

Safety-Aware Reinforcement Learning for Control via Risk-Sensitive Action-Value Iteration and Quantile Regression

Clinton Enwerem, Aniruddh G. Puranic, John S. Baras, and Calin Belta

Institute for Systems Research

University of Maryland, College Park, United States

{enwerem, puranic, baras, calin}@umd.edu

Abstract: Mainstream approximate action-value iteration reinforcement learning (RL) algorithms suffer from overestimation bias, leading to suboptimal policies in high-variance stochastic environments. Quantile-based action-value iteration methods reduce this bias by learning a distribution of the expected cost-to-go using quantile regression. However, ensuring that the learned policy satisfies safety constraints remains a challenge when these constraints are not explicitly integrated into the RL framework. Existing methods often require complex neural architectures or manual tradeoffs due to combined cost functions. To address this, we propose a risk-regularized quantile-based algorithm integrating Conditional Value-at-Risk (CVaR) to enforce safety without complex architectures. We also provide theoretical guarantees on the contraction properties of the risk-sensitive distributional Bellman operator in Wasserstein space, ensuring convergence to a unique cost distribution. Simulations of a mobile robot in a dynamic reach-avoid task show that our approach leads to more goal successes, fewer collisions, and better safety-performance trade-offs than risk-neutral methods.

1 Introduction

Standard Reinforcement Learning (RL) methods optimize expected cumulative cost but often use *misaligned* or *underspecified* cost functions that oversimplify real-world safety constraints, leading to overlooked risks. This limitation hinders deployment in high-risk domains where safety is crucial [1]. For instance, a risk-neutral autonomous robot may reach deadlock states, jeopardizing its mission and causing severe consequences [2]. Addressing this requires integrating explicit safety constraints to prevent risky behavior while preserving essential exploration.

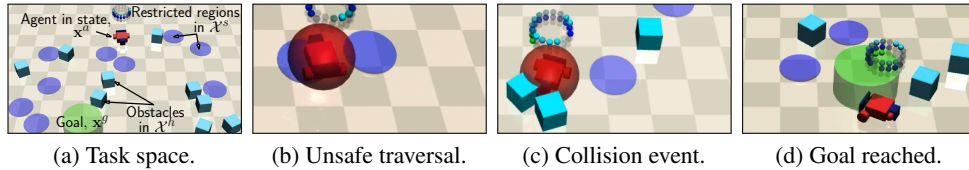


Figure 1: **Reach-avoid navigation task:** Close-ups from our experiments (Section 6) showing a differentially-driven mobile robot (depicted by a red car-like object) tasked with navigating to a uniformly randomized 2D goal location (green cylinder), denoted by $\mathbf{x}^g = [x^g, y^g]^\top$, with $x^g, y^g \in [-1, 1]$. En route to the goal, the robot must also avoid two regions within its environment: a traversable region, denoted by \mathcal{X}^s comprising purple discs shown in (b) (acting as a *soft* constraint), as well as obstacles in the set \mathcal{X}^h (represented by light blue cubes in (c)) that inhibit its motion upon collision (i.e., a *hard* constraint). These static hazards and obstacles incur *distinct* costs that the agent must minimize while learning to safely navigate to the goal.

Approximate Action-Value Iteration (hereafter, AVI) methods [3, 4] estimate the action-value function using a neural network trained on the temporal difference between a current estimate of the action value and an observed *target* value (i.e., the sum of the stage cost and the expected future action value). However, in high-variance environments, these methods often suffer from overestimation bias, where the action-value approximation predicts values that are significantly higher than the true values [5]. AVI methods based on quantile regression [5, 6] address this by learning a quantile distribution of Q-values instead of a single expected value, reducing overestimation and improving estimation accuracy. By updating value distributions across quantile fractions, quantile regression RL algorithms enhance exploration and stabilize policy learning [5].

While distributional RL mitigates overestimation bias, it struggles with instantaneous stage cost misspecification, leading to unsafe or suboptimal policies [1]. Modifying stage costs to encode safety constraints [7, 8] is common but challenging to tune, especially in environments with varied risks [9] (Fig. 1). Safe Policy Optimization [10] tackles this problem by enforcing safety through a separate network, which adds substantial computational overhead [11]. Furthermore, in dynamic environments, uncertainty from stochastic transitions, noise, perturbations, and model approximations can lead to brittle policies if not properly addressed [12]. To balance instantaneous stage cost optimization, safety adherence, and uncertainty mitigation – without the overhead of a separate cost network [10, 11] – we propose an RL framework that integrates risk-sensitive decision-making into approximate action-value iteration algorithms. Specifically, our approach penalizes safety violations by augmenting the quantile loss with a risk term derived from experience-based cost distributions.

Contributions and Paper Outline

- i. *Risk-regularized Quantile-Regression Based Approximate Action-Value Iteration (QR-AVI)* (Section 4): We enhance efficiency and safety by training a single quantile-based action-value iteration using a risk-regularized loss that integrates quantile regression with a penalty for safety violations (Section 4.3).
- ii. *Empirical uncertainty quantification via Kernel Density Estimation* (Sections 2.4 and 4.1): We approximate the cost distribution using Kernel Density Estimation (KDE), providing a probabilistic measure of safety constraints based on cost samples and enabling risk computation.
- iii. *Contraction & Finite-Time Convergence Guarantees* (Section 5): We prove the contraction of the risk-regularized Bellman operator, ensuring stable learning and finite-time convergence of the quantile-based action-value iteration algorithm to a stationary policy, leveraging the convexity of the quantile and risk loss terms.
- iv. *Modular RL Algorithmic Framework* (Section 4): We introduce a risk-sensitive QR-AVI algorithm (Algorithm 1) within a flexible and modular framework that integrates safety constraints into the quantile loss, making our approach applicable to other approximate RL methods.
- v. *Reach-Avoid Experiments (Section 6) & Comparative Study (Section 6.2)*: We evaluate a car-like agent in a dynamic reach-avoid task, comparing our algorithm to nominal and risk-neutral variants. Results show improved safety, lower quantile loss, and higher success rates.

2 Background

2.1 Reinforcement Learning (RL)

The terminology in this subsection follow [13]. We consider an *agent* as a decision-making entity that learns from interacting with an external *environment* through a sequence of *observations* (of its states and those of the environment), *actions*, and *costs*. At each discrete time step $t \in \mathbb{Z}_+$, the agent observes its current state as well as the current state of the environment, denoted for simplicity by the vector $\mathbf{x}_t = [\mathbf{x}_t^a, \mathbf{x}_t^e]^\top \in \mathcal{X} = \mathcal{X}_a \times \mathcal{X}_e$, where $\mathbf{x}_t^a \in \mathcal{X}_a$ represents the agent’s state, and $\mathbf{x}_t^e \in \mathcal{X}_e$ represents the environment’s state, i.e., every artifact within the environment outside the agent. The agent applies a control input $\mathbf{u}_t \sim \mu(\cdot | \mathbf{x}_t^a) \in \mathcal{U}$ according to a (randomized) policy or control law $\mu : \mathcal{X}_a \rightarrow \Delta(\mathcal{U})$, where \mathcal{U} denotes the control space representing the discrete set of permissible control inputs, $\Delta(\mathcal{U})$ is the set of distributions over the control space, and $\sum_{\mathbf{u}_t \in \mathcal{U}} \mu(\mathbf{u}_t | \mathbf{x}_t^a) = 1, \forall \mathbf{x}_t^a \in \mathcal{X}_a$. The agent also incurs a stage cost, $g_t = \phi(\mathbf{x}_t^a, \mathbf{u}_t) \in \mathbb{R}$, and transitions to a new state

according to $\mathbf{x}_{t+1}^a = f^a(\mathbf{x}_t^a, \mathbf{u}_t)$, where f^a is the state transition function. We define the system in which the agent interacts with the environment by a Markov Decision Process (MDP), given by the tuple $\mathcal{M} = (\mathcal{X}, \mathcal{U}, f^a, f^e, \phi, \gamma)$, where γ is a discount factor for expected future costs.

Example 1 (Running Example). *Consider the navigation task shown in Fig. 1. The robot’s state space, \mathcal{X}_a , comprises its wheel orientation, angular velocity, linear velocity, and acceleration, goal and obstacle lidar measurements, and Boolean indicators for obstacle collision events (the robot’s dynamics are outlined in Section 6). The control set, \mathcal{U} , is 2-dimensional and consists of the torques applied to the left and right wheels. The stage cost, $g_t \in \mathbb{R}$, is a function of the Euclidean distance between the robot and the prescribed goal location at each timestep (see Section 6). The robot must also avoid a keep-off region, \mathcal{X}^s and an obstacle region, \mathcal{X}^h , that both incur a violation cost distinct from g_t .*

Definition 1 (Safety). *A trajectory $\{\mathbf{x}_t^a, \mathbf{u}_t\}_{t=0}^T$ is deemed safe if it avoids environmental constraint violations, i.e., if*

$$\mathbf{x}_t^a \notin \mathcal{X}_{\text{unsafe}} = \mathcal{X}^s \cup \mathcal{X}^h \equiv \mathcal{X}_a \setminus \mathcal{X}_{\text{safe}}, \forall t, \quad (1)$$

where $\mathcal{X}_{\text{unsafe}}$ denotes the set of agent states corresponding to collisions and restricted regions, and $\mathcal{X}_{\text{safe}}$ is its complement.

Remark 1 (Enforcing Safety). *To avoid the computational challenges associated with deriving an explicit expression for $\mathcal{X}_{\text{safe}}$, we enforce safety via unique cost functions (specified for each safety constraint) that assign a positive scalar penalty whenever the agent enters an unsafe region:*

$$C_t := \begin{cases} c_t^s(\mathbf{x}_t^a) > 0, & \text{if } \mathbf{x}_t^a \in \mathcal{X}^s, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

To implicitly bias the agent toward safe behavior, we then minimize the cumulative sum of the violation costs, with the costs defined as in (2). See (26) for the full cost expression.

2.2 Action-Value Iteration (AVI)

In an MDP, the agent seeks a policy μ that minimizes the expected cumulative cost. The optimal cost-to-go function satisfies $J^*(\mathbf{x}) = \min_{\mathbf{u}_t} Q^*(\mathbf{x}_t, \mathbf{u}_t)$, where $Q^*(\mathbf{x}_t, \mathbf{u}_t)$ is the optimal action-value function. The action-value function $Q(\mathbf{x}_t, \mathbf{u}_t)$, which estimates the future cost from a given state-action pair, is updated iteratively using the Bellman recursion

$$Q(\mathbf{x}_t, \mathbf{u}_t) \leftarrow Q(\mathbf{x}_t, \mathbf{u}_t) + \alpha^t [\phi(\mathbf{x}_t, \mathbf{u}_t) + \gamma \min_{\mathbf{u}_{t+1}} Q(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) - Q(\mathbf{x}_t, \mathbf{u}_t)], \quad (3)$$

where $\phi(\mathbf{x}_t, \mathbf{u}_t)$ is the stage cost, $\alpha^t \in (0, 1]$ is the learning rate, and $\gamma \in (0, 1]$ is a discount factor. Since maintaining exact action values for all state-control pairs in large state-control spaces is impractical, we specialize our discussion to approximate value iteration methods in the next section.

2.3 Approximate AVI via Quantile Regression

Approximate action-value iteration methods replace the tabular representation with a function approximator, such as a θ -parameterized neural network, i.e., $Q(\mathbf{x}_t, \mathbf{u}_t) \approx Q(\mathbf{x}_t, \mathbf{u}_t; \theta)$, that is typically trained using data sampled uniformly from a sequence of state-control pairs, \mathcal{D} , containing tuples of the form $\{(\mathbf{x}_t, \mathbf{u}_t, g_t, \mathbf{x}_{t+1}, d_t)\}$, where d_t is a Boolean indicator variable that signifies if a learning episode has ended. The predicted Q-values (denoted by $Q(\mathbf{x}_t, \mathbf{u}_t; \theta)$) are updated by minimizing the loss function defined by

$$\mathcal{L}(\theta_i) = \mathbb{E}[(y_i - Q(\mathbf{x}_t, \mathbf{u}_t; \theta_i))^2], \quad i = 1, 2, \dots, B, \quad (4)$$

where B is the size of the batch sampled from \mathcal{D} , y_i is the i^{th} target action-value realized from a target network, $Q(\mathbf{x}_t, \mathbf{u}_t; \theta_i)$, with parameters, θ_i that are updated with a predetermined frequency to converge at a stable or low-variance value: $y_i := g_t + \gamma \min_{\mathbf{u}_{t+1}} Q(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}; \theta_i)$. To account for variability in the Q-value distribution, quantile regression-based approximate AVI (QR-AVI) methods [5, 6] estimate the target action-value distribution using a set of quantiles (e.g., the lower tail, median,

upper tail etc.) instead of predicting just the mean Q-value. Specifically, in the QR-AVI framework [6], the loss in (4) is replaced by the quantile regression loss:

$$\mathcal{L}_{\text{QR}}(\hat{\theta}_n) := \frac{1}{N_\tau} \sum_{n=1}^{N_\tau} \sum_{j=1}^{N_\tau} \rho_{\tau_n}^\kappa [y_j - \hat{\theta}_n(\mathbf{x}_t, \mathbf{u}_t)]. \quad (5)$$

In (5), $y_j = g_t + \gamma \hat{\theta}_j(\mathbf{x}_{t+1}, \mathbf{u}_{t+1})$ defines the j^{th} quantile (of the target action-value distribution) that depends on the system's (agent-environment) transition dynamics and stage cost, with $j \in \{1, \dots, N_\tau\}$, N_τ is the prescribed number of quantile fractions, and $\rho_{\tau_n}^\kappa$ is the asymmetric Huber loss (with parameter κ) corresponding to the uniformly-distributed quantile fraction, $\tau_n \in (0, 1)$ (δ_w is the Dirac delta function at $w \in \mathbb{R}$ and $n = 1, 2, \dots, N_\tau$):

$$\rho_{\tau_n}^\kappa(m) = |\tau_n - \delta_{\{m < 0\}}| \mathcal{L}_\kappa(m), \quad \tau_n = (n-0.5)/N_\tau \quad (6)$$

$$\mathcal{L}_\kappa(m) = \begin{cases} \frac{1}{2}m^2, & \text{if } |m| \leq \kappa, \\ \kappa(|m| - \frac{1}{2}\kappa), & \text{otherwise.} \end{cases} \quad (7)$$

Remark 2 (Parameters of \mathcal{L}_{QR}). *The Huber loss function, $\rho_{\tau_n}^\kappa$, in (5) serves to minimize the action-value distribution approximation error for the n -th quantile by combining the squared and absolute errors and switching between these errors based on a threshold parameter, κ . This switching strategy makes the Huber loss robust to outliers [6], which is an important requirement for real-world control applications with noisy or uncertain data. The Dirac function adjusts the magnitude of the error based on the sign of the desired cost-to-go (denoted by m) to differentiate between underestimation and overestimation of the action value distribution.*

2.4 Data-Driven Uncertainty Quantification

At each state, \mathbf{x}_t^a , since the agent can choose from multiple possible actions $\mathbf{u}_t \in \mathcal{U}$ according to a random policy, each random action thus incurs a non-deterministic stage cost given by $\phi(\mathbf{x}_t^a, \mathbf{u}_t)$, as well as a random safety violation cost not captured by the stage cost function (see Example 1 and Definition 1). We thus model the randomness in the safety violation costs *separately* by assuming C_t follows an unknown distribution with probability density, P_c , that must be learned from cost samples in the sampled batch, $\{C_t^{(1)}, C_t^{(2)}, \dots, C_t^{(|B|)}\}$, for each time step, where $C_t^{(e)}$ is the violation cost incurred at time step t in episode e . Due to the violation cost's dependence on the policy, the associated cost distribution at \mathbf{x}_t is thus conditioned on μ . We define the notion of a policy-conditioned cost distribution next.

Definition 2 (Policy-Conditioned Cost Distribution). *Given a policy, μ , and a cost distribution, P_c , the policy-conditioned cost distribution at state \mathbf{x}_t (hereafter denoted by $Z_c^\mu(\cdot \mid \mathbf{x}_t^a)$) is the cost distribution obtained by marginalizing over inputs*

$$\sum_{\mathbf{u}_t \in \mathcal{U}} \mu(\mathbf{u}_t \mid \mathbf{x}_t^a) P_c(\cdot \mid \mathbf{x}_t^a, \mathbf{u}_t), \quad (8)$$

where $P_c(\cdot \mid \mathbf{x}_t^a, \mathbf{u}_t)$ is a distribution of safety violation costs associated with the state control pair, $(\mathbf{x}_t^a, \mathbf{u}_t)$. For instance, if C_t is the cost incurred from violating a safety constraint, $P_c(C_t \mid \mathbf{x}_t^a, \mathbf{u}_t)$ provides the likelihood of incurring a particular cost C_t given the pair, $(\mathbf{x}_t^a, \mathbf{u}_t)$.

Remark 3 (Challenges with Computing $Z_c^\mu(\cdot \mid \mathbf{x}_t^a)$). *Equation (8) provides a formal framework that fully captures the cost distribution for a given state and under control inputs chosen according to a policy. Unfortunately, due to the unavailability of an exact form of $P_c(\cdot \mid \mathbf{x}_t^a, \mathbf{u}_t)$, a direct computation of a closed form expression for $Z_c^\mu(\cdot \mid \mathbf{x}_t^a)$ becomes infeasible. This challenge thus necessitates the need for an approximation of $Z_c^\mu(\cdot \mid \mathbf{x}_t^a)$, hereafter denoted as \hat{Z}_c^μ , from costs sampled from stored experience during training.*

2.5 Risk Model & Risk Measure Computation

Definition 3 (Risk Measure). *Consider the probability space, $(\Omega, \mathcal{F}, \hat{Z}_c^\mu)$, where $\Omega \supseteq \mathcal{X} \times \mathcal{U}$ is the sample space and \mathcal{F} is the σ -algebra over Ω . Suppose \mathcal{C} denotes the set of all cost random variables defined on Ω . Omitting the time argument for brevity, we define the risk measure as a*

function $\hat{\rho}_\beta : \mathcal{C} \rightarrow \mathbb{R}$ that assigns a real number representing the risk or variability of C , i.e., $\hat{\rho}_\beta$ captures the tail behavior of the cost distribution; $\beta \in (0, 1)$ is the confidence level representing the proportion of \hat{Z}_c^μ that is covered when computing $\hat{\rho}_\beta$.

Hereafter, we focus on *coherent* risk measures, widely used in risk-sensitive optimization, that satisfy key axioms: sub-additivity, positive homogeneity, translation invariance, monotonicity, and risk-utility duality [14, 15, 16]. Of these, we select the Conditional Value-at-Risk (CVaR), since it can distinguish between tail events [14] beyond β . Given the cost distribution \hat{Z}_c^μ realized from some estimation scheme (see Section 4.1), the expected cost at time step t is:

$$\mathbb{E}[C_t] = \frac{1}{B} \sum_{e=1}^B C_t^{(e)}. \quad (9)$$

Using (9), and specializing to the case where $\hat{\rho}_\beta$ is the β -level CVaR (CVaR_β), the risk measure is given by

$$\text{CVaR}_\beta(C_t) = \mathbb{E}[C_t \mid C_t \geq \text{VaR}_\beta(C_t)], \text{ where} \quad (10)$$

$$\text{VaR}_\beta(C_t) = \inf_{\star} \{ \star \in \mathbb{R} \mid \hat{Z}_c^\mu(C_t \leq \star) \geq \beta \} \quad (11)$$

is the Value-at-Risk or β^{th} percentile of \hat{Z}_c^μ .

3 Problem Formulation

Assume the setting in Section 2.1. Given an MDP representing the system, we wish to find a policy that minimizes the expected cumulative cost while minimizing the risk of safety constraint violations, that is, the *risk-sensitive* cumulative cost, where the risk is computed from an estimated (constraint-violation) cost distribution. Formally, we consider the following problem:

Problem 1. Given an MDP (Section 2.1), a stage cost function, ϕ , a cost distribution, \hat{Z}_c^μ , and an initial state, \mathbf{x}_0 , find a policy μ^* such that:

$$\mu^* = \arg \min_{\mu} \mathbb{E} \left[\sum_{t=0}^{T-1} \alpha^t \phi(\mathbf{x}_t, \mathbf{u}_t) \right] + \lambda [\hat{\rho}_\beta \left(\sum_{t=0}^{T-1} C_t \right)]. \quad (12)$$

In (12), $\hat{\rho}_\beta$ denotes the β -level CVaR corresponding to \hat{Z}_c^μ , estimated from cost samples collected during each episode, λ is a positive scalar regularization parameter that balances the cumulative stage cost minimization and risk minimization, and $C_t \sim \hat{Z}_c^\mu$ is the violation cost random variable. To solve Problem 1, we propose a risk-sensitive QR-AVI algorithm, with the risk computation enabled by Kernel Density Estimation (KDE). In Section 6.3.3, we provide a more application-centered argument in support of our choice of KDE, while Section 4 (appearing next) describes the specifics of our approach that draw on recent advances in risk-sensitive RL [12, 17, 18, 19, 20, 21].

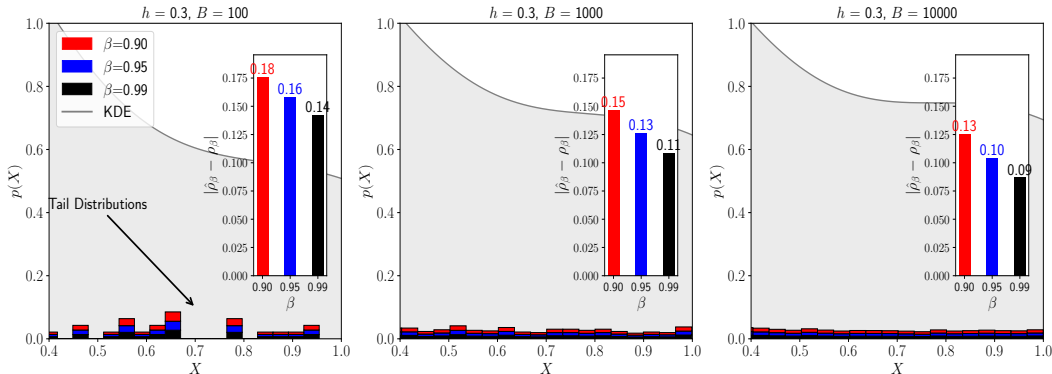


Figure 2: Graphing the KDE-estimated probability density ($p(X)$, shown as a gray line enclosing the gray-filled area) corresponding to samples of an uncertain variable (X). The inset bar plots show the evolution of the absolute error, $|\hat{\rho}_\beta - \rho_\beta|$, between the CVaR computed from $p(X)$ (i.e., $\hat{\rho}_\beta$) and the true CVaR (ρ_β) for an increasing number of KDE samples, $B = \{100, 1000, 10000\}$ and using a Gaussian kernel with a fixed bandwidth (h) of 0.3. The underlying data are from a heavy tailed distribution on $(0, 1]$, and the CVaR estimate moderately improves with the number of cost samples.

4 Approach: Risk-Regularized Quantile-Based AVI with KDE-Based Cost Distribution Estimation

4.1 KDE-Based Estimation of Z_c^μ

Following the discussions in Section 2.4 and for reasons outlined in Remark 3, we employ Kernel Density Estimation (KDE) [22], a technique that provides a practical method to approximate the policy-conditioned cost distribution (see Definition 2) from finite training samples. By leveraging observed costs during policy execution, KDE constructs an empirical distribution, \hat{Z}_c^μ (13), as a proxy for Z_c^μ , that enables the computation of distributional statistics essential for risk-sensitive decision-making. Denoting the cost samples observed from the training data at time step t by $\mathcal{C}_t := \{C_t^{(i)}\}_{i=1}^B$, we can write the likelihood of C_t as follows (k is the kernel function, and $h > 0$ is the bandwidth):

$$\hat{Z}_c^\mu(C_t | \mathbf{x}_t) = \frac{1}{Bh} \sum_{i=1}^B k\left(\frac{C_t - C_t^{(i)}}{h}\right), \quad C_t^{(i)} \in \mathcal{C}_t. \quad (13)$$

Remark 4 (KDE Caveats). *While KDE approximates Z_c^μ by smoothing over observed cost values, thus capturing both aleatoric uncertainty (due to the inherent stochasticity in the costs) and epistemic uncertainty (due to finite cost samples) as argued in [23], its estimation accuracy (depends on and) may improve albeit marginally with the number of samples [24]. For instance, limited experiments with a heavy-tailed distribution (see Fig. 2) reveal that increasing B hundredfold yields only a 5% improvement in estimation accuracy.*

4.2 Quantile Regression with Risk-Sensitive Loss Functions

As noted in Section 1, encoding safety with a monolithic cost function has limitations. To better capture safety risks, we propose the following risk-regularized loss function (\mathcal{L}) for QR-AVI (with $\lambda \in (0, 1)$):

$$\mathcal{L}(\theta) = (1 - \lambda)\mathcal{L}_{QR}(\theta) + \lambda\mathcal{L}_\rho(\hat{\rho}_\beta(C), c_{\max}), \quad (14)$$

where $\mathcal{L}_{QR}(\cdot)$ is the quantile regression loss given by (5), $\mathcal{L}_\rho(\cdot)$ is the loss corresponding to the risk (i.e., $\hat{\rho}_\beta(C)$; see (10)) computed over the cost distribution in (13), and c_{\max} is a positive cost threshold. By applying KDE on the observed cost samples \mathcal{C} , we can compute $\hat{\rho}_\beta(C)$ with respect to \hat{Z}_c^μ .

4.3 Risk-Sensitive Approximate Value Iteration with Learned Constraint-Violation Cost Distribution

With the KDE-estimated cost distribution, we construct the risk-sensitive loss function in (14) and train a function approximator that approximates the distribution of action-values using quantile regression. In the cost-minimizing context, two typical scenarios result from incorporating risk sensitivity in AVI, depending on β 's value: β values on $[0.9, 1)$ correspond to a *risk-averse* setting, while $\beta = 0.5$ corresponds to the *risk-neutral* setting, since the CVaR reduces to the expected value of the distribution [16].

Accordingly, to distinguish between these settings in our QR-AVI algorithm, each corresponding to a different instance of \mathcal{L} (see Table 1), we adopt the respective abbreviations: E-QR-AVI and ρ -QR-AVI. Our algorithm (Algorithm 1) trains a risk-sensitive function approximator for an action-value distribution by incorporating KDE-based risk estimation into the learning process. The key step that enables the computation of the risk loss is the storing of state-action-cost transitions in the replay buffer (Algorithm 1 of Algorithm 1). To update the Q-network, we compute the risk loss using the KDE-estimated distribution of costs sampled from experience.

Algorithm 1: Risk-Sensitive Approximate Value Iteration

Input: $T, \mathcal{D}, B, \gamma, \beta, \lambda, \kappa, \{\epsilon_t\}_{t=1}^T, \eta, N_\tau, c_{\max}$
Output: Q_θ

```

1 Initialize  $Q_\theta, Q_{\theta'}, \mathcal{D}, \{\tau_n\}_{n=1}^{N_\tau}, d_t = \text{false}$ 
2 Observe state  $\mathbf{x}_0$ 
3 for  $t = 0$  to  $T$  do
4   while  $d_t \neq \text{true}$  do //  $\mathbf{x}_{t+1}$  is not
      terminal
5      $\mathbf{u}_t \sim \epsilon_t$ -greedy
6     Execute  $\mathbf{u}_t$ , compute  $g_t$ , observe
        $\mathcal{C}_t, \mathbf{x}_{t+1}, d_t$ 
7     Store  $(\mathbf{x}_t, \mathbf{u}_t, g_t, \mathcal{C}_t, \mathbf{x}_{t+1}, d_t)$  in  $\mathcal{D}$ 
8     if  $|\mathcal{D}| \geq B$  then
9       Sample batch  $B$  from  $\mathcal{D}$ 
10      for
         $(\mathbf{x}_{t,j}, \mathbf{u}_{t,j}, g_{t,j}, \mathcal{C}_{t,j}, \mathbf{x}_{t+1,j}, d_{j,t})$ 
        in  $B$  do
11         $Q_{\text{target}} = g_{t,j} + \gamma(1 -$ 
           $d_{j,t}) \min_{\mathbf{u}_{t+1}} \mathbb{E}[\theta(\mathbf{x}_{t+1,j}, \mathbf{u}_{t+1}, \tau)]$ 
12      UpdateNetwork( $Q_{\text{target}}, \mathcal{C}_{t,j}, \text{args*}$ )
13       $\theta' \leftarrow \eta\theta + (1 - \eta)\theta'$  // Update
        target

```

Table 1: Training Loss Functions ($C \sim \hat{Z}_c^\mu$). Contributions in blue.

Algorithm	Loss Function (\mathcal{L})
AVI	$(y - Q(\mathbf{x}, \mathbf{u}))^2$
QR-AVI	$\mathcal{L}_{\text{QR}}(\hat{\theta}_n) := \frac{1}{N_\tau} \sum_{n=1}^{N_\tau} \sum_{j=1}^{N_\tau} \rho_{\tau_n}^\kappa[y_j - \hat{\theta}_n]$
\mathbb{E} -QR-AVI	$\lambda' \mathcal{L}_{\text{QR}} + \lambda[(\mathbb{E}[C] - c_{\max})^2]_+$ (see (9))
ρ -QR-AVI	$\lambda' \mathcal{L}_{\text{QR}} + \lambda[(\hat{\rho}_\beta(C) - c_{\max})^2]_+$ (see (10))

5 Theoretical Guarantees for Risk-Sensitive QR-AVI: Contraction, Fixed-Point Existence, & Finite-Time Convergence

Definition 4 (Risk-Sensitive Distributional Bellman Operator [25]). Let $Z \in \mathcal{Z}$ denote the expected cumulative cost distribution, with \mathcal{Z} given as $\mathcal{Z} = \{Z \mid \mathbb{E}[Z(\mathbf{x}, \mathbf{u})] < \infty, \forall \mathbf{x}, \mathbf{u}\}$. Let $\mathcal{Z}_c := \{Z_c^\mu \mid \mathbb{E}[Z_c^\mu(\mathbf{x}, \mathbf{u})] < \infty, \forall \mathbf{x}, \mathbf{u}\}$. Suppose $\hat{Z}_c^\mu \in \mathcal{Z}_c$ and $\hat{\rho}_\beta(\hat{Z}_c^\mu) \in \mathfrak{C} := \{C \mid C(\mathbf{x}^a) < \infty, \forall \mathbf{x}^a\}$, with $\hat{\rho}_\beta[\hat{Z}_c^\mu](\mathbf{x}^a) := \hat{\rho}_\beta(\hat{Z}_c^\mu[\cdot | \mathbf{x}^a])$. The risk-sensitive distributional Bellman operator, \mathcal{T}^β , corresponding to a β -level coherent risk measure, $\hat{\rho}_\beta$ (see Definition 3), is

$$\mathcal{T}^\beta Z(\mathbf{x}, \mathbf{u}) \stackrel{D}{=} g_t - \gamma \hat{\rho}_\beta[Z_c^\mu(\mathbf{x}_{t+1}, \mathbf{u}_{t+1})]. \quad (15)$$

$\stackrel{D}{=}$ denotes distributional equivalence, and $\mathbf{u}_{t+1} \sim \mu(\cdot | \mathbf{x}_{t+1})$.

Theorem 1 (Contraction of the Risk-Sensitive Bellman Operator with Cost-Based Risk Regularization). Assume the setting in Definition 4. Let Z_1 and Z_2 be two cumulative cost distributions, and suppose that $\hat{\rho}_\beta$ is non-expansive, i.e., that it satisfies the following relationship for two random costs, $C_1 \sim \hat{Z}_{1,c}^\mu \in \mathcal{Z}_c$ and $C_2 \sim \hat{Z}_{2,c}^\mu \in \mathcal{Z}_c$ (with $\hat{Z}_{1,c}^\mu \neq \hat{Z}_{2,c}^\mu$: $|\hat{\rho}_\beta(C_1) - \hat{\rho}_\beta(C_2)| \leq \|C_1 - C_2\|$). Let \hat{Z}_c^μ be related to Z by $\hat{Z}_c^\mu(\mathbf{u} | \mathbf{x}^a) = \Psi(Z(\mathbf{x}, \mathbf{u}))$, where $\Psi : \mathbb{R} \rightarrow \mathbb{R}_+$ is a coherent, Lipschitz continuous, and bounded function with Lipschitz constant, $L \in (0, 1)$. Then \mathcal{T}^β is a γ -contraction in the Wasserstein-1 (W_1) metric (see [6]), i.e., $W_1(\mathcal{T}^\beta Z_1, \mathcal{T}^\beta Z_2) \leq \gamma W_1(Z_1, Z_2)$.

Proof. Using (15), we can write:

$$\mathcal{T}^\beta Z(x, u) \stackrel{D}{=} \phi(\mathbf{x}, \mathbf{u}) - \gamma \hat{\rho}_\beta(Z_c^\mu(\mathbf{x}_{t+1}, \mathbf{u}_{t+1})), \quad (16)$$

so that by applying \mathcal{T}^β to Z_1 and Z_2 , we obtain:

$$\begin{aligned} \mathcal{T}^\beta Z_1 &= \phi(\mathbf{x}, \mathbf{u}) - \gamma \hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), \\ \mathcal{T}^\beta Z_2 &= \phi(\mathbf{x}, \mathbf{u}) - \gamma \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu). \end{aligned} \quad (17)$$

Taking W_1 , we obtain $W_1(\mathcal{T}^\beta Z_1, \mathcal{T}^\beta Z_2)$ as:

$$W_1(\phi(\mathbf{x}, \mathbf{u}) - \gamma \hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), \phi(\mathbf{x}, \mathbf{u}) - \gamma \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu)). \quad (18)$$

Since $\phi(\mathbf{x}, \mathbf{u})$ is deterministic, it cancels out, yielding:

$$W_1(-\gamma \hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), -\gamma \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu)). \quad (19)$$

By the non-expansiveness of $\hat{\rho}_\beta$:

$$W_1(\hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu)) \leq W_1(\hat{Z}_{1,c}^\mu, \hat{Z}_{2,c}^\mu). \quad (20)$$

Multiplying the L.H.S of (20) by γ , it follows that:

$$\begin{aligned} & W_1(-\gamma \hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), -\gamma \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu)) \\ &= \gamma W_1(\hat{\rho}_\beta(\hat{Z}_{1,c}^\mu), \hat{\rho}_\beta(\hat{Z}_{2,c}^\mu)) \leq \gamma W_1(\hat{Z}_{1,c}^\mu, \hat{Z}_{2,c}^\mu). \end{aligned} \quad (21)$$

By invoking the bounded-expectation assumption of the distributions in \mathcal{Z}_c (since $\hat{Z}_{1,c}^\mu, \hat{Z}_{2,c}^\mu \in \mathcal{Z}_c$), we get:

$$W_1(\hat{Z}_{1,c}^\mu, \hat{Z}_{2,c}^\mu) \leq W_1(Z_1, Z_2). \quad (22)$$

By the assumptions on Ψ , there exists an $0 < L < 1$ s.t.

$$W_1(\Psi(Z_1), \Psi(Z_2)) \leq L \cdot W_1(Z_1, Z_2), \quad (23)$$

i.e., Ψ is an L -contraction with respect to $\|\cdot\|$. Since $L < 1$ and $\|\mathcal{T}^\beta Z_1 - \mathcal{T}^\beta Z_2\| \leq \|Z_1 - Z_2\|$, applying (15) yields:

$$W_1(\mathcal{T}^\beta Z_1, \mathcal{T}^\beta Z_2) \leq \gamma W_1(Z_1, Z_2). \quad (24)$$

Hence, \mathcal{T}^β is a γ -contraction. \square

Corollary 1 (Existence of a Unique Cumulative Cost Distribution). *From Theorem 1, by Banach's fixed-point theorem [26], there exists a unique fixed point Z^* such that:*

$$Z^* = \mathcal{T}^\beta Z^*. \quad (25)$$

Remark 5 (Convergence of Risk-Sensitive QR-AVI). *Suppose \mathcal{X} and \mathcal{U} are continuous and bounded, with \mathcal{L} and \mathcal{L}_ρ convex, and assume that the agent explores the environment sufficiently often. Then, under a diminishing learning rate schedule such as $\alpha^t = k_\alpha/(t+1)$, where k_α controls the decay rate, the risk-sensitive QR-AVI algorithm will converge to a Pareto-optimal action-value in finite time. This result follows from the convexity of the combined quantile and risk-sensitive losses, which guarantees that any local Pareto optimum is also a global one [27], together with standard stochastic approximation results that ensure convergence under diminishing step sizes and sufficient exploration [13].*

6 Experiments

In this section, we provide a comprehensive overview of our experimental study, detailing our simulation setup, followed by our training and evaluation procedures, and concluding with comparisons with a risk-neutral variant.

6.1 Safety-Gymnasium Learning Environment

In this experiment, we train a differentially-driven mobile robot (depicted as a red wheeled car in Fig. 1) to navigate to a randomized goal location ($\mathbf{x}^g \in \mathbb{R}_2$) while minimizing the cost of traversing restricted areas and avoiding obstacles. The robot is equipped with a lidar sensor to measure distances to various environmental features such as obstacles and the goal. The state space, control space, and

the observations that the agent makes, as well as the goal of the experiment, are outlined below:

$$\mathcal{X} = \{\mathbf{x}_t = (\mathbf{x}_t^a, \mathbf{x}^s, \mathbf{x}^h, \mathbf{x}^g) \mid t = 1, 2, \dots, T\} \quad (\text{state space}) \quad (26a)$$

$$\mathcal{U} = \{\mathbf{u}_t = [v_t^L, v_t^R] \mid t = 1, 2, \dots, T\} \quad (\text{control space}) \quad (26b)$$

$$C_t = \begin{cases} c_t^s(\mathbf{x}_t^a) = c_s \sim \mathcal{U}_{[0,1]} \mathbb{I}_{\mathcal{X}^s}(\mathbf{x}_t^a), \\ c_t^h(\mathbf{x}_t^a) = c_h \sim \mathcal{U}_{[0,1]} \mathbb{I}_{\mathcal{X}^h}(\mathbf{x}_t^a), \quad c_h > c_s, \end{cases} \quad (\text{safety-violation costs}) \quad (26c)$$

$$f^a(\mathbf{x}_t^a, \mathbf{u}_t) = \begin{cases} x_{t+1} = x_t + \frac{r}{2}(v_t^R + v_t^L) \cos \theta_t, \\ y_{t+1} = y_t + \frac{r}{2}(v_t^R + v_t^L) \sin \theta_t, \\ \theta_{t+1} = \theta_t + \frac{r}{2d_w}(v_t^R - v_t^L), \end{cases} \quad (\text{transition dynamics}) \quad (26d)$$

In (26), $\mathbf{x}_t^a = [x_t, y_t, \theta_t]^\top$ defines the robot's state vector comprising the position of its t -step center of curvature and orientation (θ_t), and \mathbf{x}^s and \mathbf{x}^h respectively denote the position of the restricted region markers and obstacles. v_t^L and v_t^R are the robot's left and right wheel (linear) velocities, r is its wheel radius, d_w is the half-distance between the robot's two wheels, and $\mathbb{I}_{\mathcal{X}^*}$ is an indicator function. The environment contains 10 restricted region markers, 10 obstacles scattered around the map, and a single goal location. The restricted regions, goal, and obstacle locations are randomized according to a uniform distribution, $\mathcal{U}_{(a,b)}$, at the beginning of each episode (with $a, b \in [-1, 1]$) and remain static (and hence do not depend on t in (26a)) till the episode ends, i.e., after a prescribed number of time steps, T .

The observation space consists of 72 dimensions, with the last 48 representing 16 lidar measurements of the distance to the goal, restricted regions, and obstacles within a 3-meter range. Restricted areas are traversable and less severe constraints to avoid, while obstacles (cubes) are hard constraints that must be strictly avoided, incurring higher scalar costs ($c_h > c_s$; see (26c) for the definition of each cost variable). The agent must minimize the cumulative cost of violating these constraints over a training episode, as in the following equation: $\sum_{t=0}^{T-1} (c_h + c_s)$. As such, the optimal policy may not always be able to ensure total avoidance of the obstacles or restricted areas, and must rather minimize this cost. There are thus three levels of uncertainty arising from randomness or noise in: *lidar measurements*, *robot drift*, and *randomization of environment entities*. An episode ends after the prescribed maximum time horizon, T , setting a terminal state indicator, d , to 1. During an episode, the robot may reach as many goals as possible.

6.1.1 Training and Evaluation

The agent learns to minimize the stage cost, defined by the change in Euclidean distance to the static goal, \mathbf{x}^g , between consecutive time steps, as in:

$$g_t = -\gamma(\|\mathbf{x}_t^a - \mathbf{x}^g\|) - \|\mathbf{x}_{t-1}^a - \mathbf{x}^g\| - c_g,$$

where c_g is a positive scalar. To encourage exploration, we applied an epsilon-greedy linear exploration schedule, i.e., $\max\{t(\epsilon_T - \epsilon_0)\Delta t^{-1}, \epsilon_T\}$. To train all permutations of the QR-AVI algorithm (with $N_\tau = 32$ quantile fractions), we employed a feedforward neural network with three fully-connected (dense) layers containing two hidden layers (of dimension 120 and 84, respectively) with ReLU activation functions, and an output layer that produces values for each (control) input-quantile pair at each time step.

For the KDE cost distribution estimation, we used a Gaussian kernel function with a bandwidth determined by *Scott's rule* [28], i.e., $h = B^{0.2}$, and trained ρ -QR-AVI for $\beta = 0.9$ and 0.95 . On Fig. 3, we plot training curves for the expected cumulative cost Fig. 3a, risk Fig. 3b, and quantile loss Fig. 3c for all QR-AVI variants. We evaluated the models using five random seeds (0, 5, 10, 15, 20), with 20 episodes per seed, resulting in a total of 100 test episodes. Each seed ensures reproducibility by initializing the random number generator. Next, we computed the average action value, constraint violation cost, number of successful episodes, and goal success rate over all seeds to compare the performance of the algorithms. Here, the success rate represents the number of times the agent reaches the goal, averaged over all 100 test episodes. Table 2 summarizes our RL hyperparameters.

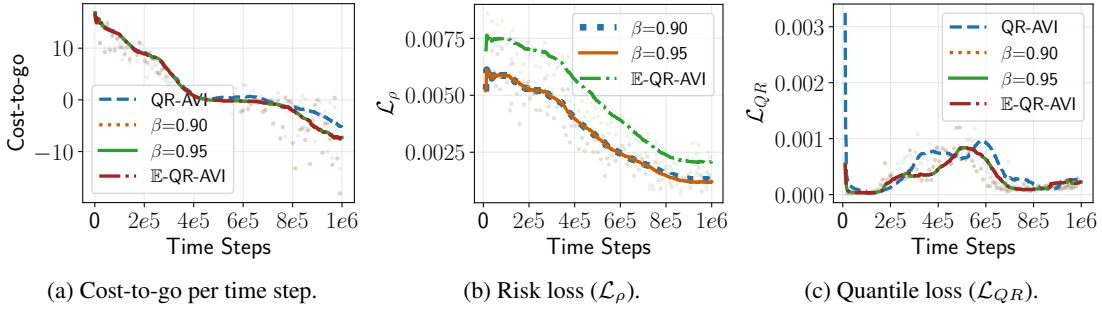


Figure 3: Evolution of the (training) expected cumulative cost, risk loss, and quantile loss for the reach-avoid navigation task.

Table 2: Training and Evaluation Hyperparameters

Param.	Value	Param.	Value	Param.	Value
B, α	128, $2.5e-4$	c_{\max}, β	0.1, $\{.9, .95\}$	γ, N_τ, T	0.99, 32, 1000
$\epsilon_0, \epsilon_T, \kappa$	1.0, 0.05, 1.0	$ \mathcal{D} $	$0.5e6$	Upd. Freq.	10 (train), 500 (target)

6.2 Juxtaposition with the Risk-Neutral Algorithm

We compare the risk-sensitive QR-AVI against the nominal and expected-value configurations across evaluation metrics. Table 3 juxtaposes our risk-sensitive adaptation with the baselines on the training and evaluation metrics. From the plots, we notice a marked improvement in the average expected cumulative cost for the risk-sensitive QR-AVI than the baseline, with a mostly consistent cost evolution. The agent demonstrates a success rate (that is approximately **37.66%** and **87.45%** higher per episode) than the baselines (nominal and risk-sensitive, respectively). This is reflected in both the increased success rate and the higher total number of successes across all scenarios and episodes.

Table 3: Comparison of success metrics and loss values across experiments. Mean and standard deviation of the evaluation cost were computed over 100 test episodes. Results using our method are highlighted in gray.

Alg.	Avg. Eval. Cost-to-go	Constraint-Violation Cost	Quantile Loss (Avg.)	Total Loss	Total Goals Reached	Normalized Success Rate (%)
QR-AVI	-0.01	0.05 ± 0.01	0.0004	0.0004	239	50%
\mathbb{E} -QR-AVI	-0.01	0.06 ± 0.01	0.0003	0.0032	329	75%
ρ -QR-AVI $^{\beta=0.9}$	-0.01	0.05 ± 0.01	0.0003	0.0023	448	100%
ρ -QR-AVI $^{\beta=0.95}$	-0.01	0.05 ± 0.01	0.0003	0.0023	448	100%

6.3 Discussions

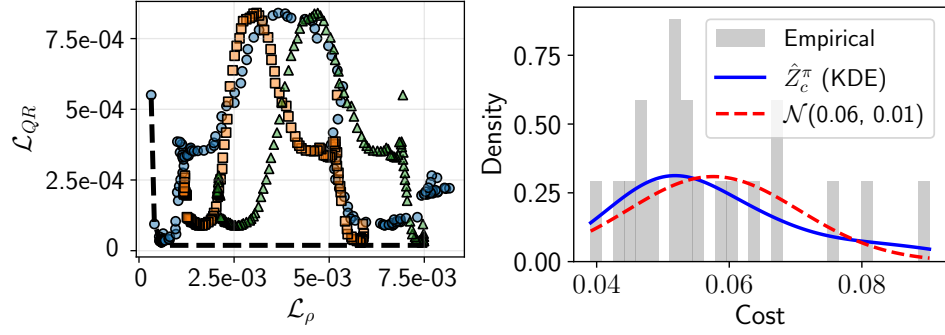
Here, we expound on the findings presented in the foregoing subsections and highlight a few implementation details.

6.3.1 Training and Evaluation Metrics

Comparing the average quantile losses (Table 3, column 4) reveals that all algorithms achieve near-identical performance, with average losses ranging between $3e-4$ and $4e-4$. The nominal QR-AVI exhibits the highest quantile loss, while the risk-sensitive ($\beta = 0.9$ and $\beta = 0.95$) and expected-value variants achieve slightly lower values of $3e-4$. Comparisons between ρ -QR-AVI and \mathbb{E} -QR-AVI show no significant difference in quantile loss, indicating comparable regression accuracy. Both variants of ρ -QR-AVI also yield similar returns and costs to the baselines but with a substantially higher success rate.

6.3.2 Risk-Performance Trade-offs

Given that we trained the QR-AVI model on a single neural network, a natural question about the performance-risk trade-off arises. Here, performance is quantified by the minimum quantile loss. The



(a) Pareto front (dashed line) corresponding to data points of the quantile loss (\mathcal{L}_{QR} , y-axis) and risk loss (\mathcal{L}_ρ , x-axis), for the risk-sensitive ($\beta = 0.95$) and risk-neutral ($\beta = 0.06$) cases. (b) Comparing the *tails* of the KDE-estimated cost points of the quantile loss (\mathcal{L}_{QR} , y-axis) and risk loss (\mathcal{L}_ρ , x-axis), for the risk-sensitive ($\beta = 0.95$) and risk-neutral ($\beta = 0.06$) cases. \bullet ; $\beta = 0.9$ \square and risk-neutral (\triangle) cases.

Figure 4: Pareto front and cost distribution approximation.

Table 4: β vs. the tail probability difference (Δ_{tail}) between \hat{Z}_c^μ (KDE-based) and $\mathcal{N}(0.06, 0.01)$.

β	Δ_{tail}
0.90	0.042
0.95	0.041
0.99	0.027

Pareto curve, presented in Fig. 4a, depicts the trade-off between the quantile (\mathcal{L}_{QR}) and risk losses (\mathcal{L}_ρ) for the expectation and risk-sensitive configurations of QR-AVI. The plot shows that \mathbb{E} -QR-AVI achieves the lowest quantile loss, indicating better accuracy in predicting cumulative cost, but at the expense of a significantly higher risk loss, highlighting its limitations in risk-sensitive settings. In contrast, the risk-sensitive configurations with $\beta = 0.9$ and $\beta = 0.95$ prioritize safety by achieving lower risk loss values while incurring moderate increases in quantile loss. For safety-critical tasks, the setting with $\beta = 0.95$ is preferable due to its low risk loss, despite a marginally higher quantile loss.

6.3.3 Determining a Fitting Cost Distribution Approximation

Due to environmental randomization, batch costs from the replay buffer may produce empirical distributions with light *right* tails (Fig. 4b) and negligible probability mass for extreme costs, leading to non-informative upper quantiles that coincide with the mean (Table 4). We thus favored KDE over simpler methods like softmax or the normal distribution for a more accurate cost distribution approximation.

7 Conclusions

We introduced a risk-sensitive quantile-based action-value iteration algorithm that balances safety and performance by augmenting the quantile loss with a risk term encoding safety constraints. Our results show that risk sensitivity preserves quantile regression accuracy and ensures consistent performance with tunable risk aversion. The method guarantees convergence to a unique risk-sensitive cost distribution, providing a theoretical foundation. The risk measure is compatible with any off-policy RL model and can be integrated into gradient ascent. Future work will explore dynamic risk parameter adjustments for improved trade-offs in varying conditions.

Acknowledgments

This work was partially supported by the Army Research Laboratory Cooperative Agreement No. W911NF-23-2-0040, a Northrop Grumman Corporation grant, and by the Lockheed Martin Chair in Systems Engineering.

References

- [1] A. Pan, K. Bhatia, and J. Steinhardt. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models. *arXiv preprint arXiv:2201.03544*, 2022.
- [2] D. K. Beyer, D. A. Dulo, G. A. Townsley, and S. S. Wu. Risk, Product Liability Trends, Triggers, and Insurance in Commercial Aerial Robots. In *We Robot Conference on Legal & Policy Issues Relating to Robotics. University of Miami School of Law*, volume 4, 2014.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013.
- [4] H. Van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
- [5] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. *arXiv*, 2020. doi: [10.48550/arXiv.2005.04269](https://doi.org/10.48550/arXiv.2005.04269).
- [6] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional Reinforcement Learning with Quantile Regression. *arXiv preprint arXiv:1710.10044*, Oct. 2017.
- [7] G. N. Tasse, T. Love, M. Nemecek, S. James, and B. Rosman. ROSARL: Reward-Only Safe Reinforcement Learning, 2023.
- [8] G. Thomas, Y. Luo, and T. Ma. Safe Reinforcement Learning by Imagining the Near Future. *NeurIPS*, 34:13859–13869, 2021.
- [9] N. P. Farazi, B. Zou, T. Ahamed, and L. Barua. Deep Reinforcement Learning in Transportation Research: A Review. *Transportation Research Interdisciplinary Perspectives*, 11:100425, 2021.
- [10] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang. Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark. *NeurIPS*, 36, 2023.
- [11] R. Fakoor, P. Chaudhari, and A. J. Smola. P3O: Policy-on Policy-off Policy Optimization. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pages 1017–1027. PMLR, 2020. URL <https://proceedings.mlr.press/v115/fakoor20a.html>.
- [12] C. Ying, X. Zhou, H. Su, D. Yan, N. Chen, and J. Zhu. Towards Safe Reinforcement Learning via Constraining Conditional Value-at-Risk, 2022.
- [13] D. P. Bertsekas. Model Predictive Control and Reinforcement Learning: A Unified Framework Based on Dynamic Programming, 2024. URL <http://arxiv.org/abs/2406.00592>.
- [14] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent Measures of Risk. *Mathematical Finance*, 9(3):203–228, 1999. ISSN 1467-9965. doi:[10.1111/1467-9965.00068](https://doi.org/10.1111/1467-9965.00068).
- [15] D. Nass, B. Belousov, and J. Peters. Entropic Risk Measure in Policy Search. In *IROS*, pages 1101–1106. IEEE, 2019.
- [16] R. T. Rockafellar. Risk and Utility in the Duality Framework of Convex Analysis. *From Analysis to Visualization: A Celebration of the Life and Legacy of Jonathan M. Borwein, Callaghan, Australia, September 2017*, pages 21–42, 2020.
- [17] C. Mavridis, E. Noorani, and J. S. Baras. Risk Sensitivity and Entropy Regularization in Prototype-Based Learning. In *2022 30th Med. Conf. on Control and Automation (MED)*, pages 194–199. IEEE, 2022.
- [18] E. Noorani and J. S. Baras. Risk-Sensitive Reinforcement Learning: A Monte Carlo Policy Gradient Algorithm for Exponential Performance Criteria. In *CDC*. IEEE, 2021.
- [19] E. Noorani and J. S. Baras. Risk-Sensitive Reinforcement Learning and Robust Learning for Control. In *CDC*. IEEE, 2021.

- [20] E. Noorani, C. Mavridis, and J. Baras. Risk-Sensitive Reinforcement Learning with Exponential Criteria. *arXiv preprint*, 2022.
- [21] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria, Apr. 2017.
- [22] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2017. ISBN 978-1-315-14091-9. doi:10.1201/9781315140919.
- [23] E. Hüllermeier and W. Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning*, 110(3):457–506, Mar. 2021. ISSN 1573-0565. doi:10.1007/s10994-021-05946-3.
- [24] Y.-C. Chen. A Tutorial on Kernel Density Estimation and Recent Advances. *Biostatistics & Epidemiology*, 2017. ISSN 2470-9360.
- [25] S. H. Lim and I. Malik. Distributional Reinforcement Learning for Risk-Sensitive Policies. *Advances in Neural Information Processing Systems*, 35:30977–30989, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/c88a2bd0e793550d0e885aa6e31ca277-Abstract-Conference.html.
- [26] A. T. Bharucha-Reid. Fixed Point Theorems in Probabilistic Analysis. *Bulletin of the American Mathematical Society*, 82(5):641–657, 1976. ISSN 0002-9904, 1936-881X. doi:10.1090/S0002-9904-1976-14091-8.
- [27] K. Van Moffaert and A. Nowé. Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.
- [28] D. W. Scott and G. R. Terrell. Biased and Unbiased Cross-Validation in Density Estimation. *Journal of the American Statistical Association*, 82(400):1131–1146, 1987. ISSN 0162-1459. doi:10.2307/2289391.