Apprentissage par Renforcement pour le Contrôle d'un Drone

Inès Hafassa-Maïza, Hugo Laval

February 7, 2025

1 Introduction

Ce rapport décrit l'utilisation de l'apprentissage par renforcement pour contrôler un drone dans un environnement 3D avec des obstacles. Le but est de permettre au drone de naviguer de manière autonome vers une destination tout en évitant les obstacles.

2 Formulation du Problème

Le problème est formulé comme un processus de décision markovien (MDP) défini par l'ensemble (S, A, P, R, γ) où :

- S est l'ensemble des états, représentant la position, la vitesse et la batterie du drone.
- A est l'ensemble des actions, représentant les accélérations possibles du drone.
- P est la fonction de transition d'état, définissant la probabilité de passer d'un état à un autre après une action.
- R est la fonction de récompense, définissant la récompense reçue après chaque transition.
- γ est le facteur de discount, représentant l'importance des récompenses futures.

L'objectif est de trouver une politique $\pi(a|s)$ qui maximise la récompense cumulée attendue :

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

3 Modélisation de la Fonction de Récompense et de l'Environnement

La fonction de récompense est un élément crucial dans l'apprentissage par renforcement, car elle guide l'agent (le drone) vers un comportement optimal. Dans notre implémentation, la fonction de récompense est conçue pour encourager le drone à atteindre sa destination tout en évitant les obstacles. Si le drone atteint sa destination, une récompense élevée de 1000 est attribuée. En revanche, une pénalité sévère de -500 est appliquée en cas de collision avec un obstacle. De plus, pour chaque mouvement, une récompense proportionnelle à la réduction de la distance par rapport à la destination est calculée, incitant le drone à se rapprocher de son objectif. Une pénalité énergétique est également appliquée pour chaque action, proportionnelle à la racine de l'accélération utilisée, afin de simuler la consommation de la batterie.

L'environnement est modélisé comme un cube tridimensionnel de taille fixe (100 unités de côté), rempli d'obstacles générés aléatoirement avec une probabilité p pour chaque position dans le cube. La destination est également générée aléatoirement à chaque initialisation de l'environnement. Le drone perçoit son environnement à travers ses états internes, qui incluent sa position, sa vitesse et son niveau de batterie. Ces informations sont encapsulées dans un vecteur d'observation qui est utilisé par le modèle Actor-Critic pour prendre des décisions. Les obstacles sont perçus indirectement par le drone lorsqu'il vérifie les collisions après chaque mouvement. Cette modélisation permet de simuler un environnement réaliste et dynamique pour l'apprentissage du drone.

Les variables d'état du drone sont modélisées comme suit :

- $X_t, Y_t, Z_t \in [0, 100]$ représentent les coordonnées de la position du drone à l'instant t.
- $V_{x_t}, V_{y_t}, V_{z_t} \in \mathbb{R}$ représentent les composantes de la vitesse du drone à l'instant t.
- $B_t \in [0, 100]$ représente le niveau de batterie du drone à l'instant t.

Nous définissons également :

- $\mathbf{X}_{\text{cible}}$: le point de livraison.
- $\mathcal{O}$: l'ensemble des obstacles.

La fonction de coût C_t à l'instant t, à minimiser est définie comme suit :

$$C_t = -R_t = -1000 \cdot \mathbf{1}_{\{\mathbf{X}_t = \mathbf{X}_{\text{cible}}\}} + 500 \cdot \mathbf{1}_{\{\mathbf{X}_t \in \mathcal{O}\}} + 1000 \cdot \mathbf{1}_{\{B_t = 0\}} - \alpha \left\| \mathbf{X}_{t+1} - \mathbf{X}_t \right\|_2 + \beta \left\| \mathbf{A}_t \right\|_2$$

où α et β sont des coefficients de pondération pour la distance parcourue et la consommation d'énergie, respectivement, $\mathbf{X}_t = (X_t, Y_t, Z_t)$ représente la position du drone à l'instant t, et $\mathbf{A}_t = (Ax_t, Ay_t, Az_t)$ représente les composantes de l'accélération appliquée au drone à l'instant t.

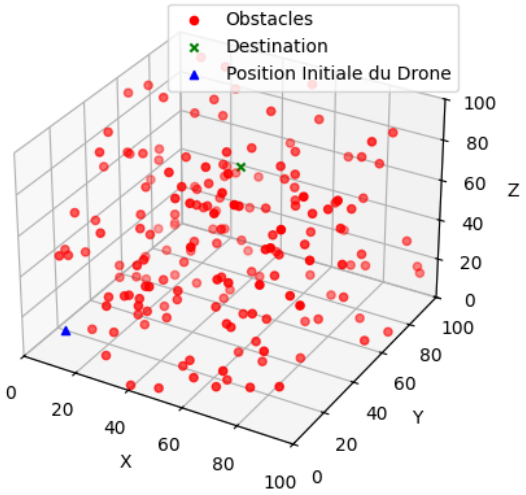

Figure 1: Représentation de l'environnement 3D avec les obstacles et la destination.

4 Modèle Actor-Critic

Le modèle Actor-Critic est utilisé pour résoudre ce problème. Il combine deux réseaux de neurones :

- L'actor, qui propose des actions basées sur les états actuels.
- Le critic, qui évalue la qualité des actions proposées en estimant la valeur des états.

4.1 Fonctionnement

Le réseau Actor-Critic fonctionne en deux étapes :

1. L'actor prend un état s et propose une action a en suivant une politique $\pi(a|s)$.
2. Le critic évalue cette action en calculant la valeur de l'état $V(s)$ et l'avantage $A(s, a)$.

5 Implémentation

Le fichier `app_renf.py` contient les classes et fonctions suivantes :

- `Environment` : Génère un environnement 3D avec des obstacles et une destination.
- `Drone` : Représente le drone avec ses propriétés et comportements.
- `DroneEnv` : Interface Gym pour l'apprentissage par renforcement.
- `ActorCritic` : Modèle Actor-Critic pour l'apprentissage.
- `train` : Fonction d'entraînement pour le modèle Actor-Critic.

6 Résultats et Discussion

L'entraînement du modèle Actor-Critic permet au drone d'apprendre à naviguer vers la destination tout en évitant les obstacles. Les récompenses sont définies pour encourager le drone à se rapprocher de la destination et à éviter les collisions.

7 Conclusion

L'apprentissage par renforcement avec le modèle Actor-Critic est une approche efficace pour le contrôle autonome de drones dans des environnements complexes. Les résultats montrent que le drone peut apprendre à naviguer de manière optimale en maximisant les récompenses cumulées.