This is the author's version of a conference publication presented at the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Changes were made to this version by the publisher prior to publication.The final version of record is available at http://dx.doi.org/10.1109/COASE.2019.8842916 . IEEE 15th International Conference on Automation Science and Engineering (CASE), PREPRINT VERSION. ACCEPTED MAY, 2019.

Motion-planning Using RRTs for a Swarm of Robots Controlled by Global Inputs

Parth Joshi, Julien Leclerc, Daniel Bao, and Aaron T. Becker

***Abstract*— Small-scale robots have great potential in medicine, micro-assembly and many other areas. For example, robots containing iron can be steered using the magnetic gradient generated by MRI scanners. Since the gradient is approximately the same everywhere inside the scanner, each robot receives the same input and therefore they all are subjected to the same force. A similar technique can be used with rotating magnetic fields. Each robot receives the same inputs, making motion planning challenging. This paper uses a Rapidly Exploring Random Tree (RRT) to plan paths that deliver multiple robots to goal positions by using obstacles to break the actuation symmetry.**

I. Introduction

This paper investigates the individual control of multiple particle-like robots actuated by the same force and torque. Each robot moves in the same direction and the same distance (a shared global input) until its movement is obstructed. Our goal is to simultaneously deliver robots at different starting positions to different goal positions with these identical inputs. The problem is trivial if there is only one robot, but as the number of robot increases it becomes more difficult. The set of goal locations often requires that the robots move in different directions, so moving one robot toward its goal may move other robots away from their goals. We derived shortest path solutions for two robots under this control paradigm in [1], but only for obstacle-free, convex workspaces. We also derived a planner under this control paradigm for the special case of a workspace with a single, square-shaped obstacle in [2]. However, no general-purpose motion planner has addressed this problem. The motion planning problem at hand is hard because the global inputs make the problem coupled and so the path-planning complexity increases exponentially with the number of robots. This problem is especially challenging because the effects of collisions are often not *time reversible*: if the robots collide with obstacles at different times, moving in the opposite direction will not undo the effect of the collision. Because such collisions cannot be undone, two configurations close in the configuration space may be unreachable or require a long movement sequence to connect. This paper presents a modified version of a motion-planning technique called Rapidly Exploring Random Tree (RRT) to search for solutions.

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

Authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA {pmjoshi, atbecker}@uh.edu

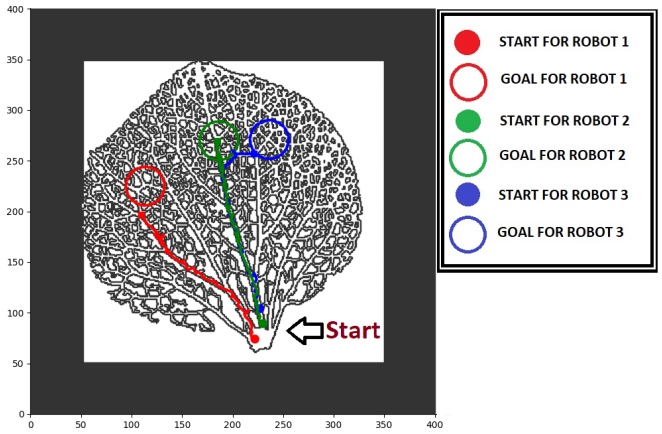

Fig. 1. Solving a complex vascular network with three robots moving under global inputs. All robots move in the same direction unless their movement is blocked by an obstacle.

Global inputs are common for tiny robots. For instance, magnetic actuation can be used to actuate a swarm of microscopic magnetic robots [3]–[6]. They could navigate inside the human body to deliver medication or create imaging contrast [7]–[10]. Because the magnetic field generators must be placed outside the patient's body, it is reasonable to assume all the robots would be subjected to same magnetic field and would move in the same direction [11].

Figure 1 presents a representative workspace containing obstacles (shown in black color) and three robots (shown as colored discs). The robots move under global inputs and their goal positions are shown by similarly colored circles. The relative initial positions of the robot are different from the relative position of the goals. In this case only one robot can reach its goal at a time unless obstacles are used to break the control symmetry. A control sequence found by our planner is shown by the correspondingly colored paths.

The contributions of this paper are (1) the introduction of a local planner for non-time reversible global inputs, (2) extensive testing of this planner in an RRT in simulation, and (3) hardware implementation of the paths generated by the RRT planner. In this paper, simulations were performed with up to four robots and the hardware experiments were performed using two robots. This paper is an expansion of work presented in the MS thesis [12].

Section II presents our model, Sec. III explores how parameters affect map coverage, and Sec. IV examines increasing numbers of robots, more complicated environments, and reuse of the RRT for multiple planning queries.

The developed algorithm was tested experimentally as reported in Sec. V. Two cylindrical permanent magnets (the

robots) were rolled on a flat surface by a rotating magnetic field. Obstacles were present in the workspace to break the actuation symmetry. Six electromagnets produced a global magnetic field with controlled time-dependent orientation. The magnetic field vectors for the tested path were computed offline and then generated by the magnetic manipulator to steer the robots to their separate goal locations.

II. Model & Methods

The goal of robot motion-planning is often to plan a collision-free path from start to goal for a robot. However, because we have more robots than control inputs, our algorithm requires collisions with obstacles to break the symmetric effect of control inputs.

This paper models robots actuated by the same field e.g., iron particles pulled by a magnetic field [13]. Consider a group of n robots with configuration $\mathbf{r} = [r_{1x}, r_{1y}, \ldots, r_{nx}, r_{ny}]^\top$, $\mathbf{r} \in \mathbb{R}^{2n}$, all controlled by the same input sequence $\mathbf{u}$ to move from a start configuration $\mathbf{s} = [s_{1x}, s_{1y}, \ldots, s_{nx}, s_{ny}]^\top$ towards their goal configuration $\mathbf{g} = [g_{1x}, g_{1y}, \ldots, g_{nx}, g_{ny}]^\top$. Each move command in the sequence $\mathbf{u}$ is a force vector in 2D. Robots slide along, but do not pass through obstacles. When any robot collides with an obstacle, components of $\mathbf{u}$ normal to the obstacle are set to zero.

A Rapidly-exploring Random Tree (RRT) [14] is a tree-based algorithm designed to explore high-dimensional configuration spaces. The tree is rooted at the initial configuration. To generate another node the following procedure is employed: First, X_{rand}, a random configuration in the free configuration space is generated [15]. Second, the node already in the tree that is nearest to the randomly generated configuration, n_{NN}, is selected [16], [17]. Third, the algorithm tries to *expand* node n_{NN} by generating a movement command from n_{NN} towards X_{rand}. If the movement results in a valid configuration, a new node is created and connected to n_{NN}. The RRT continues generating nodes until it finds the desired goal or reaches its limit of time or memory.

a) Distance Function: A distance metric is necessary for an RRT to determine the nearest node. This paper uses the Euclidean distance between two configurations as a distance metric. For two configurations $\mathbf{r}$ and $\mathbf{g}$, the distance is

$$\textsc{Distance}(\mathbf{r}, \mathbf{g}) = \sqrt{\sum_{i=1}^{n} \Big((r_{xi} - g_{xi})^2 + (r_{yi} - g_{yi})^2 \Big)}.$$

b) RRT algorithm modified for global inputs: For given start configuration $\mathbf{s}$ and goal configuration $\mathbf{g}$, an RRT Tree with K vertices is constructed as shown in Alg. 1. The key innovation in this paper is the LocalPlanner in step 5.

c) A LocalPlanner Function: Collisions are often not time-irreversible because the system has uniform inputs. This makes it difficult to determine the optimal movement command that would bring one configuration closest to another configuration. Instead of computing the optimal input, we generate k potential inputs and select the input that minimizes the distance to the goal when collisions are accounted for. In our implementation, Alg. 2, each candidate input is of length *Robostep*, and the k potential directions are evenly spaced along the unit circle.

Algorithm 1 GenerateRRT$(\mathbf{s}, K)$

$\mathbf{s}$ is the starting particle configuration and K is the maximum number of nodes. SampleRandomFreeConfig samples a configuration for which every robot is in the free configuration space.

1: *Tree.initialize*$(\mathbf{s})$
2: **for all** $i = 1$ to K **do**
3: $\quad \mathbf{x}_{\text{rand}} \leftarrow$ SampleRandomFreeConfig
4: $\quad \mathbf{x}_{\text{near}} \leftarrow$NearestNeighbor$(\textit{Tree}, \mathbf{x}_{\text{rand}})$
5: $\quad (\mathbf{x}_{\text{new}}, u) \leftarrow$ LocalPlanner$(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rand}})$
6: $\quad$ *Tree.add_vertex*$(\mathbf{x}_{\text{new}})$
7: $\quad$ *Tree.add_edge*$(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}}, u)$
8: **return** *Tree*

Algorithm 2 LocalPlanner$(\mathbf{s}, \mathbf{e})$

Computes which move of length *Robostep* in k directions from configuration $\mathbf{s}$ gets closest to configuration $\mathbf{e}$.

1: $\textit{min}_{\textit{dist}} \leftarrow \infty$
2: **for all** $a = 0$ to $k - 1$ **do**
3: $\quad \phi \leftarrow \frac{2\pi a}{k}$
4: $\quad u \leftarrow \textit{Robostep} \cdot [\cos\phi, \sin\phi]$
5: $\quad \mathbf{t} \leftarrow \{\}$
6: $\quad$ **for** $i = 1$ to n **do**
7: $\quad\quad t_i \leftarrow$ ApplyCollision(s_i, u)
8: $\quad$ **if** Distance$(\mathbf{t}, \mathbf{e}) < \textit{min}_{\textit{dist}}$ **then**
9: $\quad\quad \textit{min}_{\textit{dist}} \leftarrow$ Distance$(\mathbf{t}, \mathbf{e})$
10: $\quad\quad \mathbf{t}_{\text{best}} \leftarrow \mathbf{t}$
11: $\quad\quad u_{\text{best}} \leftarrow u$
12: **return** $(\mathbf{t}_{\text{best}}, u_{\text{best}})$

d) Calculation of Configuration Space Coverage: Map coverage is a proxy for exploration of the configuration space [18], and is computed by dividing the configuration space into equal volume hypercubes, and computing what fraction of the hypercubes contain at least one RRT node. The size of the grid (cube edge length) can be decreased to improve precision. As shown in Fig. 2, the frame and the bounds remain the same in all the environments we used in the simulation for this paper. With n robots, the configuration space has $2n$ dimensions, so as the number of robots increases, the memory requirement for the grid increases exponentially as $O\left(\left(\frac{\text{boundary length}}{\text{edge length}}\right)^{2n}\right)$. For high resolution maps, calculating coverage requires more computation time than generating the RRT.

III. Simulation Results

This section presents several simulation results, all programmed using the C++ language and the OMPL library (Open Motion Planning Library) [19]. Configuration space coverage is calculated when changing parameters including

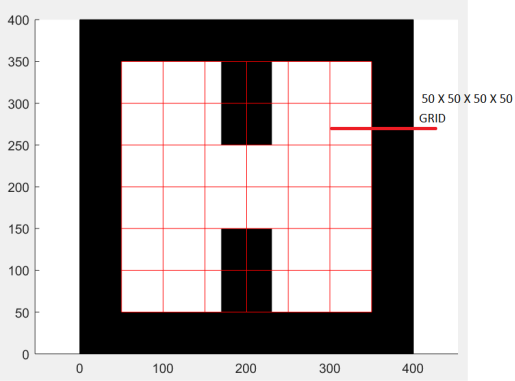

Fig. 2. Example of a grid on an environment

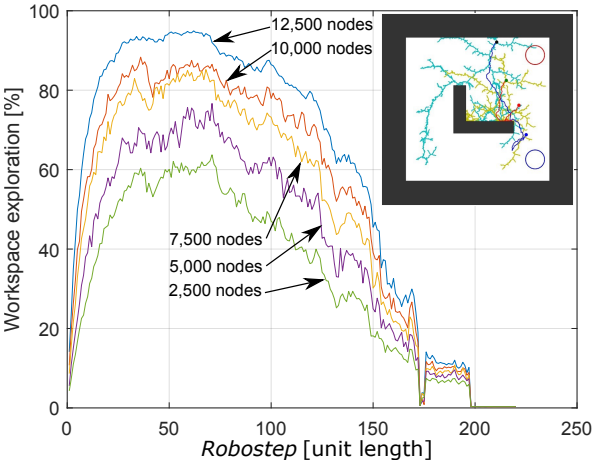

Fig. 3. Average coverage percentage as a function of Robostep length using the environment shown in the right-hand corner. Increasing numbers of RRT nodes improves coverage.

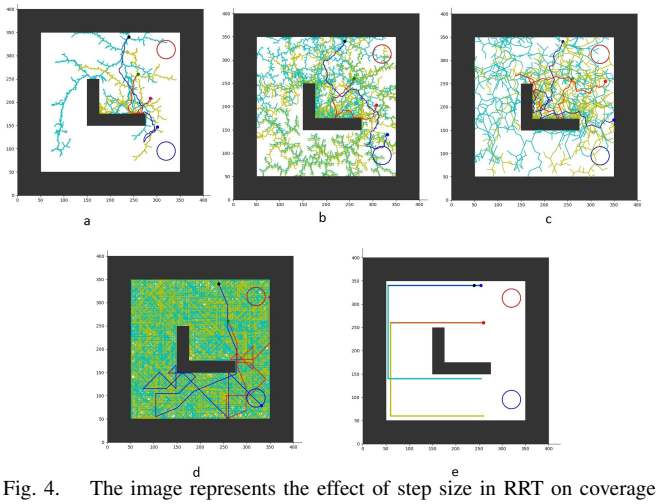

Fig. 4. The image represents the effect of step size in RRT on coverage of the search space. The RRT ran for 0.5 s with step sizes (a) 1, (b) 3, (c) 5, (d) 50, (e) 200 units.

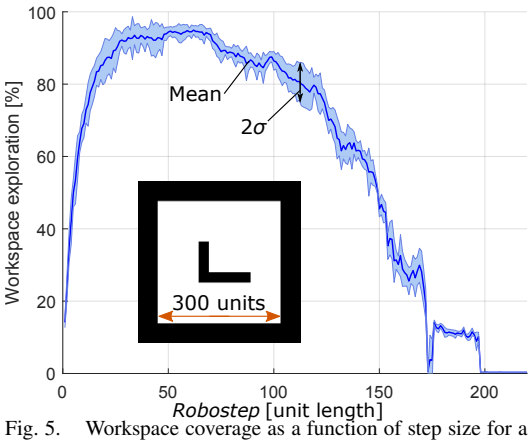

Fig. 5. Workspace coverage as a function of step size for a tree with 7500 nodes and two robots. Grid size is $50 \times 50 \times 50 \times 50$.

(1) number of nodes, (2) step size of the robot, (3) environment, (4) grid size, (5) distance between start and goal location and, (6) changing the initial location of robots. All simulations use $k = 16$ candidate control inputs in Alg. 2.

As shown in Fig. 4, environment 1 has a border of obstacles (shown in black) on the sides and an L-shaped obstacle (in black) in the center. The starting positions for the robots are shown as black discs for robot 1 and green for robot 2. The goals for robot 1 and robot 2 are shown as a blue circle and a red circle respectively. In addition, the RRT for robot 1 is shown in cyan and for robot 2 in yellow. The configuration space is 4D, but the picture projects the 4D space into 2D. The aim of each robot is to simultaneously reach its individual goal when moved by global inputs.

a) Changing the Number of Nodes: Increasing the number of RRT nodes increases coverage, as shown in Fig. 3.

b) Changing Step Size: In the RRT algorithm defined above, at every random sample, the robots are moved by a fixed step of length *Robostep*. The aim of this simulation is to observe how *Robostep* affects the coverage of the C-space. The RRT Tree for two robots projected onto 2D dimensions for five step lengths are shown in Fig. 4. The start and goal positions of the robots are the same for each experiment. The grid size is $50 \times 50 \times 50 \times 50$. The coverage of this grid as a function of *Robostep* is plotted in Fig. 5. Small step lengths results in a small tree, but too large of step lengths cause all inputs to terminate in collisions and results in a tree with poor coverage. The optimal step size was 60 units.

c) Changing Grid Size: In this experiment, the grid size was changed to observe the effect on the coverage percentage. The coverage is not actually changing, but each configuration space voxel is made smaller. The experiment is performed by changing the grid size to 40, using environment 1. As shown in Fig. 6, the coverage decreases as grid size decreases. The memory required for the grid increases exponentially with the number of the robots in the swarm. As the dimension increases, dividing the map into the grid becomes computationally complex and time-consuming.

d) Changing the Initial Distance Between Robots: The initial positions of the robots affects the performance of the RRT with the same number of nodes and *Robostep*. We selected eight initial robot positions (with decreasing spacing) and measured the effect on coverage, as shown in Fig 7. We performed 10 trials for each case. Decreasing the initial separation decreases the RRT coverage. As shown in Fig. 7 (b) the initial separation of the robots has little effect until a critical distance, after which the coverage drops quickly. These experiments indicate that the motion planner

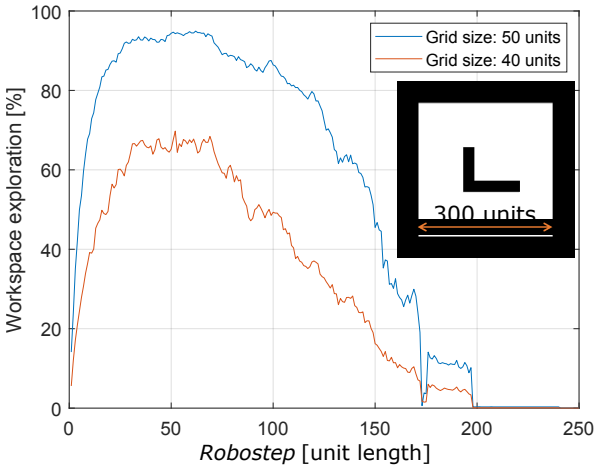

Fig. 6. Decreasing the grid size decreases the exploration percentage.

could not cover the map area when the robots are initialized too close together because it must separate the robots to achieve goal positions that are far apart.

IV. Increasing Complexity

This section reports on our RRT planner with more than two robots, and tests them inside complex environments as shown in Fig. 1 and 8. The time needed to solve an RRT increases exponentially with the number of robots. RRT algorithms generate a large number of nodes that fill the environment. The generated nodes are independent of the goal locations. It is therefore possible to reuse a computed RRT tree if only the goal locations are changed.

To solve the problem for four robots, a dense RRT tree was computed with 341,997 nodes and $Robostep = 5$. This tree was then reused to solve the problem for different locations. Results of these simulations are show in Fig. 9 and 10.

V. Hardware Experiments

Magnetic manipulation was used to test the RRT algorithm. The agents manipulated were two cylindrical permanent magnets having a radial magnetization. A workspace containing obstacles was designed and built using 3D printing (see fig. 12). The bottom of the workspace was filled with epoxy to produce a smooth flat surface. The epoxy was set to cure inside the magnetic manipulator to ensure that the surface is leveled during the experiments. A thin layer of canola oil was added to facilitate the steering of the agents.

The magnetic manipulator was configured to generate a magnetic flux density having a magnitude of 3 mT. When the agents were placed inside the workspace their magnetization aligned with the applied magnetic field. The magnetic field was then slowly rotated to move the agents.

Two reference frames were defined (see Fig. 11). The first, noted *xyz*, is linked to the workspace. The second reference frame, noted *uvw* is linked to the agents and facilitates the calculation of the magnetic field. Axis *u* is collinear to the agent revolution axis and *w* is collinear to the *z* axis. Axis *v* is situated within the *xy* plane. The agents performed two types of movements: rolling and steering.

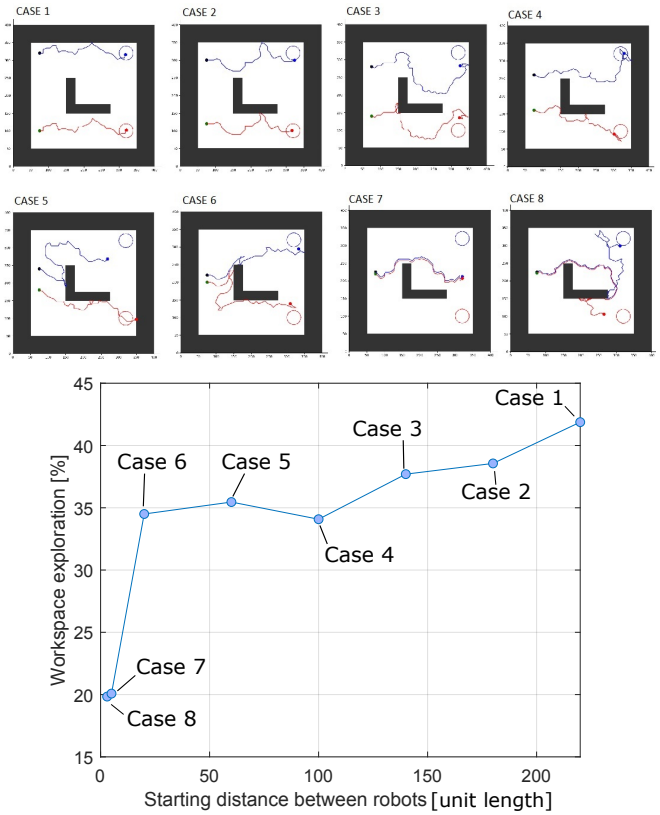

Fig. 7. Top: Maps with different initial configurations. The initial distance between two robots decreases from case 1 to case 8. Bottom: Graph of percentage coverage for all the cases presented above.

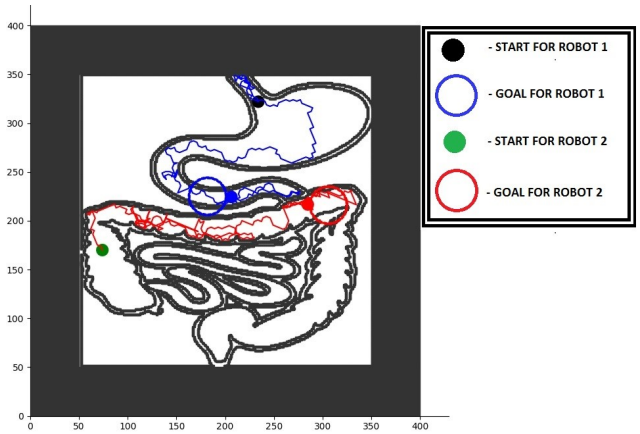

Fig. 8. 2D environment representing the human digestive system.

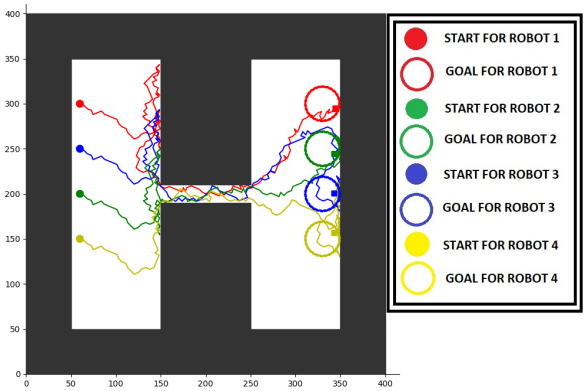

Fig. 9. Solving environment 3 for four robots.

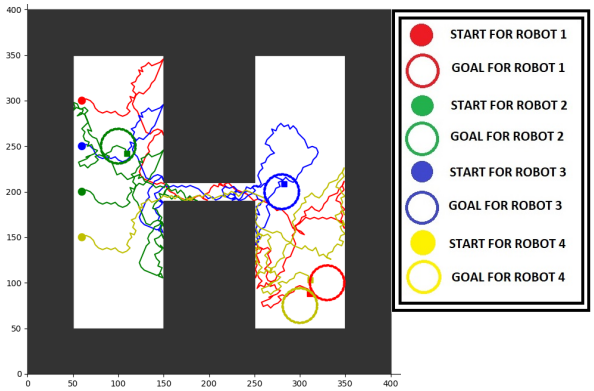

Fig. 10. Reusing an RRT generated for four robots.

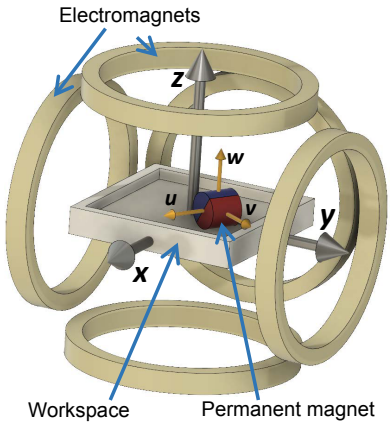

Fig. 11. Schematic drawing of the experimental setup and presentation of the reference frames used during the magnetic field calculation. The manipulator has six electromagnets. The electromagnet situated on $x+$ is not represented to show the workspace.

a) Rolling Movement: During rolling movements, the cylinders roll around their revolution axis and change position. This movement is accomplished by keeping $\mathbf{B}$ within the *vw* plane and rotating it around the *u* axis (rotational speed ω oriented along the *u* axis). The equation for $\mathbf{B}$ corresponding to this rotation is shown in (1) where B_0 is the magnitude of the applied flux density and t_i is the time at which the rolling movement begins. Complete rotations (360 degrees) are always performed. They always start and end with $\mathbf{B}$ inside the *uv* plane. During the next movement, steering, the magnetic field is always kept inside the *uv* plane. By starting and ending rolling movement with $\mathbf{B}$ inside the *uv* plane the continuity of $\mathbf{B}$ is ensured which produces a smooth transition between the two types of movements.

$$\begin{bmatrix} B_u \\ B_v \\ B_w \end{bmatrix} = \begin{bmatrix} 0 \\ B_0 \cdot \cos(\omega \cdot t - t_i) \\ B_0 \cdot \sin(\omega \cdot t - t_i) \end{bmatrix} \quad (1)$$

b) Steering Movement: During steering movements, the cylinders rotate around the *w* axis and change orientation. Steering movements are effectuated before a rolling movement to orient the revolution axis of the cylinders perpendicular to the upcoming displacement. To perform the steering movement, $\mathbf{B}$ is kept within the *uv* plane and rotated around the *w* axis (rotational speed Ω oriented along the *w* axis). The equation for $\mathbf{B}$ corresponding to the steering movement is shown in eq. 2. θ_i corresponds to the orientation of the magnets at the beginning of the movement. This equation is expressed in the *xyz* reference frame because the *uvw* reference frame in this case rotates with the magnet.

$$\begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} B_0 \cdot \cos(\Omega \cdot t - \theta_i) \\ B_0 \cdot \sin(\Omega \cdot t - \theta_i) \\ 0 \end{bmatrix} \quad (2)$$

c) Electromagnets Current Calculation: The method presented in [20] and [21] was used to compute the current to apply to the electromagnets to produce the desired flux density at the center of the workspace. The flux density produced by an air-core electromagnets is linear with the current that circulates in it. In addition, the total flux density is the sum of the flux density produced by each electromagnet. These properties allow writing linear system (3) where $\mathbf{A}$ is the actuation matrix containing coefficients calculated from the Biot-Savart law [22] and I_i is the current circulating in electromagnet i. This system is underdetermined. A least square pseudo inverse is used to find a solution (see eq. 4) as this solution minimizes the norm of the current vector and therefore minimizes Joules losses and heating.

$$\mathbf{B}_{xyz} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 & I_6 \end{bmatrix}^t \quad (3)$$

$$\begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 & I_6 \end{bmatrix}^t = \left(\mathbf{A}^\top \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^\top \cdot \mathbf{B}_{xyz} \quad (4)$$

d) Magnetic Manipulator: The magnetic manipulator possesses six electromagnets arranged in a cubic shape. The electromagnets are placed inside G10 boxes and can be cooled by liquid Nitrogen [20]. This feature is not used in this study as the flux density produced is low (3 mT). The manipulator is powered by twelve Kepco BOP 50-20 power supplies. Each electromagnet is powered by two power supplies connected in series and can receive a maximum of 20A under 100V. A National Instruments IC31-73 industrial real-time controller is used to perform the computation of the currents to generate. A Basler Aca800 camera is mounted on the top of the workspace to monitor and record the movements of the agents.

e) Experimental Results: The workspace presented in Fig. 12 was used to experimentally test the developed algorithm. The obstacles of the workspace have their sides either oriented vertically or horizontally. The RRT was configured to only move the robots along these directions to ensure that the cylindrical magnets collide with the obstacles when their revolution axis is perpendicular to the walls. If this condition is not respected, the movements of the robots while in contact with the walls are unpredictable.

The modified RRT was used to compute the global input. 40,000 nodes were computed and a solution was found. The magnetic field to apply was then computed and applied to the workspace.

Fig. 12 shows the position of the robots at different instants and the trajectory followed. Both robots reach their goal.

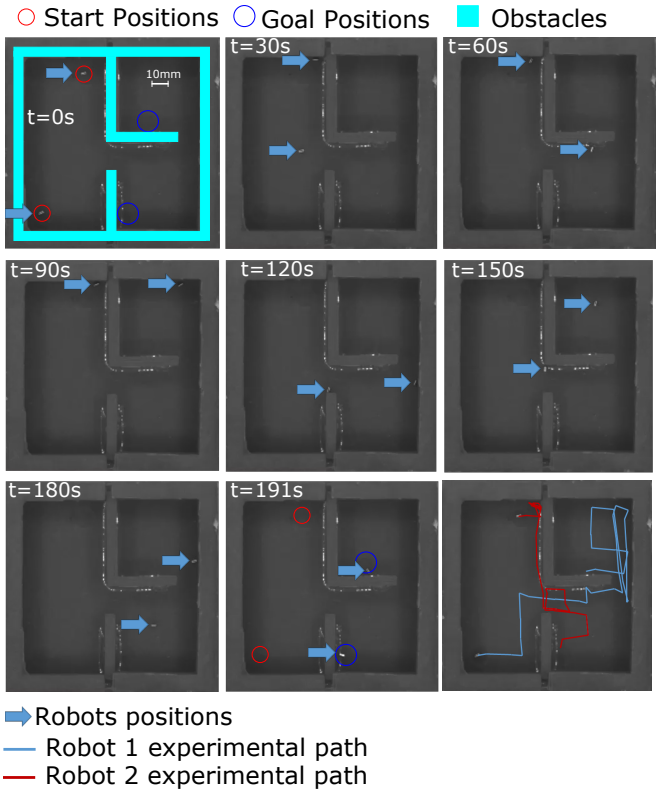

Fig. 12. The workspace used experimentally, the position of the robots at different times, and the experimental trajectory followed.

VI. Conclusion

Motion-planning for a swarm of robots is a challenging task when all the robots are moving under a global input and the inputs are not time-reversible. To address this challenge, we used open source libraries to create an RRT motion planner. Our code solution is available at github.com/pjparthjoshi/Swarm-Robots-OMPL.git. We also studied the effect of varying algorithm parameters on map coverage, including the movement step size, the number of RRT nodes, the grid size used to measure coverage, the initial positions of the robots, and different environments.

The hardware experiment demonstrated the feasibility of RRT motion planning for radially magnetized cylinders actuated by global inputs. Translating the path planned from the RRT required proper design of the workspace that allowed for non-slip rotations and accurate turning. The experiment is imperfect—the magnets sometimes deviate from the desired trajectories. Some instances of the experiment were unsuccessful because one of the magnets failed to reach the target due to an excessive accumulation of error. A solution to this problem was proposed in [23] for a single robot. Future work should implement a similar algorithm for multiple robots.

Additional algorithmic adjustments are needed to find a path that prevents the magnets from coming close enough to magnetically attract each other. Future works will, in addition, include different experiments with different shapes of robots, improving upon map coverage calculation algorithm, increasing the number of robots in the swarm, and consideration of the dynamics of the robots.

References

[1] S. Shahrokhi, J. Shi, B. Isichei, and A. T. Becker, "Exploiting nonslip wall contacts to position two particles using the same control input," *IEEE Trans. Robot.*, pp. 1–12, 2019.

[2] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin, "Massive uniform manipulation: Controlling large populations of simple robots with a common input signal," in *IROS*, Nov 2013.

[3] E. Diller, J. Giltinan, and M. Sitti, "Independent control of multiple magnetic microrobots in three dimensions," *Int J Rob Res*, vol. 32, no. 5, pp. 614–631, 2013.

[4] E. Diller, S. Floyd, C. Pawashe, and M. Sitti, "Control of multiple heterogeneous magnetic microrobots in two dimensions on nonspecialized surfaces," *IEEE Trans. Robot.*, vol. 28, no. 1, 2012.

[5] S. Martel, O. Felfoul, J.-B. Mathieu, A. Chanu, S. Tamaz, M. Mohammadi, M. Mankiewicz, and N. Tabatabaei, "Mri-based medical nanorobotic platform for the control of magnetic nanoparticles and flagellated bacteria for target interventions in human capillaries," *Int J Rob Res*, vol. 28, no. 9, pp. 1169–1182, 2009.

[6] A. Denasi and S. Misra, "Independent and leader follower control for two magnetic micro-agents," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 218–225, Jan 2018.

[7] F. Mishima, S.-i. Takeda, Y. Izumi, and S. Nishijima, "Development of magnetic field control for magnetically targeted drug delivery system using a superconducting magnet," *IEEE transactions on applied superconductivity*, vol. 17, no. 2, pp. 2303–2306, 2007.

[8] M. Arruebo, R. Fernández-Pacheco, M. R. Ibarra, and J. Santamaría, "Magnetic nanoparticles for drug delivery," *Nano today*, vol. 2, 2007.

[9] C. Sun, J. S. Lee, and M. Zhang, "Magnetic nanoparticles in mr imaging and drug delivery," *Advanced drug delivery reviews*, vol. 60, no. 11, pp. 1252–1265, 2008.

[10] N. S. Elbialy, M. M. Fathy, A.-W. Reem, R. Darwesh, U. A. Abdeldayem, M. Aldhahri, A. Noorwali, and A. A. AL-ghamdi, "Multifunctional magnetic-gold nanoparticles for efficient combined targeted drug delivery and interstitial photothermal therapy," *International journal of pharmaceutics*, vol. 554, pp. 256–263, 2019.

[11] D. De Lanauze, O. Felfoul, J.-P. Turcot, M. Mohammadi, and S. Martel, "Three-dimensional remote aggregation and steering of magnetotactic bacteria microrobots for drug delivery applications," *Int J Rob Res*, vol. 33, no. 3, pp. 359–374, 2014.

[12] P. Joshi, "Motion-planning using RRTs for a swarm of robots controlled by global inputs," Master's thesis, University of Houston, Houston, Texas, 77204, 2018.

[13] M. Salehizadeh and E. Diller, "Two-agent formation control of magnetic microrobots in two dimensions," *Journal of Micro-Bio Robotics*, vol. 12, no. 1-4, pp. 9–19, 6 2017. [Online]. Available: https://doi.org/10.1007/s12213-017-0095-5

[14] S. M. LaValle, "Rapidly-exploring Random Trees: A New Tool for Path Planning," in *TR 98-11, Computer Science Dept., Iowa State University*, Oct 1998.

[15] T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-free Paths among Polyhedral Obstacles," in *Communications of the ACM*, Oct 1979, pp. 560–570.

[16] A. Atramentov and S. M. LaValle, "Efficient Nearest Neighbor Searching for Motion Planning," in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, pp. 632–637.

[17] A. Yershova and S. M. LaValle, "Improving Motion Planning Algorithms by Efficient Nearest-neighbor Searching," in *IEEE Trans. Robot.*, 2007, pp. 151–157.

[18] C. Rodriguez, J. Denny, S. A. Jacobs, S. Thomas, and N. M. Amato., "Blind RRT: A Probabilistically Complete Distributed RRT," in *IROS*, 2013.

[19] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," in *IEEE Robot Autom Mag*, 2012, pp. 72–82.

[20] J. Leclerc, B. Isichei, and A. T. Becker, "A magnetic manipulator cooled with liquid nitrogen," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4367–4374, Oct 2018.

[21] M. P. Kummer, J. J. Abbott, B. E. Kratochvil, R. Borer, A. Sengul, and B. J. Nelson, "Octomag: An electromagnetic system for 5-dof wireless micromanipulation," *IEEE Trans. Robot.*, vol. 26, no. 6, 2010.

[22] J. C. Simpson, J. E. Lane, C. D. Immer, and R. C. Youngquist, "Simple analytic expressions for the magnetic field of a circular current loop," 2001.

[23] J. S. Lewis and J. M. OKane, "Planning for provably reliable navigation using an unreliable, nearly sensorless robot," *Int J Rob Res*, vol. 32, no. 11, pp. 1342–1357, 2013.