Artificial Control of Soft Surgical Robots with Gaussian Processes

David Eng
dkeng@stanford.edu

1. Introduction

The use of robotic devices for surgical interventions has gained significant traction in medical facilities in the last 25 years. Conventional interventions, such as the da Vinci® Surgical System, use rigid materials to fabricate these robotic systems, which are easily represented as rigid members connected at discrete joints [1].

However, rigid robots like the da Vinci system present severe limitations, especially in a surgical setting. Though their rigid frames allow for precise and predictable control of the system, these frames also introduce significant risks in surgeries involving highly sensitive and vital organs, such as the lung or heart.

Since they bend and twist with high curvatures, soft robots offer the desired properties of a more versatile tool. In surgeries such as cardiac ablation for atrial fibrillation or diagnostic biopsies of the lung, the use of soft robots instead of rigid ones can dramatically reduce the risk of inadvertent damage to impacted organs.

Despite their exciting surgical implications, soft robots lack widespread adoption due to a lack of accurate control schemes capable of governing their movements. Existing models struggle with the high dimensionality of the state space through which these robots must traverse, as well as the amount of noise introduced by the randomness of the surrounding organs [1].

To improve the viability of soft robotics in medical facilities, we propose an online reinforcement learning algorithm to control a soft surgical robot catheter in the noisy environment of the human body.

2. Problem Statement

We formulate two tasks relevant to the soft surgical robot catheter as reinforcement learning (RL) problems:

- Bringing the tip of the robot catheter into a goal region
- Drawing simple shapes, such as circles or rectangles, with the tip of the robot catheter

Importantly, the second task has immediate surgical implications. Often, in surgeries like cardiac ablation to treat cardiac arrhythmia, the surgeon must "draw shapes" around valves or "draw lines" across portions of the inside of the heart with high precision [2].

3. Theoretical Approach

3.1. Markov Decision Process

As mentioned, we formulate the problem of controlling the soft robot as a set of RL problems. We are therefore required to define a Markov decision process (MDP), which consists of states and action spaces, a reward function, and state transition dynamics.

The states in our model $\phi(\mathbf{x})$ are the activations of the robot's five motors.

$$\phi(\mathbf{x}) = \begin{bmatrix} q_1 & q_2 & \dots & q_5 \end{bmatrix}^\top$$

Four of the five motors contract and release tension in wires to bend the outer sheath of the robot catheter, and the last motor controls movement along an insertion axis.

We define a finite and relatively small number of actions by specifying a set of activations for the robot's motors. The activations remain fixed for the duration of a constant time interval until a new action is chosen for the next interval.

$$\mathbf{a} = \begin{bmatrix} dq_1 & dq_2 & \dots & dq_5 \end{bmatrix}^\top$$

The reward is defined as 0 for goal states and negative L_2 distance from the current coordinate position of the tip of the robot catheter to the goal state otherwise. These values encourage the controller to find policies that bring the robot to the goal state as quickly as possible. In addition, in order to encourage smoothness in the robot's movements, we subtract an energy penalty term from these rewards. This term is proportional to the total energy expended by all motors during each action interval.

3.2. Gaussian Process TD Learning

Since the state space of our problem is very large, some form of function approximation must be used to represent the value estimator. Temporal difference methods, such as TD(λ) and LSTD(λ), converge with linear function approximation. However, the number of basis functions required to approximate the state space grows exponentially with the size of the state space.

Engel et al. present a Bayesian approach to the problem of value function estimation in continuous spaces [3]. Their approach defines a probabilistic generative model for the value function and introduces an online sparsification to allow for a computational tractable update.

3.2.1 Generative Model

The paper proposes the following generative model for the sequence of rewards corresponding to the trajectory $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$:

$$R(\mathbf{x}_i, \mathbf{x}_{i+1}) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i)$$

This expression can be vectorized to the following form:

$$R_{t-1} = H_t V_t + N_t$$

where

$$H_t = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & -\gamma \end{bmatrix}$$

$$R_t = \begin{bmatrix} R(\mathbf{x}_1) \\ \vdots \\ R(\mathbf{x}_t) \end{bmatrix} \quad V_t = \begin{bmatrix} V(\mathbf{x}_1) \\ \vdots \\ V(\mathbf{x}_t) \end{bmatrix} \quad N_t = \begin{bmatrix} N(\mathbf{x}_1) \\ \vdots \\ N(\mathbf{x}_t) \end{bmatrix}$$

To validate the correctness of the vectorization, consider the value of the inner product of the ith row of H_t with V_t, a result consistent with our original generative model:

$$\begin{aligned} H_t^{(i)} V_t + N_t^{(i)} &= V_t^{(i)} - \gamma V_t^{(i+1)} + N_t^{(i)} \\ &= V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i) \\ &= R(\mathbf{x}_i, \mathbf{x}_{i+1}) \end{aligned}$$

We then represent the joint (N_t, V_t) as a bivariate Gaussian distribution as follows:

$$\begin{pmatrix} N_t \\ V_t \end{pmatrix} \sim \mathcal{N}\left\{ \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{bmatrix} \mathbf{\Sigma}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_t \end{bmatrix} \right\}$$

where

$$\mathbf{\Sigma}_t(\mathbf{x}) = \begin{bmatrix} \sigma_0^2 & 0 & \ldots & 0 \\ 0 & \sigma_0^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_0^2 \end{bmatrix}$$

$$\mathbf{K}_t(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \ldots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}$$

Given the vectorized expression for R_{t-1}, we also represent the joint $(R_{t-1}, V(\mathbf{x}))$ as a bivariate Gaussian distribution:

$$\begin{pmatrix} R_{t-1} \\ V(\mathbf{x}) \end{pmatrix} \sim \mathcal{N}\left\{ \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{bmatrix} \mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t & \mathbf{H}_t \mathbf{k}_t(\mathbf{x}) \\ \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top & k(\mathbf{x}, \mathbf{x}) \end{bmatrix} \right\}$$

This distribution summarizes the following expectations, variances, and covariances:

$$\begin{aligned} \mathrm{E}[R_{t-1}] &= \mathrm{E}[V(\mathbf{x})] = 0 \\ \mathrm{Var}(R_{t-1}) &= \mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t \\ \mathrm{Var}(V(\mathbf{x})) &= k(\mathbf{x}, \mathbf{x}) \\ \mathrm{Cov}(R_{t-1}, V(\mathbf{x})) &= \mathbf{H}_t \mathbf{k}_t(\mathbf{x}) \\ \mathrm{Cov}(V(\mathbf{x}), R_{t-1}) &= \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top \end{aligned}$$

These values are sufficient to compute the posterior for the normal distribution $(V(\mathbf{x})|R_{t-1} = \mathbf{r}_{t-1})$:

$$(V(\mathbf{x})|R_{t-1} = \mathbf{r}_{t-1}) \sim \mathcal{N}\{a_t(\mathbf{x}), b_t(\mathbf{x})\}$$

where

$$\begin{aligned} a_t(\mathbf{x}) &= \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{r}_{t-1} \\ b_t(\mathbf{x}) &= k(\mathbf{x}, \mathbf{x}) - \mathbf{H}_t \mathbf{k}_t(\mathbf{x}) \mathbf{Q}_t \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top \\ \mathbf{Q}_t &= (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t)^{-1} \end{aligned}$$

This result follows from the definition of the posterior for the normal distribution (detailed in the appendix).

3.2.2 Online Sparsification

In order to render the algorithm practical, we must reduce the computational burden associated with the computation of the matrix $\mathbf{Q}_t$. Engel et al. leverage an online sparsification method for kernel algorithms, previously proposed in the context of Support Vector Regression [4].

We maintain a dictionary of samples that suffice to approximate any training sample (in feature space) up to some predefined accuracy threshold. Our method starts with an empty dictionary $\mathcal{D}_0 = \{\}$ and observes the sequence of states $\mathbf{x}_1, \mathbf{x}_2, \ldots$ one state at a time, admitting $\mathbf{x}_t$ into the dictionary only if its feature representation $\phi(\mathbf{x}_t)$ cannot be approximated sufficiently well by combining the representations of the states already in $\mathcal{D}_{t-1}$. This approximation performs the following minimization task:

$$\min_{\mathbf{a}} \left\{ \mathbf{a}^\top \tilde{\mathbf{K}}_{t-1} \mathbf{a} - 2\mathbf{a}^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) + k_{tt} \right\}$$

where $\tilde{\mathbf{K}}_{t-1}$ is the kernel matrix of the dictionary states at time $t-1$. The solution to this minimization task is $\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ and substituting it back into the expression gives the minimum squared error incurred by the approximation:

$$\begin{aligned} \varepsilon_t &= k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \mathbf{a}_t \\ &= k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \end{aligned}$$

Parameter	$\mathcal{D}_t = \mathcal{D}_{t-1}$	$\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$
$\tilde{\mathbf{K}}_t^{-1}$	$\frac{1}{\delta_t}\begin{bmatrix}\delta_t\tilde{\mathbf{K}}_{t-1}^{-1} + \hat{\mathbf{a}}_t\hat{\mathbf{a}}_t^\top & -\hat{\mathbf{a}}_t \\ -\hat{\mathbf{a}}_t^\top & 1\end{bmatrix}$	Same as $\mathcal{D}_t = \mathcal{D}_{t-1}$
$\mathbf{a}_t$	$\tilde{\mathbf{K}}_{t-1}^{-1}\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$	$\begin{bmatrix}0 & 0 & \dots & 1\end{bmatrix}^\top$
$\tilde{\mathbf{c}}_t$	$\frac{\gamma^2\sigma_{t-1}^2}{s_{t-1}}\tilde{\mathbf{c}}_{t-1} + \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1}\Delta\tilde{\mathbf{k}}_t$	$\frac{\gamma^2\sigma_{t-1}^2}{s_{t-1}}\begin{bmatrix}\tilde{\mathbf{c}}_{t-1}\\ 0\end{bmatrix} + \tilde{\mathbf{h}}_t - \begin{bmatrix}\tilde{\mathbf{C}}_{t-1}\Delta\tilde{\mathbf{k}}_t\\ 0\end{bmatrix}$
$\tilde{\mathbf{d}}_t$	$\frac{\gamma^2\sigma_{t-1}^2}{s_{t-1}}d_{t-1} + r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top\tilde{\boldsymbol{\alpha}}_{t-1}$	Same as $\mathcal{D}_t = \mathcal{D}_{t-1}$
s_t	$\sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta\tilde{\mathbf{k}}_t^\top\left(\tilde{\mathbf{c}}_t + \frac{\gamma\sigma^2}{s_{t_1}}\tilde{\mathbf{c}}_{t-1}\right) - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}}$	$\sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \Delta\tilde{\mathbf{k}}_t^\top\tilde{\mathbf{C}}_{t-1}\Delta\tilde{\mathbf{k}}_t + \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}}\tilde{\mathbf{c}}_{t-1}^\top\Delta\tilde{\mathbf{k}}_t - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}}$
$\tilde{\boldsymbol{\alpha}}_t$	$\tilde{\boldsymbol{\alpha}}_{t-1} + \frac{\tilde{\mathbf{c}}_t}{s_t}d_t$	$\begin{bmatrix}\tilde{\boldsymbol{\alpha}}_{t-1}\\ 0\end{bmatrix} + \frac{\tilde{\mathbf{c}}_t}{s_t}d_t$
$\tilde{\mathbf{C}}_t$	$\tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t}\tilde{\mathbf{c}}_t\tilde{\mathbf{c}}_t^\top$	$\begin{bmatrix}\tilde{\mathbf{C}}_{t-1} & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix} + \frac{1}{s_t}\tilde{\mathbf{c}}_t\tilde{\mathbf{c}}_t^\top$

Table 1: Parameter updates for the Online GPTD Algorithm

If $\varepsilon_t > \mathcal{V}$, where $\mathcal{V}$ is an accuracy threshold parameter, we add $\mathbf{x}_t$ to the dictionary, and we set $\mathbf{a}_t = (0, \dots, 1)^\top$ since $\tilde{\mathbf{x}}_t$ is exactly represented by itself. If $\varepsilon_t \leq \mathcal{V}$, the dictionary remains unchanged.

3.2.3 Online GPTD Algorithm

Combining the ideas of the original generative model and online sparsification method, we derive the following estimates for the value function and precision for a state $\mathbf{x}$:

$$\hat{v}_t(\mathbf{x}) = \tilde{\mathbf{k}}_t(\mathbf{x})^\top\tilde{\boldsymbol{\alpha}}_t$$
$$p_t(\mathbf{x}) = k_{xx} - \tilde{\mathbf{k}}_t(\mathbf{x})^\top\tilde{\mathbf{C}}_t\tilde{\mathbf{k}}_t(\mathbf{x})$$

where

$$\tilde{\mathbf{H}}_t = \mathbf{H}_t\mathbf{A}_t$$
$$\tilde{\mathbf{Q}}_t = (\tilde{\mathbf{H}}_t\tilde{\mathbf{K}}_t\tilde{\mathbf{H}}_t^\top + \tilde{\boldsymbol{\Sigma}}_t)^{-1}$$
$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{H}}_t^\top\tilde{\mathbf{Q}}_t\tilde{\mathbf{r}}_{t-1}$$
$$\tilde{\mathbf{C}}_t = \tilde{\mathbf{H}}_t^\top\tilde{\mathbf{Q}}_t\tilde{\mathbf{H}}_t$$

We summarize the online updates for each of the parameters required for the model in cases $\mathcal{D}_t = \mathcal{D}_{t-1}$ and $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$ in Table 1.

4. Implementation

4.1. Simulation

In order to accelerate the learning process, we designed a system to simulate the robot's movements through space. This simulation comprised of three layers of abstraction:

- `robot_catheter.py` provides the underlying kinematic model described by Jones, et al [5]. This layer maintains the true position and orientation of the tip of the robot and the true activations of the motors.
- `robot_catheter_interface.py` provides the interface between the underlying kinematic model and the robot controls. This layer introduces the noise present in the actual system.
- `idm.py` exposes controls to manipulate the robot. This layer maintains beliefs of the position and orientation of the tip of the robot and beliefs of the activations of the motors, then follows a policy to reach the goal state.

We specify a trajectory of coordinate positions for the robot to visit. For the task of bringing the tip of the robot catheter into a goal region, the robot visits the single point

along the trajectory. For the task of drawing simple shapes with the tip of the robot catheter, the robot visits points along the trajectory in sequence. The simulation ends when the robot has visited all of the points along its trajectory.

We specify circular trajectories with various lengths and radii. A circular trajectory describes a sequence of points equidistant to a common point. For example, a circular trajectory with length 6 and radius 12 represents a regular hexagon with edge length 12.

4.2. Evaluation

We evaluate our policy based on two criteria, accuracy and computational tractability.

4.2.1 Accuracy

We measure the accuracy of our policy by evaluating the ratio R between $D^\pi(t)$, the length of the path traveled by the tip of the robot catheter to visit all points along the trajectory t according to the policy π, and $D^*(t)$, the length of the shortest path between the points.

For example, consider a state space $\mathbf{x} \in \mathbb{R}^2$ and a trajectory from $[0, 0]$ to $[1, 0]$. Suppose the tip of the robot catheter follows the following trajectory:

$$t = \frac{x_2}{x_1} = \begin{cases} 1 & 0 \leq x_1 \leq 0.5 \\ -1 & 0.5 < x_1 \leq 1 \end{cases}$$

Since the optimal path between the points is a line of length 1 but the robot's trajectory is a path of length $\sqrt{2}$:

$$R = \frac{D^\pi(t)}{D^*(t)} = \frac{\sqrt{2}}{1} = \sqrt{2} = 1.41$$

We aim to find a policy that minimizes R.

4.2.2 Computational Tractability

We evaluate the computational tractability of our policy by measuring the time required to compute the optimal action over the duration of the trajectory. Since our robot operates at 100 Hz, to encourage optimal performance of the robot, we aim to find a policy that requires an average of less than 10 ms to compute the optimal action.

5. Results

5.1. Single-Point Trajectories

We first attempted to learn a policy for a trajectory with only a single point, in which the online GPTD algorithm learns to estimate a value function over the motor activations. In these experiments, our online GPTD algorithm updated a Gaussian kernel parametrized by $\gamma_k = 1$ with a discount factor $\gamma = 0.25$. It seeded this kernel by sampling each action 50 times before proceeding to the goal state.

Goal	$\bar{n}$	s_n	$\bar{D^\pi}$	s_{D^π}	$\bar{R}$
(32, 4, 4)	72.9	77.3	7.55	5.18	1.62
(32, 4, -4)	68.4	62.4	8.18	7.78	1.76
(32, 6, 2)	118.6	56.5	11.5	3.35	2.16
(32, 6, -2)	125.9	66.4	12.2	3.90	2.30

Table 2: Summary statistics of single-point trajectories. n represents the number of iterations required to reach the goal region and D^π represents the length of the path traveled by the tip of the robot catheter.

Table 2 provides summary statistics from 200 trajectories (50 trajectories per goal state) with outliers removed. Figure 1 shows visualizations of a representative subset of these trajectories. Each row of the figure shows three separate trajectories (blue) of the tip of the robot catheter en route from the coordinate position (32, 0, 0) to some goal region (green) in the absence of noise. The four goal regions are centered at the coordinate positions (32, 4, 4) for (a) to (c), (32, 4, -4) for (d) to (f), (32, 6, 2) for (g) to (i), and (32, 6, -2) for (j) to (l) with a margin of 1.

5.1.1 Accuracy

Our summary statistics reveal that the robot catheter achieves reasonable accuracy, producing values of $\bar{R}$ ranging from 1 to 3. We note that the robot catheter achieves much better accuracy for some goal states, namely (32, 4, 4) and (32, 4, -4), than it does for other goal states, namely (32, 6, 2) and (32, 6, -2). From our visualizations, we infer that this difference results from a bias in our initial value function approximation that favors the consistent movement in the y and z directions. It makes sense that as long as the robot continues to receive improving rewards, the online GPTD algorithm remains content with this initial approximation. However, once the robot starts to receive worsening rewards, the online GPTD algorithm produces an on-the-fly course correction, as evidenced by the abrupt corners in the trajectories in examples (g) to (l) of the figure.

Together, these results indicate that our current approach to seed the kernel by sampling each action 50 times introduces a bias that causes our initial policy to favor actions that result in movements in certain directions.

Our visualizations reveal another weakness of our current approach. The trajectories in examples (c) and (f) of the figure indicate that the online GPTD algorithm struggles to find the optimal policy when its trajectory misses the surface of the goal region. In this situation, the robot

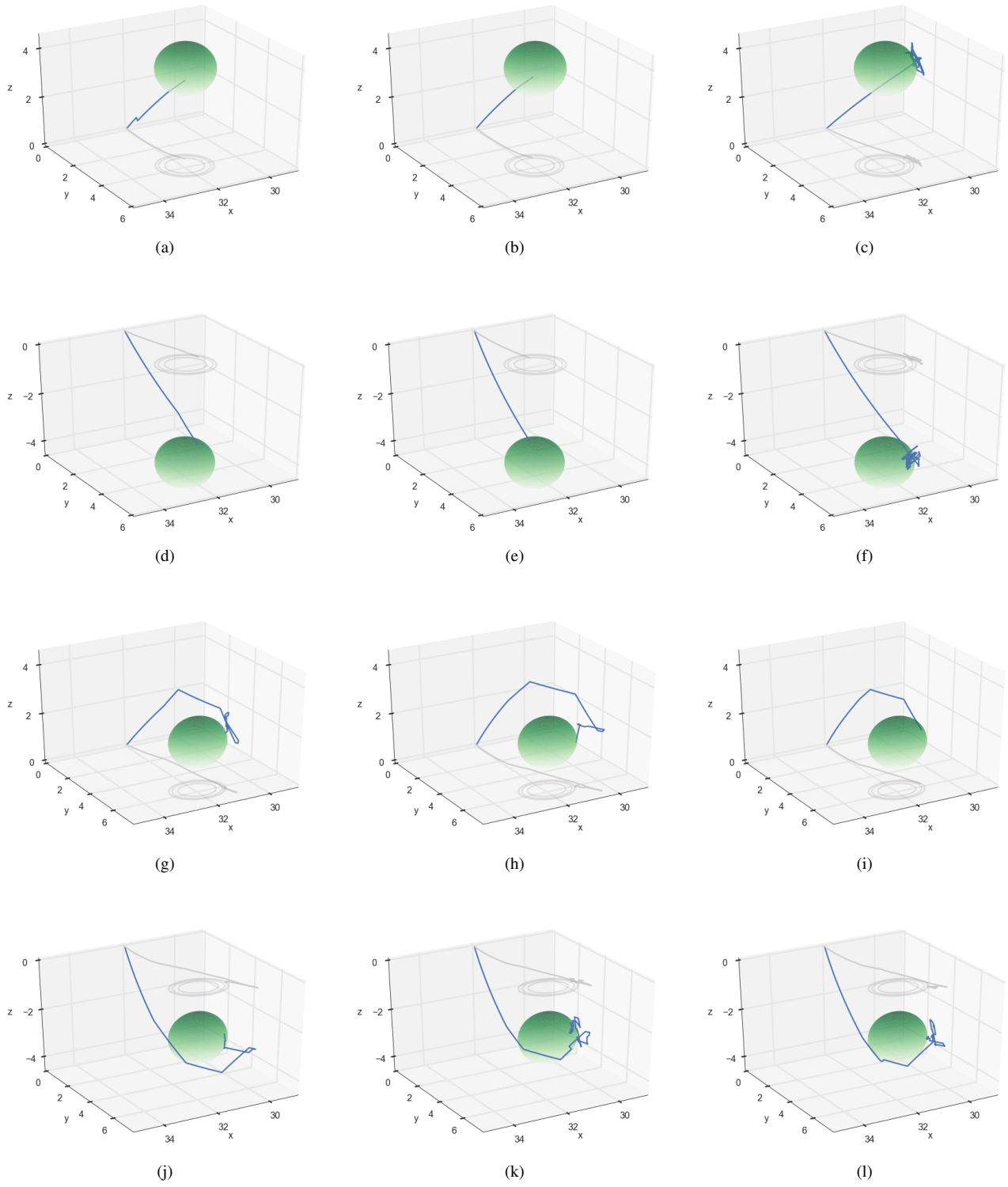

Figure 1: A sample of single-point trajectories. Each row above shows three separate trajectories (blue) of the tip of the robot catheter traveling from the coordinate position (32, 0, 0) to some goal region (green) in the absence of noise.

"ping-pongs" above the surface of the goal region because the actions are not granular enough. This behavior also explains the high variance in the number of iterations and path length for the goal regions (32, 4, 4) and (32, 4, -4). Because the online GPTD algorithm does not produce significant course corrections as the robot catheter approaches the goal region, it's far more likely to just miss the surface of the goal region and enter this degenerate "ping-pong" case, thereby producing a much more circuitous path.

5.1.2 Computational Tractability

Across all single-point trajectories, the online GPTD algorithm required a cycle time of 13.8 $\pm$ 2.15 ms. While the current implementation of the approach isn't tractable enough to maximize performance of the robot, it's not a significant bottleneck either.

6. Conclusion

The online GPTD algorithm achieves a policy with reasonable accuracy and computational tractability. Though it produces a few extremely unattractive worst-cases, these cases occur relatively infrequently, and should not discourage further exploration of this approach. Overall, we conclude that the approach performs well for single-point trajectories and remains a promising way to control soft robots in actual surgical interventions.

7. Future Work

There exists much work that still remains to produce a system that offers significant improvements to the viability of soft robotics in medical facilities.

Though our approach to seed the kernel by sampling each action 50 times results in a precise and relatively accurate initial policy, it requires a significant upfront cost. In the case of single-point trajectories, the upfront cost does not translate to much of a difference in the total runtime of the algorithm since it is only paid once per point. However, the repeated cost makes the current implementation computationally intractable for multiple-point trajectories. In order to satisfy our second task, we must determine a better means to seed the initial parameters of the online GPTD algorithm in an effective yet inexpensive way.

In addition, as the number of points along the trajectory grows, the approach becomes even less and less computationally tractable. This polynomial growth results from the quadratically increasing size of the kernel matrix that parametrizes the algorithm. To mitigate this cost, we propose a finite "memory" that discards state information received too far in the past.

In addition, most of the hyperparameters used for the online GPTD algorithm were conservative estimates of the optimal parameters. Ideally, we would learn the hyperparameters that maximize the performance of our algorithm.

8. Appendix

8.1. Posterior for the Normal Distribution

Assume the joint (X, Y) follows a bivariate Gaussian distribution. Then the posterior $(X|Y)$ follows a Gaussian distribution with parameters:

$$\mathrm{E}[X|Y] = \mathrm{E}[X] + \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(Y)}\Big(Y - \mathrm{E}[Y]\Big)$$

$$\mathrm{Var}(X|Y) = \mathrm{Var}(X) - \frac{\mathrm{Cov}^2(X,Y)}{\mathrm{Var}(Y)}$$

These expressions are derived from the Schur complement of a matrix [6].

References

[1] Daniela Rus and Michael Tolley. Design, fabrication and control of soft robots. *Nature*, 2015.

[2] Mark Earley and Richard Schilling. Catheter and surgical ablation of atrial fibrillation . *Heart*, 2006.

[3] Yaakov Engel, Shie Mannor, and Ron Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. *Proc. of the 20th International Conference on Machine Learning*, 2003.

[4] Yaakov Engel, Shie Mannor, and Ron Meir. Sparse Online Greedy Support Vector Regression . *ECML*, 2002.

[5] Bryan Jones and Ian Walker. Kinematics for multisection continuum robots . *IEEE*, 2006.

[6] Michael Jordan. The conjugate prior of the normal distribution . 2010.