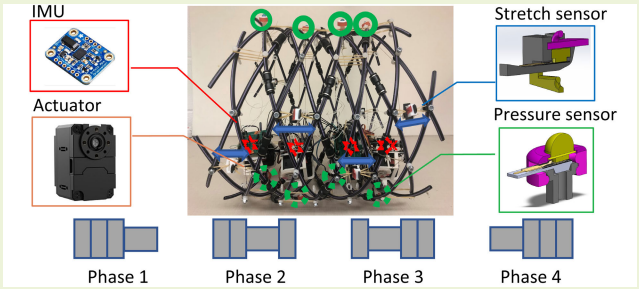
Tracking Displacement of a Worm-Like Robot With Multiple Sensor Configurations

Yifan Wang, *Member, IEEE*, Mingyi Wang, Natasha A. Rouse, *Graduate Student Member, IEEE*, and Kathryn A. Daltorio, *Member, IEEE*

***Abstract*—Worm-like robots that mimic the peristaltic locomotion of earthworms have high robustness to complex environments. These robots' movements are driven by the deformation of the body, but this compliance in the body brings challenges for tracking and control. This work compares three sensing methods to track a worm-like robot's displacement: the actuator forward modeling method (AFMM), the stretch and pressure sensor method (SPSM), and the inertial measurement unit method (IMUM). Each of these methods is compared against a true displacement determined by vision tracking. Based on experimental results, SPSM yields the lowest average error (underestimating the true value by 17% on average), AFMM is slightly higher (20% underestimation), and the IMU result has a comparatively large average error (77% overestimation). AFMM failed to track the robot's backward slip, while both SPSM and IMUM showed the ability of slip detection.**



***Index Terms*— Bio-inspired robots, contact sensing, distributed sensing, position tracking, soft robots.**

I. Introduction

SOFT robots' locomotion can be compliant with the environment [1]. Unlike rigid robots, soft robots change shape to adapt to irregular spaces with different control methods [2]. Soft robots are uniquely suited to navigate narrow spaces, or uneven ground [3]. Path planning algorithms can be applied to soft robots, such as the Gaussian mixture model and the Gaussian mixture regression [4], the Voronoi diagrams and the Dijkstra's algorithm [5], and rapid random tree algorithm with Bezier curve [6], [7]. However, with their indefinite form, forward modeling to estimate these robots' configurations and conditions can become highly unreliable [8]. Direct sensing contributes to improving controllability; it helps feedback adjustment by providing real-time proprioception and perception. In addition, in specific surroundings, the special structure and sensing ability of soft robots create challenges in modeling and control [9]. Soft robot motion includes the nonlinear response of the structure as well as unpredictable factors such as friction and contact force. Robot deformation also interrupts soft robot localization [10], and that, in combination with various environments, challenges the localization of soft robots without effective sensing methods.

For rigid robots, sensing methods for proximity awareness and obstacle detection are well-established with many sensors. Research has been done with depth sensors and camera vision for the localization and 3-D mapping of mobile robots. A sensor array has been used to compile a 2-D obstacle map of a robot's surrounding environment [11], and a depth camera has been used to build a virtual obstacle map with respect to its robot's coordinate changes on that map [12]. These methods add environmental obstacle sensing and mapping abilities for rigid robots but may have difficulties exhibiting soft robots' flexibility advantage in environmental sensing as well.

Soft robots, with a more flexible structure, require more contact and deformation sensing to use their shape-changing features. Many researchers have made progress in sensing the deformation of soft actuators with different types of sensors [13], [14], [15], [16], [17], [18]. A recent study demonstrated the control of a T-shaped soft robot using two rolled dielectric elastomer (DE) actuators [18]. That system estimates the soft robot's position by collecting electrical measurements from the material, estimating its displacement according to

Manuscript received 19 February 2023; accepted 15 May 2023. Date of publication 8 June 2023; date of current version 14 July 2023. This work was supported in part by the NSF Foundational Robotics Research CAREER Award 2047330. The associate editor coordinating the review of this article and approving it for publication was Prof. Reza Malekian. *(Corresponding author: Kathryn A. Daltorio.)*

Yifan Wang, Natasha A. Rouse, and Kathryn A. Daltorio are with the Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland, OH 44106 USA (e-mail: kathryn.daltorio@case.edu).

Mingyi Wang is with the Department of Electrical, Computer and Systems Engineering, Case Western Reserve University, Cleveland, OH 44106 USA.

Digital Object Identifier 10.1109/JSEN.2023.3281747

those measurements, and reconstructing the robot's position from the DE displacement. Giffney's team [13] applied a combination of carbon nanotube strain sensors inside a soft pneumatic actuator to detect the actuator's bending angle by measuring the strain resistance change. The final position could be estimated by the bending angle. Other flexible structures also applied sensors in position sensing. A smart SensorTape is a tape-shaped, flexible sensor network [17]. It includes inertial measurement units (IMUs) at each node; each IMU detects the tape's deformation and measures the motion information used to estimate the tape's motion and 3-D shape. These studies demonstrate the necessity and efficiency of sensor implementation for a soft robot. However, these implementations are limited to single-shape deformation and are majorly focused on sensing the actuator.

In contrast to these examples, attempts to directly obtain position or body shape readings for soft robots are still limited. A study [19] applied multiple 6-degree of freedom (DOF) force sensors to a rigid linked worm-like robot. With the ultraprecise sensors placed directly on the contact points, their experimental results perfectly matched the external vision tracking results. The deformation of a soft robot could be well determined by sensing the robot's contact points with the environment. However, this requires a suite of sensors that effectively perceive this contact force.

Our study aims for more economic and generally applicable solutions for robots that are compliant and can adaptively deform their shape to better match their environment. For our work, it is important to measure not only the actuators but also the robot's contact points and body deformation for precise locomotion tracking.

Previously, our research noticed a displacement difference between the simulation and the physical tests due to backward slip and created a control strategy in the simulation environment for slip reduction during the robot's locomotion [20]. That method searched all possible controls for slip reduction (for all actuators) and returned the optimal slip-reducing motion control. A more detailed study was also conducted to get the numerical displacement with slip along the robot's physical locomotion [21]. In addition, novel research on creating effective locomotion by utilizing each segment's force feedback was also conducted [22]. This research implemented stable heteroclinic channels (SHCs) controller to minimize the slip. This controller has also been combined with a more popular robotic control framework, and the resulting framework offers users visualization of the control system [23]. To implement these algorithms into the physical worm-like robot and further study its control and localization, methods to track real-time slip are essential. In this article, we introduce new sensing methods, including hardware designs and control algorithms to detect a soft, worm-like robot's forward motion and backward slip.

In this work, we use three different sensing methods to determine the robot's position, and designed pressure sensors and stretch sensors that accurately detect the robot's forward and fallback displacement. These sensing methods are: 1) the actuator forward modeling method (AFMM); 2) the stretch and pressure sensor method (SPSM); and 3) the inertial measurement unit method (IMUM). For the first method, four compliant mesh segments of the worm-like robot are controlled by single actuators to expand or contract the diameter of the segment [24]. When the robot locomotes, actuator velocity is recorded and the displacement of the robot is predicted in two dimensions [25]. For the second method, the pressure and stretch sensors' housing is made by an industrial Polyjet 3-D printer, and each unit contains a force-sensing resistor (FSR). The resistance of the FSR changes under load [26], thus it is used to detect vertical pressure at the contact points. This detected force is converted to displacement. For the third method, an IMU is mounted in the center of each segment. IMUs record accelerations in three directions, and thus can be used to estimate the position of their respective segments. The baseline method uses a vision tracking algorithm from the MATLAB Computer Vision Toolbox. The toolbox finds featured pixels on the robot's nodes and tracks the pixels' motions and positions based on the video recording. The vision-based tracking method is widely used in various soft robot planning scenarios for capturing motions [27], [28], [29], [30].

Comparing these three different methods with a "ground truth" from vision tracking, we show that the results of the stretch sensor and pressure sensors most accurately reflect the forward and backward displacement of the robot with a lower error rate, which will help future robots navigate challenging environments.

II. Development of the Robot

A. Platform Structure

We have built a four-segment, wormlike robot. Each segment is built separately with the same design and assembled together. As shown in Fig. 1, each segment is composed of two parts: the outer mesh frame and the actuator mount. The outer mesh frame maintains a round shape via the rhombus structure of the polyethylene tube. The outer mesh frame is compliant and can change shape when the actuator tightens the cables. When the actuator cables are released to their maximum length, the robot frame returns to its default axial shape. The rubber bands along the frame's joints provide a return force and hold the frame at its maximum radius. The actuator mount is connected to the bottom joint of the frame. It holds two actuators on the left side and the right side that control the outer mesh frame's radius. Control is achieved by adjusting the length of actuating cables that pass along the outer mesh frame's joints. Stretch sensors and pressure sensors are also mounted on the frame. All of these sensors are connected to the segment printed circuit board (PCB) on the top of the actuator mount. The IMU is directly mounted on the center of the PCB.

The robot's front and side views are shown in Fig. 1. The actuators tighten or loosen wires to tighten (or relax) the frame in the ZY-plane; this produces locomotion along the X-axis. Rubber bands are tied at the two sides of the platform and robot structure in the Y-axis; this keeps the platform stable during locomotion. The stretch sensors read each segment's extension force in the X-axis and the IMUs sense the platform's acceleration in the X-axis. When the robot's segments deform, the pressure sensors can detect various contact forces in the Z-axis.

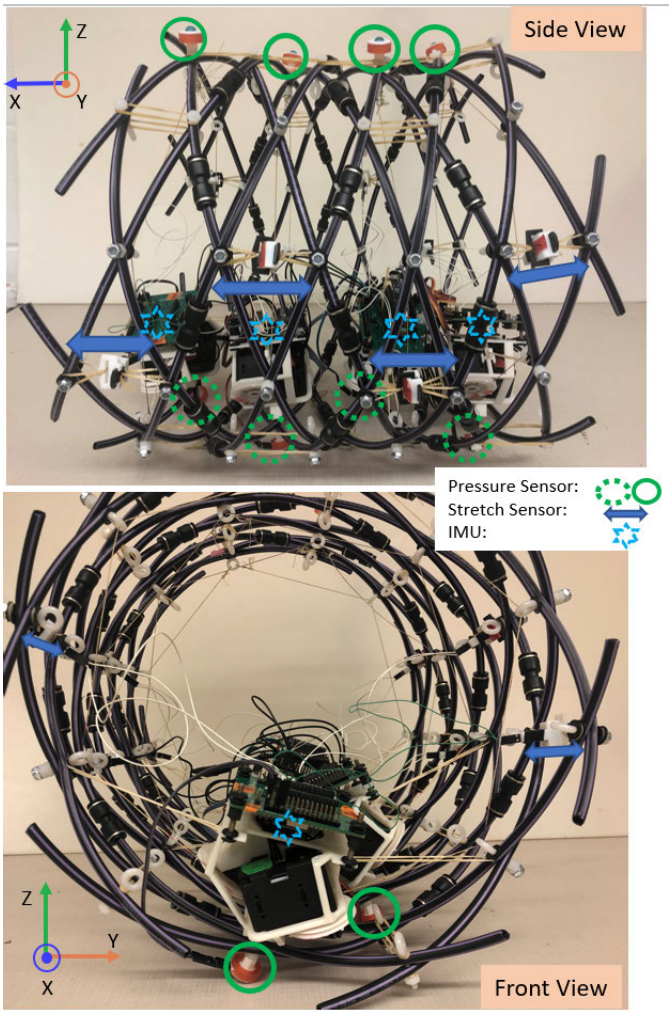


Fig. 1. Robot structure and sensor placement.

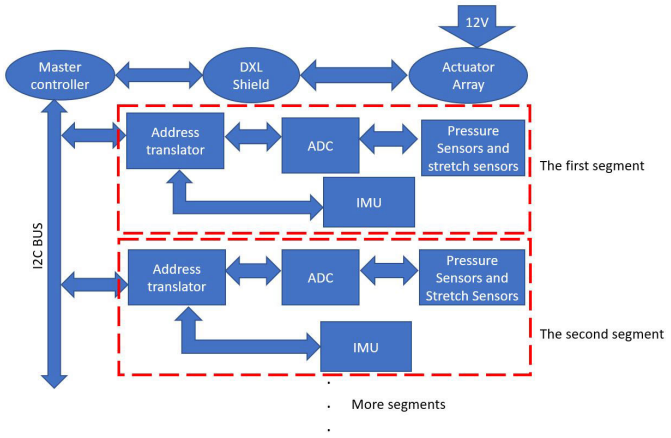


Fig. 2. Primary controller controls the PCB of each segment using the I2C channel. IMUs, pressure sensors, and stretch sensors can be installed on each segment's PCB. Sensor reading will be sent back to the primary controller through the I2C channel.

B. Circuit System

To manage a large number of sensors and actuators, we designed a multilayer system to manage the power and control. The structure of the entire system is shown in Fig. 2. In our robot, each data processing unit (red dashed box in Fig. 2) is integrated into a PCB and is mounted on the top of the actuator mount for each segment.

The PCB regulates power voltage and handles the transmission and conversion of the inter-integrated circuit (I2C) data flow. For each segment, all surface sensor data (from pressure and stretch sensors) are collected by a multichannel ADC (LTC2499). These readings then pass to the higher level controller through the I2C bus. The IMU (BNO055) also outputs its readings as I2C signals. All these sensors are connected in parallel to the PCB's local I2C bus. An address translator (LTC4316) is connected between the local bus and the main I2C bus to shift the I2C address for commands and data. The amount of shift for each address is based on its reference input voltages, which are controlled by a digital potentiometer (MCP4661) on the PCB. We assigned potentiometers to each board with different output voltages. In this way, the same type of sensor chips on each segment PCB can be assigned to different addresses, allowing easy segment addition without limits.

On the center segment's PCB (the third segment for our test), an additional central controller (Arduino MKR 1010) is mounted. It is directly connected to the major I2C bus, sends all reprogrammed commands to each segment, and transfers the merged real-time sensor readings to the computer. An additional voltage regulator is also on the center segment to convert the 12-V input voltage to 5 and 3.3 V working voltages for the entire system.

Actuator data and commands are passed through the DYNAMIXEL Protocol (DXL) bus. The 12-V voltage for the actuators (xl430-w250-t) can also be delivered through it. All actuators are serially connected through the DXL bus and the controller is also connected to it through an expansion board (DYNAMIXEL Shield).

III. Sensing System of the Robot

A. Force Sensitive Resistor

The FSR400 module is used to build our stretch sensors and pressure sensors. When force is applied to the FSR's sensitive surface, its resistance decreases accordingly.

In order to get the best FSR accuracy, each sensor needs to be calibrated separately. A sample relation between the resistance of the FSR and the applied normal force is provided in the manufacturer's manual. With the voltage divider, we obtain the relation between the normal force and the output voltage. Three repeat tests are performed on the same FSR. We gradually increase the load weight to increase the applied normal force. The result is shown in Fig. 3. Based on the test results and the sample data from the manual, we expect a logarithmic relationship

$$\ln F = p_1 \ln R + p_2 \tag{1}$$

p_1 and p_2 are parameters that can be determined by measurements.

According to our tests on different FSR units, we also notice that these parameters vary for different units. But for any given FSR, the parameters will remain consistent and will match the test results, as shown in Fig. 3(c).

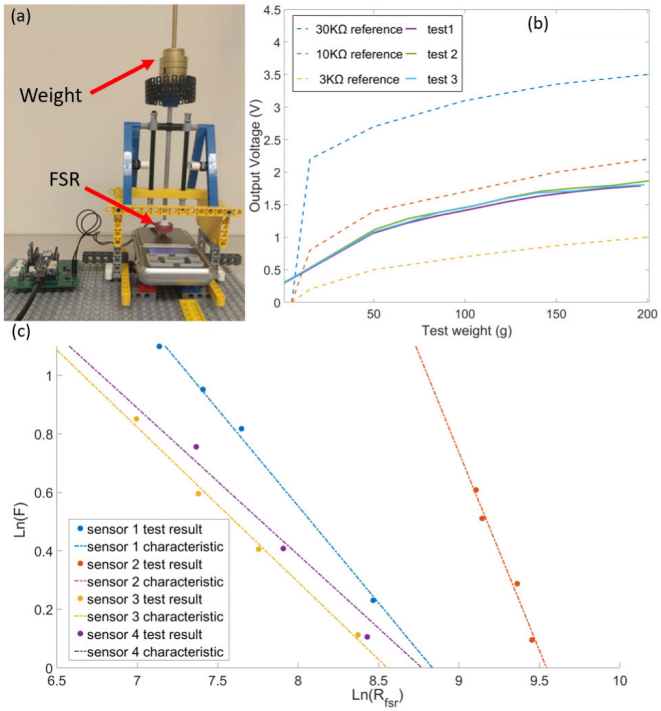


Fig. 3. (a) Platform for the FSR force-resistance relation test. (b) Solid lines are three repeated tests on the FSR with 10-KΩ reference resistor. Dashed lines are the sample curve provided in the FSR manual. (c) Force-resistance test on four FSR units of the same model (FSR400 s). The dots are the test results and the dashed lines are the linear regression between $\ln(F)$ and $\ln(R_{fsr})$.

B. Pressure Sensors

The pressure sensors are placed where the robot contacts the ground, at the bottom vertex of the robot frame of each segment (red circle in Fig. 4). When a segment contracts in length and expands along its radius, it touches the ground and supports the whole robot as an anchor while other segments move. The pressure sensor detects the normal force from the ground to indicate if and how well the segment is anchoring. Furthermore, such information is later used in combination with stretch sensor readings to help indicate slipping.

As shown in Fig. 4(c), the pressure sensor design is an assembly of four parts: base, case, plate, and contact node. Fig. 4(b) also shows the hardness assignment for each part. The plate and base are made of fully rigid material. The case is made of a material with 15% softness to provide essential deformation during assembly as well as to maintain the whole structure's shape. The bottom of the contact node is fully rigid to transfer force to the sensor. The upper sphere is 30% soft to reduce the friction and force of impact from the ground.

Pressure sensors are also installed on the vertices of the outer frame of the wormlike robot. During the robot's locomotion, The normal force from any contacted environment will be applied to the contact node and then transferred to the FSR through the node's bottom surface.

C. Stretch Sensors

Stretch sensors are designed to estimate the distance between two vertices. Elastic components (rubber bands) connect both sides of the sensors to the mounted vertices. The

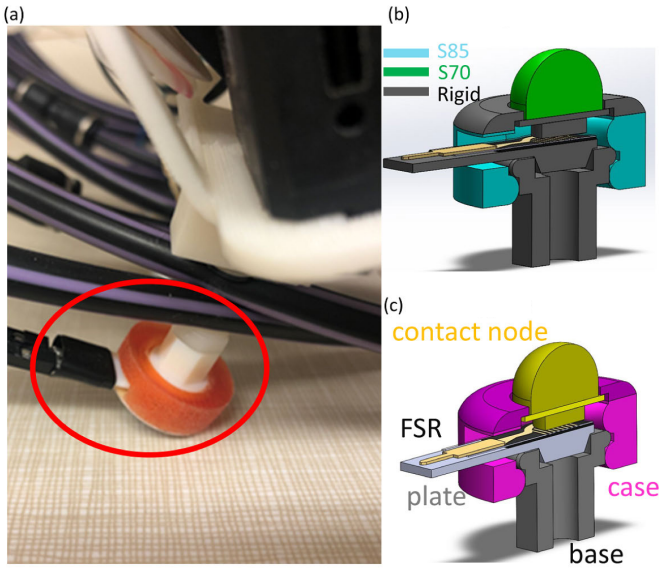


Fig. 4. (a) Installation of the pressure sensor at the bottom of the segment. (b) Assignment of the pressure sensor material. (c) Assembly of parts of the pressure sensor component.

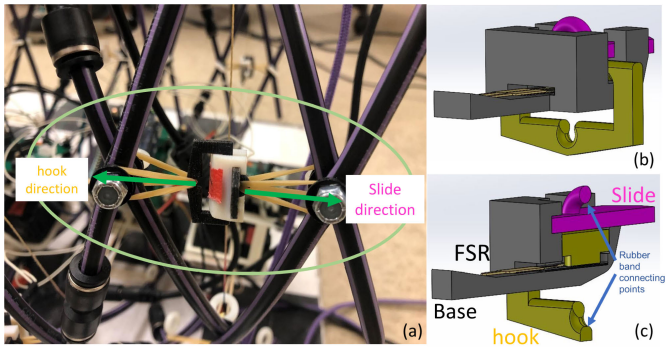


Fig. 5. (a) Installation of the stretch sensor on the side of the segment. (b) Stretch sensor assembly. (c) Parts of the stretch sensor components.

tension of the elastic components is then passed to the FSR inside its case. This tension is the only force source in the entire stretch sensor structure when gravity is ignored. The stiffness coefficient of the elastic components was measured to be 19.9 N/m. Once force is applied to a stretch sensor, we can calculate the deformed length of the elastic components and obtain the real-time distance between the two vertices.

To fulfill this function, the structure of the stretch sensors is designed to be completely rigid except for the elastic components, so the deformation can be viewed only from the elastic components. The pressure sensor contains three parts: the slide, the base, and the top hook, as shown in Fig. 5. The FSR is inserted at the bottom of the base, and the upper column of the top hook contacts its sensitive area. The slide is inserted into the base and remains fixed after assembly. The two elastic components are connected to the bottom of the top hook on one side and the hole of the slide on the other side. When tension is applied to the elastic components, it will be passed to the FSR through the structure.

D. Inertial Measurement Unit

The Adafruit BNO055 is chosen as the IMU unit in each segment. It is a 9-axis absolute orientation sensor and contains

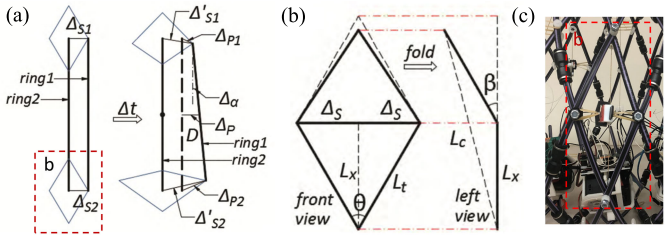


Fig. 6. (a) When $\Delta S1$ and $\Delta S2$ are known for each timestamp, we can update the shape of the segment and obtain the changed position and direction of the segment. (b) Robot frame geometry relationship: L_t is a constant length. When ΔS decreases, L_c expands. The relationship is defined by (5). (c) Geometry relationship is highlighted on the physical robot.

a magnetometer, an accelerometer, and a gyroscope. Based on the accelerometer, this IMU can read X-, Y-, and Z-axes accelerations. It calibrates all reading data by a sensor fusion algorithm. A gravity falling test was used to verify the IMU's acceleration reading on one axis. We dropped the IMU in its X-axis direction and verified that the acceleration in this direction was between 9 and 10 m/s^2. The sampling rate of the IMU in the gravity falling test is 20 Hz.

IV. Position Tracking Methods

A. Robot Kinematics and Segment Position Modeling

Assumption: The robot is a rigid body, i.e., the tube segments rotate, but do not bend.

The robot frame is cylindrical and its surface is composed of a rhombus structure. In our previous research, a simulation was built in MATLAB code and an algorithm was provided to update the corresponding node positions of the robot frame [25]. The state of the robot at any timestep could be completely represented by a matrix of these nodes. Here, we follow the same algorithm, but replace the simulated actuation and robot geometry with different sensor readings (actuator velocity, pressure, and stretch sensor outputs) and the robot's physically measured configuration. This algorithm is later applied to actuator velocity-based, and FSR-based tracking to update the segment position for each sampled time frame.

The robot's geometry connects the left and right lengths of each segment to the corresponding nodes (the vertices along that segment). According to the algorithm, with the given length of each controlling cable (L_c), we can get the right and left lengths of each segment, W_r (Δ_{S1}) and W_l (Δ_{S2}), and we can calculate the robot's node positions at each timestamp to find the segment positions and directions as shown in Fig. 6(a) [25]

$$\begin{aligned} \Delta_{P1,P2} &= \Delta'_{S1,S2} - \Delta_{S1,S2} \\ \Delta P &= (\Delta_{P1} + \Delta_{P2})/2 \\ \Delta\alpha &= \arctan\frac{\Delta_{P1} - \Delta_{P2}}{D}. \end{aligned} \tag{2}$$

With ΔP and $\Delta\alpha$, we can then update the center position and direction angle of the moved ring based on its adjacent fixed ring

$$\begin{aligned} \alpha^{+\prime} &= \alpha^{+} + \Delta\alpha \\ \begin{bmatrix} X_C^{+\prime} \\ Y_C^{+\prime} \\ Z_C^{+\prime} \end{bmatrix} &= \begin{bmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) & 0 \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} X_C^{+} \\ Y_C^{+} \\ Z_C^{+} \end{bmatrix} - \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} \right) \\ &\quad + \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} + \mu\Delta P \begin{bmatrix} \cos\alpha^{-} \\ \sin\alpha^{-} \\ 0 \end{bmatrix} \end{aligned} \tag{3}$$

where μ is the friction coefficient of the ground.

B. Actuator Forward Modeling Method

Assumption: The robot is a rigid body, i.e., its frame (rhombus edges in Fig. 6) is free to rotate, but does not bend or change lengths. Also if a slip occurs, the slip will be on the side of the robot with fewer contracted segments from the moving segments.

Each segment is controlled by two actuators. The actuators rotate according to a preprogrammed command, which sets the actuating cable to release or contract correspondingly. The control cable changes the left and right lengths of the segment (W_l, W_r). Tracking the status of the actuators allows us to estimate the configuration of each segment during locomotion.

The angular velocity of the actuator, ω, can be used to calculate the angular position of the actuator $\varphi = (\pi/30)\int_0^t \omega \mathrm{dt}$. The cross section of our worm-like robot is a hexagon ($N_r = 6$) and the robot's spool radius R_S (15 mm) and tube length L_t (91 mm) were determined when it was built. The cable length per rhombus L_c at the position angle φ can be calculated as below [25]

$$L_c = 2L_t \cos\frac{\theta_L}{2}\cos\frac{\pi}{2N_r} - \frac{2R_s}{N_r}\varphi. \tag{4}$$

When L_c reaches its maximum length, the rhombus angle θ is θ_L. Once L_c is found, we calculate the interval of adjacent segments, Δ_S, based on the relationship shown in Fig. 6(b)

$$\begin{aligned} \Delta_S &= \sqrt{L_t^2 - \left(L_c/2\cos\frac{\pi}{2N_r}\right)^2} \\ W &= 3\Delta_S. \end{aligned} \tag{5}$$

We then use a MATLAB algorithm to estimate all center positions of the segment during robot locomotion, as shown in Fig. 18 in the Appendix.

After integration and conversion using (4) and (5), we get the segment lengths (W_l, W_r). By comparing whether the current segment radius is at its maximum value for each segment, we can determine whether the corresponding segment is expanding or contracting.

For each timestamp, based on the changed control cable lengths (L_c), we update all the segments' positions in the sequence. If a slip happens, for each segment, we count the total number of contracted segments before and after it. When the total number of contracted segments after this segment is more than or equal to the number of contracted segments before it, we consider the segment as moving forward. All positional updates are done on the front edge of this segment;

the rear edge is held in place. If more contracted segments are before the updating segment than after it, then, the update will be on the rear edge, and the front edge will remain in its position.

After applying this process to all segments, we can obtain the updated positions of all segment centers for the next sensor sampling timestamp. Looping through this process for all time frames with the corresponding sensor readings, we obtain a time series for all the segments' center positions.

C. Stretch and Pressure Sensors Method

Assumption: The friction coefficient remains constant, and the difference between dynamic and static friction is negligible.

In this method, we use readings from stretch sensors and pressure sensors to obtain real-time conditions for segment lengths, $W_l^{(i)}$, $W_r^{(i)}$, and the friction force from the ground, $f^{(i)}$.

For stretch sensors, the total length between its two end joints is linear to the force applied to the FSR

$$W = L_{\text{init}} + \frac{K}{2}F \tag{6}$$

where L_{init} is the total length when no force is applied on the sensor, K is the spring rate of the connecting spring of each side, and F is the measured force.

For a pressure sensor, the friction is proportional to the measured force

$$f = \mu N \tag{7}$$

where μ is the friction coefficient between the robot contact joint and the ground. Due to the aforementioned assumption, all comparisons between f are equivalent to the comparison between N.

As shown in Fig. 19 in the Appendix, we apply sensor readings together with all previous configurations of the segment edges (position and heading) to obtain the displacement of the current time frame. By evaluating $W_l^{(i)}$ and $W_r^{(i)}$, we can obtain the changed shape of each segment i.

In the horizontal plane, all forces applied to each segment are from the friction force of other segments

$$F^{(i)} = \sum_{i+1}^{n} f - \sum_{1}^{i-1} f. \tag{8}$$

Therefore, we can determine the direction of slipping as the same as the direction of F

$$Dir^{(i)} = \text{sign}(F^{(i)}). \tag{9}$$

When $Dir^{(i)} \geq 0$, the segment is pushing forward, therefore, its rear joint remains still while all deformation from $W_l^{(i)}$, $W_r^{(i)}$ occurs on its front. For the case $Dir^{(i)} < 0$, the segment is pushing backward. Its front joint remains fixed and its rear deforms.

Each iteration starts from the head ($i = 1$). For the first case ($Dir^{(i)} \geq 0$), all previous segments ($1{:}i-1$) need to undergo a position translation to match the newly added deformation of the segment i.

Using the above method, we get the updated joint and center positions of all segments. We can then update the time frame to the next set of sensor readings. Similar to the AFMM, we can loop through this process for all time frames with the corresponding sensor readings to obtain a time series for all the segments' center positions.

D. IMU Method

Assumption: There is no significant rotation of the entire robot body, i.e., the X-axis of the robot does not move/rotate.

An IMU is mounted in the center of each segment. We collect real-time acceleration of the axial direction of the segment to indicate the displacement of the segment. The algorithm is discussed below and outlined in Fig. 17 in the Appendix. During robot movement, a given segment's actuator velocities are used to determine whether that segment is moving or fixed. If the sum of absolute actuator velocities for the segment is greater than zero, this segment is in the moving stage, and its IMU's axial acceleration reading, $A_x^{(i)}$, is used to calculate the displacement. Otherwise, the segment is in a fixed stage. For the moving stage and the fixed stage, different calibrations are applied to cancel the accumulative errors for accelerations and velocities.

For the fixed stage, the segment should not move, which means that the average acceleration during this stage is zero. We apply linear regression to the original acceleration reading from IMU and consider it as the accumulated error for this stage. Subtracting this error from the original acceleration, we get the calibrated acceleration. The calibrated acceleration is then integrated over time to obtain the velocity of the segment during this fixed stage. Since there is no movement, the average velocity will also remain zero. Therefore, we apply linear regression to the velocity as the cumulative error of the velocity and subtract it from the original velocity for calibration. Finally, by integrating the calibrated velocity, we obtain the displacement of the segment during this fixed stage.

For the moving stage, the actual acceleration changes during the locomotion of the segment. There is also vibration in the segment, which further complicates the acceleration results. However, at the start and end time of the moving stage, both acceleration and velocity will still remain zero. Therefore, we apply linear regression to the accelerations of these two times to calibrate the acceleration. Integrating the calibrated acceleration, we get the velocity during this stage. We then apply linear regression to the velocities of the start and end time of this moving stage as the cumulative error of the velocity. Subtracting the error from the velocities of the whole moving stage, we get the calibrated velocity. We then apply integration to the calibrated velocity over time and obtain the displacement of the segment during this moving stage.

At the end of each stage, both acceleration and velocity are zero. Therefore, we can directly sum up these integrated displacements and get the final displacement of the entire locomotion.

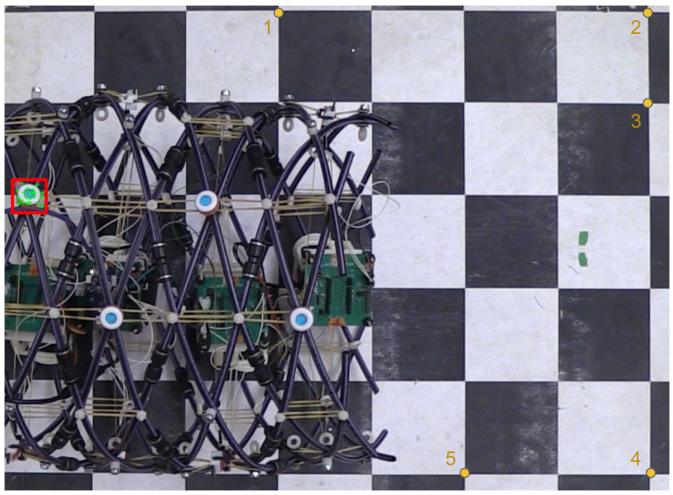


Fig. 7. Camera captured image from the top of the robot, five checkerboard points (yellow), the specified square region (red), and the vision tracking target spots (green crosses).

E. Vision Tracking

During our tests, we placed an overhead camera about 1.5 m above the testing ground to record the wormlike robot's whole real-time locomotion in top view, as shown in Fig. 7. A vision-tracking algorithm was applied to the recorded video to get the robot's real-time reference positions. These reference positions are used later in this article as the ground truth positions when comparing the measured errors of different tracking methods. Note that such overhead recording is not a part of the robot system, and is normally considered impractical to implement in applications outside of a laboratory environment. Therefore this method is only used as a test reference, not an implementable approach like the other methods introduced earlier in this article.

The vision tracking method uses a top-view video recording and the algorithm from the MATLAB Computer Vision Tracking Toolbox to track an observed point from the video. The algorithm code reads the first frame from the video as an object frame, and we manually specify a square area of the object frame (the red square shown in Fig. 7) for the algorithm. The vision tracking algorithm will then focus on a significant visible object within that range (a pressure sensor holder of the robot's fourth segment) and tracks it in later frames. This is achieved by the minimum eigenvalue algorithm that captures all feature pixel points within the region and tracks them in video using the Kanade–Lucas–Tomasi algorithm. Finally, the vision tracking algorithm can show captured points' displacement in each frame from the video (green crosses within the red rectangle shown in Fig. 7).

The vision tracking algorithm records all captured points' coordinates in pixels and converts one of the coordinates of the well-captured points into millimeters. The conversion rate is calculated based on the checkerboard background. The checkerboard is printed with a 100-mm side length for each square block. In the experimental test video, we track checkerboard block corner positions 1 through 4, shown as yellow circles in Fig. 7. The conversion rate is calculated as the four points' pixel distances divided by their physical measured distance in mm, which is 1.23 (pixel/mm) for Fig. 7. Additional verification is then applied using the corner position of point 5. Its in-video distance to corner position 4 (243.46 pixels) is converted to physical distance (197.93 mm). The true physical distance is 200 mm; this yields an error of 1.0%.

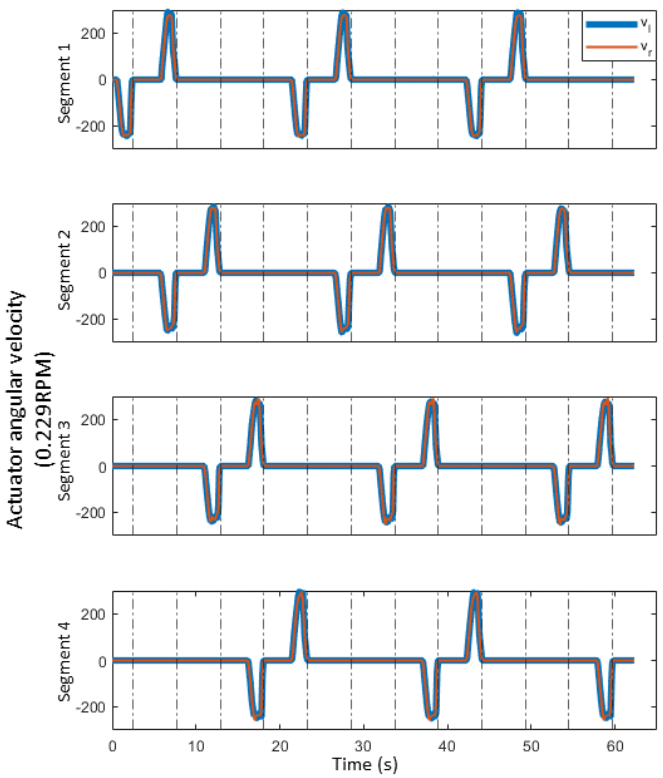


Fig. 8. Actuator angular velocities during locomotion.

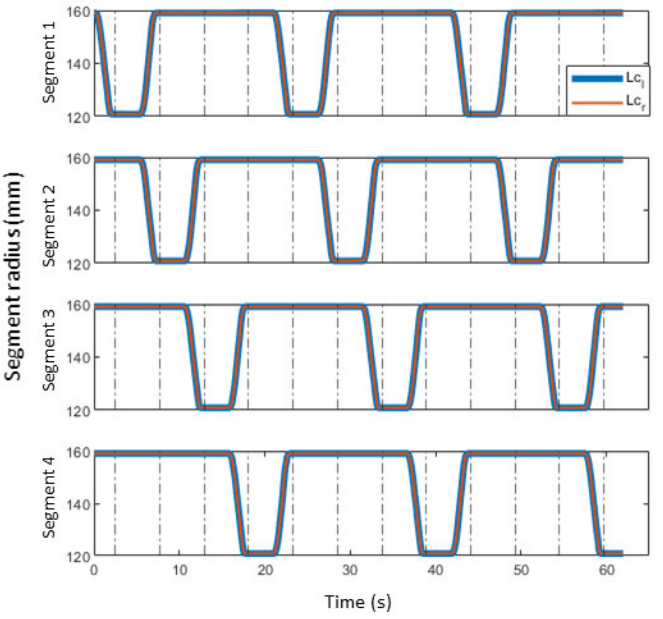


Fig. 9. Segment radii of both sides determined from actuator velocities during locomotion.

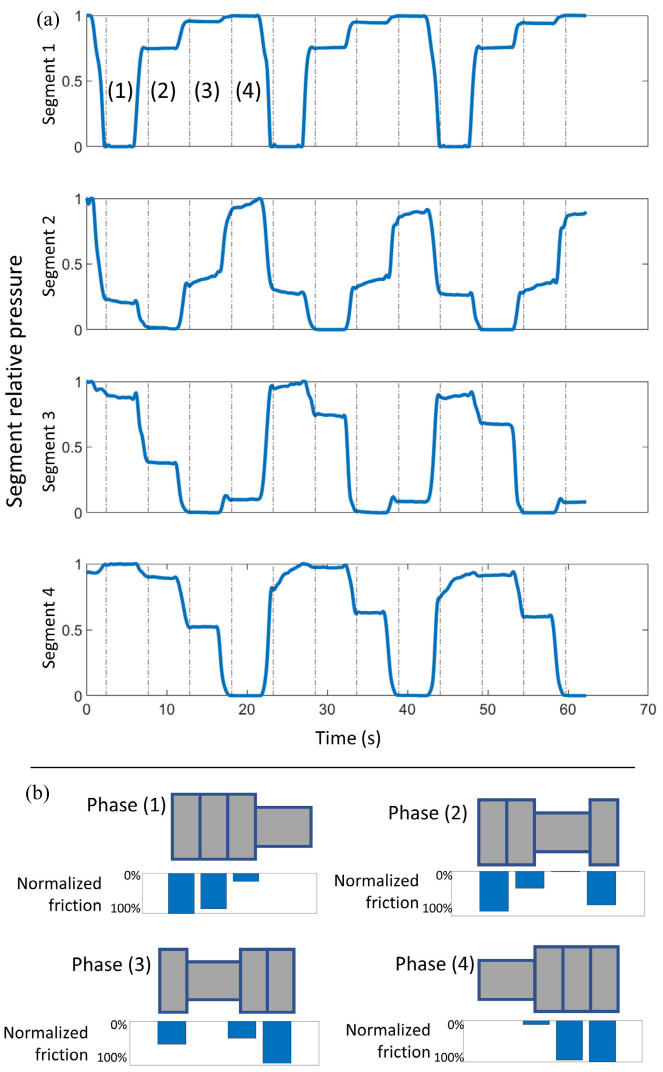


Fig. 10. (a) Relative pressure changes for each segment along the robot's locomotion based on the pressure sensor reading. (b) Normalized friction distribution is applied on each segment in each phase of the robot's movement.

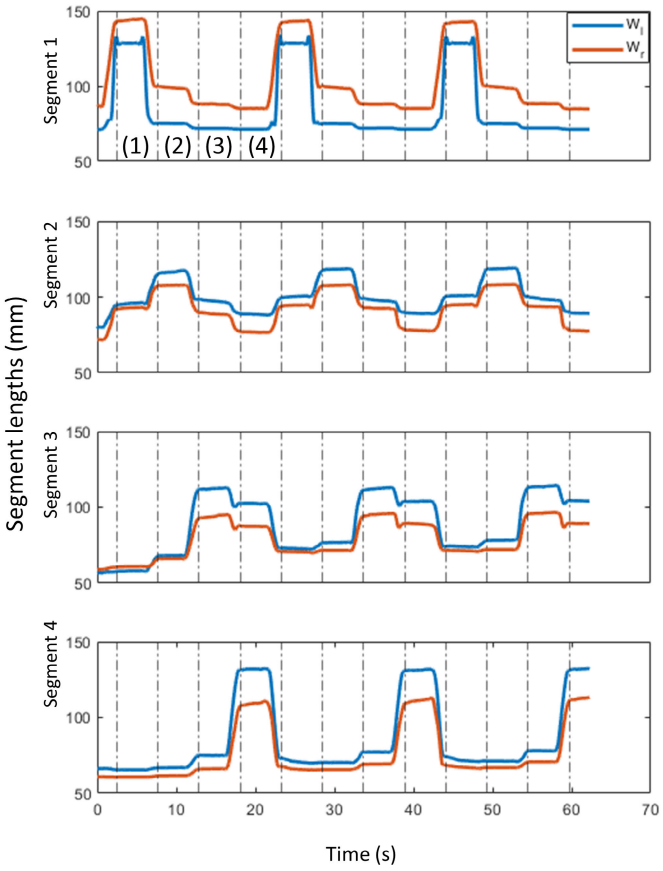


Fig. 11. Segment length changes for each segment based on stretch sensors.

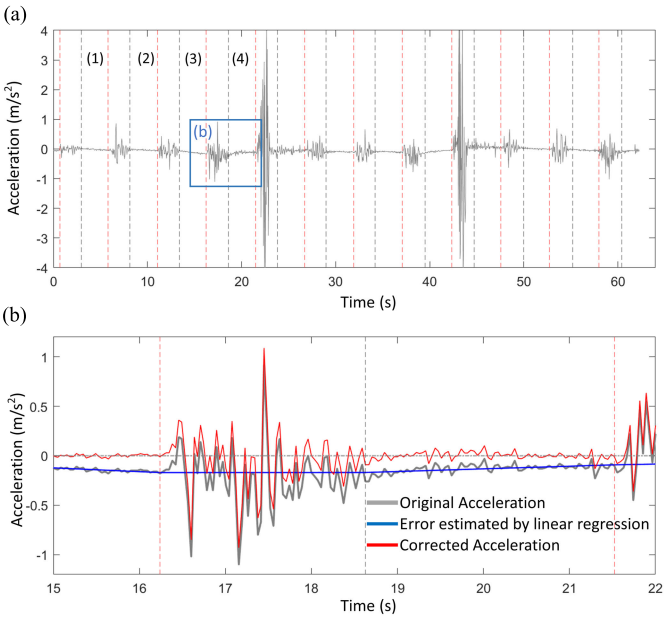


Fig. 12. (a) Original acceleration. There are three waves with four phases each. (b) Correcting the velocity by canceling a linear error to make the velocity at the starting and end time of the moving period equal to zero. The gray line is the original integrated velocity. The blue line is the estimated linear error. The red line is the corrected velocity.

Additional testing for tracking consistency and reliability was also conducted. We placed the robot as shown in Fig. 7 and kept it stationary. The vision tracking method was applied to the red rectangle area and tracked the stationary object for 63 s. The observed position data indicated a standard deviation of 0.21 mm on the X-axis and 0.12 mm on the Y-axis.

Based on the above analysis and test, we consider the vision tracking method acceptably accurate and stable. The same vision tracking method was applied for all the tests for the reference displacement of the robot.

V. Results

A. AFMM Tracking Result

The actuator outputs predict a steady motion pattern for the robot. The segments are actuated in sequence with the velocities shown in Fig. 8. After conversion and integration, as detailed in Section IV-B, the radius of each segment L_c is predicted to increase and decrease, as shown in Fig. 9. These measurements can be combined to predict robot locomotion with a forward model. For example, the motion of the center

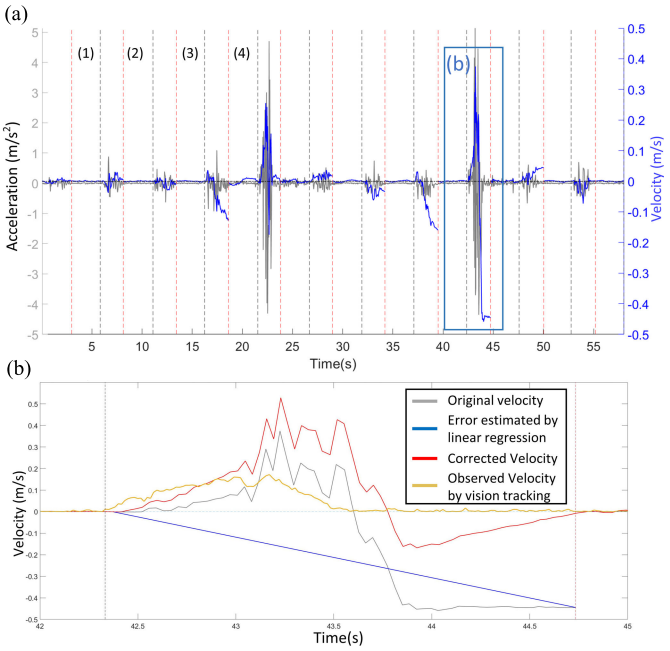


Fig. 13. (a) Original acceleration and integrated velocity. There are three waves with four phases each. (b) Correcting the velocity by canceling a linear error to make the velocity at the starting and end time of the moving period equal to 0. The gray line is the original integrated velocity. The blue line is the estimated linear error. The red line is the corrected velocity. The yellow line is the "true" observed velocity from vision tracking.

of the fourth segment is shown in Fig. 15 to be compared with the other methods below.

B. SPSM Tracking Result

We also tested displacement tracking using distributed sensors along the surface of the robot frame, as described in Section IV-C. Stretch sensors and pressure sensors are used in combination.

This tracking starts with collecting data from all these sensors and converting them into segment lengths (Fig. 11) and relative pressures: 0 for no pressure and 1 for the sensor's maximum measurement range (Fig. 10).

Note that, unlike the actuator predictions, there are small differences between the measured widths of the left and right sides of the segments. Part of the reason for this is that the left and right sensors are not at the same height (1). Internal tension is different at different heights to counteract the deformation of gravity.

Applying those real-time sensor readings to the forward modeling, we obtain displacements of each segment during the worm-like robot's locomotion. The displacement of the fourth segment is shown in Fig. 15.

C. IMU Method Tracking Result

For displacement tracking using the IMU acceleration measurements, we collected all axial accelerations for all segments. Based on this data, we estimated the final displacement by the method introduced in Section IV-D.

Fig. 12(a) shows an example of the collected data. The IMU in the fourth segment vibrates when any segment moves.

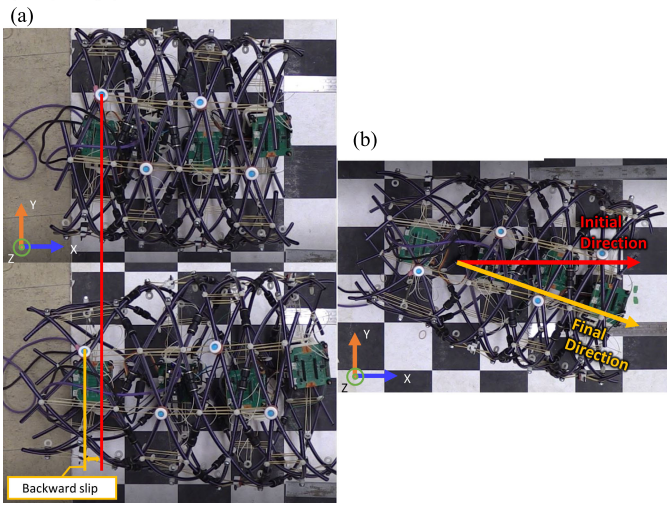


Fig. 14. (a) Backward slip of the fourth segment. The fourth segment is deforming, and its tracking point slips backward. (b) Deformation of the movement direction. There are errors in the final direction and the initial direction.

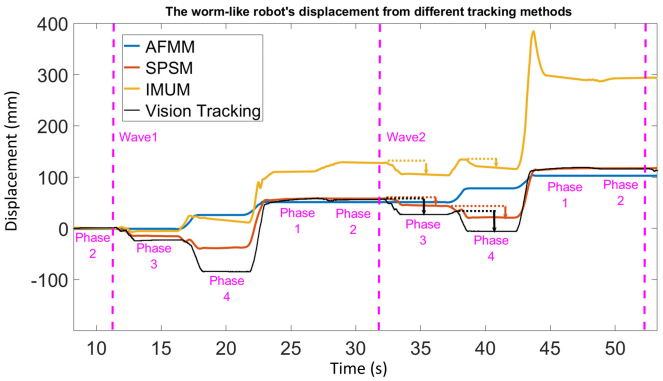


Fig. 15. Tracking results of displacement of the fourth segment over time. The robot completed two full waves during its locomotion.

We applied an error cancellation algorithm based on two principles: 1) the average axial acceleration of the segment should be zero when no actuator is functioning and 2) the start and end of the axial acceleration should be zero during the period when the actuator is running. The linear regression of the errors and the acceleration after these errors have been corrected are shown in Fig. 12(b).

Furthermore, we applied the corrected acceleration to get the integrated velocity, as shown in Fig. 13, and applied a similar algorithm to correct the velocity error. The average axial velocity should be zero when no actuator is functioning. The start and end of the axial acceleration should be zero during the period when the actuator is running. The estimated errors and corrections are shown in Fig. 13.

After the above process, we can obtain the final displacement by integrating the corrected velocity. The result of our sample test for the fourth segment is shown in Fig. 15.

D. Comparison With Vision Tracking

The displacement tracking results of each method show consistent differences. Fig. 15 shows the displacement tracking

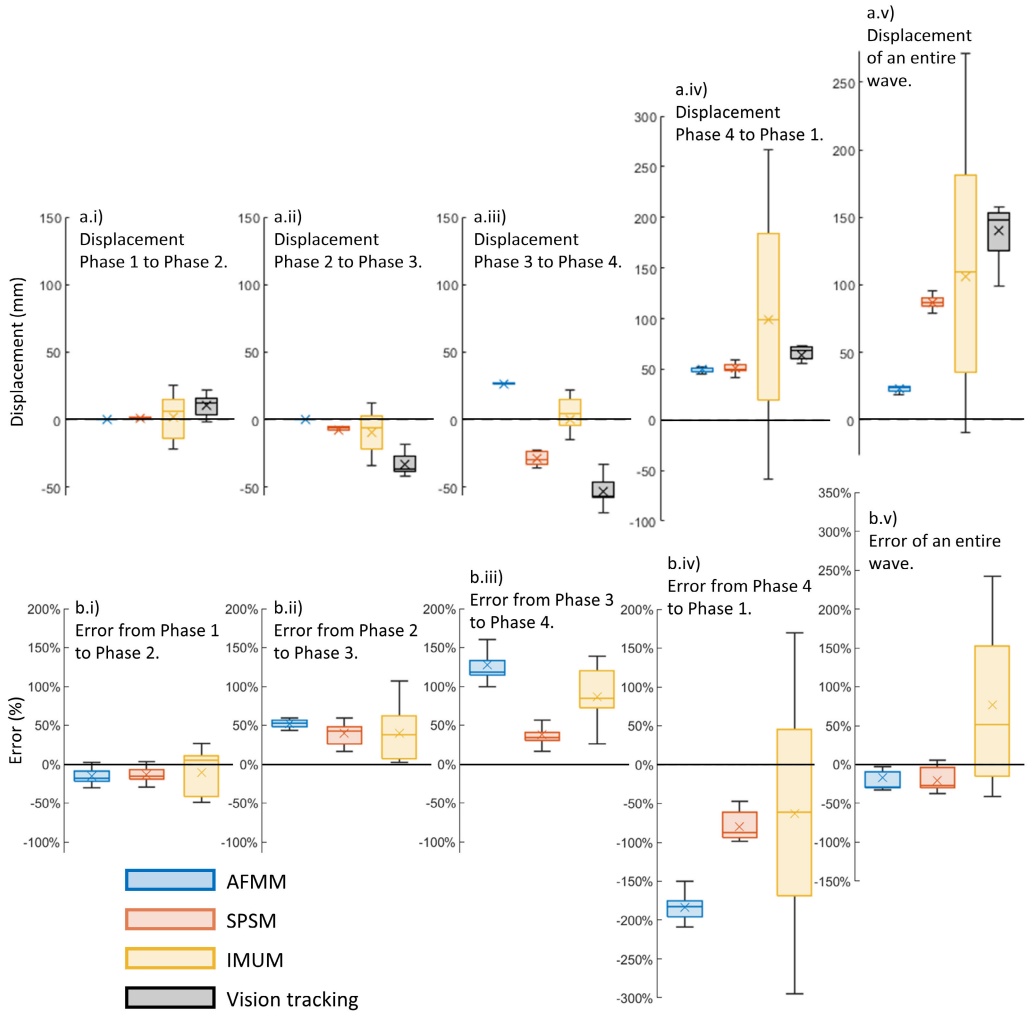


Fig. 16. (a) Displacements between each phase and during an entire wave. (b) Accumulated errors during each phase and during an entire wave. Normalized by wave displacement [data in (a.v)].

results of these methods over time for the fourth segment of the robot.

In our test, the robot completed two full waves. Each moving wave contains four segments that deform and move in sequence. The segments move via deformation, but this deformation causes slips due to the change in the relative distance between the segments and nonuniform manufacturing errors along the robot frame. Slip can reduce total displacement [Fig. 14(a)] and cause directional drifting [Fig. 14(b)]. This can be visually observed and tracked using the vision tracking method. During the test, backward slip (Fig. 14) has a more significant impact on the final displacement of the robot. All segment deformations can cause the backward slip of other segments, including those not being actuated.

We divided locomotion during a whole wave into four phases which encompass the change between each moving and stopping stage, as shown by the purple numbering in Fig. 15. The deformation and movement of the other segments will also cause slips in the fourth segment. Fig. 15 shows a backward slip during phases 3 and 4 of each wave. The vision tracking method, SPSM, and IMUM detected that the fourth segment slipped back as the displacement reading decreased in phases 3 and 4 (downward arrows in Fig. 15). In contrast, though AFMM is smooth and reasonable, it does not capture backward slips or the influence of neighboring segments on a given segment.

We repeated the displacement test five times, and in each test, the robot moved at least two full waves. The first two phases are excluded due to transients. Vision tracking and the three other methods are compared after a pause at the end of each phase (which allows vibrations of the IMU to diminish). The average displacements and accumulated errors for each wave are shown in Fig. 16(a.v) and 16(b.v). The whole wave error is also summarized in Table I. The relative error results are in percentage and are calculated based on (10) and the absolute error results are in millimeters based on (11)

$$\text{relErr}_i = \frac{Xm_i - Xv_i}{Xv_i} \tag{10}$$

$$\text{absErr}_i = Xm_i - Xv_i. \tag{11}$$

In the equations, relErr is the relative error, absErr is the absolute error, i is the wave index, Xm is the total displacement measured from the different tracking methods, and Xv is the total displacement of the vision tracking record.

TABLE I
OVERALL COMPARISON OF THE THREE TRACKING METHODS: RELATIVE ERROR (PERCENTAGE) AND ABSOLUTE ERROR (MM)

	AFMM Errors		SPSM Errors		IMUM Errors	
Mean Value	-20%	-14.4	-17%	-13.0	77%	34.6
Maximum Value	38%	13.9	38%	13.7	240%	197.3
Minimum Value	-37%	-25.1	-37%	-24.1	-41%	-125.9
Standard Deviation	22%		24%		105%	
Slip Detection	No		Clearly		Vaguely	

In addition, we use the same test results to analyze the relative errors generated for each phase. Since all waves repeat the same locomotion command, the errors during each phase are consistent. Phase-level errors allow us to better understand how errors occur and the performance of each method during the different locomotion stages of the robot. The results of the displacement and error statistics are shown in Fig. 16. Note that because the actual total displacements for each phase vary significantly, we define the comparable error with respect to the whole wave displacement of that phase, as shown in the following:

$$\text{phsErr}_{i,j} = \frac{(Xm_{i,j} - Xm_{i,j-1}) - (Xv_{i,j} - Xv_{i,j-1})}{Xv_{i,2} - Xv_{i-1,2}}. \quad (12)$$

Here, phsErr is the relative error of each phase, i is the wave index, and j is the phase index. When $j-1=0$, we decrease i by 1 and set j to 4.

From these results, we can see that for AFMM, the outputs of the actuators are stable. The actuators followed our designed velocity commands to provide periodic actuation of each segment to move forward. Very limited environment errors and interference are introduced at this level. Therefore, the AFMM results show the lowest standard deviation of errors among all tracking methods, and we consider this method to be the most stable. However, because it lacks the ability to sense environmental interference, it yields a result that is far different from the actual. As a result, although this method showed a relatively low error during the displacement of full waves [Fig. 16(b.v)], it provides the least accurate tracking result during locomotion phases 3-4 and 4-1 (Fig. 16). These two phases introduce a large amount of slip that the actuators cannot track.

SPSM results best reflect the actual position and give stable results in different waves. This method introduces a relatively larger standard deviation of error for most cases compared to AFMM, however, it can detect a backward slip. The statistics in Fig. 16 also show good tracking for all four phases; its mean error when compared to vision tracking was the smallest of the three other methods.

IMUM results in the largest standard deviation for all tracking. From Fig. 16, we can see that most of this deviation is introduced from the phase 4-1 movement. For all other phases, the IMUM's average errors are relatively reasonable. Its mean error during the phase 3-4 movement in Fig. 16 is also lower than that of AFMM. This shows that it can also detect the error generated by the environment, such as the backward slip.

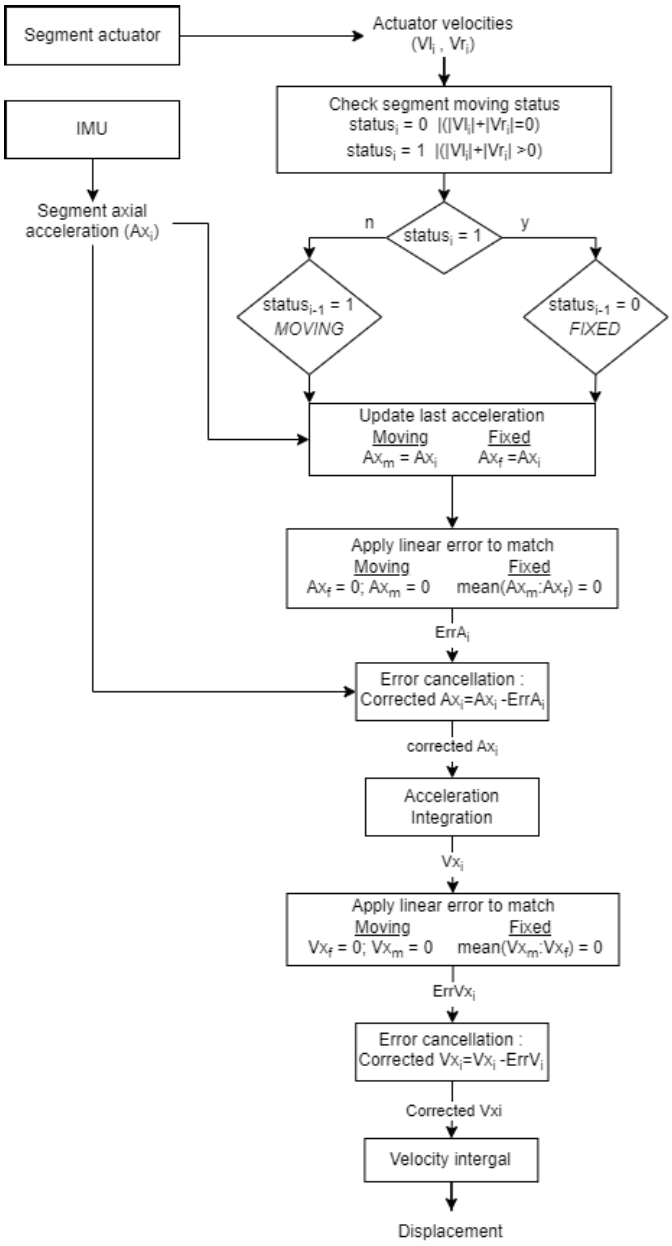


Fig. 17. IMUM diagram. The segment displacement is updated using the acceleration reading from the IMU.

VI. CONCLUSION

In this research, we designed a four-segment worm-like robot with multiple sensor setups. The design includes the platform structure (Fig. 1), the data handling circuit system (Fig. 2), and the sensing system. Specifically, in order to ensure the accuracy and reliability of the sensors, we demonstrated calibration methods for the FSRs (Fig. 3) and their corresponding holding stand for their application as pressure sensors (Fig. 4), and calibration methods for the stretch sensors (Fig. 5). We presented a simplified kinematic model (Fig. 6) for the robot locomotion. Based on this model and

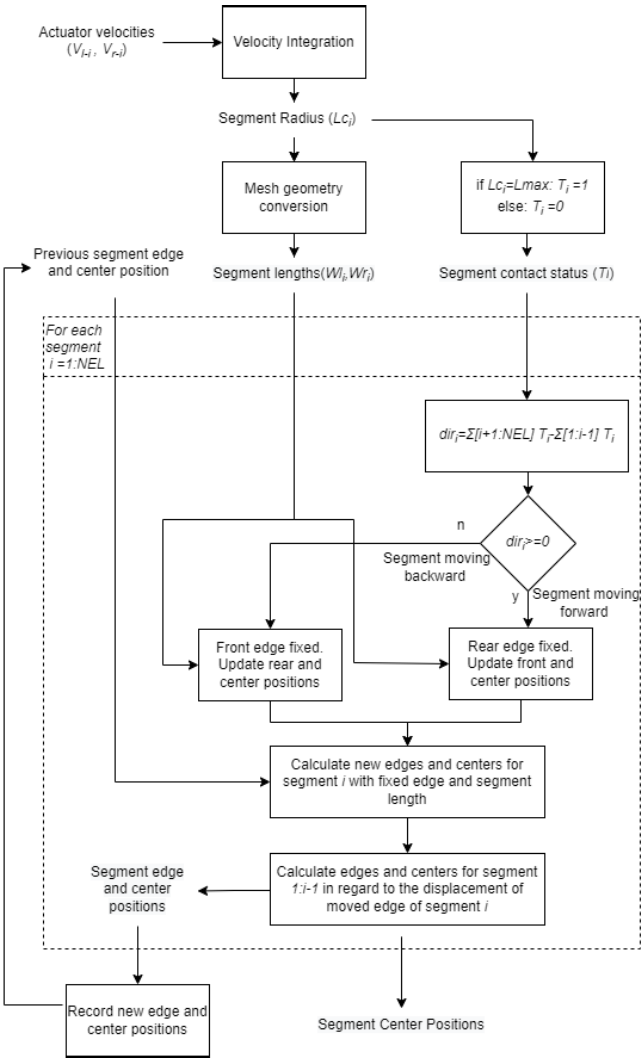


Fig. 18. AFMM diagram. The center position of the segment is updated using actuator velocity readings.

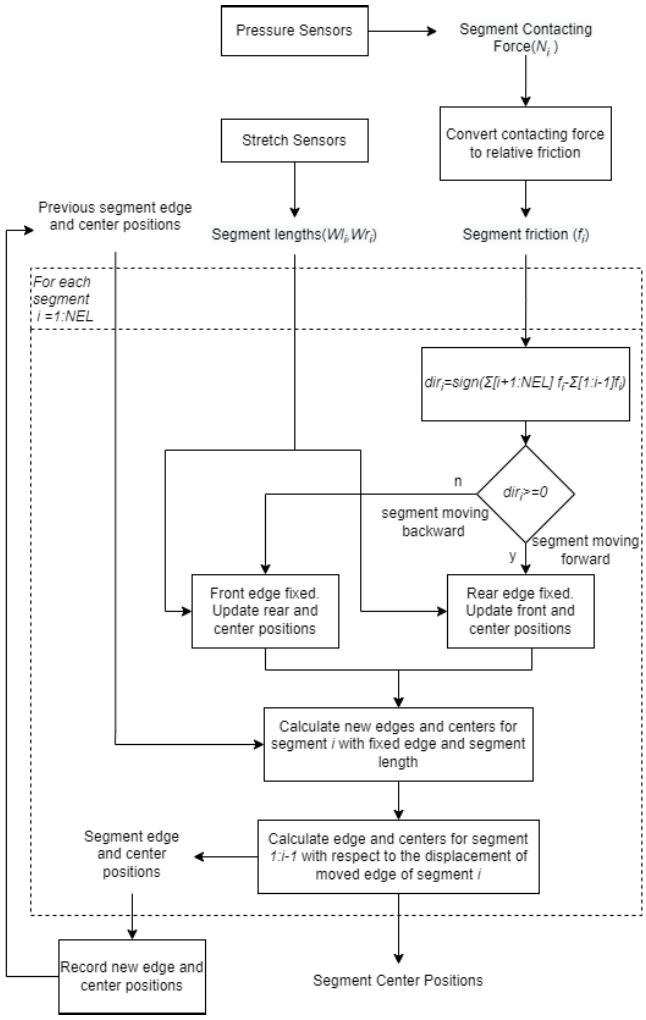


Fig. 19. SPSM diagram. The center position of the segment is updated using readings from the stretch sensors and pressure sensors.

the applied sensors, we introduced three different methods to track the displacement of the worm-like robot: the AFMM, the stretch and pressure sensors method (SPSM), and the IMUM. An additional vision-tracking method is also provided as the displacement reference (Fig. 7).

Multiple repeated tests are conducted with the same environment and the same locomotion control. We collected displacement results (Figs. 8–13) for each tracking method with the visual records (Fig. 14) and compared them with vision tracking data to obtain the tracking errors of each method (Figs. 15 and 16).

From the results, we evaluated and compared the three methods. AFMM can provide tracking with good average errors (it underestimated the displacement of a wave by an average of 20%) and the lowest standard deviation (22%). But it lacks the ability to detect detailed information about the interaction with the environment, such as a backward movement due to slipping. SPSM indicates a similar error standard deviation (24%) and the lowest average error (17% underestimation). It can also detect backward slips. IMUM records and integrates the segment acceleration data to obtain real-time displacement. It has a more significant average error (77% overestimation) and a less ideal error standard deviation (105%). The acceleration reading is independent of the relative deformation of the segments. Therefore, this method can naturally track the backward slip.

If we sort them according to the stability of the result, AFMM and SPSM are good choices. For tracking displacement that contains an unfinished step with the best error stability, the stretch sensor method is the best.

In conclusion, we have demonstrated several possible ways to do locomotion tracking for a worm-like robot with distributed sensors. These methods can also be adopted for other compliant robots that have high degrees of freedom.

One limitation of this work is that we did not compare combinations of sensor configurations. The three tested methods are based on different algorithms and assumptions. They each have their best performance during different phases of the locomotion and therefore can be complementary (Fig. 16). Together, they have the best potential for accurate localization through sensor fusion. An extended Kalman filter [31], for example, may utilize all three methods and merge the results with dynamical weights based on the reliability of the sensor

readings. This approach may effectively reduce signal noise and more accurately estimate the robot's position.

Another limitation of this work is that the experiments were performed on flat ground. On flat, smooth ground, wheels, or other locomotion approaches are likely to be the simplest way of locomoting. The potential advantages of peristaltic locomotion occur when space for large wheels is limited and when terrain roughness precludes the use of small wheels. Tracking locomotion positions is likely to be more difficult on more challenging terrain.

For the current stage, we tested the robot's locomotion on a plane surface with a straightforward path. These results will be beneficial for further research in a more complex test ground and for locomotion paths with turning. Locomotion tracking will be an essential input into peristaltic gait control. Within a gait period, any fallback can be limited by balancing the extension and retraction of segments. Within a gait cycle, especially along nonuniform terrain, a fallback is an indicator that the anchoring time of one segment is over and the next should begin. Over several gait cycles, locomotion tracking can serve as a reward as new gaits are created, executed, and compared—allowing a robot to try gaits with different numbers of moving and anchoring segments or different combinations of peristalsis, undulation, turning, and rolling.

Appendix

Figs. 17–19 correspond to the IMUM, AFMM, and SPSM algorithms described in Sections IV-B–IV-D, respectively.

Acknowledgment

The authors would like to thank Yifan Huang for sharing the original 3-D kinetic simulation code that greatly expedited their tests.

References

[1] A. Kandhari and K. A. Daltorio, "A kinematic model to constrain slip in soft body peristaltic locomotion," in *Proc. IEEE Int. Conf. Soft Robot. (RoboSoft)*, Apr. 2018, pp. 309–314.

[2] H. Wang, M. Totaro, and L. Beccai, "Toward perceptive soft robots: Progress and challenges," *Adv. Sci.*, vol. 5, no. 9, Sep. 2018, Art. no. 1800541.

[3] A. Kandhari, A. Mehringer, H. J. Chiel, R. D. Quinn, and K. A. Daltorio, "Design and actuation of a fabric-based worm-like robot," *Biomimetics*, vol. 4, no. 1, p. 13, Feb. 2019. [Online]. Available: https://www.mdpi.com/2313-7673/4/1/13

[4] H. Wang, J. Chen, H. Y. K. Lau, and H. Ren, "Motion planning based on learning from demonstration for multiple-segment flexible soft robots actuated by electroactive polymers," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 391–398, Jan. 2016.

[5] E. Gough, A. T. Conn, and J. Rossiter, "Planning for a tight squeeze: Navigation of morphing soft robots in congested environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4752–4757, Jul. 2021.

[6] Y. Wang, P. Pandit, A. Kandhari, Z. Liu, and K. A. Daltorio, "Rapidly exploring random tree algorithm-based path planning for worm-like robot," *Biomimetics*, vol. 5, no. 2, p. 26, Jun. 2020.

[7] Y. Wang, Z. Liu, A. Kandhari, and K. A. Daltorio, "Obstacle avoidance path planning for worm-like robot using Bézier curve," *Biomimetics*, vol. 6, no. 4, p. 57, Sep. 2021.

[8] D. S. Shah, J. P. Powers, L. G. Tilton, S. Kriegman, J. Bongard, and R. Kramer-Bottiglio, "A soft robot that adapts to environments through shape change," *Nature Mach. Intell.*, vol. 3, no. 1, pp. 51–59, Nov. 2020.

[9] F. Xu, H. Wang, J. Wang, K. W. S. Au, and W. Chen, "Underwater dynamic visual servoing for a soft robot arm with online distortion correction," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 3, pp. 979–989, Jun. 2019.

[10] T. Umedachi, T. Kano, A. Ishiguro, and B. A. Trimmer, "Gait control in a soft robot by sensing interactions with the environment using self-deformation," *Roy. Soc. Open Sci.*, vol. 3, no. 12, Dec. 2016, Art. no. 160766.

[11] F. Burian, P. Kocmanova, and L. Zalud, "Robot mapping with range camera, CCD cameras and thermal imagers," in *Proc. 19th Int. Conf. Methods Models Autom. Robot. (MMAR)*, Sep. 2014, pp. 200–205.

[12] P. Benavidez and M. Jamshidi, "Mobile robot navigation and target tracking system," in *Proc. 6th Int. Conf. Syst. Syst. Eng.*, Jun. 2011, pp. 299–304.

[13] T. Giffney et al., "Soft pneumatic bending actuator with integrated carbon nanotube displacement sensor," *Robotics*, vol. 5, no. 1, p. 7, Feb. 2016.

[14] K. C. Galloway, Y. Chen, E. Templeton, B. Rife, I. S. Godage, and E. J. Barth, "Fiber optic shape sensing for soft robotics," *Soft Robot.*, vol. 6, no. 5, pp. 671–684, Oct. 2019.

[15] A. Al-Ibadi, S. Nefti-Meziani, S. Davis, and T. Theodoridis, "Novel design and position control strategy of a soft robot arm," *Robotics*, vol. 7, no. 4, p. 72, Nov. 2018.

[16] J. Li, "Position control based on the estimated bending force in a soft robot with tunable stiffness," *Mech. Syst. Signal Process.*, vol. 134, Dec. 2019, Art. no. 106335.

[17] A. Dementyev, H.-L. Kao, and J. A. Paradiso, "Sensortape: Modular and programmable 3D-aware dense sensor network on a tape," in *Proc. 28th Annu. ACM Symp. User Interface Softw. Technol.*, 2015, pp. 649–658.

[18] J. Prechtl, M. Baltes, J. Kunze, S. Seelecke, and G. Rizzello, "Towards sensorless configuration estimation in multi-DoF soft robotic structures driven by rolled dielectric elastomer actuators," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2022, pp. 1152–1158.

[19] D. Zarrouk, I. Sharf, and M. Shoham, "Conditions for worm-robot locomotion in a flexible environment: Theory and experiments," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 4, pp. 1057–1067, Apr. 2012.

[20] Y. Huang, A. Kandhari, H. J. Chiel, R. D. Quinn, and K. A. Daltorio, "Slip reduction controls of mesh-body worm robot developed from a mathematical model," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2017, pp. 1474–1479.

[21] A. D. Horchler et al., "Peristaltic locomotion of a modular mesh-based worm robot: Precision, compliance, and friction," *Soft Robot.*, vol. 2, no. 4, pp. 135–145, Dec. 2015.

[22] K. A. Daltorio, A. S. Boxerbaum, A. D. Horchler, K. M. Shaw, H. J. Chiel, and R. D. Quinn, "Efficient worm-like locomotion: Slip and control of soft-bodied peristaltic robots," *Bioinspiration Biomimetics*, vol. 8, no. 3, Aug. 2013, Art. no. 035003.

[23] N. A. Rouse and K. A. Daltorio, "Visualization of stable heteroclinic channel-based movement primitives," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2343–2348, Apr. 2021.

[24] A. D. Horchler et al., "Worm-like robotic locomotion with a compliant modular mesh," in *Proc. Conf. Biomimetic Biohybrid Syst.* Cham, Switzerland: Springer, 2015, pp. 26–37.

[25] Y. Huang, A. Kandhari, H. J. Chiel, R. D. Quinn, and K. A. Daltorio, "Mathematical modeling to improve control of mesh body for peristaltic locomotion," in *Proc. Conf. Biomimetic Biohybrid Syst.* Cham, Switzerland: Springer, 2017, pp. 193–203.

[26] J. A. Florez and A. Velasquez, "Calibration of force sensing resistors (FSR) for static and dynamic applications," in *Proc. IEEE ANDESCON*, Sep. 2010, pp. 1–6.

[27] Z. Zhang, T. M. Bieze, J. Dequidt, A. Kruszewski, and C. Duriez, "Visual servoing control of soft robots based on finite element model," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 2895–2901.

[28] F. Xu, H. Wang, W. Chen, and Y. Miao, "Visual servoing of a cable-driven soft robot manipulator with shape feature," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4281–4288, Jul. 2021.

[29] Z. J. Patterson, A. P. Sabelhaus, K. Chin, T. Hellebrekers, and C. Majidi, "An untethered brittle star-inspired soft robot for closed-loop underwater locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 8758–8764.

[30] Z. Gong et al., "An opposite-bending-and-extension soft robotic manipulator for delicate grasping in shallow water," *Frontiers Robot. AI*, vol. 6, p. 26, Apr. 2019.

[31] J. Sasiadek and P. Hartana, "Sensor data fusion using Kalman filter," in *Proc. 3rd Int. Conf. Inf. Fusion*, Apr. 2018, pp. 29–36.

Yifan Wang (Member, IEEE) received the B.E. degree in measurement and control technology and instrumentation from Tianjin University, Tianjin, China, in 2014, the M.S. degree in electrical engineering from Washington University, St. Louis, MO, USA, in 2016, and the Ph.D. degree in mechanical engineering from Case Western Reserve University, Cleveland, OH, USA, in 2022.

He is currently a Robotics Engineer with EDDA Technology, Princeton, NJ, USA. His research interests include robotics and control, gait control, and path planning.

Mingyi Wang received the B.S. degree in electrical engineering from the University of California at Davis, Davis, CA, USA, in 2020, and the M.S. degree in electrical engineering from Case Western Reserve University, Cleveland, OH, USA, in 2022. He is currently pursuing the Ph.D. degree in electrical engineering with Texas A&M University, College Station, TX, USA.

His research interests include robotics and control, electrical/mechanical design, occupancy detection, and machine learning.

Natasha A. Rouse (Graduate Student Member, IEEE) received the B.S. degree in mechanical engineering from Howard University, Washington, DC, USA, in 2018. She is currently pursuing the Ph.D. degree in mechanical engineering with Case Western Reserve University, Cleveland, OH, USA.

She conducts research in the Bio-Inspired Robotics Laboratory, Case Western Reserve University, under the advisorship of Dr. Kathryn Daltorio. Her research interests include biologically inspired robotics and robotic control methods based on neuroscience.

Kathryn A. Daltorio (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in mechanical engineering from the Mechanical and Aerospace Engineering Department, Case Western Reserve University, Cleveland, OH, USA, in 2005, 2007, and 2013, respectively.

She is an Assistant Professor and the Co-Director of the Bio-Inspired Robotics Laboratory, Case Western Reserve University. She is interested in solving robotics problems that animals demonstrate are possible from climbing like insects, walking in surf zones like crabs, and squeezing into small spaces like worms.