Mapping, Foraging, and Coverage with a Particle Swarm Controlled by Uniform Inputs

Arun Mahadev, Dominik Krupke, Sándor P. Fekete, and Aaron T. Becker

***Abstract*— We propose a novel approach to mapping tissue and vascular systems without the use of contrast agents, based on moving and measuring magnetic particles. To this end, we consider a swarm of particles in a 1D or 2D grid that can be tracked and controlled by an external agent. Control inputs are applied uniformly so that each particle experiences the same applied forces. We present algorithms for three tasks: (1) *Mapping*, i.e., building a representation of the free and obstacle regions of the workspace; (2) *Foraging*, i.e., ensuring that at least one particle reaches each of a set of desired locations; and (3) *Coverage*, i.e., ensuring that every free region on the map is visited by at least one particle. These tasks relate to a large body of previous work from robot navigation, both from theory and practice, which is based on individual control.**

We provide a spectrum of theoretical and practical new insights that have particular relevance for fast MRI scans with magnetically controlled contrast media. In particular, we develop a fundamentally new approach for searching for an object (a *membrane* in the context of tissue) at an unknown distance D, where the search is subject to two different and independent cost parameters for *moving* and for *measuring*. We show that regardless of the relative cost of these two operations, there is a simple $O(\log D/\log\log D)$-competitive strategy, which is best possible. We extend this to to other settings. In addition, we provide alternative, practically useful strategies for higher-dimensional settings, as well as experimental results. These algorithms extend to any number of particles, and show that additional particles tend to reduce the mean and the standard deviation of the time required for each task.

I. Introduction

In MR imaging, some tissues have poor *contrast*, which means that the boundaries between tissue types cannot be determined. To overcome this, particulate solutions of a contrast agent are used to illuminate regions of interest [1]. Drawbacks include that the contrast agent diffuses quickly and must be injected repeatedly during long scans. Additionally, many contrast agents such as Gadolinium chelates are toxic, and prolonged exposure causes medical complications [2]. This paper explores using steerable magnetic microparticles to map a region. These particles can be steered by the global magnetic gradient of an MRI and visualised by the MRI [3], even when the tissues they move through have poor contrast. As a current example for micro- and nano-particles that can be manufactured in large numbers, see [4]–[10].

*This work was supported by the National Science Foundation under Grant No. [IIS-1553063] and [IIS-1619278].

A. Mahadev and A. Becker are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4005 USA aviswanathanmahadev@uh.edu, atbecker@uh.edu

S. Fekete and D. Krupke are with the Dept. of Computer Science, TU Braunschweig, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany, s.fekete@tu-bs.de, d.krupke@tu-bs.de

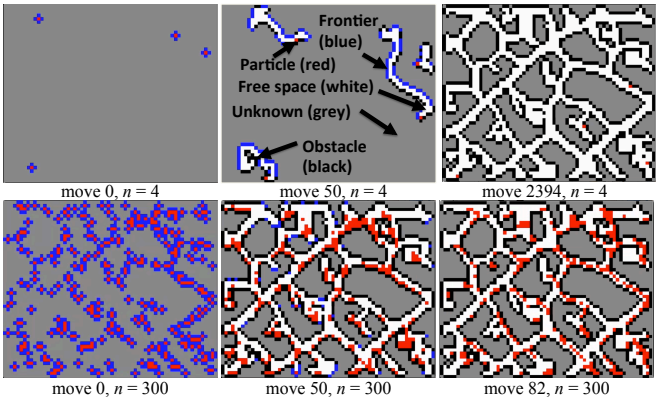

Fig. 1. Mapping a 2D environment with 500 blank spaces using $n = 4$ (top) and $n = 300$ particles, all controlled by a external global force. After 82 moves, the 300 particles have mapped the entire space, while the 4 particles require a total of 2394 moves to fully map the area.

The particles considered in this paper move synchronously under the influence of a global command. Unless they get stopped by an obstacle or another stopped particle, they move by the same vector when the force is activated. Using MRI scans, it is possible to detect the location of particles, but not of the presence of tissue; the key idea is to deduce the presence of obstructing tissue by differences between the expected motion vectors and the measured location of particles.

In previous work [11] we provided an algorithm that guarantees the collection of particles. In this work we explore the field of mapping, coverage, and foraging using globally controlled particles. This paper focuses on discrete 2D workspaces. Fig. 1 represents the complete mapping of a workspace using a large number of particles. At the initial step, all particles (red circles) are in free cells (white squares) and are surrounded by blue squares that represent the unknown frontier cells. By commanding the particles to take one step in a particular direction, we can categorize the the frontier cell in this direction as either obstacle or free. If the particle was able to move, that frontier cell is labelled as free, and new frontier cells are added to adjacent areas that have not been mapped. If the particle was unable to move, that frontier cell is labelled as obstacle. The goal is to explore all frontier cells, thereby discovering all connected free cells and the obstacles that surround them.

The paper is arranged as follows. After a review of recent related work in Sec. II, we introduce the algorithms to perform mapping, coverage, and foraging in Sec. III. Sec. IV

describes implementations of the algorithms in simulation and theoretical validation of efficiency. We discuss the performance of the algorithms on parameters which determine efficiency and also provide directions for further research in Sec. V.

II. Related Work

Coverage using one robot is a canonical robotics problem [12]. It has been studied in-depth for many applications including lawn mowing, harvesting, floor cleaning, 3D printing, robotic painting and others.

Coverage means the robot has passed within $d/2$ of every location in the workspace where d is the diameter of the robot's footprint. Coverage with a swarm of robots is a key ability for a range of applications because swarms have higher fault tolerance and reduce completion time. Correspondingly, it has been studied from a control-theoretic perspective in both centralized and decentralized approaches. For examples of each, see [13], and [14].

Previous methods focus mostly on extending single robot coverage techniques to multi-robot systems. Solving coverage for synchronous multi-robots using on-line coverage techniques such as the boustrophedon technique of subdividing the 2D space into cells as in [15] focuses on moving the robot teams in unison until they identify obstacles in their path. Once that happens, the team divides into smaller teams that continue the search in the smaller cells. This method resembles our approach in the sense that robots try to move in the same direction as long as possible. In our problem of interest the particles always move in the same direction. The frontier cells exploration mentioned in [16] is an algorithm that is highly explorative as target locations to expand are selected using information from each robot. Many algorithms have been developed after this pioneering work, and it showed how algorithms for single robot could be expanded to multi-robot systems. The explorative bias of the technique allows it to define target cells of high priority to explore.

Fundamental problems of robot navigation in an unknown environment have also received a large amount of attention from the theoretical side. The classic prototype is the *linear-search problem*, which was first proposed by Bellman [17] and, independently, by Beck [18]: An (immobile) object is located on the real line at an unknown distance D and in an unknown direction. Because the time necessary for locating the object may be arbitrarily high (as the object may be hidden far from the origin), a useful measure for the performance of a search strategy is the *competitive ratio*: This is the supremum of the ratio between the time the searcher actually travels and the time she would have taken if she had known the hiding place. The competitive ratio is a standard notion in the context of online algorithms; see [19] for a comprehensive overview. For the linear-search problem, the optimal competitive ratio is 9, as was first shown by Beck and Newman [20] and generalized by Gal [21], [22]: The search should alternate between going to the right and to the left, at each iteration doubling her step size. This can be extended to other scenarios: For the case in which changing direction during the course of the search incurs an additional cost of d, Demaine et al. [23] showed that an optimal strategy can achieve locate an object with total cost $9D + 2d$, which is optimal; see Arkin et al. [24] for the generalized offline problem of covering with turn cost. Other related algorithmic work in an unknown setting includes Kao et al. [25] in a randomized setting, Baeza-Yates et al. [26] for searching in the plane; for broad surveys on mathematical methods, see the books by Gal [27] and by Alpern and Gal [28]. Of particular relevance for the content of this paper is Fekete et al. [29], [30] for online searching by an autonomous robot in an unknown environment, where both moving and measuring incur individual costs, and Fekete et al. [31] for an (offline) setting that studies the closely related bicriteria version of covering with travel cost. For another recent work on mapping and coverage by a swarm of robot with limited information and capabilities, see [32].

However, all these approaches assume a level of intelligence and autonomy in individual robots that exceeds the capabilities of many systems, including current micro- and nano-robots. Current micro- and nano-robots, such as those in [4], [33], [34] cannot have onboard computation. Thus, we will be referring to them as *particles*.

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm.

III. Theory

This section examines the problem of mapping with uniform inputs in 1 and 2 dimensions.

A. Mapping in 1D

We begin with the single particle case, then proceed to the n particle case.

1) 1D mapping with 1 particle: A particle is initialized uniformly randomly in a linear free-space m units wide. To *map* this region the particle needs to choose one direction, move until it hits a boundary, and then switch direction and move until it reaches the other boundary.

Without loss of generality, assume the particle always starts going left, and label the free-space from 1 to m left to right. If the initial position is 1, the particle tries to move 1 unit to the left, but is stopped by the boundary. The particle then moves $m - 1$ moves to the right. The final m^{th} move right results in a collision with the right wall, and thus mapping requires $m + 1$ moves. This is the minimum number of moves. The worst case is if the particle starts at m, requiring $2m$ moves: m moves to the left and m moves to the right.

The expected number of moves for one particle to cover a 1D area of length m is

$$\frac{1}{m}\sum_{i=1}^{m}(i+m) = \frac{3m+1}{2} \tag{1}$$

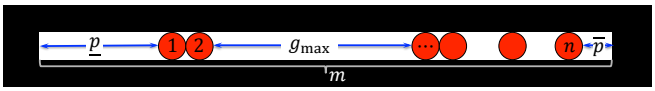

Fig. 2. Exploring a 1D environment of size m with n particles. Here $m = 20$ and $n = 6$. $\underline{p} = 4$, $\overline{p} = 1$ and $g_{max} = 7$

2) 1D mapping with n particles: Let $\mathbf{p}$ be the list of positions of n particles uniformly distributed from $[1, m]$. As shown in Fig. 2, the number of moves to discover the left and right boundaries is bounded by the maximum and minimum particles $\underline{p} = \min(\mathbf{p})$, $\overline{p} = \max(\mathbf{p})$, requiring moving left $\underline{p}$, followed by a move of $m - (\overline{p} - \underline{p})$ right. When $n = m$, the algorithm requires 2 moves, one left, one right. The minimum time with $n \in [2, m-1]$ occurs with $\underline{p}$ at 1 and $\overline{p}$ at m, requiring 3 moves, 1 left followed by 2 moves to the right.

The maximum $2(m-n+1)$ occurs when the particles are arranged from $m-n$ to m, requiring $m-n+1$ moves to the left, followed by $m-n+1$ moves to the right.

This is drawing without replacement n times from the set $[1, m]$. The minimum is distributed between 1 and $m-n$, the maximum is distributed between n and m.

The expected number of moves to reach both boundaries for n particles in 1D is

$$\frac{1}{\binom{m}{n}} \sum_{\underline{p}=1}^{m-n+1} \sum_{\overline{p}=\underline{p}+n-1}^{m} \binom{\overline{p}-\underline{p}-1}{n-2} (2\underline{p} + m - \overline{p} + 1) = \frac{3(1+m)}{1+n} \quad (2)$$

To fully map the area from 1 to m requires that every position from 1 to m be visited by at least one particle. This time is dominated by the maximum gap $\overline{g}$. The total number of moves is then $2\underline{p}+m-\overline{p}+1+\max\left(\overline{g} - \left(\underline{p} + m - \overline{p}\right), 0\right)$.

If all the particles are unit size, there are $m-n$ spaces, and these can be located before, between, or after the n particles in $n+1$ locations giving $\binom{m-n}{n+1}$ possible configurations. The largest gap can be calculated exactly using a recurrence equation [35], but a tight bound when $m > n \log n$ is $\frac{m-n}{n+1} + \Theta\left(\sqrt{\frac{(m-n)\log(n+1)}{n+1}}\right)$ [36].

3) 1D mapping with scan and move costs: Often scanning (imaging) and moving the particles costs time and energy. When controlling particles with MRI as in [37], the MRI machine iterates between imaging and applying gradient forces to move the particles. This section examines 1D mapping when scanning the workspace and moving the particles a unit distance have associated costs. The objective is to minimize a linear combination of costs for moving and measuring; however, the precise respective coefficients may be subject to change, or even unknown in advance turning this into a *bicriteria problem*, in which both parameters need to be within a bounded ratio of those in an optimal solution. For simpler notation, we write (a, b) for a schedule that involves a unit steps and b scans.

For a more detailed analysis, assume that the left boundary is located D units to the left of the leftmost particle. (This analysis can be applied in both directions.) The theoretically optimal, yet elusive, solution requires scanning the workspace to map particle locations, moving $D+1$ units to the left, then scanning to detect that the leftmost particle has only moved D units and thus has encountered the wall, for a total cost of $(D+1, 2)$ for the schedule.

We can achieve a schedule with $D+1$ steps by scanning after each step, for a total cost of $(D+1, D+2)$; while this is optimal with respect to steps, the involved scan cost is large compared to the optimum. At the expense of increasing the number of steps we can reduce the number of scans by successively doubling the number of steps between scans, i.e., performing the i^{th} scan after 2^i steps, resulting in total cost at most $(2D+2, \log_2 D)$; replacing the base of 2 by an arbitrary constant k, we get $(k \times (D+1), \log_k D)$. This is within a constant of the optimal. On the other hand, moving a sufficiently large number M of steps (known to satisfy $M \geq D$) before performing the second scan yields $(M, 2)$, which is optimal with respect to scan cost, but bad in terms of the cost for motion.

Balacing the competitive factor for both parameters can be achieved as follows: perform the ith scan at position i^i. As it turns out, this yields a simultaneous competitive factor of $O(\log D / \log\log D)$ for *both* parameters.

Theorem 1: The hyperexponential search sequence i^i yields an simultaneous competitive factor of $O(\log D / \log\log D)$ for *both* parameters of the bicriteria search problem; this is the best possible.

Proof: Let us first consider the number of scans. If the boundary is properly detected in step $j+1$, the particle must have encountered it between steps j and $j+1$, i.e. $j^j \leq D < (j+1)^{j+1}$. Now we can employ the Lambert W function, which is the inverse function of $f(x) = xe^x$; note that

$$\log x = W(\log x) \cdot e^{W(\log x)},$$

so

$$\log\log x = (\log W(\log x) + W(\log x)),$$

and therefore

$$W(\log x) \in \Theta(\log\log x).$$

This implies that $j \leq \log D / W(\log D)$ (the inverse of j^j), hence $\Theta(\log D / \log\log D) + 1$ scans have been made, while the optimum are 2 scans.

The moved distance is $(j+1)^{j+1}$, while the optimum is $D \geq j^j$. Hence, we get the ratio

$$\frac{(j+1)^{(j+1)}}{j^j} = (j+1)\frac{(j+1)^j}{j^j}, \quad (3)$$

where $0 \leq \frac{(j+1)^j}{j^j} \leq \mathrm{e}$ for $j > 0$.

Because $j \leq \log D / W(\log D)$, we obtain

$$\frac{(j+1)^{(j+1)}}{j^j} \leq \mathrm{e}\frac{\log n}{W(\log n)} + \mathrm{e} \quad (4)$$

for $j > 0$. Clearly, this is again within a factor of $\Theta(\log D / \log\log D) + 1$ of the optimum. ∎

B. Mapping in 2D

1) 2D mapping with 1 particle: The shortest path for mapping with 1 particle is a version of depth-first search that halts when all frontier cells have been explored. As long as the freespace is connected, depth-first-search (DFS) is the optimal solution to mapping. Even if the environment is known in advance, the problem is NP-hard as can be shown by a trivial reduction to Hamiltonian paths in grid graphs [38]. One can easily show that a simple DFS guarantees an optimal ratio of 2: the depth-first tree has $m-1$ edges and each edge is traversed at most twice. Any path that covers m fields needs to traverse least $m-1$ edges, and hence the depth-first-search is at most twice as long as an optimal coverage path.

For showing that no algorithm can perform better one needs only a simple 1-dimensional graph that goes to the left and to the right. If the algorithm chooses to go arbitrarily to one side, we can make it do a long walk of length m and then return it just for a single field on the other side ($2m+1$ vs. $2+m$). If the algorithm decides to switch the direction after some time after arbitrary zig-zags (of increasing cost) of cost z (center to one side to other side) we decide that there is a single field on both sides. The algorithm now needs to go one additional time from one side to the other and back (cost $> z$) while the optimum cost would have been $\leq z+3$. If the algorithm switches from the second form to the first, the first argument still applies.

Most previous work on grid graph exploration focused on exploration tours, i.e., after exploration one has to go back to the start position. If the environment is known in advance, this equals the traveling salesman problem and a polynomial-time approximation scheme is known [39]. If the environment is unknown, as it is in our case, the best achievable competitive ratio is 2 in general grid graphs (achieved by depth first search) and $7/6$ for simple grid graphs ($4/3$ achieved by smartDFS [40]).

2) 2D mapping with n particles: The problem with mapping with n particles is also to identify which move sequence guarantees the shortest path in the worst case. At the beginning there can be at most four frontier cells per particle to be explored. As the particles begin to move, the number of frontier cells explored and the number of free cells identified increases.

We describe three algorithms for 2D mapping. If we implement a random move algorithm as described in Alg. 1, at each step the particles all move in the same randomly selected direction until there are no frontier cells left on the map. $MoveType$ is a vector that holds the four possible move types. The map $\mathbf{M}$ is a matrix the size of the work space. Each cell of $\mathbf{M}$ holds one of five values that denote: $Particle$, $Frontier$, $Unknown$, $Freespace$ and $Obstacle$. At each step FRONTIER returns the locations of frontier cells in $\mathbf{M}$ and $\mathbf{r}$ has the list of particle locations. The $move$ is implemented to update the map $\mathbf{M}$ and the particle locations $\mathbf{r}$ by calling MOVE&UPDATE. Alg. 1 requires minimal computation and is probabilistically complete, so eventually the swarm maps the free space [41]. However, this method of mapping is inefficient, resulting in long mapping times, especially with small numbers of particles in large, torturous freespaces.

Algorithm 1 RANDOMMOVES($\mathbf{M}, \mathbf{r}$)

```
MoveType = {l, u, r, d}
while |FRONTIER(M)| > 0 do
    move ← RANDOM(MoveType)
    {M, r} ← MOVE&UPDATE(move, M, r)
end while
```

A better way to map the world is to deliberately move particles toward frontier cells. We could choose one particle as the $elect$ particle and perform motion planning using this particle. In Alg. 2, one of the particles is selected as $elect$. As long as the number of frontiers to be visited is at least one, the algorithm proceeds by generating a $moveSeq$ from the current position of the elect particle to the nearest frontier cell. The $moveSeq$ is generated by a breadth-first-search (BFS) shortest path algorithm which requires the map $\mathbf{M}$, source $elect$, and the cells FRONTIER($\mathbf{M}$). A representative $moveSeq$ is $\langle u, r, d, d, r, u, \ldots \rangle$. The list of moves in $moveSeq$ are implemented by iterativly calling the MOVE&UPDATE function for the length of the list.

Alg. 2 will explore the target frontier cell by the end of $moveSeq$. However, often with large-population swarms the whole $moveSeq$ need not be implemented. Every time MOVE&UPDATE is called, the nearest frontier is updated and $moveSeq$ is also updated because as the particles start to move, the target frontier cell might be explored by a non-elect particle.

Alg. 3 exploits this fact by computing a BFS shortest path from all particles to all frontier cells.

Algorithm 2 ELECTPARTICLE($\mathbf{M}, \mathbf{r}$)

```
elect ← RANDOM(r)
while |FRONTIER(M)| > 0 do
    moveSeq ← BFS(M, elect, FRONTIER(M))
    for iter := 1 to |moveSeq| step 1 do
        {M, elect} ← MOVE&UPDATE(moveSeq, M, elect)
    end for
end while
```

In each loop of Alg. 3, all the moves in $moveSeq$ are implemented to explore the target frontier cell since it is the shortest possible route to a frontier cell. At time 0, there will be at most $4n$ equally valid destinations that can be visited since cells to the side of each particle not neighboring another particle are frontier cells. BFS expands the search in a clockwise order, l, u, r, d, so the first move is l. Each $moveSeq$ guarantees classification of one frontier cell into obstacle or free space. When a frontier cell is explored it is labelled either free or obstacle. There can be a net gain of at most two frontier cells per particle that encounters a free cell or no new frontier cells if the frontier cell contained an obstacle.

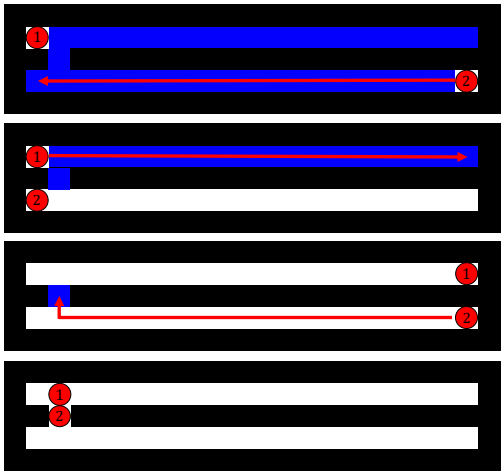

Fig. 3. In this example, the greedy strategy using Alg. 3 is 1.5× worse than DFS using only the bottom left particle.

The simulation results in Section IV show that both map complexity and distribution affect the number of moves taken to map the work space. Alg. 1 uses no information from the data except for checking completion. Alg. 2 uses the location and distance data from one particle to map the work space. Alg. 3 improves the performance of mapping by using all the data.

Algorithm 3 CLOSESTFRONTIER($\mathbf{M}, \mathbf{r}$)

```
while |FRONTIER(M)| > 0 do
    moveSeq ← BFS(M, r, FRONTIER(M))
    for iter := 1 to |moveSeq| step 1 do
        {M, r} ← MOVE&UPDATE(moveSeq, M, r)
    end for
end while
```

While the greedy strategy is the more reasonable approach in practice than DFS with a single particle, we are only able to show a trivial weaker bound on the corresponding moves. However, in some scenarios the greedy strategy can perform worse than DFS with a single particle, e.g., in Fig. 3.

Theorem 2: The greedy strategy needs at most $0.5 \cdot m \cdot (m+1)$ moves where m is the number of fields.

Proof: The distance between a particle and the closest boundary can be at most the number of all already visited fields. Hence, the distance for visiting the i^{th} field is bounded by i. The overall number of moves is bounded by $\sum_{i=1,\ldots,m} i = 0.5 \cdot m \cdot (m+1)$. ■

The greedy strategy can need $\Omega(m^2)$ moves while an optimal strategy only needs $O(m)$. An example can be seen in Fig. 4.

Theorem 3: The greedy strategy has a computational complexity in $O(m^2)$.

Proof: For an environment with m fields, there are at most m iterations. Since the edges in the grid graph are not weighted and each field only has at most four neighbors, the shortest path from a particle to the boundary can be calculated in $O(m)$ time by a simple breadth first search. ■

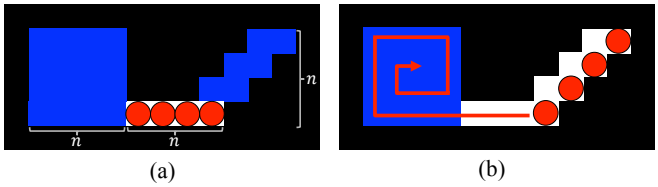

Fig. 4. If the move preference is counter clockwise, the greedy strategy Alg. 3 will go right first and then cover the square with a single particle which takes $\Omega(n^2)$ moves while the optimal strategy, which visit the square in parallel using all n particles, only needs $O(n)$ moves.

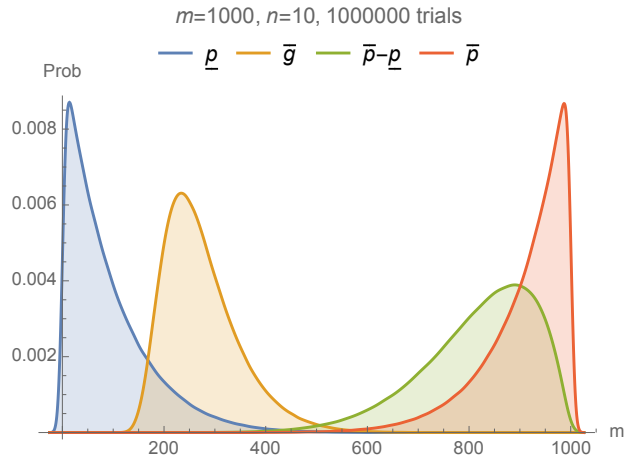

Fig. 5. With a connected, 1D freespace m = 1000 and $n = 10$ particles, the distributions for the gap before the first $\overline{p}$ and after the last $\underline{p}$ gaps are symmetric. The maximum gap $\overline{g} \approx 250$.

The minimum mapping time is a rectangle n units tall and m/n wide, with the n particles arranged along the left wall. 1 move left registers the left wall, repeating m/n moves up and right to find the top boundary and the right boundary, followed by $m/n-1$ moves down and left to find the bottom boundary, with one final move down to register the furthest left bottom boundary. This is $4(m/n)$ moves.

Finally, completion time is also a function of the map geometry. Mapping requires exploring all the free spaces and the boundary of the free spaces. The number of map cells that need to be explored is the $Area + Perimeter - n$. This is minimized by a circular region and maximized by a linear region. For example, a linear region has $m + 2m + 2 - n$ cells to explore, while a square region has $m + 4\sqrt{m} - n$ cells to explore.

IV. SIMULATION

A. 1D mapping

1D simulations were conducted in Mathematica, with code available at [42]. Fig. 5 shows the distributions for the minimum and maximum initial particle locations $\underline{p}$ and $\overline{p}$, the maximum gap $\overline{g}$, and the spread between the minimum

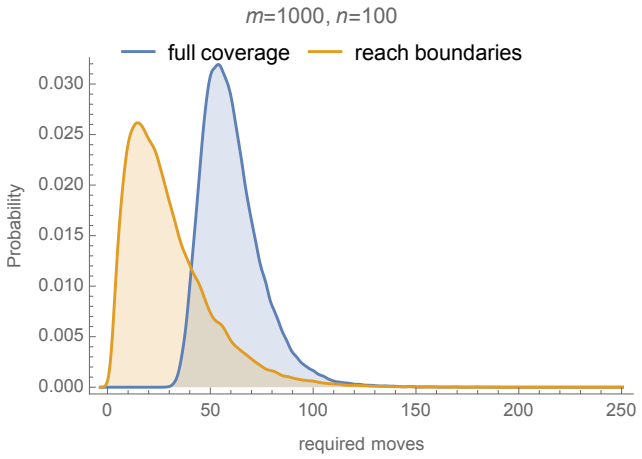

Fig. 6. Full coverage in 1D requires 60.7 moves on average, while reaching the boundaries requires only 29.8.

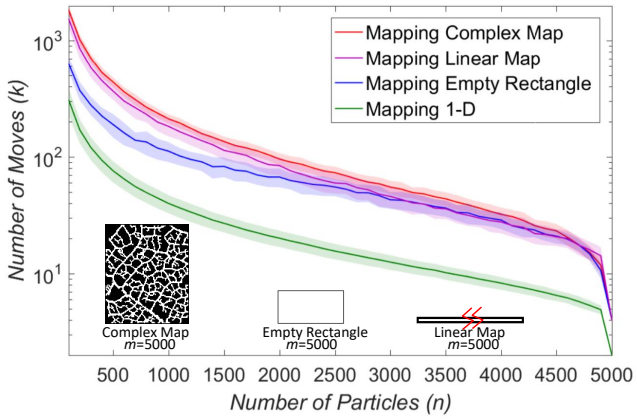

Fig. 7. Comparison of mapping 2D maps of three types using the Closest Frontier algorithm and the 1-D mapping of the linear map. The complex map, empty rectangle map, and linear map (similar to a 1D map, but with obstacles to detect on top and bottom) have 5000 free spaces.

and maximum $\overline{p} - \underline{p}$ for 1,000,000 Monte Carlo trials. The expected gap between the first particle and the boundary $\underline{p}$ is 90.94. The expected gap between the last particle and the boundary $\overline{p}$ is 90.98. The expected maximum gap is $\overline{g}$ is 273.9.

Fig. 6 shows that full coverage requires approximately twice the time required to explore the left and right boundaries when $m = 1000$ and $n = 100$.

B. 2D mapping

The mapping simulation used a map with 5000 free spaces. Each simulation trial was repeated 100 times. The number of particles ranged from 100 to 5000 by increments of 100. In each run the particles were placed randomly throughout the workspace. A comparison between the simulation results of the three algorithms is shown in Fig. 9. The mean completion time and standard deviation of the completion time decreased with increasing numbers of particles. The log plot shows that all three algorithms have an almost logarithmic relationship between m, the number of free spaces, and k, the number of particles. The maximum number of moves required using the closest frontier algorithm was for k=100 with an average moves ≈1816 and standard deviation of 160 moves. This reduces to four moves with 0 standard deviation when n= 5000 (the total number of free spaces).

The comparison plot Fig. 7 between the mapping of three 2D maps *complex*, *empty rectangle* and *linear* and a 1D map, shows that the complex map requires the most moves.

In the Fig. 7 there is an observable difference in moves between the linear and rectangular workspaces. One reason is because the number of useful moves is reduced in the linear case, where only left and right are useful moves. Up and down eliminate boundaries, but never discover free cells. Whereas in the *empty rectangle* case most of the moves are moves which add to the number of mapped spaces. Only when the number of particles is around 2/3 of the number of free spaces is there an overlap between the moves taken to map the rectangular space and the linear space.

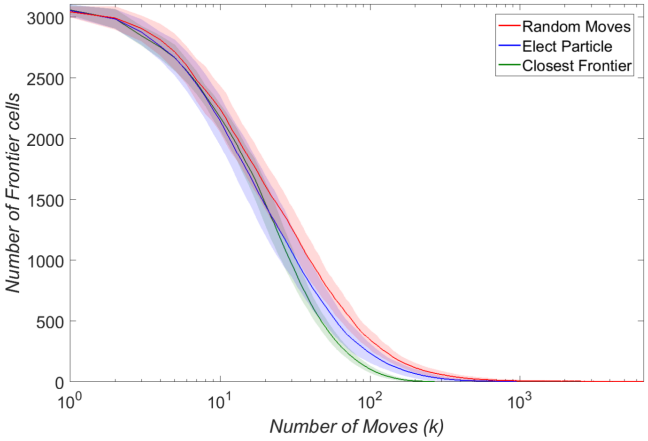

Fig. 8. Performing mapping on the complex 2D map with $n = 1000$ particles. Random moves requires an average of 4364 moves. Elect particle requires 933 moves and Closest Frontier requires 438 moves

The difference between algorithms is highlighted in Fig. 8, which shows the number of unexplored cells as a function of the number of moves commanded. All tests used $n = 1000$ particles, but elect requires on average twice as many moves as closest frontier and random requires ten times as many moves as closest frontier.

Fig. 9 compares the performance of Algs. 1, 2, and 3 on the complex 2D map. Random moves performs worst, with the largest number of required moves and the largest standard deviation of required moves. Random moves is slightly better than elect particle for large numbers of particles, but both algorithms are beat by closest frontier, which has the minimum number of required moves and the smallest standard deviation.

Fig. 10 compares mapping, coverage, and foraging on the complex 2D map. All are using Alg. 3. Coverage is performed with a known map, but with all free cells initialized to be frontier cells. Similarly, foraging has a known map, but 10% of the empty cells are labelled as frontier cells.

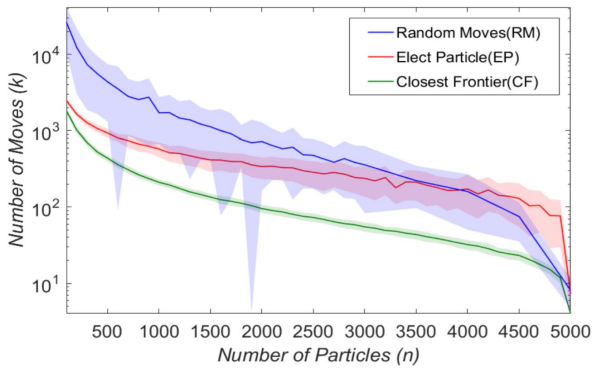

Fig. 9. Comparison of three Algorithms - Random Moves (blue), Elect Particle (red) and Closest Frontier (green) for mapping the 2D Complex Map of 5000 free spaces.

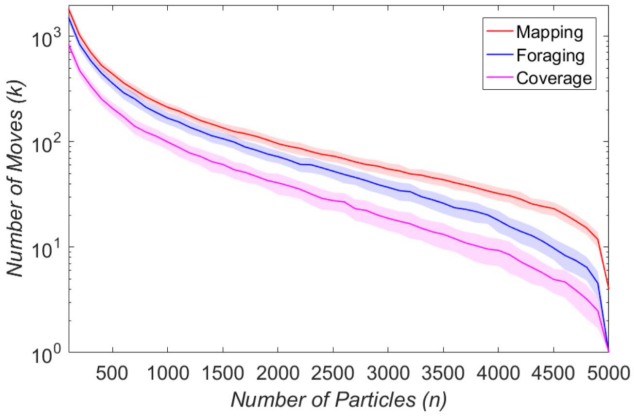

Fig. 10. Comparison of three related problems: mapping, coverage, and foraging on the complex 2D map.

Foraging is a constant factor easier than coverage, which is a constant factor easier than mapping.

The final simulation test, shown in Fig. 11, compares the effect of different initial particle distributions in the complex 2D map. Region fill places all n particles at a minimum Manhattan distance from a randomly selected location on the map. Flood fill places one particle at a randomly selected location in the free space, and places the remaining particles according to a breadth-first expansion inside the free space. Uniform distribution places the particles uniformly randomly. Region fill and flood fill have similar performance, while uniform distribution requires many fewer moves. This is because dispersing particles using only global inputs is difficult, and a uniform distribution starts with the particles dispersed.

V. Conclusion and Future Work

This paper presented techniques for controlling particle swarms in 1D and 2D grids. These particles can be tracked and controlled by an external agent, but control inputs are applied uniformly so that each particle experiences the same applied forces.

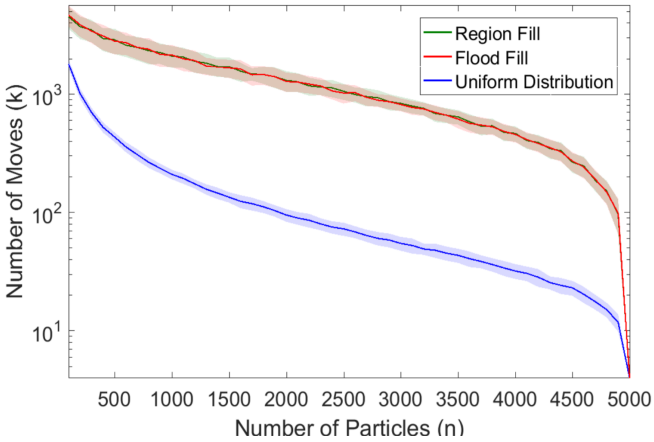

Fig. 11. Comparison with different distributions: flood-fill, region fill, and uniform distribution for mapping on the complex 2D map.

We provided a spectrum of theoretical and practical new insights that have particular relevance for fast MRI scans with magnetically controlled contrast media. In particular, we developed a fundamentally new approach for searching for an object at an unknown distance D, where the search is subject to two different and independent cost parameters for *moving* and for *measuring*. We showed that regardless of the relative cost of these two operations, there is a simple $O(\log D/\log\log D)$-competitive strategy. This paper also presented benchmark algorithms for 2D mapping, foraging, and coverage problems. These results form a baseline for future work, which should focus on improving performance. Extensions to 3D are especially relevant to the motivating problem of MR-scanning in living tissue.

References

[1] H. B. Na, I. C. Song, and T. Hyeon, "Inorganic nanoparticles for mri contrast agents," *Advanced materials*, vol. 21, no. 21, pp. 2133–2148, 2009.

[2] P. Caravan, J. J. Ellison, T. J. McMurry, and R. B. Lauffer, "Gadolinium (iii) chelates as mri contrast agents: structure, dynamics, and applications," *Chemical reviews*, vol. 99, no. 9, pp. 2293–2352, 1999.

[3] P. Vartholomeos, M. Akhavan-Sharif, and P. E. Dupont, "Motion planning for multiple millimeter-scale magnetic capsules in a fluid environment," in *IEEE Int. Conf. Rob. Aut.*, May 2012, pp. 1927–1932.

[4] S. Chowdhury, W. Jing, and D. J. Cappelleri, "Controlling multiple microrobots: recent progress and future challenges," *Journal of Micro-Bio Robotics*, vol. 10, no. 1-4, pp. 1–11, 2015. [Online]. Available: http://dx.doi.org/10.1007/s12213-015-0083-6

[5] S. Martel, S. Taherkhani, M. Tabrizian, M. Mohammadi, D. de Lanauze, and O. Felfoul, "Computer 3D controlled bacterial transports and aggregations of microbial adhered nano-components," *Journal of Micro-Bio Robotics*, vol. 9, no. 1-2, pp. 23–28, 2014.

[6] P. S. S. Kim, A. Becker, Y. Ou, A. A. Julius, and M. J. Kim, "Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control," *Journal of Nanoparticle Research*, vol. 17, no. 3, pp. 1–15, 2015.

[7] B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus, "Planning and control for microassembly of structures composed of stress-engineered MEMS microrobots," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 218–246, 2013. [Online]. Available: http://ijr.sagepub.com/content/32/2/218.abstract

[8] A. Ghosh and P. Fischer, "Controlled propulsion of artificial magnetic nanostructured propellers," *Nano Letters*, vol. 9, no. 6, pp.

2243–2245, 2011/10/23 2009. [Online]. Available: http://dx.doi.org/10.1021/nl900186w

[9] Y. Ou, D. H. Kim, P. Kim, M. J. Kim, and A. A. Julius, "Motion control of magnetized tetrahymena pyriformis cells by magnetic field with model predictive control," *Int. J. Rob. Res.*, vol. 32, no. 1, pp. 129–139, Jan. 2013.

[10] F. Qiu and B. J. Nelson, "Magnetic helical micro-and nanorobots: Toward their biomedical applications," *Engineering*, vol. 1, no. 1, pp. 21–26, 2015.

[11] A. V. Mahadev, D. Krupke, J.-M. Reinhardt, S. P. Fekete, and A. T. Becker, "Collecting a swarm in a grid environment using shared, global inputs," in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1231–1236.

[12] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[13] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3852–3857.

[14] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999.

[15] D. Latimer, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst, "Towards sensor based coverage with robot teams," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 961–967.

[16] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 47–53.

[17] R. Bellman, "An optimal search problem," *SIAM Reviews*, vol. 5, p. 274, 1963.

[18] A. Beck, "On the linear search problem," *Naval Research Logistics*, vol. bf 2, pp. 221–228, 1964.

[19] A. Fiat and G. J. Woeginger, *Online optimization, the state of the art*, ser. Lecture Notes in Computer Science. Springer, 1998, vol. 1442.

[20] A. Beck and D. J. Newman, "Yet more on the linear search problem," *Israel Journal of Mathematics*, vol. 8, pp. 419–429, 1970.

[21] S. Gal, "A general search game," *Israel Journal of Mathematics*, vol. bf 12, pp. 32–45, 1972.

[22] ——, "Minimax solutions for linear search problems," *SIAM Journal on Applied Mathematics*, vol. bf 27, pp. 17–30, 1974.

[23] E. D. Demaine, S. P. Fekete, and S. Gal, "Online searching with turn cost," *Theoretical Computer Science*, vol. 361, no. 2–3, pp. 342–355, 2006.

[24] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia, "Optimal covering tours with turn costs," *SIAM Journal of Computing*, vol. 35, pp. 531–566, 2005.

[25] M. Y. Kao, J. H. Reif, and S. R. Tate, "Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem," *Information and Computation*, vol. 131, pp. 63–79, 1996.

[26] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins, "Searching in the plane," *Information and Computation*, vol. 106, pp. 234–252, 1993.

[27] S. Gal, *Search Games*. Academic Press, 1980.

[28] S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishers, 2003.

[29] S. P. Feketea, R. Kleina, and A. Nüchter, "Searching with an autonomous robot. (video + abstract)," in *Proceedings of the 20th ACM Symposium on Computational Geometry*, 2004, pp. 449–450.

[30] S. P. Fekete, R. Klein, and A. Nüchter, "Online searching with an autonomous robot," *Computational Geometry: Theory & Applications*, vol. 34, pp. 102–115, 2006.

[31] S. P. Fekete, J. S. B. Mitchell, and C. Schmidt, "Minimum covering with travel cost," *Journal of Combinatorial Optimization*, vol. 20, pp. 32–51, 2010.

[32] S. K. Lee, S. P. Fekete, and J. McLurkin, "Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers," *International Journal of Robotics Research*, vol. 9, no. 35, pp. 1234–1260, 2016.

[33] S. Martel, "Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects," *Micro-and Nanomanipulation Tools*, 2015.

[34] X. Yan, Q. Zhou, J. Yu, T. Xu, Y. Deng, T. Tang, Q. Feng, L. Bian, Y. Zhang, A. Ferreira, and L. Zhang, "Magnetite nanostructured porous hollow helical microswimmers for targeted delivery," *Advanced Functional Materials*, vol. 25, no. 33, pp. 5333–5342, 2015.

[35] P. Reviriego, L. Holst, and J. A. Maestro, "On the expected longest length probe sequence for hashing with separate chaining," *Journal of Discrete Algorithms*, vol. 9, no. 3, pp. 307–312, 2011.

[36] M. Raab and A. Steger, *"Balls into Bins" — A Simple and Tight Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 159–170. [Online]. Available: http://dx.doi.org/10.1007/3-540-49543-6_13

[37] A. Chanu, O. Felfoul, G. Beaudoin, and S. Martel, "Adapting the clinical MRI software environment for real-time navigation of an endovascular untethered ferromagnetic bead for future endovascular interventions," *Magnetic Resonance in medicine*, vol. 59, no. 6, pp. 1287–1297, Jun. 2008.

[38] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM Journal on Computing*, vol. 11, no. 4, pp. 676–686, 1982.

[39] P. N. Klein, "A linear-time approximation scheme for tsp in undirected planar graphs with edge-weights," *SIAM Journal on Computing*, vol. 37, no. 6, pp. 1926–1952, 2008.

[40] C. Icking, T. Kamphans, R. Klein, and E. Langetepe, "Exploring simple grid polygons," in *International Computing and Combinatorics Conference*. Springer, 2005, pp. 524–533.

[41] J. D. Kahn, N. Linial, N. Nisan, and M. E. Saks, "On the cover time of random walks on graphs," *Journal of Theoretical Probability*, vol. 2, no. 1, pp. 121–128, 1989.

[42] A. Mahadev and A. T. Becker, ""Mapping with Particles under Uniform Inputs", https://github.com/roboticswarmcontrol/iros2017mappingparticles," Mar. 2017.