Autonomous Mobile Robot with Independent Control and Externally Driven Actuation

Hanlin Wang[1] and Michael Rubenstein[2]

Abstract— **Complexity, cost, and power requirements for actuation of individual robots are large factors in limiting the size of robotic swarms. Here we present a prototype robotic system that allows for externally powered motion in 2D without sacrificing individual autonomy, which simplifies the robot hardware, possibly enabling larger swarm sizes. This is accomplished using a table surface that is moving in an orbital fashion, and where robots can move to any point on the table surface simply through a series of carefully timed attachment and detachment steps. We present a model for the robot's motion, and use this model to create a motion controller that allows the robot to move from its current position to any other position on the table in approximately a straight line. We show this controller working in simulation as well as on an experimental hardware system.**

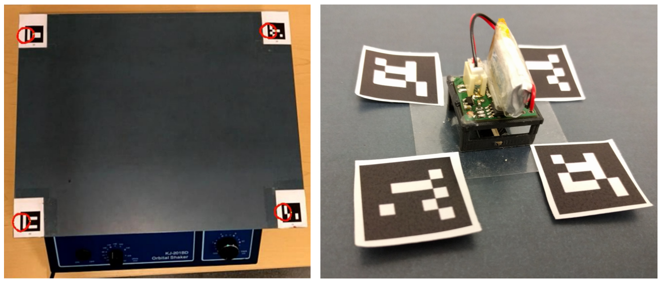

Fig. 1. Picture of the experimental system. A 40 cm x 30 cm shake table with paths (red) of the fiducials taken during orbit overlayed on image (left). Close up of robot (2 cm x 2 cm) with attached fiducials (right).

I. INTRODUCTION

Traditionally, robots used for swarm applications can independently control their position in the environment using on-board power, often in the form of batteries, coupled with multiple on-board actuators such as electric motors. This use of multiple self-powered actuators significantly contributes to a robot's complexity, as approximated by part count, hardware cost, assembly time, and stored energy requirements. This greatly contributes to limiting the size of robotic swarms, currently on the order of 100-1000 robots [1]–[3].

One approach that could reduce the complexity of a swarm robot, and therefore enable larger swarm sizes, is to use an apparatus that can create external forces which are utilized by robots for individual motion. While the apparatus to create these forces will contribute to the complexity of the overall system, where system complexity is (robot complexity $*$ number of robots $+$ apparatus complexity), it may be possible to scale this type of system to larger numbers as the robots may be less complex, and the apparatus complexity could be a constant or fixed cost if it is shared amongst all robots.

In past work, researchers have investigated using external forces to move and/or power robots, however, most are not able to scale to large numbers of independently controlled and autonomous robots. There are many approaches that move a passive robot, often on the micro-scale, through the use of external fields such as magnetic [4] or electro-static [5]. While this enables the robots to be very small, it does not provide autonomous control, and as the robot motion is controlled only by the external fields, the robot itself cannot decide its own motion. Additionally, if multiple robots are present, they will all be exposed to the same field, and will react and move in nearly identical ways, making it difficult to control multiple robots with the same external field.

Methods have been developed to take advantage of differences or variability between robots to allow for independent control of many robots with the same external field. These robot differences could be explicitly created, for example by creating individuals with varying physical properties that respond to the external field uniquely [5], but there is a limit to the number of robots that can be created. Another approach [6] takes advantage of naturally occurring differences between individuals, but the speed of control decreases as more robots are used. In both these cases, the individuals are still under control from the field; they do not make their own decisions about motion, so they still lack individual autonomy.

Other approaches use an apparatus that can control many localized forces on its surface, such as isolated electromagnetic fields or vibration [7], [8]. As these apparatuses generally address each location to be controlled individually, the complexity of the apparatus will increase as more robots are used and therefore require more addressable locations. In these systems, the individuals are still not autonomous as their motion is still solely decided by the external field applied to them.

Other systems use stochastic forces, such as thermal motion, random shaking, or fluid mixing [9]–[11] to power the movement of individuals. Generally the apparatus for applying stochastic forces is scalable to larger numbers, however, the motion of individuals is not controlled. Some

[1]Hanlin Wang is with the Mechanical Engineering Department, McCormick School of Engineering, Northwestern University `HanlinWang2015@u.northwestern.edu`

[2]Michael Rubenstein is faculty with the Mechanical Engineering Department, Electrical Engineering and Computer Science Department, McCormick School of Engineering, Northwestern University `Rubenstein@northwestern.edu`

stochastic systems have control whether a robot is moving or not, for example [14], but not its direction, and an individual cannot be guaranteed to reach a desired location in a bounded amount of time.

One approach makes use of external forces to reconfigure a group of modular robots [12], where all robots are on a table moving in an orbital manner, and robots can use this orbital force to rotate from one docking site to an adjacent one. This is the most similar related work to the work discussed here. The main difference is that the work in [12] only allows robots to reconfigure, they are not independently mobile, where the work discussed here allows each robot to move as a mobile robot.

We will present a robot that uses externally powered locomotion, but is autonomous and capable of controlled motion in 2D, with no theoretical limit to the number of robots. The robot and apparatus are described in section 2, analysis of motion and position control is described in section 3, and experimental tests of this controller in real and simulated systems are presented in section 4. A video demonstrating the results of this work is included in the supplementary files.

II. Hardware and Simulation

Similar to the approach in [12], the approach described here uses an orbital shake table to apply an external force to the robots. This table has a flat 2D surface that moves in an orbital manner, where the axis of orbit is normal to the table, see fig. 1. The robots sit on this surface, and have the ability to strongly attach to the surface below, or release from it. If attached they move in a circular path, and if released, their motion is a function of their velocity at release, their inertia, and the friction between the robot and the table surface. By controlling when it attaches and detaches from the table, a robot can control its motion along the table surface, see fig. 2 for an example of this motion in both the real and simulated system. To analyze and test the behavior of this robot system, we used both experimental hardware, as well as a physically realistic simulation.

A. Hardware

The table consists of a steel plate mounted to an orbital shake table, see fig. 1, commonly used for mixing liquids. The table orbits at approximately 3.8Hz and has an orbital diameter of 22mm. To attach to the table, the robot, shown in fig. 1, uses an electro-permanent magnet (EPM), described in [13], which is mounted on its bottom in contact with the table. The EPM can be switched on or off by applying 20V across its coil for 80 μs creating a brief current pulse of approximately 3A. By controlling the direction of current, the magnetic field of the EPM can be switched on or off, where the desired field is maintained with no power consumption until the next switching event. When the field is on, the robot is pulled strongly to the table surface, quickly and firmly attaching it to the table surface. When the field is off, the robot detaches from the table.

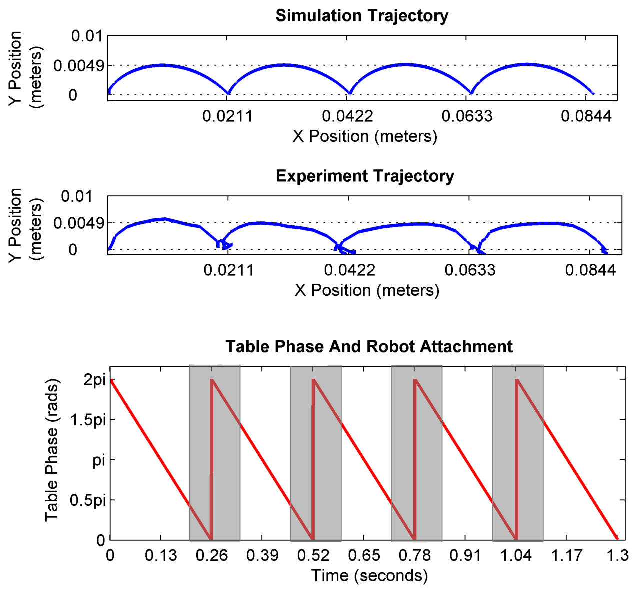

Fig. 2. Example motion path of robot moving in a straight line in simulation (top) and hardware (center). The path for hardware is measured using camera tracking. The phase, or rotation angle of the table, along with the robot control signal is shown in (bottom). The time marked in gray is time where the robot is detached, and white is the time where the robot is attached.

The robot is powered by a 110mAh 3.4V lithium battery, and the battery voltage is increased to 20V with an on-board boost regulator. An H-bridge is used to connect the 20V to the EPM and control the direction of current. The H-bridge is controlled by an Atmega micro-controller, which gives the robot autonomy in switching the EPM on and off, and therefore autonomy in its motion. In the ideal case (ignoring electrical losses and micro-controller power usage), the battery can supply enough energy for approximately 300,000 switching events. Given the table orbit radius, this is enough to allow the robot to move approximately 3.3 kilometers.

The prototype robot has no sensors; however, the robot and the orbital shake table are marked with fiducials to track the position of the robot as well as the table. This tracking is not used for controlling the robot; it is used solely for capturing experimental results. All figures in this paper of robot experiments extract these marker positions from video to show the robot's position on the shake table surface.

B. Simulation

A physics based simulation was also created to aid in testing the control and design. To simulate the robot's motion on the shake table, it is modeled as a point mass with Coulomb friction between the robot's bottom and the shake table, and the friction is isotropic. The table's rotation frequency is constant. The properties of the system: robot mass (5 grams), magnetic attachment force (5 Newtons), coefficient of friction (0.31), table frequency (3.8 Hz), and rotation diameter (0.022 meters), are all chosen to closely match the real system. A robot has two behaviors: attach to the table and detach from the table. When the robot is attaching to the table, the normal force between the robot and the table surface is the combination of gravity and the magnetic attraction force, and when robot detaches the normal force is only due to gravity. The simulated robot runs an autonomous controller that can switch between these two states instantly.

III. Position Control

When a robot releases from the orbiting table, it will begin to move relative to the table surface. This motion is a function of the angular phase of the table orbit at release, the table's frequency of motion, the radius of the table's orbit, the robot's mass, and the friction between the robot and the table surface. Here we will describe the robot's motion after release in terms of these variables, as well as present a method to control the robot so that the path it takes approximates a straight line towards any goal location on the table surface.

A. Motion Description

The analysis of the robot trajectory is essentially the analysis of the relative motion between the robot and the table. To describe this relative motion problem we introduce two reference frames:

1) The world frame: s which is fixed in the space, and has an origin in the center of the table, which does not move with the table's orbital motion.
2) The table frame: T which is attached to the table, origin is the center of the table surface, and moves with the table surface during its orbital motion.

First, consider the scenario that there is ideal friction between robot and table, which means there is zero friction when the robot moves relative to the table, and there is infinite friction when it attaches to the table, i.e., the robot can move friction-free and stop on the table immediately. The table's motion is expressed in the s frame as follows:

$$x_{ts} = r\cos(wt)$$

$$y_{ts} = r\sin(wt)$$

where, r is the table orbit radius and w is angular velocity. We will use the term orbital phase to represent the current angle of the table in its rotation. The instantaneous velocity of any point on the table in the s frame follows accordingly:

$$v_{x_{ts}} = rw\cos(wt + \pi/2)$$

$$v_{y_{ts}} = rw\sin(wt + \pi/2)$$

Combine the above equations to compute the robot's motion in frame s at time t after detaching the table at time t_0:

$$x_{rs} = (t - t_0)wr\cos(wt_0 + \pi/2) + x_{0s}$$

$$y_{rs} = (t - t_0)wr\sin(wt_0 + \pi/2) + y_{0s}$$

where (x_{0s}, y_{0s}) is the robot's position in frame s at time t_0. The robot's position in frame T follows accordingly:

$$x_{rT} = x_{rs} - x_{ts} = (t - t_0)wr\cos(wt_0 + \pi/2) + x_{0s} - r\cos(wt)$$

$$y_{rT} = y_{rs} - y_{ts} = (t - t_0)wr\sin(wt_0 + \pi/2) + y_{0s} - r\sin(wt)$$

Substituting $\Delta t = t - t_0$ and $\phi = wt_0 + \pi/2$, we get:

$$x_{rT} = \Delta t wr\cos(\phi) + x_{0s} + r\cos(w\Delta t + \phi + \pi/2)$$

$$y_{rT} = \Delta t wr\sin(\phi) + y_{0s} + r\sin(w\Delta t + \phi + \pi/2)$$

Next, we introduce a more realistic model of friction to the system where friction is modeled as Coulomb friction: $f = \mu F_b$, in which μ is the friction coefficient between the robot and the table, and F_b is the normal force. The robot motion in frame s can be computed numerically with the following ODEs:

$$\ddot{x}_{rs} = -\frac{\mu F_b}{m}\left(\frac{\dot{x}_{rs} - \dot{x}_{ts}}{\sqrt{(\dot{x}_{rs} - \dot{x}_{ts})^2 + (\dot{y}_{rs} - \dot{y}_{ts})^2}}\right)$$

$$\ddot{y}_{rs} = -\frac{\mu F_b}{m}\left(\frac{\dot{y}_{rs} - \dot{y}_{ts}}{\sqrt{(\dot{x}_{rs} - \dot{x}_{ts})^2 + (\dot{y}_{rs} - \dot{y}_{ts})^2}}\right)$$

with the initial conditions of

$$\begin{aligned} \dot{x}_{rs}|_{t=t_0} &= v_{x_{ts}}|_{t=t_0} \\ \dot{y}_{rs}|_{t=t_0} &= v_{y_{ts}}|_{t=t_0} \\ x_{rs}|_{t=t_0} &= x_{ts}|_{t=t_0} \\ y_{rs}|_{t=t_0} &= y_{ts}|_{t=t_0} \end{aligned}$$

in which t_0 is the time when the robot detaches, and m is the robot's mass. Using the numeral solution we can compute the robot's motion in frame T. Substituting $\Delta t = t - t_0$ and $\phi = wt_0 + \pi/2$, the robot's motion in frame T is:

$$\begin{aligned} x_{rT} &= x_{rs} + r\cos(w\Delta t + \phi + \pi/2) \\ y_{rT} &= y_{rs} + r\sin(w\Delta t + \phi + \pi/2) \end{aligned} \tag{1}$$

B. Motion Controller

The characteristics of the robot's cycloid-like motion on the table makes the robot difficult to control as it cannot move on a straight line path. Here we will introduce a controller where given a robot's starting position on the table A and a desired goal position B, the controller will automatically generate a series of release and attachment times to move the robot to position B as quickly as possible while never moving farther than a user defined distance, or offset tolerance α, from the line segment $\overline{AB}$. To simplify the controller, we also add the following constraints:

1) The robot must eventually stop exactly at the goal point B.
2) Every time the robot attaches to the table during the motion, it must stop exactly on the line segment $\overline{AB}$.

At the core of the robot controller is a precomputed array that specifies that for a given offset tolerance α, at what phase of the table's orbit it should release, how long until it should re-attach, and how far in the desired direction it will have traveled. It is assumed that the friction, μ, table orbit radius r, and table orbit angular velocity, w, are fixed and known. We start by first making the simplifying assumption that the angle, γ, between the desired trajectory, $\overline{AB}$, and the X axis in frame t is zero.

The array is iteratively generated by the following process: with the robot starting at position $(0, 0)$, choose a table phase,

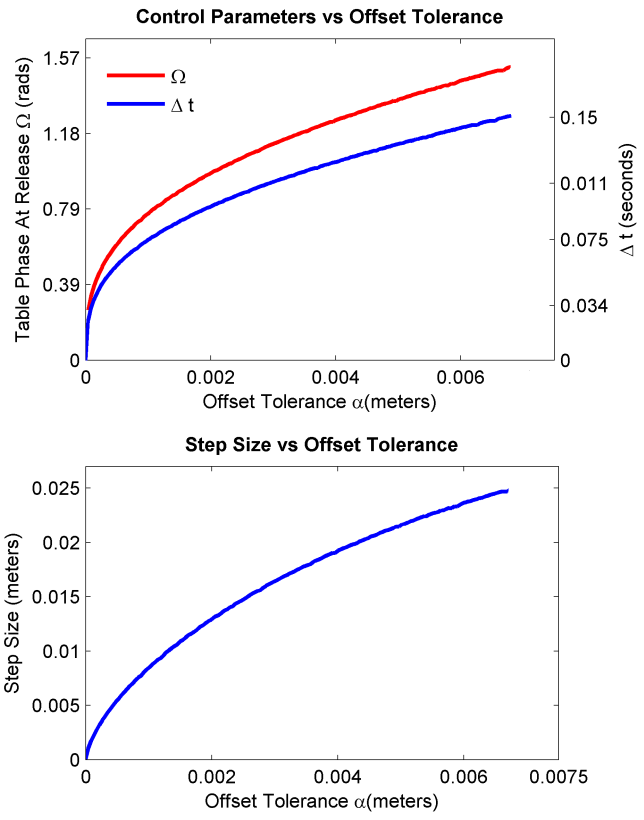

Fig. 3. Graphical representation of the precomputed controller array showing the table phase to detach, time between detach and attach (top), and expected distance traveled towards the goal point after re-attachment (bottom).

Ω, ranging from 0 to 2π at a step size ϵ, and use that Ω to initialize the ODE equations (1), and using the numerical method compute the robot's path when it releases at table phase Ω until either it crosses over the x axis, or the table orbits a full rotation without the robot crossing the x axis. During each of these numerical simulations using the ODEs, we record the maximum distance the robot travels in the y axis direction, and its new x position when the x axis is crossed. If the x axis is crossed, and the new x position is greater than its starting x position, zero, we place an entry in the array at the position corresponding to the robots maximum distance traveled in the y axis direction. The entry includes the starting table phase Ω, the change in x position, and time taken from release until the x axis is crossed, Δt . This array is generated offline and then given to the robot(s). A graphical representation of this array for the parameters of the experimental system is shown in fig. 3.

To move one step (one detach, attach cycle) towards the goal potion B, the robot first computes the angle, γ, between the desired trajectory, $\overline{AB}$ and the x axis direction, as well as the remaining distance from its current position to B. It looks into the array at the entry for the desired α and finds the distance it will move toward B. If that distance is less than the current distance to B, then the robot will detach for the time specified, Δt, in the array entry, detaching at the

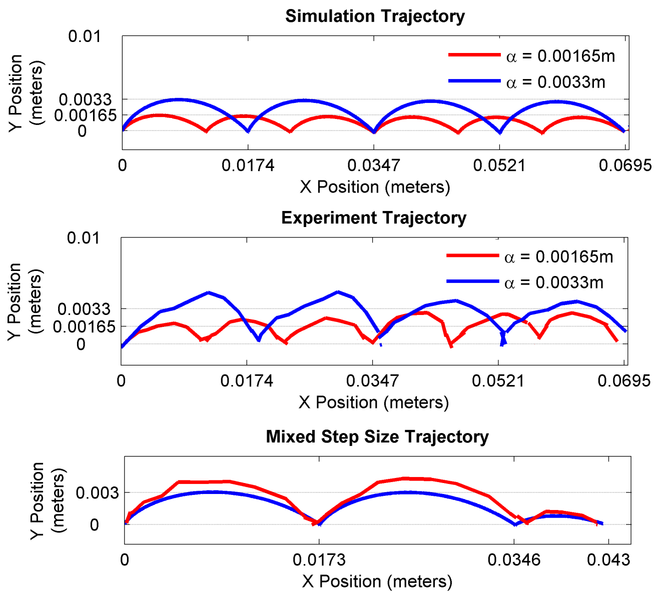

Fig. 4. Simulation (top) and experimental (center) paths for robot using different offset values α to move in a straight line along the x axis. The robot chooses a smaller step size for the final step (bottom) to avoid overshoot of goal location in experiment (red) and simulation (blue).

table phase specified at the entry, plus γ. If the distance in the array entry for α is greater than its current distance to B, i.e. the robot would move past B, then the robot searches the array by step size, finds the biggest step size which is smaller than the current distance between the robot and the destination B. It uses the table phase specified at that entry, plus γ, as well as the time specified, Δt, in that entry, to move, stopping on the point B. Upon completing one step, the robot updates its current position based on the distance it expected to move, and the direction of motion, and then progresses to the next step. See fig. 4 for examples of robot motion in simulation and experimentally using different α values, as well as changing step size to avoid overshoot of the goal location.

IV. Experimental Results

To test the controller running on the robot hardware, we tested the robots ability to move in a straight line with different values for offset tolerance, α, and tasked it with traversing a complex "N" shaped trajectory. These tests use a precomputed controller array that uses a friction coefficient μ of 0.31, which was experimentally measured, a rotation frequency of 3.8Hz and rotation radius of 0.011m, both of which can be manually set on the orbital table.

The robot uses its internal clock to keep track of the table's current orbital phase; however, since the prototype robot contains no sensors, it has no way of detecting the table's current orbital phase. As a result there will be a constant offset between the robots prediction of the table phase and the actual phase; the offset depends on the initial orbital phase of the table when the robot's program begins

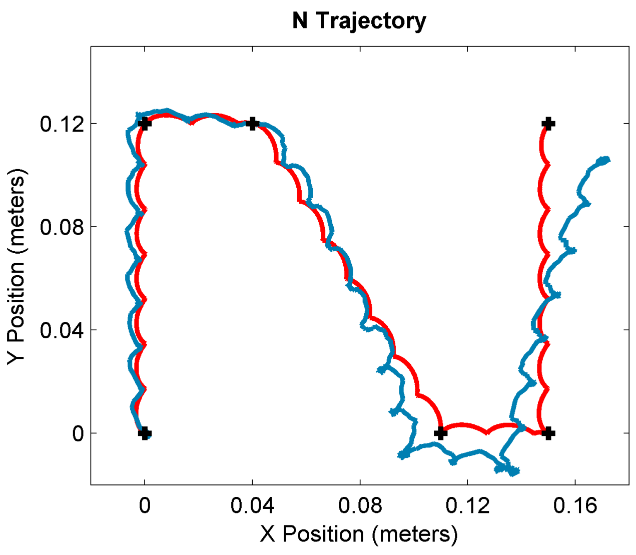

Fig. 5. Simulation (red path) and real robot (blue path) moving to desired way-points (shown as black crosses) to create an "N trajectory. The robot starts in the $(0, 0)$ position.

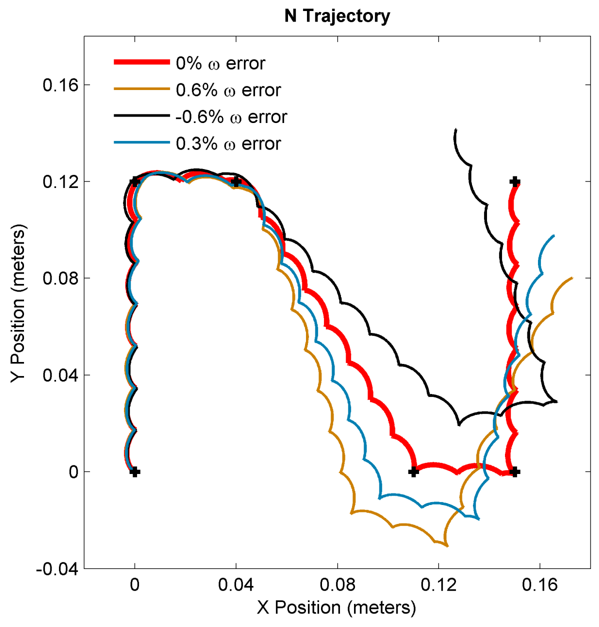

Fig. 6. Simulation runs of a robot creating the "N" trajectory with mismatches between the table frequency and the robot's estimation of table frequency. An error of 0.3% in the robot's estimated table frequency closely matches the path of the experimental test.

execution. This constant offset will cause the direction of all straight line motions to be in an angle that is the desired angle plus this constant offset. For example, this would cause the desired "N" trajectory to be arbitrarily rotated. To help correct for this, prior to starting any motion the robot briefly blinks an LED to indicate its belief of the table phase allowing the table phase to be manually adjusted.

For the first set of experiments, the robot was placed at position $(0, 0)$ and tasked to move in a straight line in positive x direction, once with α of 1.65mm, and once with α of 3.3mm. The path taken by the robot was computed by tracking the attached fiducials and can be seen in fig. 4. Additionally, the robot was tasked to reach a point using α of 3.3mm, and the goal point was chosen so that this α value would cause it to overshoot. The controller correctly chose a smaller α for the last step to reach the desired point, as shown in fig. 4.

To test a more complex multi-point trajectory we also tasked the robot with moving to a series of 5 way-points, shown on fig. 5, with an α of 3.3mm. The positions of some way-points did require the robot to make shorter steps to prevent overshoot of some way-points, for example near the lower right point of the "N".

V. Discussion and Future Work

Much of the non-ideal behavior of the experimental system is a result of the current lack of sensing on-board the robot. As can be seen in the experimental results, especially the "N" test in fig. 5, the angle of path taken by the robot can differ from the desired path and may drift over time. This is the result of the frequency of the table's orbit not exactly matching the expected value, which causes the robot's prediction of table phase to slowly drift away from the actual table phase. Fig. 6 shows different runs of the simulation where the robot initially knows the correct phase of the table, but the table frequency differs slightly from the robot's predicted table frequency. This results in the warping of the robot path, and an error of 0.3% in the robot's estimated table frequency closely matches the path of the experimental test from fig. 5. To correct this, future iterations of the robot will have an on-board IMU, and can use the accelerometer to accurately measure table frequency, reducing this type of error.

We also plan on including robot-to-robot sensing and communication in the next iteration of the robot, allowing multiple robots to interact in a distributed way. This will be used to test traditional swarm algorithms in large numbers, as well as investigate any effects this form of locomotion has on a swarm system.

References

[1] Rubenstein, Michael, Alejandro Cornejo, and Radhika Nagpal. "Programmable self-assembly in a thousand-robot swarm." Science 345, no. 6198 (2014): 795-799.

[2] Konolige, Kurt, Dieter Fox, Charlie Ortiz, Andrew Agno, Michael Eriksen, Benson Limketkai, Jonathan Ko et al. "Centibots: Very large scale distributed robotic teams." Experimental Robotics IX 21 (2006): 131-140.

[3] Wurman, Peter R., Raffaello D'Andrea, and Mick Mountz. "Coordinating hundreds of cooperative, autonomous vehicles in warehouses." AI magazine 29, no. 1 (2008): 9.

[4] Kummer, Michael P., Jake J. Abbott, Bradley E. Kratochvil, Ruedi Borer, Ali Sengul, and Bradley J. Nelson. "OctoMag: An electromagnetic system for 5-DOF wireless micromanipulation." Robotics, IEEE Transactions on 26, no. 6 (2010): 1006-1017.

[5] Donald, Bruce R., Christopher G. Levey, Craig D. McGray, Igor Paprotny, and Daniela Rus. "An untethered, electrostatic, globally controllable MEMS micro-robot." Microelectromechanical Systems, Journal of 15, no. 1 (2006): 1-15.

[6] Becker, Aaron, Cem Onyuksel, Timothy Bretl, and James McLurkin. ”Controlling many differential-drive robots with uniform control inputs.” The international journal of Robotics Research 33, no. 13 (2014): 1626-1644.

[7] Pelrine, Ron, Annjoe Wong-Foy, Brian McCoy, Dennis Holeman, Rich Mahoney, Greg Myers, Jim Herson, and Tony Low. ”Diamagnetically levitated robots: An approach to massively parallel robotic systems with unusual motion properties.” In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 739-744. IEEE, 2012.

[8] Coutinho, Murilo G., and Peter M. Will. ”Using dynamic vector force fields to manipulate parts on an intelligent motion surface.” In Assembly and Task Planning, 1997. ISATP 97., 1997 IEEE International Symposium on, pp. 200-205. IEEE, 1997.

[9] White, Paul, Viktor Zykov, Josh C. Bongard, and Hod Lipson. ”Three Dimensional Stochastic Reconfiguration of Modular Robots.” In Robotics: Science and Systems, pp. 161-168. 2005.

[10] Bishop, J., S. Burden, Eric Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen. ”Self-organizing programmable parts.” In International Conference on Intelligent Robots and Systems, pp. 3684-3691. 2005.

[11] Gilpin, Kyle, Ara Knaian, and Daniela Rus. ”Robot pebbles: One centimeter modules for programmable matter through self-disassembly.” In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 2485-2492. IEEE, 2010.

[12] White, Paul J., and Mark Yim. ”Scalable modular self-reconfigurable robots using external actuation.” In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pp. 2773-2778. IEEE, 2007.

[13] Knaian, Ara Nerses. ”Electropermanent magnetic connectors and actuators: devices and their application in programmable matter.” PhD diss., Massachusetts Institute of Technology, 2010.

[14] Groß, Roderich, Stphane Magnenat, Lorenz Kchler, Vasili Massaras, Michael Bonani, and Francesco Mondada. ”Towards an autonomous evolution of non-biological physical organisms.” In European Conference on Artificial Life, pp. 173-180. Springer Berlin Heidelberg, 2009.