2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561987

Autonomous Decentralized Shape-Based Navigation for Snake Robots in Dense Environments

Guillaume Sartoretti[1], Tianyu Wang[2], Gabriel Chuang[2], Qingyang Li[2], and Howie Choset[2]

***Abstract*— In this work, we focus on the autonomous navigation of snake robots in densely-cluttered environments, where collisions between the robot and obstacles are frequent, which could happen often in disaster scenarios, underground caves, or grassland/forest environments. This work takes the view that obstacles are not to be avoided, but rather exploited to support and direct the motion of the snake robot. We build upon a decentralized state-of-the-art compliant controller for serpenoid locomotion, and develop a bi-stable dynamical system that relies on inertial feedback to continuously steer the robot toward a desired direction. We experimentally show that this controller allows the robot to autonomously navigate dense environments by consistently locomoting along a given, global direction of travel in the world, which could be selected by a human operator or a higher level planner. We further equip the robot with an onboard vision system, allowing the robot to autonomously select its own direction of travel, based on the obstacle distribution ahead of its position (i.e., enacting feedforward control). In those additional experiments on hardware, we show how such an exteroceptive sensor can allow the robot to steer before hitting obstacles and to preemptively avoid challenging regions where proprioception-only (i.e., torque and inertial) feedback control would not suffice.**

I. Introduction

Animals are particularly adept at navigating obstacle-ridden environments (see Fig. 1-A) and even at benefiting from these interactions; their behavior has yet to be replicated by robotic systems [1]–[3]. One of the challenges in achieving obstacle-aided locomotion lies in the high number of degrees of freedom snake robots possess. In particular, recent works [4]–[7] have shown how using *shape-based* control, where the robot's shape remains within a family of efficient locomotive shapes, enables motion in complex, unstructured environments by performing control in a fixed-, lower-dimensional *shape space*. However, these works only focused on preventing the robot from getting stuck in these environments, but did not control its direction of locomotion. In this paper, we build upon this decentralized shape-based framework and introduce an approach that allows the robot to autonomously navigate densely-cluttered environments.

Early work in serpenoid locomotion and control [8] explored the use of contact force sensors along the body in order to avoid obstacles. However, a more modern approach has been to try to understand collisions and use them to one's advantage. More recently, Liljebäck et al. [9] model interactions between the robot and obstacles as a hybrid dynamic system, and develop a hybrid controller for locomotion. This approach, however, is limited in that the controller only becomes active when the robot detects it is stuck on an obstacle. Travers et al. [6], [7], [10] introduced a shape-based controller for compliant slithering and sidewinding, where gait parameters are continuously updated online to produce steady locomotion in unstructured environments, based on onboard torque sensing at the joints. Their work also introduces the concept of decentralized shape-based framework, which proposes shape parameters be independently updated in different portions of the robot's body based on local sensing and then propagated down the snake's backbone.

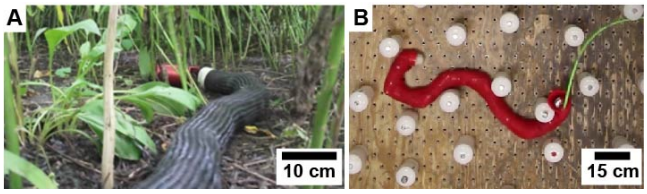

Fig. 1: Snake robots in environments of varying complexity. A. Naturally-cluttered environment; B. a model of an artificial densely-cluttered environment (a pegboard).

Our approach continuously varies the robot's gait parameters to control the heading of the robot while navigating a dense environment like the pegboard shown in Fig. 1-B. To this end, we devise a bi-stable dynamical system that allows us to incorporate inertial feedback (from onboard sensors) and obtain a steering offset that helps us stabilize the robot's LD in the world to match a given, desired direction. In particular, we show that initiating changes to the waveform at the head of the robot (using this steering offset) and subsequently propagating these changes down the body allows the robot to effectively steer by leveraging contacts with surrounding obstacles.

We further extend this control mechanism by installing an onboard vision system at the head of the robot, allowing it to explicitly select its direction of travel based on visual feedback. We experimentally demonstrate that our steering controller can enable the robot to autonomously navigate its environment by following the direction recommended by the vision system, and to negotiate more complex problems that could not be handled without relying on visual feedback and feedforward control.

The paper is organized as follows: Section II summarizes the decentralized shape-based control framework [10] which serves as a base for decentralized control in this paper.

[1] G. Sartoretti is with the Department of Mechanical Engineering at the National University of Singapore, 117575 Singapore. mpegas@nus.edu.sg

[2] T. Wang, G. Chuang, Q. Li, and H. Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213, USA. {tianyuw2, gtchuang, qingyanl, choset}@andrew.cmu.edu

Section III introduces our continuous, feedback steering controller. Section IV describes the vision-based controller used to enable the robot to select its own LD. Section V explains our experimental validation of the complete close-loop autonomous navigation framework, composed of the proprioception-only (i.e., torque and inertial feedback) steering controller and the vision-enabled controller. Finally, Section VII presents concluding remarks.

II. Background on Decentralized Shape-Based Control for Locomotion

This section presents the general framework [10] upon which we build to control a snake robot in a decentralized manner, by adapting gait parameters in different portions of the snake based on local sensing before propagating these values down the snake's backbone.

A. Serpenoid Curve Overview

We consider an N-jointed snake robot, with state variable $\theta(t) = (\theta_1(t), ..., \theta_N(t)) \in \mathbb{R}^N$ expressing the joint angles $1 \leq i \leq N$ at time t. Hirose et al. [8] described a family of curves, the serpenoid curves, that can parameterize the shape of a snake robot during locomotion under a variety of gaits. In this work, we use a gait called *planar slithering* which is essentially controls the snake robot to assume the shape of a time-varying serpenoid curve, i.e., [7], [8], [11]:

$$\theta_i(t) = \theta_0 + A \sin\left(\omega_S \, s_i - \omega_T \, t\right), \tag{1}$$

where θ_0 is a constant curvature offset, s_i is the distance from the head of the snake to the joint i along the snake's body, A, ω_S and ω_T respectively are the amplitude, the spatial and temporal frequencies of the serpenoid curve. The temporal frequency ω_T defines the (temporal) period of the serpenoid gait ($T = 2\pi/\omega_T$), whereas the spatial frequency ω_S determines the number of sinusoidal periods n_S along the robot's body (via $\omega_S = 2\pi \, n_S$).

B. Shape Functions for Dimensionality Reduction

Throughout this paper, we consider two controllers for the robot that build upon the decentralized *shape-based control* introduced in [7], [10], which we briefly summarize in this section. Shape-based control makes use of shape functions, as a natural way to reduce the dimensionality in systems with high degrees-of-freedom [10]. The goal of such a shape function $h : \Sigma \rightarrow \mathbb{R}^N$ is to determine $\theta(t)$ as a function of a small number of control parameters. Those parameters create a shape space Σ, which is usually of a lower dimension than $\mathbb{R}^N$. In our case, we consider the case where the amplitude and spatial frequency in Eq.(1) can be dynamically set and therefore form a two-dimensional shape space. The shape function $h(A, \omega_S)$ of our system is simply the serpenoid curve Eq.(1), with the two control parameters being the amplitude and spatial frequency, while the other parameters are fixed:

$$\begin{aligned} h : \Sigma = \mathbb{R}^2 &\mapsto \mathbb{R}^N \\ h_i(A(t), \omega_S(t)) &= \theta_i(t) \\ &= \theta_0 + A(t) \sin\left(\omega_S(t) \, s_i - \omega_T \, t\right). \end{aligned} \tag{2}$$

C. Decentralized Control

More recent works on the shape-based compliant framework proposed the idea of *decentralizing* control, to endow the robot with local reflexes and better handle situations where different portions of the robot would need different modifications of the shape parameters [7]. These sub-groups of joints are usually created based on the current shape of the robot, and propagate down its backbone during locomotion. Specifically, let us define W windows (i.e., groups of joints), in which independent values $A_j(t)$ and $\omega_{S,j}(t)$ $(1 \leq j \leq W)$ are defined for the amplitude and the spatial frequency. The decentralized time- and space-dependent control parameters $\xi(t, s)$ $(\xi = \{A, \omega_S\})$ are now expressed in a sum-of-sigmoids form:

$$\xi(t, s) = \sum_{j=1}^{W} \xi_j \cdot \left(\frac{1}{1 + e^{-m(s - a_j(t))}} + \frac{1}{1 + e^{-m(b_j(t) - s)}} \right), \tag{3}$$

with $a_j(t), b_j(t)$ respectively the start and end position of the j-th window along the snake's body, and $m \in \mathbb{R}$ the slope of the sigmoid functions.

In this paper, following [7], we also let the windows' position and number be dynamically set based on the current shape of the robot's body. The first window encompasses all joints from the robot's head to the first point of zero-curvature along its body. The following windows are defined between two successive points of zero-curvature along the robot's body, and the last window between the last point of zero curvature and the tail. We also let the windows naturally move down the body at the same rate as the serpenoid curve (i.e., $2\pi/\omega_T$). This process enables us to naturally pass information along the robot's body, as the snake locomotes through its environment.

III. Continuous Steering Controller

For navigation in dense environments, we build upon the decentralized compliant controller [7], [10] to endow a snake robot with steady, uninterrupted locomotion in unstructured environments. On top of this controller, we then present a continuously active controller to keep the snake robot's motion in a desired direction of travel by relying on onboard inertial sensing.

To this end, we first present a novel method to predict the current direction of motion from the robot's current gait phase and head orientation in the world. Similar to the discrete case, we show that the head joint is central when estimating the locomotive direction of the snake, especially during interactions with obstacles. Then, we describe our dynamical system approach to autonomous steering, which relies on the current and desired directions of motion of the robot to continuously steer it.

A. Locomotive Direction Estimation

In dense environments, a snake robot needs to constantly be correcting it direction of motion, since this direction may be changed by the many collisions with obstacle the robot experiences. To this end, we first need to estimate

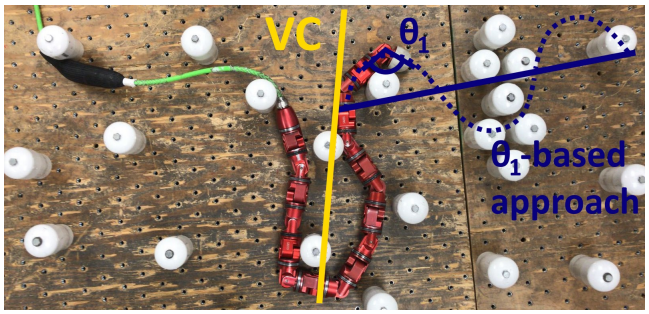

Fig. 2: Locomotive direction (LD) of the snake robot, estimated incorrectly by the Virtual Chassis (VC) and correctly with our method which relies on the serpenoid phase of the head module, $\theta_1(t)$. Additionally, note that the VC is struggling particularly in this case: it is hard for this approach to predict the direction of the yaw, in addition to mis-estimating its orientation.

the snake robot's current locomotive direction (LD). To do so, one approach would be to estimate the "main axis" of the robot in the world, i.e., the axis in the world to which the serpenoid shape of the robot can be best fitted. When considering snake robots following regular gaits (such as slithering, sidewinding, etc.), the robot's main axis can be estimated using the Virtual Chassis (VC) [12]. In this approach, the snake's main axis is computed by obtaining the largest singular value of the set of the joints' positions in the frame of the robot's head joint. The positions of joints can be obtained in this frame, by finding their relative positions and orientations from the current shape of the snake. Once the main axis of the snake is computed in the frame of the snake's head, it can be transformed into the world frame based on the head's orientation in that frame (measured from onboard IMUs [13]).

The VC approach gives very robust results when the shape of the snake is regular, that is, when the snake robot's shape is close to a serpenoid curve and the main axis is well therefore well defined. During sharp direction changes, however, the robot's shape may become irregular enough to be estimated incorrectly by using the VC method (as shown in Fig. 4). In this context, we propose a new approach to estimate the robot's main axis, that performs better than the VC when the robot's shape is not regular, and gives the same result in the other cases. With our method, the snake's planar LD $\varphi(t) \in [-\pi; +\pi]$ is measured from predicted joints positions rather than from the current ones. We predict the future evolution of the head joint's position in the world frame, using the serpenoid curve's shape from Eq.(1) with the current control parameter values. That is, we propagate the serpenoid Eq.(1) in time from the current head angle $\theta_1(t)$, and extract the main x-axis in the world frame with respect to which the sinusoidal curve is currently being defined. We then perform a singular value decomposition on this set of joint positions to estimate the main axis of the current robot's LD. Fig. 2 shows an example robot's body shape confusing the VC method into giving an erroneous reading for the main axis, while our LD estimation method is able to handle the same scenario.

Our approach uses the current time and the robot's state

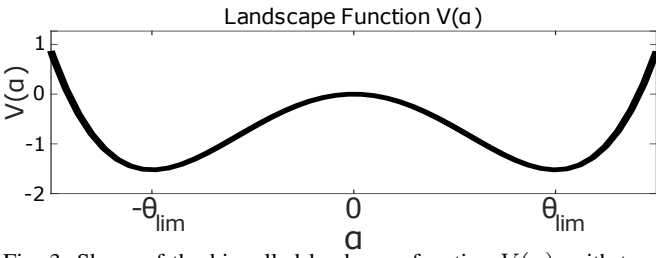

Fig. 3: Shape of the bi-welled landscape function $V(\alpha)$, with two minima located at $\alpha = \pm\theta_{lim}$ and a saddle point at $\alpha = 0$.

(i.e., control parameter values) of Eq.(1) to infer the robot's main axis, rather than just using the current joints' positions, as is done with the VC method. Note that when the robot's shape is regular, our method returns the same main axis as the VC. In our case, we rely on Eq.(1) to predict the current planar angle between the head joint's orientation and the LD, $\frac{\mathrm{d}}{\mathrm{d}\tau}\theta_1(\tau)\Big|_{\tau=t}$ at time t. Finally, we are able to analytically express the current LD with respect to the current time, control parameters values and head joint's yaw angle $\alpha_H(t)$ (measured from onboard IMUs)

$$\varphi(t) = \alpha_H(t) - \frac{\mathrm{d}}{\mathrm{d}\tau}\theta_1(\tau)\Big|_{\tau=t}. \tag{4}$$

B. Steering Window

In this work, we propose a steering mechanism where a time-dependent curvature offset $\theta_s(t)$ is added to each of the joint angles in the first window of the snake, which become

$$\theta_i(t) \longmapsto \theta_i(t) + \delta_{a_1(t)\leq x\leq b_1(t)}(s_i)\cdot\theta_s(t), \tag{5}$$

with $a_1(t), b_1(t)$ respectively the start and end positions of the first window along the snake's body, and the function $\delta_{a_1(t)\leq x\leq b_1(t)}$ returns 1 if and only if $a_1(t) \leq x \leq b_1(t)$ (if x lies in the head window), and 0 otherwise.

Naturally extending the use of decentralized control windows being passed along the snake's body, we also let the steering offset be propagated down together with the first window. In particular, when a discrete point of higher curvature is created at the end of the first window by the steering mechanism Eq.(5), this bend in the robot's body is passed along its body as the robot continues its evolution, by relying on the window propagation mechanism described in Sect. II-C. Therefore, when the robot is able to successfully leverage a contact from its surrounding environment, around which it is able to hook in order to change its direction, the rest of the robot's body will naturally follow the first joints and steering will be achieved.

During a large steer, Eq.(4) allow us to link the snake robot's current shape with its future motion direction. This prediction step is an approximation, and relies on the assumption of a dense and contact-full environment surrounding the robot (which is often validated in practice). In this environment, the underlying compliant control for locomotion [7], [10] helps the robot differentiate between contacts that can be leveraged for efficient locomotion and contacts that could get the robot stuck. In doing so, the compliant controller effectively aims to keep the snake under force

closure most of the time. During a steering maneuver, the head window of the robot gets an additional steering offset that allows the robot to search for contacts with obstacle that can be leveraged into changing the direction of travel. Once such contacts are found, the front of the robot can be assumed to be under force closure again. Under this assumption, we can now approximately model the robot under a *follow-the-leader* planar evolution, where the snake's joints simply retrace the head joint's position in space with a delay. The windows and steering offset being passed down the robot's body naturally help support this model, since successive windows share the same control parameters with a delay.

C. Dynamical System Modeling

We model the steering offset $\theta_s(t)$'s evolution with time as a dynamical system. This approach has previously been used to successfully implement dynamic gaits changes in mobile robots [6], [14]. In our case, we want the steering offset to build up with time as the heading remains incorrect, and to remain within the joint angles' limits $\theta_s(t) \in [-\theta_{\text{lim}}; \theta_{\text{lim}}] \subset [-\pi; \pi]\ \forall t$. Additionally, we experimentally observed that this steering mechanism should be active (non-zero) as often as possible to ensure stable locomotion in the desired direction of travel. To this end, we construct a bi-stable landscape function $V(\alpha) : [-\pi; \pi] \to \mathbb{R}$, with two potential wells located at $\pm\theta_{lim}$ (and a saddle point at 0):

$$V(\alpha) = \alpha^4 - 2\,\alpha^2\,\theta_{lim}^2. \tag{6}$$

Fig. 3 depicts the general shape of the landscape function $V(\alpha)$. We now let $\theta_s(t)$ evolve in the landscape defined by $V(\alpha)$, following the differential equation:

$$\dot{\theta}_s(t) = -\beta_1 \left.\frac{\mathrm{d}}{\mathrm{d}\alpha} V(\alpha)\right|_{\alpha=\theta_s(t)} + \beta_2 \cdot ((\varphi(t) - \varphi_d(t)) - \theta_s(t)), \tag{7}$$

with $\beta_1, \beta_2 \in \mathbb{R}_+$ two control parameters weighting the two contributions, and $\varphi_d(t) \in [-\pi; \pi]$ the desired robot's heading in the world frame. The first contribution in Eq.(7) drives the steering offset to gradually build up, as the offset is naturally increased toward one of the extrema $\pm\theta_{lim}$. The second contribution links the current heading error and the steering offset, to enable the offset to switch from one of the wells of $V(\alpha)$ to the other one, as the heading error $(\varphi(t) - \varphi_d(t))$ changes its sign. In other words, the additional steering offset Eq.(7) acts as an additional, window-specific curvature offset, that is always introduced at the head window based on inertial feedback. This curvature offset is then naturally passed down the body of the robot, by relying on the shape-based decentralized framework for locomotion.

IV. Vision-Enabled Autonomous Navigation

The steering controller presented in Section III allows the snake robot to locomote in any desired direction using onboard inertial and torque sensing. However, a human operator is still needed for manually inputting the desired LD φ_d. In order to realize fully autonomous navigation in dense environments, we further propose to rely on visual feedback from an onboard camera, which allows the robot to reason about the choice of its path, i.e., the choice of the desired direction φ_d, based on the obstacle distribution ahead of the robot.

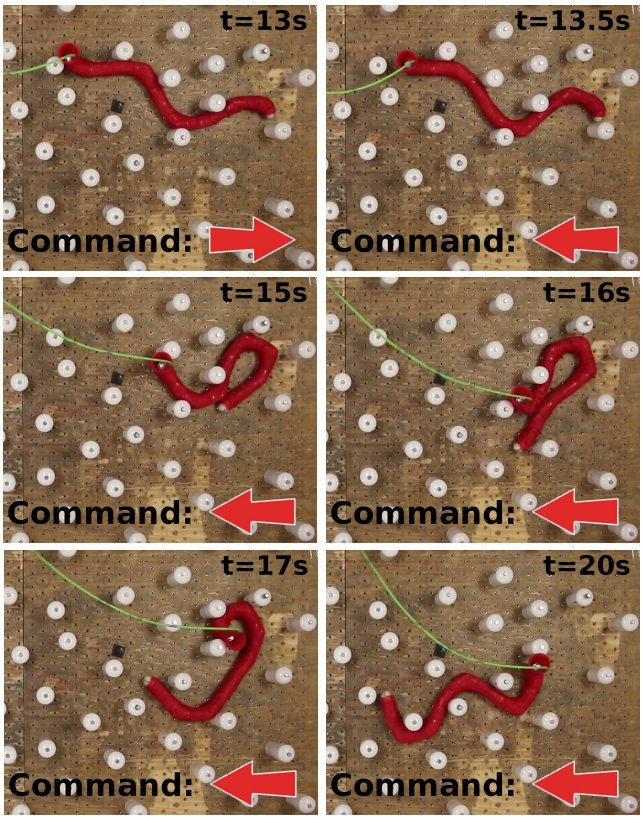

Fig. 4: Successive frames from the experimental validation video for the continuous inertial steering controller, showing a complete 180 degree turn commanded to the robot. Note how the head window is responsible for initiating the turn, followed by the other joints, until the whole robot is turned in the new desired direction.

To this end, we rigidly attach an Intel RealSense Depth Camera D435 to the head of the robot; the camera generates raw 3D point clouds, which we use to create a local map of the environment immediately ahead of the robot. Based on this map, we then show how the robot can explicitly select its own safe LD.

A. Obstacle Heat Map Generation

We generate a 2D heat map of the distribution of obstacles based on the points given by the depth camera. We first filter our points to only include a cuboidal (3D) region of interest (ROI). Then, we discard the z-coordinate of each point in that ROI, and combine them into discretized cells, resulting in an $n \times n$ heat map for a chosen resolution n, as shown in Fig. 5-A. The intensity $I_{x,y}$ of a given cell is simply the number of points located within the boundaries of that cell. The heat map is quadratically scaled to account for the fact that farther obstacles intercept smaller solid angles and thus have fewer total points in the point cloud:

$$I'_{x,y} = \left(\frac{y}{n}\right)^2 I_{x,y}. \tag{8}$$

Finally, all cells with intensity below a fixed threshold ϵ are set to 0, with ϵ defined as follows:

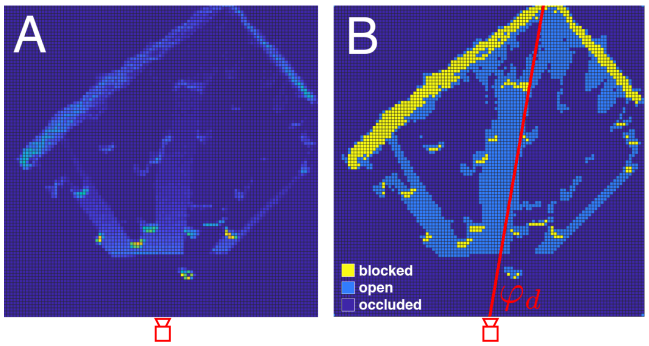

Fig. 5: A. Original heat map with resolution $n = 100$; B. Ternary classified heat map showing the resulting desired locomotive direction (LD) φ_d, which is then used by our navigation controller.

$$\epsilon = c \cdot \left(\max_{x,y} I_{x,y} + \text{median}(I) \right), \tag{9}$$

where c is a constant threshold factor between 0 and 1 ($c = .25$ in practice). This threshold essentially filters for points whose intensities are "at least c above average," leaving nonzero values only at cells with obstacles.

Based on the heat map generated as described above, we construct a ternary (three-valued) heat map, containing three distinct values, classifying each cell as either *blocked*, *open*, or *occluded*:

- A cell is *blocked* if it has a nonzero value in the original heat map, i.e. there is an obstacle in that space.
- A cell is *open* if the cell's intensity $I_{x,y}$ was below the threshold ϵ but greater than 0 (i.e. that cell was set to 0 during thresholding; these are navigable "floor" points).
- A cell is *occluded* if the camera did not detect any points in that cell at all; these cells are in areas that are outside the FOV of the camera or entirely behind an obstacle, that is, located in the obstacle's "shadow".

An example ternary heat map is shown in Fig. 5-B.

B. Desired locomotive direction

Based on the ternary heat map, the robot's desired LD φ_d is determined by minimizing the sum of $B(x, y)$ values for x, y cells along a line in the direction of φ_d, where $B(x, y)$ is defined as follows:

$$B(x,y) = \begin{cases} 1 & \text{if cell } (x,y) \text{ is blocked} \\ 0 & \text{if cell } (x,y) \text{ is occluded} \\ -1 & \text{if cell } (x,y) \text{ is open} \end{cases} \tag{10}$$

We choose the direction φ_d that minimizes $B(x, y)$. An example of determining φ_d based on the ternary heat map is shown in Fig. 5-B. By choosing a φ_d that favors *open* cells and avoids *blocked* cells, we ultimately end up with a feedback mechanism that guides the robot to avoid obstacle clusters which are hard to slither through and favors navigable gaps between obstacles where the robot can leverage contacts for its propulsion.

V. Experimental Validation

We implemented the steering mechanism presented in this paper on a snake robot, composed of sixteen identical series-elastic actuated modules [15]. The modules were arranged such that two neighboring modules were torsionally rotated 90 degrees relative to each other. Only the planar modules (i.e., those actuating in the world's xy-plane) are actuated, while the dorsal modules are commanded to constant zero joint angles. The deflection between the input and output of a rubber torsional elastic element is measured using two absolute encoders, allowing us to read the torque measured at each module's rotation axis. A braided polyester sleeve covered the snake, reducing friction with the environment and forcing the snake to leverage contacts to locomote.

A. Proprioception-Only, User-Controlled Navigation

We first experimentally validated the proprioception-only steering mechanism described in Section III in an artificial environment, where obstacles (vertical posts of different sizes) were randomly distributed to form obstacles with varying sizes and densities. There, external high-level manual input was used to allow changes in the desired LD $\varphi_d(t)$ with time in the presented 8min-long experiment. Via the use of four buttons, the desired LD in the world frame could be changed by a user online between the four cardinal directions (the north being defined as the initial heading of the robot). The video of the experimental validation experiment is available at `http://bit.ly/ICRA21-SnakeSteering`, while a sequence of still images is presented in Fig. 4.

B. Vision-Enabled, Autonomous Navigation

We then implemented the vision-enabled autonomous navigation framework composed of the same steering controller, as well as the vision-based controller presented in Section IV on the same snake robot. A similar (humanly-)randomized environment was used for these experiments.

In this experiment, the robot paused approximately every 5 seconds during locomotion, at times when the camera's principal axis was aligned with the robot's LD, to allow the acquisition of precise visual feedback without any motion blur. During each pause, the vision-based controller determined the desired LD $\varphi_d(t)$, which guided the robot to avoid large clusters of obstacles that are impossible for the robot to slither through. While the robot was locomoting, the steering controller was continuously active to keep the desired direction $\varphi_d(t)$. The image processing and computation of $\varphi_d(t)$ took less than 0.5s; thus, the short pauses minimally affected the continuity of locomotion.

A video of this experimental validation experiment is available at `http://bit.ly/ICRA21-SnakeSteering`. A sequence of video frames is shown in Fig. 6, illustrating how the proposed vision-enabled autonomous navigation system allows the snake robot to traverse a cluttered environment without any human inputs.

C. Comparison of Both Navigation Controllers

Finally, we compared the performance of the proprioception-only, user-controlled navigation to the implementation of the vision-enabled autonomous

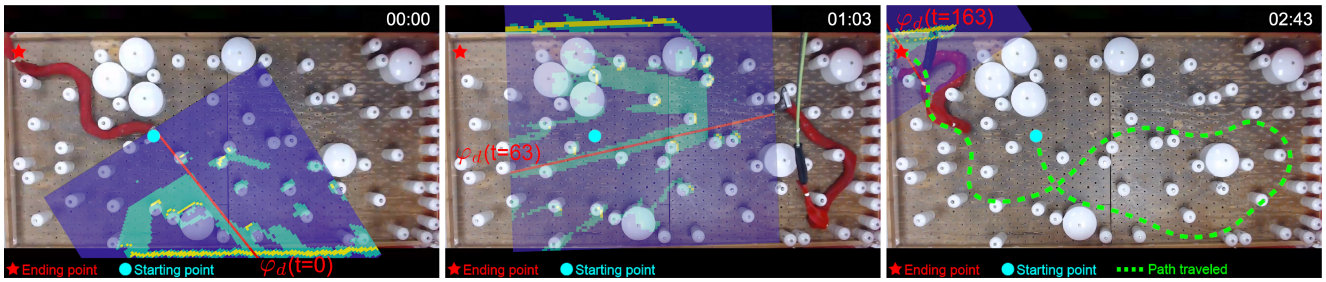

Fig. 6: Snapshots from the experimental validation video for the proposed vision-enabled autonomous navigation framework. The robot is capable of autonomously traversing the densely-cluttered environment. In each frame, the obstacle heat map generated by the vision-based controller is displayed in the shaded blue area; the robot's desired propagating direction determined by the vision system is shown by the solid red line.

navigation in locally traversing an unstructured environment. A video comparison can be found at `http://bit.ly/ICRA21-SnakeSteering`.

As expected, navigation with vision feedforward is a significant improvement in environments with visually distinguishable, non-navigable clusters of obstacles, since the feedforward system can pre-emptively identify potential "problem areas" and steer ahead of time to avoid them. Conversely, proprioception-only navigation can only attempt to negotiate the obstacles once it is aware of their existence, i.e., after making contact with them, at which time the robot could already be stuck against a row of obstacles with no clear means to get itself unstuck short of tracing back its path.

VI. Discussion

In practice, the full, vision-enabled navigation framework detailed in this work was very accurate in choosing an appropriate direction $\varphi_d(t)$, in an environment with a range of obstacle shapes and sizes (small pegs, large pegs, pegboard frame, and wall, as well as clusters of the above). Our framework also appeared robust to the presence of large non-navigable empty spaces (such as off the edge of the elevated pegboard), even if those areas had no obstacles.

However, our current use of visual feedback still exhibits several physical limitations. For instance, the camera has a minimum viable range of around 11cm, which prevents it from detecting obstacles that the robot is directly adjacent to. Similarly, at ranges above 1.5m, the 3D point cloud generated by the camera becomes relatively unstable and noisy, limiting our ability to detect further-off "dead ends" if their end is outside of the field of view of the camera. Finally, we note that the camera must be relatively steady in order to generate usable visual feedback, thus necessitating the 0.5s pauses described previously, which might limit the use of our method in time-critical scenarios.

Importantly, since the system is inherently an local planner, it does no high-level mapping or localization. In contrast to "proprioception-only" *global* navigation, which chooses a target direction and preserves a constant LD to that direction, our vision-based *local* navigation system currently only selects a locally high-quality LD, without considering prior information nor global targets.

This means that the robot could be caught in a cycle of short-term optima, rather than progressing towards some global target. For instance, the vision feedforward system may cause the robot to get caught in a loop of successive open areas that cause it to periodically re-visit the same locations, while making no progress in the long term - a scenario that the proprioception-only controller would avoid due to its global LD. Future work includes integration of limited vision-based localization, which can be used to combine these two approaches (local and global planning), which due to scope and space constraints were not treated in this work.

VII. Conclusion and Perspectives

In this work, we developed a closed-loop autonomous navigation framework for snake robot locomotion in unstructured dense environments, which we experimentally validated on hardware in a number of experiments. Our framework is built upon a novel, dynamical system based continuous steering controller, which builds upon the state-of-the-art shape-based decentralized compliant control framework. While this controller allows the robot to continuously steer and keep a given direction of travel in the world, we further proposed to allow the robot to explicit select this direction based on visual feedback from an onboard camera, e.g., to steer away and avoid challenging regions that could not be negotiated through feedback control only.

The continuous steering controller and the vision feedforward system can be seen as modular pieces that we would like to further investigate the coordination of them in the future work. That is, our steering controller enables the robot to navigate in a given *global* direction in the world, while visual feedback can allow *local*, predictive path planning. However, integrating these long- and short-term planning methods is nontrivial, since their individual goals might not agree, especially around complex scenarios, e.g., situations where multiple routes are possible, or situations where the only safe course of action does not agree with the desired global direction of travel. Future works will look to filtering/consensus methods to dynamically combine these two modular controllers, as well as to explicit SLAM methods to allow the robot a higher-level of reasoning about global path planning and already visited areas, e.g., for autonomous exploration tasks.

REFERENCES

[1] F. Sanfilippo, J. Azpiazu, G. Marafioti, A. A. Transeth, Ø. Stavdahl, and P. Liljebäck, "Perception-driven obstacle-aided locomotion for snake robots: the state of the art, challenges and possibilities," *Applied Sciences*, vol. 7, no. 4, p. 336, 2017.

[2] J. Whitman, N. Zevallos, M. Travers, and H. Choset, "Snake robot urban search after the 2017 mexico city earthquake," in *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2018.

[3] P. E. Schiebel, J. M. Rieser, A. M. Hubbard, L. Chen, and D. I. Goldman, "Collisional diffraction emerges from simple control of limbless locomotion," in *Conference on Biomimetic and Biohybrid Systems*. Springer, 2017, pp. 611–618.

[4] M. Travers, J. Whitman, P. Schiebel, D. Goldman, and H. Choset, "Shape-based compliance in locomotion," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[5] T. Wang, J. Whitman, M. Travers, and H. Choset, "Directional compliance in obstacle-aided navigation for snake robots," in *2020 American Control Conference (ACC)*, 2020, pp. 2458–2463.

[6] F. Ruscelli, G. Sartoretti, J. Nan, Z. Feng, M. Travers, and H. Choset, "Proprioceptive-inertial autonomous locomotion for articulated robots," in *ICRA 2018 - IEEE International Conference on Robotics and Automation*, 2018, pp. 3436–3441.

[7] J. Whitman, F. Ruscelli, M. Travers, and H. Choset, "Shape-based compliant control with variable coordination centralization on a snake robot," in *CDC 2016*, December 2016.

[8] S. Hirose, *Biologically Inspired Robots: Serpentile Locomotors and Manipulators*. Oxford University Press, 1993.

[9] P. Liljebäck, K. Pettersen, Ø. Stavdahl, and J. Gravdahl, *Snake Robots: Modelling, Mechatronics, and Control*, 01 2013.

[10] M. Travers, J. Whitman, and H. Choset, "Shape-based coordination in locomotion control," *The International Journal of Robotics Research*, p. 0278364918761569, 2018.

[11] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, 2009.

[12] D. Rollinson and H. Choset, "Virtual chassis for snake robots," in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 221–226.

[13] D. Rollinson, H. Choset, and S. Tully, "Robust state estimation with redundant proprioceptive sensors," in *ASME 2013 Dynamic Systems and Control Conference*, October 2013.

[14] M. Travers, A. Ansari, and H. Choset, "A dynamical systems approach to obstacle navigation for a series-elastic hexapod robot," in *2016 IEEE Conference on Decision and Control*, December 2016.

[15] D. Rollinson, Y. Bilgen, B. Brown, F. Enner, S. Ford, C. Layton, J. Rembisz, M. Schwerin, A. Willig, P. Velagapudi, and H. Choset, "Design and architecture of a series elastic snake robot," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 4630–4636.