Controlling swarms of robots with global inputs: breaking symmetry

Aaron T. Becker

N338 Engineering Building 1 (D), Houston, TX 77204-4005, USA

Abstract

Roboticists, biologists, and chemists are now producing large populations of simple robots, but controlling large populations of robots with limited capabilities is difficult, due to communication and onboard-computation constraints. This chapter provides controllability proofs for control of mobile robots that move in a 2D workspace where each robot receives exactly the same control inputs.

We focus on two types of control: when control inputs are the desired angular and linear velocity for the robots, and secondly when control inputs are the desired direction and speed for the robots. Both use broadcast control inputs: the first uses control inputs specified in the local reference frame of each robot, while the second uses control inputs specified in the global reference frame. Each method allows steering each robot to a desired goal location in $O(n^2)$ time, but the second option enables a class useful of swarm manipulation tasks to be accomplished efficiently.

Keywords: micro robot, nano robot, global inputs, nonprehensile, under actuation

2016 MSC: 00-01, 99-00

1. introduction

Micro and nano robotics hold great promise for precision material delivery and for micro construction. According to Sitti et al. “One of the highest potential scientific and societal impacts of small-scale (millimeter and submillimeter
size) untethered mobile robots would be their healthcare and bioengineering applications [1].

The flagship application for precision delivery is targeted therapy [2, 3, 4, 5]. The current dominant practice in cancer treatment uses chemotherapy applied systemically, but has associated collateral damage to hair follicles and other fast-growing tissue. Successful precision delivery would steer toxins directly to tumors.

Health care is also a driver for micro construction, which includes minimally invasive surgery [7, 8, 9, 10], and tissue engineering [11, 12, 13, 14, 15]. The surgical industry is rapidly switching to minimally invasive surgery, which places surgical instruments at the end of long, slender kinematic chains and inserts this apparatus through small incisions, called ports, in the body. There is a desire to shrink the size and number of ports, but as the ports decrease in size, the kinematic chain to the external world becomes less rigid. This flexibility makes dexterity difficult and limits surgical forces.

Long before the advent of minimally invasive surgery, authors dreamed of doing away with incisions and kinematic chains by shrinking the surgeon and tools into a compact submarine-like vehicle that could be piloted through the many fluid-filled lumens of the body. This dream is in its infancy. There has been notable progress with pill cameras, tiny cameras that record a passage through the digestive track, from swallowing to expelling.

Shrinking the surgeon to make a capable autonomous robot is hard for two main limitations: power and computation. As the length of the axis \( \ell \) decreases, the surface area decreases as \( \ell^2 \), but the volume at \( \ell^3 \). This relationship is plotted in Fig. 1. Nanocars are perhaps the smallest possible robots, but at 1.4 × 1.7 nm they are smaller than the smallest transistors currently in production (14 nm, beginning in 2014 by Intel). This limited volume effectively prevents onboard computation in nano robots and severely limits computation in microrobots. Power is limited for the same reason because stored power is also a function of volume.

In the 2014 Disney movie *Big Hero 6*, the protagonist Hiro offers a pro-
Figure 1: Scaling laws show that microrobots lose volume faster than surface area and much faster than length as length decreases. This is significant because onboard power and computation is proportional to volume, while forces required for propulsion are proportional to length and surface area.

found view into the future by manufacturing a swarm of $10^5$ microbots. Hiro controls them to self-assemble, to build structures, and to transport goods and materials. While the “microrobots” of the film are fantasy, the ideas are rooted in reality. Today, micro- and nanorobots can be produced in extremely large quantities. Once a manufacturing process is developed, the marginal cost of producing one additional robot is small. Microrobots can be fabricated using microelectromechanical system (MEMS) techniques, e.g. scratch-drive micro robots [16, 17, 18, 19]. These robots are 60 by 250 microns in size, and can be mass-produced with many robots tiled on a single silicon wafer. Perhaps the best examples of large populations are robotic nanocars—synthetic molecules with integrated axles, rolling wheels, and light-driven molecular motors, that are $1.4 \times 1.7$ nm in size [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. These are routinely produced in tremendous quantities—a batch the size of an aspirin tablet contained $\approx 4 \times 10^{19}$ nanocars [21]. This dwarfs the number of automobiles
Figure 2: Uniform control inputs are ubiquitous in micro/nano robotics. Top: differential-drive robots with broadcast control [38], microrobots controlled by shared electrical signal [17], light-driven nanocars [21]. Bottom: r-one swarm under broadcast control [39], photophile kilobot robots [40], and magnetically steered protozoa [41].

produced in the history of the world—90 million automobiles were manufactured in 2014 [31]. Also, biological agents such as bacteria [32, 33, 34, 3, 35] and paramecium [36, 37] can be grown to achieve large swarms.

Ideally, we would design a system that would allow each robot to be controlled individually. However, next-generation micro- and nanorobotic systems have minimal on-board processing and communications bandwidth. The lack of significant on-board computation makes autonomous operation infeasible. Sending individual control signals to each robot requires communications bandwidth that scales with population sizes. Because these systems are only useful when their populations are immense, the bandwidth required for individual unit control is impractical.

Instead, this chapter focuses on systems with uniform control inputs. Some representative systems are shown in Fig. 2: light-driven nanocars are uniformly actuated by a certain wavelength of light, scratch-drive microrobots are uniformly actuated by varying the electric potential across a substrate, and multi-
Figure 3: Example of uniform control inputs. (A) After feeding iron particles to ciliate eukaryon (*Tetrahymena pyriformis*) and magnetizing the particles with a permanent magnet, the cells can be turned by changing the orientation of an external magnetic field (B). Using two orthogonal Helmholz electromagnets, Kim et al. demonstrated steering many living magnetized *T. pyriformis* cells [41, 57]. All cells are steered by the same global field (C). (D) Target applications are in biological vascular networks, such as the vasculature in this cottonwood leaf. Photo: Royce Bair/Flickr/Getty Images.

Robot systems are uniformly controlled by a broadcast radio signal. Other uniform input examples include the magnetic resonant microrobots of Hsi-Wen et al. [42]; the magnetic helical swimming micro- and nanorobots of Ghosh and Fischer [43], Tottori et al. [44], and Schlüre and Nelson et al. [45, 46]; the magnetic microparticles of Diller and Floyd et al. [47, 48]; the magnetic milli-scale capsules of Vartholomeos et al. [50]; the magnetic particles studied by Snezhko et al. and Orduño et al. [51, 52]; and the tumbling magnetic microrobots of [55]. Biological examples include the electric-field controlled paramecium studied by Hashimoto et al. [36] and Hasegawa et al. [37], galvanotactic the electrokinetic and optically controlled bacteria demonstrated by Steager et al. [32], the magnetic-field controlled bacteria demonstrated by Martel et al. [33, 34, 35] and magnetic-field steered protozoa demonstrated by Ou et al. [56]. Figure 3 shows an example of bacteria steered by a global control input provided by an external magnetic field.
2. Breaking symmetry

A swarm of robots controlled by a 2 DoF (degree of freedom) signal is inherently under-actuated since each robot has two to six DoF. To make robots behave differently requires a mechanism to break symmetry from the control input. Breaking symmetry enables the same control input to steer individual robots to different locations.

Our previous work applied ensemble control theory \[58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69\] and control Lyapunov functions \[70\] to steer swarms of robots. Our work, \[39, 41, 57, 71, 72, 73, 74, 75, 76, 77\], applied these techniques to steer swarms of single-celled organisms using an external magnetic field \[41\], because they enabled swarms of robots to be asymptotically driven toward goal states—see overview in Fig. 3. On a larger scale, but using similar control laws, we applied ensemble control to drive a swarm of motors to desired velocities using the uniform magnetic gradient of a clinical MRI scanner \[78\]. All this work has one serious drawback: the complexity of the control law increases quadratically with the number of robots, as shown in Figs. 4 and 8. Because swarms of robots can now number in the millions, progress requires techniques that scale sublinearly (or are constant) with population size. Section 4 discusses techniques that use collisions with obstacles to break the symmetry of the control input. These techniques can often exploit the environment to efficiently reconfigure the swarm. A concluding medical example: our work using an MRI to self-assemble Gauss gun components floating inside a spinal-fluid model relied on obstacles to break symmetry and achieve quasi-independent control of three components \[7\].

3. Breaking Symmetry With Robot Inhomogeneity

Many micro robots, including all those in Fig. 2, have kinematics that match the kinematic unicycle model. This model describes each robot with an \(x, y\) location and heading \(\theta\), with two control inputs; linear velocity \(v(t)\) and angular velocity \(\omega(t)\). \(v(t)\) and \(\omega(t)\) are functions of time, but we assume that the same \(v(t)\) and \(\omega(t)\) are applied to every robot...
With this model, consider a collection of $n$ unicycles that each roll without slipping. Following the terminology of [58, 59, 64], we call this collection an ensemble and describe the configuration of the $i$th robot by $\mathbf{q}_i = [x_i, y_i, \theta_i]^T$ and its configuration space by $\mathcal{Q} = \mathbb{R}^2 \times S^1$. The global control inputs are the forward speed $u \in \mathbb{R}$ and turning rate $\omega \in \mathbb{R}$. We assume that each robot has a nonzero parameter $v_i$ that scales the linear velocity and a unique nonzero parameter $\epsilon_i$ that scales the turning rate ($|\epsilon_i| \neq |\epsilon_j| \forall i, j$). These $v_i, \epsilon_i$ values may arise from stochastic processes during manufacturing [16], or as design decisions [80]. The kinematics of the unicycle are given by

$$\mathbf{q}_i(t) = v_i u(t) \begin{bmatrix} \cos \theta_i \\ \sin \theta_i \\ 0 \end{bmatrix} + \epsilon_i \omega(t) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$  \hfill (1)

If $v_i$ is zero the robot cannot move. Similarly, $\epsilon_i = 0$ prevents the robot from turning. On a collection of differential-drive robots, these parameters can be mapped to unique wheel sizes and $\epsilon_i = v_i$.

We model our robotic system with a discrete-time model. We can simplify (1) by splitting each $\Delta T$ time step into two stages with piecewise constant inputs. During the first stage of round $k$ we command the robots to turn in
place φ, and during the second stage command the linear movement $u(k)$.

$$k = \left\lfloor \frac{t}{\Delta T} \right\rfloor$$

$$\begin{cases} 
\left[ u(t), \omega(t) \right] = \begin{bmatrix} 0, \frac{2}{\Delta T} \phi \\ \frac{2}{\Delta T} u(k), 0 \end{bmatrix} & t - k\Delta T < \frac{\Delta T}{2} \\
\left[ \frac{2}{\Delta T} u(k), 0 \right] & \text{else} 
\end{cases} \quad (2)$$

A control law that steers each robot to their goal alternates between having all robots turn in place and then commanding all the robots to move forwards or backwards. During the first stage of round $k$ we command the robots to turn in place $\phi$, and during the second stage command the linear movement

$$u(k) = -\frac{1}{n} \sum_{i=1}^{n} \left( x_i(k) \cos(\theta_i(k)) + y_i(k) \sin(\theta_i(k)) \right) \quad (3)$$

This controller requires that either all robots turn at slightly different rates or that robots when commanded to turn have stochastic variations. Either condition is possible by naturally occurring or designed parameter variations during robot construction. As long as $\phi$ meets the constraints on the sampling frequency given by the Nyquist frequency, our globally asymptotically stable control results follow. The control policy is easy to implement, never increases the summed distance of the ensemble from the goal, and is robust to standard models of noise.

Figure 5 shows simulations of the first 100 commands to an ensemble of two robots. Below each drawing of the robots at a specified number of turns is a plot of the sum squared distance of each robot to their goal locations. This error function is always quadratic. At each step, the linear velocity control input is chosen that minimizes the error function. This results in a control law that is globally asymptotically stable. Fig. 6 shows the same procedure with six robots. The error function is always quadratic. The fact that the error function is quadratic for arbitrary goal locations and any number of robots was surprising to us, so we made an interactive online demonstration you can use to steer robots to goal locations. Alternately, purchase multiple RC cars that all have the same control frequency and practice steering them to goals. This experiment costs less than $20, and is popular with all ages.
Figure 5: Ensemble control consists of (a) commanding each robot to rotate in place, and then (b) commanding each robot to move a distance $d$ that is selected to minimize the average distance of the swarm to their respective goals. The top row shows the robots, drawn as a circle with a tick mark indicating the forward direction of the robot and the dashed lines achievable positions by moving forward or backward. The bottom row shows the average distance of the swarm to their respective goals as a function of the forward command applied to all the robots. This function is always quadratic for any number of robots. After the turn in step 1a, a commanded distance of +2.1 minimizes the average distance error. In step 2 a forward command of +2.0 minimizes the error. By step 100 the robots are each on their respective goal locations. Simulation is available online at [SI].
Figure 6: Ensemble control with six robots. The average distance of the swarm to their respective goals as a function of the forward command applied to all the robots is always quadratic for any number of robots. Convergence with six robots requires more time than with two robots. Simulation is available online at [81].
4. Breaking Symmetry with Obstacles

Global inputs dictate that every robot receives the same control commands. The previous section exploited the fact that all robots, especially at the micro scale, experience stochastic disturbances and so react differently to inputs. This section instead exploits the fact that robots often move in an environment rich with obstacles. These obstacles exert position-dependent forces on the robots, and can therefore often be used to efficiently manipulate the swarm towards a desired configuration.

**Model:** At micro-scales, viscous forces dominate inertial forces \cite{79}, giving a simple kinematic model

\[ c_i \dot{x}_i = u_x, \quad c_i \dot{y}_i = u_y. \]

Here the control input \([u_x, u_y] \) is globally applied to robots with positions \([x_i, y_i] \) for \(i \in [1, n] \).

4.1. Nonprehensile manipulation

In nonprehensile manipulation, a robot affects its environment without grasping \cite{82, 83, 84}. In some ways, our problem formulation is the inverse of nonprehensile manipulation. Rather than just use a robot to restructure the environment, we use the environment to restructure a population of robots.

We can also use a large population of robots for traditional nonprehensile tasks, such as transporting objects using the flow of the robots \cite{85}, and manipulating an object too heavy for a single robot. Our control formulation enables efficient control of this kind of transport.

4.2. BlockWorld Abstraction

We illustrate our points with a simplified BlockWorld abstraction. The workspace is a rectangular \(m_1 \times m_2 \) grid in which each square is marked either free, fixed, or robot. All robots are controlled by a shared input command from the set \(\{\uparrow, \rightarrow, \leftarrow, \downarrow, \emptyset\} \), and can move horizontally and vertically in the grid, as
Figure 7: A single rectangular obstacle is sufficient to enable position control of \( n \) robots. We provide an \( O(n^2) \) algorithm to accomplish this. Shown above are frames from moving the \( k \)th robot into position. The robots are initially within the box at \( S(t) \), which is of width \( S_w \) and height \( S_h \). We want to move these robots to their final positions within a box at \( F(t) \), which is of width \( F_w \) and height \( F_h \) and disjoint from \( S(t) \). Given a simple square obstacle \( O \), the algorithm requires at least \( S_w + F_w + 1 \) space on the left, \( S_w + F_w \) on the right, \( S_h + F_h + 1 \) above, and \( S_h + F_h \) below the obstacle.

long as there are no fixed squares stopping the robot. The boundary of the grid is composed of fixed squares.

The general case of motion-planning in a world composed of even a single robot and both fixed and moveable squares is in the complexity class PSPACE-complete \([\text{86]}\). Adding an additional robot does not decrease this complexity: given any single-robot problem, we can place a second robot in the boundary of the world and surround it with fixed squares without changing the original problem’s complexity. Still, there are many tractable subproblems.

4.3. Position control

This section presents an algorithm to control the position of \( n \) robots using a single obstacle. We employ the BLOCKWORLD abstraction, where the robots and the obstacle are unit squares. Each call to Algorithm \([\text{1}]\) moves one robot from its starting position to its goal position.
**Notation.** The starting position of the $k$th robot in world coordinates is $k^W(0)$, its desired final position is $k^{W}_{\text{goal}}$, and its position at time $t$ is $k^W(t)$. We define fixed-size, axis-aligned bounding boxes $S$ and $F$ such that $k^W(0) \in S^W(0)$ and $k^{W}_{\text{goal}} \in F^W(0) \ \forall k \in [1,n]$. The bottom left corners of $S$ and $F$ are $[S_x^W(t), S_y^W(t)]$ and $[F_x^W(t), F_y^W(t)]$, and are of width $S_w, F_w$ and height $S_h, F_h$.

Because all robots are identical, without loss of generality the robot indices are arranged in raster-scan order left-to-right, top-to-bottom in $S$ and top-to-bottom, left-to-right in $F$. We note that the position of the $k$th robot may be specified in local reference frame: $k^W(t) = F^W(t) + k^F(t)$. The unmoving obstacle is located at $[O^W_x, O^W_y]$. We assume the obstacle position $O_{x,y}$, the starting positions $S_{x,y}$, and the final positions $F_{x,y}$ are disjoint. Without loss of generality, we will assume that $S$ is to the lower right of the obstacle and $F$ is to the upper left of the obstacle, as illustrated in Fig. 7.

**Procedure.** At the beginning of the $k$th call, the time is $t$, the bounding boxes $S$ and $F$ have been returned to their initial positions on opposite corners of $O$, the first $k-1$ robots have been moved to their proper positions in $F$, the remaining robots are in their original columns in $S$, and $O$ is between $S$ and $F$. The $k$th robot starts in position $[k^W_x(t), k^W_y(t)]$ and should be moved to $[k^W_{\text{goal},x}, k^W_{\text{goal},y}]$.

The algorithm consists in “popping” the $k$th robot out of the $S(t)$ bounding box (steps 1–3), pushing the $k$th robot to the correct $x$ coordinate relative to $F_x(t)$ (steps 4–7), pushing the $k$th robot to the correct $y$ coordinate relative to $F_y(t)$ (steps 8–10), and returning the $S$ and $F$ bounding boxes to their original positions on either side of $O$ (steps 11–12).

The commanded distance to move the $k$th robot from $k^W(0)$ to the final destination $k^{W}_{\text{goal}}$ is bounded by:

$$\text{Commanded distance}(k) \leq 2(2S_h + S_w + F_h + F_w + 2)$$
The total distance commanded for position control of \( n \) robots is the sum:

\[
\text{Commanded distance} = \sum_{k=1}^{n} \text{Commanded distance}(k) \leq 2n(2S_h + S_w + F_h + F_w + 2).
\]

**Algorithm 1 PositionControl\((S,F,O,k)\)**

1. move \( \uparrow \) until \( S^W_y(t) > O^W_y \)
2. move \( \leftarrow \) until \( k^W_x(t) = O^W_x \)
3. move \( \downarrow \) until \( k^W_y(t) > S^W_y(t) + S_h \)
4. move \( \uparrow \) until \( S^W_y(t) > O^W_y \)
5. move \( \leftarrow \) until \( S^W_x(t) < O^W_y - S_w \)
6. move \( \downarrow \) until \( k^W_y(t) = O^W_y \)
7. move \( \rightarrow \) until \( k^W_x(t) = F_{\text{goal},x} + k^F_{\text{goal},x} \)
8. move \( \uparrow \) 1
9. move \( \rightarrow \)
10. move \( \downarrow \) until \( k^W_y(t) = F^W_{\text{goal},y} + k^F_{\text{goal},y} \)
11. move \( \uparrow \) until \( F^W_y(t) > O^W_y \)
12. move \( \leftarrow \) until \( F^W_x(t) < O^W_x - F_w \)

**Analysis.** Algorithm 1 always requires \( 12n \) control switches. The worst-case running time for Algorithm 1 occurs when \( S \) and \( F \) are sparse and/or have large aspect ratios, and the algorithm runs in \( O(n \cdot \max\{S_w, S_h, F_w, F_h\}) \) time.

For more reasonable arrays, when \( S \) and \( F \) are dense with aspect ratios near 1, the running time approaches \( O(n\sqrt{n}) \).

Algorithm 1 requires at least \( S_w + F_w + 1 \) free space to the left, \( S_w + F_w \) to the right, \( S_h + F_h + 1 \) above, and \( S_h + F_h \) below the obstacle:

\[
(2S_h + 2F_h + 1) \times (2S_w + 2F_w + 1).
\]

**Simulation.** Simulation results are shown in Fig. 8 for five arrangements with an increasing number of robots. We compare the total distance moved and
commanded with the *LAP distance*—the shortest distance according to the Linear Assignment Problem using Manhattan distance. Because all robots are interchangeable, the LAP distance reduces to

\[
\text{LAP} = \sum_{k=1}^{n} |k_{W}^{x}(0) - k_{\text{goal},x}^{W}| + |k_{W}^{y}(0) - k_{\text{goal},y}^{W}|.
\]

5. Conclusion

Manufacturing micro robots has a host of challenges, but there is significant progress by roboticists, biologists, and chemists, who are now producing large populations of simple robots [32, 34, 21]. Controlling large populations of robots with limited capabilities is difficult, due to communication and onboard-computation constraints. Rather than focus on a particular design, this chapter reviewed recent work on controlling large numbers of agents using a global input. Section 3 examined global controls given in the local reference frame of each robot, in the form of angular and velocity commands. Section 4 examined a class of controllers that applies controls in a global references frame.

For an in depth analysis of the complexity of motion planning problem with global inputs see [89, 90, 91, 92, 93]. For recent work on swarm manipulation and shape control of a swarm see Shiva and Becker [94, 95, 96].

References


Figure 8: The required number of moves using our algorithm [40], which employs a single square obstacle to rearrange $n$ square-shaped robots, that all respond identically to a global signal. The log-log plot compares Total distance—the sum of the moves made by every robot, with LAP distance—the shortest distance according to the Linear Assignment Problem using Manhattan distance. In each metric moving {↑, →, ←, ↓} one unit counts as one move. These were calculated for five patterns. Dark blue is target position, red is obstacle, light blue is initial configuration. The outline shows the minimum required free space. See hardware implementation and simulation at [87] and code at [88]. This chapter reviews techniques that do not grow polynomially with population size.


URL http://ijr.sagepub.com/content/32/2/218.abstract


[38] A. Becker, C. Onyuksel, T. Bretl, Feedback control of many differential-drive robots with uniform control inputs, in: IEEE/RSJ International Con-


URL http://www.mathworks.com/matlabcentral/fileexchange/38190


URL http://dx.doi.org/10.1119/1.10903


URL https://youtu.be/tCej-9e6-4o