholds τ_{match} and $\tau_{no\ match}$ have a significant impact on performance. Fortunately, there is clear guidance in setting them. τ_{match} governs the minimum cluster size at which to deposit a matching puck. We have found good results in setting τ_{match} so that the threshold is exceeded by a single puck present in BZ. This facilitates early cluster formation. If τ_{match} was set to correspond to two pucks, for example, then a random field of isolated pucks would remain static until two pucks happened to be shifted close enough together, at which point these two pucks would serve as a seed for cluster growth. By setting τ_{match} to correspond to only one puck, we see clusters form whenever a carried puck is moved up to any single matching puck.

The value of $\tau_{no\ match}$ governs the minimum size of cluster that should be preserved when carrying a non-matching puck. For the same rationale as mentioned above, we wish to preserve two-puck clusters. To achieve this, we simply set $\tau_{no\ match} = 2\,\tau_{match}$ (i.e. a value corresponding to two pucks in BZ). It is important to note that this does not guarantee the survival of all clusters containing two or more pucks. If a two-puck cluster is approached from an angle such that both pucks are not fully contained in BZ then the cluster may still be destroyed. This can occur also for larger clusters, or non-compact clusters. Indeed, the destruction of small clusters is crucial to the long-term formation of larger clusters.

Four example scenarios summarizing the responses of *colour-back-up* are shown in figure 2. In scenario A, the cluster in front of the robot is heterogeneous. The behaviour is not triggered and the robot will travel straight ahead, possibly collecting one of the pucks and certainly disrupting the cluster. Scenario B shows a depositing event. The single green puck in the 'cluster' is sufficient to trigger a deposit cycle since the robot is already carrying a matching puck. Scenario C shows the robot in front of a cluster of size three. There is a mismatch between the cluster colour and the carried puck. However, since the cluster lies only partially within the detection zone, it is treated as a cluster of size one, leading the robot to travel forwards, most likely disturbing the cluster. Finally, scenario D illustrates a cluster preserving event. The cluster of two lies within the detection zone. Since the colour does not match the carried puck a turning cycle is triggered which causes the robot to move off in a different direction.

3) Steer-away: The *steer-away* behaviour establishes circular avoidance zones, one for other robots and the other for walls. These are shown in figure 3(a). If another robot or the wall of the environment begins to intersect an avoidance

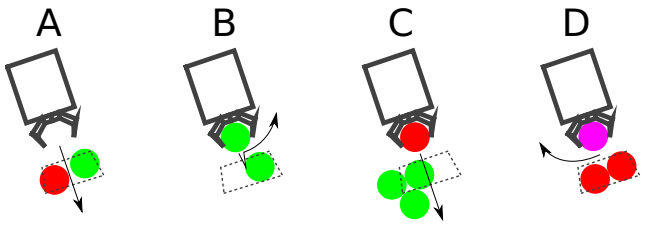


Figure 2. Four scenarios showing possible responses of *colour-back-up*. The ground area corresponding to image area BZ is shown in dotted lines. See text for explanation.

zone, then the robot will steer in the opposite direction. This strategy is implemented by determining the image pixels corresponding to each zone through the calibration procedure mentioned in section III-A. The image regions corresponding to the avoidance zones are shown in figure 3(b). The direction in which to turn is chosen as the opposite of the side that contains more of the obstacle within its avoidance zone.

The avoidance zone for other robots is rather large to mitigate the possibility of colliding and becoming entangled with other robots. The wall avoidance zone is chosen to be quite small (just greater in radius than the robot's width) to allow the robot to skirt the boundary of the environment and collect pucks that would otherwise remain stranded there. To prevent the robot from spending too much time tracing along the boundary we also introduce large on-the-spot turns which are triggered at a probability of 5% whenever another robot or the wall intersects the avoidance zones.

B. PROXIMAL

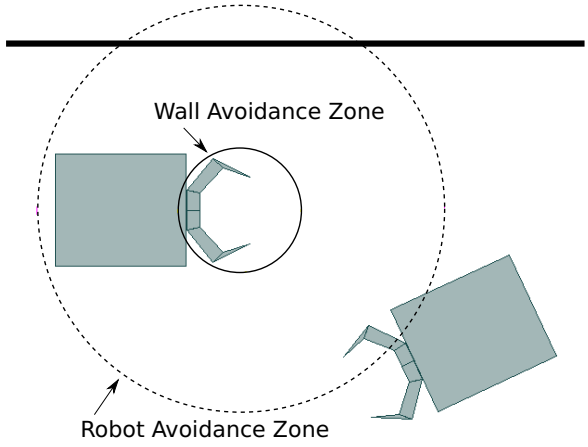
PROXIMAL consists of the three common behaviours of *escape*, *colour-back-up*, and *steer-away*.

C. SEEKER

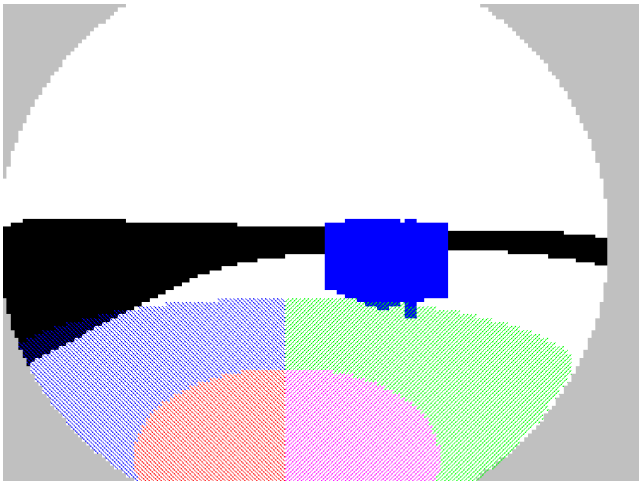
The SEEKER control algorithm adds the *Colour-seek* behaviour. The intention of *Colour-seek* is to accelerate the clustering task by bringing about those events which will affect some change to existing clusters. Those events are picking up isolated pucks and depositing carried pucks at large clusters.

1) Colour-seek: Like *colour-back-up*, the *colour-seek* behaviour samples the pixels centred on the robot's gripper to determine whether a puck is currently being carried and its colour c.

The next step is to identify a target. If the robot is not carrying a puck then the target should be an isolated puck of any colour. We wish to allow the possibility of extracting pucks from existing clusters. This facilitates the destruction of smaller clusters in favour of larger clusters, as mentioned above. Therefore, for an unladen robot we simply target the smallest blob, without considering its colour. If the robot is carrying a puck then the target should be a large cluster that matches the carried puck.



(a) Avoidance zones



(b) Corresponding image regions

Figure 3. Avoidance zones established for the *steer-away* behaviour (a) and the pixels that correspond to these zones (b). The avoidance zones in (b) are shaded differently for the left and right halves.

Potential targets are identified by applying connected components labelling to the image [14]. This process creates blobs of connected pixels of the same colour. See figure 4 for an example. We can easily determine the size of these blobs in pixels, but due to perspective distortion this would not be an accurate measure of the number of pucks involved. Instead we utilize pre-computed values for the ground-plane area imaged by all pixels (see section III-A). Roughly speaking, this area increases with distance from the robot's camera.

Once the blobs have been extracted, one is selected as the target. If the robot is already carrying a puck, the largest matching colour blob is chosen. If not carrying, the smallest blob is chosen. If no blobs are present then the *Colour-seek* behaviour will remain inactive. When active *colour-seek* will steer the robot in the direction of the target blob, so that it

is centred and approached. Once the cluster (possibly just one puck) corresponding to the blob is reached, the *colour-back-up* behaviour will determine the robot's response. Thus, SEEKER separates the *proximal* and *distal* responses.

There are two additional phenomena that affect this process. One occurs when two blobs are nearly equal in size: the *colour-seek* behaviour can cause the robot to oscillate such that it actually travels between the two clusters. Therefore we introduce a maximum on the allowed shift of the target between frames. This introduces sufficient hysteresis to allow the robot to adhere to the same target. The other phenomenon that can occur is one robot seeking the puck carried by another. This creates the opportunity for collision and does little to enhance clustering. Hence, we filter the extracted blobs and exclude any that lie within a threshold image distance of any other robot.

The example shown in figure 4 illustrates several aspects of *colour-seek*. The robot is not carrying a puck in this case, so it is seeking an isolated puck to pick up. There are three isolated red pucks visible and one cluster of three green. The top-leftmost puck is within the threshold distance of another robot and is therefore not extracted as a blob. Of the two remaining isolated pucks, the one to the right has a lower ground area (by one unit). It is therefore targeted and the robot moves towards it.

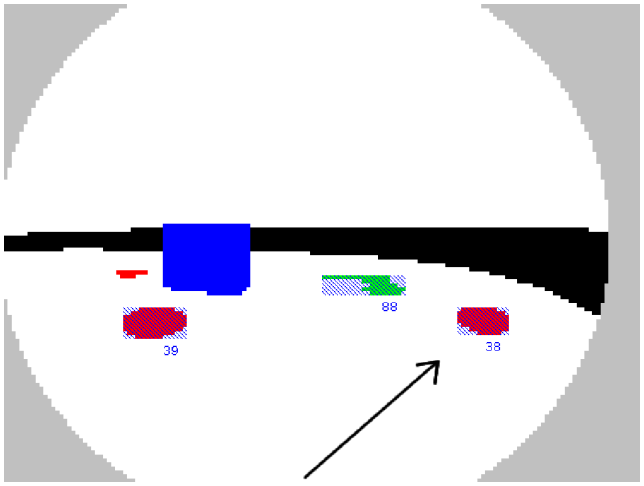


Figure 4. Blobs extracted from an input image. The hatched areas show the blobs with numbers underneath each one corresponding to the blob's ground area.

III. EXPERIMENTS

Experiments were conducted using a custom simulation environment to determine if SEEKER enhances patch sorting performance over PROXIMAL. The robot platform modelled by the simulation is the SRV-1[1]. The SRV-1 is a differentially steered tracked vehicle measuring 12.5×10.8 cm. Most important for this application, it includes a forward-facing camera and a reasonably capable processor (500MHz Blackfin). The standard camera lens has been replaced with a fisheye lens which provides a horizontal field of view of 187^o along the image equator.

The simulator was written in Java and utilizes the JBox2D[2] physics library. The environment modelled is an air hockey table with two rounded ends. The table's dimensions are 177×111 cm. The shape of the table facilitates the sorting task in that there are no corners for pucks to become trapped in. The pucks are modelled after the actual air hockey pucks used on this table and measure 6.3 cm in diameter.

In our experiments the robots and pucks are randomly positioned within the environment. Each trial uses a different seed for the random number generator, which yields both different initial conditions and different sequences of robot actions.

A. Calibration

The geometry and optics of the camera system have been calibrated via the OCamCalib omnidirectional calibration toolbox [15] so as to determine the three-dimensional ray corresponding to each image pixel. We can then determine the intersection point of each such ray with the ground plane. Knowing these intersection points allows us to compute the distance of viewed objects (or at least, the distance of their nearest points along the ground plane) as well as the area imaged by each pixel.

B. Performance Metrics

Various metrics can be used to judge the performance of distributed clustering and sorting algorithms: number of clusters [4], [5], size of the largest cluster [4], mean cluster size [5], [6], and spatial entropy [16]. For sorting in particular, a metric for completion was introduced by [8] whereby completion is declared when some fixed percentage of all pucks lie within the largest cluster of their particular colour. In our results we determine cluster membership by writing the puck positions into a coarse grid and applying connected components labelling to identify groups of adjacent pucks.

IV. RESULTS

We first determine an appropriate number of robots for further experiments. There exists a balance between the number of robots in play and their collective performance. Beckers et al found optimal performance for three robots in their experiments [4]. The optimal number of robots to deploy is clearly dependent upon the size of the environment and the scope of the task. We tested performance for 2-5 robots when sorting three groups of 10 pucks in colours red, green, and pink. Performance is measured at the end of 60,000 time steps as the percentage of pucks that lie within

[1]See http://www.surveyor.com.

[2]See http://jbox2d.org.

the largest homogeneous cluster for each colour. Figure 5 shows the resulting completion percentages. Note that for all tested robot quantities SEEKER outperforms PROXIMAL on average. The peak in performance for SEEKER is four robots. However, PROXIMAL peaks at three robots so we will use three robots for subsequent experiments.

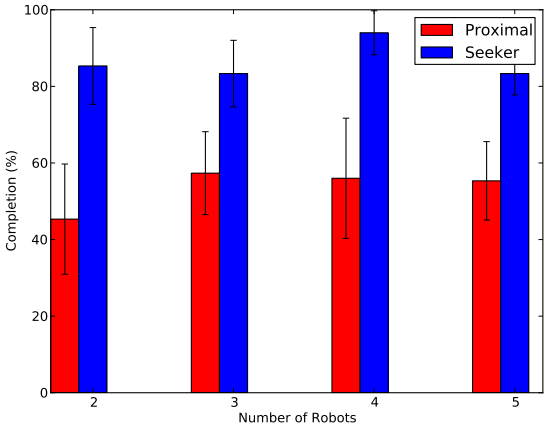
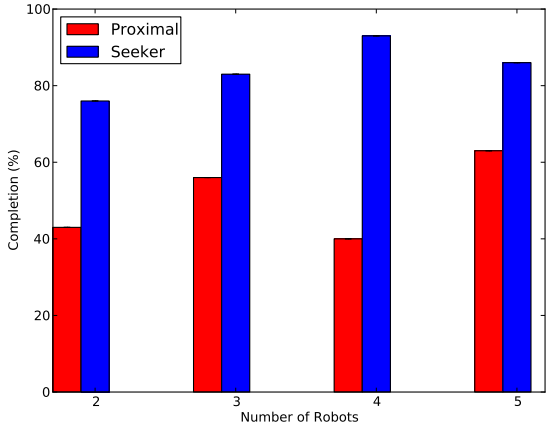


Figure 5. Average percentage completion of sorting for 2-5 robots acting upon 10 red, 10 red, and 10 pink pucks. The error bars are positioned at $\pm$ one standard deviation.

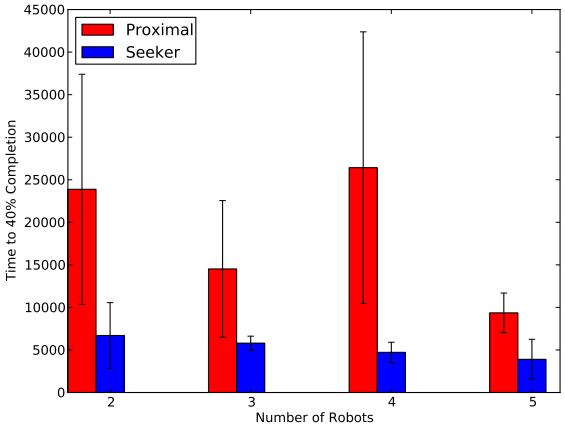
Figure 6 shows the state of sorting for three robots after 30,000 and 60,000 time steps. The difference between the performance of PROXIMAL and SEEKER is particularly clear at 30,000 time steps where there remains significant mixing between the three colours for PROXIMAL.

We must determine a particular percentage of completion in order to compare the sorting speed of our two methods. 100% completion implies one cluster per object type. This is achieved for some trials as can be seen in figure 6. For SEEKER at 60,000 steps 100% completion is apparent in the second and fourth rows and the other trials are quite near completion. Note that in the experiment below we do not require that 100% completion be maintained. Indeed, consider the second row for SEEKER at 60,000 steps. One green puck is out of place because it is being carried by a robot even though 100% completion was previously achieved. For PROXIMAL 100% completion is not achieved at all. Therefore, we must establish some minimum level that both methods can achieve.

For each number of robots from 2-5 we search for the highest percentage completion that is reached or exceeded in all five trials (i.e. the minimum percentage completion over all five trials). Figure 7(a) shows these results. In all cases SEEKER achieved a higher minimum degree of completion over PROXIMAL. The highest percentage completion that could be achieved for both methods on all trials was 40%. We then analyzed the time required by all methods to reach this minimum level of completion. Figure 7(b) shows these results which illustrate the dramatic acceleration of sorting achieved by SEEKER.



(a) Minimum completion achieved over all trials



(b) Time to 40% completion

Figure 7. (a) Minimum degree of completion over five trials and (b) time to reach 40% completion.

V. DISCUSSION AND FUTURE WORK

The most important next step for this research is to verify the findings presented here using real robots. We have attempted to model the geometry of our physical setup, as well as the optics of the robot camera systems with some fidelity. However, there may be aspects of the physical system which will affect performance in unexpected ways.

Another interesting direction for future work is developing a method which guarantees 100% completion of the sorting task. Kazadi et al have studied the convergence properties of single object-type clustering and have established conditions for eventual convergence to a single cluster [7]. Extending

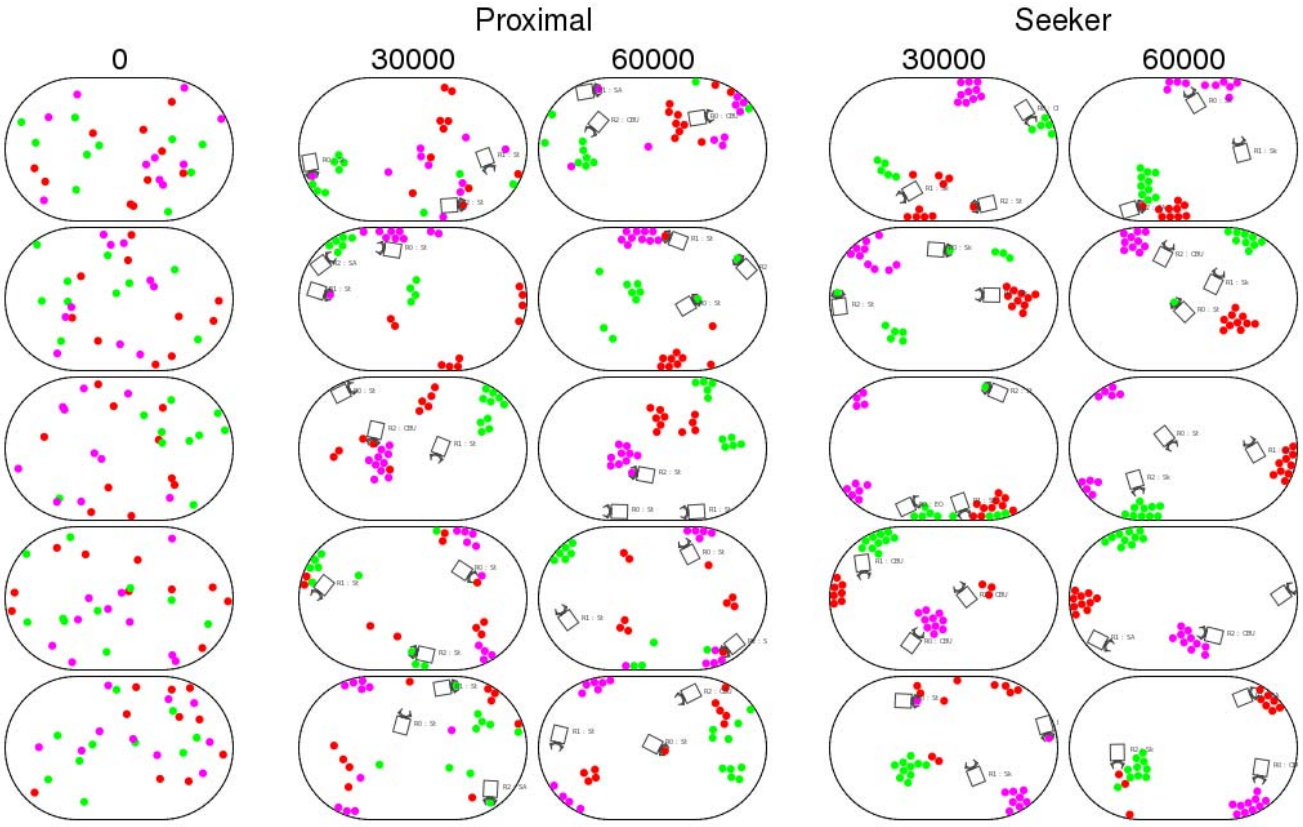


Figure 6. Snapshots of sorted pucks after 30,000 and 60,000 time steps using three robots. The initial conditions for each trial are shown at left. Note that overlap in the initial positions of pucks is quickly resolved by the physics engine.

this model to multiple object types is one challenge. Perhaps more significant is incorporating the distal response of a method such as SEEKER. Methods which lack a distal response follow random trajectories. We have made the case here that sorting can be accelerated by actively seeking out pucks for pick-up or clusters at which to deposit. However, analyses such as Kazadi et al's take advantage of the random nature of robot movement. It is not clear if the same results can be applied when the robot's movements are much more dependent on the state of the environment.

VI. CONCLUSIONS

We have demonstrated here how the addition of one relatively simple behaviour can accelerate the sorting performance of a swarm of distributed robots. The only difference between PROXIMAL and SEEKER is the addition of the *colour-seek* behaviour, which adds distal visual guidance to bring the robot more quickly into contact with isolated pucks or large clusters.

Other swarm robotics researchers working on the ideas pioneered by Deneubourg et al seem to have avoided the use of vision, at least for sensing and reacting to distant parts of the environment[3]. This bias may be due to the belief that adherence to the social insect metaphor requires extreme minimalism in sensing and perception. However, social insects such as ants and bees make quite significant use of visual cues. Vision is crucial, for example, in their navigational strategies [17]. We intend to further probe the limits of what can be achieved in swarms of simple robots that make significant use of vision.

REFERENCES

[1] M. Beekman, G. A. Sword, and S. J. Simpson, "Biological foundations of swarm intelligence," in *Swarm Intelligence*, ser. Natural Computing Series, C. Blum and D. Merkle, Eds. Springer Berlin Heidelberg, 2008, pp. 3–41.

[2] A. B. Sendova-Franks, S. R. Scholes, N. R. Franks, and C. Melhuish, "Brood sorting by ants: two phases and differential diffusion," *Animal Behavior*, vol. 68, pp. 1095–1106, 2004.

[3]To our knowledge, the only models which utilize vision are due to Melhuish et al,where the camera is angled downwards to judge the density of objects directly in front of the robot [10], and Verret et al who employ an overhead camera [12].

[3] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The dynamics of collective sorting robot-like ants and ant-like robots," in *First Int. Conf. on the Simulation of Adaptive Behaviour*. Cambridge, MA, USA: MIT Press, 1990, pp. 356–363. [Online]. Available: http://dl.acm.org/citation.cfm?id=116517.116557

[4] R. Beckers, O. Holland, and J.-L. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," in *Artificial Life IV*, R. Brooks and P. Maes, Eds., 1994, pp. 181–189.

[5] M. Maris and R. Boeckhorst, "Exploiting physical constraints: heap formation through behavioral error in a group of robots," in *IEEE/RSJ International Conference on Robots and Systems (IROS)*, 1996.

[6] A. Martinoli, A. Ijspeert, and L. Gambardella, "A probabilistic model for understanding and comparing collective aggregation mechanisms," in *Advances in Artificial Life*, ser. Lecture Notes in Computer Science, D. Floreano, J.-D. Nicoud, and F. Mondada, Eds. Springer Berlin / Heidelberg, 1999, vol. 1674, pp. 575–584. [Online]. Available: http://dx.doi.org/10.1007/3-540-48304-7_77

[7] S. Kazadi, A. Abdul-Khaliq, and R. Goodman, "On the convergence of puck clustering systems," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 93–117, 2002.

[8] C. Melhuish, O. Holland, and S. Hoddell, "Collective sorting and segregation in robots with minimal sensing," in *5th Int. Conf. on the Simulation of Adaptive Behaviour*, 1998.

[9] C. Melhuish, M. Wilson, and A. B. Sendova-Franks, "Path sorting: Multi-object clustering using minimalist robots," in *Advances in Artificial Life - Proceedings of the 6th European Conference on Artificial Life (ECAL)*, 2001.

[10] C. Melhuish, A. B. Sendova-Franks, S. Scholes, I. Horsfield, and F. Welsby, "Ant-inspired sorting by robots: the importance of initial clustering," *Journal of the Royal Society: Interface*, vol. 3, no. 7, pp. 235–242, 2006.

[11] T. Wang and H. Zhang, "Multi-robot collective sorting with local sensing," in *IEEE Intelligent Automation Conference (IAC)*, 2003.

[12] S. Verret, H. Zhang, and M. Q.-H. Meng, "Collective sorting with local communication," in *IEEE/RSJ International Conference on Robots and Systems (IROS)*, 2004.

[13] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[14] R. Gonzalez and R. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall, 2002.

[15] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easy calibrating omnidirectional cameras," in *IEEE/RSJ International Conference on Robots and Systems (IROS)*, 2006.

[16] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxf, 1999.

[17] T. Collett and M. Collett, "Memory use in insect visual navigation," *Nature Reviews Neuroscience*, vol. 3, pp. 542–552, 2002.