

Modeling Rolling Gaits of A Snake Robot

Weikun Zhen, Chaohui Gong and Howie Choset

Abstract—Successful deployment of a snake robot in search and rescue tasks requires the capability of generating controls which can adapt to unknown environments in real-time. However, available motion generation techniques can be computationally expensive and lack the ability to adapt to the surroundings. This work considers modeling the rolling motion of a snake robot by applying the *Bellows model* with computation reduction techniques. One benefit of this is that controllers are defined with physically meaningful parameters, which in turn allows for higher level control of the robot. Another benefit is that it allows controllers to be defined by “composing shapes”, which enables developing controllers that can adapt to the surroundings. Using shape composition, we implemented a novel gait, named *rolling hump*, which forms a contour-fitting hump to negotiate obstacles. The efficacy of a snake robot climbing over obstacles by using the *rolling hump* is experimentally evaluated. An autonomous control strategy is presented and realized in simulation.

I. INTRODUCTION

Mobility in challenging environments is one of the essential requirements of field robots. A crucial element of successful locomotion in unstructured environments is fast and robust motion generation that can adapt to the surroundings. Snake robots, a type of field robot inspired by biological snakes, have the potential to handle complex and unstructured environments by taking advantage of their many degrees of freedom (DoFs). This also poses challenges in planning and control of these robots.

The motion generation techniques for a snake robot can be divided into two categories. One approach is to directly generate the joint angle trajectory through a set of parametrized equations, i.e. gait equation. The gait equation simplifies the control by reducing the effective degrees of freedom of a snake robot while sacrificing its “flexibility” and potentially restricting its mobility in unstructured environments. An alternate approach is modeling the shape of the snake robot in the workspace as a spatial curve, called *backbone curve* [2], which allows users to intuitively design motions in the form of shape changes in the backbone curve. The drawback of the backbone curve approach is its reliance on computationally expensive fitting algorithms [1][3], which compute the joint angles of a discrete mechanism such that it can be shaped into the target backbone curve. Due to the relative computational efficiency, our work is built on the curve fitting method originally proposed in [1].

Weikun Zhen is with the Department of Mechanical Engineering, Carnegie Mellon University, weikunz@andrew.cmu.edu

Chaohui Gong is with the Department of Mechanical Engineering and the Robotics Institute, Carnegie Mellon University, chaohuig@cmu.edu

Howie Choset is with the Robotics Institute, Carnegie Mellon University, choset@cmu.edu

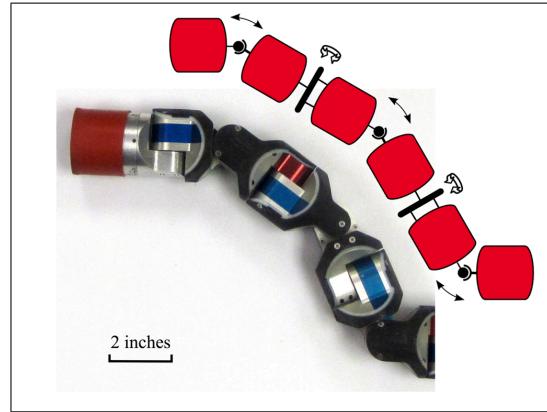


Fig. 1: The CMU modular snake robot is composed of 16 identical modules whose joints alternatively rotate in the dorsal and lateral plane of the robot.

This work focuses on modeling rolling motion which is used for efficiently climbing poles or moving the snake robot sideways. Rolling motion is classically generated by specifying parameters of the *gait equation*, which is associated with the drawbacks such as the lack of flexibility and adaptivity to the environment. We propose to generate controls directly based on the shape of a backbone curve in the workspace, thus making the specification of the control target more intuitive. We show that, when modeling rolling motions, a computational simplification can be achieved to avoid the unnecessary computations associated with the fitting method proposed in [1]. The simplified model allows the generation of smooth and fast control inputs. Using the rolling motion model presented in this paper, we derived controllers for two existing rolling gaits: *rolling arc* and *rolling helix*, now with a physically intuitive parameterization, simplifying the high-level control.

The shape modeling technique presented in this work allows the design of gaits from a new perspective: “composing shapes”. To demonstrate this idea, we devise a novel gait called *rolling hump* which forms a contour-fitting hump to negotiate obstacles. This type of gait differs from obstacle-aided locomotion [5] since the *rolling hump* gait enables a snake robot to roll over instead of moving between obstacles. A series of experiments were conducted to quantify the capability of the *rolling hump* gait to move over obstacles. We show that, in tele-operation mode, this *rolling hump* motion successfully moves across an unstructured terrain, occupied with rocks up to 5 inches in diameter. In simulation, we further show that tactile sensing can be easily incorporated

to automatically adjust the target shape of the rolling hump motion to the contour of obstacles.

The rest of the paper is structured as follows. In Sec. II, we discuss the previous work on modeling the motion of the snake robot. In Sec. III, we introduce a method for modeling the rolling motion and derive controllers for existing rolling gaits. In Sec. IV, the design of the rolling hump motion and results of experiments are given in detail, and the autonomous rolling hump is realized in simulation. Finally, in Sec. V we give conclusions and present the directions for the future work.

II. BACKGROUND

To represent the configuration of a redundant robot, Burdick *et al.* introduced a *backbone curve* [2]. The backbone curve abstracts the detailed mechanical implementation of the mechanism, thus keeps the focus on the motion design in the form of shape changes in the backbone curve. Using the backbone curve, designing motions is intuitive and straightforward because the geometric properties of the motion can be easily manipulated.

A backbone curve is usually described by a set of *parametric equations* that express the coordinates of the backbone curve as a function of a set of parameters. For example, Hatton *et al.* [6] modeled sidewinding gait as an elliptical helix and Gong *et al.* [7] used a conical helix to model the turning behavior of sidewinding. In these works, while intuition helps to design the backbone curve, a fitting algorithm is still needed for mapping the shape of backbone curve to the joint angles. Hatton *et al.* [3] proposed a fitting algorithm which is computationally expensive because the algorithm is aimed at directly fitting the positions of the robot joints onto a continuous backbone curve. The *Bellows model* [4] takes advantage of the high order information (curvature, torsion) contained in a backbone curve to facilitate the fitting process. Specifically, the Bellows model represents the shape of a backbone curve using the dorsal curvature, $\kappa_d(s)$, and lateral curvature, $\kappa_l(s)$. Those two curvature components are obtained by decomposing the curvature, $\kappa(s)$, onto the dorsal direction, $\vec{d}(s)$, and lateral direction, $\vec{l}(s)$:

$$\kappa_d(s) = \kappa(s) \cos(\phi(s)) \quad \text{dorsal} \quad (1)$$

$$\kappa_l(s) = \kappa(s) \sin(\phi(s)) \quad \text{lateral} \quad (2)$$

Here $\phi(s)$ is the rolling offset and can be found using Eqn. 3

$$\phi(s) = \phi_0 + \int_0^s \tau(s) ds. \quad (3)$$

where ϕ_0 is the initial angle offset and $\tau(s)$ is the torsion along the backbone curve. Using the Bellows model, rolling motion of the modular snake robot can be achieved by defining the initial angle offset $\phi_0(t)$ a function of time.

In the Bellows model, the two curvature components are naturally related to the curvature and the torsion in Frenet-Serret formula [8], and they can be interpreted as a continuous analogy to the dorsal and lateral joint angles of the modular snake robot.

III. ROLLING MOTION MODELING

The goal of motion modeling is to generate controls for a discrete mechanism to form a shape similar to the desired backbone curve. In this section, we first briefly introduce the fitting method proposed in [1], which computes the joint angles by integrating the dorsal and lateral curvatures of a backbone curve. Then, a method for reducing unnecessary computations is presented.

A. Curvature Integration Algorithm

Dorsal and lateral curvatures, $\kappa_d(s)$ and $\kappa_l(s)$, of a desired backbone curve were discussed in Sec. II. The corresponding joint angles can be computed by integrating each curvature components over the length of two modules (joints alternatively rotate in the dorsal and lateral plane). Formally, let m denote the length of one module and i denote the index of a joint. Then the i -th joint angle $\alpha(i)$ is computed by:

$$\begin{aligned} \alpha(i) &= \int_{(i-1)m}^{(i+1)m} \kappa_d(s) ds && \text{dorsal} \\ \alpha(i) &= \int_{(i-1)m}^{(i+1)m} \kappa_l(s) ds && \text{lateral} \end{aligned} \quad (4)$$

B. Computation Reduction

When modeled as bellows, the rolling motion of a snake robot can be achieved by making the initial offset angle, $\phi_0(t)$, a function of time. Without loss of generality, we simply set $\phi_0(t) = t$. Correspondingly, the dorsal and lateral curvatures, $\kappa_d(s, t)$ and $\kappa_l(s, t)$, and the joint angles $\alpha(i, t)$ are also time dependent. At each time step, the curvature integration is computed and the joint angles are updated. Though the curvature integration algorithm is considerably fast, it still slows down generating the control inputs. The following steps speed up the controller by avoiding the unnecessary computations.

First, we define torsion component as $T(s) = \int_0^s \tau(s) ds$. Then, substituting Eqn. 3 into Eqn. 1 results in:

$$\begin{aligned} \kappa_d(s, t) &= \kappa(s) \cos(\phi_0(t) + T(s)) \\ &= \kappa(s) (\cos(\phi_0(t)) \cos(T(s)) - \sin(\phi_0(t)) \sin(T(s))) \\ &= \cos(\phi_0(t)) \kappa(s) \cos(T(s)) - \sin(\phi_0(t)) \kappa(s) \sin(T(s)) \\ &= \cos(\phi_0(t)) \kappa_{\cos}(s) - \sin(\phi_0(t)) \kappa_{\sin}(s), \end{aligned} \quad (5)$$

where $\kappa_{\cos}(s) = \kappa(s) \cos(T(s))$, $\kappa_{\sin}(s) = \kappa(s) \sin(T(s))$. Note that $\kappa_{\cos}(s)$ and $\kappa_{\sin}(s)$ are both time invariant. We further introduce two variables, $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$, to make the derivation more compact:

$$\alpha_{\cos}(i) = \int_{(i-1)m}^{(i+1)m} \kappa_{\cos}(s) ds \quad (6)$$

$$\alpha_{\sin}(i) = \int_{(i-1)m}^{(i+1)m} \kappa_{\sin}(s) ds. \quad (7)$$

Note that $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$ are also time invariant and do not need to be recomputed at every time step. Plugging

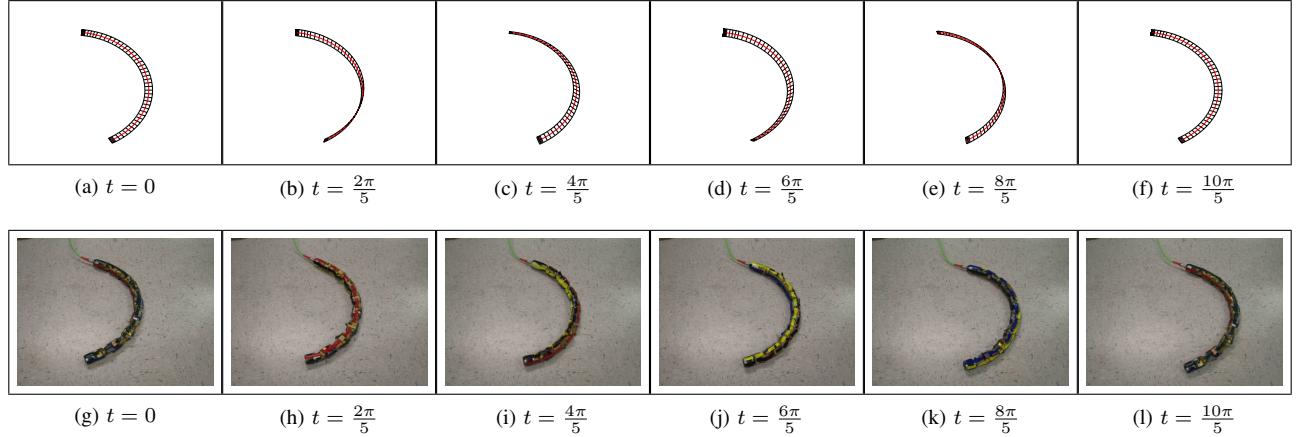


Fig. 2: The rolling arc represented with the Bellows model and the execution of the controller on the robot.

Eqn. 5 into Eqn. 4 results in:

$$\alpha(i, t) = \int_{(i-1)m}^{(i+1)m} \kappa_d(s, t) ds \quad \text{dorsal} \quad (8)$$

$$= \cos(\phi_0(t))\alpha_{\cos}(i) - \sin(\phi_0(t))\alpha_{\sin}(i).$$

The time dependent variable $\phi_0(t)$ is taken out of the integral since it does not depend on the arc length s . In the same manner, the lateral joint angles can be computed as:

$$\alpha(i, t) = \int_{(i-1)m}^{(i+1)m} \kappa_l(s, t) ds \quad \text{lateral} \quad (9)$$

$$= \sin(\phi_0(t))\alpha_{\cos}(i) + \cos(\phi_0(t))\alpha_{\sin}(i).$$

It is not necessary to recalculate the curvature integration in Eqn. 8 and 9 since the integral term is independent of time. As long as $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$ are determined, the time dependent joint angles $\alpha(i, t)$ can be readily computed. In addition to the computational benefits of this method, the resultant controller for the rolling motion is also smooth and well-parameterized.

C. Rolling Arc

We derived the controllers which allow the modular snake robot to execute one type of rolling motion, named *rolling arc*. As the name suggests, the robot forms an arc shape and it twists its body to generate a net displacement (see Fig. 2). We show that the coefficients of the derived controllers are physically meaningful.

The parametric equations of an arc parametrized by θ are given as:

$$\begin{aligned} x &= r \cos(\theta) \\ y &= r \sin(\theta) \\ z &= 0 \end{aligned} \quad (10)$$

where r is the radius. The shape of the rolling arc can be manipulated by adjusting the value of radius r . The curvature of the arc can be computed as $\kappa(s) = \frac{1}{r}$ and the torsion is $\tau(s) = 0$.

Using the method of curvature integration,

$$\kappa_{\cos}(s) = \kappa(s) \cos \left(\int_0^s \tau(s) ds \right) = \frac{1}{r} \quad (11)$$

$$\kappa_{\sin}(s) = \kappa(s) \sin \left(\int_0^s \tau(s) ds \right) = 0 \quad (12)$$

The, $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$ are:

$$\alpha_{\cos}(i) = \int_{(i-1)m}^{(i+1)m} \kappa_{\cos}(s) ds = 2 \frac{m}{r} \quad (13)$$

$$\alpha_{\sin}(i) = \int_{(i-1)m}^{(i+1)m} \kappa_{\sin}(s) ds = 0. \quad (14)$$

The joint trajectories for the dorsal joints can be expressed as:

$$\alpha(t, i) = \cos(t)\alpha_{\cos}(i) - \sin(t)\alpha_{\sin}(i) = 2 \frac{m}{r} \cos(t). \quad (15)$$

And the joint trajectories for the lateral joints are:

$$\alpha(t, i) = \sin(t)\alpha_{\cos}(i) + \cos(t)\alpha_{\sin}(i) = 2 \frac{m}{r} \sin(t). \quad (16)$$

the derived joint trajectories of the dorsal and lateral joints have the identical parametric form as the gait equation for rolling arc motion [12]. However, the coefficients in the controllers are now directly related to the shape of the robot. Fig. 2 shows the rolling arc motion represented with the Bellows model and executed on the modular snake robot.

D. Rolling Helix

The *rolling helix* is another form of rolling motion. Rolling helix can be used to crawl both inside and outside pipes (see Fig. 3). The parametric equations of a helix is,

$$\begin{aligned} x &= r \cos(\theta) \\ y &= r \sin(\theta) \\ z &= p\theta \end{aligned} \quad (17)$$

where $2\pi p$ denotes the pitch of the helix, and r is the radius which can be adjusted accordingly when a robot is climbing

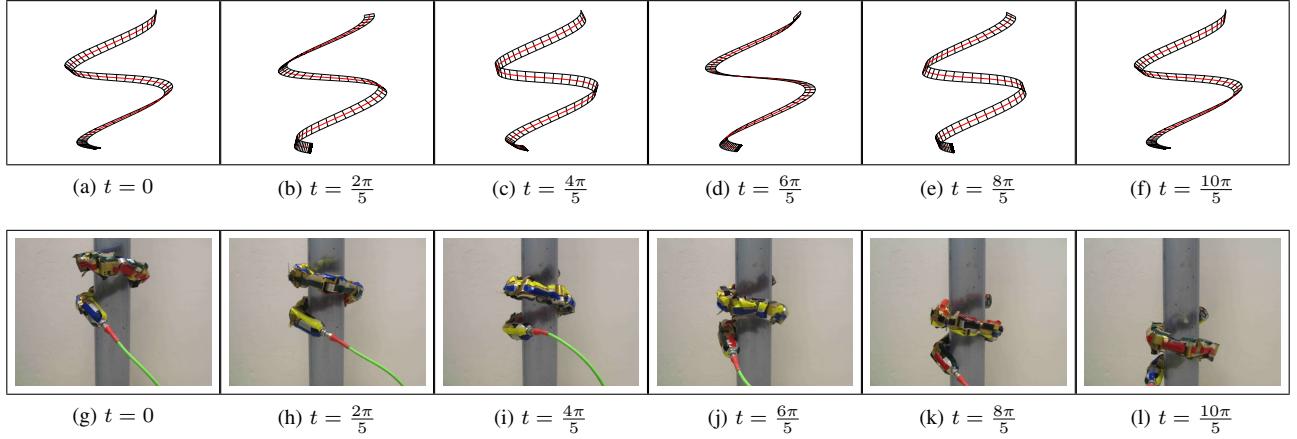


Fig. 3: The rolling helix represented with the Bellows model and the execution of the controller on the robot.

pipes of different sizes. The curvature of the helix is:

$$\kappa(s) = \frac{r}{r^2 + p^2} = \bar{\kappa} \quad (18)$$

And the torsion is:

$$\tau(s) = \frac{p}{r^2 + p^2} = \bar{\tau} \quad (19)$$

The curvature and the torsion of a helix are both constant and are denoted by $\bar{\kappa}$ and $\bar{\tau}$, respectively.

We use the curvature integration to derive controllers for the rolling helix motion. $\kappa_{\cos}(s)$ and $\kappa_{\sin}(s)$ are computed as:

$$\kappa_{\cos}(s) = \kappa(s) \cos \left(\int_0^s \tau(s) ds \right) = \bar{\kappa} \cos(\bar{\tau}s) \quad (20)$$

$$\kappa_{\sin}(s) = \kappa(s) \sin \left(\int_0^s \tau(s) ds \right) = \bar{\kappa} \sin(\bar{\tau}s) \quad (21)$$

Then, $\alpha_{\cos}(i)$ can be obtained using:

$$\begin{aligned} \alpha_{\cos}(i) &= \int_{(i-1)m}^{(i+1)m} \kappa_{\cos}(s) ds \\ &= \frac{\bar{\kappa}}{\bar{\tau}} (\sin(\bar{\tau}(i+1)m) - \sin(\bar{\tau}(i-1)m)) \\ &= \frac{2\bar{\kappa}}{\bar{\tau}} \sin(\bar{\tau}m) \cos(\bar{\tau}mi) \end{aligned} \quad (22)$$

And similarly $\alpha_{\sin}(i)$ is:

$$\alpha_{\sin}(i) = \int_{(i-1)m}^{(i+1)m} \kappa_{\sin}(s) ds = \frac{2\bar{\kappa}}{\bar{\tau}} \sin(\bar{\tau}m) \sin(\bar{\tau}mi)$$

Denoting $\frac{2\bar{\kappa}}{\bar{\tau}} \sin(\bar{\tau}m)$ as A , the joint trajectories of the dorsal joints are:

$$\begin{aligned} \alpha(t, i) &= A \cos(t) \cos(\bar{\tau}mi) - A \sin(t) \sin(\bar{\tau}mi) \\ &= A \cos(t + \bar{\tau}mi) \end{aligned} \quad (23)$$

And the trajectories of the lateral joints are:

$$\alpha(t, i) = A \sin(t + \bar{\tau}mi) \quad (24)$$

The rolling helix motion has an identical form as the gait equation in [12]. However, the coefficients of the controller

derived using the curvature integration method are parameterized in terms the pitch and the radius of the helix.

IV. ROLLING HUMP

The curvature integration allows designing motions by composing shapes. Here we present a novel gait, called *rolling hump*, which forms a contour-fitting hump to negotiate obstacles.

A. Arc with a Hump

Although the rolling arc can effectively move on flat grounds, this gait has difficulty in going over moderate obstacles due to the limited ground clearance. The ground clearance of the rolling arc can be improved by lifting up the portion of the body in collision with the obstacles. Hence, we add a hump on top of the arc for clearing obstacles. The parametric equations of the backbone curve of the rolling hump are given as:

$$\begin{aligned} x &= r \cos(\theta) \\ y &= r \sin(\theta) \\ z &= he^{-\frac{(\theta-\theta_0)^2}{\sigma^2}}, \end{aligned} \quad (25)$$

where h and σ respectively control the height and the width of the hump, and θ_0 determines the position of the hump along the backbone curve. We parameterize the hump as an exponential function since the exponential functions are concise, smooth and differentiable. Only three parameters are needed to fully specify the geometry of the hump.

B. Computing Curvature

Unlike the rolling arc or the rolling helix, it is difficult to derive a closed form equation for the rolling hump motion. Hence, we numerically compute the curvature, torsion, $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$. After numerically computing $\alpha_{\cos}(i)$ and $\alpha_{\sin}(i)$, the joint trajectories can be computed using Eqn. 4. Recomputation is only necessary when r, h, θ_0 or σ are modified.

On a standard laptop with a 2.4 GHz i5 processor and with non-optimized MATLAB code, the curvature integration

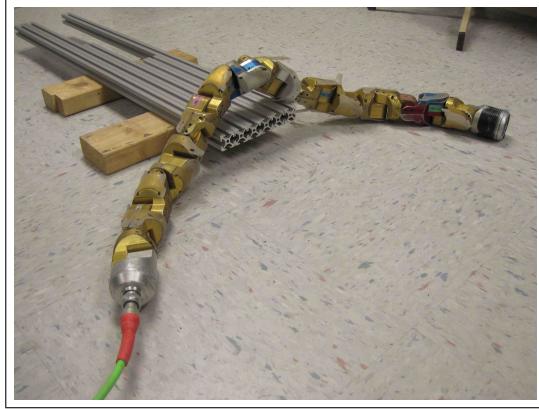


Fig. 4: The robot uses the rolling hump motion to step over a cuboid obstacle.

can be numerically computed at 20 Hz. After applying computation reduction, the control inputs can be computed at about 100 Hz.

C. Climbing over an Obstacle

In this section, we quantify the capability of the rolling hump motion to climb over obstacles. Obstacles are represented as cuboid blocks of different widths and heights (see Fig. 4). During the experiments, the user commands a desired h , σ and θ_0 to form a proper hump shape to help the robot climb over obstacles.

The experiments involved climbing over obstacles of different widths and heights (see Fig. 4). To account for the uncertainties, all the experiments were repeated 5 times. Fig. 5 shows the results of over 100 trials. The successful trials are denoted by green dots and the failures are marked with red dots. Note that many of the results are overlapped, and hence may not be noticeable.

The experiment results show that obstacles that are higher than 4.25 inches (the snake diameter is 2 inches) are beyond the capability of the rolling hump. The snake robot can easily climb over obstacles lower than 1 inch. For obstacles of heights between 1 to 4.25 inches, we fit a decision boundary to the experimental results using linear support vector machine (SVM). The cuboid obstacles of height h ($1 \text{ in} < h < 4.25 \text{ in}$) and width w that satisfy Eqn. 26 can be climbed over (see Fig. 5).

$$h + 0.3314w \leq 5.4760 \quad (26)$$

Note that length of the snake robot will affect the scale of h , δ , θ_0 , hence influence the obstacle height limit. Experiments will be conducted and described in future papers on this project.

D. Field Experiment

We also tested the rolling hump gait in a mocked-up outdoor environment (see Fig. 6). There are two pieces of rocks with approximately 5 inches in diameter, which is more than 2 times the diameter of the robot module. Compared to cuboid obstacles, the rocks have smooth surfaces and are

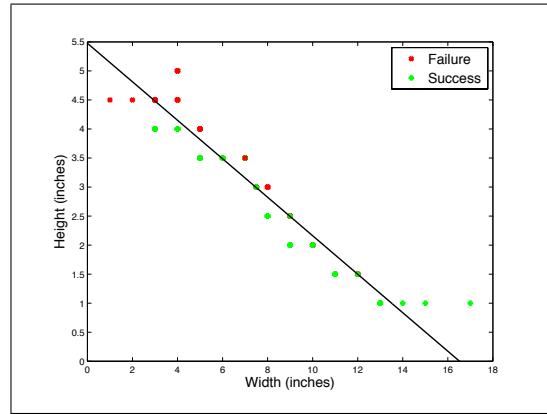


Fig. 5: Experiment results for climbing over obstacles of different sizes using the rolling hump motion. Every point denotes one trial with an obstacle having a particular width and height.



Fig. 6: A montage of the snake robot moving over two pieces of rock using the rolling hump motion.

relatively easy to be climbed over. During a bumpy field experiment full of rocks with various sizes, we noticed that the rolling hump gets stuck when the robot collided with multiple obstacles. This situation can be remedied if we can generate multiple humps.

E. Autonomous Rolling Hump

Here we present an autonomous rolling hump approach, to overcome the drawbacks mentioned in the previous section. Due to hardware limitation, autonomous rolling hump was realized in simulation and the contact detection was implemented in software by evenly mounting 8 virtual binary tactile sensors around the surface of each module. The portion of the snake body that is in contact with the obstacle automatically forms a hump to climb over obstacles. In Fig. 7, the red and green arrows indicate contacts with the obstacle. Then, the corresponding humps are represented by a Gaussian and shown in red and green. Adding the two Gaussian humps together will give the desired backbone curve (blue Gaussian hump). Afterwards, curvature integration method can be applied to generate joint angle trajectory for the snake robot.

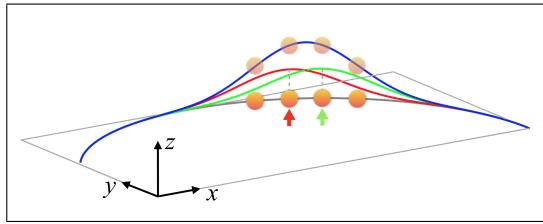


Fig. 7: The 3D backbone curve (blue) is constructed by adding the Gaussian kernels (green and red) at different modules (yellow circles).



Fig. 8: A snapshot from the simulation showing the snake robot climbing over a cubic obstacle using autonomous rolling hump gait.

The hump unit centered at each module is defined by a Gaussian kernel function:

$$g_i = h e^{-\frac{(\theta - \theta_{0i})^2}{\sigma^2}} \quad (27)$$

where i is index of the module, h and σ are predefined amplitude and variance of Gaussian kernel, corresponding to the height and the width of the hump unit; θ_{0i} is the position of module i along the body of snake robot. The parametric function of the backbone curve is given by:

$$\begin{aligned} x &= r \cos(\theta) \\ y &= r \sin(\theta) \\ z &= \sum_i \omega_i g_i \end{aligned} \quad (28)$$

Here ω_i is a weight coefficient that is defined as the number of the hump units added at the position of module i and can be updated at each time step using:

$$\omega_i^{t+1} = \omega_i^t + I_i^t - \varepsilon \quad (29)$$

where I_i is an contact indicator defined as:

$$I_i = \begin{cases} 1 & \text{module } i \text{ in contact with obstacles} \\ 0 & \text{module } i \text{ not in contact with obstacles} \end{cases}$$

and the constant ε acts as the gravity pulling the module down to the ground, which allows the snake robot to restore its original shape, i.e. a planar arc, after rolling over the obstacle. A larger ε indicates a faster recovery of the original shape but slower rising speed of the humps. Fig. 8 shows a snapshot from the simulation of the snake robot climbing over a cubic obstacle. The autonomous

rolling hump simulation can be viewed in the attached video.

V. CONCLUSIONS

This paper deals with modeling the rolling motion of a snake robot. There are three types of rolling motions, namely rolling arc, rolling helix and rolling hump. This work employs the Bellows model to represent the rolling motion. A computational reduction method was developed for curvature integration used in the Bellows model. The controllers derived using the proposed model have physically meaningful parameterizations which allow intuitive understanding of the snake robot's locomotion.

Additionally, using the concept of 'composing shapes', we designed the rolling hump gait which enables the snake robot to climb over obstacles and locomote on bumpy terrains. The simulation results show that if the snake robot has contact detection capability, the rolling hump gait can be automated. Efforts are being taken to develop real tactile sensors. Those sensors are designed to be mounted evenly around the surface of each module, in the same way we arrange the virtual sensors in the simulation. Hence, in the future, we can endow the snake robot with adaptivity in unstructured environments through contact sensing.

REFERENCES

- [1] H. Yamada and S. Hirose, "Approximations to continuous curves of active cord mechanism made of arc-shaped joints or double joints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 703–708.
- [2] J. W. Burdick, J. Radford, and G. S. Chirikjian, "A'sidewinding' locomotion gait for hyper-redundant robots," *Advanced Robotics*, vol. 9, no. 3, pp. 195–216, 1994.
- [3] R. L. Hatton and H. Choset, "Generating gaits for snake robots: annealed chain fitting and keyframe wave extraction," *Autonomous Robots*, vol. 28, no. 3, pp. 271–281, 2010.
- [4] H. Yamada and S. Hirose, "Study on the 3d shape of active cord mechanism," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2890–2895.
- [5] A. A. Transeth, R. I. Leine, C. Glocker, K. Y. Pettersen, and P. Liljeback, "Snake robot obstacle-aided locomotion: modeling, simulations, and experiments," *Robotics, IEEE Transactions on*, vol. 24, no. 1, pp. 88–104, 2008.
- [6] R. L. Hatton and H. Choset, "Sidewinding on slopes," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 691–696.
- [7] C. Gong, R. L. Hatton, and H. Choset, "Conical sidewinding," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4222–4227.
- [8] J. A. Thorpe, *Elementary topics in differential geometry*. Springer, 1979.
- [9] H. Mochiyama, E. Shimemura, and H. Kobayashi, "Direct kinematics of manipulators with hyper degrees of freedom and frenet-serret formula," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1653–1658.
- [10] H. Mochiyama and H. Kobayashi, "The shape jacobian of a manipulator with hyper degrees of freedom," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 4. IEEE, 1999, pp. 2837–2842.
- [11] C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, and H. Choset, "Design and architecture of the unified modular snake robot," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4347–4354.
- [12] 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.