Collision Avoidance Using Deep Reinforcement Learning

Bart Bacon
NASA Langley Research Center

SciTech 2022, January 4

Background: Multi-Agent Collision Avoidance with DLR

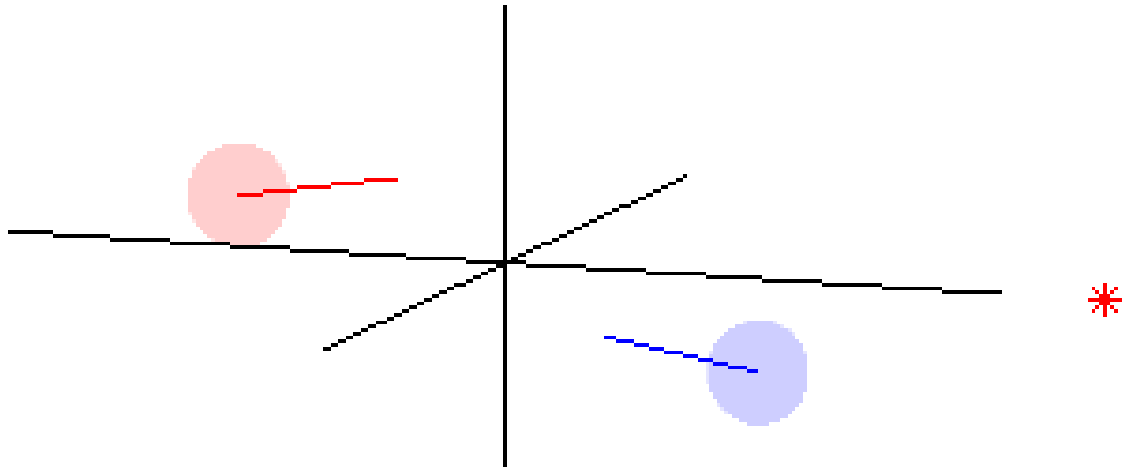
- Multi-Agent CA not just a problem for urban air vehicles
  - Multi agent coordination
  - Autonomous navigation through human/robot crowds
  - Computer crowd simulation
- Challenges are similar
  - Requires predicting the other agent's motion/anticipating interaction patterns
  - Computationally tractable for real time implementation
- Deep Reinforcement Learning (DRL) has been used to improve baseline policies for autonomous navigation through moving crowds (How, MIT 2016-current)

Desirable Features of Existing Approach

- DRL offloads the online computation of anticipating interaction to an off-line learning procedure
  - Computationally tractable for real time implementation
- Solution starts with training from a known solution, i.e. supervisory learning. DRL uses an epsilon-greedy version of baseline policy to explore other options and improve it.
- Each agent is aware of only a subset of the other agent's states
- Solution is completely decentralized as it would have to be for the urban air collision avoidance problem

Problem Statement

- **Objective** Two agents, each with a set of states observable and unobservable states to the other, attempt to reach their goals in a minimum amount of time without colliding
  - Observable: positions, velocities, size
  - Unobservable: preferred speed and direction, goal position



- **Sequential decision-making problem** At the start of a specified time interval each agent will select a new velocity vector, without knowledge of what the other agent selects, to reach its goal and avoid collision
- **Assumption** Each agent will select a new velocity vector using the same **policy**, a function of the agent's complete state and the observable portion of the other agent's state

DRL Approach to Collision Avoidance

Approach

- Start with a known solution, using supervisory learning to train a network on results to create a baseline policy
- Exploit Deep Reinforcement Learning (DRL) to explore other options

Known Solution

- Optimal Reciprocal Collision Avoidance (ORCA)
  - In the event of an imminent collision, each agent will split the required change in relative velocity to avoid the collision in a manner that minimizes the course correction from its desired course.
  - No optimization of time to goal

DRL

- Value Function, representing the return for taking the best action, determines the next action

Reward and Return

Reward Over each interval, award the agent for reaching the goal and penalize the agent for getting too close or colliding

$$R\left(\boldsymbol{s}_t^{jn}, \boldsymbol{a}_t\right) = \begin{cases} -0.25 & if\ d_{min} \leq 0 \\ -0.25 + d_{min}(0.25/0.2) & else\ if\ d_{min} < 0.2 \\ 1 & else\ if\ \boldsymbol{p} = \boldsymbol{p}_g \\ 0 & o.w. \end{cases}$$

Return Over the entire trajectory, the accumulated sum of discounted Rewards. Immediate rewards are deemed higher than future rewards

$$V^*\left(\boldsymbol{s}_{t_k}^{jn}\right) = \sum_{t=t_k}^{T-1} \gamma^{(t-t_k)v_{pref}} R\left(\boldsymbol{s}_t^{jn}, \boldsymbol{a}_t^*\right) + \gamma^{(T_g-t_k)v_{pref}} R\left(\boldsymbol{s}_T^{jn}, \boldsymbol{a}_T^*\right) \quad \forall\ t_k \leq T-1$$

$$\pi^*\left(\boldsymbol{s}_t^{jn}\right) \rightarrow \boldsymbol{a}_t^*$$

Value Function

Bootstrap Expression

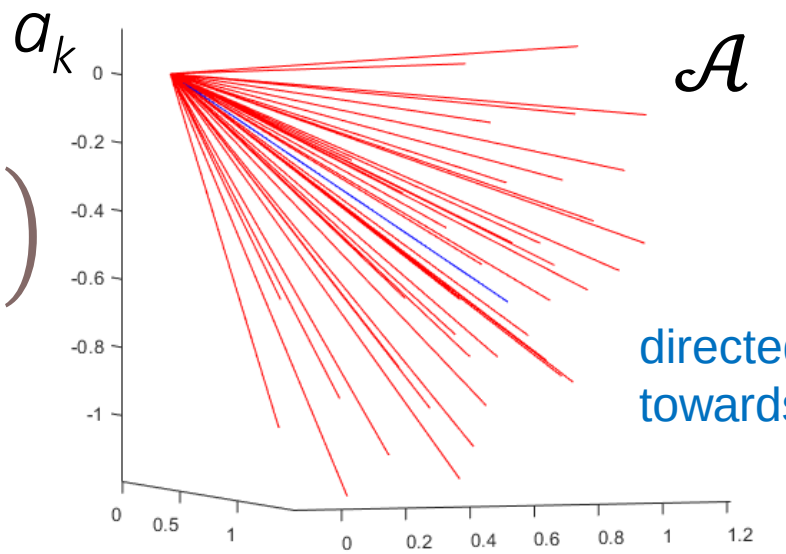
$$\boldsymbol{V}^*\left(\boldsymbol{s}_t^{jn}\right) = R\left(\boldsymbol{s}_t^{jn}, \boldsymbol{a}_t^*\right) + \bar{\gamma} V^*\left(\boldsymbol{s}_{t+\Delta T}^{jn}\right)$$

if $\boldsymbol{s}_{t+\Delta T}^{jn}$ is within goal on next $\boldsymbol{a}_{t+\Delta T}^*$

$$\bar{\gamma} = \gamma^{(T_G - t)v_{pref}} \qquad V^*\left(\boldsymbol{s}_{t+\Delta T}^{jn}\right) = 1$$

Action Selection

$$\pi\left(\boldsymbol{s}_t^{jn}\right) = \underset{\boldsymbol{a}_t \in \mathcal{A}}{\operatorname{argmax}} \left(R\left(\boldsymbol{s}_t^{jn}, \boldsymbol{a}_t\right) + \bar{\gamma} V\left(\hat{\boldsymbol{s}}_{t+\Delta T}^{jn}\right) \right)$$



- The next joint state is an estimate of both the agent's next state as a result of the action selected and the observable portion of the other agent's state
- The estimate of the agent's next state must be checked on whether the goal can be obtained from it before $t + 2\Delta T$

Agent-Centric Parameterization of Net

- Remove redundancy in joint state

$$rotate\left(s^{jn}\right) \rightarrow s'$$

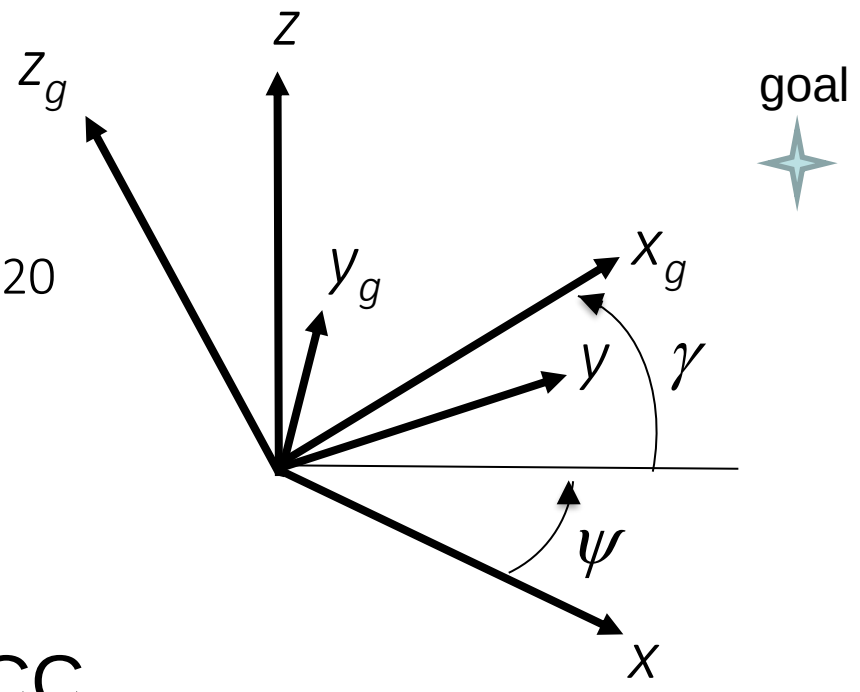
$$s' = [d_g, v_{pref}, v', r, \psi', \gamma', \tilde{v}', \tilde{p}', \tilde{r}, r + \tilde{r}, \ldots$$

$$\cos(\psi'), \sin(\psi'), \sin(\gamma'), d_a] \in \mathbb{R}^{20}$$

$$d_g = \left\| p_g - p \right\|_2 \qquad d_a = \left\| p - \tilde{p} \right\|_2$$

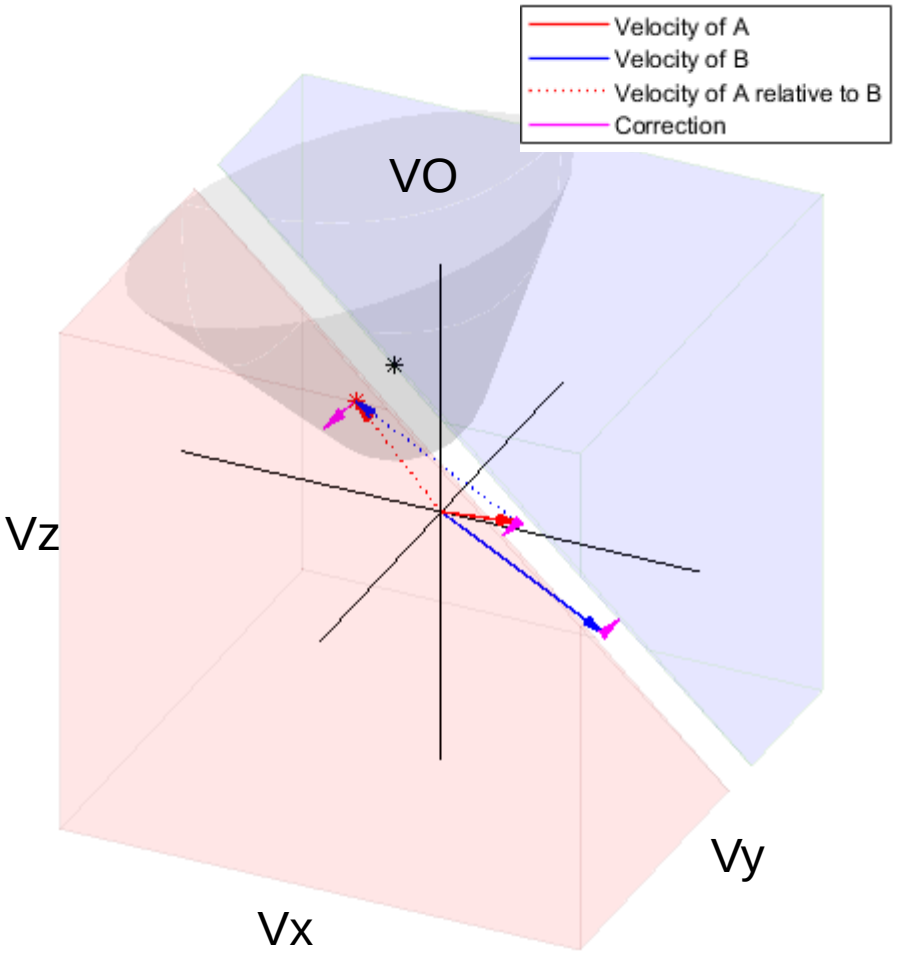
ψ', γ' current heading and flight path in ACC

- origin at agent position
- x-axis directed to goal



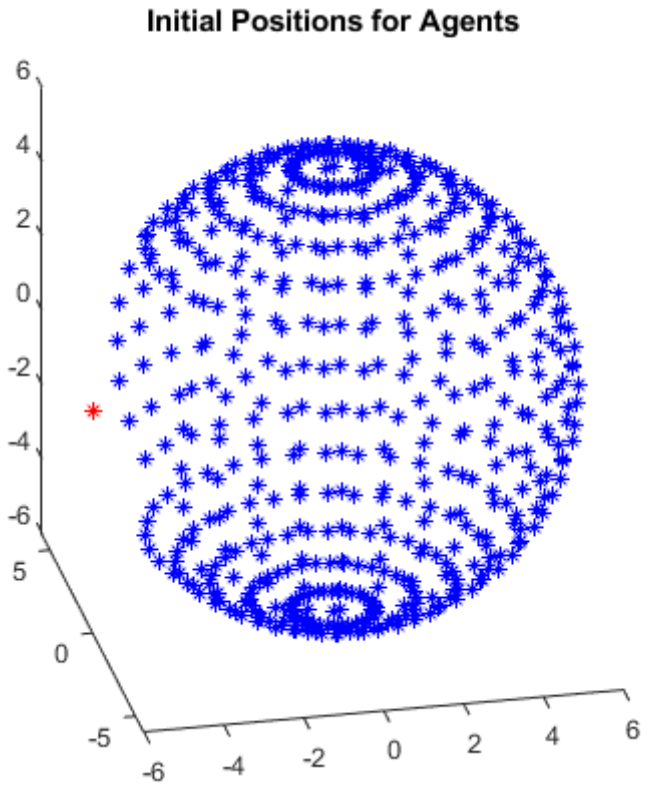
Optimal Reciprocal Collision Avoidance (ORCA)

- A Velocity Obstacle (VO) is defined based on the size and relative position of the agents and the time interval of interest.
- If the relative velocity of one agent to the other falls within the VO cone and is unchanged, the agents will collide within the time interval.
- The closest point on the surface of the VO cone determines the correction required in the relative velocity to avoid collision.
- If the agents are reciprocal, half the correction is applied to each agent's velocity, the admissible velocity set resides in a half space.
- A convex hull is created as more agents are added. The velocity in the region closest to the desired velocity determines the course change.

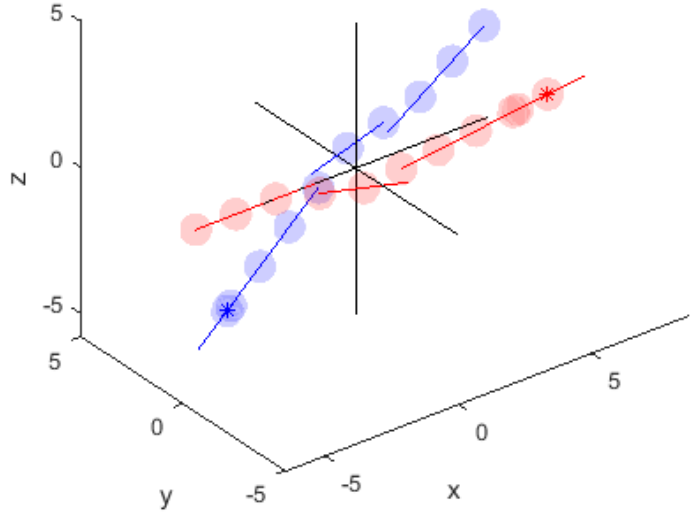


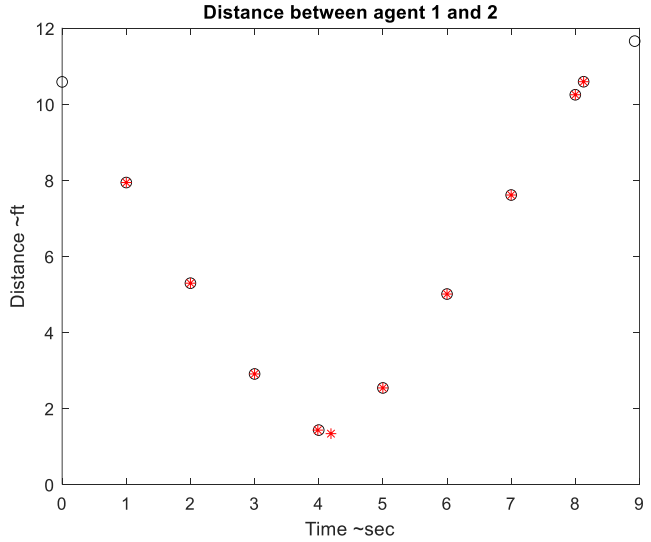
Training Set and Network

- ORCA generated 565 episodes
- Original set: 5650 state-value pairs
- Expanded set: 197750 state-value pairs
- Net consisted of three hidden layers of widths (250,200,200), activation nonlinearity: ReLU



Original Set

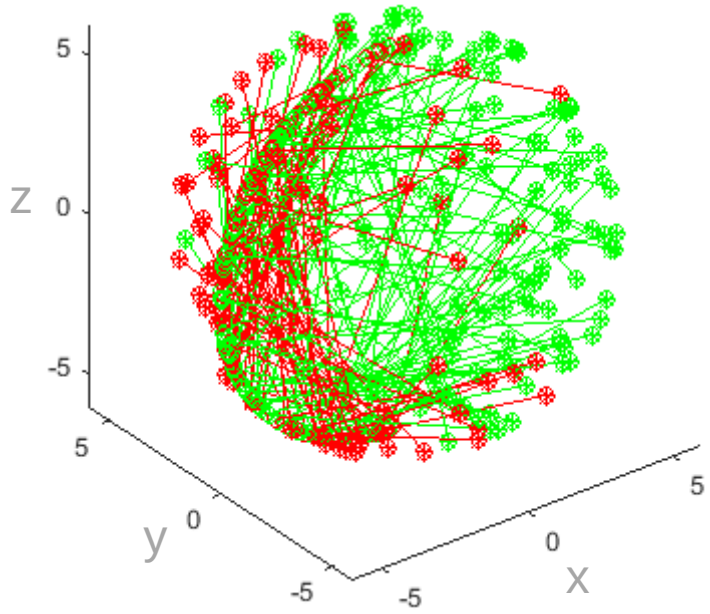




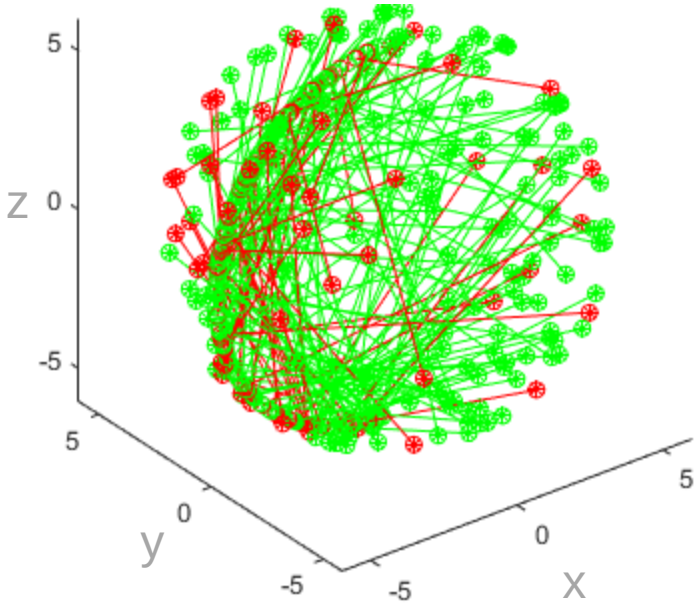
Supervisory Training Results

- ORCA trajectories converted to state-value pairs
- Adam solver: 5000 epochs using mini-batches of 500 state-value pairs

Original training set: 565
61% collision free

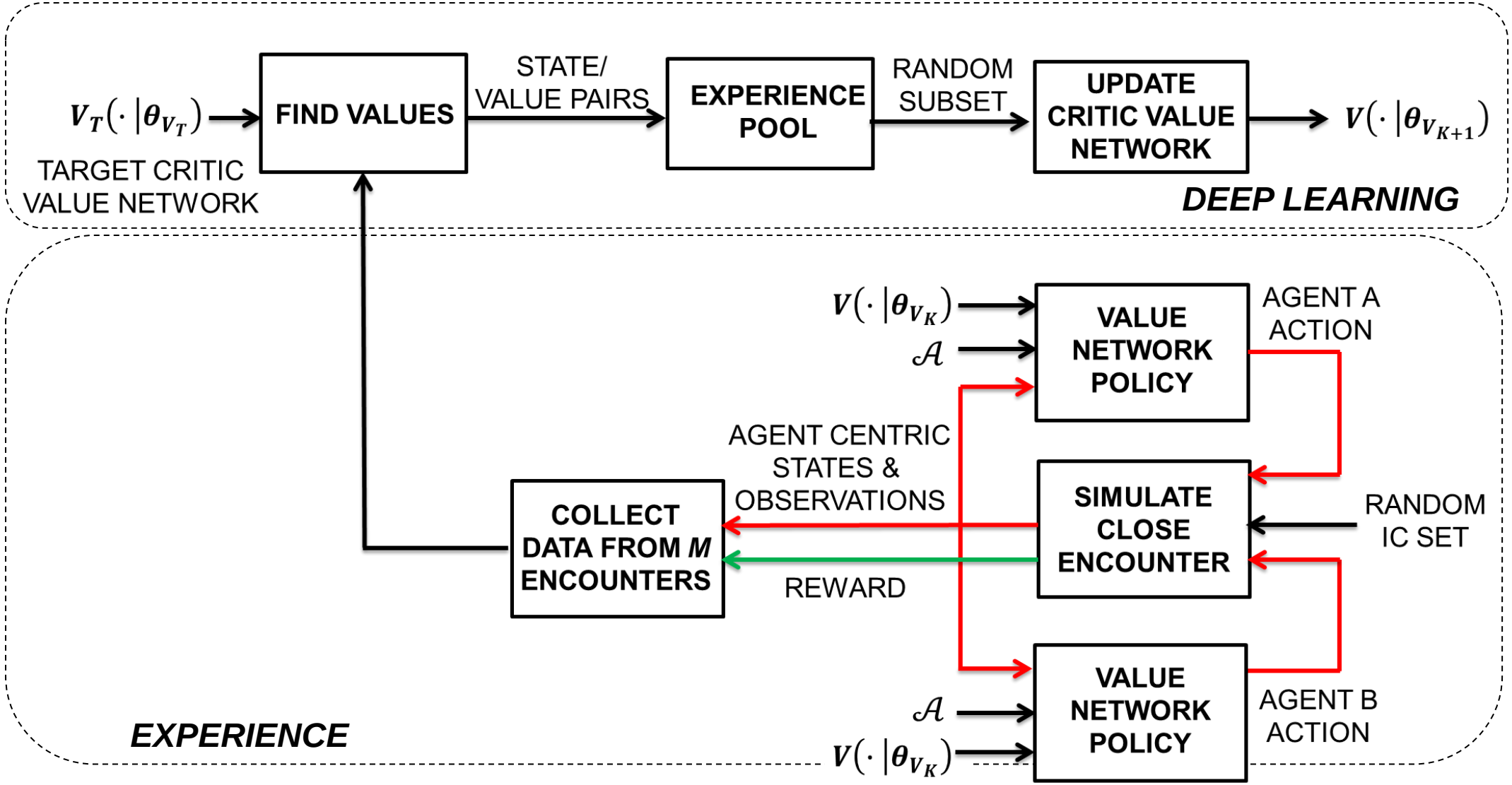


Expanded training set: 19775
76.4% collision free



$$\underset{\boldsymbol{\theta}_V}{\operatorname{argmin}} \sum_{j=1}^{N_{eps}} \sum_{k=t_0}^{T_j} \left(y_k - V\left(\boldsymbol{s}_k^{jn} \middle| \boldsymbol{\theta}_V \right) \right)^2$$

Deep Reinforcement Learning: One Episode



Find Values

- Motivated by Q-Learning, use $\boldsymbol{f}(S_i, A_i, R_i, S_i') \rightarrow y_i$

Deep Double Q Network (DDQN)

$$A'_{MAX} = \underset{A' \epsilon \mathcal{A}}{\operatorname{argmax}}\, Q\left(S_i', A' \middle| \theta_Q\right)$$

$$y_i = R_i + \gamma Q'\left(S_i', A'_{MAX} \middle| \theta_{Q'}\right)$$

Deep Double V Network (DDVN): two step to explicitly see the next action

$$V\left(\boldsymbol{s}^{jn}_{t_{k+1}} \middle| \boldsymbol{\theta_V}\right) \leftarrow R\left(s_{t_{k+1}}, a_{t_{k+1}}\right) + \bar{\gamma}_{t_{k+1}} V'\left(\boldsymbol{s}^{jn}_{t_{k+2}} \middle| \boldsymbol{\theta}_{V'}\right)$$

$$\bar{\gamma} \leftarrow \bar{\gamma}_{t_k}$$

$$y_{t_k} \leftarrow R\left(s_{t_k}, a_{t_k}\right) + \bar{\gamma} V\left(\boldsymbol{s}^{jn}_{t_{k+1}} \middle| \boldsymbol{\theta_V}\right)$$

DRL Parameter Space

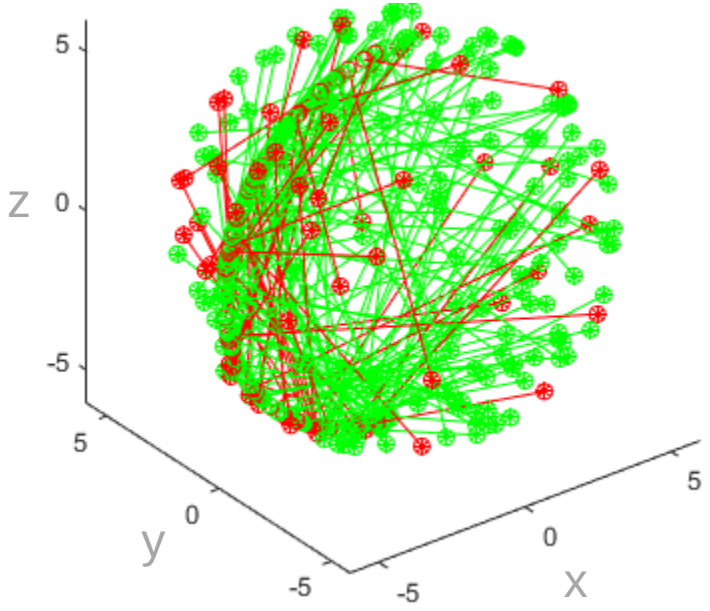
- Number of episodes: 800
- Number of encounters per episode: 10
- Size of random sample drawn from experience pool: 1000
- Number of epochs per update: 100
- Size of mini-batch size for update: 1000
- Epsilon schedule for random action choice, function of episode

$$\begin{bmatrix} 1 & 400 & 800 \\ .3 & .1 & .1 \end{bmatrix}$$

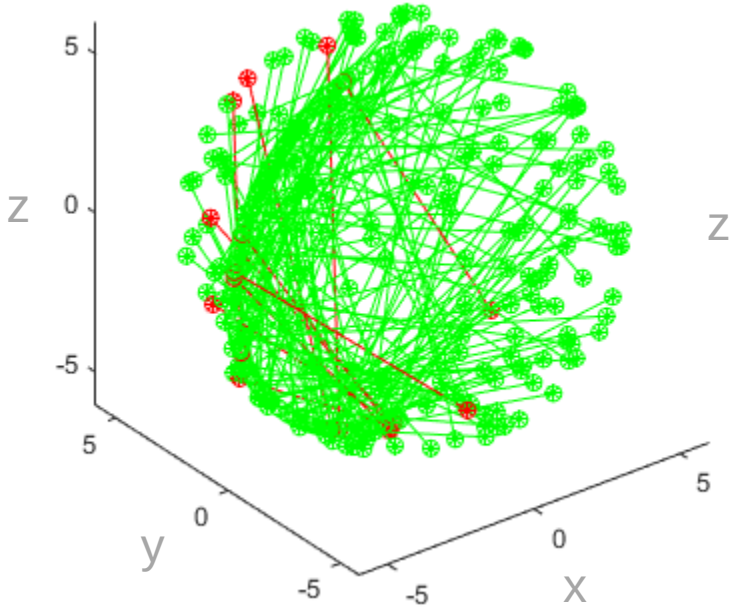
- Frequency of evaluation: every 50 episodes
- Network structure

Collision Avoidance after Two Learning Sessions

Initial 250-200-200 net
76.5% collision free
200 cases

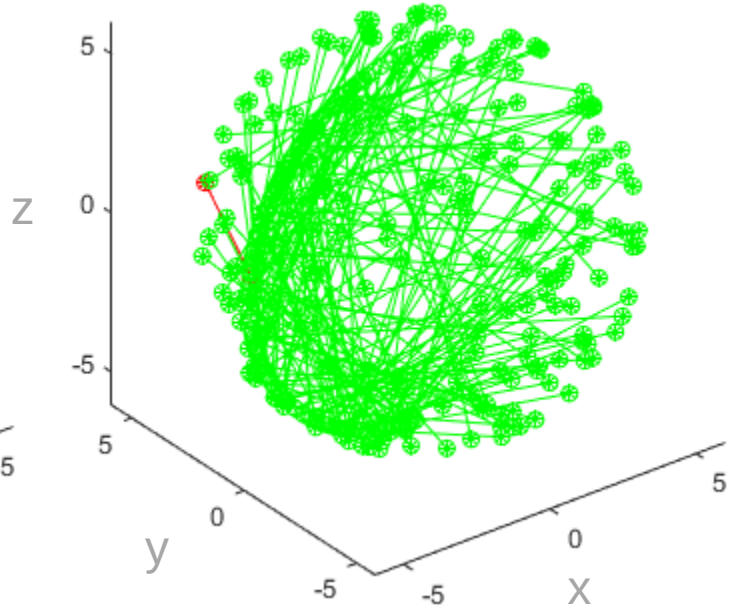


DRL: 250-200-200 net
95% collision free
200 cases



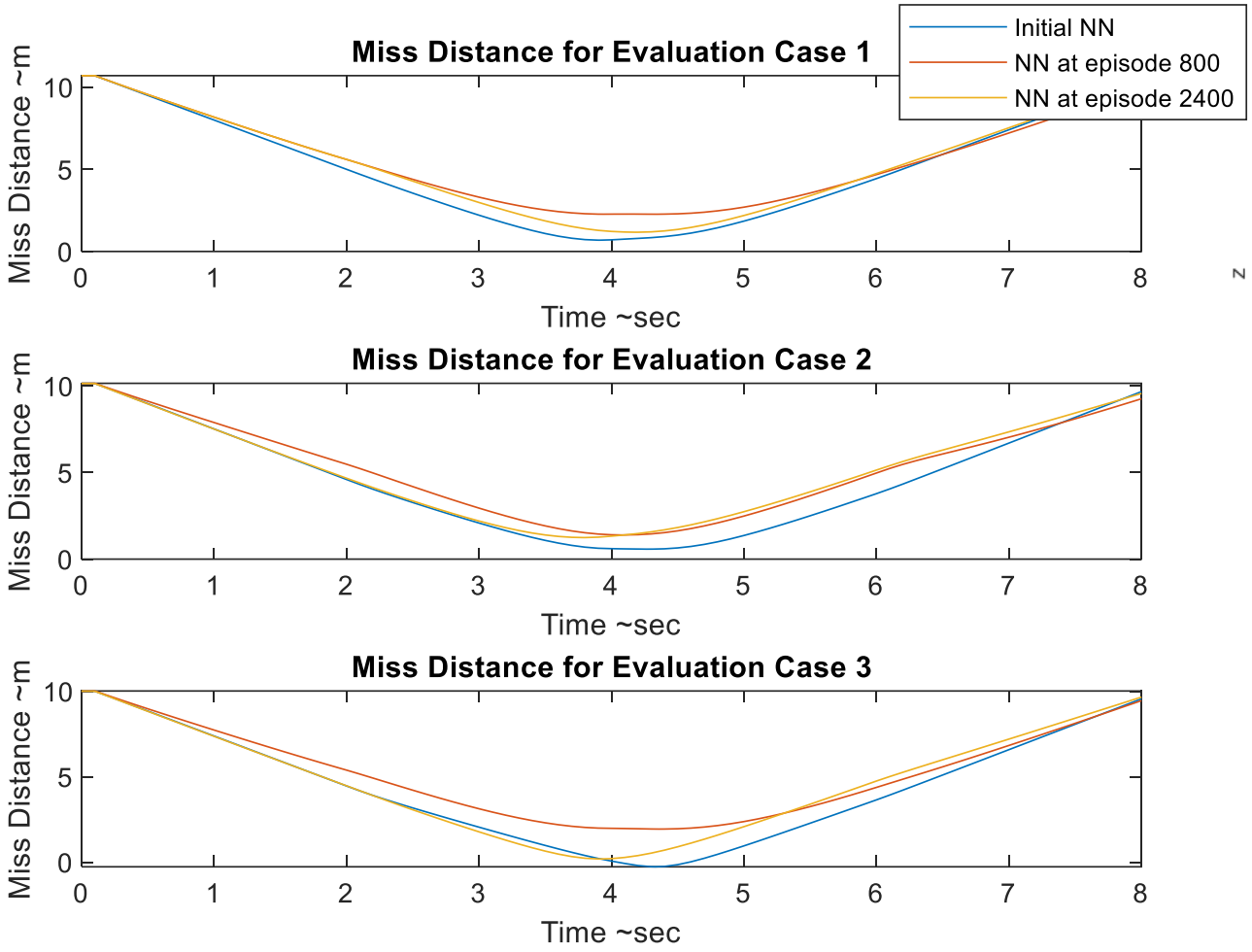
First Learning Session

DRL: 250-200-200 net
99.5% collision free
200 cases

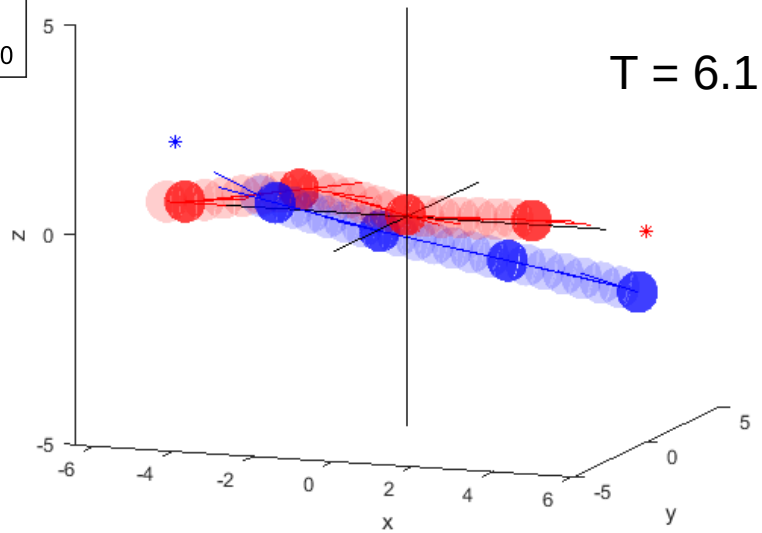


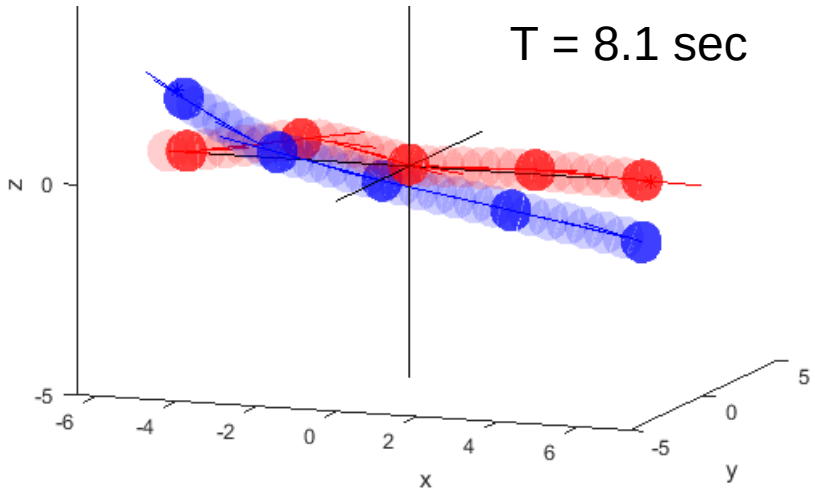
Second Learning Session

Evaluation Cases for Two Learning Sessions

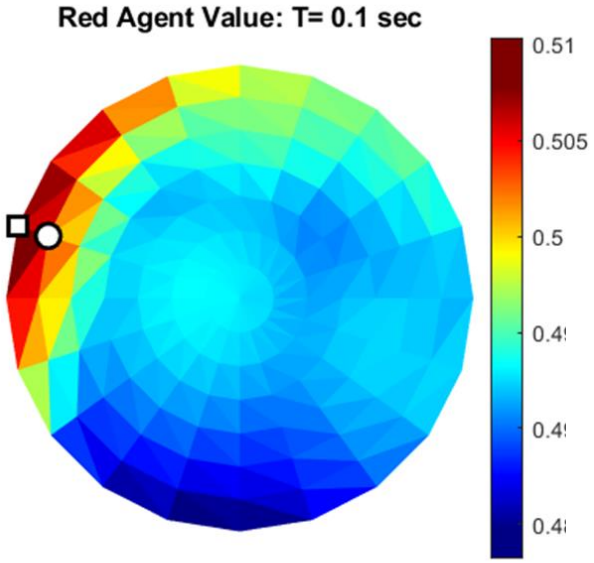


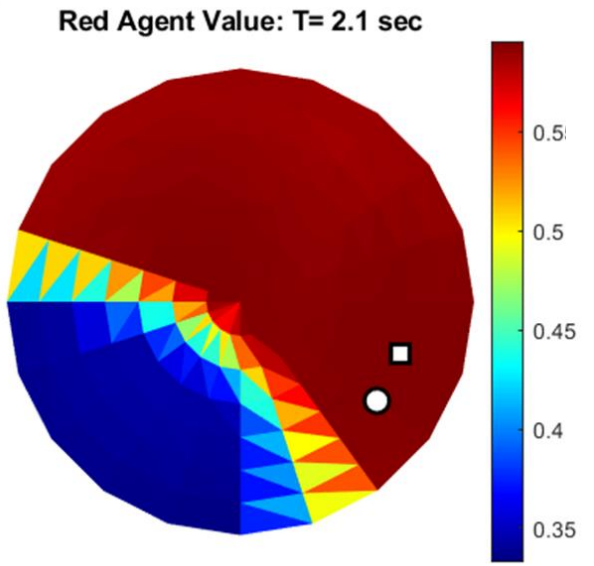
Evaluation case 3: 2400 episode



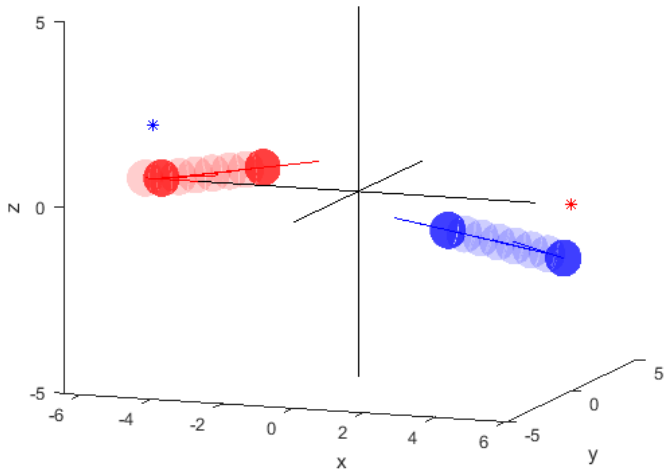


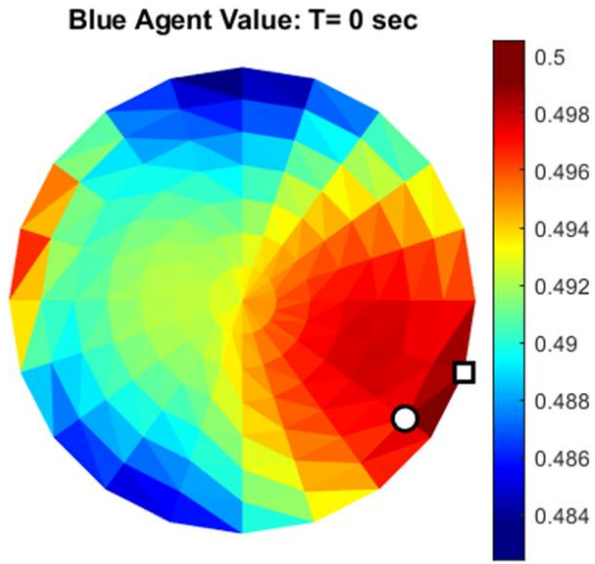
Evaluation Case 3: Decisions from Value Network

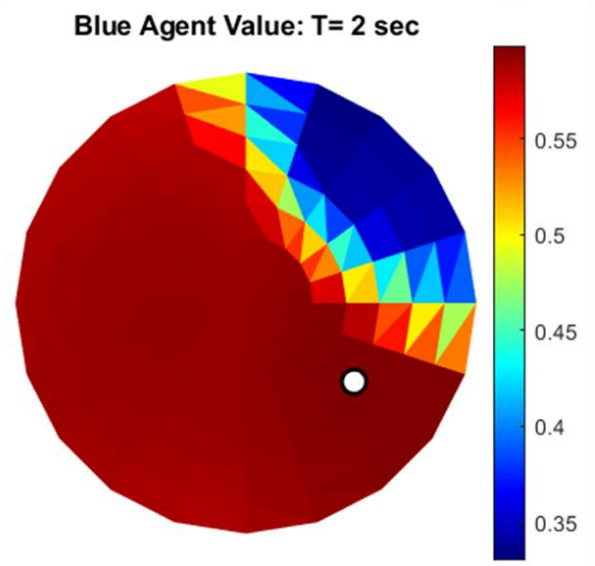




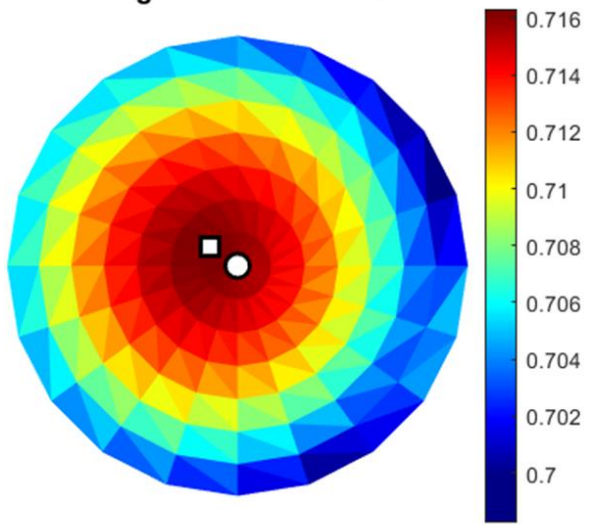
Time: 0-2.1 sec

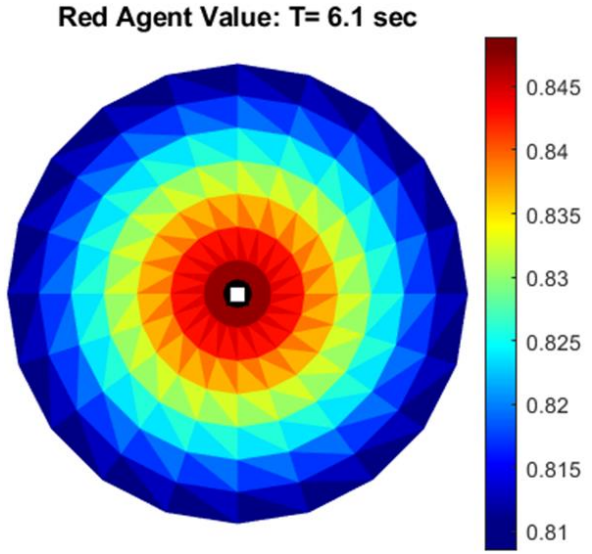




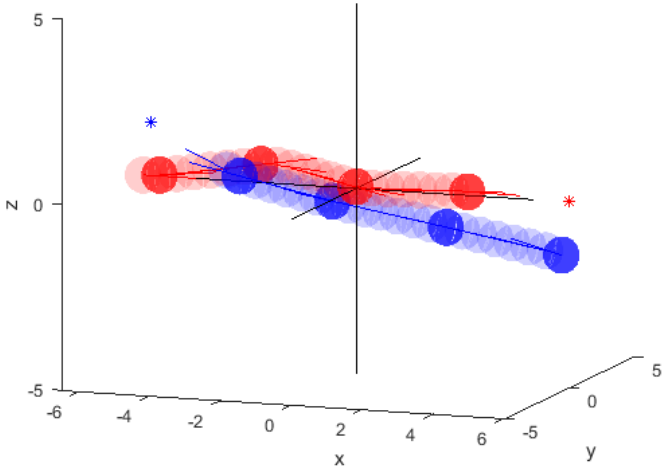


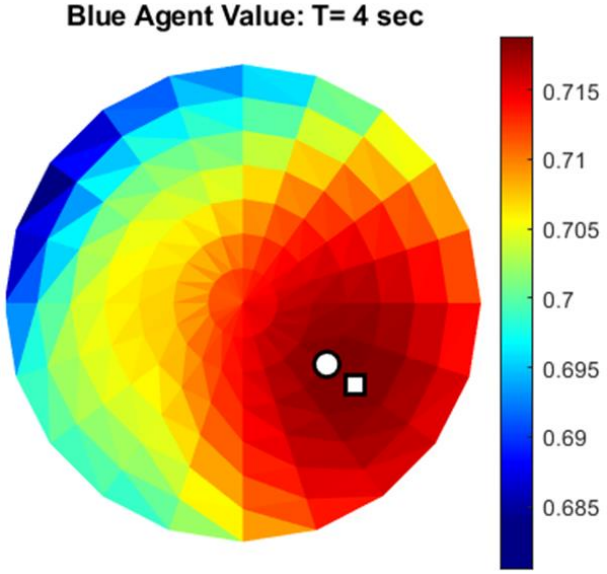
Evaluation Case 3: Decisions from Value Network

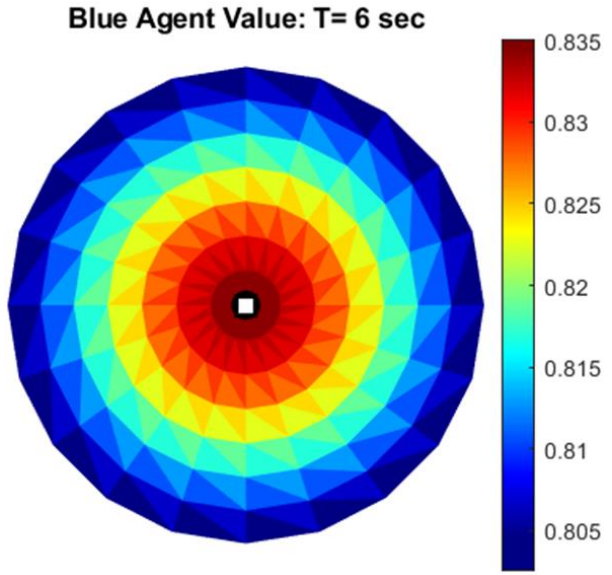




Time: 0-6.1 sec

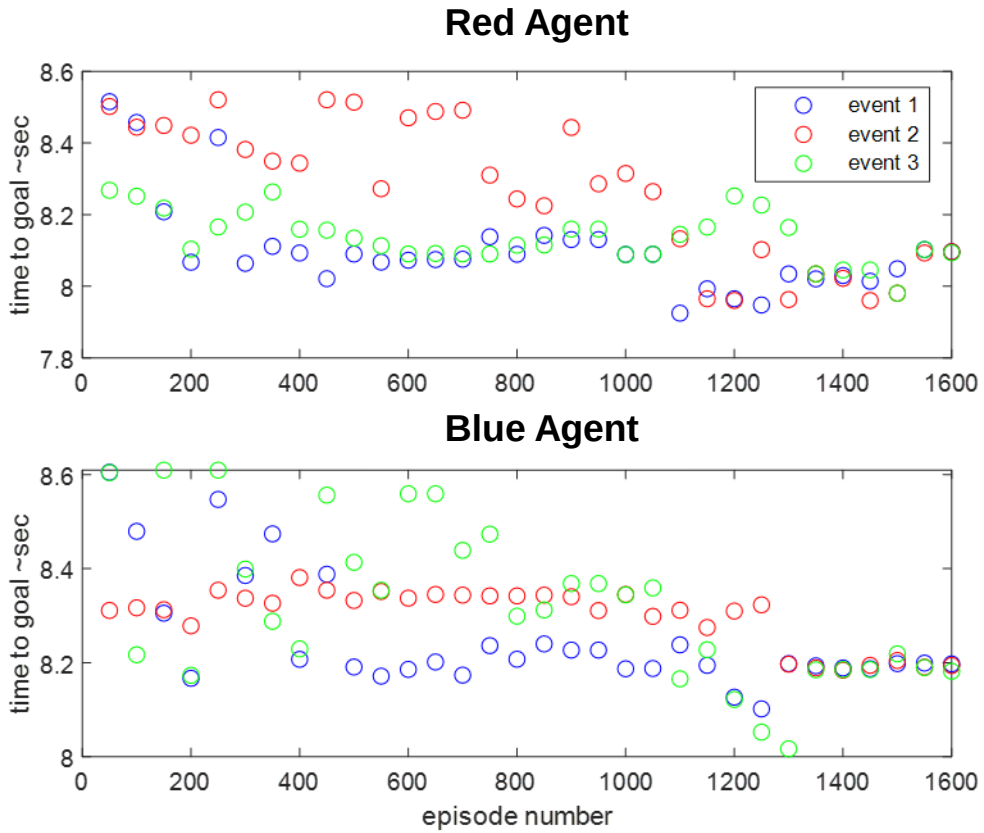




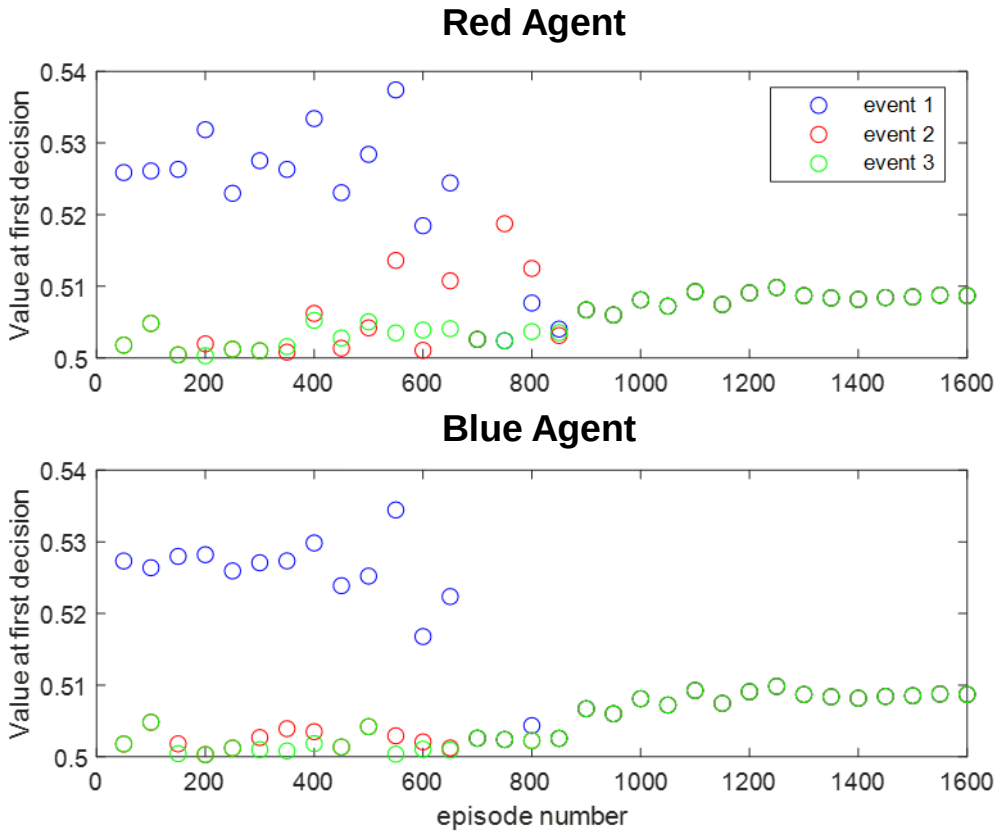


Evaluation Results over Second Learning Session

- 1600 episodes: time to goal & value at first decision state
- Epsilon is 0.05 beyond episode 800



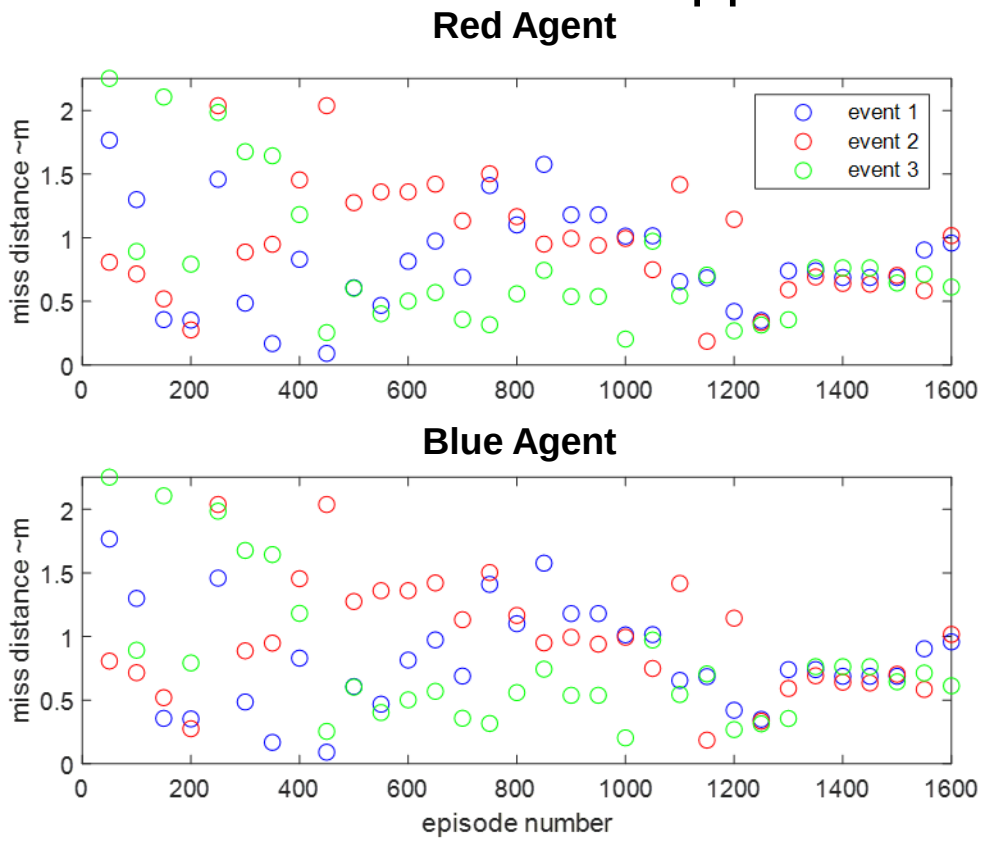
Time to Goal



Value at First Decision

Evaluation Results over Second Learning Session

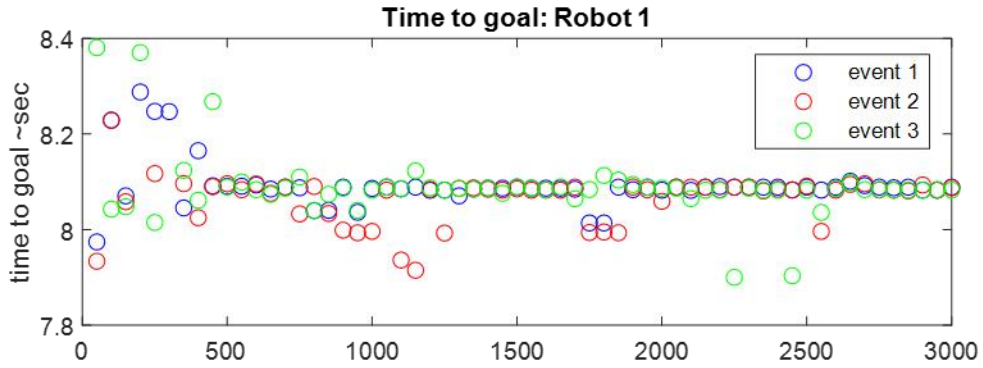
- Evaluation cases over 1600 episodes
- Miss distance ends at a factor of three above upper bound of penalty

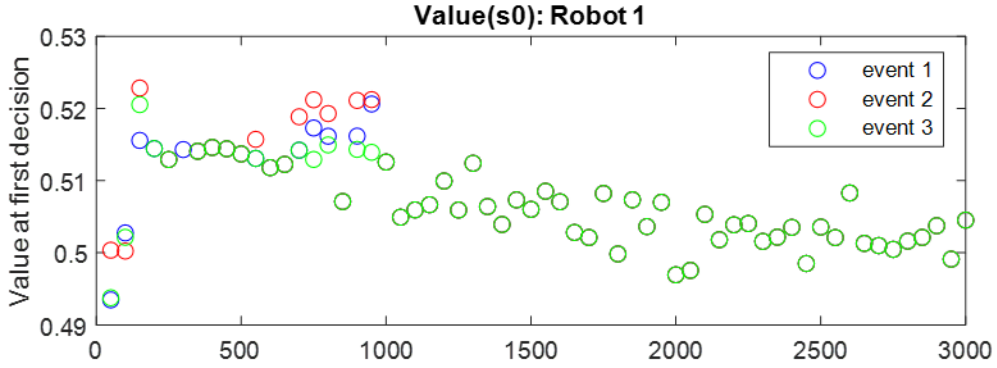


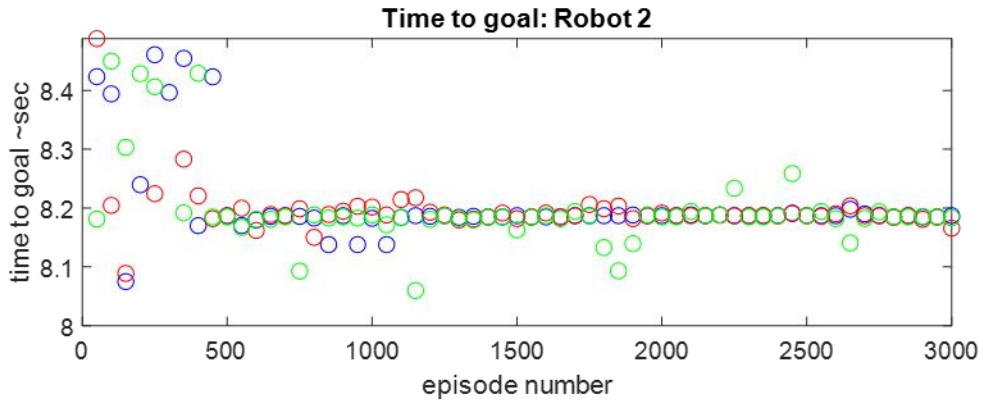
Miss Distance

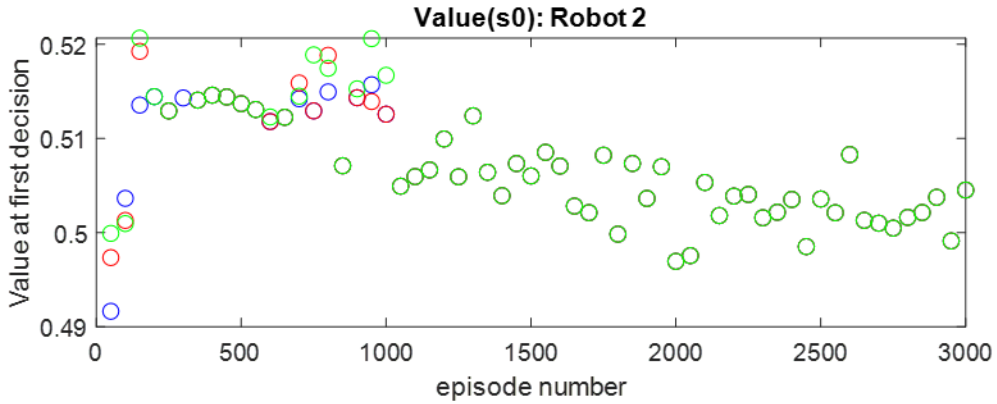
Robustness of Learning Session: Repeat Experiment

- Evaluation results from 3000 episodes
- Time to goal & value of state and first decision point



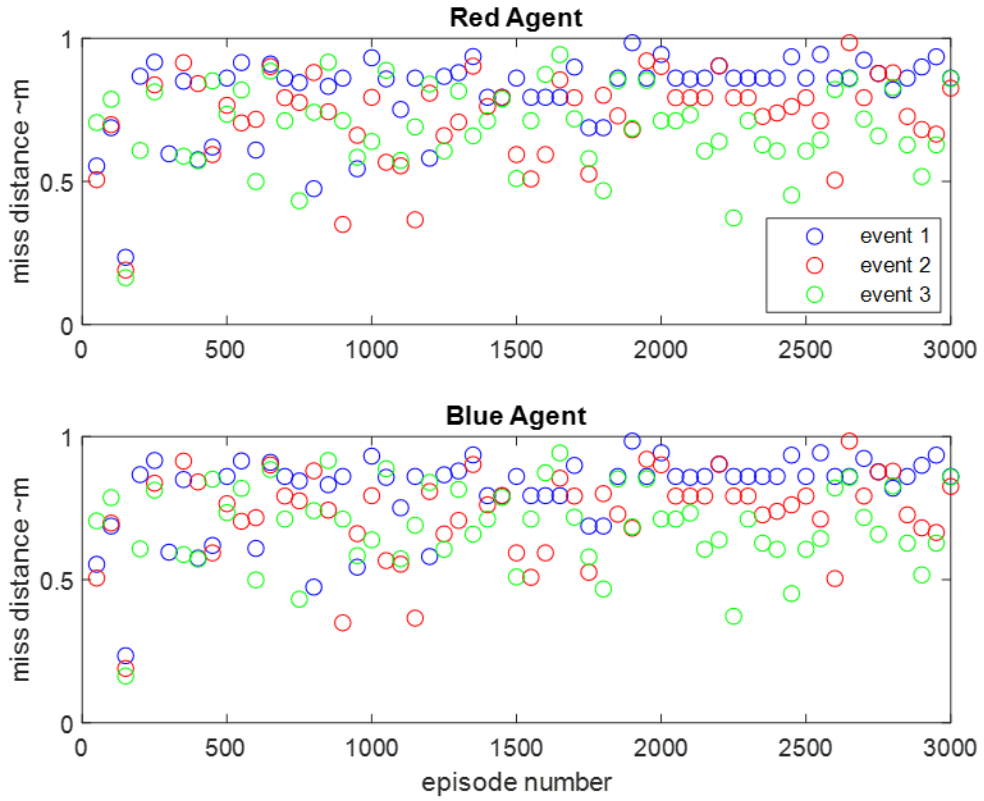






Evaluation Cases for Repeat Learning Session

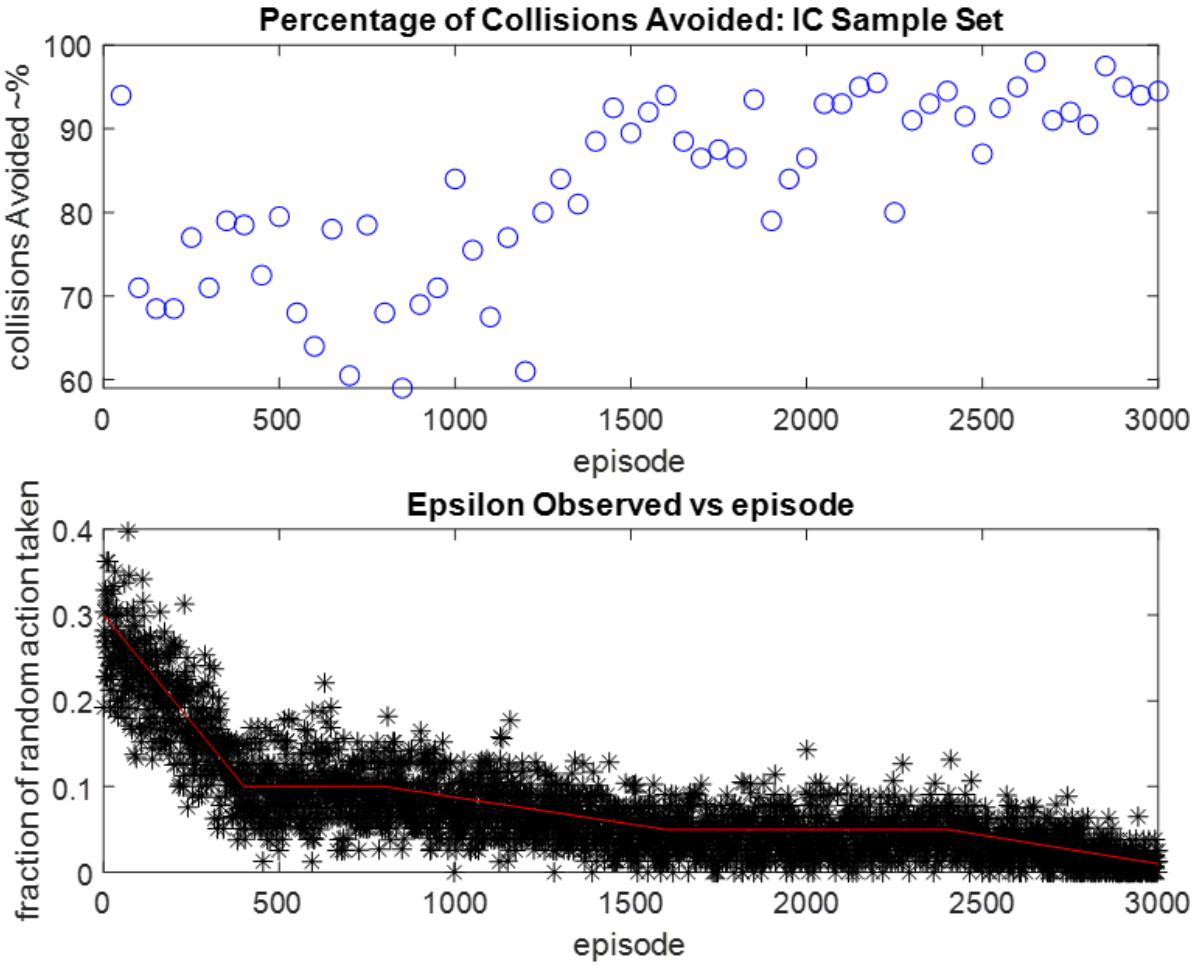
- Evaluation cases over 3000 episodes: Miss Distance
- A repeat of conservative results, penalty starts at smaller miss distance



Miss Distance

Collision Avoidance Performance of Repeat Session

- Network Performance at evaluation points over IC sample set

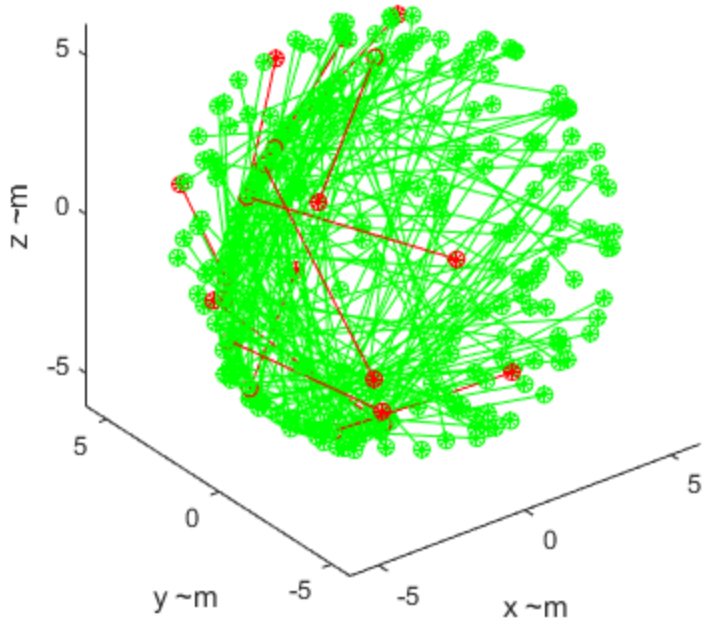


AFTER 3000 EPISODES:

DRL: 250-200-200 net

94.5% collision free

200 cases



A lot of fluctuation from

Conclusions

The 3D extension has not lived up to the promise of the 2D collision avoidance algorithm: mixed results

- Lack of agent-centric coordinate system bullet proof to arbitrary rotations of environment
- Upward trend to avoiding collisions over the IC sample set encouraging
- A lack of continuous improvement in the value network.
- Rapid changes in the network over 50 episodes are the culprit
  - Larger network needed?
  - A different set of training parameters?
- Not surprising that the expected trend towards a time optimal collision avoidance solution not realized yet.