BTP-1 Final Report: Adversarial learning based Mutli UAV motion planning

Shivam Mittal
CSE
2022101105

Harpreet Singh
CSE
2022101048

Abstract

This project presents an adversarial learning-based framework for motion planning in Multi-Unmanned Aerial Vehicle (UAV) systems. The primary goal is to develop collision-free, efficient trajectories for multiple UAVs navigating in dynamic environments. A hybrid expert system combining Velocity Obstacle (VO) and Rapidly-exploring Random Tree (RRT) is implemented in 2D space as a baseline for future expert policy development. Currently, Generative Adversarial Imitation Learning (GAIL) is trained using collision-avoiding expert demonstrations to learn basic navigation behaviors. The framework is further extended to high-fidelity 3D simulation using Microsoft AirSim, where GAIL is applied for end-to-end policy learning. Initial results demonstrate that the learned policies achieve effective obstacle avoidance and coordination, highlighting the potential of adversarial imitation learning for decentralized UAV motion planning. We make our code public on the following repository[1]

1 Velocity Obstacle (VO) Method for UAV Collision Avoidance

In the VO approach, each UAV treats other agents and obstacles as moving objects in the plane and computes a *velocity obstacle* – the set of its own velocities that would lead to collision if the other object maintains its current motion. In other words, a VO is a cone in velocity space representing unsafe velocities. Geometrically, for two circular UAVs A and B (with positions $\mathbf{x}_1$, $\mathbf{x}_2$ and radii r_1, r_2), the VO for A induced by B is the set of velocities $\mathbf{v}$ such that the relative velocity $\mathbf{v} - \mathbf{v}_B$ eventually drives A into B's expanded collision region. Formally,

$$VO_{A|B} = \{\mathbf{v}_A \,|\, \exists t > 0 : (\mathbf{v}_A - \mathbf{v}_B)t \in D(\mathbf{x}_B - \mathbf{x}_A,\, r_A + r_B)\}\,,$$

where $D(c, r)$ is a disc of radius r centered at c. Intuitively, any chosen velocity of A inside this cone (the grey region in Figure 1) will result in a collision if B does not change course, whereas any velocity outside the cone is guaranteed collision-free under constant motion assumptions.

1.1 Preferred Velocity Calculation

Each UAV has a *preferred velocity* $\mathbf{v}_{\text{pref}}$ pointing toward its current goal. In practice, the preferred direction is the unit vector from the UAV's position towards its target waypoint, scaled by the UAV's preferred speed. For example, if a UAV at $\mathbf{p}$ seeks to go to goal $\mathbf{g}$, then

$$\mathbf{v}_{\text{pref}} = v'_{\text{pref}} \frac{\mathbf{g} - \mathbf{p}}{\|\mathbf{g} - \mathbf{p}\|},$$

[1] https://github.com/Harpreet287/btp1

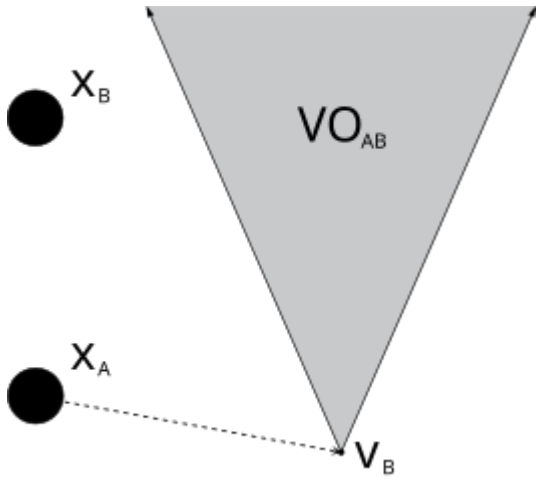


Figure 1: Velocity Obstacle

where v'_{pref} is the chosen cruising speed. (If the UAV is already at or very near its goal, this preferred velocity is taken as zero.) This preferred vector represents the ideal motion toward the goal before considering any obstacles. It is used as a baseline for choosing the actual collision-free velocity.

1.2 Velocity Obstacle Formulation and Geometry

A VO is constructed by "translating" the collision cone of two agents by the obstacle's velocity. Concretely, consider agent A at $\mathbf{p}_A$ with radius r_A, and obstacle B at $\mathbf{p}_B$ with velocity $\mathbf{v}_B$ and radius r_B. First form the *collision cone* $CC_{AB} = \{\mathbf{v} \mid (\mathbf{v} \cdot \mathbf{d}) \geq \|\mathbf{v}\|\|\mathbf{d}\| \cos\theta\}$, where $\mathbf{d} = \mathbf{p}_B - \mathbf{p}_A$ and $\theta = \arccos\left(\frac{r_A + r_B}{\|\mathbf{d}\|}\right)$ is the half-angle of the cone. Then the velocity obstacle is the Minkowski sum of this cone with B's velocity vector:

$$VO_{A|B} = CC_{AB} \oplus \mathbf{v}_B = \{\mathbf{u} + \mathbf{v}_B \mid \mathbf{u} \in CC_{AB}\}.$$

Equivalently, one can think of treating B as a static obstacle by expanding its radius by r_A (forming the disc D above) and then shifting the cone by $\mathbf{v}_B$. In multiple-obstacle scenarios, each UAV forms a VO for every other UAV (or moving obstacle), then takes the *union* of all these cones to define its unsafe velocity region. In summary, a velocity $\mathbf{v}$ for A is **safe** if and only if $\mathbf{v}$ lies outside every VO induced by all other objects.

Mathematically, if there are n other UAVs or moving obstacles indexed by j, the combined VO region is

$$VO_{\text{combined}} = \bigcup_{j=1}^{n} VO_{A|j}.$$

Any candidate velocity $\mathbf{v}$ of A must satisfy $\mathbf{v} \notin VO_{\text{combined}}$ to guarantee no imminent collision. Thus the permissible velocities form the complement of this union of cones.

1.3 Static and Dynamic Obstacles

Static obstacles are handled as a special case of VO by setting their velocity to zero. A fixed obstacle of radius r_O at position $\mathbf{p}_O$ yields a velocity obstacle at $\mathbf{p}_O$ with $\mathbf{v}_O = 0$, i.e., the collision cone emanating from the obstacle's relative position. Dynamic (moving) obstacles simply shift this cone by their current velocity as above. In effect, every other UAV (moving with $\mathbf{v}_j$) produces a cone in A's velocity space; any $\mathbf{v}$ inside that cone leads to intersection of their future paths.

By taking the union of all static and dynamic VOs, each UAV accounts simultaneously for walls, buildings, and other agents. If $\mathbf{v}$ is inside the combined VO region (grey cones), collision is certain; if $\mathbf{v}$ is outside, no collision will occur under the constant-velocity assumption.

1.4 Decentralized Real-Time Collision Avoidance

In practice, each UAV executes VO-based avoidance in real time and in a fully decentralized manner. At each control cycle, a UAV (say A) uses onboard sensing or communication to obtain the positions $\mathbf{p}_j$, velocities $\mathbf{v}_j$, and sizes of nearby UAVs and obstacles. *It then independently constructs each VO*$\{A|j\}$ as described above. Using its own computation, A then selects a new velocity $\mathbf{v}_A$ that lies outside the union of all VOs. Typically $\mathbf{v}_A$ is chosen to be the feasible velocity *closest* to the preferred velocity $\mathbf{v}_{\text{pref}}$; for example, one can solve a small optimization or linear program to minimize $\|\mathbf{v} - \mathbf{v}_{\text{pref}}\|$ subject to $\mathbf{v} \notin \text{VO}_{\text{combined}}$.

In cases where no collision-free velocity exists within the UAV's kinematic limits, the algorithm can apply a penalty (e.g. slow down or allow a slight risk), but this situation is usually avoided in well-spaced scenarios. The process repeats at each time step: each UAV independently recalculates its VOs and picks a new safe velocity. Because each agent does its own calculations without a central planner, the system is fully decentralized. (This also matches the principle of Reciprocal Velocity Obstacles (RVO) or Optimal Reciprocal Collision Avoidance (ORCA), where each agent "shares" responsibility but still computes its own velocity update.

The decentralized VO approach has been demonstrated in 2D UAV simulations: for instance, Daramouskas *et al.* implemented a Simple RVO in C++ with 25 drones and a static obstacle, with each UAV exchanging state information and resolving collisions locally.

1.5 Key Steps of the VO Algorithm

1. **Preferred-velocity calculation:** Compute $\mathbf{v}_{\text{pref}}$ toward the goal (magnitude = cruise speed).
2. **Obstacle sensing/communication:** Acquire positions and velocities of nearby UAVs and obstacles.
3. **VO construction:** For each obstacle j, compute the velocity obstacle cone $\text{VO}_{\text{A}|j} = (\text{collision cone at relative pos}_j) \oplus \mathbf{v}_j$.
4. **Combine VOs:** Form the union of all $\text{VO}_{\text{A}|j}$ regions to get the total forbidden velocity set.
5. **Velocity selection:** Choose a new velocity $\mathbf{v}_A$ outside this forbidden set. Ideally pick the $\mathbf{v}$ closest to $\mathbf{v}_{\text{pref}}$ (e.g. by minimizing deviation). If necessary, apply small delays or adjustments to avoid locking.
6. **Update position:** Move UAV according to $\mathbf{v}_A$ for one time step and repeat.

This procedure runs iteratively in the 2D simulation. The VO cones (as in the figure above) steer each UAV away from collision paths by design. Importantly, because the UAVs operate on a discrete planar grid in simulation, the method only needs 2D geometry. The overall system has no central controller: each UAV independently performs the above steps using only local information. In summary, the VO method provides a reactive, decentralized collision-avoidance strategy where each UAV continuously forbids velocities that would lead to a future collision, thereby safely navigating among static and moving obstacles in the 2D simulation environment.

2 RRT with Velocity Obstacle Integration

We implement a decentralized collision avoidance framework for multiple UAVs in a 2D grid-based simulation by combining Rapidly-exploring Random Trees (RRT) with the Velocity Obstacle (VO) method.

2.1 RRT Planning

RRT is a sampling-based algorithm that incrementally builds a tree from a UAV's start position by randomly sampling the space and connecting new nodes through collision-free paths. It is suitable for navigating in cluttered environments due to its efficient space exploration.

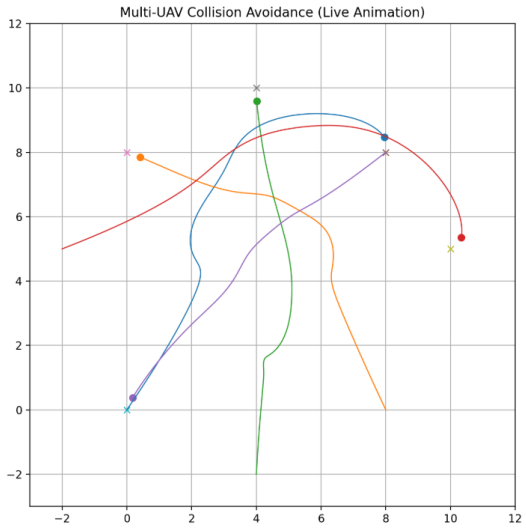


Figure 2: Velocity Obstacle Simulation

2.2 Velocity Obstacles

VO defines the set of velocities that would result in a collision with moving obstacles (other UAVs). A UAV avoids future collisions by selecting velocities outside these VO regions. This method is decentralized and assumes reciprocal collision avoidance.

2.3 Integration Approach

To handle dynamic obstacles, we modify the RRT extension step:

- When extending the tree, each UAV checks if the velocity from the current node to the sampled node lies outside all VO regions.
- Only extensions that are free of both static obstacles and potential dynamic collisions are accepted.
- The UAV replans its path at every timestep based on current observations of other agents' positions and velocities.

2.4 Decentralized Replanning

Each UAV independently builds an RRT at every step and selects a short-term feasible trajectory. This decentralized, reactive strategy ensures real-time collision-free navigation without central coordination.

2.5 Simulation Setting

This system is tested in a 2D simulation with static and dynamic obstacles. UAVs use simple point-mass models, and dynamic replanning ensures safety and coordination among agents.

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by ½ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow ¼ inch space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

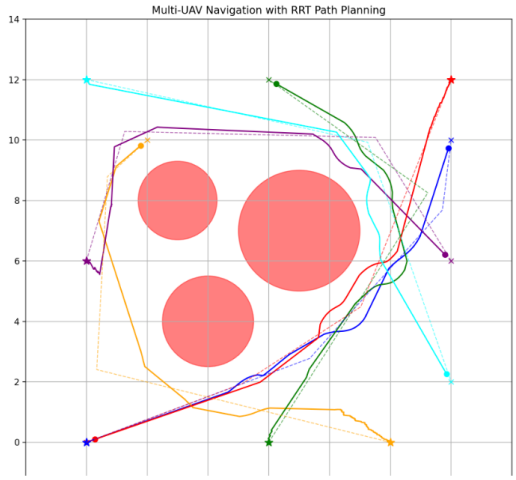


Figure 3: RRT Planning Path

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section **??** regarding figures, tables, acknowledgments, and references.

3 Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) is a model-free framework for imitation learning that draws a formal analogy to Generative Adversarial Networks. In GAIL, the policy π plays the role of the generator, producing trajectories of state–action pairs, while a discriminator D is trained to distinguish between those generated trajectories and expert demonstrations. Concretely, GAIL seeks a saddle point of the objective

$$\mathbb{E}_{(s,a)\sim\pi}[\log D(s,a)] \;+\; \mathbb{E}_{(s,a)\sim\pi_E}[\log(1-D(s,a))] \;-\; \lambda\, H(\pi),$$

where $H(\pi)$ is the causal entropy regularizer. The discriminator's feedback $-\log\big(1-D(s,a)\big)$ serves as a surrogate reward for the policy, pushing π to produce behaviors that the discriminator cannot distinguish from the expert's. By directly matching occupancy measures via adversarial training, GAIL avoids costly inverse-reinforcement-learning loops and hand-crafted reward design, and has been shown to achieve significant performance gains in high-dimensional control tasks.

3.1 Grid Setup

Our project focuses on teaching a team of multiple unmanned aerial vehicles (UAVs) to navigate a shared grid world from arbitrary start points to designated goals while avoiding both static obstacles and collisions with one another. Rather than hand-crafting a reward function to penalize every undesirable outcome, we leverage expert demonstrations generated by a customized, time-aware A* planner. This approach rooted in the principles of Generative Adversarial Imitation Learning frames expert trajectories as "real" data and policy roll-outs as "generated" data, allowing a discriminator network to distinguish between the two distributions and guide the policy toward expert-like behavior.

Our simulated environment is a discrete 10×10 grid populated with a cross-shaped obstacle at its center and three UAVs, each with a fixed goal position. At each timestep, the full joint state is encoded as a 300-dimensional vector: one channel for UAV locations, one for goal locations, and one for obstacles. Actions are simple moves—up, down, left, right, or stay—that are validated against grid

boundaries, obstacle occupancy, and potential UAV-UAV collisions before being applied. Moving yields a small positive reward, reaching a goal yields a substantial bonus, and any collision triggers a significant penalty, ensuring that expert planners and learned policies alike prioritize safe, efficient routes.

To produce expert data, we implemented a sequential, time-aware A* planner. For each UAV in turn, the planner computes a shortest path to the goal while marking its future positions in spacetime so that subsequent UAVs cannot occupy those cells at those times. Executing these locked-step plans in the simulator yields synchronized sequences of states and expert actions, capturing rich coordination patterns without collision.

The policy network employs a shared encoder a fully connected layer mapping the 300-dimensional observation into a 256-unit hidden representation followed by three separate actor "heads," one per UAV, each outputting a probability distribution over the five possible moves. In parallel, the discriminator network takes as input the concatenation of a state vector with a one-hot encoding of all three agents' chosen actions, passes them through its own 256-unit hidden layer, and outputs a single probability estimate of whether the sample came from the expert.

Training proceeds by alternating between policy updates where we compute adversarial rewards for each (state, action) pair via negative log-likelihood of the discriminator's output and optimize the policy to maximize these rewards and discriminator updates, which minimize the binary cross-entropy loss distinguishing expert from learner behavior. The result is a model-free, sample-efficient algorithm that quickly converges to trajectories mirroring the expert planner.

Finally, to demonstrate real-time coordination, we animate each UAV as a colored block moving across the grid at several frames per second. Obstacles appear as gray cells and goals as highlighted targets, allowing observers to see how UAVs detour around both static hazards and each other in pursuit of their individual objectives. This visual playback confirms that the learned policy not only avoids collisions but also reaches all goals in near-optimal time (Figure-4).

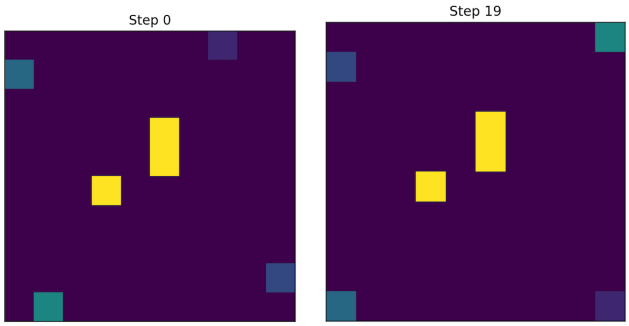


Figure 4: Step 0, all UAVs are at their starting position and Step 19 where all UAVs are at their destination while avoiding other UAVs and Static Obstacles colored yellow

4 GAIL-Based Multi-UAV Navigation in AirSim

4.1 Expert Demonstration via VO-Driven RRT

A specialized expert, DroneExpert, leverages Rapidly-exploring Random Trees (RRT) to compute globally feasible waypoints in a 3D environment with static obstacles, then refines each step with local repulsion based on the Velocity Obstacle (VO) concept to ensure dynamic collision avoidance with other UAVs. This VO-RRT expert is executed within Microsoft's AirSim simulator, producing state–action pairs (positions, velocities, and VO-filtered control commands) that capture both global path planning and reactive local avoidance.

4.2 Adversarial Imitation via GAIL

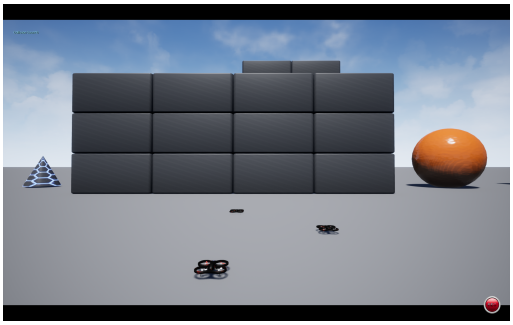
Generative Adversarial Imitation Learning (GAIL) trains a neural policy to mimic the VO-RRT expert without explicit reward engineering, by pitting a policy "generator" against a discriminator

that distinguishes expert demonstrations from policy rollouts. At each iteration, the policy interacts with the AirSim environment (wrapped in a Gym interface), and the discriminator assigns high imitation "rewards" when the policy's state–action distribution aligns with the expert's. The policy is updated via Proximal Policy Optimization (PPO) using these adversarial rewards, ensuring stable and sample-efficient convergence.

4.3 Policy Deployment in AirSim

Once trained, the distilled policy replaces the RRT+VO expert entirely: UAVs query the neural network for velocity commands in real time, executing directly in the high-fidelity AirSim world.

Because the policy has internallized both the global path intentions and local VO-based collision avoidance behaviors, deployment requires only onboard observation (positions, velocities, goals) and no further planning or expert queries.



(a) Initial position of UAVs in AirSim environment

(b) Final position of UAVs in AirSim environment

Figure 5: UAV positions at the beginning and end of the simulation

Collectively, this conceptual pipeline unifies classical sampling-based planning (RRT), reactive collision avoidance (VO), and modern adversarial imitation (GAIL+PPO) to yield a fully learned, decentralized multi-UAV navigation system in simulation.

5 Citations

1. Generative Adversarial Imitation Learning
2. Motion Planning in Dynamic Environments using Velocity Obstacles
3. RRT and Velocity Obstacles-based motion planning for Unmanned
4. RaCIL: Ray Tracing based Multi-UAV Obstacle Avoidance through Composite Imitation Learning Aircraft Systems Traffic Management (UTM)