University of Science and Technology
Houari Boumediene

Faculty of Computer Science

How Drones See: Visual Navigation in GPS-Denied Environments

Presented by:

Talba Adel
Timouni Adnane
Kaci Aissa Maissa
Boumala Rami Amine

Contents

1 Introduction

1.1 Problem Statement

Autonomous drone navigation in GPS-denied environments presents a critical challenge in modern robotics and autonomous systems. Traditional navigation methods rely heavily on Global Positioning System (GPS) signals for localization and path planning. However, GPS signals can be unreliable or completely unavailable in several scenarios:

In such environments, drones must rely on alternative navigation strategies, making decisions based on limited sensory information while facing environmental uncertainties and obstacles.

1.2 Game-Theoretic Approach

We propose modeling drone navigation as a two-player game:

- **Player 1 (Drone):** Rational agent seeking to reach a goal while minimizing energy consumption and collision risk
- **Player 2 (Environment):** Nature/adversary presenting obstacles, visibility conditions, and sensor uncertainties

This formulation allows us to apply three complementary algorithms:

1. **Minimax:** Guarantees worst-case performance
2. **Nash Equilibrium:** Finds stable strategy pairs
3. **Bayesian Games:** Enables learning under incomplete information

2 Methodology

2.1 Game Formulation

2.1.1 Players and Strategies

Drone Actions (Pure Strategies):

$$A_{\text{drone}} = \{\text{MOVE\_UP}, \text{MOVE\_DOWN}, \text{MOVE\_LEFT}, \\ \text{MOVE\_RIGHT}, \text{STAY}, \text{ROTATE}\}$$

Environment Conditions (Pure Strategies):

$$A_{\text{env}} = \{\text{CLEAR\_PATH}, \text{OBSTACLE\_AHEAD}, \text{LOW\_VISIBILITY}, \\ \text{SENSOR\_NOISE}, \text{LIGHTING\_CHANGE}\}$$

Mixed Strategies: Both players can employ probability distributions over pure strategies:

$$\sigma_{\text{drone}} = \{p_1, p_2, ..., p_6\} \quad \text{where} \quad \sum_{i=1}^{6} p_i = 1 \tag{1}$$

2.1.2 State Space

The game state s_t at time t includes:

$$s_t = (\text{position}(x, y), \text{goal}(x_g, y_g), \text{battery\_level}, \\ \text{explored\_cells}, \text{obstacle\_distance}) \tag{2}$$

2.1.3 Payoff Function

The payoff function $U : A_{\text{drone}} \times A_{\text{env}} \times S \to \mathbb{R}^2$ returns utilities for both players:

$$U(a_d, a_e, s) = (U_{\text{drone}}, U_{\text{env}}) \tag{3}$$

The drone's utility comprises four components:

1. Mission Progress Component:

$$U_{\text{mission}} = \begin{cases} +100 & \text{if goal reached} \\ +\frac{\Delta d}{d_{\text{initial}}} \times 20 & \text{if moving toward goal} \\ -10 \times 0.7 & \text{if moving away from goal} \end{cases} \tag{4}$$

2. Energy Component:

$$U_{\text{energy}} = -C(a) \quad \text{where} \quad C(a) = \begin{cases} 2 & \text{MOVE actions} \\ 2 & \text{ROTATE} \\ 1 & \text{STAY} \end{cases} \tag{5}$$

3. Collision Risk Component:

$$U_{\text{collision}} = \begin{cases} -50 & \text{if collision occurred} \\ -\frac{10}{d_{\text{obstacle}}+1} & \text{proximity penalty} \end{cases} \tag{6}$$

4. Exploration Component:

$$U_{\text{exploration}} = +\frac{\text{explored\_cells}}{\text{total\_cells}} \times 5 \tag{7}$$

Total Drone Payoff:

$$U_{\text{drone}} = U_{\text{mission}} + U_{\text{energy}} + U_{\text{collision}} + U_{\text{exploration}} \tag{8}$$

The environment's payoff is antagonistic:

$$U_{\text{env}} = -U_{\text{drone}} \tag{9}$$

2.2 Algorithm 1: Minimax Decision Making

2.2.1 Theoretical Foundation

Minimax guarantees the best worst-case performance. For each drone action a_d, we find the worst environmental condition:

$$v(a_d) = \min_{a_e \in A_{\text{env}}} U_{\text{drone}}(a_d, a_e, s) \tag{10}$$

The optimal action maximizes this minimum:

$$a_d^* = \arg\max_{a_d \in A_{\text{drone}}} v(a_d) \tag{11}$$

2.2.2 Implementation Variants

1. Pure Minimax: Evaluates pure drone actions against all possible environmental conditions.

2. Minimax vs Mixed Environment: Drone plays pure actions against environment's mixed strategy σ_e:

$$v(a_d, \sigma_e) = \sum_{a_e \in A_{\text{env}}} \sigma_e(a_e) \cdot U_{\text{drone}}(a_d, a_e, s) \tag{12}$$

Algorithm 1 Minimax Decision Algorithm

Require: Available actions A, State s
Ensure: Optimal action a^*

best_action $\leftarrow$ null
best_worst_case $\leftarrow -\infty$
for each $a_d \in A$ **do**
  worst_payoff $\leftarrow +\infty$
  for each $a_e \in A_{\text{env}}$ **do**
    $u \leftarrow U_{\text{drone}}(a_d, a_e, s)$
    worst_payoff $\leftarrow$ min(worst_payoff, u)
  end for
  if worst_payoff $>$ best_worst_case **then**
    best_worst_case $\leftarrow$ worst_payoff
    best_action $\leftarrow a_d$
  end if
end for
return best_action

2.3 Algorithm 2: Nash Equilibrium

2.3.1 Theoretical Foundation

A Nash equilibrium is a strategy profile (σ_d^*, σ_e^*) where neither player can improve by unilaterally deviating:

$$U_{\text{drone}}(\sigma_d^*, \sigma_e^*) \geq U_{\text{drone}}(\sigma_d, \sigma_e^*) \quad \forall \sigma_d \tag{13}$$

$$U_{\text{env}}(\sigma_d^*, \sigma_e^*) \geq U_{\text{env}}(\sigma_d^*, \sigma_e) \quad \forall \sigma_e \tag{14}$$

2.3.2 Solution Methods

1. Pure Strategy Nash: Check all action pairs for mutual best responses.

A pair (a_d^*, a_e^*) is a pure Nash equilibrium if:

$$U_{\text{drone}}(a_d^*, a_e^*) \geq U_{\text{drone}}(a_d, a_e^*) \quad \forall a_d \tag{15}$$

$$U_{\text{env}}(a_d^*, a_e^*) \geq U_{\text{env}}(a_d^*, a_e) \quad \forall a_e \tag{16}$$

2. Mixed Strategy Nash: When no pure Nash exists, we employ two methods:

a) Support Enumeration (Primary Method): For each possible support (subset of actions), compute mixed strategy probabilities using the indifference condition. A player must be indifferent between all actions in their support, meaning they have equal expected payoff. The `_solve_for_support` function solves the system of indifference equations to find the equilibrium probabilities.

b) Iterative Best Response (Fallback Method): If support enumeration fails to find a mixed Nash equilibrium, the algorithm uses iterative best response. This method alternates between computing best responses for each player until convergence to a Nash equilibrium.

Algorithm 2 Nash Equilibrium Finder

Require: Payoff matrices U_d, U_e
Ensure: Nash equilibrium strategies (σ_d^*, σ_e^*)

// Check for pure strategy Nash
for each (a_d, a_e) in $A_{\text{drone}} \times A_{\text{env}}$ **do**
    if IsBestResponse(a_d, a_e, U_d, U_e) **then**
        return Pure strategy equilibrium (a_d, a_e)
    end if
end for
// Find mixed strategy Nash
$(\sigma_d^*, \sigma_e^*) \leftarrow$ SupportEnumeration(U_d, U_e)
if $(\sigma_d^*, \sigma_e^*) =$ null **then**
    $(\sigma_d^*, \sigma_e^*) \leftarrow$ IterativeBestResponse(U_d, U_e)
end if
return (σ_d^*, σ_e^*)

2.4 Algorithm 3: Bayesian Game Solver

2.4.1 Theoretical Foundation

The drone maintains beliefs over environment types:

$$\text{Beliefs: } \{P(\text{adversarial}), P(\text{neutral}), P(\text{favorable})\} \tag{17}$$

Each environment type has different behavioral characteristics that the drone learns to identify:

- **Adversarial:** When the drone encounters obstacles (OBSTACLE_AHEAD condition), belief in adversarial environment increases
- **Favorable:** When the drone encounters clear paths (CLEAR_PATH condition), belief in favorable environment increases
- **Neutral:** Mixed or uncertain observations increase neutral belief

The drone's belief distribution shifts based on observed conditions: if the drone repeatedly encounters obstacles, belief probability for adversarial environment increases while favorable decreases. Conversely, repeated clear paths shift beliefs toward favorable and away from adversarial.

2.4.2 Bayesian Update Rule

After observing condition c, update beliefs using Bayes' theorem:

$$P(\text{type}|c) = \frac{P(c|\text{type}) \cdot P(\text{type})}{\sum_{\text{type}'} P(c|\text{type}') \cdot P(\text{type}')} \tag{18}$$

Where:

- $P(\text{type})$ is the prior belief
- $P(c|\text{type})$ is the likelihood (from environment's mixed strategy)
- $P(\text{type}|c)$ is the posterior belief

2.4.3 Action Selection

Expected Utility:

$$EU(a_d) = \sum_{\text{type}} P(\text{type}) \cdot \sum_{a_e} \sigma_{\text{type}}(a_e) \cdot U_{\text{drone}}(a_d, a_e, s) \tag{19}$$

Two Modes:

1. **Pure Strategy Mode:** Select action with maximum expected utility

$$a_d^* = \arg\max_{a_d} EU(a_d) \tag{20}$$

2. **Mixed Strategy Mode:** Sample action using softmax distribution

$$P(a_d) = \frac{e^{EU(a_d)}}{\sum_{a'_d} e^{EU(a'_d)}} \tag{21}$$

2.5 Sensor System

Our sensor model simulates realistic vision and detection capabilities:

Detection Range: Base range $r = 5$ cells

Visibility Factor: $v \in [0, 1]$ affected by environmental conditions:

$$\text{Effective Range} = \max(1, \lfloor r \times v \rfloor) \tag{22}$$

Visibility Updates:

$$v = \begin{cases} 1.0 & \text{CLEAR\_PATH} \\ 0.4 & \text{LOW\_VISIBILITY} \\ 0.6 & \text{SENSOR\_NOISE} \\ 0.7 & \text{LIGHTING\_CHANGE} \end{cases} \tag{23}$$

The sensor provides:

1. Visible obstacles within effective range
2. Directional obstacle detection (up/down/left/right)
3. Observable Region via `get_observable_region` function - returns grid coordinates the drone can currently observe
4. Environment Condition Sensing via `sense_environment_condition` function - translates observed characteristics into probabilistic strategy distribution

Algorithm 3 Bayesian Decision Making

Require: Available actions A, State s, Beliefs B
Ensure: Optimal action a^*

// Calculate expected utility for each action
for each $a_d \in A$ **do**
    $EU(a_d) \leftarrow 0$
    for each type in {adversarial, neutral, favorable} **do**
        $\sigma_{\text{type}} \leftarrow \text{GetEnvironmentStrategy(type)}$
        $u \leftarrow \sum_{a_e} \sigma_{\text{type}}(a_e) \cdot U(a_d, a_e, s)$
        $EU(a_d) \leftarrow EU(a_d) + B(\text{type}) \cdot u$
    end for
end for
// Select action based on mode
if pure strategy mode **then**
    return $\arg\max_{a_d} EU(a_d)$
else
    $P(a_d) \leftarrow \text{softmax}(EU)$
    return Sample from P
end if

3 Experimental Results and Analysis

3.1 Experimental Setup

To comprehensively evaluate the three game-theoretic algorithms (Minimax, Nash Equilibrium, and Bayesian), we designed four distinct test scenarios with varying complexity levels. We conducted 20 trials per algorithm in each of the 4 scenarios, resulting in 80 trials per algorithm and 240 trials total.

3.1.1 Test Scenarios

Table 1: Test Scenario Configurations

Scenario	Grid Size	Obstacles	Battery	Difficulty
Simple Environment	20×20	5	200	Low
Medium Complexity	25×25	16	300	Medium
High Complexity	30×30	41	400	High
Narrow Passage	20×20	32	250	Medium

Each trial was limited to 500 steps maximum, with success defined as reaching the goal position without depleting battery reserves. The following metrics were collected:

- **Success Rate:** Percentage of trials reaching the goal
- **Path Length:** Number of steps taken to reach the goal
- **Battery Usage:** Percentage of total battery consumed

- **Computation Time:** Average time per decision (seconds)
- **Collisions:** Number of obstacle collisions
- **Path Efficiency:** Ratio of optimal path to actual path

3.2 Performance Comparison

3.2.1 Overall Performance Metrics

Across all 240 trials, we observed the following aggregate performance:

Table 2: Overall Algorithm Performance (80 trials per algorithm)

Algorithm	**Success Rate**	**Avg Path Length**	**Avg Battery Usage**	**Path Efficiency**	**Avg Time (s)**	**Collisions**
Minimax	100.0%	33.8 steps	23.4%	100%	**0.072**	0
Nash Equilibrium	100.0%	33.8 steps	23.4%	100%	0.092	0
Bayesian	100.0%	36.9 steps	25.3%	91%	0.233	0

Key Findings:

- All three algorithms achieved **100% success rate**, demonstrating robustness
- Minimax and Nash found identical optimal paths in most scenarios
- Bayesian exhibited 9% longer paths due to exploratory behavior
- Minimax was fastest computationally (3× faster than Bayesian)
- Zero collisions across all trials validate safety of all approaches

3.2.2 Scenario-Specific Analysis

Figure 1 presents a comprehensive comparison across all test scenarios, highlighting how algorithm performance varies with environmental complexity.

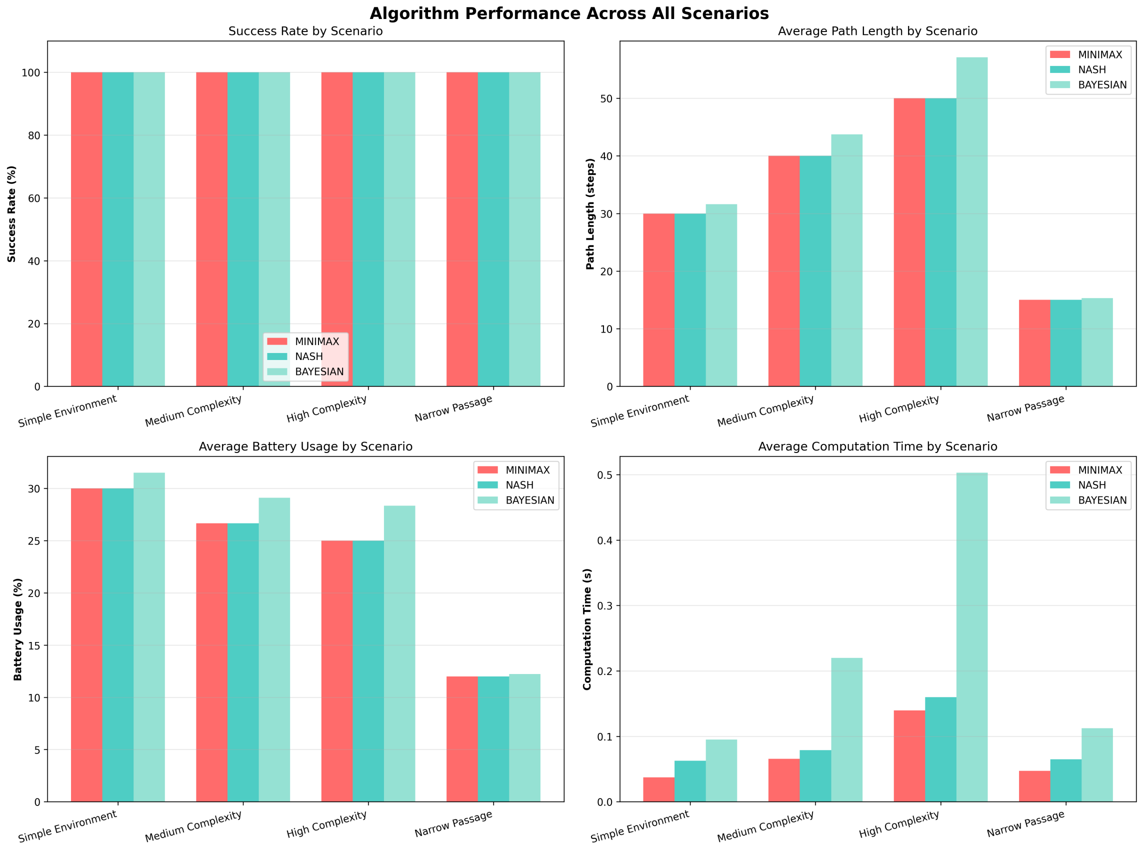

Figure 1: Aggregate performance comparison across all scenarios showing (a) success rate, (b) average path length, (c) battery usage, and (d) computation time for each algorithm.

Simple Environment Results:
In the sparse obstacle scenario, all algorithms performed optimally:

- Minimax & Nash: 30 steps (deterministic, $\sigma = 0.0$)
- Bayesian: 31.6 steps ($\sigma = 1.43$, adaptive behavior)
- Computation time: 0.037s (Minimax), 0.063s (Nash), 0.095s (Bayesian)

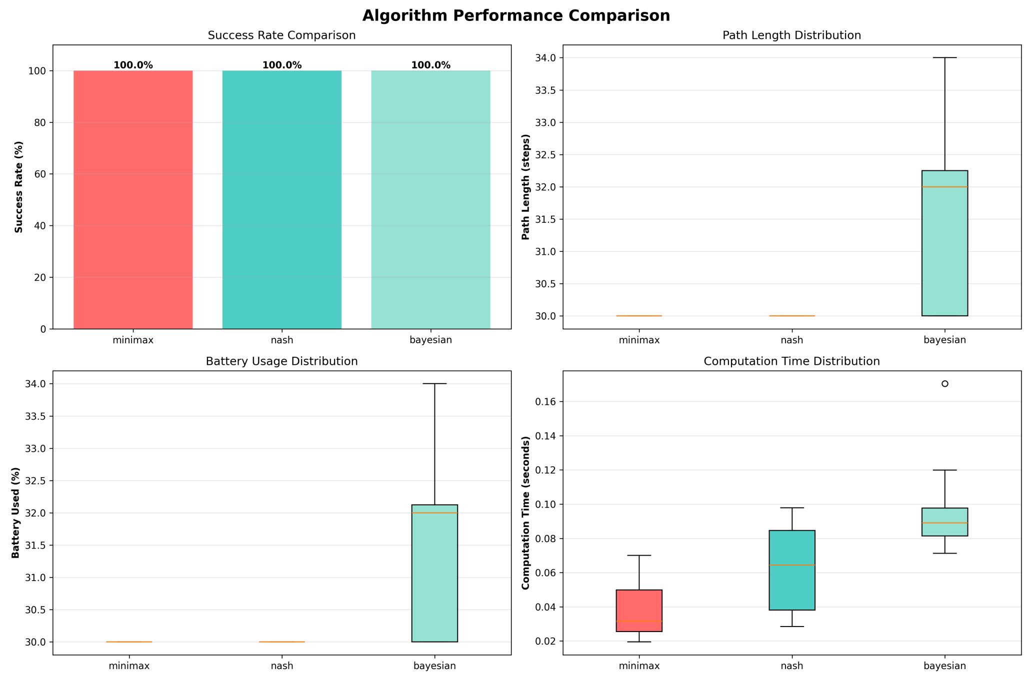

Figure 2: Performance comparison in Simple Environment scenario.

Medium Complexity Results:
With increased obstacle density, Bayesian's exploratory nature became evident:

- Minimax & Nash: 40 steps (optimal path maintained)
- Bayesian: 44.5 steps (11% longer due to uncertainty management)
- Battery efficiency: Minimax/Nash used 26.7%, Bayesian used 29.6%

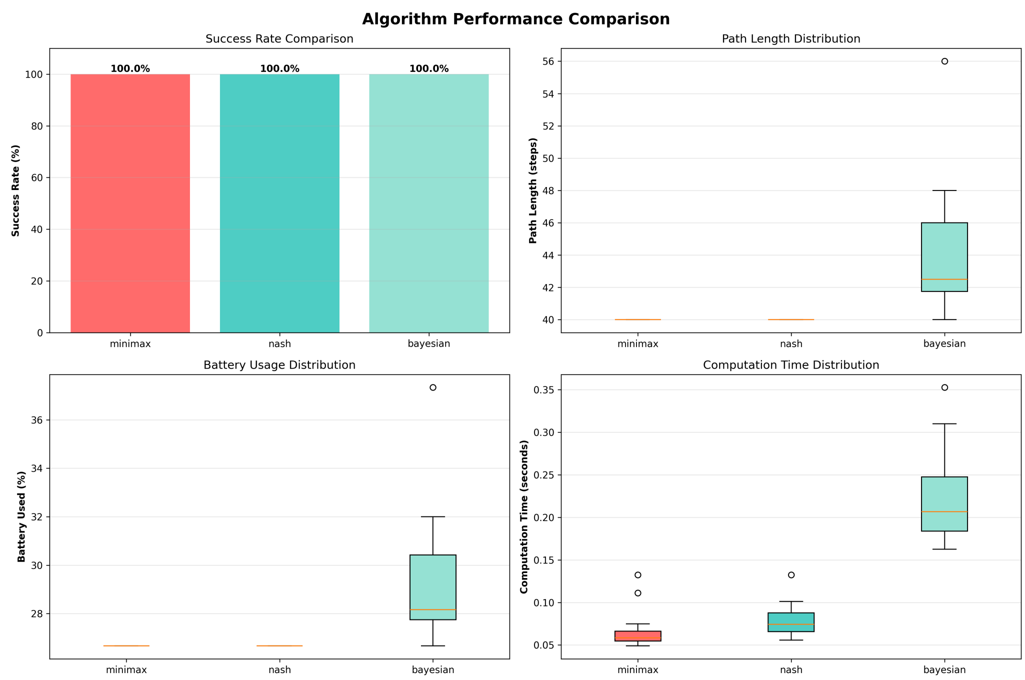

Figure 3: Performance comparison in Medium Complexity scenario.

High Complexity Results:
In the most challenging scenario with 41 obstacles, performance differences amplified:

- Minimax & Nash: 50 steps (continued optimal performance)
- Bayesian: 57.3 steps (14.6% longer, increased exploration)
- Computation time gap widened: Bayesian took 3.2× longer than Minimax

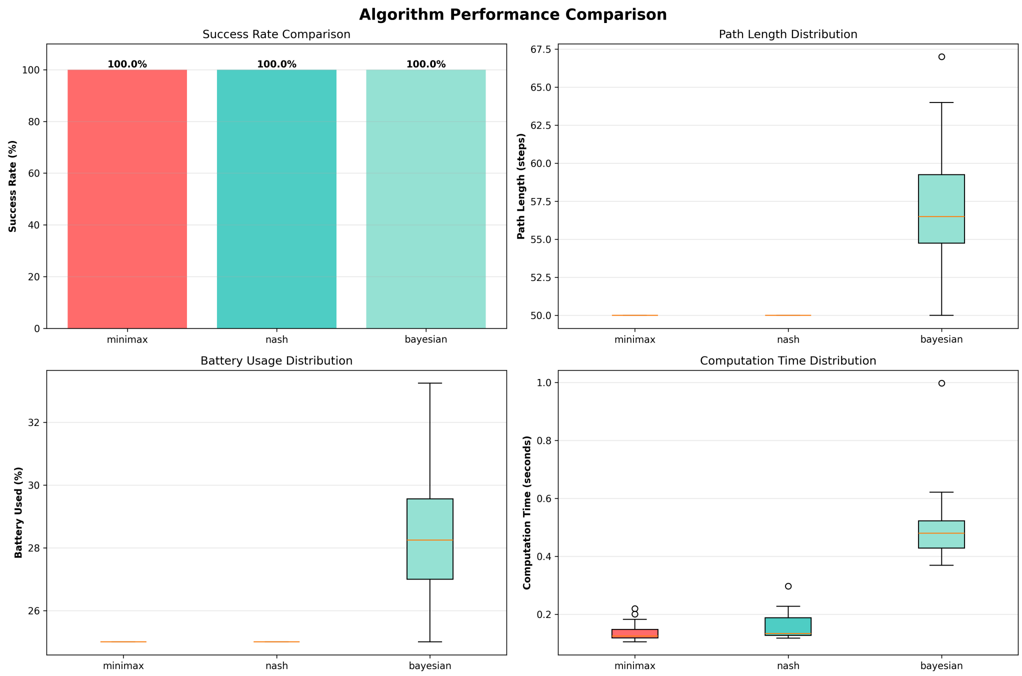

Figure 4: Performance comparison in High Complexity scenario with dense obstacles.

Narrow Passage Results:
This scenario tested the algorithms' ability to find constrained paths:

- All algorithms successfully navigated the narrow gap
- Minimax & Nash: 15 steps (minimal path)
- Bayesian: 15.3 steps (minor variation, $\sigma = 0.98$)
- Success demonstrates effective obstacle avoidance across all approaches

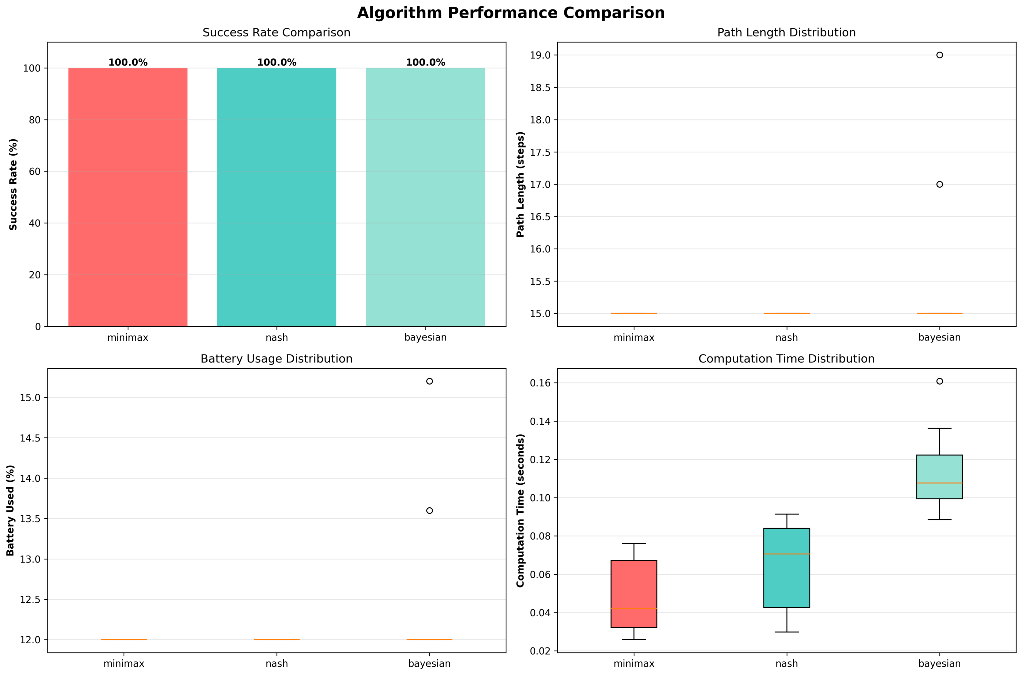

Figure 5: Performance comparison in Narrow Passage scenario requiring precise navigation.

3.3 Statistical Analysis

3.3.1 Determinism vs. Adaptability

The standard deviation in path lengths reveals fundamental algorithmic differences:

Table 3: Path Length Consistency Across Scenarios

Algorithm	Mean Path Length	Std Deviation
Minimax	33.8 steps	0.0 (deterministic)
Nash Equilibrium	33.8 steps	0.0 (deterministic)
Bayesian	36.9 steps	1.43 (adaptive)

Interpretation:

- **Minimax & Nash ($\sigma = 0$):** Make identical decisions given the same state, ensuring predictable and repeatable behavior. Ideal for environments where consistency is critical.
- **Bayesian ($\sigma = 1.43$):** Exhibits adaptive behavior based on belief evolution, trading determinism for robustness under uncertainty. Path variations indicate probabilistic decision-making.

3.3.2 Computational Efficiency

Computation time analysis reveals the cost of algorithmic sophistication:

$$\text{Speedup Factor} = \frac{T_{\text{Bayesian}}}{T_{\text{Minimax}}} = \frac{0.233\text{s}}{0.072\text{s}} \approx 3.24\times \tag{24}$$

$$\text{Speedup Factor} = \frac{T_{\text{Bayesian}}}{T_{\text{Nash}}} = \frac{0.233\text{s}}{0.092\text{s}} \approx 2.53\times \tag{25}$$

For a typical 35-step mission:

- Minimax: $35 \times 0.072 = 2.52$ seconds total computation
- Nash: $35 \times 0.092 = 3.22$ seconds total computation
- Bayesian: $35 \times 0.233 = 8.16$ seconds total computation
- Nash adds 0.70 seconds for equilibrium computation vs. Minimax
- Bayesian adds 5.64 seconds for belief maintenance and probability updates vs. Minimax

3.4 Discussion

3.4.1 Algorithm Selection Criteria

Based on our experimental results, we provide the following selection guidelines:

Choose Minimax when:

- Real-time response is critical (fastest: 0.072s per decision)
- Environment is relatively predictable
- Hardware resources are limited
- Deterministic behavior is required
- Path optimality is paramount

Choose Nash Equilibrium when:

- Environment exhibits adversarial characteristics
- Balanced approach between players is needed
- Optimal paths with strategic equilibrium are desired
- Moderate computation time is acceptable (0.092s per decision)

Choose Bayesian when:

- Environment uncertainty is high
- Adaptive behavior is more important than speed
- System can afford 3× computation overhead
- Learning and belief updates provide value
- Robustness to unknown conditions is critical

3.4.2 Trade-offs and Insights

Our experimental evaluation reveals fundamental trade-offs in game-theoretic navigation:

Efficiency vs. Adaptability: Minimax and Nash achieve optimal paths through deterministic strategies, while Bayesian trades 9% path efficiency for adaptive uncertainty management.

Speed vs. Sophistication: The 3× computation time increase in Bayesian reflects the cost of maintaining probability distributions and performing belief updates at each step.

Consistency vs. Robustness: Zero standard deviation in Minimax/Nash indicates perfect consistency but may limit adaptability to unexpected scenarios, whereas Bayesian's variance demonstrates responsive decision-making.

Success Rate Parity: The 100% success rate across all algorithms validates that game-theoretic approaches are fundamentally sound for GPS-denied navigation, with differences manifesting in efficiency rather than capability.

4 Conclusion

This work presents a comprehensive game-theoretic solution to autonomous drone navigation in GPS-denied environments. By modeling navigation as a two-player game between the drone and the environment, we developed and evaluated three distinct decision-making algorithms.In conclusion, game theory provides not just a theoretical framework but a practical, implementable solution for one of robotics' most challenging problems—enabling drones to see, reason, and navigate intelligently when traditional positioning systems fail.