2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Optimizing a Continuum Manipulator's Search Policy Through Model-Free Reinforcement Learning

Chase Frazelle[†], Jonathan Rogers, Ioannis Karamouzas, and Ian Walker

Abstract— **Continuum robots have long held a great potential for applications in inspection of remote, hard-to-reach environments. In future environments such as the Deep Space Gateway, remote deployment of robotic solutions will require a high level of autonomy due to communication delays and unavailability of human crews. In this work, we explore the application of policy optimization methods through Actor-Critic gradient descent in order to optimize a continuum manipulator's search method for an unknown object. We show that we can deploy a continuum robot without prior knowledge of a goal object location and converge to a policy that finds the goal and can be reused in future deployments. We also show that the method can be quickly extended for multiple Degrees-of-Freedom and that we can restrict the policy with virtual and physical obstacles. These two scenarios are highlighted using a simulation environment with 15 and 135 unique states, respectively.**

I. Introduction

Continuum robots have a number of characteristics that distinguish them from their rigid-link counterparts. One such characteristic is the source of actuation often being located away from the core structure of the manipulator [1], transferring actuation through either tendons, pneumatic pressure, or synthetic muscles that rely on external power sources to drive local locomotion. This absence of the actuation source in the body of the manipulator makes continuum robots excellent candidates for exploration and manipulation in restricted environments [2]–[5].

The "continuum" element of this class of robots draws many parallels to the biological world, ranging from vertebrates with continuum appendages such elephant trunks to invertebrates whose entire body is made up of compliant, soft material capable of extreme dexterity and manipulation [6]–[8]. Along with the complexity of these continuum structures found in nature, continuum manipulators carry with them the potential for hyper-redundancy [9] and an infinite number of degrees-of-freedom (DoF) due to their structure being deformable at any point along their backbone. This phenomenon manifests in complex kinematics [10]–[12] and dynamics [13], [14].

There have been a number of explorations into motion planning methods for continuum robots, which often can help avoid the challenges that redundancy and complex configuration spaces bring to solving problems like inverse kinematics. Configuration space exploration and exploration through sample-based methods such as Rapidly-exploring Random Tree (RRT) methods have been deployed for maneuvering continuum manipulators through various task spaces [15]–[19], even adaptively updating the knowledge and trajectory during execution [20]. Once configuration space driven methods are implemented to locate points of interest in the continuum manipulator's task space, kinematics driven methods such as visual servoing [21] can be used to refine the manipulator's interaction with the environment.

Reinforcement learning (RL) provides an attractive alternative where a robot/agent learns to take actions that maximize its cumulative reward through interactions with the environment. Many early success stories exist, from training robots to compete in RoboCup competitions to enabling robots to acquire advanced manipulation skills [22]–[25]. More recently with the rise of deep learning, impressive results have been obtained on physical articulated robots for a wide range of motor and manipulation tasks [26]–[29]. In the continuum robotics domain, a number of works have explored the use of reinforcement learning to improve motion planning methods and improve upon various control schemes [30], [31]. Most relevant to this work is the use of a Soft Actor-Critic (SAC) method in [32] to optimize a continuum manipulators policy for reaching a point in space with the robot's end-effector. In that work, the authors employ a Random Network Distillation (RND) method to train a series of neural networks and then use the SAC algorithm to maximize the return of the policy designed to capture a known object in space. They report a dependency on sparse reward and a need for the RND method in order to promote adequate exploration. Of these works, many rely on a need of *a priori* knowledge of the environment or of a specific goal state.

Here, we investigate the feasibility of using a RL framework to train policies on continuum manipulators. In particular, this paper explores the application of reinforcement learning for continuum manipulators with the aim of automating continuum robots being used for inspection. An actor-critic policy gradient method is applied with the purpose of locating a goal object and creating a global policy that determines how the robot behaves when deployed with the task of observing various points of interest in its task space. The method can be expanded for encompassing extra DoF, as well as be used to develop policies simultaneously for different points of interest within the robot's workspace.

[†] To whom all correspondence should be addressed (cfrazel@clemson.edu).

C. Frazelle and I. Walker are with the Dept. of Electrical & Computer Engineering, Clemson University, Clemson, SC, 29634.

J. Rogers is with NASA Johnson Space Center, Houston, TX, 77058.

I. Karamouzas is with the School of Computing, Clemson University, Clemson, SC, 29634.

This research was supported by the U.S. NSF under grants IIS-0844954, IIS-0904116, and IIS-1718075 and by a NASA Space Technology Research Fellowship, contract 80NSSC17K0173.

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

We summarize the algorithm and problem specific adjustments for our solution in Section II. Section III explores both a simple and extended example of our method in action and results, as well as the impact of learning rates on solution convergence. Discussion and conclusions are presented in Sections IV and V, respectively.

II. Policy Optimization

A model-free policy optimization method is proposed in this work in order to shape a continuum robot's search policy for objects of interest. This method allows us to develop a global policy quickly while removing potential problems that redundancy and aliased states cause for deterministic methods.

A. Formulation

We formulate the control problem of a continuum robot section as a discounted Markov Decision Process defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space available to the robot, $P : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward that the environment emits on each transition, and $\gamma \in [0, 1]$ is the discount factor. Regarding $\mathcal{S}$, for a single continuum section, we define two DoF: κ, the curvature of the continuum section, and ϕ, the plane of bending for each section. Traditionally, many continuum robots also have the ability to extend and retract along their backbone; we assume fixed backbone length in this work. We create a discrete set of all states by describing each DoF as spanning $n_\kappa \in \mathbb{N}$ and $n_\phi \in \mathbb{N}$ discrete values, distributed evenly over a defined range for each value. Given i number of sections, the total number of states is $(n_\kappa \times n_\phi)^i$, giving $\mathcal{S}_{n_\kappa \times n_\phi \times \ldots}$. In this work, we limit our actions to a single step transition along one DoF at a time in order to simplify the list of available actions and create simple connections between states. For each DoF, we increment the DoF up ($a = +1$) or down ($a = -1$) along their discrete set of values, or remain at the current value ($a = 0$). Given i sections as before, the total number of actions is $3 \times i \times n_{DoF}$, where n_{DoF} is the number of DoF available to the section.

As is common among reinforcement learning methods, we empirically define a series of rewards and penalties associated with actions taken by the robot in order to shape the final policy. The reward function is primarily designed to promote actions that lead to a state that can view the goal object. Equally, it penalizes actions that leave such states in order to return to states that are not able to see the goal. We design the largest penalty to occur when a chosen action leads to an invalid state or invalid state transition. Examples of this would be trying to bend a continuum section beyond the physical limits of the robot or attempting to transition to a neighboring state that is blocked by a physical object. For all other actions, the algorithm issues a step penalty in order to encourage reaching the goal state in a finite number of steps. Thus, the reward structure, R_s^a we employ is as follows:

$$R_s^a = \begin{cases} -100 & \text{, if } s' \text{ is invalid} \\ 5 & \text{, new state sees goal} \\ -5 & \text{, leaves state that sees goal} \\ -0.05 & \text{, general movement cost} \end{cases} \qquad (1)$$

B. Actor-Critic Policy Optimization

We assume a model-free reinforcement learning setup, where the robot does not have direct knowledge about the transition function, P, and reward, r, and can only experience them through interacting with the environment. In particular, at a given time step t, the robot observes the current state $\mathbf{s}_t \in \mathcal{S}$ and samples an action $\mathbf{a}_t \in \mathcal{A}$ from a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. This leads to a new state $\mathbf{s}_{t+1}$ that rewards the robot with r_t. Our goal is to solve for the policy that optimizes the robot's expected sum of discounted rewards.

Policy gradient methods allow us to maximize the expected cumulative reward by directly searching in the policy space, reducing the amount of memory needed to store quality of states and actions information as with Value Iteration and Q-learning methods, while they are also the preferred class of methods for learning controls in continuous state-action spaces. Here, we consider parameterized policies $\pi_\theta(\mathbf{a}|\mathbf{s})$ and hence the objective of the learning process is to find the parameters θ that maximize

$$J(\theta) = \mathbb{E}_{\mathcal{M}, \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t | \pi_\theta] \qquad (2)$$

Given the above objective function, $J(\theta)$, we adjust θ through gradient ascent where the gradient of the expected reward can be determined according to the policy gradient theorem [33]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|s_t)} \left[\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) \,|\, \mathbf{s}_t \right] \qquad (3)$$

where $Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{a \sim \pi(\cdot|s_t), \mathcal{M}} \left[\sum_{l=0} r_{t+l} \right]$ denotes the action-value Q function. To reduce the variance of the policy gradient estimate and increase stability, we consider an actor-critic policy gradient framework [33]. In particular, we replace the estimate of the Q-value provided by the cumulative reward in Eq. 3 with a function approximator (critic) which is learned in tandem with the policy (actor). The critic evaluates the quality of the policy for a current set of policy parameters. The actor then shapes the policy parameters in response to the output from the critic. It is important to note that the vector value θ is of the same dimension as our state-action feature vector, which we define when implementing the solution in Section III.

In the problem we are exploring in this work, our continuum manipulator is capable of assuming a discrete number of states and to perform a discrete set of actions in order to transition between these states. As such, we have chosen to use the Softmax policy [33] which states that the probability of an action is proportional to the exponential of a linear combination of features $\Phi(s, a)$:

$$\pi_\theta(s,a) \quad \propto \quad e^{\Phi(s,a)^T\theta} \tag{4}$$

Using this policy, our learned policy parameters θ are defined to be coefficients for each of our features.

Given the well-defined nature of the Softmax policy , the relevant score function is:

$$\nabla_\theta \log \pi_\theta(s,a) = \Phi(s,a) - \mathbb{E}[\Phi(s,\cdot)] \tag{5}$$

where $\mathbb{E}[\Phi(s,\cdot)]$ is the expected feature vector at state s.

Regarding the critic that evaluates our policy, we consider a linear approximation of the value function, $Q^\pi(s,a)$, by linearly combining the features via a weight vector w:

$$Q_w(s,a) = \Phi(s,a)^T w \approx Q^{\pi_\theta}(s,a) \tag{6}$$

The critic is updated at each time step using linear Temporal Difference (TD) learning that adjusts the parameters w of the Q-function based on the TD error, $delta$, and the state-action features. We refer the reader to Algorithm 1 for an overview of our actor-critic framework for learning an optimal policy. Here, each learning iteration generates sample(s) from the current policy, uses these samples to update the critic function, and updates the policy parameters based on the critic and the gradient of the objective function. Learning rates α and β adjust step size for learned parameters, and the discount factor γ determines the impact of future rewards.

Algorithm 1 Actor-Critic Policy Gradient

```
function QAC
    Initialize $s, \theta$
    Sample a $\sim \pi_\theta$
    for each step do
        Sample reward $r = R_s^a$, get transition $s' \sim P_s^a$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
        $w \leftarrow w + \beta \delta \Phi(s, a)$
        $a \leftarrow a', s \leftarrow s'$
    end for
end function
```

III. Simulation Validation

We verify the functionality of the algorithm using a simulation model of the Tendril robot [34] placed in the Gazebo physics simulator environment [35]. The Tendril is a continuum robot comprising of a backbone made using a carbon fiber tube, plastic spacers for the routing actuating tendons, and an actuator package that pulls on the tendons to create bending. The physical Tendril is long and thin, with either 2 or 3 independent continuum sections. The Tendril is simulated in Gazebo using a series of small, rigid linkages connected in series that approximate the continuum shape of the actual Tendril. The end-effector of the simulated manipulator is fitted with a camera that is oriented in line with the tendril's backbone, much like an endoscope. In order to simplify the image processing task and focus on the policy optimization, the Tendril is placed in an empty simulation world with a single object that represents the goal we wish to locate with the robot's camera. An example of the empty world scenario and the viewpoint of the simulated Tendril is given in Figure 1, where the view of the Tendril is seen in the lower left corner.

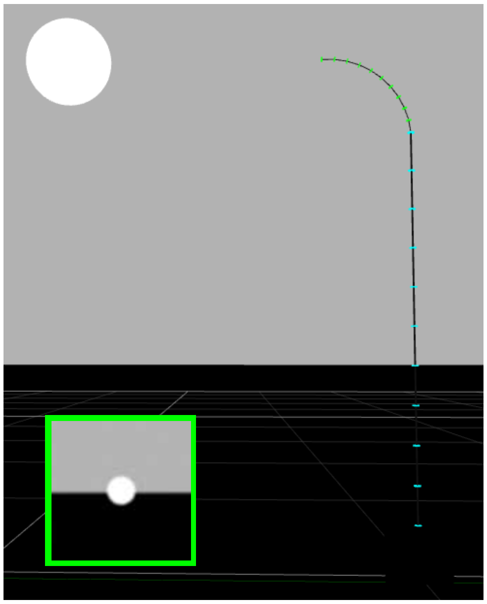

Fig. 1. Empty World Simulation with Tendril Robot

A. Defining Features

Necessary to the implementation of our formulation is the definition of the feature vector. As this solution is designed for inspection purposes, our state feature vector $\Phi(s)$ is defined based on feedback from a camera and low-level image processing:

$$\Phi(s) = [I_{avg}, S_{obj}] \tag{7}$$

where I_{avg} is the average intensity of the image and S_{obj} indicates if a goal object is present and the size of the object relative to the camera frame size. Both feature values are normalized to the closed range $[0, 1]$, and the size of goal object is saturated to a threshold equivalent to occupying 1/10th the area of the camera view.

In order to simplify the execution of the method around edge states (i.e. the boundaries of our state space), we preserve the same action set for all states and instead apply an action-based feature, and corresponding penalty in our reward function, for state-action pairs that attempt to assume an invalid state. The feature, represented as B in equation 8, exists as a binary feature: 1 when the chosen action crosses a boundary (such as exceeding bending limits), and 0 when the chosen action leads to another valid state.

$$\Phi(s,a) = [\Delta I_{avg}, \Delta S_{obj}, B] \tag{8}$$

The remaining features in our state-action feature vector $(\Delta I_{avg}, \Delta S_{obj})$ are the changes in the state features I_{avg} and S_{obj}, respectively, between state s and the state s' reached upon performing action a.

In analyzing the algorithm given this feature definition, it can be seen that the nature of our state-action feature vector will always produce a non-zero probability of staying in a arbitrary state at any given time. In other words, $\Phi(s,a) = [0,0,0]$ when the action is $a = 0$ across all DoF, giving $e^{[0,0,0]^T\theta} = 1$. Therefore, we modify the policy for these actions in each state as:

$$\pi_\theta = S_{obj} e^{\Phi(s,a)^T\theta} \tag{9}$$

This modification to the policy removes the probability of choosing actions that stay in a state for any configuration that does not see the goal object and scales the probability of staying in a goal state according to how well the state "sees" the goal, as designated by S_{obj}.

B. Planar Task Space Exploration

In this first experiment, we start with a two-section continuum manipulator capable of independent planar bending in each section (i.e. n_{DoF} = 1). For clarity of visualizing the states, we indirectly provide curvature values (κ) as bending angles, which can then be converted to curvature for a fixed length backbone using $\kappa = \frac{\text{bending angle}}{\text{arc-length}}$. We allow the proximal section three bending values: bending angle of zero (straight backbone), and ±90°(i.e. bending left and right at 90°). Separately, we allow the distal section five values: straight, and bending angles of ±90°, ±180°. Therefore, the total number of possible states is 15. Examples of physical meaning for these states can be seen depicted in Figure 2. In evaluating the ability to locate an object of interest, we placed the singular goal object 0.8m left of the base of the robot and 1m vertically up from the base, which is conveyed by the blue orb in the upper left corner of each state image in Figure 2. Finally, for this experiment, we set the learning and discount rates to: $\alpha = 0.1$, $\beta = 0.3$, and $\gamma = 0.95$, which we experimentally found to work well for solution convergence.

We ran the algorithm 10 times while initialized at each of the possible configurations for a total of 150 executions. Each execution was allowed to run for 1500 iterations with the learning rates given above. The average of the policies obtained from each of the 150 runs is described in Table I, where the numbers given per action per state are the percent chance that the action will be taken when in that state. The actions listed in the table (Up/Down/Left/Right) refer to the transitions seen in the visual interpretation of the final policy in Figure 2. For each depicted state, the arrows flowing from the state represent the possible action transitions and are shaded according to the likelihood of that action being taken and transitioning to a neighboring state. All action transitions appearing as grey have a probability of either zero or near zero (<0.5%) of being chosen. The three states capable of seeing the goal are highlighted in the figure (States 5, 9, and 12), and are the only states that contain action transitions indicating a probability of staying in the present state.

As can be seen in the results of our simplified example, in the states where the goal object is well seen (States 9 and 12), we see a greater chance of staying in those states. In states neighboring the goal states (i.e. one action step), we also observe a markedly high percent chance (>75%) of taking the action that get us directly to a goal state. In states more than a step away, we see nearly uniform distribution among the valid actions, which is to be expected in an aliased state that does not provide much feedback to the system. Also, as designed in the feature set, in all edge states, any actions that lead to states beyond the limitations of the robot converge to zero or near zero percent chance of begin chosen. Overall, this is the expected optimal policy.

C. Spatial Task Space Exploration

We extend the above example by adding two additional DoF: direction of bending for the proximal section (ϕ_{prox}) and for the distal section (ϕ_{dis}), giving $n_{DoF} = 2$. The number of states quickly extends beyond the amount that can be reported here in detail. Instead, we report the total number of states, location of the goal, and the convergence to a stable policy. To start, we define our state set. In this experiment, we allow ϕ_{prox} and ϕ_{dis} to have 3 values: 0°, 120°, and 240°. We keep the same range of bending angles for the proximal and distal sections as the planar experiment. Given this, our total number of states is: $3{\cdot}3{\cdot}3{\cdot}5 = 135$. We can see a visual expression of these states in Figure 3, where we have also placed an example goal object at $[x,y,z] = [-1, 0.5, 1.25]$. We use the same learning rate α and discount value γ from before. We modify the learning rate β to be 0.2, which we found slightly improved performance on our hardware.

Given that our state space in this example is too large to reproduce visually here, we instead track the convergence of our policy parameters θ to a stable set of values. We ran the optimization algorithm 5 times from randomly selected starting states. Figure 4 depicts the average change in the three θ values over 2000 iterations of the policy optimization algorithm and includes the standard deviation of the 5 trials as shaded regions.

As can be seen, values θ_1 and θ_3 settle around approximately 800 iterations. The value of θ_2 increases slightly after this point, but generally begins to plateau enough to consider it a sufficient condition to exit the learning process. In practicality, we can design exit conditions (such as no change in policy for x iterations) to exit the learning process.

From our knowledge of the features that describe the state of our robot, we can draw conclusions from the relative magnitude and sign of the three θ values and their impact on our policy. It can be seen that any action that crosses a boundary (B=1), simulated or physical, will have a large negative component in the exponent, giving a probability approaching zero of that action being chosen. Even in the event that a goal state is on the other side of the boundary, the magnitude of the boundary associated parameter is generally higher than that of the goal. This is in part due to the penalty associated

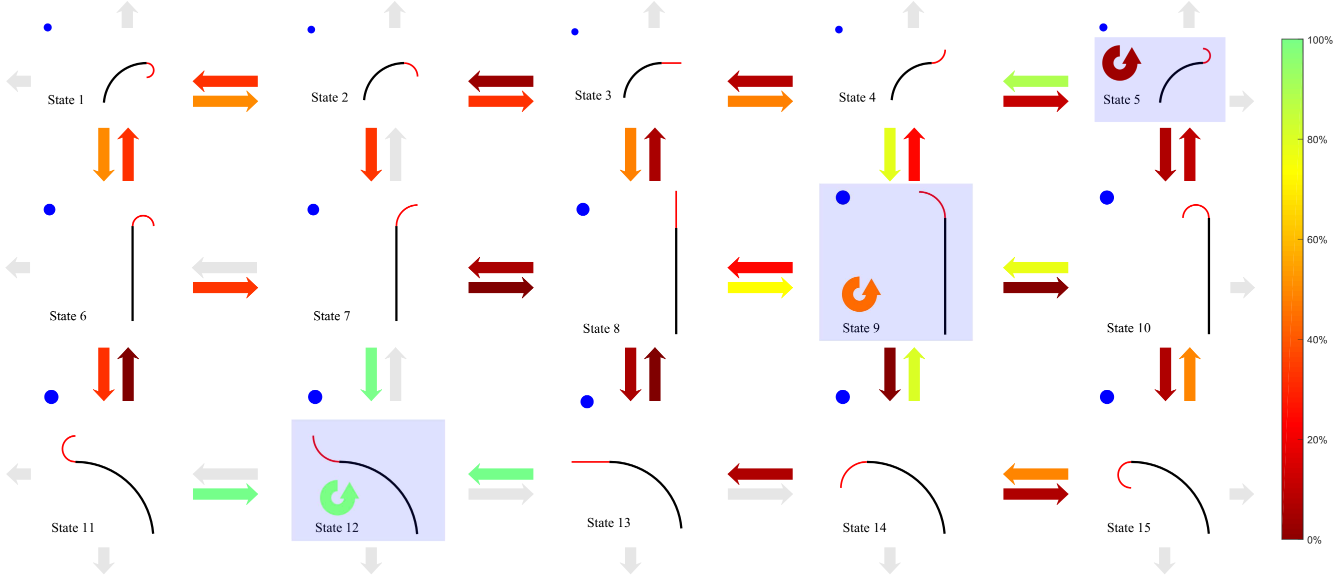

Fig. 2. Optimized Search Policy for Planar Two-section Continuum Manipulator: Colors correspond to probability of state-action transition occurring. Grey transitions indicate probability of zero.

TABLE I

STATE ACTION PROBABILITIES ACCORDING TO POLICY FOR PLANAR SEARCH

Action	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Up	0.1	0.0	0.0	0.0	0.0	33.1	0.3	9.0	25.8	13.3	0.7	0.3	0.8	80.0	49.9
Stay	0.0	0.0	0.0	0.0	2.1	0.0	0.0	0.0	21.9	0.0	0.0	49.6	0.0	0.0	0.0
Down	49.9	33.7	47.5	77.0	8.5	33.2	98.8	9.0	2.3	9.6	0.0	0.0	0.0	0.0	0.1
Left	0.1	33.2	4.9	9.7	87.3	0.0	0.3	9.2	25.7	77.2	0.0	0.3	98.8	10.1	49.9
Stay	0.0	0.0	0.0	0.0	2.1	0.0	0.0	0.0	21.9	0.0	0.0	49.6	0.0	0.0	0.0
Right	49.9	33.1	47.6	13.2	0.0	33.7	0.7	72.7	2.3	0.0	99.3	0.3	0.4	9.9	0.1

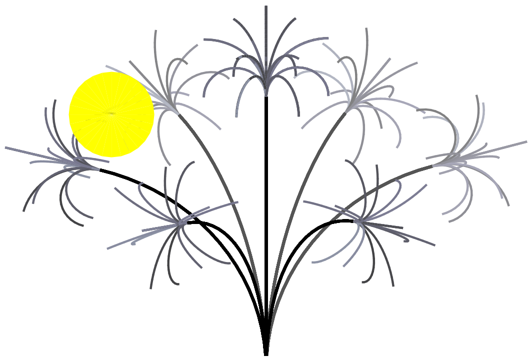

Fig. 3. Task Space of Tendril in Open space

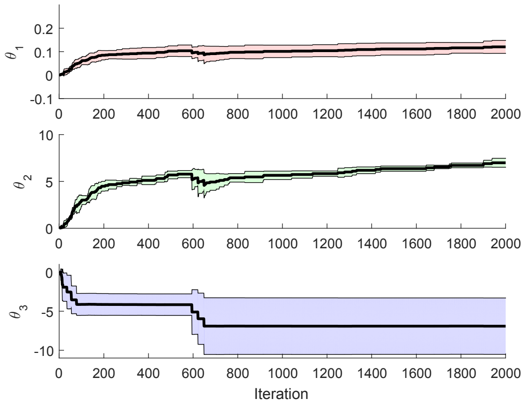

Fig. 4. Convergence of Policy Parameters for 4 DoF (Solid lines denotes the average and shaded regions the standard deviation of policy parameter values over 5 trials.)

with crossing a boundary being significantly higher than the reward for reaching the goal.

In evaluating values θ_1 and θ_2, we can see a positive association with both the average intensity of the image (I_{avg}) and the feature indicating the size of the goal object (S_{obj}). Clearly, seeing the goal is more impactful on our policy as indicated by the difference in magnitude of the parameters. However, because our goal is a bright object in a dark environment, there is still a positive association between increased light intensity and reward for finding the goal. Seeing this behavior, we can potentially draw parallels to the policy slightly favoring an increase in brightness.

D. Impact of Learning Rates

Following our observations with the convergence of policy parameters in the previous experiment, we conducted an empirical study of the impact of varying the learning rates, α and β, on the solution. For each learning rate, we adjusted the values independently and averaged 10 samples at each of the selected test values. For varying α, these values were: 0.001, 0.01, 0.1, 0.5, and 1.0. For β, we tested β values: 0.002, 0.02, 0.2, 0.3, 0.6, and 1.0. When varying α, we set $\beta = 0.2$, and when varying β, $\alpha = 0.1$. Figures 5 and 6 show the result of varying α and β, respectively on all three of the policy parameters while using the same experimental setup as the previous experiment.

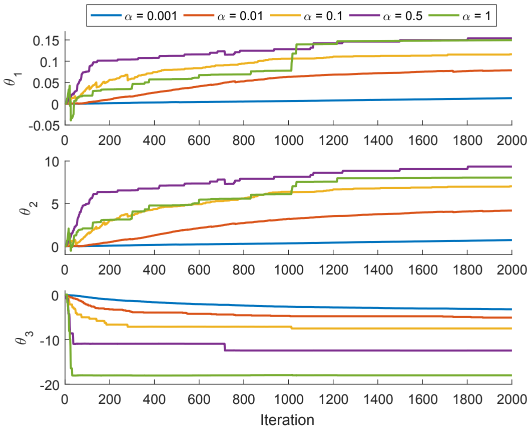

Fig. 5. Impact of learning rate α on solution convergence ($\beta = 0.2$)

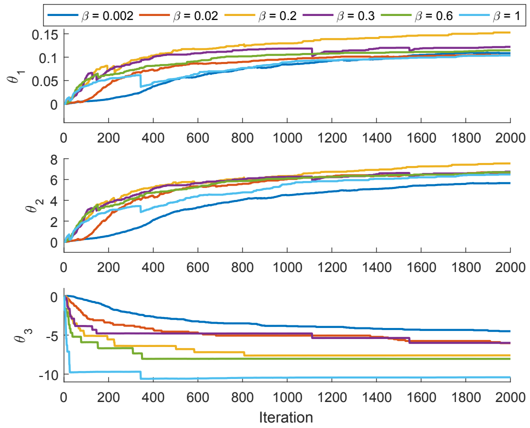

Fig. 6. Impact of learning rate β on solution convergence ($\alpha = 0.1$)

As can be seen in Figure 5, we generally see an increase in the rate of convergence as α increases, with some instability when $\alpha = 1$. With respect to the α value used in our experiments, $\alpha = 0.1$, we see that this choice of α shares desirable characteristics with both the smaller and larger α values. At α=0.1, the solution has a relatively smooth convergence, similar to the small α values, while having a faster rate of convergence like the higher values.

In observing the impact of β on our solution, it is clear that the value of β does not an impact solution convergence with the same distinction when observing policy parameters θ_1 and θ_2. However, in observing parameter θ_3, we see a trend similar to that of α, in that higher β values cause faster, less smooth convergence, and lower values have smooth curves, but are slower to converge. The one value that breaks this trend for this example is $\beta = 0.3$, which acts closer to the small values of β than values of a similar magnitude.

IV. Discussion and Future Work

A. Discussion

As we expand the DoF and range of motion, we quickly arrive at a scenario where the state-space contains a multitude of local states that are able to "see" the goal object. The algorithm presented here does not guarantee that the robot will arrive at exactly the "best state" that has the closest view of the object. However, the method can arrive to one of the local "best states" distributed throughout the state-space.

Indeed, when also taking into account our modification in Eq. 9, we can see an example of the policy settling in "good enough" states in our simple scenario. States 5, 9, and 12 are candidates for acceptable states by simply seeing the goal object. In purely quantitative reasoning, state 12 is the "best" goal state in that it has the best view of the goal ($S_{obj} = 1$), followed by state 9 ($S_{obj} = 0.36$), and then state 5 ($S_{obj} = 0.05$). The uniform random policy in aliased states prevents the system from settling in, or oscillating between, two non-goal states, and the remaining states clearly have an eventual path to either state 12 or 9. Even in state 5, in part receiving help from our modification, the policy has greater chance of moving to state 4 and then to state 9 over staying in 5. However, there is not an overwhelming likelihood of the system leaving state 9 in order to settle at state 12.

In order to address scenarios in which the continuum robot is given a higher number of states, we could able to add additional features and reward values that could encourage the robot to converge to states that have a "better" view of the goal. For example, if the planar robot has multiple, neighboring states in which the goal is visible, we could add a feature and related reward based on the distance of the object from the center of the camera. This could drive the robot to take actions that align the object in the middle of the camera for a better view.

In presenting this material, we simplify the image processing method of recognizing our goal object. In practical application, we can substitute our goal related features with those corresponding to the location of an actual feature of interest. Examples could be the results from a template matching algorithm that returns confidence values and locations, among other details.

Another design choice was the use of configuration space over actuation space. By using the configuration space, we have a meaningful interpretation of the system states that can be applicable to a variety of continuum manipulators. In converting to actuation space, we can rely on closed-loop controllers adapted to individual continuum robots to convert values such as curvature and orientation to physical values such as tendon lengths or pneumatic pressures.

As a function of using a simulation model, we make the assumption in this work that our state space and actions are well within the defined configuration space of our system. We also assume that transitions between neighboring states are guaranteed and deterministic. In practice, and subsequently the scope of future application, we will have to rely on control methods to achieve state transitions, and relax the guarantee of expected state transition. The need for reliable control methods in spatial motion is still a topic of research in continuum robotics [7], but it will not be necessary to require methods dependent on complex dynamic models to perform state transitions for a set of static states.

B. Future Work

The simulation model in Gazebo enables us to quickly test multiple scenarios and extend the size of the robot's state-space. Future work will explore applying this learning algorithm to a physical continuum robot in real-time, as well as exploring offline training in simulation with online execution of the learned policy. The Tendril robot was originally designed for deployment on, and inspection of, the International Space Station. The application of Gazebo and ROS can allow us to develop and test life-like scenarios aboard the ISS and in collaboration with Robonaut [36].

Another expansion of this work will be to switch to a more continuous state-space as well as a continuous action space. We would then explore the refinement of a Gaussian distribution based policy. As we move to continuous action spaces, we plan to test high-capacity function approximators such as neural networks to represent the policy and value function where raw pixel data can be used as state inputs. This will require a more robust policy optimization approach such as the Proximal Policy Optimization [37] that uses a modified objective to the MDP problem to estimate the expectation of the current policy. We also want to see how maximum entropy RL frameworks such as the recent Soft Actor-Critic model [38] can extend to our domain, as they are known to be sample efficient and more robust to hyper-parameter tuning which will be needed when training neural network-based Gaussian policies.

V. Conclusion

We successfully implement an actor-critic policy optimization method in order to improve the search method of a continuum manipulator used for inspection. Our scenario involves a continuum manipulator deployed to locate objects of interest without prior knowledge of the object's location. The implemented algorithm allows the robot to converge to a desirable state with a view of the object while learning a reusable search policy for future deployments. We provide detailed results of a simple example and examine the global implications of a more complex example with a high dimensional state-space. Future work will convert the solution to a continuous state and action space and see implementation on a physical robot.

References

[1] I. Walker, "Continuous backbone "continuum" robot manipulators: A review," *ISRN Robotics*, vol. 2013, no. 1, pp. 1–19, Jul. 2013.

[2] J. Burgner-Kars, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015.

[3] R. Buckingham, "Snake arm robots," *Ind. Robot: An Int. Jour.*, vol. 29, no. 3, pp. 242–245, Mar. 2002.

[4] J. Mehling, M. Diftler, M. Chu, and M. Valvo, "A minimally invasive tendril robot for in-space inspection," in *Proc Biorobotics Conference*, Pisa, Italy, 2006, pp. 690–695.

[5] J. C. Crusan, R. M. Smith, D. A. Craig, J. M. Caram, J. Guidi, M. Gates, J. M. Krezel, and N. B. Herrmann, "Deep space gateway concept: Extending human presence into cislunar space," in *2018 IEEE Aerospace Conference*. IEEE, 2018, pp. 1–10.

[6] W. Kier and K. Smith, "Tongues, tentacles, and trunks: The biomechanics of movement in muscular hydrostats," *Zoological Journal of the Linnean Society*, vol. 83, no. 4, pp. 307–324, 1985.

[7] T. G. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018.

[8] D. Trivedi, C. Rahn, W. Kier, and I. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 2, pp. 99–117, Jun. 2008.

[9] G. Chirikjian and J. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 343–354, Jun. 1994.

[10] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–57, Feb. 2006.

[11] J. Burdick, "On the inverse kinematics of redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Scottsdale, AZ, 1989, pp. 264–270.

[12] I. Godage, D. Branson, E. Guglielmino, G. MedranoCerda, and D. Caldwell, "Shape function-based kinematics and dynamics for variable-length continuum robotic arms," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 452–457.

[13] H. Mochiyama and T. Suzuki, "Kinematics and dynamics of a cable-like hyper-flexible manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, 2003, pp. 3672–3677.

[14] I. Gravagne, C. D. Rahn, and I. D. Walker, "Large deflection dynamics and control for planar continuum robots," *IEEE Trans. Mecha.*, vol. 8, no. 2, pp. 299–307, Jun. 2003.

[15] K. Wu, L. Wu, and H. Ren, "Motion planning of continuum tubular robots based on centerlines extracted from statistical atlas," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5512–5517.

[16] Z. Hawks, C. Frazelle, K. E. Green, and I. D. Walker, "Motion planning for a continuum robotic mobile lamp: Defining and navigating the configuration space," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2559–2566.

[17] A. Kuntz, A. W. Mahoney, N. E. Peckman, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Motion planning for continuum reconfigurable incisionless surgical parallel robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6463–6469.

[18] A. Ataka, P. Qi, H. Liu, and K. Althoefer, "Real-time planner for multi-segment continuum manipulator in dynamic environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4080–4085.

[19] A. Kuntz, M. Fu, and R. Alterovitz, "Planning high-quality motions for concentric tube robots in point clouds via parallel sampling and optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2205–2212.

[20] J. Xiao and R. Vatcha, "Real-time adaptive motion planning for a continuum manipulator," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Taipei, Taiwan, 2010, pp. 5919–5926.

[21] J. D. Greer, T. K. Morimoto, A. M. Okamura, and E. W. Hawkes, "A soft, steerable continuum robot that grows via tip extension," *Soft robotics*, vol. 6, no. 1, pp. 95–108, 2019.

[22] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[23] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 2619–2624.

[24] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *International Symposium on Experimental Robotic*, 2003, pp. 363–372.

[25] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[26] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," in *Robotics: Science and Systems*, 2018.

[27] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Robotics: Science and Systems*, 2018.

[28] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," in *Conference on Robot Learning*, 2019.

[29] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 634–641.

[30] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, "Open loop position control of soft continuum arm using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5133–5139.

[31] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2018.

[32] C. Yang, J. Yang, X. Wang, and B. Liang, "Control of space flexible manipulator using soft actor-critic and random network distillation," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 3019–3024.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[34] M. Tonapi, I. S. Godage, A. M. Vijaykumar, and I. Walker, "A novel continuum robotic cable aimed at applications in space," *Advanced Robotics*, vol. 29, no. 13, pp. 861–875, Jul. 2015.

[35] O. S. R. Foundation. (2020, February) Gazebo: Robot simulation made easy. Http://gazebosim.org/.

[36] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter, *et al.*, "Robonaut 2-the first humanoid robot in space," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2178–2183.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.