`CDRTrainer`:
Safe Reinforcement Learning with Human Feedback and Expert Demonstrations for 2D Aircraft Conflict Resolution

Clark Borst, TU Delft
https://github.com/clarkborst/CDRTrainer

February 2026

This document provides a high-level and implementation-aligned description of the reinforcement learning (RL) approach taken in a 2D aircraft conflict resolution example. The system combines classical Q-learning with linear function approximation, a geometrically informed action-shielding mechanism, human-feedback mechanisms (pairwise preference or rating), and expert demonstrations that directly nudge Q-values. In technical terms, the approach taken can best be described as (shielded) off-policy Q-learning in a constrained Markov Decision Process (MDP). The objective is to learn safe, human-aligned maneuvering policies that avoid loss of separation while minimizing additional flown track miles and aligning with operationally accepted ways in solving conflicts.

***Disclaimer**: The contents of this document and the associated materials are intended solely for educational and research purposes. They do not represent operational guidance, certified procedures, or validated aviation decision-support tools, and must not be used for real-world air traffic control or flight operations.*

1 Problem Setting and Goal

Let's consider a 2D horizontal-plane aircraft encounter between an *ownship* (blue) and an *intruder* (red), see Figure 1. At each decision time step t, the Reinforcement Learning (RL) agent selects a heading-change action

$$a_t \in \{-50^\circ, -40^\circ, \ldots, 0^\circ, \ldots, +40^\circ, +50^\circ\}.$$

Additionally, the RL agent can optionally perform speed control by small speed changes,

$$\Delta V \in \{-10,\ 0,\ +10\}\ \text{kts}, \tag{1}$$

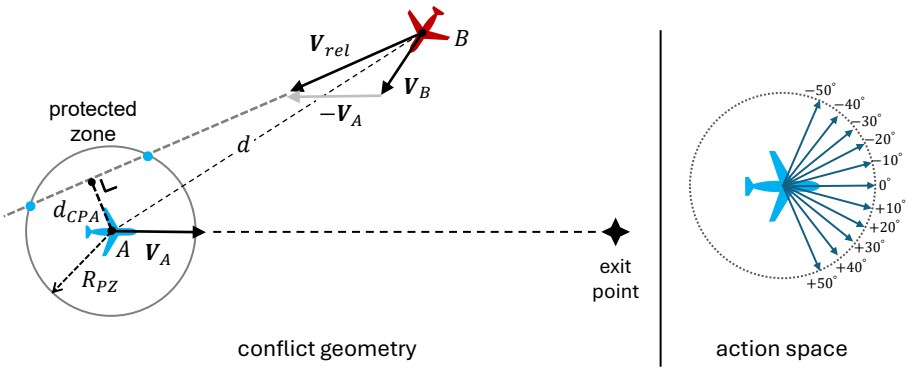

Figure 1: Conflict geometry and action space.

such that the combined action set is the Cartesian product

$$\mathcal{A} = \{(\Delta\psi, \Delta V)\} = \{\Delta\psi\} \times \{\Delta V\}, \tag{2}$$

yielding $|\mathcal{A}| = 11 \times 3 = 33$ discrete actions. When speed control is disabled, the executable action space is restricted to the heading-only subset with $\Delta V = 0$. Speed acts as a secondary degree of freedom to modulate relative geometry (e.g. t_{CPA} and d_{CPA}) while preserving the ability to recapture the original route.

The goal of the agent is to avoid violating the minimum allowed separation of $R_{\text{PZ}} = 5$ NM, minimize the route and heading deviation from the original trajectory, and minimize the number of (heading and/or speed) maneuvers taken.

2 Q-Learning with Linear Function Approximation

A linear Q-learning with function approximation is employed, where the action–value function is parameterized as:

$$Q(s, a) = \mathbf{w}_a^\top \boldsymbol{\phi}(s), \tag{3}$$

with feature vector $\boldsymbol{\phi}(s) \in \mathbb{R}^F$ and action-specific weight vectors $\mathbf{w}_a$. Q-learning with linear function approximation can be regarded as a linear neural network with only one layer. The output layer has no non-linear activation function (like ReLU or Sigmoid) and it performs a direct linear projection, mapping input features to Q-values.

The standard Q-learning update for a transition (s, a, r, s') is:

$$\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a), \tag{4}$$

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha\, \delta\, \boldsymbol{\phi}(s), \tag{5}$$

where α is the learning rate and γ the discount factor. Exploration is generally performed with ε-greedy over the action set. Epsilon-greedy is a strategy where the RL agent chooses the action it currently believes is best most of the time (exploitation),

but occasionally picks a random action (exploration). By introducing a small chance (epsilon) to try something new, the agent ensures it doesn't get stuck in a suboptimal routine and can discover potentially better rewards it hasn't found yet.

More advanced RL setups, like Deep Q-Networks (DQN), use (deep) neural networks to automatically learn rich state representations, enabling them to scale to high-dimensional and complex environments, but at the cost of increased computational demand, reduced transparency, and greater training instability. The key trade-offs to be made is between expressiveness and scalability on the one hand, and simplicity, stability, and interpretability on the other.

Explanation

Q-learning with linear function approximation can regarded as learning the parameters of a straight line that predicts how good an action is. Instead of storing a separate value for every state–action pair, the agent learns a linear equation whose parameters determine the Q-value. This allows the agent to generalize across similar states and actions and makes learning scalable to large problems.

Consider first the simplest linear equation,

$$y = ax + b,$$

where a controls the slope and b the offset. In Q-learning:

- y is the Predicted Q-value (how good it thinks an action is)
- x is the Feature (what the agent sees, e.g., "Distance to Goal")
- a is the Weight (how much importance is given to that feature)
- b is the Bias/Intercept (the baseline value).

In calculus, the gradient is just the derivative with respect to the variables that can be changed. In Q-learning, the environment (x) cannot be changed, only the internal settings (a and b). Thus, the gradient ∇ with respect to a is x and the gradient with respect to b is 1.

Suppose this line predicts a quantity of interest (for example, the value of taking an action). If the prediction is too low compared to a desired target value y_{target}, we want to increase the output of the line for the current input x; if it is too high, we want to decrease it. The prediction error is

$$\delta = y_{\text{target}} - y_{\text{pred}}.$$

To reduce this error, the parameters are adjusted in the direction that increases or decreases the prediction appropriately. As mentioned before, the gradient of y with respect to a is x, and with respect to b is 1. A gradient update there-

fore takes the form

$$a \leftarrow a + \alpha\,\delta\,x, \qquad b \leftarrow b + \alpha\,\delta,$$

where α is a learning rate. Intuitively, if the prediction is too small ($\delta > 0$), both parameters are increased; if it is too large ($\delta < 0$), they are decreased. Over time, the line rotates and shifts so that its predictions better match the desired targets.

Now consider a slightly more complex linear model with multiple inputs,

$$y = a_1 x_1 + a_2 x_2 + b.$$

This is still a linear function, but it depends on multiple features. Each parameter a_i determines how strongly its corresponding feature influences the output. If the prediction is incorrect, the error

$$\delta = y_{\text{target}} - y_{\text{pred}}$$

is computed and each parameter is updated proportionally to both the error and its associated feature:

$$a_1 \leftarrow a_1 + \alpha\,\delta\,x_1, \quad a_2 \leftarrow a_2 + \alpha\,\delta\,x_2, \quad b \leftarrow b + \alpha\,\delta.$$

Features that are large in the current example produce larger updates, allowing the model to learn which inputs matter most and how strongly they should influence predictions.

Q-learning with linear function approximation follows exactly this principle, but the predicted quantity is the action-value function,

$$Q(s,a) = \mathbf{w}^\top \boldsymbol{\phi}(s,a),$$

where $\boldsymbol{\phi}(s,a)$ is a feature vector describing the state–action pair and $\mathbf{w}$ is a vector of learnable parameters. Instead of a supervised target, Q-learning uses a temporal-difference (TD) target based on observed reward and estimated future value:

$$y_{\text{TD}} = r + \gamma \max_{a'} Q(s',a').$$

The prediction error becomes

$$\delta = y_{\text{TD}} - Q(s,a),$$

and since the gradient of $Q(s,a)$ with respect to the weights ($\nabla_w Q(s,a)$) is simply the feature vector $\boldsymbol{\phi}(s,a)$, the update rule is

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\,\delta\,\boldsymbol{\phi}(s,a).$$

Intuitively, if an action leads to better outcomes than expected, the weights associated with its features are increased so that similar situations will receive higher Q-values in the future. If the outcome is worse than expected, those weights are decreased. Through repeated updates, the linear function gradually reshapes itself so that it assigns high values to actions that lead to long-term reward and low values to those that do not. It is essentially online linear regression.

2.1 Feature Set and L2 Regularization

In general, the state representation $\phi(s)$ seen by the RL agent can be as large and expressive or as small, yet informative as possible. For example, it can include raw absolute (x, y) and relative positions (dx, dy), absolute and relative velocities, distance to the intruder, direction and distance to the exit waypoint, normalised CPA time/distance, and perhaps a binary predicted-conflict flag. While expressive, such a representation has many redundant and strongly correlated elements, which can hinder data efficiency and stability in linear function approximation.

The performance of Q-learning with linear function approximation depends critically on the choice of feature representation, commonly referred to as *feature engineering*. Well-designed features embed domain knowledge into the learning process and allow the agent to generalize effectively across similar situations despite limited model capacity. In the conflict-resolution setting, features are commonly constructed from *relative* geometry between ownship and intruder, such as distance, closing rate, and relative bearing, which naturally capture the invariances of the problem. Radial basis–style encodings of these relative quantities emphasize local interactions (e.g. proximity and convergence) and would enable smooth generalization over continuous state spaces, making them particularly suitable for safety-critical encounter dynamics.

Based on domain knowledge and the goal (i.e., desired behavior) that needs to be learned, it is possible to design the feature space to a set that captures the essential decision variables for conflict resolution and trajectory restoration:

$$\phi(s) = \left[d,\ \dot{d},\ \sin\theta_{\text{rel}}, \cos\theta_{\text{rel}},\ d_{\text{CPA}},\ t_{\text{CPA}},\ s_{\perp},\ s_{\parallel},\ \sin\theta_{\text{orig}}, \cos\theta_{\text{orig}},\ 1\right]^{\top}.$$

Here, d is the normalized intruder distance, $\dot{d}$ is the normalized closing speed (radial rate of change of d), θ_{rel} is the bearing of the intruder relative to the ownship heading, d_{CPA} is the CPA margin relative to the required minimum separation, t_{CPA} is the normalized urgency when a conflict is present, $s_{\perp}$ the normalized cross-track deviation, $s_{\parallel}$ the normalized along-track distance remaining parallel to the original route, θ_{orig} is the angular difference between the current heading and the original route. The last component is a bias feature. The bias feature provides a constant offset that allows the Q-function to represent a baseline value for each action independent of the state, ensuring the model is not forced to predict a zero value when all other features are zero.

Feature normalization is essential because Q-learning updates scale directly with feature values. Without normalization, large-magnitude features (or the bias term) dominate learning, even if they are not informative. Normalizing features keeps updates balanced, improves convergence stability, and makes learned feature importance meaningful and interpretable. As a rule of thumb, normalized features should lie in a bounded range (e.g., $[-1, 1]$ or $[0, 1]$), so that learning dynamics are governed by structure in the data and not by arbitrary units.

Conceptually, the chosen feature set is designed to (i) separate conflict-resolution from recovery behavior, (ii) avoid hard saturation, and (iii) provide smooth gradients for learning. Here, the features are calculated are follows:

Conflict resolution features

- Normalized intruder range:

$$\phi_d = 1 - \exp\left(-\frac{d}{D_{\text{scale}}}\right) - 0.5 \tag{6}$$

  where d is the intruder distance. Here, D_{scale} can be chosen based on an encounter distance that is typical for the chosen environment (e.g., airspace sector).

- Intruder closing rate along line-of-sight:

$$\phi_{\dot{d}} = \frac{\mathbf{v}_{\text{rel}} \cdot \hat{\mathbf{r}}}{V_{\text{max}}} \tag{7}$$

  where $\mathbf{v}_{\text{rel}}$ is the relative speed, $\hat{\mathbf{r}}$ the radial unit vector, and V_{max} the maximum relative speed (i.e. when aircraft approach each other head on) that is typical for the aircraft speeds in the environment (e.g., 450 kts en-route, resulting in $V_{\text{max}} = 900$ kts).

- Relative bearing to intruder:

$$(\sin\theta_{\text{rel}}, \cos\theta_{\text{rel}}) \tag{8}$$

  The bearing of the intruder relative to the ownship heading is computed as:

$$\theta_{\text{rel}} = \text{wrap}\big(\text{atan2}(dy, dx) - \psi_{\text{own}}\big),$$

  where ψ_{own} is the ownship heading in radians and wrap$(\cdot)$ maps angles to $(-\pi, \pi]$. Rather than using θ_{rel} directly, the implementation encodes this information via sine and cosine:

$$(\sin\theta_{\text{rel}}, \cos\theta_{\text{rel}}),$$

  which avoids discontinuities at $\pm\pi$ and yields smoother gradients for learning.

- CPA distance margin relative to minimum required separation:

$$\phi_{\text{dCPA}} = \tanh\left(\frac{d_{\text{CPA}} - d_{\text{sep}}}{d_{\text{sep}}}\right) \tag{9}$$

The dCPA margin is transformed with a tanh to produce a bounded $[-1, 1]$ feature that is most sensitive near the separation threshold, ensuring stable learning while emphasizing safety-critical differences without letting large safe distances dominate the Q-function.

- time to CPA urgency (gated):

$$\phi_{\text{tCPA}} = \begin{cases} (1 - t_{\text{CPA}}), & \text{if conflict predicted} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Here, t_{CPA} is normalized by the conflict look-ahead horizon. The feature is then gated by predicted conflict so it encodes urgency only when a conflict is relevant, preventing spurious gradients in safe states and cleanly separating conflict-resolution behavior from recovery behavior.

Route restoration features

- Original-track alignment:

$$(\sin\theta_{\text{orig}}, \cos\theta_{\text{orig}}) \tag{11}$$

Similarly to the relative bearing to the intruder aircraft, the relative bearing to the original route's heading is encoded via a via sine and cosine.

- Cross-track deviation:

$$\phi_{s_\perp} = 1 - \exp\left(-\frac{|y_\perp|}{X_{\text{scale}}}\right) \tag{12}$$

The cross-track distance measures the lateral deviation of the ownship from its original intended track, computed as the perpendicular distance to the initial heading line; it directly captures how far the aircraft has strayed sideways due to avoidance maneuvers. This feature is sign-agnostic in magnitude so the policy learns to minimize deviation regardless of side. It is normalized by a factor X_{scale} that can be tuned, based on the environment. The exponent gives a strong, informative gradient when the aircraft is close to the desired track. For large deviations, the exponential saturates toward 1.

- Along-track progress parallel to original route:

$$\phi_{\text{along}} = \frac{\Delta s_\parallel}{S_{\text{scale}}} \tag{13}$$

The along-track distance measures progress along the original track direction toward the exit, computed as the projection of the vector from the current position

to the exit onto the initial heading direction. By rewarding reductions in remaining along-track distance only once the aircraft is safe and roughly re-aligned, the agent is encouraged to restore the nominal route and exit efficiently, rather than "sniping" the exit from an off-track heading.

Finally, to prevent dominance of the bias feature and uncontrolled weight growth of the other features, *L2 regularization* is added to the update:

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha \left(\delta \, \boldsymbol{\phi}(s) - \lambda \mathbf{w}_a\right), \tag{14}$$

where $\lambda > 0$ is the regularization coefficient. L2 regularization discourages large weights that are not supported by informative state features and mitigates the tendency of the bias term to absorb most of the value function. In general, this should stabilize learning, improves generalization across encounter geometries, and promotes feature-driven policies rather than constant action preferences.

2.2 Reward Function Design and Phase Gating

In linear Q-learning, the agent can only "see" relationships that are directly proportional to your the features, meaning it cannot learn complex patterns on its own. Because of this limitation, reward shaping acts as a vital guide, providing "breadcrumbs" that nudge the agent toward the goal when the features are not descriptive enough to show the full path. While a good feature set provides the eyes, a well-designed reward function provides the motivation; without careful shaping, a linear agent often fails to bridge the gap between simple observations and complex objectives.

Here, the reward function is designed to explicitly decompose the conflict-resolution task into two sequential phases: (i) *conflict resolution* and (ii) *trajectory recovery*. This separation is enforced through *phase gating*, which activates or suppresses specific reward terms depending on whether a predicted conflict exists.

Let s_t denote the state at time t, a_t the applied action, and let $\mathbb{I}_{\text{conf}}(s_t)$ be an indicator that equals 1 when a conflict is predicted and 0 otherwise. The total reward is written as

$$r_t = r_{\text{base}} + r_{\text{conf}} + r_{\text{dcpa}} + r_{\text{man}} + r_{\text{rec}} + r_{\text{xt}} + r_{\text{term}}. \tag{15}$$

Base and safety penalties. A small living cost encourages timely task completion,

$$r_{\text{base}} = -c_{\text{live}}, \tag{16}$$

while large terminal penalties ensure that unsafe outcomes are always suboptimal,

$$r_{\text{term}} = \begin{cases} -C_{\text{conflict}}, & \text{loss of separation}, \\ -C_{\text{oob}}, & \text{leaving the airspace}. \end{cases} \tag{17}$$

These penalties are scaled to be worse than any reasonable maneuvering strategy, but not orders of magnitude larger than other terms, preventing dominance of the bias feature.

Conflict persistence penalty. To avoid local minima where the agent "waits" inside a predicted conflict, a dense per-step penalty is applied whenever a conflict is predicted:

$$r_{\text{conf}} = -\mathbb{I}_{\text{conf}}(s_t)\, C_{\text{conf-step}}. \tag{18}$$

This term ensures that doing nothing during a conflict is always worse than taking a small corrective maneuver.

dCPA improvement shaping (conflict phase). During the conflict-resolution phase, the primary guidance signal is improvement in distance at closest point of approach (d_{CPA}). When aircraft are closing,

$$r_{\text{dcpa}} = \mathbb{I}_{\text{conf}}(s_t)\, \left[k_{\Delta d}\, \frac{\Delta d_{\text{CPA}}}{d_{\text{sep}}} + k_{\text{hinge}}\, \frac{\min(d_{\text{CPA}} - d_{\text{target}},\, d_{\text{sep}})}{d_{\text{sep}}} \right], \tag{19}$$

where $\Delta d_{\text{CPA}} = d_{\text{CPA}}(t) - d_{\text{CPA}}(t-1)$. The first term provides a dense, action-attributable gradient that rewards immediate improvement, while the hinge term softly stabilizes solutions around a desired safety margin $d_{\text{target}} > d_{\text{sep}}$ without encouraging excessive deviation.

Maneuver event cost. To promote sparse and decisive control, maneuvers are penalized as *events* rather than per-step actions:

$$r_{\text{man}} = -\mathbb{I}_{\text{event}}(a_t)\, \left(c_0 + c_1\, \max(0, N_{\text{event}} - 2)^2 \right), \tag{20}$$

where $\mathbb{I}_{\text{event}}(a_t)$ indicates a change to a non-zero command and N_{event} is the cumulative number of maneuver events. This quadratic growth strongly discourages dithering while still allowing one avoidance maneuver and one recovery maneuver. As such, taking more maneuvers than two that are desired (i.e., one avoidance, one restore) will be much more penalized.

Recovery-phase heading shaping. Once the conflict is resolved ($\mathbb{I}_{\text{conf}} = 0$), the reward shifts focus to restoring the original heading ψ_0. Two complementary terms are used:

$$r_{\text{rec}} = (1 - \mathbb{I}_{\text{conf}}(s_t)) \Big(k_{\Delta\psi} (\, |\psi_{t-1} - \psi_0| - |\psi_t - \psi_0| \,) \tag{21}$$

$$- k_{\psi}\, |\psi_t - \psi_0| \Big). \tag{22}$$

The first term rewards progress toward the nominal heading, while the second penalizes sustained deviation, ensuring convergence rather than oscillation.

Cross-track and along-track shaping. To prevent the agent from exiting the airspace on an offset parallel trajectory, cross-track deviation $x_\perp$ from the nominal track is penalized after conflict resolution:

$$r_{\text{xt}} = (1 - \mathbb{I}_{\text{conf}}(s_t)) \left[-k_\perp \left(1 - e^{-|x_\perp|/X_{\text{scale}}} \right) + k_{\Delta\perp}\, \Delta|x_\perp| \right]. \tag{23}$$

The exponential form provides high sensitivity near the track while saturating for large

deviations, preventing domination of the value function. An additional along-track progress term rewards forward motion toward the exit only when the heading has been sufficiently restored, preventing "exit sniping".

Task completion bonus. Finally, successful completion (defined as conflict resolved, heading restored within tolerance, and exit on the correct boundary) yields a bounded terminal bonus:

$$r^{+}_{\text{term}} = C_{\text{success}}. \tag{24}$$

This bonus must be deliberately scaled to be comparable to the accumulated dense rewards, ensuring that learning remains driven by state-action structures rather than a single terminal spike.

Overall, phase gating ensures that conflict-resolution rewards dominate early in the encounter, while recovery and trajectory-quality rewards dominate afterward. This structured decomposition would be crucial for avoiding local minima, reducing bias dominance, and achieving the desired two-maneuver strategy.

3 Shaping RL Behavior to Operational "Best Practices"

Reinforcement learning based on Q-learning can be substantially accelerated and guided toward operationally acceptable behavior by incorporating additional sources of structure beyond sparse trial-and-error rewards. Action shielding restricts the agent's action space online by filtering out commands that are known *a priori* to violate hard safety constraints (e.g., predicted loss of separation), ensuring that exploration remains within a safe and feasible region of the state–action space. Human feedback, provided either as corrective signals or preferences over actions, can further shape the value function by nudging Q-updates toward behaviors that align with domain expertise, even when the reward function alone is ambiguous. Finally, expert demonstrations, such as rule-based or human-designed conflict resolution maneuvers, can be integrated into the Q-learning updates to bias early learning toward proven strategies while still allowing the agent to improve upon them through exploration. These possibilities are integrated into a single RL architecture (Figure 2), enabling the evaluation of each technique both individually and in combination.

Together, these mechanisms reduce the effective exploration burden, improve sample efficiency, and help the learned policy converge toward solutions that are both performant and consistent with established operational practices.

3.1 Action Shielding

To speed up learning and to prevent the RL agent from taking unsafe or obviously suboptimal actions, a *pre-selection action shielding* mechanism is employed, see Figure 2. This means that action filtering occurs *before* the policy selects an action. Thus, the

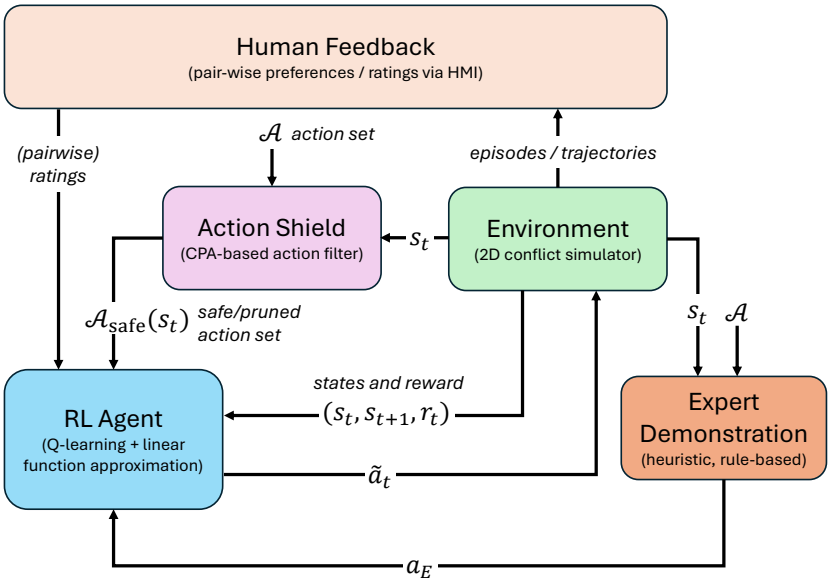

Figure 2: A single Reinforcement Learning architecture with pre-selection (pre-shielding) action filter, human feedback and learning from demonstrations. Each shaping technique can be activated both individually and in combination.

agent samples from a reduced action set

$$\mathcal{A}_{\text{safe}}(s) \subseteq \mathcal{A},$$

and the unsafe actions are never passed to the Q-learning update. Action shielding is most common in the field of *Safe Reinforcement Learning*.

Here, an action a is considered safe if and only if all of the following geometric constraints hold:

1. **Immediate separation constraint:** Simulate the next-step relative positions; if the action would result in immediate loss of separation ($d_{t+1} < 5$ NM), then a is rejected.

2. **CPA monotonicity constraint:** Let d_{CPA}^{current} denote the predicted minimum separation if maintaining the current heading, and let d_{CPA}^{next} be the predicted CPA after applying action a. If

$$d_{CPA}^{\text{next}} < d_{CPA}^{\text{current}} - \eta,$$

with η a small marging factor, then the action is rejected. Thus, actions that worsen predicted separation are blocked.

3. **Conflict introduction constraint:** If no predicted conflict is present in the current state, an action may not introduce a predicted conflict.

4. **Exit-heading constraint (when safe):** If no predicted conflict exists, an action

that increases the exit-heading error beyond a small tolerance is rejected. Note: this shielding factor is not a safety-related constraint, but a performance-related constraint, yielding more desired behavior by preventing the ownship from flying in the opposite direction of the exit waypoint.

The shield shown in Figure 2 is a *pre-shielding* mechanism that guarantees the agent only explores safe or geometrically meaningful actions. Q-learning itself never sees unsafe actions, which would accelerate training and stabilizes convergence. In this pre-shielding mechanism, the Q-value updates via temporal-difference become:

$$\delta_t = r_t + \gamma \max_{a' \in \mathcal{A}_{\text{safe}}(s_{t+1})} w_{a'}^\top \boldsymbol{\phi}(s_{t+1}) - w_{a_t}^\top \boldsymbol{\phi}(s_t),$$

$$w_{a_t} \leftarrow w_{a_t} + \alpha\, \delta_t\, \boldsymbol{\phi}(s_t),$$

Pre-action vs. post-action shielding. Safety constraints in shielded RL can be enforced either before action selection (pre-action shielding) or after an action has been proposed (post-action shielding). Pre-action shielding like in Figure 2 restricts the agent to selecting actions only from a state-dependent safe set, ensuring safety by construction and maintaining consistency between the agent's decisions and the executed behavior. This would lead to more stable learning dynamics and stronger compatibility with formal verification and constrained control frameworks, although overly conservative constraints may reduce exploration and slow convergence.

In contrast, post-action shielding allows the agent to propose arbitrary actions that are subsequently filtered or corrected by a safety mechanism. While this approach is easier to integrate with existing agents and can support broader exploration, it introduces a mismatch between the agent's internal policy updates (i.e. what the agent thinks it has executed) and the actions actually executed in the environment, which can degrade learning stability and complicate analysis. For safety-critical applications and systems requiring strong guarantees, pre-action shielding is generally preferred when feasible, whereas post-action shielding remains a practical alternative when architectural constraints limit tighter integration.

Shielding vs. reward shaping. In safety-critical reinforcement learning, there is a fundamental design choice between encoding constraints and desired behaviour in (i) an explicit *action shielding* mechanism, or (ii) the *reward function*. In the aircraft conflict-resolution setting, both approaches can accelerate learning, but they have different implications for convergence speed, policy flexibility, and human-feedback integration.

Reward shaping provides a principled way to encode behavioral preferences (e.g. minimizing extra track miles, limiting the number of maneuvers, returning to a nominal heading). However, reward-only approaches can converge slowly in this domain for three reasons:

1. **Delayed credit assignment:** separation-loss penalties and track-mile penalties may accumulate over multiple steps, requiring TD learning to propagate information backward in time.

2. **Small reward differences between actions:** many heading-change actions lead to similar short-term outcomes, creating a low signal-to-noise ratio for Q-learning.

3. **Exploration burden:** without constraints, the agent must explore many suboptimal but feasible behaviours before learning to avoid them, especially with function approximation.

These effects often produce long transients before the policy becomes useful.

Pre-selection action shielding reduces the effective action space from $\mathcal{A}$ to a state-dependent subset $\mathcal{A}_{\text{safe}}(s)$, dramatically increasing sample efficiency. In particular, shielding i) blocks actions that violate hard safety constraints (immediate separation loss), ii) removes actions that degrade predicted safety margins (e.g. lower predicted d_{CPA}), iii) prevents exploration of actions that introduce conflicts from otherwise safe states. By filtering clearly undesirable actions, the agent focuses learning on the remaining meaningful decisions and typically achieves safe behaviour with fewer episodes.

3.2 Human Feedback as Policy Shaping

After (or during) autonomous Q-learning, optional human-feedback mechanisms shape the policy without discarding the learned Q-function. Here, human feedback is given after each episode and not after each individual action.

Pairwise preference learning. Given two trajectories A and B, the human selects the preferred one. The Q-function is then updated as:

$$w \leftarrow w + \alpha_h \big(\nabla_w Q(A) - \nabla_w Q(B)\big),$$

pushing up the value of preferred actions and decreasing the value of dispreferred ones. Here, α_h is the feedback gain. To ensure a fair comparison, both trajectories are initialized with identical encounter geometry, including ownship and intruder positions, headings, and speeds.

To avoid trivial duplication of behavior, the second trajectory B must be explicitly forced to diverge from the first at the earliest non-zero maneuver decision by excluding the action taken in trajectory A at that decision step. Apart from this enforced divergence, both trajectories are generated using the same policy, action shielding, and constraints.

After execution, the user selects the more acceptable trajectory (or indicates no clear preference), and the agent's action-value estimates are updated by reinforcing actions along the preferred trajectory while penalizing those of the non-preferred one.

This mechanism allows human judgment to directly shape the policy while preserving safety and consistency across comparisons.

With linear function approximation $Q(s,a) = \mathbf{w}^\top \boldsymbol{\phi}(s,a)$, the Q-function update simplifies to adding a scaled difference of feature vectors, $\alpha_h(\boldsymbol{\phi}(s,A) - \boldsymbol{\phi}(s,B))$, to the weights. Geometrically, this moves the parameter vector in a direction that increases the margin $Q(s,A) - Q(s,B)$, thereby reshaping the value function so that the preferred action becomes more attractive in the current state and in similar states that share feature structure. Repeated updates of this kind gradually enforce consistency with human preferences by increasing the separation between favored and disfavored actions in value space.

This update can be interpreted as stochastic gradient ascent on an implicit objective that maximizes the difference between the Q-values of preferred and rejected actions, or equivalently minimizes a ranking loss that penalizes violations of the desired ordering. Unlike standard temporal-difference learning, which adjusts values to match observed rewards and bootstrapped returns, pairwise feedback imposes inequality constraints of the form $Q(s,A) > Q(s,B)$, making it a form of preference-based reinforcement learning. The resulting mechanism resembles large-margin ranking updates, in which each comparison incrementally pushes the value function toward a configuration that respects accumulated human judgments. When integrated with ordinary Q-learning updates, these preference gradients act as an *auxiliary* shaping signal that biases the learned policy toward actions that align with human expectations, while still allowing environmental rewards to determine long-term optimal behavior.

Rating-based feedback. A trajectory receives a human rating $r \in \{1,2,3,4,5\}$, mapped to a scalar signal $R \in [-1,1]$. The weights are updated:

$$w_a \leftarrow w_a + \alpha_h(R - Q(s,a;w))\,\phi(s).$$

Highly rated trajectories increase the value of their action sequences; low-rated ones decrease it.

Explanation

Human pairwise feedback can be understood intuitively by thinking about how one would adjust a simple linear model when told that one prediction should be higher than another. Consider first a basic linear equation,

$$y = ax + b,$$

which produces a score for some input x. Suppose two inputs are evaluated, x_A and x_B, and a human tells us that the output for x_A should be higher than for x_B. If currently the model produces similar values or even assigns a higher value to x_B, parameters need to be adjusted so that the output for x_A increases and the output for x_B decreases. The direction that increases the dif-

ference between the two predictions is determined by how each prediction depends on the parameters. Since the gradient of y with respect to a is x, increasing a in proportion to $x_A - x_B$ raises the output for x_A relative to x_B. The resulting update takes the form

$$a \leftarrow a + \alpha_h(x_A - x_B), \qquad b \leftarrow b + \alpha_h(1 - 1),$$

so the slope changes in a way that enlarges the gap between the preferred and rejected inputs. Repeating this process gradually reshapes the line so that preferred inputs consistently receive higher outputs.

The same idea applies to a multi-feature linear model,

$$y = a_1 x_1 + a_2 x_2 + b.$$

If a human indicates that input A should be preferred over input B, the parameters are adjusted using the difference between their feature vectors. Each parameter a_i is increased in proportion to the difference between the corresponding features of the preferred and rejected inputs. In Q-learning with linear function approximation,

$$Q(s, a) = \mathbf{w}^\top \boldsymbol{\phi}(s, a),$$

this becomes a weight update

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha_h\big(\boldsymbol{\phi}(s, A) - \boldsymbol{\phi}(s, B)\big).$$

This update increases the value of the human-preferred action and decreases the value of the rejected action in proportion to their features. Over time, the learned linear function is reshaped so that actions aligned with human preferences receive higher predicted values, while disfavored actions receive lower ones. In this way, pairwise feedback acts as a direct mechanism for reshaping the value landscape according to human judgments.

3.3 Learning from Expert Demonstrations

In addition to learning-based conflict resolution, rule-based heuristic algorithms exist that reflect established air traffic control (ATC) best practices. Here, a rule-based Velocity Obstacle (VO) algorithm formulation is used to identify control actions that place the relative velocity of the ownship outside the collision cone induced by the intrude's protected zone. In the proposed rule-based variant, candidate maneuvers are further constrained to ensure a fixed *pass-behind* geometry: among all safe actions that remove the ownship from the velocity obstacle, the algorithm selects the smallest available heading and/or speed deviation for which the intruder lies ahead of the ownship at the predicted closest point of approach. This criterion directly encodes the operational preference for predictable behind-passing maneuvers, which

are commonly issued by human controllers.

Once the intruder has passed the ownship along the original route direction and a safety margin is guaranteed, the rule-based logic transitions to a recovery phase. In this phase, the ownship applies the minimal corrective action that reduces deviation from the original heading (and speed, if enabled), while ensuring that no new predicted conflict is introduced. By defining "passed" relative to the initial route direction rather than the instantaneous heading, the algorithm robustly initiates recovery even after large avoidance maneuvers.

Incorporating expert demonstrations into Q-learning. To accelerate learning and bias the policy toward operationally acceptable behavior, expert demonstrations are incorporated into the Q-learning updates. Let $\pi_E(s)$ denote an expert policy, here realized by a rule-based velocity-obstacle (VO) conflict resolution controller that selects safe and structured maneuvers (i.e., pass-behind strategies). Given a state s_t and the expert-selected action $a_t^E = \pi_E(s_t)$, the agent receives an additional learning signal beyond the standard temporal-difference (TD) update.

When an expert action a_t^E is available, an additional supervised-style update is applied to encourage the agent to assign higher value to the demonstrated action. Specifically, the agent performs an imitation update of the form

$$\mathbf{w}_{a_t^E} \leftarrow \mathbf{w}_{a_t^E} + \alpha_E \left(Q_E - Q(s_t, a_t^E)\right) \boldsymbol{\phi}(s_t), \tag{25}$$

where α_E is a demonstration gain and Q_E is a target value that biases the demonstrated action upward relative to competing actions. In practice, Q_E can be chosen implicitly (e.g., by applying a fixed positive margin) or implemented as a small additive boost to the demonstrated action's Q-value.

In a margin-based formulation of learning from expert demonstrations, the demonstrated action is encouraged not merely to have a high value in absolute terms, but to be decisively better than *all other* competing actions in the same state (with $a \neq a^E$). Concretely, a target value is constructed as $Q_E = \max_a Q(s,a) + m$, where $m > 0$ is a fixed margin that specifies how much better the expert action should be compared to the agent's current best alternative. The Q-update can then be interpreted as stochastic gradient descent on a regression objective that drives the predicted value of the demonstrated action toward this margin-based target. If the demonstrated action already exceeds all other actions by at least the margin, the update vanishes or becomes negative, preventing unnecessary growth; otherwise, the update increases its value until the inequality $Q(s,a^E) \geq \max_a Q(s,a) + m$ is satisfied. This creates a clear separation between the expert action and its competitors, improving policy robustness by ensuring that small value estimation errors or noise do not easily cause the policy to deviate from demonstrated behavior.

The weight update does not replace the TD update, but rather complements it, resulting in a combined learning signal:

$$\Delta\mathbf{w} = \Delta\mathbf{w}_{\text{TD}} + \Delta\mathbf{w}_{\text{demo}}. \tag{26}$$

Selective and gated imitation. To avoid over-constraining the policy, expert imitation is applied selectively. In particular, demonstration updates are gated to states in which expert knowledge is most relevant, such as when a conflict is predicted or imminent. Formally, the imitation update is applied only if

$$\mathbb{I}_{\text{demo}}(s_t) = \begin{cases} 1, & \text{if predictedConflict}(s_t) = \text{true}, \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

ensuring that the agent remains free to explore during benign flight phases while receiving strong guidance during safety-critical situations.

Explanation

Expert demonstrations can be interpreted as supervised corrections that push the value of a demonstrated action toward a higher target. Consider again a simple linear model,

$$y = ax + b.$$

If an expert indicates that a particular input x_E should produce a high output, a desired target value can be defined y_E for that input. If the current prediction is too low, the output is increased by adjusting the parameters in proportion to the prediction error

$$\delta = y_E - y_{\text{pred}}.$$

Since the gradient of the prediction with respect to a is x_E, increasing a by $\alpha_E \delta x_E$ raises the output for that input. The resulting update

$$a \leftarrow a + \alpha_E \delta x_E, \qquad b \leftarrow b + \alpha_E \delta$$

gradually shifts the line so that the demonstrated input achieves the desired high value. If the prediction becomes too high, the sign of the error reverses and the update stabilizes the output around the target.

For a model with multiple inputs,

$$y = a_1 x_1 + a_2 x_2 + b,$$

the same principle applies: each parameter is adjusted proportionally to both the prediction error and its associated feature. In Q-learning with linear function approximation,

$$Q(s, a) = \mathbf{w}_a^\top \boldsymbol{\phi}(s),$$

a demonstrated action a^E is assigned a higher target value Q_E. Rather than choosing this target as a fixed constant, it is defined relative to the values of competing actions. Intuitively, the demonstrated action should not merely be good in isolation, but clearly better than *all other* alternatives (excluding a^E!) available in the same state. For this reason, the target can be defined using a

margin,

$$Q_E = \max_{a \neq a^E} Q(s, a) + m,$$

where $m > 0$ is a small positive constant. This means that the demonstrated action is encouraged to exceed the value of the best competing action by at least a margin gap. Such a margin ensures that the expert action becomes decisively preferred and remains robust to small estimation errors or noise in the value function.

The update

$$\mathbf{w}_{a^E} \leftarrow \mathbf{w}_{a^E} + \alpha_E \big(Q_E - Q(s, a^E)\big) \boldsymbol{\phi}(s)$$

then increases the predicted value of the demonstrated expert action a^E when it is too low relative to this margin-based target Q_E and reduces the update once the target is reached. Here, $Q(s, a^E)$ is simply the agent's current estimate of how good the expert's action is in state s.

Defining the target relative to competing actions makes the update scale-invariant and adaptive: as the overall magnitude of Q-values changes during learning, the demonstrated action is always pushed to remain clearly better than the alternatives. Repeated application of such updates reshapes the linear value function so that actions chosen by the expert receive consistently higher Q-values in similar states, encouraging imitation while still allowing the agent to refine values through interaction (including exploration) with the environment.

Relation to existing methods. This approach is closely related to demonstration-augmented reinforcement learning methods such as Deep Q-learning from Demonstrations (DQfD), but differs in two key aspects. First, demonstrations are incorporated online rather than through a fixed replay buffer, allowing the agent to continuously query an expert policy. Second, no hard constraint is imposed to force imitation; instead, demonstrations softly bias the value function while preserving ϵ-greedy exploration. As a result, the agent can initially imitate expert behavior to achieve safe and efficient conflict resolution, and subsequently refine or simplify these strategies as dictated by the learned reward structure.

Effect on learning dynamics. By injecting expert demonstrations into the Q-updates, the agent receives informative gradients even in regions of the state space where reward signals are sparse or delayed. This significantly reduces the number of episodes required to discover effective avoidance maneuvers, stabilizes early learning, and promotes convergence toward policies that align with established operational practices while retaining the flexibility of reinforcement learning.

3.4 Stability Considerations with Multiple Learning Signals

When off-policy Q-learning with linear function approximation is combined with human pairwise feedback and expert demonstrations, multiple weight updates influence the value function simultaneously. Temporal-difference (TD) learning adjusts Q-values according to environmental rewards, while demonstration and human feedback updates reshape them according to external guidance. If applied with large learning rates or inconsistent signals, these updates may compete and lead to oscillations or overestimation, particularly because off-policy Q-learning with function approximation is already sensitive to instability.

In the present setting, pre-shielding is used to prevent unsafe actions before they are executed. This means the agent only ever observes transitions resulting from admissible actions, and unsafe actions are removed from the decision set in advance. For consistency, TD targets should therefore be computed only over actions that remain available after pre-shielding, ensuring that value estimates reflect the true set of executable actions. A stable integration strategy when safety is critical is to treat TD learning as the primary learning signal with pre-shielding activated, use expert demonstrations mainly during *early* learning to bootstrap reasonable value estimates, and occasionally apply human pairwise feedback as a weaker auxiliary signal for refining action preferences. Gradually reducing the influence of demonstration and human feedback updates over time may help prevent conflicting gradients and supports stable convergence of the learned value function.

4 JavaScript implementation

The proposed framework is implemented as an interactive JavaScript/HTML application that enables real-time visualization and analysis of learning-based aircraft conflict detection and resolution in a two-dimensional airspace (see Figure 3).

The application displays the simulated environment, including ownship and intruder trajectories within a 150×100 NM sector, along with configurable control options for training and evaluation (e.g., exploration rate, shielding, and expert demonstrations). During execution, the interface provides live plots of key learning metrics such as episode reward, success and failure rates, maneuver counts, and feature importance derived from the learned Q-function, allowing the evolution of the policy to be monitored over time. A detailed, step-by-step event log further supports traceability and debugging by recording actions, rewards, and termination causes. Training scenarios are generated by sampling randomized conflict geometries, including relative positions, headings, and speeds, ensuring exposure to a diverse set of encounter configurations while maintaining a controlled and reproducible experimental setup.

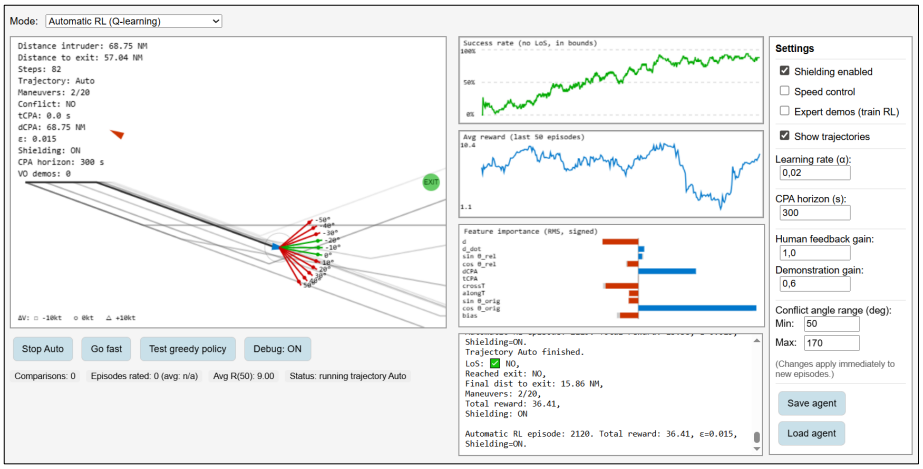

Figure 3: HTML/JavaScript application.

4.1 Q-Value Matrix and Learnable Weights

The action space consists of a discrete set of heading combined with optional speed changes:

$$\mathcal{A} = \{(\Delta\psi, \Delta V)\} = \{\Delta\psi\} \times \{\Delta V\}, \tag{28}$$

yielding $|\mathcal{A}| = 11 \times 3 = 33$ discrete actions. The HTML implementation represents the Q-function using a separate weight vector for each action:

$$Q(s, a) = w_a^\top \phi(s),$$

where $w_a \in \mathbb{R}^{11}$ is the parameter vector associated with action a. Stacking these vectors row-wise gives a weight matrix

$$W = \begin{bmatrix} w_{a_1}^\top \\ w_{a_2}^\top \\ \vdots \\ w_{a_{|\mathcal{A}|}}^\top \end{bmatrix} \in \mathbb{R}^{|\mathcal{A}| \times 11},$$

so that, for a given state s, the vector of Q-values over all actions can be computed as

$$Q(s, \cdot) = W\, \phi(s).$$

In the JavaScript code, this matrix W is stored as a two-dimensional array `weights[a][i]`, where the first index corresponds to the discrete

4.2 From Weights and Features to Actions

Given the feature representation $\phi(s)$ and the weight matrix W, the sequence of actions executed by the agent in the HTML application emerges from a simple, but fully

parametric, mechanism. This subsection explains how i) Q-values are computed from the weights, ii) how the policy selects actions (with shielding and ε-greedy exploration), and iii) how this leads to a concrete sequence of maneuvers in time.

Computing Q-values from the weight matrix. The Q-function is parameterized as

$$Q(s, a) = w_a^\top \phi(s),$$

with a separate weight vector $w_a \in \mathbb{R}^F$ (here $F = 11$) for each discrete action $a \in \mathcal{A}$. Stacking these vectors produces a weight matrix $W \in \mathbb{R}^{|\mathcal{A}| \times F}$.

For a given state s_t, the application computes:

1. The feature vector $\phi_t = \phi(s_t)$.
2. For each action index a, the scalar

$$Q_t(a) = Q(s_t, a) = w_a^\top \phi_t.$$

In code, this corresponds to iterating over the rows `weights[a]` and computing the dot products with the feature array `phi`.

Shielded action set and ε-greedy policy. The agent does not choose among all actions $\mathcal{A}$, but among a state-dependent, *shielded* action subset

$$\mathcal{A}_{\text{safe}}(s_t) \subseteq \mathcal{A},$$

constructed by the CPA-based action shield (Section 3.1). Only actions that satisfy the instantaneous and predicted safety constraints (e.g. non-worsening CPA, no immediate loss of separation, no unnecessary deviation from the exit in safe conditions) are retained in $\mathcal{A}_{\text{safe}}(s_t)$.

Given the Q-values $Q_t(a)$ and the shielded set $\mathcal{A}_{\text{safe}}(s_t)$, the policy uses an ε-greedy strategy:

$$a_t = \begin{cases} \text{a random element of } \mathcal{A}_{\text{safe}}(s_t), & \text{with probability } \varepsilon, \\ \arg\max\limits_{a \in \mathcal{A}_{\text{safe}}(s_t)} Q_t(a), & \text{with probability } 1 - \varepsilon. \end{cases} \tag{29}$$

Thus, the learned weights w_a determine the *ranking* of actions via the Q-values, while the shield restricts the candidate set and ε controls the amount of exploration.

In the implementation, these steps correspond to:

1. Computing `safeActions = getSafeActions(env, state)`.
2. Computing all Q-values `qs[a] = dot(weights[a], phi)`.
3. With probability ε, choosing a random index from `safeActions`; otherwise, choosing the index in `safeActions` with maximal `qs[a]`.

From policy to an action sequence. Over the course of an episode, this mechanism generates a sequence of states, Q-values, and actions:

$$s_0 \xrightarrow{\phi_0, Q_0, a_0} s_1 \xrightarrow{\phi_1, Q_1, a_1} s_2 \xrightarrow{\phi_2, Q_2, a_2} \cdots$$

More explicitly:

1. At time t, the environment provides the current state s_t (ownship and intruder positions, velocities, CPA prediction, distance to exit, etc.).
2. The agent computes $\phi_t = \phi(s_t)$ and Q-values $Q_t(a) = w_a^\top \phi_t$ for all a.
3. The shield computes $\mathcal{A}_{\text{safe}}(s_t)$, discarding unsafe or disallowed actions.
4. The policy selects a_t using (29).
5. The environment applies a_t as a heading change, advances the simulation, and returns the next state s_{t+1} and reward r_t.

In this way, the weight matrix W completely determines–through the Q-value ordering and the ε-greedy choice within the shielded action subset–which *sequence of maneuvers* the agent will execute for any given encounter geometry. Different choices of W (e.g. after training with automatic TD learning versus after additional human-feedback shaping) result in different preferences over actions in each state, and therefore in different avoidance and steer-back patterns observed in the trajectories.

Effect of learning on future action sequences. During training, the Q-learning update adjusts w_{a_t} at each step,

$$w_{a_t} \leftarrow w_{a_t} + \alpha\Big(r_t + \gamma \max_{a'} w_{a'}^\top \phi(s_{t+1}) - w_{a_t}^\top \phi(s_t)\Big)\phi(s_t),$$

so that actions leading to higher long-term returns under the shaped reward (higher safety, fewer maneuvers, shorter track miles) receive higher Q-values. Consequently, in future episodes, the greedy part of the policy ($1-\varepsilon$ branch in (29)) will increasingly favor those actions, altering the entire *sequence* of maneuvers.

Human feedback and expert deomonstrations further modifies the same weights w_a, biasing the Q-values so that, among the actions allowed by the shield, those that produce more human-preferred trajectories become more likely to be selected. Over time, the combination of (i) TD learning driven by environment rewards and (ii) preference- or rating-based shaping gives rise to action sequences that jointly reflect safety constraints, geometric efficiency, and human acceptance.

4.3 Speed Control

Enabling speed control expands the agent's action space from heading-only maneuvers to combined heading–speed decisions, which typically leads to slower learning and may manifest as a drop and subsequent plateau in the moving-average reward.

First, the larger action set increases exploration requirements: with the same ϵ-greedy schedule, each individual action is sampled less frequently, delaying reliable value estimates and policy improvement. Second, the credit-assignment problem becomes harder because speed changes often have delayed and more subtle effects on encounter geometry (e.g., shifting t_{CPA} and d_{CPA} over multiple steps), while the reward is accumulated step-wise. Third, the reward shaping may implicitly discourage speed usage: an explicit speed-change penalty, combined with existing turn, deviation, and maneuver-count penalties, can make beneficial speed adjustments appear unattractive until the agent has sufficiently explored states where the safety or efficiency benefit outweighs the cost. Finally, action shielding further constrains exploration; with more candidate actions, the filtered "safe" subset may become highly state-dependent, which can increase variance in updates and slow convergence.

To mitigate these effects without changing the task definition, the following options can be explored:

- increasing exploration when speed control is enabled (e.g., a higher initial ϵ and/or slower ϵ decay) to compensate for the enlarged action space;
- modestly increasing the learning rate to offset reduced per-action visitation, while monitoring stability;
- weakening the speed-change penalty during early training (or applying it more strongly only outside predicted-conflict states) so that speed actions are not prematurely suppressed;
- using staged training, first learning a robust heading-only policy and then enabling speed control for refinement.

In practice, these adjustments restore learning progress by ensuring adequate coverage of the expanded action space and by aligning reward incentives with the intended operational use of speed as a secondary, conflict-driven control dimension.

4.4 Feature Importance and Interpretability

To analyze policy interpretability, a signed feature relevance metric can be computed from the learned weights. For each feature i, one can calculate calculate:

$$\text{RMS}_i = \sqrt{\frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} w_{a,i}^2}, \tag{30}$$

$$\mu_i = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} w_{a,i}. \tag{31}$$

The RMS value measures the overall importance of a feature, while the sign of μ_i indicates whether the feature tends to increase or decrease action values. This analysis provides a practical diagnostic tool to detect reward imbalance, feature saturation, and premature convergence.

The resulting visualization is a two-sided bar chart in which the bar length corresponds to the RMS importance, while the bar orientation (left or right of the zero line) indicates the sign of μ_i. Features with large bars dominate the policy's decision-making, whereas features with near-zero magnitude have little influence.

By overlaying consecutive snapshots of this plot during training, convergence can be assessed visually: once the bar lengths and directions stabilize, the learned value function has reached a steady regime. This representation provides an intuitive diagnostic tool for understanding which aspects of the encounter geometry the agent relies on and how this reliance evolves over time. However, it is important to remark that feature relevance is not directly tied to performance: weights can stabilize at a bad policy, or keep drifting while performance is fine (especially with ongoing ϵ-exploration, changing encounter distribution, or shielding).

One feature in particular deserves special attention: the bias. This bias is a constant component of the feature space, $\phi_{\text{bias}}(s) = 1$, and provides each action with a state-independent offset in the linear action-value approximation. A large bias weight therefore indicates a baseline preference for an action that does not depend on the encounter geometry or relative motion between aircraft. When the bias term dominates the feature importance plot, it implies that the learned policy relies primarily on such state-independent action preferences rather than exploiting the available state features. In practical terms, this suggests that the agent has converged to a default maneuver that performs reasonably well across many encounters, but does not adapt strongly to encounter-specific conditions.

Although a dominant bias can arise in highly symmetric or narrowly distributed scenarios, it more commonly signals that the remaining features or reward shaping do not provide sufficient discriminatory power. Consequently, a strong bias term should be interpreted as a warning that the policy may be brittle or overgeneralized, motivating further feature normalization, reward refinement, or regularization to encourage state-dependent decision making.

Here, when action shielding is active, a dominant bias term is particularly informative. Shielding constrains the available action set to those deemed safe, often removing state-dependent distinctions between actions in many regions of the state space. As a result, the learning problem presented to the agent becomes less informative, and the value function can collapse toward action-dependent constants captured by the bias term. In this case, a strong bias does not necessarily indicate poor safety performance, but rather that safety is enforced externally by the shield while the learned policy contributes limited additional state-dependent reasoning. Consequently, a dominant bias under shielding should be interpreted as a sign that either the shield is overly restrictive or that additional state features or reward signals are required for the agent to learn nuanced, context-sensitive behavior beyond the enforced safety constraints.

In summary, a desirable outcome is not uniform importance across all features, but: i) reduced dominance of the bias term, ii) clear relevance of conflict-related features during avoidance, iii) increased relevance of cross-track and heading features during recovery.

4.5 Behavioral Shielding and Learning

While adding behavioral objectives (e.g. "always point to exit" or "never increase heading error to a goal") to the shield can accelerate early learning, it risks *objective mismatch*:

- If the reward favors one notion of behavior (e.g. return to the original heading) while the shield enforces another (e.g. align to the exit waypoint), the agent experiences inconsistent optimization pressures.
- The agent cannot explore potentially beneficial actions that are filtered out, making it difficult to learn trade-offs (e.g. slightly increased heading error might reduce future maneuvers or track miles).
- Excessive filtering can collapse Q-value differences among remaining actions, reducing $\overline{\Delta Q}$ and keeping policy entropy high, which hinders learning and limits the utility of human-feedback shaping.

In the demonstrator, "safety only" vs. "safety + behavioral" shielding can be controlled by the boolean flag `ENABLE_HEADING_STYLE_SHIELD`. For faster learning while preserving (high) flexibility, the following architecture would be recommend:

1. **Safety-only shielding:** use the shield strictly to enforce hard safety constraints (separation and CPA-based feasibility).
2. **Behavior via reward shaping:** use dense, difference-based shaping terms (e.g. Δ extra track miles, Δ waypoint distance, heading error to the original route, maneuver penalties).
3. **Human feedback as refinement:** apply human preferences to reshape the same value function among already-safe actions, optionally with policy regularization to prevent behavioral drift.

4.6 Initial Results

Table 1 summarizes the hyperparameters set in the application. These parameters have been tuned based on several trial and error iterations.

Table 1: Hyperparameter settings

Parameter	Value
Learning rate α	0.01
Discount factor γ	0.99
Initial exploration rate ε_0	0.40
Exploration decay (per episode)	0.995
Minimum exploration rate ε_{min}	0.015
Maximum episode length	200 steps
Maneuver budget N_{event}	20 events
L2 regularization coefficient λ	10^{-4}

Baseline Q-learning. When trained using plain Q-learning with linear function approximation and the proposed feature set and reward function, the agent typically requires on the order of **3,000–4,000 episodes** to converge to a reasonably stable policy. During early training, learning is dominated by the bias feature, indicating that the agent primarily exploits global reward structure before discovering meaningful state–action correlations. As training progresses, useful conflict resolution behavior emerges gradually, but convergence remains sensitive to reward scaling and exploration scheduling. After having learned conflict resolution, the reward gating mechanism shifts the focus more towards operational performance to restore the trajectory close to its initial route and exit the airspace from the appropriate side.

Action shielding. Enabling action shielding—by pruning actions that violate kinematic or separation constraints—leads to **almost immediate emergence of desired conflict resolution behavior**. Since unsafe or clearly suboptimal actions are never explored, the agent effectively operates within a reduced and well-structured action space. While this produces near-operationally acceptable trajectories from the start, it also **strongly limits exploration**, preventing the agent from discovering alternative avoidance strategies or learning fine-grained trade-offs between maneuver magnitude, timing, and recovery. As a result, shielding is highly effective for deployment and demonstration, but less suitable as a standalone training mechanism.

Human feedback. Incorporating human feedback, for example by penalizing or discouraging undesirable actions during training, proves ineffective when applied from scratch. Without a baseline policy that already achieves conflict avoidance, the feedback signal is overwhelmed by random exploration and sparse terminal rewards. Empirically, human feedback becomes useful only once the agent has learned a *minimum viable policy*, after which feedback can bias learning toward smoother trajectories, fewer maneuvers, or earlier recovery. This observation aligns with the interpretation of human feedback as a *policy shaping* mechanism rather than a replacement for reinforcement *learning*.

Expert demonstrations. Expert demonstrations generated by a rule-based Velocity Obstacle (VO) pass-behind controller significantly reduce training time. Compared to baseline Q-learning, the agent typically reaches stable conflict resolution behavior within **500–1000 episodes**. By biasing exploratory actions toward expert-selected maneuvers, the agent quickly learns to associate key geometric features (e.g., relative bearing and d_{CPA} margin) with effective avoidance actions. However, because the expert follows a fixed steer-behind heuristic, the learned policy may inherit a **strong inductive bias**, defaulting to this behavior even in situations where alternative maneuvers would be equally valid or more efficient. Consequently, while expert demonstrations accelerate convergence, they may also limit policy diversity and adaptability if not combined with continued exploration or regularization.

5 Limitations of the Current Approach

Although the presented framework demonstrates a practical combination of Q-learning, action shielding, human-feedback shaping, and expert demonstration, several methodological and conceptual limitations remain.

Function Approximation Constraints

Linear function approximation used for the Q-function restricts the expressiveness of the value representation. Aircraft encounter geometry exhibits strong nonlinearities (e.g., angular wrap-around, CPA surfaces, velocity coupling). A linear model cannot capture these higher-order structures without extensive feature engineering. This may limit the performance and the robustness of the learned avoidance strategies.

Non-Stationarity Introduced by Action Shielding

The action-shielding mechanism filters unsafe or undesirable actions before the policy selects among them. While this provides safety guarantees, it alters the effective action space

$$\mathcal{A}_{\text{safe}}(s) \subseteq \mathcal{A},$$

making the learning problem *state-dependent non-stationary*. The Q-learning update implicitly assumes a fixed action set; however, the shield removes actions dynamically. As a result, theoretical convergence guarantees of standard Q-learning no longer apply. Shielding may also over-constrain exploration, preventing the agent from discovering beneficial maneuvers outside the filtered set.

Myopic Reward Design

The reward function contains several shaping terms (track deviation, heading error, maneuver penalties) that encode specific designer preferences. Although shaping accelerates learning, it may inadvertently dominate the long-term return, guiding policy behavior in unintended ways. Additionally, shaping is manually tuned and does not necessarily correspond to actual operational metrics such as fuel burn, pilot workload, or ATC constraints.

Human Feedback Integration

Human feedback (pairwise and ratings) is injected directly into the Q-function parameters. This has two limitations:

1. Human feedback competes with the environment-based TD signal rather than being integrated into a consistent reward model.
2. Updates are sparse and depend on episodic human evaluation, making it difficult to propagate fine-grained preferences into earlier parts of the policy.

More advanced methods such as learned reward models or preference prediction could more accurately represent human intent.

Absence of Hierarchical or Multi-Phase Reasoning

The agent implicitly learns an "avoid then return" pattern via a gated reward function and shaping, but no structural bias exists to enforce such a two-phase policy. This may result in occasional unnecessary maneuvers or delayed return-to-exit actions. A hierarchical RL architecture or an explicit high-level phase-switch policy could better reflect the decomposed structure of human conflict-resolution strategies.

Limitations in Safety Guarantees

The shield prevents immediate or predicted conflicts, but:

- It does not guarantee globally optimal avoidance trajectories.
- It cannot prevent loss of separation in inherently unavoidable geometries.
- It introduces brittleness if prediction errors occur.

A full safety guarantee would require formal verification or control-theoretic safety filters, which are outside the scope of this approach.

Scalability Limitations for Multi-Aircraft Scenarios

While the proposed approach demonstrates the feasibility of shielded off-policy Q-learning for pairwise aircraft conflict resolution, its current formulation exhibits several limitations when scaling to scenarios involving multiple intruding aircraft.

State dimensionality growth. The present feature representation encodes the relative geometry between the ownship and a single intruder. In multi-aircraft encounters, a naive extension would require concatenating feature vectors for each intruder, leading to a state dimensionality that grows linearly with the number of aircraft. This rapidly increases the complexity of the value-function approximation, degrades generalization, and exacerbates sample inefficiency, particularly for linear function approximators.

Action coupling and non-additive effects. Avoidance maneuvers that are safe with respect to one intruder may reduce separation margins with another. The current action shielding mechanism evaluates safety constraints independently for a single intruder and does not account for such coupled effects. In multi-aircraft settings, safety evaluation must consider the joint impact of actions across all intruders, which substantially increases the computational burden of the shield and complicates the definition of the safe action set $\mathcal{A}_{\text{safe}}(s)$.

Combinatorial growth of safety checks. The CPA-based shielding logic requires predicting future separation for each candidate action. With N intruders and $|\mathcal{A}|$ discrete actions, the cost of evaluating the shield grows as $\mathcal{O}(N|\mathcal{A}|)$ per time step. In dense traffic scenarios, this may limit real-time applicability unless aggressive pruning or hierarchical screening mechanisms are introduced.

Limited expressiveness of linear value functions. The linear Q-function approximation employed in the current implementation assumes that the effect of features on action values is approximately additive. In multi-aircraft encounters, however, optimal decisions often depend on nonlinear interactions between intruders (e.g., prioritizing the most critical conflict while temporarily accepting reduced margins elsewhere). Capturing such interactions typically requires more expressive function approximators or explicit prioritization mechanisms.

Single-agent perspective. The formulation treats conflict resolution as a single-agent problem in which only the ownship is controlled and all intruders follow fixed trajectories. In multi-aircraft scenarios, especially those involving other adaptive or autonomous agents, the environment becomes non-stationary from the perspective of any single learner. Standard off-policy Q-learning does not directly address this non-stationarity and may exhibit instability or slow convergence.

Implications for future extensions. Scaling the approach to multi-aircraft scenarios will likely require a combination of architectural and algorithmic changes, such as attention-based or set-invariant state representations, hierarchical or receding-horizon decision structures, prioritized conflict selection, and more expressive value or policy models. Additionally, extending the shielding mechanism to reason over multiple intruders jointly, rather than independently, will be essential to preserve safety guarantees without overly restricting the action space.

Human Feedback for More Complex Agents

In contrast to linear function approximation, where human feedback can directly modify individual Q-weights, incorporating human preferences into more complicated model structure, like Deep Q-Networks (DQN), requires indirect mechanisms that shape the learned action-value function through additional training signals. Humans typically do not give feedback on gradients, network weights, and TD targets. Several indirect approaches are commonly considered, each offering different trade-offs between simplicity, scalability, and stability.

Direct reward shaping. A straightforward option is *direct reward shaping*, where human feedback is converted into an auxiliary reward signal and added to the environment reward. If a human provides scalar feedback r_t^{h} (e.g. an episodic rating mapped

to a per-step bonus), the environment reward can be augmented as

$$\tilde{r}_t = r_t + \lambda r_t^{\mathsf{h}},$$

While simple to implement, this approach suffers from poor credit assignment when feedback is sparse or delayed, and may destabilize learning if human signals are noisy or inconsistent. As a result, direct reward shaping is typically best suited for prototyping rather than long-term deployment.

Preference-based learning. A more principled and effective approach is *preference-based learning*, in which humans provide comparative judgments over actions or trajectories (e.g."solution A is better than solution B"). These preferences are translated into a ranking loss that encourages the Q-network to assign higher cumulative values to preferred behaviors.

When humans provide pairwise preferences over trajectories $\tau_A \succ \tau_B$, the feedback is incorporated via an auxiliary loss defined over cumulative Q-values:

$$Q_\theta(\tau) = \sum_{(s_t,a_t)\in\tau} Q_\theta(s_t, a_t).$$

A common preference loss is the Bradley–Terry formulation

$$\mathcal{L}_{\mathsf{pref}}(\theta) = -\log \sigma\big(Q_\theta(\tau_A) - Q_\theta(\tau_B)\big),$$

where $\sigma(\cdot)$ is the logistic function. The overall optimization objective becomes

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathsf{TD}}(\theta) + \lambda_{\mathsf{pref}}\mathcal{L}_{\mathsf{pref}}(\theta),$$

allowing human judgments to directly shape the relative ordering of Q-values while preserving off-policy learning.

Preference losses can be combined additively with the standard temporal-difference loss used in DQN, allowing automatic reinforcement learning and human guidance to co-exist within a unified optimization framework. This method aligns well with sparse human feedback and preserves the off-policy nature of DQN.

Reward modeling. An increasingly popular alternative is to train a separate *reward model* from human feedback, following the reinforcement learning from human feedback (RLHF) paradigm. In this setting, human preferences are used to learn a parametric reward function, which then replaces or augments the hand-crafted reward during DQN training. This decouples preference learning from policy optimization, improves scalability, and provides a clean interface for integrating feedback from multiple users. However, it introduces additional model complexity and training overhead.

Human Steering via Behavioral Biases and Policy Priors. Beyond reward shaping and preference-based learning, human influence can be incorporated into DQN training by

steering the agent's behavior without modifying the underlying temporal-difference objective.

One effective class of approaches biases the *behavior policy* rather than the learned value function, for example by restricting or softly weighting admissible actions based on heuristic or human-derived rules. In the aircraft conflict-resolution task, such rules may encode domain knowledge such as maintaining the original heading in the absence of predicted conflicts, favoring minimal heading changes, and prioritizing prompt return-to-route maneuvers after avoidance. Because DQN is trained off-policy, these behavioral biases do not alter the Bellman target and therefore preserve stability and convergence properties.

Related techniques include residual reinforcement learning, in which the agent learns small corrections on top of a baseline heuristic controller, and demonstration-biased replay, where human or scripted trajectories are oversampled during training without introducing imitation or preference losses. Collectively, these methods provide a stable and scalable means of nudging learned policies toward human-acceptable behavior while retaining the theoretical simplicity of standard DQN updates.

Limitations of Expert Demonstrations

The current expert demonstration mechanism relies on a deterministic Velocity Obstacle (VO) *pass-behind* rule to provide guidance during exploration. While this heuristic is operationally sound, inline with ATCO "best practices", and ensures conflict-free trajectories in most encounters, it introduces several limitations when used as a source of expert supervision for reinforcement learning.

First, the pass-behind rule represents a *single, fixed avoidance strategy*. As a result, the learning agent is repeatedly exposed to the same type of maneuver, independent of encounter geometry or alternative feasible solutions. This induces a strong inductive bias in the learned Q-function, causing the agent to default to the demonstrated behavior even in situations where other actions (e.g., passing ahead, smaller lateral deviations, or speed adjustments) would be equally safe or more efficient. Empirically, this manifests as reduced policy diversity and limited generalization outside the demonstrated regime.

Second, the expert policy is *myopic* with respect to long-term objectives such as minimizing cross-track deviation, reducing total maneuver count, or optimizing recovery to the original flight path. The VO rule focuses exclusively on instantaneous collision avoidance and does not encode preferences for smoothness, early restoration of heading, or strategic timing of maneuvers. Consequently, the reinforcement learner must infer these objectives solely from the reward signal, which can conflict with or be overshadowed by the demonstrated action choices.

Third, because demonstrations are injected at the level of individual actions, the expert does not provide guidance on *temporal structure*. In particular, the demonstrations do not explicitly encode the desired two-phase behavior of (i) a single decisive avoidance maneuver followed by (ii) a single recovery maneuver. Without sequence-

level supervision, the agent may learn to repeatedly reapply avoidance actions, leading to maneuver budget depletion or oscillatory behavior.

Several improvements can address these limitations. A first recommendation is to introduce *multi-modal expert policies*, for example by randomly selecting between pass-behind, pass-ahead, or speed-based resolution strategies when multiple safe options exist. This would reduce demonstration bias and expose the agent to a broader set of valid behaviors. Second, expert actions can be augmented with *context-dependent weighting*, such that demonstrations are emphasized only during the conflict-resolution phase and gradually faded during recovery, allowing the reward function to dominate heading restoration and cross-track minimization. Third, demonstrations can be elevated from single-step actions to *short expert trajectories*, enabling the agent to learn temporal patterns such as "avoid once, then recover once."

Overall, while the VO pass-behind rule provides a valuable mechanism, its deterministic and single-strategy nature limits the richness of the learned policy.

6 Lessons Learned

Personally, the development of the Q-learning demonstrator was much fun! The main lesson I learned from making it, is that it underscored the substantial amount of domain knowledge required to obtain meaningful learning behavior, even in a comparatively simple two-dimensional aircraft conflict-resolution scenario. In practice, successful training required careful feature design (and normalization), reward shaping, phase gating between conflict resolution and recovery, maneuver budgeting, regularization, and controlled exploration scheduling. Without these design elements, the agent either converged to trivial local optima (e.g., bias-dominated policies or persistent maneuvering) or failed to learn stable avoidance behavior at all.

This experience underscores that reinforcement learning does not eliminate the need for problem-specific insight; rather, it shifts it into the design of state representations, reward functions, and training constraints. In contrast, simple heuristic approaches—such as rule-based Velocity Obstacle pass-behind strategies—produced operationally acceptable results almost immediately and with minimal tuning. While heuristics may lack adaptability or optimality in more complex environments, they do offer robustness and interpretability that are difficult to achieve with learning-based methods without significant engineering effort.

In the present 2D setting, reinforcement learning may of course be considered overly complex relative to the problem difficulty. However, the exercise has been instructive to me in clarifying when RL is justified: namely, when the problem dimensionality, uncertainty, or objective trade-offs exceed what can be reliably captured by fixed rules. The demonstrator may thus serve as a useful case study in balancing learning-based approaches against heuristic solutions, particularly in safety-critical domains where predictability, transparency, and validation requirements play a central role.