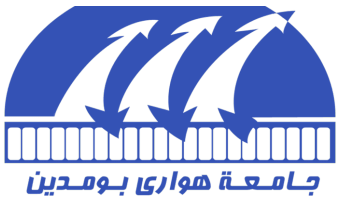

University of Science and Technology
Houari Boumediene

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 **2**
1.1 Problem Statement . . . 2
1.2 Game-Theoretic Approach . . . 2

2 Methodology **2**
2.1 Game Formulation . . . 2
2.1.1 Players and Strategies . . . 2
2.1.2 State Space . . . 3
2.1.3 Payoff Function . . . 3
2.2 Algorithm 1: Minimax Decision Making . . . 3
2.2.1 Theoretical Foundation . . . 3
2.2.2 Implementation Variants . . . 4
2.3 Algorithm 2: Nash Equilibrium . . . 4
2.3.1 Theoretical Foundation . . . 4
2.3.2 Solution Methods . . . 4
2.4 Algorithm 3: Bayesian Game Solver . . . 5
2.4.1 Theoretical Foundation . . . 5
2.4.2 Bayesian Update Rule . . . 5
2.4.3 Action Selection . . . 6
2.5 Sensor System . . . 6

3 Conclusion **7**

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 ← null
best_worst_case ← −∞
for each a_d ∈ A do
    worst_payoff ← +∞
    for each a_e ∈ A_env do
        u ← U_drone(a_d, a_e, s)
        worst_payoff ← min(worst_payoff, u)
    end for
    if worst_payoff > best_worst_case then
        best_worst_case ← worst_payoff
        best_action ← 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, find mixed strategies using support enumeration. For a given support (subset of actions), compute mixed strategy

probabilities using the indifference condition: a player is indifferent between all actions in the support, meaning they have equal expected payoff. This is implemented in the `_solve_for_support` function which solves the system of indifference equations.

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

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

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 ∈ A do
    EU(a_d) ← 0
    for each type in {adversarial, neutral, favorable} do
        σ_type ← GetEnvironmentStrategy(type)
        u ← Σ_{a_e} σ_type(a_e) · U(a_d, a_e, s)
        EU(a_d) ← EU(a_d) + B(type) · 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) ← softmax(EU)
    return Sample from P
end if
```

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

3 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.