



OPEN ACCESS

EDITED BY

Anthony Tzes,
New York University Abu Dhabi, United
Arab Emirates

REVIEWED BY

Nikolaos Evangeliou,
New York University Abu Dhabi, United
Arab Emirates
Yadan Zeng,
Nanyang Technological University, Singapore

*CORRESPONDENCE

Yuchen Zhao,
✉ yuchen.zhao078@seu.edu.cn

RECEIVED 22 May 2025

ACCEPTED 02 October 2025

PUBLISHED 05 November 2025

CITATION

Zhao Y and Chen Y (2025) A shape control
and object manipulation technique based on
function approximation for robotic surfaces.
Front. Robot. AI 12:1633131.
doi: 10.3389/frobt.2025.1633131

COPYRIGHT

© 2025 Zhao and Chen. This is an
open-access article distributed under the
terms of the [Creative Commons Attribution
License \(CC BY\)](#). The use, distribution or
reproduction in other forums is permitted,
provided the original author(s) and the
copyright owner(s) are credited and that the
original publication in this journal is cited, in
accordance with accepted academic practice.
No use, distribution or reproduction is
permitted which does not comply with
these terms.

A shape control and object manipulation technique based on function approximation for robotic surfaces

Yuchen Zhao ^{1,2*} and Yuxin Chen ^{1,2}

¹School of Automation, Southeast University, Nanjing, Jiangsu, China, ²Ministry of Education Key Laboratory of Measurement and Control of Complex Systems of Engineering, Southeast University, Nanjing, Jiangsu, China

Robotic surfaces consisting of many actuators can change shape to perform tasks, such as object transportation and sorting. Increasing the number of actuators can enhance the robot's capacity, but controlling a large number of actuators is a challenging problem that includes issues such as the increased system-wide refresh time. We propose a novel control method that has constant refresh times, no matter how many actuators are in the robot. Having a distributed nature, the method first approximates target shapes, then broadcasts the approximation coefficients to the actuators and relies on itself to compute the inputs. To confirm the system size-independent scaling, we build a robot surface and measure the refresh time as a function of the number of actuators. We also perform experiments to approximate target shapes, and a good agreement between the experiments and theoretical predictions is achieved. Our method is more efficient because it requires fewer control messages to coordinate robot surfaces with the same accuracy. We also present a modeling strategy for the complex robot–object interaction force based on our control method and derive a feedback controller for object transportation tasks. This feedback controller is further tested by object transportation experiments, and the results demonstrate the validity of the model and the controller.

KEYWORDS

robotic surfaces, object manipulation, cellular robots, distributed robot systems, pin array, refresh time scaling

1 Introduction

Robotic surfaces (Liu et al., 2021; Walker, 2017) typically consist of many actuation modules arranged in an array and can serve as intelligent conveyors (Uriarte et al., 2019; Chen et al., 2024), adaptive structures (Wang et al., 2019; Salerno et al., 2020), molding tools (Tian et al., 2022), treadmills (Smoot et al., 2019), shape displays, or haptic interfaces (Leithinger et al., 2014; Nakagaki et al., 2019). The capability of a robotic surface is related to the number of actuators it has as the robot can perform multiple tasks in parallel with more actuators. Developments in soft robotics also bring new solutions to meet the demand of actuators (Liu et al., 2021; Johnson et al., 2023; Robertson et al., 2019). However, coordinating many actuators is challenging. Generating control commands for them requires a large amount of resources, such as

physical space, equipment, and communication bandwidth (Winck and Book, 2013). A noticeable quantity is the time delay between the first and the last actuator when updating the system to a new shape, which we note as the refresh time τ . A small τ is preferable in real-time tasks. Standard communication methods send control messages to each actuator in a sequential fashion (Follmer et al., 2013; Xue et al., 2024; Siu et al., 2018; Stanley et al., 2016; Leithinger and Ishii, 2010), so τ has an $O(N)$ scaling where N is the number of actuators. To control more actuators, strategies such as using multiple communication channels, sharing one motion controller in a small group of motors, or performing multithreading in the central computer have been employed (Johnson et al., 2023; Follmer et al., 2013; Xue et al., 2024; Siu et al., 2018). By optimizing system architectures, these methods are helpful in reducing refresh time, but the fundamental scaling relation of the refresh time remains unchanged, and the scalability can be improved.

A non-sequential and more scalable approach is to drive each row and column of the actuator array, similar to the matrix drive technique used in LED displays (Chen et al., 2011). This method simultaneously controls all actuators on the same row or column. Early works from Zhu, Winck, and Book et al. show that the scaling of τ can be reduced to $O(\sqrt{N})$. They developed a control loop structure based on singular value decomposition that can drive a hydraulic cylinder robotic array to any shape via $2\sqrt{N}$ row and column controllers (Zhu and Book, 2004; 2006; Winck and Book, 2012; Winck et al., 2012; Winck and Book, 2017). In this method, a “control coupler” valve is needed for each cylinder to couple the row and column control signals (Ferguson et al., 2020). In soft robotics, a compact fluidic logic module is designed to regulate the input row and column pressures for a pneumatic soft linear actuator array (Jadhav et al., 2023). In addition, a robotic surface is fabricated with ionic polymer stripes and controlled using peripheral voltages (Wang et al., 2023). However, these methods also lack scalability because the τ is still dependent on system size. The τ scaling issue can become more severe when the area density of actuators increases.

We propose a novel control method for robotic surfaces that has a system size-independent refresh time. In general, a continuous surface profile is first discretized, and the relevant actuation commands are passed to each actuator. Our method focuses on the second part of shape control, so we simply discretize the surface profile on a square lattice representing the pin array. When approximating the discretized surface profile, it is worth noting that (1) neighboring actuators usually have similar inputs, so there is no need to send the inputs to every one of them; (2) the discretized surface profile may be simply parameterized, such as Gaussian function-like patterns used in object manipulation tasks (Johnson et al., 2023), in which only two center coordinates are important. Therefore, in our method, a central computer broadcasts features of the target shape to individual actuation modules and allows them to calculate their inputs. Our method is illustrated in Figure 1. This approach results in a size-independent scaling $O(1)$ of the refresh time, and the residual error of shape approximation becomes dependent on the target shape and approximation algorithm. We experimentally validate the refresh time scaling on a 4×4 pin array setup and compare it to the sequential control method. In order to achieve any shape, we use function forms with universal approximation properties and employ

the discrete cosine transform (DCT) and the matching pursuit (MP) algorithms (Mallat and Zhang, 1993) to compute shape features. We further characterize shape change capability by displaying six distinct shapes and measuring their height profiles. In addition to the improvement in the refresh time scaling, our control method allows an interpretable modeling procedure for the complex robot–object interaction force F . Based on the force model, we design a feedback controller for object transportation tasks and perform experiments to confirm its validity.

The rest of the article is organized as follows. In Section 2, we present the control method and algorithms to compute the shape features, derive equations for the object manipulation problem, and describe the robot setup. In Section 3, we present our experimental results on the time-delay scaling, quantification of shape-changing capacity, and object manipulation tasks. Section 4 contains a discussion and concluding remarks.

2 Materials and methods

2.1 Control message calculation

Our control method is illustrated in Figure 1. One highlight is that the control messages γ_t are broadcast to all actuation modules at each time step t to eliminate communication time delays between the modules. In our case, an actuation module refers to an independent system that includes an actuator, a sensor, and microcontrollers (see Section 2.3). The actuator control subsystem is responsible for regulating the state of each actuator to a given reference so that all actuators can represent a given surface shape. γ_t is a set of shape feature parameters used by each module to compute the reference input for the actuator control subsystem. To be able to generate any target shape using broadcast control message γ_t , we utilize function forms with universal approximation ability. To be more specific, once the modules receive the same γ_t , they can use a single function form $f(x_n, \gamma_t)$ with universal approximation properties and a local parameter x_n to compute their inputs, making it possible for the robot to approximate any target shape. In practice, the function forms are stored on each module microcontroller board for computing the input, and x_n is an identification value stored on the n th module.

The universal approximation ability guarantees that any target shape can be exactly represented by a set of γ_t . Depending on the function forms, different algorithms exist for computing γ_t . We use two function forms to test the refresh time scaling and shape-changing ability. We first choose the cosine function because we can use off-the-shelf discrete cosine transform (DCT) algorithms to compute the shape features as γ_t . The input f_n to the actuator control subsystem in the n th module is represented as:

$$f_n = a_0 + 2 \sum_{t=1}^{N-1} a_t \cos(k_t(2x_n + 1)), \quad k_t = \frac{\pi t}{2N} \quad (1)$$

where a_t and k_t are the amplitude and the wave vector at time t , and $\gamma_t = (a_t, k_t)$. The shape features a_t and k_t are computed using the DCT algorithm from the Scipy package `fftpack`. To construct an accurate target shape, several γ_t are broadcast to all modules. Upon receiving the messages, each module computes the input f_n

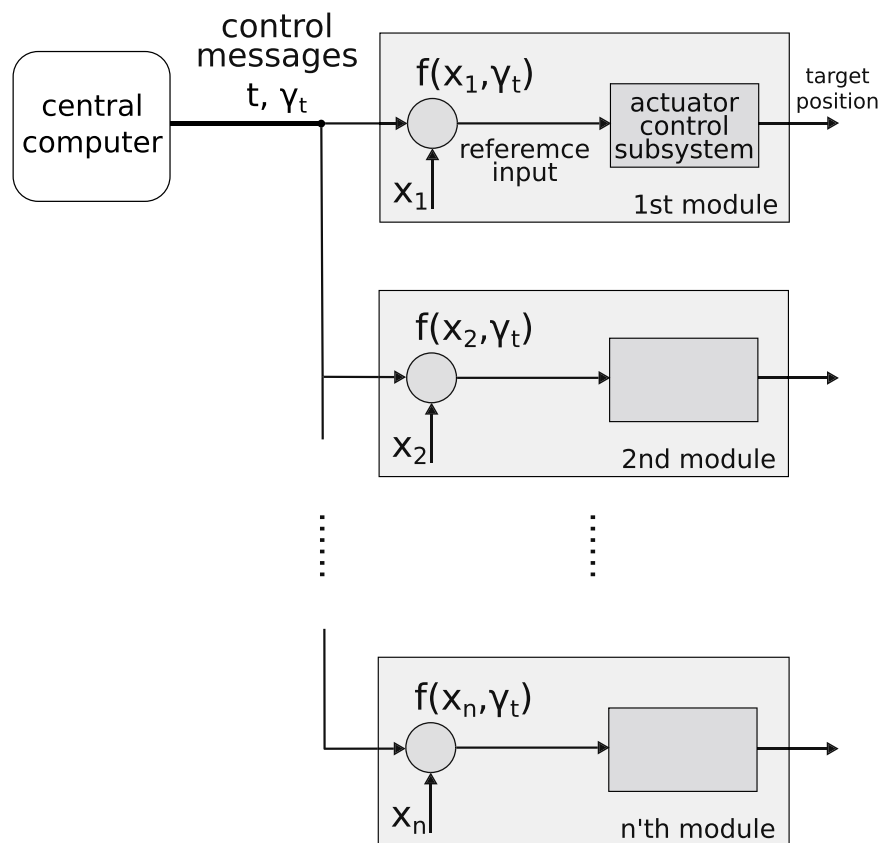


FIGURE 1

An illustration of our control method. At time t , the control message γ_t is broadcast to all modules. In the n th module, the reference input is calculated via a function f and its arguments γ_t and x_n . This input is sent to an actuator subsystem that generates a targeted position.

according to Equation 1 with its own x_n . There is no system size-dependent refresh time because all modules can simultaneously receive γ_t and compute f_n in parallel.

Considering that the cosine function may not be efficient in representing localized patterns, we use time-frequency functions to capture both extended and localized patterns:

$$f_n = \sum_{t=0}^{\infty} a_t g_{\sigma_t}(x_n - p_t) \cos\left(\frac{2\pi k_t}{N} x_n + \phi_t\right) \quad (2)$$

where $g_{\sigma}(x) = \sum_{j=-\infty}^{+\infty} \exp(-\pi(\frac{x-jN}{\sigma})^2)$ is a Gaussian function made periodically on the domain $[0, N]$. The coefficients in $\gamma_t = (a_t, \sigma_t, p_t, k_t, \phi_t)$ are the amplitude, scale, position, frequency, and phase of the time-frequency function, respectively. If σ_t is large, and $g_{\sigma}(x) \approx 1$ over all x in the domain, then Equation 2 reduces to the cosine functions in Equation 1. To compute γ_t for any target shape, we use the matching pursuit algorithm (MP) for a dictionary of time-frequency atoms. The details of the algorithm can be found in Mallat and Zhang (1993). We implement the MP algorithm in Python. It takes a given pattern $\{f_n\}_{n=1}^N$ and computes $\{\gamma_t\}_{t=0}^M$ sequentially to some prescribed M . The reconstructed pattern using finite terms converges to the original one exponentially fast, as proven in Mallat and Zhang (1993). In practice, we find that only a few terms are needed to approximate a pattern with good accuracy (see Section 3.2).

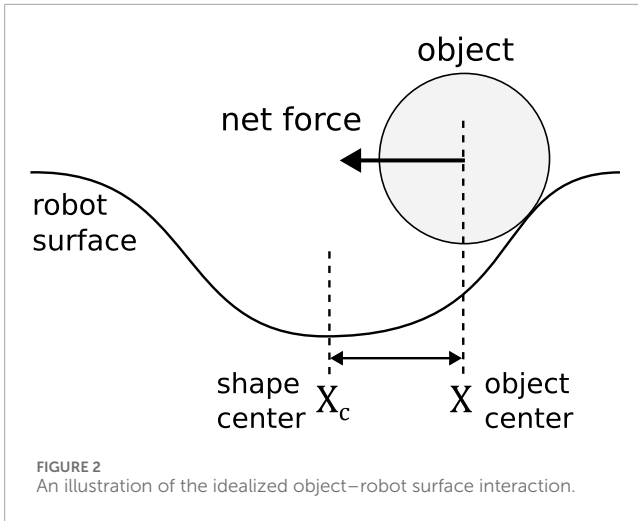
For object manipulation tasks, we use the Gaussian radial basis function (GRBF):

$$f_n = \sum_{t=0}^{\infty} a_t \exp\left(-\frac{(x_n - d_t^{(x)})^2 + (y_n - d_t^{(y)})^2}{\sigma_t^2}\right) \quad (3)$$

where the coefficients in $\gamma_t = (a_t, \sigma_t, d_t^{(x)}, d_t^{(y)})$ are the amplitude, width, and center coordinates of each Gaussian function. (x_n, y_n) are two identification values stored on the n th module, representing the 2D physical coordinates of the module in the array. Although the GRBF is capable of universal approximation (Park and Sandberg, 1991), algorithms computing γ_t may require time-consuming optimization. For this reason, we do not use this function form in shape-changing experiments. Instead, we use the GRBF in object manipulation tasks, and one term in Equation 3 is sufficient.

2.2 Force model and controller design

Generally speaking, an object is governed by the equation of motion of its center of mass $F = m\ddot{X}(t)$, where m is the object mass, and F is the total force acting on the object. F includes the object's gravity, and, importantly, the nonlinear interactions between the object and the robotic surface. Because the surface usually involves



many degrees of freedom, determining a sequence of shapes to manipulate an object is a difficult task. Previous works achieve object manipulations with empirical relations (Johnson et al., 2023; Chen et al., 2021) or black-box machine learning models (Wang et al., 2023; Xue et al., 2024), both of which require data for calibration. Our proposed control method utilizes shape features to generate surfaces, therefore effectively reducing the computational burden of searching for the right shapes. Furthermore, the shape we used allows us to build a reasonable force model to guide feedback controller design for object manipulation tasks; hence, no training data are needed to design the controller.

In this work, we manipulate an object with a shape defined by one concave ($a < 0$) GRBF in Equation 3 and a time-varying center coordinate $X_c(t) := (d^{(x)}, d^{(y)})$. The robot shape control problem is drastically reduced to determining two scalars contained in $X_c(t)$. We first derive the force model. We assume that the surface is smooth, the object is always in contact with the surface, and the net force is purely determined by the static geometric configurations of the shape and the object. The case is illustrated in Figure 2. Depending on the local shape gradient, the object experiences a small net force either near or far away from the shape center $X_c(t)$, and a large net force when $\|X(t) - X_c(t)\|$ is comparable to the width of the Gaussian σ . Hence, the magnitude of F can be expressed as

$$\|F\| = g\left(\frac{\|X(t) - X_c(t)\|}{\sigma}\right), \quad (4)$$

where $g(s)$ is a single-peak function that takes the maximum value at $s = 1$ and remains zero when $s \approx 0$ and $s \gg 1$. Furthermore, because the GRBF has radial symmetry, the direction of F points to the shape center $X_c(t)$:

$$\hat{F} = \frac{X_c(t) - X(t)}{\|X_c(t) - X(t)\|}. \quad (5)$$

Equations 4, 5 together provide a force model when using a single GRBF to generate the shape.

$$F = \frac{X_c(t) - X(t)}{\|X_c(t) - X(t)\|} g\left(\frac{\|X(t) - X_c(t)\|}{\sigma}\right). \quad (6)$$

The force model is obtained via order-of-magnitude analysis and a symmetry consideration; hence, many realistic details are not

captured, such as the discreteness of the robot shape due to finite actuator size, object rotation, friction, and visco-elastic force at contact. We also focus on the planar movement of the object and ignore its vertical motion.

The feedback controller needs to compute the shape center $X_c(t)$ based on the object center $X(t)$ and the target location X_{ref} . Based on the force model in Equation 6, two heuristics can be used to guide the design of the controller. First, we want to maintain the net force direction $\hat{F}$ toward the target position X_{ref} , so the object will not go elsewhere. Moreover, we should keep $X_c(t)$ next to $X(t)$, so the driving force F does not vanish. Mathematically, we first demand that F point toward X_{ref} . This constraint is expressed as

$$X_c(t) - X(t) = \lambda (X_{\text{ref}} - X(t)), \quad (7)$$

where λ determines the magnitude of $X_c(t)$. Based on the assumption that $g(s)$ takes the maximum value at $s = 1$, we can demand

$$\|X_c(t) - X(t)\| = \sigma \quad (8)$$

so that $\|F\|$ attains its greatest value. Equations 7, 8 together determine $X_c(t)$:

$$X_c(t) = (\lambda - 1)(X_{\text{ref}} - X(t)) + X_{\text{ref}} \quad (9)$$

$$\lambda = \frac{\sigma}{\|X_{\text{ref}} - X(t)\|}. \quad (10)$$

The first and second terms (Equation 9) are the proportional feedback and feed-forward control, respectively. The control loop is shown in Figure 3.

To ensure control process stability when $X(t)$ approaches X_{ref} and λ becomes large, we restrict λ to $[0, 1]$, so that $X_c(t) = X_{\text{ref}}$ whenever $\|X_{\text{ref}} - X(t)\| \leq \sigma$. This nonlinear constraint replaces (10) and can be expressed as

$$\lambda = \min\left(\frac{\sigma}{\|X_{\text{ref}} - X(t)\|}, 1\right), \quad (11)$$

The controller only has one parameter σ to tune. In fact, σ should be comparable to the characteristic size of the object, leaving no additional parameter to adjust. Even though the force model is under many assumptions, the controller derived from it can accomplish object transportation tasks, as demonstrated in the experiments (Section 3.3), showing that this procedure is effective. However, as there is no explicit incorporation of disturbance rejection mechanisms, its performance may be compromised if the object-surface interactions largely deviate from ideal assumptions, as shown in Section 3.3. In practice, the control loop functions as follows: first, the position of the object $X(t)$ is captured by a camera, and this information is sent to the central computer to compute the shape center $X_c(t)$ according to Equations 9, 11. The central computer then sends $X_c(t)$ along with other shape features required by the GRBF (Equation 3) to the robot, and the robot displays the shape to manipulate the object. The GRBF is not used in the feedback controller. Details of the hardware implementation are described in Section 2.3.

2.3 Experimental setup

The robot has a modular design and consists of 16 identically built linear actuation modules arranged in an 85-mm-long square area. As

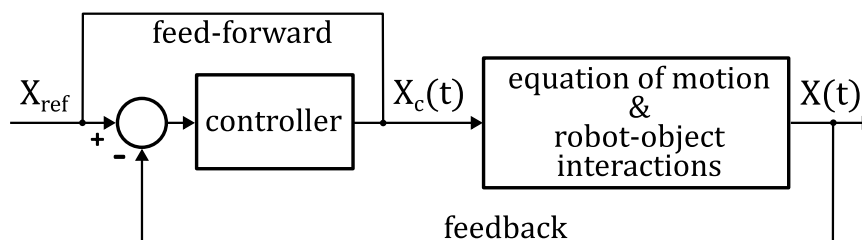


FIGURE 3
A control block diagram for the object manipulation task.

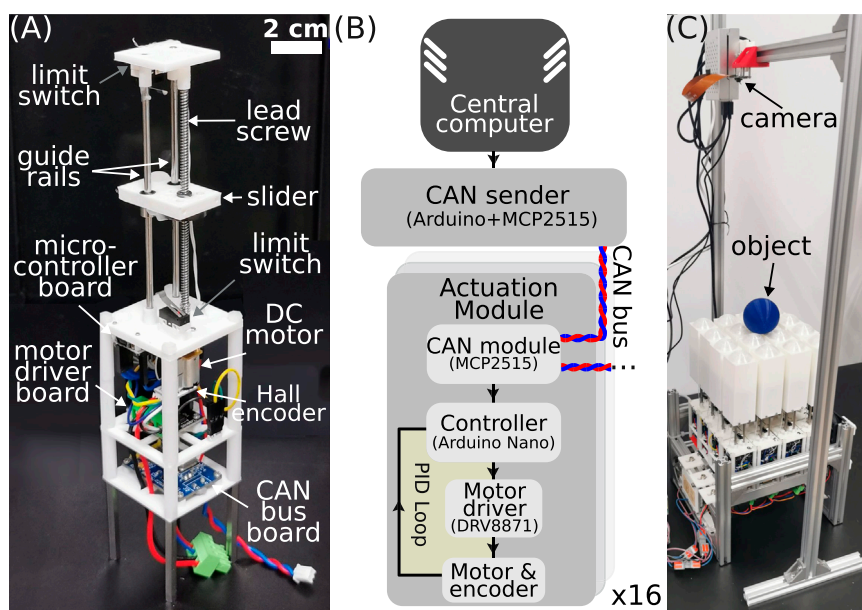


FIGURE 4
(A) A picture of a single linear actuation module. The rectangular cover is removed to expose mechanical components. (B) Block diagram of the electronic system. The arrows indicate information flows. (C) The robot and vision servo system for object manipulation.

shown in Figure 4A, the module is approximately 47 mm wide and 200 mm long. A lead screw of 100 mm in length and 2.5 mm in pitch converts the rotary motion of a DC motor to linear motion. The screw is attached to a slider with two additional guide rails parallel to the lead screw to reduce friction. Two limit switches are installed at the two ends to prevent overshoot that may damage the motor, and the overall arrangement of mechanical parts results in a linear stroke of 70 mm. A complete module also has a rectangular cover attached to the slider (see Figure 4C). The DC motor (Tianqu Motor, N20VA, 1:10) has a rated maximum speed of 50 revolutions per second, leading to a nominal speed of 125 mm/s of the linear motion. The motor's tail has a Hall rotary encoder to measure the angular position of the shaft, defined as e_n for the n th module. The vertical position h_n of the linear actuator is proportional to e_n . All mechanical components and electronics are mounted on 3D printed frames, and the modules are mounted on a portable aluminum frame. When varying the system size N , we simply connect or disconnect modules from the robot.

The electronics block diagram is shown in Figure 4B. The controller of the actuator is an Arduino Nano board, which is

programmed as a closed-loop control system for shaft position e_n and can receive control messages from a central computer. A PID controller running every 16 ms (62.5 Hz) is in use. Two additional modules are connected to the controller board. A DRV8871 board is used to drive the motor with the standard pulse width modulation (PWM) technique. An MCP2515 CAN bus board is used for receiving control messages. All actuation modules are on one CAN bus, and the protocol is CAN 2.0B. While identical in hardware, there is a unique identification variable x_n ranging from 0 to 15 in the n th module's software. This variable is involved in reference input calculation in Equations 1–3, or acts as the CAN message identifier in the sequential control method. A central computer (Raspberry Pi 4B) is used to generate target shapes $\{f_n\}$ and the control messages $\{y_t\}$ to drive the modules. In our method, we broadcast a single y_t in one standard CAN data frame (which holds 8 bytes of data), meaning the coefficients are represented with limited resolution. The data frame also contains information on which function form to use, so we can switch between different approximation methods. An additional Arduino Nano and an MCP2515 module serve as

the CAN bus sender that interfaces with the central computer and the modules.

For the object manipulation tasks, a simple vision servo system is built to track the object and compute the control output $X_c(t)$ to drive the robot. The setup is shown in Figure 4C. A web camera (Raspberry Pi Camera V1) is mounted above the robot and faces downward to capture an image sequence at a rate about 22Hz. Each image contains a colored object and the actuation modules as the background. The object coordinate $X(t)$ is the center of mass of the largest connected component in a binarized image, and we use a color threshold to identify the object. All images are processed with OpenCV in Python. A calibrated 2D affine transform is used to relate the object coordinates in images to those in the lab. Given $X(t)$ and the target position X_{ref} , the new shape center position $X_c(t)$ is computed according to Equations 9, 11. We reduce noise in $X_c(t)$ with a moving average filter of size 15 to reduce actuator jittering. The smoothed $X_c(t)$ is sent to the robot for actuation, and the control loop update frequency is the same as that of the image.

3 Results

3.1 Experiments on refresh time

We first perform an experiment to measure the refresh time where there is no actuator dynamics and only communication delay between the central computer and the actuator modules. The robot is refreshed between two uniform patterns $\{f_n = 0\}_{n=1}^N$ and $\{f_n = 1\}_{n=1}^N$ at a constant rate using either the standard sequential control or our method described in (1). Note that because all actuators have equal f_n values, only one CAN message is needed. To measure f_n on each actuator, we correlate the variable with a PWM output on the controller board and convert the PWM output using a digital-to-analog converter based on an LCR low-pass circuit. The signals of the first and the Nth actuation module are simultaneously measured on an oscilloscope (Rigol DS1202EZ), and the time delay is extracted. We vary N from 2 to 16. Typical signals are shown in Figure 5A and (B) for a system with $N = 2$ actuators. We extract the refresh time using normalized cross-correlation and average over at least 20 values.

The average refresh time is plotted as a function of N for both methods in Figure 5C. As expected, for the sequential control method, the refresh time shows a linear increasing trend, and the slope equals the time period T_{msg} to send one control message. We vary T_{msg} from 5 ms to 20 ms, and the predicted trends agree well with experimental results. In contrast, for our method, the refresh time remains at zero for all N and all T_{msg} , showing that there is no time delay among the modules due to system size or communication delay. Therefore, we confirm that our method indeed achieves a system size-independent refresh time when no actuator dynamics is present.

In the second experiment, we take the actuator dynamics into account by measuring the refresh time between the shaft positions e_n of the first and the last actuator. In addition to the above, the robot is refreshed with a traveling wave pattern, so the refresh time is also determined by the wave speed v . The traveling wave is a quarter of a

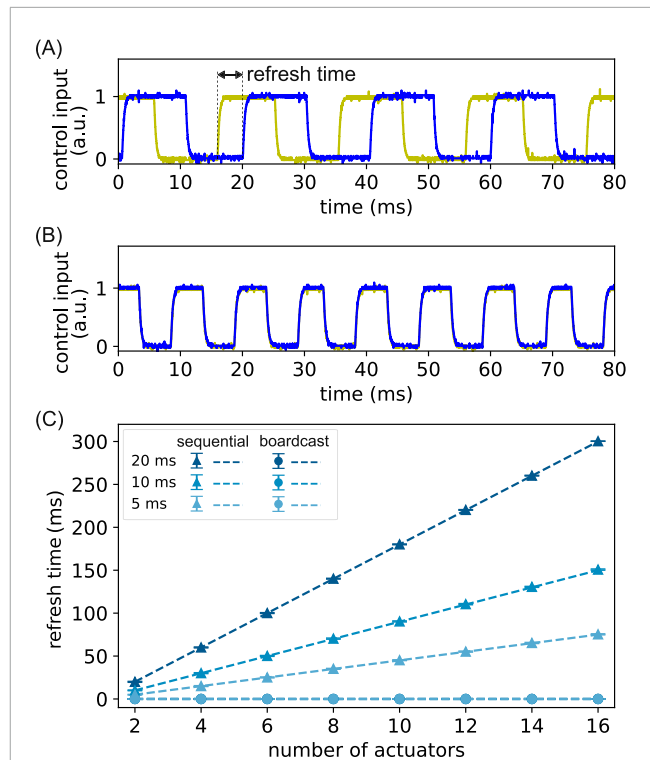


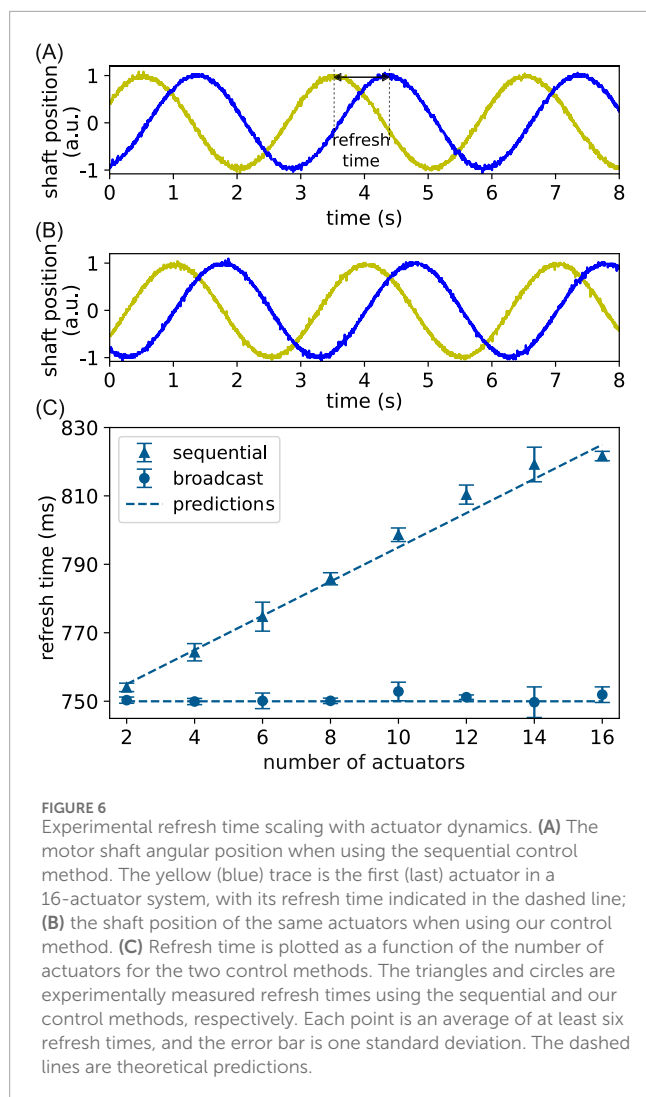
FIGURE 5

Experimental refresh time scaling without actuator dynamics. (A) The reference inputs of two actuators when using a sequential control method. The yellow (blue) line is the first (last) actuator in a two-actuator system; (B) the reference inputs of the same actuators when using our control method; (C) the averaged refresh time is plotted as a function of the number of actuators for the two control methods and at different communication rates (expressed in T_{msg}). Each point is an average of at least 20 refresh time values observed in (A) or (B). The darker color represents larger T_{msg} . The triangles (circles) are experimentally measured refresh times with the sequential (our) control method, and the dashed lines are theoretical predictions.

moving sinusoid over N actuators, given by

$$f_n = \sin(k_N x_n - vt) \quad (12)$$

where $k_N = \frac{\pi}{2(N-1)}$ is the system-size-dependent wave vector chosen such that the last actuator $x_N = N-1$ always has a quarter phase, $k_N x_N = \frac{\pi}{2}$. t is the current time. The control message is sent every $T_{msg} = 5$ ms from the central computer. For the sequential method, f_n is computed and transmitted to each actuator. For our method, we express f_n using a simplified version of Equation 2 because Equation 12 can be rewritten as $f_n = a \cos(k_N x_n - \pi/2) + b \sin(k_N x_n)$, where $a = \cos(vt)$ and $b = -\sin(vt)$. The coefficients k_N , a , and b are broadcast to all modules in one CAN data frame. The observed traveling waves for both methods are shown in Figures 6A,B, and the average refresh time τ as a function of the number of actuators N is shown in Figure 6C. For our method, τ stays constant for all N , hence validating the system size-independent refresh time when both signal transmission and actuator dynamics are present. According to Equation 12, the refresh time between the first and the last actuator should be the time to catch up their phase difference, that is, $\tau = k_N(x_N - x_1)/v = \pi/(2v)$. In all experiments, we set $v = 2\pi/T$ and $T = 3000$ ms, so



$\tau = T/4 = 750$ ms. Our experimental results in Figure 6C agree with this value, with a relative error less than 1% for all N , and the discrepancy is likely due to random errors. For the sequential control method, τ is larger than the theoretical 750 ms and linearly increases as N increases. When one module is added to the system, it takes an additional amount of time to transmit control messages to that module, so the refresh time for the sequential control method is $\tau_{\text{seq}} = T/4 + T_{\text{msg}}(N-1)$. This prediction is plotted as a dashed line in Figure 6C, which also agrees with the experimental results.

3.2 Characterization of shape change

To quantify the approximation accuracy of our control method, we drive the robot to six distinct shapes and quantify the errors between the target and the measured shapes. The shape measurement apparatus is shown in Figure 7A. A laser distance meter (Shanghai Kedi, KG01) is used to measure the height change of each actuator, and the scan process is automated via a homemade Cartesian robot. The system has an accuracy of 0.2 mm. The six

shapes are listed in Table 1. All shapes are represented as a 4×4 matrix, where the elements are actuator vertical positions h_n . The actuation module's planar coordinates (x_n, y_n) are taken from $\{0, 1, 2, 3\}$, and we stretch the vertical scale so that each shape can fill the entire 70-mm stroke of the actuator. The parabola shape displayed by the robot is shown in Figure 7B, and all shapes are illustrated as the insets in Figures 7C–H. For each run of the experiment, we start with the robot leveled at half stroke length. Then, the robot is actuated through a series of intermediate shapes. Following each shape actuation, we perform a height scan to track the shape change and compute the relative error between the intermediate shape and the target shape using root mean squared error. The intermediate shape is achieved incrementally with additional information from one control message y_t that contains all coefficients in one approximation term in Equations 1, 2. For the sequential control method, the term is the reference vertical position h_n of a single actuator. For our method, we test both function approximation formulas in Equations 1, 2. To compute the coefficients, we reshape the 4×4 shape matrix into a vector and apply the DCT or MP algorithm. For the order of shape actuation, we prioritize terms with a higher amplitude or h_n values.

The relative error as a function of the number of terms (or equivalently, the number of control messages) is shown in Figures 7C–H. Our control method with the MP algorithm can outperform the sequential control method in the sense that it requires fewer terms to approximate the target shape for the same error. For example, it takes 11 terms for the sequential method to approximate the parabola shape to a 20% relative error, while it takes six and seven terms for the MP and DCT algorithms, respectively (Figure 7D). We also find that within our methods, MP can approximate both extended and localized patterns with fewer terms, while DCT approximates extended patterns better than localized patterns. The DCT algorithm outperforms the sequential method in three target shapes (plane, parabola, and checker) but can be worse for localized patterns. For example, for the peak shape, it takes only one term for MP to reach a perfect match, while DCT needs all 16 terms. The relative errors obtained from experiments are well captured by theoretical predictions based on Equations 1, 2, except in the parabola, checker, and random cases, where MP leads to residual errors that are not captured by Equation 2. These residual errors are due to the limited coefficient resolution used in computing the approximation terms. To include the five coefficients from Equation 2 into one CAN data frame, s_t and a_t are allocated with 2 bytes each, and p_t , k_t , and ϕ_t are 1 byte each. By recalculating intermediate shapes using these less precise numbers, we can match the experimental results, as shown by the dotted lines in Figures 7D–F.

3.3 Closed-loop object manipulation

We demonstrate the object manipulation capability of the robot based on our closed-loop control method derived in Section 2.2. Motion videos are provided in the Supplementary Material. The objects are 3D printed light-weight spheres of diameter $D_{\text{sphere}} = 6, 8$ cm and cubes of edge length $D_{\text{cube}} = 6, 8$ cm. A spherical cap is attached to the top of each rectangular cover to prevent the object from stabilizing itself on the cover. A single GRBF generates robot

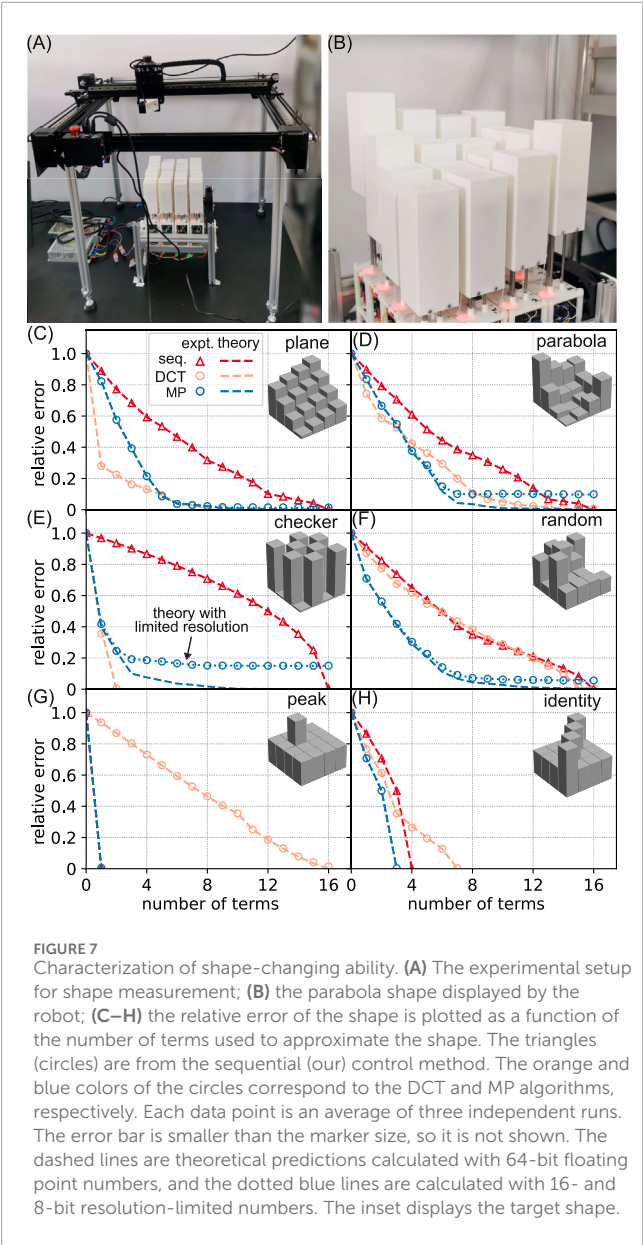


TABLE 1 Names and expressions of the six shapes.

Name	Expression	Name	Expression
Identity	a 4×4 identity matrix	Plane	$z = x + 2y$
Parabola	$z = 2x^2 + 3y^2 - 3xy$	Checkers	Checkerboard pattern
Peak	raise a single module at $x = 1, y = 2$	Random	Random uniform distribution

shape to drive the object, and its amplitude $a = -D$ and width $\sigma = D$, where D is the object's characteristic size and equals D_{sphere} or D_{cube} , so that the robot shape is comparable to the object size. A negative a implies a concave valley for holding the object. Although the object can remain stable inside the valley when the shape is frozen, our controller always changes the shape such that the object never

establishes force equilibrium until the target position is reached. Note that our controller does not assume the shape of the object, and the only tunable parameter in the controller is σ , which we set to D .

The initial and target position of the object are at opposite corners of the robot, and the snapshots of typical transportation processes are shown in Figures 8A,B. The objects can be repeatedly transported to their target positions, showing our closed-loop control method is robust. We collect 10 independent runs for each object, and the trajectories are visualized in Figure 8C and (E) for spheres and cubes, respectively. For the $D_{\text{sphere}} = 6\text{ cm}$ sphere, at the beginning of the transportation, different trajectories are close to each other, as shown by the thin blue lines in Figure 8C. They start to diverge when the object hits the cover of an actuation module, marked by a dashed blue square region. Despite this random disturbance, our closed-loop control method can manipulate the sphere to the target position X_{ref} , shown by the converging trajectories towards X_{ref} marked by a black dot. For the $D_{\text{sphere}} = 8\text{ cm}$ sphere, the trajectories diverge less than the smaller sphere. This trend is likely due to the fact that the object-to-actuator size ratio is greater for the larger sphere; hence, the disturbance due to the finite size effect is smaller, and the force model in Equation 6 is more applicable. We also compare the dynamics of the shape center $X_c(t)$ with the object center $X(t)$, and the relative distance to X_{ref} as a function of time is shown in Figure 8D. Initially, $X(t)$ remains constant, presumably due to the response time of the actuator and the inertia of the object. Then, the $D_{\text{sphere}} = 6\text{ cm}$ sphere undergoes rapid movement with an average speed of about 8.8 cm/s. Finally, the object is stabilized near the target position, and a constant offset of approximately 0.75 cm persists. This offset is likely due to the fact that steady-state errors cannot be eliminated by the proportional feedback controller we designed in Equation 9. The convergence of $X_c(t)$ is similar to $X(t)$, and it converges to X_{ref} around $t = 2\text{ s}$, effectively stops the motion of the shape center. For the $D_{\text{sphere}} = 8\text{ cm}$ sphere, the convergence is faster, and the constant offset is also presented.

Similar analysis can be performed for the cubic objects, and the results are shown in Figures 8E,F. The trajectories of cubes are much more scattered than those of the spheres. This can be understood as cubes are more irregular than spheres and experience more random forces during transportation. In addition, the trajectories of the larger cube diverge less than those of the smaller cube. The convergence of the distance to the target position is also less smooth, and sometimes cubes can go away from the target, as shown by the increasing trend in Figure 8F. The $D_{\text{cube}} = 6\text{ cm}$ cube may not converge to a stable final position, as indicated by the nonzero $\|X_c(t) - X_{\text{ref}}\|$. This may be due to its small size and shape so that the force disturbance is relatively large.

To further test the robustness of the design controller, we change the surface property by removing all spherical caps attached to the pins, so the surface becomes flat. This modification increases the randomness during object manipulation, as the objects can sometimes stabilize themselves on the pins or roll away quickly on the flat surface. In addition, the pins are sharper and stiffer, giving greater force disturbance than the soft spherical cap. Without modifying any control parameter, we perform the same object transportation experiments for two spheres and collect 10 independent runs for each object. The resulting trajectories are

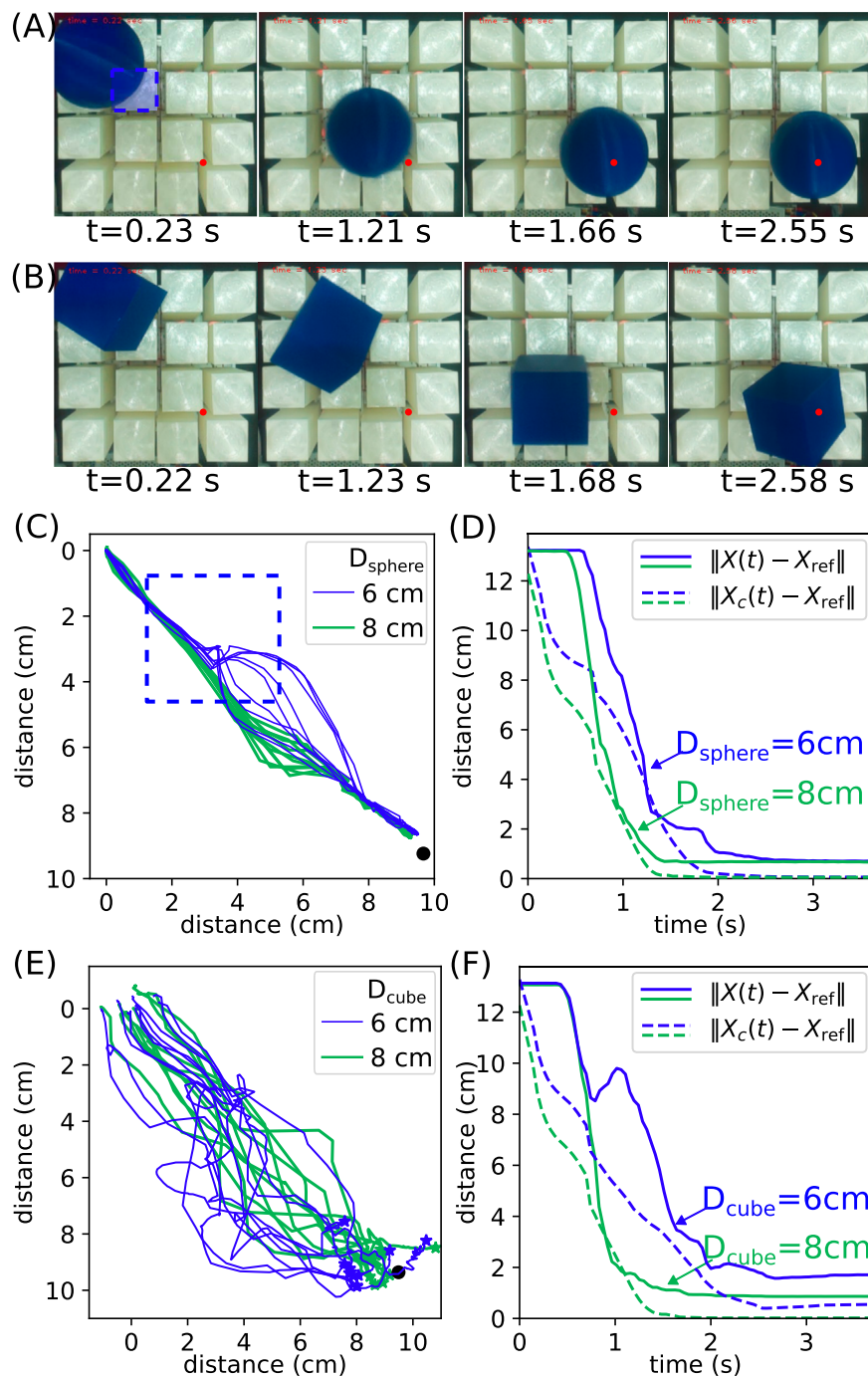


FIGURE 8

Object transportation experiments. (A) Snapshots of transportation processes of a sphere with $D_{\text{sphere}} = 8$ cm. The target position is marked by a red dot in each image; (B) the same plot as (A) except for cubes with $D_{\text{cube}} = 6$ cm. (C) trajectories of 10 independent runs for each sphere. The thin blue (thick green) lines are for spheres with $D_{\text{sphere}} = 6$ (8) cm. The target position is marked by a black dot. The blue dashed square region indicates an actuation module, which corresponds to the blue shaded region in the first image of (A). (D) The distance between the object center $X(t)$ (shape center $X_c(t)$) and the target position X_{ref} is plotted as a function of time in solid (dashed) line. The blue (green) lines are for spheres with $D_{\text{sphere}} = 6$ (8) cm; (E) the same plot as (C) except for cubes. The star markers indicate the end of the trajectories; (F) the same plot as (D) except for cubes.

shown in Figure 9. As expected, the trajectories are more scattered than those in Figure 8. For the $D_{\text{sphere}} = 8$ cm sphere, the all 10 tested trajectories can still converge to the target position (the same), while for the $D_{\text{sphere}} = 6$ cm sphere, four of ten trajectories fail to get close the target. The closest position is where the $D_{\text{sphere}} = 6$ cm sphere

falls onto a single pin next to the target position, and that creates a distance gap in Figure 9 because the two positions do not coincide. This failure is likely due to the net effect of small sphere size and the removal of the spherical cap so that the total disturbance is large, and the controller is incapable of transporting the object. We also

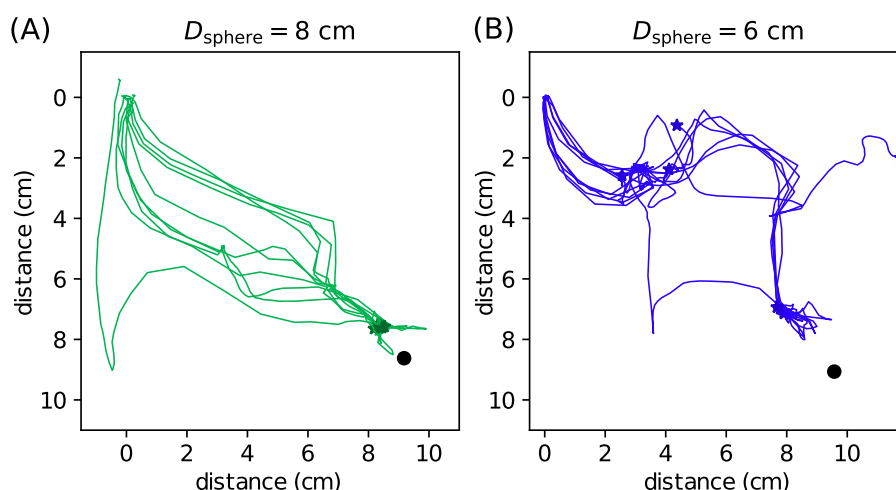


FIGURE 9
Object transportation experiments without the spherical caps. The target position is marked by a circle, and the end position of each trajectory is marked by a star. (A) and (B) are trajectories of 10 independent runs for the $D_{\text{sphere}} = 8$ cm and $D_{\text{sphere}} = 6$ cm spheres, respectively.

note that the flat surface makes it difficult to control the two cubes in the current parameter setting. The cubes can sit perfectly on the flat surface, even if the shape is generated; hence, no experimental result is shown. Based on the comparison experiment, it is clear that the spherical caps should be used in practice. This robustness test on the controller suggests that it is still capable of transporting objects under increased disturbance but needs improvements such as additional sensors or advanced feedback methods to estimate and reduce the disturbance when the object stabilizes itself on the surface before reaching the target position.

It is worth noting that our proposed controller does not explicitly require any constraint on the object shape. The ideal object-surface interaction model in Equation 6 considers the object as a point mass, and the robotic surface being smooth. In this regard, a smooth spherical object would be close to the ideal model. In other words, the controller is more applicable to spherical objects. The effects of shape or surface irregularities are shown in Figure 8 or Figure 9, respectively. Irregularities in object shapes and the robotic surface are both considered disturbances and can lead to unstable trajectories, as the interaction force can have greater fluctuations via unstable contacts, rolling, or slipping, etc.

Finally, to showcase the object manipulation capability of the robot, we have the robot continuously controlling the $D = 8$ cm sphere along a bowtie-shaped trajectory (with the spherical caps installed). The bowtie shape consists of four linear segments, two of which are the diagonals and the other two are the edges of the 4-by-4 pin array, and we let the sphere trace out of the shape 10 times without stopping. The whole trajectory is shown in Figure 10, and except for the beginning of the first time, all trajectories stay close and track out the bowtie shape.

4 Discussion

We present a novel control method for robotic surfaces that can substantially reduce the number of independent inputs. The

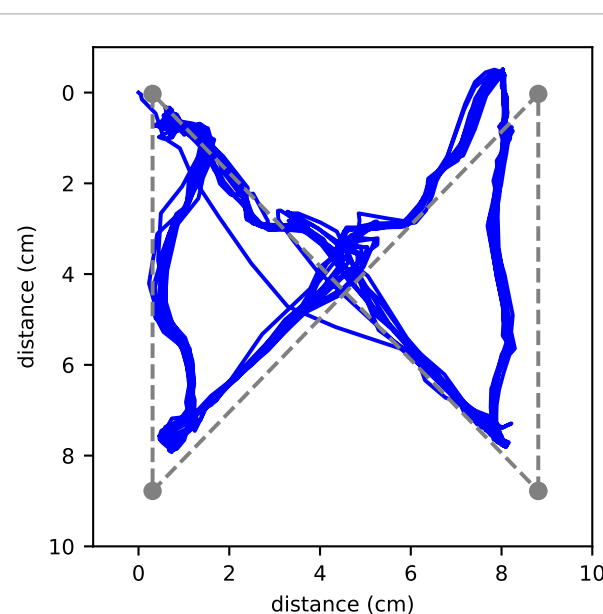


FIGURE 10
The robot continuously controls the $D = 8$ cm sphere along a bowtie-shaped trajectory.

control method has size-independent refresh time and can lead to an effective object manipulation controller when a suitable approximation function is used. We implement the control method in a robotic surface and experimentally confirm the system size-independent refresh time. In addition, the presence of actuator dynamics does not affect this refresh time scaling behavior. Based on the discrete cosine transform and the matching pursuit algorithm, different shapes are efficiently approximated because fewer control messages are required when compared to the standard sequential method. Hence, our control method is more scalable and has the potential to control robotic surfaces with more actuators. Note that

the practical upper limit depends on whether the communication technology can reliably broadcast signals to every module in time. Based on our control method, we also provide a modeling method for object manipulation tasks. Using the GRBF as the shape generator, we simplify the complex interaction force with reasonable assumptions based on order-of-magnitude analysis and symmetry considerations and derive a compact feedback controller for object transportation tasks. The validity of the force model and controller is confirmed by the successful transportation of objects of different sizes and shapes.

Because all modules receive the same control message every time step, our control method can update the state of all modules simultaneously. Alternatively, one may design a schedule procedure that updates only relevant modules at a time and hence saves communication bandwidth even if serial communication is used. In this regard, the advantage of our method is that the schedule step is implicitly performed in computing the control message. For example, if only one module is determined to be actuated, a sufficiently small width parameter (such as σ_i in Equation 2 or Equation 3) will lead to the same control objective, thereby effectively performing the schedule procedure. This is the case in Section 3.2 for the “peak” shape. For cases when the shape of the robot requires changes in a few modules, our method and the schedule-based procedure can perform equally well. However, for complex tasks requiring control over a fraction of the system, our method is more scalable because it uses a system size-independent number of control messages to coordinate the robot. In practice, a hybrid control strategy that can alternate between the scheduled sequential control and a broadcast control method may be suitable to deploy various shapes quickly. Alternatively, one can switch between the DCT and MP algorithms at runtime by setting a threshold value for the fraction of pins that need to have noticeable movements. When the fraction is large, the DCT algorithm is preferable to the MP because it can handle extended patterns. When the fraction is small, MP can be used to actuate a few pins quickly.

As a multi-actuator system, robotic surfaces benefit from a large number of actuators working together to accomplish various tasks, while suffering from the cost and complexity of coordinating many actuators. In essence, our method sends compressed coordination commands to all actuators. A trade-off may exist between the complication due to system size and the complexity of the commands. Although the refresh time scaling is only validated on a small set of actuation modules, and the closed-loop controller is quite simple, we demonstrate its scalable performance. It can be interesting to achieve shape control and object manipulation with distributed control methods, such as designing a sparse state-feedback gain matrix K (Chanfreut et al., 2021; Babazadeh and Nobakhti, 2017), and compare with our method. In our work, each module has its own microcontroller for processing incoming signals, computing function approximations, and performing position servoing. It may also be interesting to design simpler circuits or even mechanical components, such as the fluidic coupler (Jadhav et al., 2023), for the function approximation purposes. Ongoing work focuses on theoretical controllability and closed-loop stability of this control method, as well as more advanced controllers to handle the disturbance when manipulating deformable or irregular geometries.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

YZ: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Project administration, Software, Writing – original draft, Writing – review and editing. YC: Data curation, Software, Writing – review and editing, Formal Analysis.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This work was supported by the start-up research fund of Southeast University.

Acknowledgments

YZ thanks Cheng Zhao, Yifan Wang, Shihua Li, and Xin Xin for helpful discussions.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/frobt.2025.1633131/full#supplementary-material>

References

- Babazadeh, M., and Nobakhti, A. (2017). Sparsity promotion in state feedback controller design. *IEEE Trans. Automatic Control* 62, 4066–4072. doi:10.1109/TAC.2016.2626371
- Chanfreut, P., Maestre, J. M., and Camacho, E. F. (2021). A survey on clustering methods for distributed and networked control systems. *Annu. Rev. Control* 52, 75–90. doi:10.1016/j.arcontrol.2021.08.002
- Chen, J., Cranton, W., and Fihn, M. (2011). *Handbook of visual display technology*. Incorporated: Springer Publishing Company.
- Chen, Z., Deng, Z., Dhupia, J. S., Stommel, M., and Xu, W. (2021). Motion modeling and trajectory tracking control for a soft robotic table. *IEEE/ASME Trans. Mechatronics*, 1–11. doi:10.1109/TMECH.2021.3120436
- Chen, Z., Deng, Z., Dhupia, J. S., Stommel, M., and Xu, W. (2024). Trajectory planning and tracking of multiple objects on a soft robotic table using a hierarchical search on time-varying potential fields. *IEEE Trans. Robotics* 40, 351–363. doi:10.1109/tro.2023.3337291
- Ferguson, K. M., Tong, D., and Winck, R. C. (2020). “Multiplicative valve to control many cylinders,” in 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (Boston, MA, USA: IEEE), 673–678.
- Follmer, S., Leithinger, D., Olwal, A., Hogge, A., and Ishii, H. (2013). “inFORM: dynamic physical affordances and constraints through shape and object actuation,” in *Proceedings of the 26th annual ACM symposium on user interface software and technology* (New York, NY, USA: Association for Computing Machinery), 417–426.
- Jadhav, S., Glick, P. E., Ishida, M., Chan, C., Adibnazari, I., Schulze, J. P., et al. (2023). Scalable fluidic matrix circuits for controlling large arrays of individually addressable actuators. *Adv. Intell. Syst.* 5, 2300011. doi:10.1002/aisy.202300011
- Johnson, B. K., Naris, M., Sundaram, V., Volchko, A., Ly, K., Mitchell, S. K., et al. (2023). A multifunctional soft robotic shape display with high-speed actuation, sensing, and control. *Nat. Commun.* 14, 4516. doi:10.1038/s41467-023-39842-2
- Leithinger, D., and Ishii, H. (2010). “Relief: a scalable actuated shape display,” in *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction* (New York, NY, USA: Association for Computing Machinery), 221–222.
- Leithinger, D., Follmer, S., Olwal, A., and Ishii, H. (2014). “Physical telepresence: shape capture and display for embodied, computer-mediated remote collaboration,” in *Proceedings of the 27th annual ACM symposium on user interface software and technology* (New York, NY, USA: Association for Computing Machinery), 461–470.
- Liu, K., Hacker, F., and Daraio, C. (2021). Robotic surfaces with reversible, spatiotemporal control for shape morphing and object manipulation. *Sci. Robotics* 6, eabf5116. doi:10.1126/scirobotics.abf5116
- Mallat, S., and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.* 41, 3397–3415. doi:10.1109/78.258082
- Nakagaki, K., Fitzgerald, D., Ma, Z. J., Vink, L., Levine, D., and Ishii, H. (2019). “inFORCE: bi-directional ‘force’ shape display for haptic interaction,” in *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction* (New York, NY, USA: Association for Computing Machinery), 615–623.
- Park, J., and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Comput.* 3, 246–257. doi:10.1162/neco.1991.3.2.246
- Robertson, M. A., Murakami, M., Felt, W., and Paik, J. (2019). A compact modular soft surface with reconfigurable shape and stiffness. *IEEE/ASME Trans. Mechatronics* 24, 16–24. doi:10.1109/tmech.2018.2878621
- Salerno, M., Paik, J., and Mintchev, S. (2020). Ori-pixel, a Multi-DoFs origami pixel for modular reconfigurable surfaces. *IEEE Robotics Automation Lett.* 5, 6988–6995. doi:10.1109/Lra.2020.3028054
- Siu, A. F., Gonzalez, E. J., Yuan, S., Ginsberg, J. B., and Follmer, S. (2018). “shapeShift: 2D spatial manipulation and self-actuation of tabletop shape displays for tangible and haptic interaction,” in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA: Association for Computing Machinery), 1–13. doi:10.1145/3173574.3173865
- Smoot, L. S., Niemeyer, G. D., Christensen, D. L., and Bristow, R. (2019). Floor system providing omnidirectional movement of a person walking in a virtual reality environment
- Stanley, A. A., Hata, K., and Okamura, A. M. (2016). “Closed-loop shape control of a haptic jamming deformable surface,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2718–2724. doi:10.1109/icra.2016.7487433
- Tian, Y., Fang, G., Petrulis, J. S., Weightman, A., and Wang, C. C. L. (2022). Soft robotic mannequin: design and algorithm for deformation control. *IEEE/ASME Trans. Mechatronics* 27, 1820–1828. doi:10.1109/tmech.2022.3175759
- Uriarte, C., Asphandiar, A., Thamer, H., Benggolo, A., and Freitag, M. (2019). Control strategies for small-scaled conveyor modules enabling highly flexible material flow systems. *Procedia CIRP* 79, 433–438. doi:10.1016/j.procir.2019.02.117
- Walker, I. D. (2017). “Continuum robot surfaces: smart saddles and seats,” in *Mechatronics and robotics engineering for advanced and intelligent manufacturing* (Cham: Springer), 97–105.
- Wang, Y., Frazzelle, C., Sirohi, R., Li, L., Walker, I. D., and Green, K. E. (2019). “Design and characterization of a novel robotic surface for application to compressed physical environments,” in 2019 International Conference on Robotics and Automation (ICRA), 102–108. doi:10.1109/icra.2019.8794043
- Wang, J., Sotzing, M., Lee, M., and Chortos, A. (2023). Passively addressed robotic morphing surface (parms) based on machine learning. *Sci. Adv.* 9, eadg8019. doi:10.1126/sciadv.adg8019
- Winck, R. C., and Book, W. J. (2012). *A control loop structure based on singular value decomposition for input-coupled systems*. American Society of Mechanical Engineers Digital Collection, 329–336.
- Winck, R. C., and Book, W. J. (2013). Dimension reduction in a feedback loop using the SVD: results on controllability and stability. *Automatica* 49, 3084–3089. doi:10.1016/j.automatica.2013.07.017
- Winck, R. C., and Book, W. J. (2017). Stability and performance of the SVD system. *IEEE Trans. Automatic Control* 62, 6619–6624. doi:10.1109/TAC.2017.2717808
- Winck, R. C., Kim, J., Book, W. J., and Park, H. (2012). Command generation techniques for a pin array using the SVD and the SNMF. *IFAC Proc. Vol.* 45, 411–416. doi:10.3182/20120905-3-hr-2030.00072
- Xue, Z., Zhang, H., Cheng, J., He, Z., Ju, Y., Lin, C., et al. (2024). “ArrayBot: reinforcement learning for generalizable distributed manipulation through touch,” in 2024 IEEE International Conference on Robotics and Automation (ICRA), 16744–16751. doi:10.1109/ICRA57147.2024.10610350
- Zhu, H., and Book, W. J. (2004). *Practical structure design and control for digital clay*. American Society of Mechanical Engineers Digital Collection, 1051–1058.
- Zhu, H., and Book, W. J. (2006). “Construction and control of massive hydraulic miniature-actuator-sensor array,” in 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 820–825. doi:10.1109/cacsd-cca-isc.2006.4776751