Algorithmic Gait Synthesis for a Snake Robot

Gagan Khandate[1], Emily Hannigan[1], Maximilian Haas-Heger[1], Bing Song[1], Ji Yin[1] and Matei Ciocarlie[1]

I. Motivation

The development of snake robots is driven by the promise of capabilities, which traditional robotic platforms do not possess — their highly articulated nature as well as their long and slender bodies enable them to reach confined spaces. They are also of great interest for amphibious locomotion as a single robotic platform can negotiate various environments both on land and in the water. Motion patterns of snakes in nature have been extensively studied by researchers in order to equip snake-like robots with agile locomotion abilities. Mimicking of natural snake gaits has been demonstrated to be effective for locomotion in simple environments (such as flat surfaces devoid of obstacles) but this approach breaks down in more complicated settings and cannot equip snake robots with the agility of their biological counterparts.

Little work has been done on generating the gait patterns for robotic snakes completely algorithmically. We believe this would allow us to take the next step towards snake robots that can navigate general unstructured environments. We will show that efficient and effective gaits can be synthesized purely algorithmically without any prior knowledge of any locomotion strategies. We will compare three different methods - including a model-based and a model-free approach - for synthesizing motion strategies with respect to their performance for both land-based and aquatic locomotion.

II. Dynamic Model

We model the robot as a simple 2D kinematic chain consisting of N links of uniform mass connected by N-1 joints. We consider the snake robot dynamics as a discrete time problem with timestep dt. The state of our system at time t is given by $\mathbf{X}(t) = (x, y, q_0, \dots, q_{N-1}, \dot{x}, \dot{y}, \dot{q}_0, \dots, \dot{q}_{N-1})$ and consists of the Cartesian head position of the robot, the orientation of the links with respect to a global frame as well as their time derivatives. The snake is controlled by commanding the torques at the joints between links with actuator commands $\mathbf{u}(t) = [\tau_1, \dots, \tau_{N-1}]$. We setup the Newton-Euler equations for each link of the chain and integrate the resulting equations of motion using an explicit Runge-Kutta method of the 4th order. We obtain $\mathbf{X}(t + dt)$ and thus use this explicit time-stepping scheme for the simulation of the dynamics of the snake robot.

In order to investigate locomotion of the snake robot both on land and under water we introduce models of the forces acting on the robot in both environments. For water based locomotion we consider both low and moderate Reynolds number flow regimes. Low Re flows arise for small, slow swimmers such as tadpoles and viscous forces dominate. Larger water animals like water snakes encounter higher Re flows where drag forces and the added mass effect come into effect. We lump the interaction forces at the center of mass of the links in order to avoid integrating forces across each link. In the context of robot control this is a reasonable assumption.

Dry Friction describes the frictional forces a snake encounters on land. In nature snakes require greater frictional effects acting across their body than along its length for locomotion. They achieve this anisotropic friction through strategies such as weight redistribution as well as the microscopic structure of their scales. A snake robot must have similar frictional characteristics for meaningful locomotion.

We chose an anisotropic friction law derived from Coulomb friction. This friction law confines the contact force to the interior of a cone of elliptical cross-section with semimajor axes μ_n and μ_t, which correspond to the maximum friction forces in the normal (subscript n) and tangential (subscript t) directions respectively. Using the *Maximum Dissipation Principle* we can express the friction force $f = (f_n, f_t)^T$ in terms of the relative sliding velocity of the link center of mass v. Recall, that for locomotion we require $\mu_n > \mu_t$.

$$\begin{pmatrix} f_n \\ f_t \end{pmatrix} = \begin{pmatrix} -\mu_n \sin(\arctan(\frac{\mu_n v_n}{\mu_t v_t})) \\ -\mu_t \cos(\arctan(\frac{\mu_n v_n}{\mu_t v_t})) \end{pmatrix}$$

Viscous Friction dominates the forces acting on small swimming animals due to the viscosity of the water. These viscous forces are linearly dependent on the flow velocity with a scaling determined by the body shape. The flat bodies of many water based animals aid locomotion as viscous forces across the body are greater than along it.

$$\begin{pmatrix} f_n \\ f_t \end{pmatrix} = \begin{pmatrix} -\mu_n v_n \\ -\mu_t v_t \end{pmatrix}$$

Fluid Dynamics describe the interaction forces acting on larger animals in an environment such as swimming or diving. For robots similar in size to water snakes traveling at comparable speeds two effects dominate the interaction forces. Firstly, fluid dynamic drag acts on each link in normal and tangential directions. Futhermore, the density of the surrounding water contributes to the so called *added mass effect*.

[1]Department of Mechanical Engineering, Columbia University, New York, NY 10027, USA.
{gk2496, ejh2192, mkh2149, bing.song, ji.yin, matei.ciocarlie}@columbia.edu

Assuming the links individual to be cylinders of length l with an elliptical cross-section of height a and width b we can derive expressions for the fluid dynamic drag $(d_n, d_t)^T$. The additional inertial mass m_a due to the surrounding water is negligible in the tangential direction and therefore only the normal direction is considered. Typical values for these coefficients are $C_d \sim 1, C_f \sim 0.01$ and $C_a \sim 1$.

$$\begin{aligned} \begin{pmatrix} d_n \\ d_t \end{pmatrix} &= \begin{pmatrix} -\frac{1}{2}\rho C_d a l |v_n| v_n \\ -\frac{1}{2}\rho \pi C_f \frac{a+b}{4} l |v_t| v_t \end{pmatrix} \\ m_a &= \rho \pi C_a \frac{a^2}{4} l \end{aligned}$$

III. Planning Algorithms

Robotic locomotion requires the computation of control actions that move the robot along a trajectory from its starting position to a desired goal state. We employ three different algorithms to generate sequences of actions and compare the gaits exhibited by the resulting trajectories.

Sampling Based Planning We use a kinodynamic-RRT [1], which is an extension of the RRT pathplanning algorithm for nonholonomic systems that incorporates the dynamic constraints. The algorithm constructs a tree by sampling random states, finding the closest node and taking random actions until it finds a way to extend the tree towards the sampled state from this closest node. In order to move from an initial to a goal state we then follow a branch of the resulting tree that leads us towards that goal.

Model Predictive Control MPC uses a model of the dynamics in order to optimize the trajectory starting from its current position within a certain horizon. It thus requires a trajectory optimizer that can determine appropriate control action. We chose iLQG [2], an iterative method for nonlinear MPC based on the differential dynamic programming (DDP) algorithm. Once an appropriate sequence of actions has been found the robot applies the first action in the sequence and the process is repeated from the new state.

Deep RL We use Proximal Policy Optimization (PPO) [3] a model-free reinforcement learning algorithm to learn a neural network policy which determines the control action. In PPO, an actor-critic policy is trained to maximize cumulative reward of the resulting robot trajectory while avoiding network updates that cause the policy to diverge.

While kinodynamic-RRT has no notion of optimality both iLQG and PPO attempt to optimize the robot actions with respect to a cost (reward) function. In order to be able to make comparisons we chose to use an identical cost function for both, which consists of a weighted sum of the distance of the snake head from a goal and the norm of the commanded actuator torques.

IV. Results

In this section we present the motion synthesized with the aforementioned algorithms when given a goal to the left of a five link snake (see Figs. 1 - 3.). The vertical direction

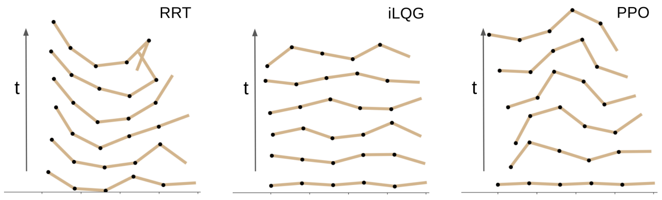


Fig. 1. Snake gaits for dry friction

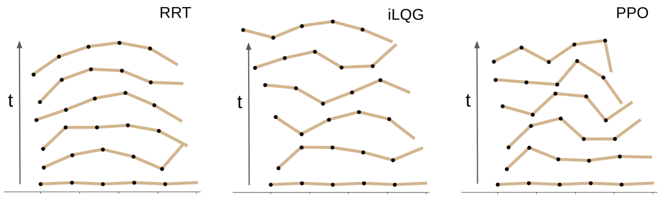


Fig. 2. Snake gaits for viscous friction

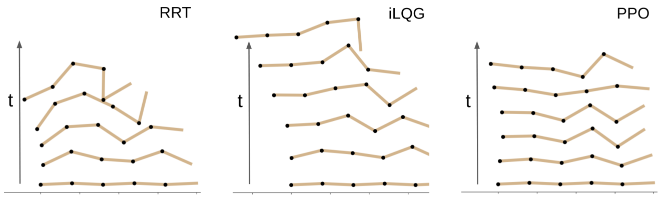
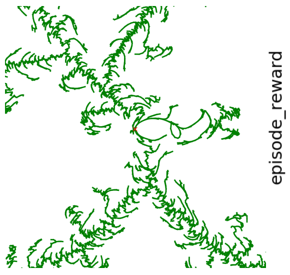


Fig. 3. Snake gaits for fluid dynamics



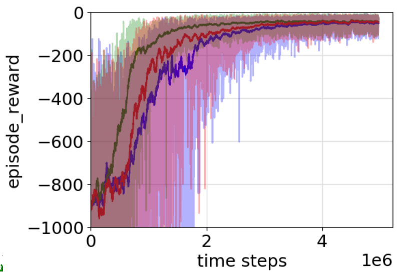


Fig. 4. Part of the tree generated by Kinodynamic-RRT and training curve for PPO policy and three different random initial seeds

signifies a progression in time. Part of an RRT tree can be seen in Fig. 4 as well as the training curve of the PPO policy for dry friction. Fig. 5 compares the performance of the three different trajectory generation approaches in terms of the the time and energy required to reach the desired goal. For a more thorough comparison we apply all three methods to a set of ten randomly generated goals sampled roughly in front of the snake. For simplicity we assume no model noise.

While we compare three different methods - only iLQG and PPO make an attempt to optimize the trajectory. This is reflected by the chaotic and inefficient gait derived from RRT. Hence the RRT trajectory should only be viewed as a highly suboptimal yet viable baseline trajectory. The gaits synthesized using iLQG and PPO, however, closely resemble gaits observed in nature. They are highly efficient, fast and give rise to great agility. For an example of such a motion pattern for an underwater snake robot synthesized using only iLQG with the above dynamic model see the animation at `https://roamlab.github.io/`.

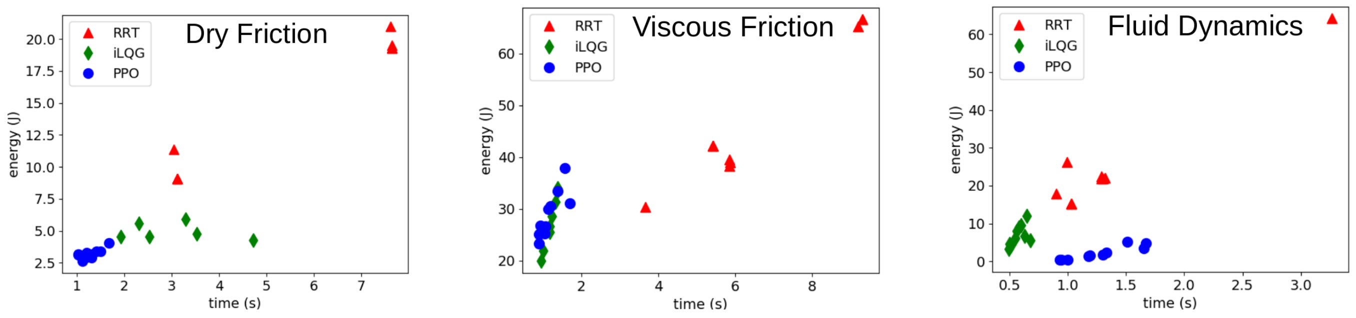


Fig. 5. Time and energy required to reach the randomly sampled goals

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[2] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347