2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Algorithms for Simultaneous Motion Control of Multiple *T. Pyriformis* Cells: Model Predictive Control and Particle Swarm Optimization

Yan Ou[1] and Peter Kang[1] and Min Jun Kim[2] and A. Agung Julius[1]

***Abstract*— This paper investigates the use of single control signal (magnetic field direction) and PSO-MPC algorithm to control multiple magnetized *Tetrahymena pyriformis* (*T. pyriformis*) cells to move from their initial positions to their target positions simultaneously while avoiding the obstacle. The magnetized *T. pyriformis* cells are generated by adding iron-oxide spherical particles into the cells. We control the cells' moving direction by changing the magnetic field direction. Based on Model Predictive Control (MPC) algorithm, we define a cost function which is composed of the target cost function and the obstacle potential function. The target cost function is to measure the sum of differences between cells' predicted positions and their target positions. The obstacle potential function is used to measure the repulsive force of the obstacle. The input variables of the cost function are the sequence of control signals. We use Particle Swarm Optimization (PSO) method to find a cost value which is close to the global minimum of the cost function. In the experimental result section, we show the control of three m3pi robots to move from their initial positions to their target positions with avoiding the obstacle. Since the similar control strategy has successfully controlled one *T. pyriformis* cell in our previous work, we believe our PSO-MPC algorithm is applicable on the multiple *T. pyriformis* cells' control task.**

I. INTRODUCTION

In micro-scale robotics, micro artificial robots and microbiorobots are most widely investigated. It is much easier to model micro artificial robots [1], [2], [3], [4], [5] than microbiorobots. The models of the micro artificial robots have less uncertainty and they are more controllable than the microbiorobots. However, there are two drawbacks of the micro artificial robots. On one hand, their manufacturing costs are relatively higher than the microbiorobots. On the other hand, to power the micro artificial robots is not easy [6]. The rise of microbiorobots' research in recent years has overcome these drawbacks. Microbiorobots have found applications in many areas including micro-delivery [7], parallel assembly [8], [9], and micro-manipulation [10].

Microorganisms [11], [12], such as *Escherichia coli*, are easy and cheap to produce. In the low Reynolds number environment, the biomolecular motors embedded in the microorganisms, such as cilia and flagella, generate swimming force by consuming chemical energy from the fluidic environment. Using multiple stimuli, such as magnetic field, microorganisms can be controlled as microbiorobots.

[1]Yan Ou, Peter Kang, and A. Agung Julius are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA, Email: ouy2@rpi, kangp3@rpi.edu, agung@ecse.rpi.edu

[2]Min Jun Kim is with the Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA 19104, USA, Email: mkim@coe.drexel.edu

T. pyriformis is a eukaryotic pear-shaped cell with size 50 μm long and 25 μm wide. *T. pyriformis* cell is larger than many other microorganisms, such as *Escherichia coli* [13]. Therefore, it is easy for us to receive imaging feedback information of *T. pyriformis* cells and then control them. The body of *T. pyriformis* is covered by approximately 600 cilia, both oral and locomotive. The locomotive cilia facilitate the swimming behavior of *T. pyriformis*, which can be influenced by the external stimuli, such as magnetic field [14].

Researchers have used many path planning and control algorithms to control the *T. pyriformis* cells as microbiorobots. Kim *et al* [15] use real-time feedback control and the Rapidly-exploring Random Tree (RRT) for path planning to control the magnetotactic *T. pyriformis* as a microbiorobot. Ou *et al* [14] investigate the motion control of the *T. pyriformis* cell using Model Predictive Control algorithm. In our previous work [16], we use Feedback Ensemble Control algorithm to control multiple *T. pyriformis* cells simultaneously using single control input, magnetic field direction. A Control Lyapunov Function (CLF) is defined to measure the sum of distances between the robots' current positions and the target positions. The control algorithm consists of two control modes: the mode with rotating magnetic field and the mode with the magnetic field turning off. The rotating magnetic field is used to make the robots turn in a spiral pattern to find the feasible moving directions in order to reduce the CLF value. The mode without magnetic field makes the robots move straight to reduce the CLF value until the CLF value has an increasing tendency. The control signal is changed between those two modes in order to gradually reduce the CLF value. However, there are three drawbacks of the Feedback Ensemble Control algorithm. Firstly, the high-frequency rotating magnetic field is harmful to the cells. Secondly, the Feedback Ensemble Control technique requires a large experimental space for the motion of the cells. Since it is hard to generate a stable magnetic field in a relatively large space with constant intensity, the Ensemble Control technique is not applicable in the experiment of multi-cell motion control. Thirdly, using this algorithm, the robots move to their targets with large steady-state errors.

In this paper, we use PSO-MPC algorithm to control multiple robots simultaneously to move from their initial positions to their target positions with avoiding the obstacle. MPC stands for Model Predictive Control [17] while PSO stands for Particle Swarm Optimization [18]. Using this method, we will not generate the rotating magnetic field which is harmful to the cells. We can achieve our control objective within a smaller field of view, which makes it easier

978-1-4799-6923-4/15/$31.00 ©2015 IEEE

to do the experiment than the case when we use the Feedback Ensemble Control algorithm. With PSO-MPC algorithm, the cells end up with smaller distances to their targets than those of the Feedback Ensemble Control algorithm. We use an obstacle potential function to represent the repulsive force of the obstacle. We prove the effectiveness of our control algorithm by running an experiment using the m3pi robot. The ratio between the area of the field of view of the m3pi robots to their speed is set roughly the same with the corresponding ratio in the system that controls the *T. pyriformis* cells [14]. Since the similar control strategy has successfully controlled single *T. pyriformis* cell experimentally in our previous work [14], we believe the PSO-MPC control strategy for multiple m3pi robots is applicable on the *T. pyriformis* cells. Please refer to Section IV for details about PSO-MPC algorithm.

This paper is structured as follows. Section II shows the problem formulation. Section III shows the approach to deal with our control objective. Section IV shows the PSO-MPC algorithm. Section V describes the Simulation results and Experimental results of controlling the m3pi robots.

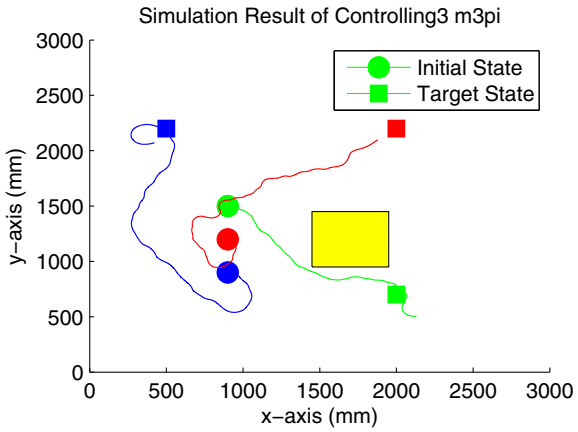

Fig. 1: Simulation result. In this figure, the circles with green, blue, and red colors are initial positions of the robots; the squares are the target positions of the robots; the yellow circle is the obstacle. All the robots are controlled using the single control input. And they reach the targets simultaneously with avoiding the obstacle.

II. PROBLEM FORMULATION

A. Plant Model

In our previous work [19], we introduce the experiment setup to control *T. pyriformis* cells as microbiorobots as well as the system identification of *T. pyriformis* cells. The plant model of the *T. pyriformis* cell is as follows,

$$\dot{x}_i = v_i \cos(\theta_i), \tag{1}$$
$$\dot{y}_i = v_i \sin(\theta_i), \tag{2}$$
$$\dot{\theta}_i = a_i \sin(u - \theta_i), \tag{3}$$

where x_i is the x axis position of the ith cell; y_i is the y axis position of the ith cell; θ_i is the ith cell's orientation; u is the magnetic field direction (the magnetic field strength is set to be constant); v_i is the speed of the ith cell, which is constant during the experimental process.

By discretizing the continuous-time plant model of the *T. pyriformis* cell, we get the following discrete-time plant model,

$$x_i(k+1) = x_i(k) + T v_i \cos(\theta_i(k)), \tag{4}$$
$$y_i(k+1) = y_i(k) + T v_i \sin(\theta_i(k)), \tag{5}$$
$$\theta_i(k+1) = \theta_i(k) + T a_i \sin(u(k) - \theta_i(k)), \tag{6}$$

where k is the control step; T is the sampling time of the control algorithm.

B. Control Objective

As is shown in Figure 1, we are trying to use single control signal (magnetic field direction) to control multiple robots to move from their initial positions to their target positions simultaneously while avoiding the obstacle.

III. APPROACH

In this paper, the control objective is twofold. On one hand, we want to control multiple *T. pyriformis* cells to move from their initial positions to their targets simultaneously using single control input (magnetic field direction). On the other hand, the cells need to avoid the obstacle in the field of view.

Considering the target cost function and the obstacle potential function that measures the obstacle repulsive force, we can define the following cost function,

$$J(\mathbf{u}) \triangleq f(\mathbf{u}) + \omega_p P(\mathbf{s}_i(k+1|k), \ldots, \mathbf{s}_i(k+h+1|k)), \tag{7}$$
$$f(\mathbf{u}) \triangleq \sum_{i=1}^{N} \|\mathbf{s}_i(k+h+1|k) - \mathbf{tar}_i\|_2, \tag{8}$$
$$P(\mathbf{s}_i(k+1|k), \ldots, \mathbf{s}_i(k+h+1|k)) \triangleq \sum_{i=1}^{N} \left(\sum_{j=1}^{h+1} p\left(\mathbf{s}_i(k+j|k)\right) \right), \tag{9}$$
$$\mathbf{s}_i(k+j|k) \triangleq [x_i(k+j|k), y_i(k+j|k)], \tag{10}$$
$$\mathbf{tar}_i \triangleq [x_{tar,i}, y_{tar,i}], \tag{11}$$
$$\mathbf{u} \in [\mathbf{lb}, \mathbf{ub}], \tag{12}$$

where k is the control step; J is the cost function which needs to be minimized to achieve the control objective; P represents the sum of the obstacle potential functions for all the robots at each control step; f is the cost function that measures the sum of distances between robots' positions after h control steps and their target positions; h is the predicted horizon; $\mathbf{u} \triangleq [u(k), u(k+1), \ldots, u(k+h)]$ is the sequence of control signals; $\mathbf{lb} = [-\pi, -\pi, \ldots, -\pi]$ is the lower bound and $\mathbf{ub} = [\pi, \pi, \ldots, \pi]$ is the upper bound of the sequence of control signals; p is the obstacle potential function as is described above; $\mathbf{s}_i(k+h+1|k)$ is the ith robot's $h+1$th step predicted position in the kth control step, which is gotten based on Equations (4)-(6); $\mathbf{tar}_i$ is the ith robot's target position; N is the cell number; x_{tari} is the target x axis position and y_{tari} is the target y axis position of the ith cell; ω_p is the weighting factor for the obstacle potential field function, which is chosen to achieve a balance between

reaching the targets and avoiding the obstacle.

In the kth control step, if a certain sequence of control signals $\mathbf{u}_{gb}$ makes $f(\mathbf{u}_{gb})$ close to 0, $\mathbf{u}_{gb}$ is a good candidate to control the robots to their targets. In the condition without modeling error, measuring error, and environmental disturbance, we can generate $\mathbf{u}_{gb}$ once and then use it to control the robots to move to their targets. However, we have to find $\mathbf{u}_{gb}$ at each control step to compensate the errors.

Many potential field functions [20], [21], such as Impulse function and Exponential function, can be used to represent the obstacle's repulsive force to the robots. In this paper, in order to reduce the computational time, we choose the impulse function as the obstacle potential function. The obstacle potential function is as follows,

$$p(\mathbf{s}) = \begin{cases} 1 & \text{if } \mathbf{s} \in O, \\ 0 & \text{if } \mathbf{s} \notin O, \end{cases} \tag{13}$$

where p is the obstacle potential function; $\mathbf{s} \triangleq [x, y]$ is the position of the robot; O is the obstacle.

IV. PSO-MPC ALGORITHM

A. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [18] is a random-walk optimization method that finds a small value which is close to the global minimum of a non-convex cost function. PSO generates a swarm of particles, and moves them in the search domain of the cost function. The velocity ($\mathbf{v}_n$) of each particle ($\mathbf{u}_n$) depends on its local best coordinate ($\mathbf{u}_{lb,n}$) and global best coordinate ($\mathbf{u}_{gb}$). $\mathbf{u}_{lb,n}$ is the minimum cost value of a particle along its searching history. $\mathbf{u}_{gb}$ is the minimum cost value among all particles' local best values. n is the index of the particles. The particles are guided by PSO to move toward the small value which is close to the global minimum of the cost function as shown in Figure 2 [22]. The values of $\mathbf{u}_n$, $\mathbf{v}_n$, $\mathbf{u}_{lb,n}$, and $\mathbf{u}_{gb}$ are updated based on Equations (14)-(19). PSO does not require the calculation of the cost function gradient, which makes it capable of generating a control strategy within the sampling time, which is $100ms$ in our project.

The equations to update the coordinates and the velocities of the particles are as follows,

$$\mathbf{v}_n^{new} = \omega_v \mathbf{v}_n^z + c_1 r_1 (\mathbf{u}_{lb,n}^z - \mathbf{u}_n^z) + c_2 r_2 (\mathbf{u}_{gb}^z - \mathbf{u}_n^z), \tag{14}$$

$$\mathbf{v}_n^{z+1} = \max(-|\mathbf{ub} - \mathbf{lb}|, \min(\mathbf{v}_n^{new}, |\mathbf{ub} - \mathbf{lb}|)), \tag{15}$$

$$\mathbf{u}_n^{new} = \mathbf{u}_n^z + \mathbf{v}_n^{z+1}, \tag{16}$$

$$\mathbf{u}_n^{z+1} = \max(\mathbf{lb}, \min(\mathbf{u}_n^{new}, \mathbf{ub})), \tag{17}$$

$$\mathbf{u}_{lb,n}^{z+1} = \{\mathbf{u}_n | \min(J(\mathbf{u}_n^{z+1}), J(\mathbf{u}_{lb,n}^z))\}, \tag{18}$$

$$\mathbf{u}_{gb}^{z+1} = \{\mathbf{u}_n | \min(\min_n J(\mathbf{u}_{lb,n}^{z+1}), J(\mathbf{u}_{gb}^z))\}, \tag{19}$$

where z is the iteration step; f is the cost function that measures the difference between the predicted final states and the target states; ω_v, c_1, and c_2 are positive numbers; ω_v is the momentum value; c_1 is the weighting factor of $\mathbf{u}_{lbi}$; c_2 is the weighting factor of $\mathbf{u}_{gb}$; r_1 and r_2 are random numbers with uniform distribution in the range of $[0, 1]$; $\mathbf{u}_n \in [\mathbf{lb}, \mathbf{ub}]$ is the coordinate of each particle; $\mathbf{v}_n \in [-|\mathbf{ub} - \mathbf{lb}|, |\mathbf{ub} - \mathbf{lb}|]$ is the velocity of each particle, $\mathbf{u}_{lb,n} \in [\mathbf{lb}, \mathbf{ub}]$ is the local best coordinate of each particle; $\mathbf{u}_{gb} \in [\mathbf{lb}, \mathbf{ub}]$ is the global best coordinate of the particles; $n = 1, 2, \ldots, np$ represents the particle index and np is the number of the particles; $\mathbf{lb} = [-\pi, -\pi, \ldots, -\pi]$ is the lower bound and $\mathbf{ub} = [\pi, \pi, \ldots, \pi]$ is the upper bound.

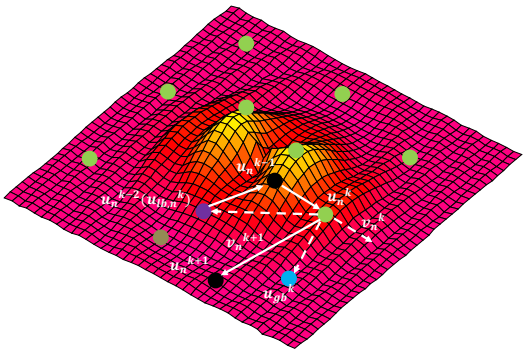

Fig. 2: Particle Swarm Optimization. In this picture, each green circle represents the current coordinate of a particle; the purple circle and the black circles are the history coordinates of a particle, among which the purple circle is the local best coordinate ($\mathbf{u}_{lb,n}^k$) of the particle; the blue circle is the global best record of all the particles ($\mathbf{u}_{gb}^k$); the gray circle is the global minimum of the cost function.

1) Pseudo-code of PSO: In summary, PSO with the stop criterion is presented in Algorithm 1.

Algorithm 1 $pso(\mathbf{u}_{gb}^0)$

```
Initialize u_n, v_n, and u_{lb,n} with uniformly distributed values in their searching spaces.
if u_{gb}^0 == NULL then
    u_{gb} = {u_{lb,n} | min_n(J(u_{lb,n}))}
else
    u_{gb} = u_{gb}^0
end if
for z = 1, 2, ..., z_max do
    Update v, u_n, u_{lb,n}, and u_{gb} based on Equation (14)-(19).
    if z > 10 and the stop criterion (20) is met then
        break
    end if
end for
return u_{gb}
```

2) Stop Criterion: Since Particle Swarm Optimization is a random-walk optimization strategy, there is no standard stop criterion with a proof of global convergence to find the global minimum. Here, we use a stop criterion based on the

decrease tendency of the cost function value as follows,

$$C^z = \frac{\sum_{m=z-10}^{z} \log(J(\mathbf{u}_{gb}^m))}{2},$$

$$\Delta^z = \sum_{m=z-9}^{z} (\log(J(\mathbf{u}_{gb}^{m-1}) - J(\mathbf{u}_{gb}^m)))$$

$$\text{if } z > 10 \text{ and } \Delta^z < C^z, \text{ stop the optimization process,} \quad (20)$$

where z is the iteration step; C^z is the stop criterion value; Δ^z represents the decreasing tendency. From Equation 20, we find that as the decrease of the cost function value, the stop criterion value also decreases. As shown in Figure 3, the $J(\mathbf{u}_{gb})$ value decreases exponentially with the iteration step.

Figure 3 shows the decreasing tendency of the cost function value in all iteration steps. At the initial iteration steps, the cost function value decreases quickly. But the decreasing tendency of the cost function value becomes smooth and steady after few iteration steps. If the decreasing tendency of the past 10 steps' cost function values (Δ) is smaller than a certain stop criterion value (C), the iteration is stopped.

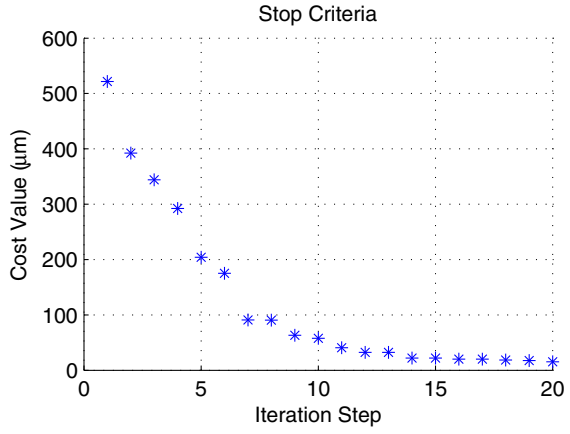

Fig. 3: The stop criterion of the Particle Swarm Optimization method. Asterisks represent $f(X_{gb})$ values in different iteration steps; the $f(X_{gb})$ value decreases gradually and satisfies the stop criterion in step 20.

B. Model Predictive Control

MPC has long been designed as a controller for industrial applications and academic studies [17]. In this paper, a cost function (7) is defined to measure the sum of differences between the final predicted positions and the target positions at each control step. Many optimization methods can be used to find the global minimum (sequence of control signals) of the cost function. After the optimization process, the current-step control signal is exerted to the system. However, it is hard and takes long time to find the global minimum of the cost function (7). The difficulty is twofold. On one hand, since all the robots receive the same control signal, it takes a long control horizon to move the robots from their initial positions to the target positions simultaneously, which makes the cost function high dimensional. On the other hand, the cost function has multiple local minima with different cost values. As a real-time optimization solver, PSO rapidly finds a small value which is close to the global minimum of a high-dimensional cost function. Using the PSO algorithm, it is easy for us to experimentally control multiple robots using the single control signal.

1) Choosing the Control Horizon: At each control step, we need to choose a suitable control horizon. The control horizon is the length of the sequence of control signals that make the robots move to their targets. As is shown in Algorithm 2 Part I, we can find the global minimum of the cost function given different control horizons and then choose the global minimum with the smallest cost value as the sequence of control signals. However, it takes a long time to find the global minimum of each cost function. At the 0th control step, we can make the robots turn in a square shape repetitively while calculating the best control horizon. After we find the best control horizon and the robots move back to their initial states, we exert the control signal to the system. This method, if applied at each control step, would be time-consuming. Therefore, from 1st control step, we need to find a method to generate the control signals by trying shorter control horizons.

Warm Start method [23] can be used to reduce the computational time at each control step. In the previous ($k-1$th) control step, we already know a set of control signals ($\mathbf{u}_{(opt)}$) that move the robots to their targets. Therefore, in the current (kth) control step, $\mathbf{u}_{(warm)} = [\mathbf{u}_{(opt)}(2), \mathbf{u}_{(opt)}(3), \ldots, \mathbf{u}_{(opt)}(end)]$ would be a good candidate as the sequence of control signals. We can set $\mathbf{u}_{(warm)}$ as a good initial guess of the global minimum and run the PSO iteration from this initial guess. As is shown in Algorithm 2 Part II, we can run the optimization function with the control horizons around the horizon of $\mathbf{u}_{(warm)}$, then find the best control horizon as well as the sequence of control signals ($\mathbf{u}_{(opt)}$).

2) Pseudo-code of MPC: In summary, the MPC algorithm is presented in Algorithm 2.

V. EXPERIMENTAL RESULT

Experimentally, we test the effectiveness of the PSO-MPC algorithm using the m3pi robots. The m3pi robot consists of a 3pi robot base and a m3pi expansion board. As is shown in Figure 4, with the xbee module and the mbed microcontroller, m3pi robot can move on the ground based on the received signal from xbee. xbee is a wireless communication module. Using xbee, we can easily send wireless signal from PC to the m3pi robot.

The model of the m3pi robot is as follows,

$$\dot{x}_i = (v_{r,i} + v_{l,i})\cos(\theta_i)/2, \quad (21)$$

$$\dot{y}_i = (v_{r,i} + v_{l,i})\sin(\theta_i)/2, \quad (22)$$

$$\dot{\theta}_i = (v_{r,i} - v_{l,i})/(2r). \quad (23)$$

where v_{li} is the left wheel speed; v_{ri} is the right wheel speed; (x_i, y_i) is the coordinate of the ith robot;θ_i is the orientation of the robot; r is the distance from the wheel to the center of the robot. In order to control the m3pi robot as the T.

Algorithm 2 *mpc*

// Part I: At the 0th control step, try the possible control horizons and generate the set of control signals.
for $h = 1, 2, \ldots, h_{max}$ **do**
  $\mathbf{u}_{(h)} = pso(\text{NULL})$
end for
$\mathbf{u}_{(opt)} = \{\mathbf{u}_{(h)} | \min_h J(\mathbf{u}_{(h)})\}$
$\mathbf{u}_{(warm)} = [\mathbf{u}_{(opt)}(2), \mathbf{u}_{(opt)}(3), \ldots, \mathbf{u}_{(opt)}(end)]$
Exert $\mathbf{u}_{(opt)}(1)$ into the system. // Here, $\mathbf{u}_{(opt)}(1)$ is the 0th step control signal.
// Part II: From the 1st control step, use $\mathbf{u}_{(warm)}$ to generate the set of control signals.
for $k = 1, 2, \ldots, k_{max}$ **do**
  $H = length(\mathbf{u}_{(warm)})$ // $length()$ is a function to find the dimension of a vector.
  for $h = H-1, H, H+1$ **do**
    Obtain the feedback information of the robots.
    if The robots are close enough to the target **then**
      break
    end if
    if $h == H+1$ **then**
      $\mathbf{u}^0_{gb} = [\mathbf{u}_{(warm)}, 0]$
    else
      $\mathbf{u}^0_{gb} = [\mathbf{u}_{(warm)}(H-h+1), \mathbf{u}_{(warm)}(H-h+2), \ldots, \mathbf{u}_{(warm)}(H)]$
    end if
    $\mathbf{u}_{(h)} = pso(\mathbf{u}^0_{gb})$
  end for
  $\mathbf{u}_{(opt)} = \{\mathbf{u}_{(h)} | \min_h J(\mathbf{u}_{(h)})\}$
  $\mathbf{u}_{(warm)} = [\mathbf{u}_{(opt)}(2), \mathbf{u}_{(opt)}(3), \ldots, \mathbf{u}_{(opt)}(end)]$
  Exert $\mathbf{u}_{(opt)}(1)$ into the system. // Here, $\mathbf{u}_{(opt)}(1)$ is the kth step control signal.
end for

pyriformis cell, we need to transform the *T. pyriformis* cell's model to the m3pi robot model as follows,

$$v_{l,i} = v_i - a_i r \sin(u - \theta_i), \tag{24}$$
$$v_{r,i} = v_i + a_i r \sin(u - \theta_i), \tag{25}$$

Figure 5 shows the experimental result of the motion control of 3 m3pi robots using signal control input. We use the OptiTrack system with 120Hz sampling frequency to get the feedback of the robots. The control frequency is 5Hz. The initial positions of the robots are $[(900, 800), (900, 1200), (900, 1600)]mm$. The target positions of the robots are $[(1700, 500), (1700, 1500), (1700, 2000)]mm$. a values of the robots are $[6, 4, 3]rad/s$. The speeds of the robots are $[180; 200; 150]mm/s$. The obstacle is located at [left,bottom,width,height]= $[1200, 1100, 200, 200]mm$. Using the PSO-MPC algorithm, we are able to control the robots to move to their targets simultaneously with the total steady-state error 858.8mm while avoiding the obstacle. The total steady-state error is the sum of distances between the

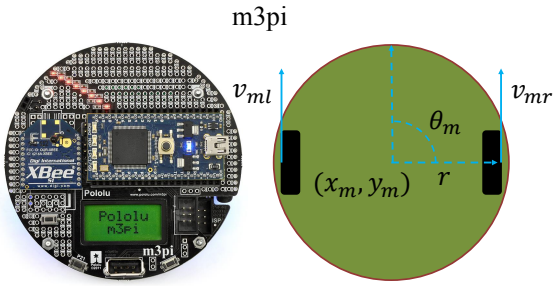

Fig. 4: m3pi robot. The left figure is the front view of the robot while the right figure is the back schematic view of the robot.

robots' final positions and the target positions. We repeat this experiment for 20 times. Among these 20 experiments, the mean value of the total steady-state error is 648.6mm and the standard deviation is 225.1mm. Since the total steady-state error is 858.8mm and we control 3 robots as is shown in Figure 5, each robot has a steady-state error around 280mm. The speed of the *T. pyriformis* cell is around $300\mu m/s$. Considering the speed ratio between the m3pi robot and the *T. pyriformis* cell, if we use the PSO-MPC algorithm to control *T. pyriformis* cells, we will possibly have steady-state error around $400\mu m$ for single cell. This steady-state error roughly equals to 8 *T. pyriformis* cell's body lengths, which is really small compared to our previous work [16]. The size of the field of view that controls the m3pi robots is $1600 \times 1800mm$. The size of the field of view that controls the *T. pyriformis* cells is $5000\mu m \times 5000\mu m$. Considering the speed ratio as well as the ratio of the field of view, we believe that the algorithm proposed in this paper can be applied to the *T. pyriformis* cells' control task.

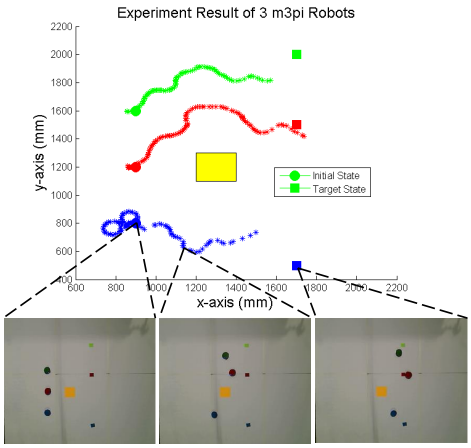

Fig. 5: Experimental result. In this figure, the circles with green, blue, and red colors are initial positions of the robots; the squares are the target positions of the robots; the yellow square is the obstacle. All the robots are controlled using the single control input. And they are trying to reach the targets simultaneously with avoiding the obstacle.

VI. CONCLUSION AND FUTURE WORK

In this paper, we control multiple robots simultaneously using the uniform control signal (magnetic field direction). All the robots are controlled to move from their initial positions to their target positions while avoiding the obstacle. Model Predictive Control is used to generate the control signal at each control step. The cost function is composed of the target cost function and the obstacle potential function. The variables of the cost function are the sequence of control signals. The target cost function is defined to measure the sum of distances between the robots' final predicted positions and their target positions. The obstacle potential function is used to measure the repulsive force of the obstacle. We use the Particle Swarm Optimization method to find a cost value that is close to the global minimum of the cost function and generate the current step's control signal. We test the effectiveness of this control algorithm by controlling 3 m3pi robots to move from their initial positions to their target positions with avoiding the obstacle. In the future, we are trying to implement this control algorithm to the microbiorobots control task.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from the NSF through grants number CMMI-10000284 and CMMI-10000255, and the ARO through grant number W911NF-11-1-0490 for the research reported in this paper. We are also thankful to Aaron Becker and Dalhyung Kim for the constructive discussions, from which this research benefited.

REFERENCES

[1] J. J. Abbott, Z. Nagy, F. Beyeler, and B. J. Nelson, "Robotics in the Small, Part I: Microbotics," IEEE Robotics Automation Magazine, vol. 14, no. 2, pp. 92–103, 2007.

[2] C. Pawashe, S. Floyd, and M. Sitti, "Multiple magnetic microrobot control using electrostatic anchoring," Applied Physics Letters, vol. 94, no. 16, p. 164108, 2009.

[3] L. Zhang, J. J. Abbott, L. Dong, B. E. Kratochvil, H. Zhang, K. E. Peyer, and B. J. Nelson, "Micromanipulation using artificial bacterial flagella," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1401–1406, 2009.

[4] M. R. Edwards, R. W. Carlsen, J. Zhuang, and M. Sitti, "Swimming characterization of serratia marcescens for bio-hybrid micro-robotics," Journal of Micro-Bio Robotics, vol. 9, no. 3, pp. 47–60, 2014.

[5] Z. Ye, C. Edington, A. J. Russell, and M. Sitti, "Versatile non-contact micro-manipulation method using rotational flows locally induced by magnetic microrobots," in Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on. IEEE, 2014, pp. 26–31.

[6] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, "Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system," Adaptive Behavior, vol. 17, no. 3, pp. 237–259, 2009.

[7] S. Martel, "Targeted delivery of therapeutic agents with controlled bacterial carriers in the human blood vessels," IEEE Conf. on Bio-Micro- and Nanosystems, p. 9, 2006.

[8] B. R. Donald, C. G. Levey, and I. Paprotny, "Planar microassembly by parallel actuation of MEMS microrobots," Journal of Microelectromechanical Systems, vol. 17, no. 4, pp. 789–808, 2008.

[9] S. Yim and M. Sitti, "Softcubes: Stretchable and self-assembling three-dimensional soft modular matter," The International Journal of Robotics Research, p. 0278364914527630, 2014.

[10] I. W. Hunter, S. Lafontaine, P. M. F. Nielsen, P. J. Hunter, and J. M. Hollerbach, "Manipulation and dynamic mechanical testing of microscopic objects using a tele-micro-robot system," IEEE Control Systems Magazine, vol. 10, no. 2, pp. 3–9, 1990.

[11] A. Itoh, W. Tamura, and T. Mishima, "Motion control of euglena group by weak laser scanning system and object manipulation using euglena group," Proceedings of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, pp. 43–47, 2005.

[12] A. Itoh, "Motion control of protozoa for bio-MEMS," Proceedings of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, pp. 27–32, 1999.

[13] B. Volkmer and M. Heinemann, "Condition-dependent cell volume and concentration of *escherichia coli* to facilitate data conversion for systems biology modelling," PLoS ONE, vol. 6, no. 7, 2011.

[14] 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. Journal of Robotics Research, vol. 32, no. 1, pp. 130–140, 2013.

[15] D. H. Kim, S. Brigandi, A. A. Julius, and M. J. Kim, "Real-time feedback control using artificial magnetotaxis with rapidly-exploring random tree (RRT) for *Tetrahymena pyriformis* as a microbiorobot," IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 3183–3188, May 2011.

[16] A. Becker, Y. Ou, P. Kim, M. J. Kim, and A. A. Julius, "Feedback control of many magnetized: *Tetrahymena pyriformis* cells by exploiting phase inhomogeneity," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 3317–3323, 2013.

[17] C. E. Garca, D. M. Prett, and M. Morari, "Model predictive control: Theory and practicea survey," Automatica, vol. 25, no. 3, pp. 335–348, 1989.

[18] R. C. Eberhart and S. Yuhui, "Particle swarm optimization: developments, applications and resources," Proceedings of Congress on Evolutionary Computation, vol. 1, pp. 81–86, 2001.

[19] 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," The International Journal of Robotics Research, 2012.

[20] H. H. Wesley, R. F. Brett, R. F. Jonathan, and H. W. William, "Visual navigation and obstacle avoidance using a steering potential function," Robotics and Autonomous Systems, vol. 54, no. 4, pp. 288 – 299, 2006.

[21] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential fields," IEEE Transactions on Robotics and Automation, vol. 8, no. 5, pp. 501–518, 1992.

[22] B. Yang, Y. Chen, and Z. Zhao, "Survey on applications of particle swarm optimization in electric power systems," IEEE Int. Conf. on Control and Automation, pp. 481–486, 2007.

[23] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," Control Systems Technology, IEEE Transactions on, vol. 18, no. 2, pp. 267–278, March 2010.