LUND UNIVERSITY

Path Planning Using Wassertein Distributionally Robust Deep Q-learning

Alpturk, Cem; Renganathan, Venkatraman

Published in:
European Control Conference

2023

Link to publication

Citation for published version (APA):
Alpturk, C., & Renganathan, V. (in press). Path Planning Using Wassertein Distributionally Robust Deep Q-learning. In *European Control Conference* IEEE - Institute of Electrical and Electronics Engineers Inc..

Total number of authors:
2

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Path Planning Using Wassertein Distributionally Robust Deep Q-learning

Cem Alptürk and Venkatraman Renganathan

arXiv:2211.02372v1 [cs.RO] 4 Nov 2022

***Abstract*— We investigate the problem of risk averse robot path planning using the deep reinforcement learning and distributionally robust optimization perspectives. Our problem formulation involves modelling the robot as a stochastic linear dynamical system, assuming that a collection of process noise samples is available. We cast the risk averse motion planning problem as a Markov decision process and propose a continuous reward function design that explicitly takes into account the risk of collision with obstacles while encouraging the robot's motion towards the goal. We learn the risk-averse robot control actions through Lipschitz approximated Wasserstein distributionally robust deep Q-learning to hedge against the noise uncertainty. The learned control actions result in a safe and risk averse trajectory from the source to the goal, avoiding all the obstacles. Various supporting numerical simulations are presented to demonstrate our proposed approach.**

I. Introduction

Given the tremendous increase in the computing power, many computationally expensive control theory problems can now be addressed using the deep reinforcement learning approaches [1], [2]. So far, the motion planning problem with uncertainty has been investigated from two different perspectives namely the control theory [3] and the reinforcement learning (RL) [4]. When stochastic uncertainties are considered in the problems such as path planning, both the above said approaches resort to the powerful stochastic optimization techniques as in [5] to ensure satisfaction of specifications with high probability. However, when assumptions of certain functional forms for the system uncertainties are made in the name of tractability, they may lead to potentially severe miscalculation of risk when the uncertain robot is made to operate in a dynamic environment [6]. Such shortcomings can be addressed through carefully designed risk bounded motion planning approaches using distributionally robust optimization techniques. The interested readers are referred to these non-exhaustive list of papers on risk averse motion planning [7]–[11].

Risk averse path planning problems emphasize the need for exact propagation of uncertainties. For instance, either the distributions of all the uncertainties or the moments defining the distributions are required to be known in advance or calculated exactly for all time steps to evaluate the risk of obstacle collision as in [12]. It is an usual practice to associate a particular distribution to the uncertainty (often Gaussian) just for the sake of tractability [3]. But often in reality, all we have is just a collection of samples of the uncertainty and trying to fit a distribution to it may cause undue risk. On a parallel note, the central idea of safe and robust RL as described in [13], [14] is to learn control policies for agents that encourage safety or robustness, and to design methods that can formally certify the safety of a learned control policy. For instance, a maximum entropy based lower bound on a robust RL objective was used to learn policies that are robust to some disturbances in the dynamics and the reward function in [15]. But analysis of safe and robust RL algorithms with distributional uncertainty has received very less attention. Authors in [5] use the Wasserstein distributionally robust deep Q-learning to hedge against the distributional uncertainty and approximately solve the Bellman equation associated with the deep Q-learning approach given in [16]. In this paper, we stick to the sample based uncertainty modeling of process noise and take a similar approach as [5], and further use the Lipschitz constant based approximations advocated in Theorem 5 of [17] to learn risk-averse robot control actions. A similar problem was investigated by [4], albeit with usual Gaussian assumptions and no formal risk consideration.

Contributions: This article leverages powerful results in deep reinforcement learning theory and distributionally robust optimization to learn control policies for robots to operate in a risk-averse manner in an environment. Our main contributions are:

1) we learn safe robot control actions at all the state space positions to infer a trajectory to move from source to goal by avoiding all obstacles. We account for the uncertainty due the robot initial states and the process noise through reward function design and learn the risk averse control actions using approximated Wasserstein distributionally robust Q-learning.
2) we demonstrate our proposed approach using a series of numerical simulations and show the effectiveness of our proposed approach.

Following a short summary of notations and preliminaries, the rest of the paper is organized as follows. In §II, the risk-averse path planning problem associated with the uncertain robot system is presented. The Wasserstein distributionally robust Q-learning approach is discussed in §III. The proposed idea is then demonstrated using a numerical simulation in §IV. Finally, the paper is closed in §V. Due to the page restrictions, some proofs are available in the appendix.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 834142 (Scalable Control). C. Alptürk was a Masters Thesis student at the Department of Automatic Control, LTH, Lund University, Sweden. V. Renganathan is with the Department of Automatic Control, LTH, Lund University, Sweden. Email: venkatraman.renganathan@control.lth.se, cem.alpturk@gmail.com

NOTATIONS & PRELIMINARIES

The set of real numbers, integers are denoted by $\mathbb{R}, \mathbb{Z}$. The subset of real numbers greater than $a \in \mathbb{R}$ is denoted by $\mathbb{R}_{>a}$. The set of integers between two values $a, b \in \mathbb{Z}$ with $a < b$ is denoted by $[a : b]$. The set of non-negative integers is denoted by $\mathbb{Z}_+$. We denote by $\mathcal{B}(\mathbb{R}^d)$ and $\mathcal{P}(\mathbb{R}^d)$ the Borel $\sigma-$algebra on $\mathbb{R}^d$ and the space of probability measures on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ respectively. A probability distribution with mean μ and covariance Σ is denoted by $\mathbb{P}(\mu, \Sigma)$, and specifically $\mathcal{N}_d(\mu, \Sigma)$ if the distribution is normal in $\mathbb{R}^d$. An uniform distribution over a set A is denoted by $\mathcal{U}(A)$. Given a constant $q \in \mathbb{R}_{\geq 1}$, the set of probability measures in $\mathcal{P}(\mathbb{R}^d)$ with finite $q-$th moment is denoted by $\mathcal{P}_q(\mathbb{R}^d) := \left\{\mu \in \mathcal{P}(\mathbb{R}^d) \mid \int_{\mathbb{R}^d} \|x\|^q \, d\mu < \infty\right\}$. The type-$q$ Wasserstein distance $\forall q \geq 1$ between $\mathbb{Q}_1, \mathbb{Q}_2 \in \mathcal{P}_q(\mathbb{R}^d)$ with $\Pi(\mathbb{Q}_1, \mathbb{Q}_2)$ being the set of all joint probability distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals $\mathbb{Q}_1$ and $\mathbb{Q}_2$ is

$$W_q(\mathbb{Q}_1, \mathbb{Q}_2) \triangleq \left(\inf_{\pi \in \Pi(\mathbb{Q}_1, \mathbb{Q}_2)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|z_1 - z_2\|^q \, \pi(dz_1, dz_2) \right)^{\frac{1}{q}}. \tag{1}$$

II. PROBLEM FORMULATION

A. Robot & Environment Model

The robot is modeled as a stochastic discrete time linear time invariant system and it is assumed to move within a bounded environment $\mathcal{X} \subset \mathbb{R}^{n_x}$. There are in total $M \in \mathbb{Z}_+$ obstacles in the environment, each disjoint with the other and they are collectively referred as $\mathcal{O}$ with $|\mathcal{O}| = M$. Further, each obstacle is assumed to be static and of convex polytope shape. Then, the free space that the robot can traverse namely $\mathcal{X}_{\mathbf{free}} \subset \mathcal{X}$ is given by

$$\mathcal{X}_{\mathbf{free}} = \mathcal{X} \setminus \mathcal{X}_{\mathbf{obs}}, \quad \text{and} \quad \mathcal{X}_{\mathbf{obs}} := \bigcup_{i=1}^{M} \mathcal{X}_{\mathbf{obs}}^{(i)}, \tag{2}$$

where $\mathcal{X}_{\mathbf{obs}}^{(i)} \subset \mathcal{X}$ is the space occupied by the obstacle $i \in \mathcal{O}$. Similar to the obstacles, we define a goal region, $\mathcal{X}_{\mathbf{goal}} \subset \mathcal{X}$, that is both static and circular in shape with constant radius $R_{\mathbf{goal}} > 0$. This is a fair assumption [1] given that all the robot states that are inside the region $\mathcal{X}_{\mathbf{goal}}$ which is centered at the goal point $\bar{x}_{\mathbf{goal}} \in \mathcal{X}_{\mathbf{goal}}$ are considered to be goal states. The position of the robot at time $k \in \mathbb{Z}_+$ is denoted as $p_{r,k} \in \mathbb{R}^{n_r}$. The robot is limited to move within the environmental boundaries whose limits are $[\underline{p}, \overline{p}]$ with $\underline{p}, \overline{p} \in \mathbb{R}^{n_r}$. Hence, just like the obstacles, the environmental boundaries are also treated as terminal states. The state of the robot at time k is represented as $x_k \in \mathbb{R}^{n_x}$ and it may include the robot's position, velocity and other states of interest so that $n_x \geq n_r$. The robot is controlled through a control input u_k which is selected from $\mathcal{U}$ such that $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$. Given the above description, we define the dynamics (evolution) of the robot in $\mathcal{X}$ as

$$x_{k+1} = Ax_k + Bu_k + w_k. \tag{3}$$

[1]Our problem formulation works perfectly fine even with convex polytopic goal regions too.

The robot is subject to a process disturbance $w_k \in \mathbb{R}^{n_x}$. The time invariant true distribution of the process noise w_k at any time k namely $\mathbb{P}_w$ is unknown, however it is assumed that a collection of $N \in \mathbb{N}$ independent samples of w_k are available beforehand. That is, an i.i.d. sequence $\hat{w}_1, \ldots, \hat{w}_N \in \mathbb{R}^{n_r}$ is assumed to be known in advance. However, at any time k, the distribution of w_k can be approximated through the following empirical distribution, $\hat{\mathbb{P}}_w = \frac{1}{N} \sum_{i=1}^{N} \delta_{\hat{w}_i}$, where δ_{w_i} is the Dirac delta function. Note that, $\hat{\mathbb{P}}_w$ need not necessarily be the true distribution of the w_k. This is precisely where our approach differs from the existing safe RL literature, where it is a common practice to either assume a distribution for w or bound for w. The initial state of the robot is assumed to be random and it is modelled as $x_0 \sim \mathbb{P}_{x_0}(\bar{x}_0, \Sigma_{x_0})$, where $\mathbb{P}_{x_0}$ is assumed to be known with the mean $\bar{x}_0 \in \mathbb{R}^{n_x}$, and the covariance $\Sigma_{x_0} \in \mathbb{R}^{n_x \times n_x}$ also being assumed to be known or estimated from prior experiments. It is clear from the above setting that $\mathbb{P}_{x_k}$ for $k \geq 1$ is not known exactly despite $\mathbb{P}_{x_0}$ being known exactly.

Assumption 1. *There exists a minimum separation distance $L_{min} > 0$ between the goal and any of the obstacle regions. That is, $\mathcal{X}_{\mathbf{goal}} \cap \mathcal{X}_{\mathbf{obs}} = \emptyset$ and $\forall x_{\mathbf{goal}} \in \mathcal{X}_{\mathbf{goal}}, \forall x_{\mathbf{obs}} \in \mathcal{X}_{\mathbf{obs}}$, we see that*

$$\|x_{\mathbf{goal}} - x_{\mathbf{obs}}\|_2 \geq L_{min}. \tag{4}$$

Main Problem Statement: Given the uncertain robot evolution as in (3) with sample based process noise model, we learn the risk-averse control policy for all state space positions of the robot and hence design a trajectory for the robot from its initial state x_0 to the goal region $\mathcal{X}_{\mathbf{goal}}$ without colliding with any of the obstacles $\mathcal{O}$.

B. Markov Decision Process (MDP) Formulation

Given that we have to learn what actions to take provided we land anywhere in $\mathcal{X}$ given the uncertainty in the distributional information of w, taking control theory perspective can be hard. Hence, we resort to the RL approaches to address this shortcoming. That is, the above planning problem can be cast as a Markov decision process that consists of the tuple $\langle \mathcal{S}, \mathcal{A}, \mathbb{P}, r \rangle$. Here, $\mathcal{S} \subset \mathbb{R}^{n_\mathcal{S}}$ is the state space, $\mathcal{A} \subset \mathbb{R}^{n_\mathcal{A}}$ is a finite set called the action space with $|\mathcal{A}| \in \mathbb{N}_+ \setminus 0$, $r : \mathcal{S} \to \mathbb{R}$ is the reward function and $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the state transition probability which defines the probability distribution over the next states. We denote by $\hat{a}$, the null action where it does not cause a change in the position of the robot. We denote the total action space as $\hat{\mathcal{A}} = \mathcal{A} \cup \hat{a}$. Due to the Markov property of the system, the transition probabilities only depend on previous state and action such that for a given state s_k and action a_k and the history $h_k = \{s_0, a_0, ..., s_k, a_k\}$, we see that $\mathbb{P}(s_{k+1} \mid h_k) = \mathbb{P}(s_{k+1} \mid s_k, a_k), \forall s_{k+1} \in \mathcal{S}$. At step k, the state $s_k \in \mathcal{S}$ contains the state of the robot x_k, the center of the goal p_g, and the centers of the obstacles $p_{\mathbf{obs}}^{(i)}$.

$$s_k := \left\{ x_k, p_g, p_{\mathbf{obs}}^{(i)} \right\} \in \mathbb{R}^{n_\mathcal{S}}, \tag{5}$$

where, $n_{\mathcal{S}} = (2+M)n_x$, and $i = 1, \ldots, M$. The state s_k is referred as a *terminal state* if $x_k \in \mathcal{X}_{\mathbf{obs}} \cup \mathcal{X}_{\mathbf{goal}}$ or if $x_k \notin \mathcal{X}$ and as *non-terminal state* otherwise. The dimensions of MDP state s_k depend on the number of obstacles in the environment. Increasing the number of obstacles will cause the dimension of s_k to increase as well [2]. An action $a_k \in \hat{\mathcal{A}}$ performed at state $s_k \in \mathcal{S}$, will cause a transition to a new state $s_{k+1} \in \mathcal{S}$ with the probability $\mathbb{P}(s_{k+1} \mid s_k, a_k)$. After the transition, a deterministic reward $r_k = r(s_{k+1})$ is obtained based on the state where we land. The actions, $a_k = \pi(s_k)$ are chosen based on a deterministic policy $\pi : \mathcal{S} \to \hat{\mathcal{A}}$. The set of all admissible control policies is denoted by Π. A sequence $\tau_k^{(\pi)} := (s_k, a_k, s_{k+1}, a_{k+1}, \ldots, s_{K-1}, a_{K-1}, s_K)$ with a terminal state s_K is called a sample path under the policy $\pi \in \Pi$. The cumulative discounted reward for this sample path is

$$R^{\pi}(\tau_k) = \sum_{i=0}^{K-k-1} \gamma^i r(s_{i+k+1}), \tag{6}$$

where $\gamma \in [0,1]$ is the discount factor. The discount factor is used in order to take in to account the future rewards. The value function is defined as the expected value calculated for the discounted returns starting from state s and following policy π and the Q-function is defined as the expected discounted return if action a is taken at state s and following policy π. That is,

$$V^{\pi}(s) = \mathbb{E}\left[R^{\pi}(\tau_k \mid s_k = s)\right], \quad \text{and} \tag{7}$$

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{K-1} \gamma^k r(s_{k+1}) \mid s_0 = s, a_0 = a\right]. \tag{8}$$

The Q values for a state determine what action is the best to take. Hence, the *modified* policy π, tailored for this path planning problem is related to the Q-function as

$$\pi(s) = \begin{cases} \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a), & \text{if } s \notin \text{terminal state} \\ \hat{a}, & \text{if } s \in \text{terminal state} \end{cases}. \tag{9}$$

C. Reward Function Design & Its Approximation

In this work, we consider rewards that depend only on s_{k+1} and it includes a penalty for both traveling and collision with obstacles along with an incentive for being in the goal. That is,

$$\hat{r}(s') = \mathbf{r_{travel}} + \begin{cases} \mathbf{r_{goal}}, & \text{if } s' \in \mathcal{X}_{\mathbf{goal}}, \\ \mathbf{r_{obs}}, & \text{if } s' \in \mathcal{X}_{\mathbf{obs}} \text{ or } s' \notin \mathcal{X} \end{cases}, \tag{10}$$

where $\mathbf{r_{travel}}$ is the travel penalty, $\mathbf{r_{goal}}$ is the reward for reaching the goal, and $\mathbf{r_{obs}}$ is the penalty for obstacle collision. The discontinuity in the reward function $\hat{r}(\cdot)$ due to the switching in (10) causes its Lipschitz constant $L_{\hat{r}} \to \infty$ when we approximate the Q-function later on using a neural network. Hence, it has to be approximated by a Lipschitz continuous function $r(s')$. The radial step function associated with switching to the goal reward can be approximated as

$$f_{\mathbf{goal}}(p_r) = \frac{\mathbf{r_{goal}}}{2}\left(1 + \tanh\left(\frac{d_2(p_r, \bar{p}, R_{\mathbf{goal}})}{\delta}\right)\right), \tag{11}$$

where $p_r, \bar{p} \in \mathbb{R}^{n_r}$ denote the positions of the robot and the center of the goal respectively with $d_2(p_r, \bar{p}, R_{\mathbf{goal}}) = R_{\mathbf{goal}} - \|p_r - \bar{p}\|_2$ being the distance function and $\delta \in \mathbb{R}_{>0}$ is the slope. A similar structure, $f_{\mathbf{bor}}(p_r)$ can be used for the borders that takes the distance to the borders defined using limits $[\underline{p}, \overline{p}]$ across all the position dimensions. The step function associated with switching to the obstacle collision penalty will have the convex polytope shape as its support. Let $\mathbf{q} := \{q_i\}_{i=1}^{n_r}$, with each q_i being a large, positive and even integer. Then, using the modified distance function [3]

$$d_{\mathbf{q}}(p_r, \bar{p}, R_{\mathbf{obs}}) = R_{\mathbf{obs}} - \left(\sum_{i=1}^{n_r} \left|p_r^{(i)} - \bar{p}^{(i)}\right|^{q_i}\right)^{\frac{1}{\max\{\mathbf{q}\}}}, \tag{12}$$

with $R_{\mathbf{obs}} > 0$, we can obtain a smooth approximation of a rectangle whose centroid is at $\bar{p}$ and length of its biggest side being $2R_{\mathbf{obs}}$. Then, the polytopic obstacle $i \in \mathcal{O}$ can be represented by stitching together several such rectangles defined using the tuple $\{\bar{p}_j, R_{\mathbf{obs}}^{(j)}\}_{j=1}^{M_i}, M_i \in \mathbb{N}_{\geq 2}$. Hence,

$$f_{\mathbf{obs}}^{(i)}(p_r) = \frac{\mathbf{r_{obs}}}{2}\left(1 + \tanh\left(\frac{\sum_{j=1}^{M_i} d_{\mathbf{q}}(p_r, \bar{p}_j, R_{\mathbf{obs}}^{(j)})}{\delta}\right)\right), \tag{13}$$

where, $i \in \mathcal{O}$. Then, the Lipschitz continuous approximation $r(s')$ of the original reward function $\hat{r}(s')$ is given by

$$r(s') = \mathbf{r_{travel}} + f_{\mathbf{goal}}(p_r) + f_{\mathbf{bor}}(p_r) + \sum_{i \in \mathcal{O}} f_{\mathbf{obs}}^{(i)}(p_r). \tag{14}$$

III. Learning Risk-Averse Control Actions

If the Q-values for a system is known, a policy π can be used to maximize the expected returns. In order to estimate the Q-values, the standard temporal difference learning based Q-Learning procedure is usually employed, [18]. From now on, we drop the superscript π on $Q^{\pi}(s,a)$ for the brevity of notation. We now define the Bellman operator, $\mathcal{T} : \mathbb{R}^{\mathcal{S} \times \hat{\mathcal{A}}} \to \mathbb{R}^{\mathcal{S} \times \hat{\mathcal{A}}}$ as

$$\mathcal{T}Q(s,a) = \mathbb{E}_{s'}\left[r(s') + \gamma \max_{a' \in \mathcal{A}} Q(s', a')\right], \tag{15}$$

where the outer expectation is over the next states s', which come from the transition probability $\mathbb{P}(s' \mid s, a)$. Given the continuous state space setting, we propose to use the Deep Q Learning (DQN) approach as in [16], which estimates the Q values by using a deep neural network. Specifically, it utilizes two neural networks namely: i) Q-network $Q(s,a,\theta)$, and ii) the target network $Q(s,a,\theta^-)$. The Q network is

[2] The increase in the dimension of s_k is the price that we need to pay to handle potentially dynamic obstacles.

[3] For $n_r \geq 2$, the distance function should be defined using the level set of the convex obstacle obtained from its compact support and smooth approximation can be done using the $\tanh$ (or logistics) function in $\mathbb{R}^{n_r}$.

trained by experience replay, where a random batch of experiences are sampled from the memory buffer and with the Bellman equation, the targets are calculated. For an experience $\langle s, a, r, s' \rangle$, the target y is calculated as

$$y = \begin{cases} r + \gamma \max_{a' \in \mathcal{A}} Q(s', a', \theta^-), & \text{if } s' \text{ is non-terminal,} \\ r, & \text{if } s' \text{ is terminal.} \end{cases} \tag{16}$$

A. The Lipschitz Approximated Wasserstein Distributionally Robust Deep Q-Learning

The Q-function estimation defined in section II-B will be formulated as a distributionally robust optimization problem. Taking an action a at state s causes the transition to an unknown state s' with unknown distribution $\mathbb{P}_{s'} \triangleq \mathbb{P}(s' \mid s, a)$. For brevity, a will be omitted in the notation from now.

Assumption 2. *The true distribution $\mathbb{P}_{s'}$ is a light-tailed distribution [19]. That is, $\exists p > 1$ such that,*

$$\mathbb{E}_{\mathbb{P}_{s'}}\left[e^{\|s'\|^p}\right] = \int_{\mathcal{S}} e^{\|s'\|^p} d\mathbb{P}_{s'}(s') < \infty. \tag{17}$$

1) The Wasserstein Ambiguity Set: Given a robot state $x_k \in \mathcal{X}$ and an input $a_k \in \hat{\mathcal{A}}$, an empirical distribution for x_{k+1} is given by,

$$\hat{\mathbb{P}}_{x_{k+1}} := \frac{1}{N}\sum_{i=1}^{N} \delta_{\hat{x}_{k+1}^{(i)}} = \frac{1}{N}\sum_{i=1}^{N} \delta_{Ax_k + Ba_k + \hat{w}_i}. \tag{18}$$

By knowing the state of the robot x_k, the full state s_k can be obtained by using the positions of the goal and obstacles of the current environment (which do not depend on the position of the robot), since these stay constant during an episode. For ease of notation, we refer to the next state s_{k+1} as s', and the samples of s' obtained from (18), are denoted as $\hat{s}'^{(i)}$ for $i = 1, \dots N$. Then, the empirical distribution is given by $\hat{\mathbb{P}}_{s'} = \frac{1}{N}\sum_{i=1}^{N} \delta_{\hat{s}'^{(i)}}$. When an action a is performed while in state s, the nominal distribution for the center of the Wasserstein ball will be $\hat{\mathbb{P}}_{s'}$, and the worst case transition will be coming from a distribution that is inside this ball. We define the ambiguity set $\mathcal{B}_{s,a}$,

$$\mathcal{B}_{s,a} := \left\{ \mathbb{P}_{s'} \in \mathcal{P}(\mathcal{S}) \mid W_1\left(\hat{\mathbb{P}}_{s'}, \mathbb{P}_{s'}\right) < \epsilon_{s'} \right\}. \tag{19}$$

The Wasserstein ball radius $\epsilon_{s'}$ is chosen such that the true distribution $\mathbb{P}_{s'}$ lies within this Wasserstein ball with probability greater than $1-\beta$. The β parameter will determine the allowed risk factor for the solution. A smaller β will result in a larger radius which causes the generated policy to be much more risk averse and vice-versa. Since, the radius $\epsilon_{s'}$ quantifies the amount of trust (distrust) that we have over the $\hat{\mathbb{P}}_{s'}$, it is chosen such that

$$\mathbb{P}\left(\mathbb{P}_{s'} \in \mathcal{B}_{s,a}\right) \geq 1 - \beta, \quad \beta \in [0, 1]. \tag{20}$$

Lemma 1. *Based on Assumption 2, for an empirical distribution $\hat{\mathbb{P}}_{s'}$ with N atoms and $\rho = \mathbf{diam}(\mathbf{supp}(\hat{\mathbb{P}}_{s'}))$, the radius of the Wasserstein ambiguity set for the state distributions is*

$$\epsilon_{s'} = \rho\sqrt{\frac{2}{N}\ln\left(\frac{1}{\beta}\right)}. \tag{21}$$

2) Approximated Solution to The Wasserstein Distributionally Robust Q-learning Problem: We define the distributionally robust Bellman operator $\hat{\mathcal{T}} : \mathbb{R}^{\mathcal{S}\times\hat{\mathcal{A}}} \to \mathbb{R}^{\mathcal{S}\times\hat{\mathcal{A}}}$ to represent the worst case expected returns, so that risk can be incorporates into the Q-values. That is,

$$\hat{\mathcal{T}}Q(s,a) := \inf_{\mathbb{P}_{s'} \in \mathcal{B}_{s,a}} \mathbb{E}_{s' \sim \mathbb{P}_{s'}}\left[h(s')\right], \quad \text{where,} \tag{22}$$

$$h(s') := \underbrace{r(s')}_{:=h_r(s')} + \underbrace{\gamma \max_{a' \in \mathcal{A}} Q(s', a')}_{:=h_Q(s')}. \tag{23}$$

Since the Q-function is approximated using a neural network with hidden layers and non-linear activation functions, $h(s')$ turns out to be a non-convex function of the states. Since an exact solution to the infinite dimensional problem (22) using duality theory is difficult to find when the objective function is non-convex, we resort to the Lipschitz constant based approximation.

Lemma 2. *The Lipschitz approximation of the right hand side of (22) has an equivalent solution for the case when the objective function h is to be minimized and this results in a lower bound for (22), where $\epsilon_{s'}$ becomes larger in practice and the solution can become more risk averse. That is,*

$$\inf_{\mathbb{P}_{s'} \in \mathcal{B}_{s,a}} \mathbb{E}_{s'}[h(s')] \geq \mathbb{E}_{s'}[h(s')] - \epsilon_{s'} L_h. \tag{24}$$

3) Calculating the Lipschitz Constant L_h of $h(s')$.: The Lipschitz constant for $h_r(s')$ and $h_Q(s')$ can be calculated or estimated independently and then combined to get the Lipschitz constant of $h(s')$. The second part of $h(s')$ given by $h_Q(s')$ contains the Q-function which is approximated by a neural network. The neural network takes the state s as an input and returns the Q-values for each action. An upper bound for the Lipschitz constant of a dense neural network with ReLU activation functions can be approximated using the LipSDP package developed by [20].

Lemma 3. *Let $f_i : \mathbb{R}^n \to \mathbb{R}, i = [1 : N]$ be Lipschitz continuous with constants K_{f_i}. Then, the Lipschitz constant of the functions $f_g = \max\left(\{f_i(x)\}_{i=1}^N\right)$ and $f_f = \sum_{j=1}^N f_j(x)$ are respectively*

$$K_{f_g} = \max\left\{K_{f_1}, \dots, K_{f_N}\right\}, \quad \text{and } K_{f_f} = \sum_{j=1}^{N} K_{f_i}. \tag{25}$$

Lemma 4. *Let $A \in \mathbb{R}$, $\delta \in \mathbb{R}_{>0}$ and $p, g \in \mathbb{R}^2$. Then, the Lipschitz constants of the scalar functional $F(x) = \frac{A}{2}\left(1 + \tanh\left(\frac{x}{\delta}\right)\right)$, the function $f(p) = \frac{A}{2}\left(1 + \tanh\left(\frac{R - \|p-g\|_2}{\delta}\right)\right)$, and the function $\mathcal{K}(p) = \frac{A}{2}\left(1 + \tanh\left(\frac{d_q(p,\bar{p},R)}{\delta}\right)\right)$ are equal and $K_f = L_f = \mathcal{K}_f = \frac{|A|}{2\delta}$ respectively.*

Theorem 1. *Given assumption 1, the Lipschitz constant of the reward function r given by (30) is*

$$L_r = \frac{\max\{|\mathbf{r_{goal}}|, |\mathbf{r_{obs}}|\}}{2\delta}.$$

Proof. Based on Assumption 1, the individual terms that contribute to the total reward function r in (30) do not interfere with each other. Then, it follows from Lemma 3 that the Lipschitz constant of the reward function is the maximum of the Lipschitz constants of the individual terms. □

Lemma 5. *Given that $h_Q(s') = \gamma \max\{Q(s', a_1), \ldots, Q(s', a_{n_\mathcal{A}})\}$, where $n_\mathcal{A} = |\mathcal{A}|$, and the network has the upper bounded Lipschitz constants $K_{a_i}, \forall a_i \in \mathcal{A}$, the Lipschitz constant of $h_Q(s')$ is*

$$L_Q = \gamma \max\left\{K_{a_1}, \ldots, K_{a_{n_\mathcal{A}}}\right\}. \tag{26}$$

Proof of the above lemma follows by direct application of Lemma 3 on $h_Q(s')$. Having found the Lipschitz constants of both $h_r(s')$ and $h_Q(s')$, the following theorem establishes the Lipschitz constant for the objective function $h(s')$.

Theorem 2. *The upper bound of the Lipschitz constant of $h(s')$ defined in (22) is given by*

$$L_h \leq \frac{\max\{|\mathbf{r_{goal}}|, |\mathbf{r_{obs}}|\}}{2\delta} + \gamma \max\left\{K_{a_1}, \ldots, K_{a_{n_\mathcal{A}}}\right\}. \tag{27}$$

Proof. Using (23), the upper bound for the Lipschitz constant of $h(s')$ can be computed by using Lemma 3 as,

$$L_h \leq \frac{\max\{|\mathbf{r_{goal}}|, |\mathbf{r_{obs}}|\}}{2\delta} + \gamma \max\left\{K_{a_1}, \ldots, K_{a_{n_\mathcal{A}}}\right\}. \tag{28}$$

The result is an upper bound due to L_Q being an upper bound to the $h_Q(s')$. □

IV. Numerical Results

We consider the robot to be moving in an environment $\mathcal{X} \subset \mathbb{R}^2$ with the limits of $\mathcal{X}$ being $[-10, 10]^2$ in both dimensions [4]. There are in total two obstacles that are circular in shape (most simple convex shape assumption made for the sake of simplicity) and a goal region with equal radius namely, $R_{\mathbf{goal}} = R^{(1)}_{\mathbf{obs}} = R^{(2)}_{\mathbf{obs}} = 2$. The robot moves within $\mathcal{X}$ according to the following dynamics,

$$x_{k+1} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{A} x_k + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{B} u_k + w_k. \tag{29}$$

The state of the robot x_k represents the position of the robot in $\mathcal{X} \subset \mathbb{R}^2$. The process disturbance $w_k \in \mathbb{R}^2$ shifts the position of the robot by a random amount in each axis. For simulation purposes, we considered 10^4 samples of process noise w that were sampled from distributions with zero mean and covariance being equal to $0.15 I_2$. The action space $\hat{\mathcal{A}}$ consists of $\left|\hat{\mathcal{A}}\right| = 9$ actions where each action is a $\mathbb{R}^2$ vector with unit norm, that represent a step that can be taken in one of the 8 equally spaced radial directions along with a null action. The robot takes a step in a specified direction for each action and stays still if a null action is selected. The reward function has the constants $\mathbf{r_{travel}} = -0.001$, $\mathbf{r_{goal}} = 1$ and $\mathbf{r_{obs}} = -1$. The steepness of the tanh functions is chosen as $\delta = 0.1$. The continuous reward function that is used here is,

$$\begin{aligned} r(s') = \mathbf{r_{travel}} &+ \frac{\mathbf{r_{goal}}}{2}\left(1 + \tanh\left(\frac{R_{\mathbf{goal}} - \|p_r - p_g\|_2}{\delta}\right)\right) \\ &+ \frac{\mathbf{r_{obs}}}{2}\sum_{j=1}^{n_r}\left(2 + \tanh\left(\frac{\underline{p}(j) - p_r(j)}{\delta}\right) + \tanh\left(\frac{p_r(j) - \overline{p}(j)}{\delta}\right)\right) \\ &+ \sum_{i=1}^{M}\frac{\mathbf{r_{obs}}}{2}\left(1 + \tanh\left(\frac{R^{(i)}_{\mathbf{obs}} - \left\|p_r - p^{(i)}_{obs}\right\|_2}{\delta}\right)\right). \end{aligned} \tag{30}$$

Algorithm 1 Lipschitz Approximated Wasserstein Distributionally Robust DQN

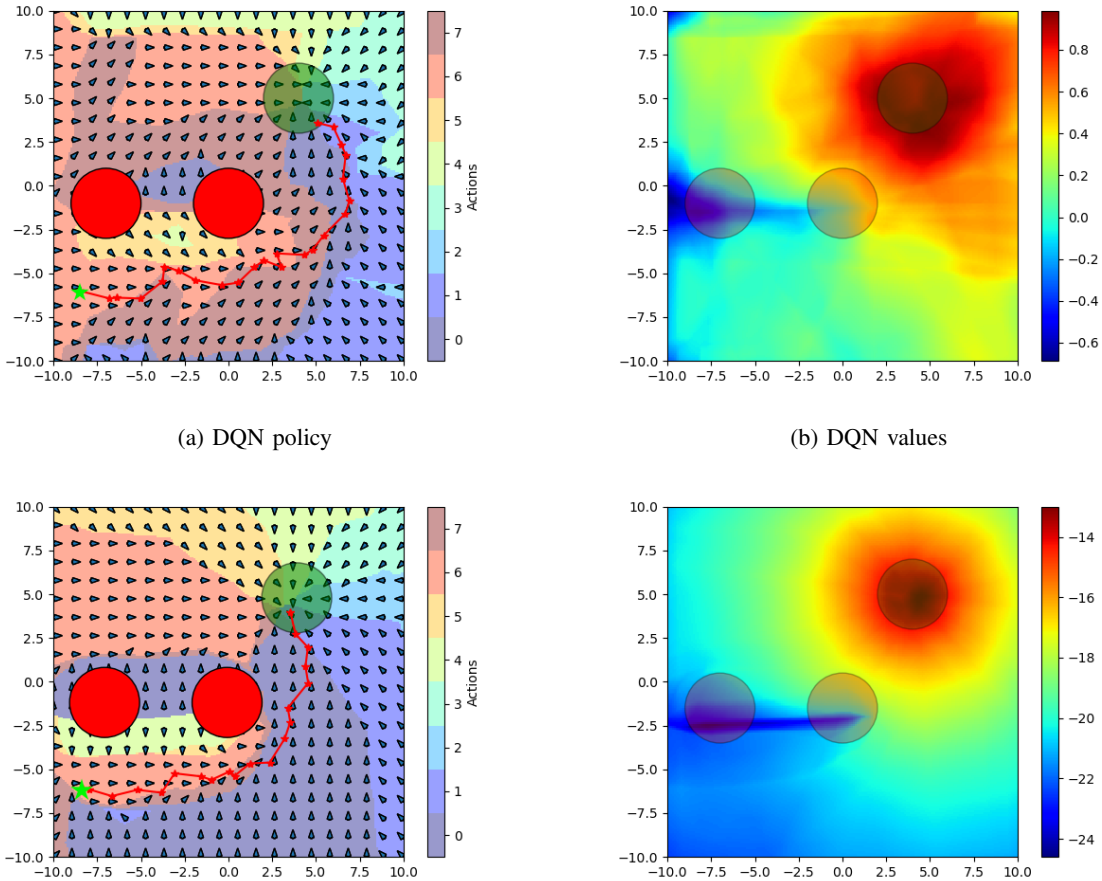
Require: Disturbance samples $\hat{w}_1, \ldots \hat{w}_N$, Learning rate η, Max episodes N_{ep}, Episode Length N_{step}, Batch size N_{batch}
Initialize replay memory $\mathcal{M} \leftarrow \emptyset$
Initialize network weights θ, θ^-
Estimate Lipschitz constant of network θ^-
Compute ϵ_s by (21)
for episode $= 1 : N_{ep}$ **do**
  Initialize $s \in \mathcal{S}$
  for $k = 1 : N_{step}$ **do**
    Select action a_k with ϵ-greedy policy π
    Observe next state s' and reward r
    Append experience (s, a, r, s') to $\mathcal{M}$
    Initialize loss $\delta \leftarrow 0$
    for $j = 1 : N_{batch}$ **do**
      Sample experience (s_j, a_j, r_j, s'_j) from $\mathcal{M}$
      Compute nominal distribution $\hat{\mathbb{P}}_{s'}$
      Approximate target y_j by (24) with the target network
      Accumulate loss $\delta \leftarrow \delta + (y_j - Q(s_j, a_j; \theta))^2$
    Update weights θ by loss δ with backpropagation
    Set $\theta^- \leftarrow \theta$ and compute Lipschitz constant of network θ^- every Γ steps

A. Discussion of Results

Models	$\epsilon_{s'}$	Noise covariance					
		$\Sigma_w = 0_2$		$\Sigma_w = 0.15 I_2$		$\Sigma_w = 0.3 I_2$	
		Reward		Reward		Reward	
		Mean	Std	Mean	Std	Mean	Std
DQN	N/A	0.636	0.545	0.662	0.514	0.627	0.573
DRDQN	0	0.850	0.334	0.811	0.401	0.749	0.510
DRDQN	0.067	0.829	0.356	0.811	0.387	0.756	0.489

TABLE I: The mean and the standard deviation of the total rewards with different noise covariances corresponding to different training models are tabulated here.

[4] We believe that such a toy example is rich enough to demonstrate our proposed approach given the infinite dimensional DRDQN objective.



(a) DQN policy

(b) DQN values

(c) DRDQN policy with $\epsilon_s = 0.067$.

(d) DRDQN values with $\epsilon_s = 0.067$.

Fig. 1: The result of training DQN model and DRDQN model with $\epsilon_s = 0.067$ is shown here. Also, the solution trajectories from a starting position in green star to the goal region in green color avoiding both the red color obstacles are shown in both cases. The plot on the left shows the learned control policies and the one on right depicts the learned Q values.

Models	$\epsilon_{s'}$	Reached Goal			Resulted in Collision			Wandering Around		
		$\Sigma_w(\times I_2)$			$\Sigma_w(\times I_2)$			$\Sigma_w(\times I_2)$		
		0	0.15	0.3	0	0.15	0.3	0	0.15	0.3
DQN	N/A	76.7%	82.2%	81.4%	1.5%	6.6%	6.9%	21.7%	13.9%	11.9%
DRDQN	0	97.9%	96.4%	93.1%	0.7%	3.1%	6.5%	1.3%	0.3%	0.2%
DRDQN	0.067	95.3%	95.7%	93%	0.5%	2.4%	5.5%	4.1%	1.8%	1.4%

TABLE II: The percentage of trajectories that reached the goal, resulted in collision and those that did neither are tabulated here for different noise covariances corresponding to different training models.

The hyperparameter details of the training results are made available in the supplementary material. The resulting policy and the state values for the trained models can be seen in Figure 1. The arrows represent the action that the policy gives at the respective robot position and goal/obstacle positions. The heatmap represents the same values with color but in higher resolution to better understand the decision boundaries. The figures on the right side of Figure 1 represent the value of each state s which can be computed by $\max_{a \in \mathcal{A}} Q(s, a)$. It can be seen that the rewards propagate from the goal and the obstacles. Further, the resulting learned policy restricts the robot moving between the obstacles and there exists a boundary around the obstacles. When compared with the DQN model, our solution exhibits the most minimum pessimism (risk aversion). This difference is due to the fact that DQN learns the expected rewards, while the DRDQN learns the worst case expected rewards. Due to the noise samples used in DRDQN model with $\epsilon_{s'}$ calculated using (21), learning the policy occurs in less steps compared to the DQN model. However DRDQN can take more time since

the computational load is higher for calculating the targets. The DRDQN with $\epsilon_{s'} = 0$ is virtually the same as DQN as it learns faster since it calculates the expected values in (15) more accurately compared to that of DQN which uses only one sample. DQN achieves a lower score overall, since the method only uses one experience per experience replay to train itself, while DRDQN uses the samples provided which results in a much better approximation of (worst case) expected future returns. The models have been evaluated by running 10^5 episodes each with random goal/obstacle configurations, for three different noise distributions. As seen in Table I both versions of DRDQN have a higher average total reward compared to DQN with lower variances. Also in Table II, the percentage of trajectories that have reached the goal, collided with an obstacle or border or have not reached the goal or collided, has been provided. It can be inferred that as the covariance of the noise increases, the DRDQN model is able to maintain a low collision rate due to the worst case approximations. Any safe RL algorithm that assumes a particular distribution for w or a bound for w has a greater chance to fail in this setting as the unknown true noise distribution will lead to different transitions than the one assumed.

V. Conclusion

We proposed a path planning using approximated Wasserstein distributionally robust deep Q-learning approach. Through carefully designed reward function, we showed how to learn safe control policy for uncertain robots operating in an environment. Our numerical simulation results demonstrated our proposed approach.

References

[1] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105 669–105 679, 2019.

[2] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 297–309, 2020.

[3] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *AIAA guidance, navigation, and control conference*, 2010, p. 8160.

[4] J. Raajan, P. V. Srihari, J. P. Satya, B. Bhikkaji, and R. Pasumarthy, "Real time path planning of robot using deep reinforcement learning," vol. 53. Elsevier B.V., 2020, pp. 15 602–15 607.

[5] A. Kandel and S. J. Moura, "Safe Wasserstein constrained deep q-learning," *arXiv preprint arXiv:2002.03016*, 2020.

[6] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," in *Robotics Research*. Springer, 2020, pp. 75–84.

[7] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[8] D. N. Subramani and P. F. Lermusiaux, "Risk-optimal path planning in stochastic dynamic environments," *Computer Methods in Applied Mechanics and Engineering*, vol. 353, pp. 391–415, 2019.

[9] X. Xiao, J. Dufek, and R. Murphy, "Explicit-risk-aware path planning with reward maximization," *arXiv preprint arXiv:1903.03187*, 2019.

[10] P. Lathrop, B. Boardman, and S. Martinez, "Distributionally safe path planning: Wasserstein safe rrt," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 430–437, 2021.

[11] S. Safaoui, B. J. Gravell, V. Renganathan, and T. H. Summers, "Risk-averse rrt planning with nonlinear steering and tracking controllers for nonlinear robotic systems under uncertainty," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3681–3688.

[12] V. Renganathan, S. Safaoui, A. Kothari, B. Gravell, I. Shames, and T. Summers, "Risk bounded nonlinear robot motion planning with integrated perception & control," *arXiv preprint arXiv:2201.01483*, 2022.

[13] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[14] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[15] B. Eysenbach and S. Levine, "Maximum entropy rl (provably) solves some robust rl problems," *arXiv preprint arXiv:2103.06257*, 2021.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[17] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh, "Wasserstein distributionally robust optimization: Theory and applications in machine learning," in *Operations research & management science in the age of analytics*. Informs, 2019, pp. 130–166.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[19] R. Chen and I. Paschalidis, *Distributionally Robust Learning*, 12 2020, vol. 4.

[20] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[22] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015. [Online]. Available: https://arxiv.org/abs/1511.05952

Appendix

Proof of Lemma 1: Based on Assumption 2, the risk factor β and the radius $\epsilon_{s'}$ are related as [19],

$$\mathbb{P}(W_1(\mathbb{P}_{s'}, \hat{\mathbb{P}}_{s'}) \geq \epsilon_{s'}) \leq \begin{cases} c_1 e^{-c_2 N \epsilon_{s'}^{\max(d,2)}}, & \text{if } \epsilon_{s'} \leq 1 \\ c_1 e^{-c_2 N \epsilon_{s'}^{a}}, & \text{if } \epsilon_{s'} > 1 \end{cases},$$

where d is the dimension of $s' \in \mathbb{R}^d$. In order to obtain $\mathbb{P}(W_1(\hat{\mathbb{P}}_{s'}, \mathbb{P}_{s'}) \leq \epsilon_{s'}) \geq 1-\beta$, the radius has to be selected as follows

$$\epsilon_{s'}(\beta) = \begin{cases} \left(\frac{\ln(c_1\beta^{-1})}{c_2 N}\right)^{1/\max(d,2)}, & \text{if } N \geq \frac{\ln(c_1\beta^{-1})}{c_2} \\ \left(\frac{\ln(c_1\beta^{-1})}{c_2 N}\right)^{1/a}, & \text{if } N < \frac{\ln(c_1\beta^{-1})}{c_2} \end{cases}$$

where $c_1, c_2 \in \mathbb{R}$ are constants that depend on d and N. For a discrete nominal distribution $\hat{\mathbb{P}}_{s'}$, the radius for the ambiguity set can be calculated as,

$$\mathbb{P}(W(\mathbb{P}_{s'}, \hat{\mathbb{P}}_{s'}) \leq \epsilon_{s'}) \geq 1 - \underbrace{e^{-\frac{\epsilon_{s'}^2 N}{2\rho^2}}}_{:=\beta},$$

where ρ is the diameter of the support of the true distribution $\mathbb{P}_{s'}$. For this paper, ρ is estimated with $\rho \sim \mathbf{diam}(\mathbf{supp}(\hat{\mathbb{P}}_{s'}))$. By solving for $\epsilon_{s'}$, we can get,

$$\epsilon_{s'} = \rho\sqrt{\frac{2}{N}\ln\left(\frac{1}{\beta}\right)}.$$

Proof of Lemma 2: The Lipschitz approximation can be converted in to a minimization problem by switching the objective function h with $-h$ and multiplying by -1. So,

$$\sup_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[h(s')] = -\inf_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[-h(s')]$$

If we substitute h with $-h$ in the maximization problem, the Lipschitz approximation becomes,

$$\begin{aligned}
&\sup_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[-h(s')] \leq \mathbb{E}_{s'\sim\hat{\mathbb{P}}_{s'}}[-h(s')] + \epsilon_{s'}K_{-h} \\
\iff &\inf_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[h(s')] \geq -\mathbb{E}_{s'\sim\hat{\mathbb{P}}_{s'}}[-h(s')] - \epsilon_{s'}K_{-h} \\
\iff &\inf_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[h(s')] \geq \mathbb{E}_{s'\sim\hat{\mathbb{P}}_{s'}}[h(s')] - \epsilon_{s'}K_{-h},
\end{aligned}$$

where K_{-h} is the Lipschitz constant of $-h$. The Lipschitz constant L_{-h} is equivalent to L_h since,

$$\underbrace{\|-h(x)-(-h(y))\|}_{=\|h(x)-h(y)\|} \leq K_{-h}\|x-y\|, \quad \forall x,y\in\mathbb{R}^{n_S}, x\neq y$$

Hence, $\inf_{\mathbb{P}\in\mathcal{B}_{s,a}} \mathbb{E}_{s'\sim\mathbb{P}}[h(s')] \geq \mathbb{E}_{s'\sim\hat{\mathbb{P}}_{s'}}[h(s')] - \epsilon_{s'}K_h$.

Proof of Lemma 3: We will prove for the case $N=2$ and the result for $N>2$ follows similarly. For the two Lipschitz continuous functions f_1, f_2, and $g = f_1 + f_2$, we see that

$$\begin{aligned}
|g(x)-g(y)| &= |f_1(x)+f_2(x)-f_1(y)-f_2(y)| \\
&\leq |f_1(x)-f_1(y)| + |f_2(x)-f_2(y)| \\
&\leq \underbrace{(L_1+L_2)}_{:=L_g}\|x-y\|_2, \quad \forall x,y\in\mathbb{R}^n
\end{aligned}$$

For the function $g=\max\{f_1,f_2\}$, $f_1,f_2:\mathbb{R}^n\to\mathbb{R}$, where f_1 and f_2 are Lipschitz continuous with constants K_{f_1}, K_{f_2}, the Lipschitz constants can be defined by,

$$\|\nabla f_1\| \leq K_{f_1}, \quad \|\nabla f_2\| \leq K_{f_2}.$$

The gradient of g is,

$$\|\nabla g\| = \begin{cases} \|\nabla f_1\|, & \text{if } f_1(x) > f_2(x) \\ \|\nabla f_2\|, & \text{if } f_2(x) > f_1(x) \end{cases} \leq \max\{\|\nabla f_1\|, \|\nabla f_2\|\}.$$

Thus $K_g = \max\{K_{f_1}, K_{f_2}\}$. For a function that is the maximum of N functions, this process can be applied inductively to find $K_g = \max\{K_{f_1},\dots K_{f_N}\}$.

Proof of Lemma 4: For the given scalar functional, its Lipschitz constant corresponds to the maximum magnitude of its slope.

$$L_f = \sup_{x\in\mathbb{R}} |F'(x)| \implies F'(x) = \frac{A}{2\delta}\left(1-\tanh^2\left(\frac{x}{\delta}\right)\right)$$

$$F''(x) = 0 \implies x = 0 \implies L_f = |F'(0)| = \frac{|A|}{2\delta}.$$

Similarly, given that $p=[x,y]$ and $g=[g_x,g_y]$ and $f(p) = \frac{A}{2}(1+\tanh(h(p)))$, where

$$h(p) = \frac{R-\|p-g\|_2}{\delta}, \quad \text{and}$$

$$\left|\frac{\partial f}{\partial x}\right| = \frac{|A|\,|x-g_x|}{2\delta\|p-g\|_2}\left(1-\tanh^2\left(\frac{R-\|p-g\|_2}{\delta}\right)\right)$$

The maximum slope occurs at $\tanh(0)$, which corresponds to $\|p-g\|_2 = R$. For a point on this circle such as $|x-g_x| = R$ and $y=g_y$, the slope becomes $\frac{|A|}{2\delta}$. Thus the Lipschitz constant of the function f is $K_f = \frac{|A|}{2\delta}$. A similar reasoning can be applied for finding the Lipschitz constant of the function $\mathcal{K}(p)$. The maximum value of the derivative of $\mathcal{K}(p)$ occurs at the points where $R_{\mathbf{obs}} = \left(\sum_{i=1}^{n_r}\left|p_r^{(i)} - \bar{p}^{(i)}\right|^{q_i}\right)^{\frac{1}{\max\{\mathbf{q}\}}}$. The points that satisfy this create a rectangle that makes up one of the borders of the obstacle. Since our convex polytope is made up of several such stitched together rectangles, it is straightforward to see that $\mathcal{K}_f = |A|/(2\delta)$.

Implementation Details

The simulations were performed on a Dell R530 with 2 Xeon E5-2620 6-core 12-thread CPU's and 132GB of RAM. Simulation time can be improved with a GPU during training. The episodes are limited to 50 steps. We use a dense neural network with $n_{\mathcal{S}} = 8$ inputs that correspond to the current states to approximate the Q-values. The network has two hidden layers with 150 neurons each that have ReLU activation functions. The network has 9 outputs where each output corresponds to the Q-value for the state action pair. The DQN model has a duelling architecture which is explored in [21]. Both models utilize prioritized experience replay in order to prioritize rare experiences during training and the hyperparameters were used as recommended in [22]. The memory buffer $\mathcal{M}$ is a fixed size buffer that once full, a new experiences replaces the oldest one. During training, collisions do not end the simulation in order to allow the robot to explore further and gain experiences that reach the goal. This does not change anything for the experience replay part since terminal states are handled separately. The probability of taking a random action ϵ is reduced from 1 to 0.1 linearly for the first $3/4$ of training and kept at 0.1 for the remaining episodes.

Hyperparameters	DQN	DRDQN	
		$\epsilon_{s'}=0$	$\epsilon_{s'}=0.067$
γ	0.9	0.9	0.9
η	10^{-4}	10^{-4}	10^{-4}
Steps per episode	50	50	50
Total steps	10^7	2.4×10^5	2.4×10^5
N_{batch}	32	32	32
β	N/A	N/A	0.1
N (samples)	N/A	10^4	10^4
Γ	5000	1500	1500
$\lvert\mathcal{M}\rvert$	5000	5000	5000
ϵ	$1\to0.1$	$1\to0.1$	$1\to0.1$

TABLE III: Hyperparameters used in the simulation results.