Projet : Conduite autonome en trafic urbain dense par apprentissage par renforcement

Nabil El Malki

Année universitaire 2025–2026

Contexte

La conduite automobile exige une attention constante et une réactivité précise. Pourtant, une majorité des accidents de la route sont dus à des erreurs humaines. À titre d'exemple, en France, le dernier bilan officiel Observatoire national interministériel de la sécurité routière de 2022 indique que les facteurs humains sont impliqués dans 92% des accidents mortels, loin devant les causes liées à l'infrastructure, au véhicule ou aux conditions de circulation. Une étude réalisée en 2016 par la National Highway Transportation Safety Administration (NHTSA) a révélé que l'erreur humaine est responsable de 94% à 96% de tous les accidents de la route : manque de vigilance, distractions ou mauvaises prises de décision. L'usage croissant du smartphone en conduite aggrave encore ce constat.

Pour répondre à ces enjeux, l'industrie automobile développe des fonctionnalités intelligentes visant à améliorer la sécurité. Des systèmes tels que le freinage automatique ou l'évitement de collision s'appuient sur des capteurs embarqués pour anticiper les risques et réagir rapidement. Ces innovations constituent les premières étapes vers des véhicules autonomes, capables de conduire sans intervention humaine tout en assurant la sécurité des passagers et des autres usagers de la route.

Grâce aux avancées en apprentissage profond et du matériel de traitement parallèle, il est désormais possible de concevoir des agents intelligents capables d'analyser leur environnement, de prendre des décisions en temps réel et d'interagir de manière autonome avec leur environnement.

Cependant, la conduite autonome reste un défi de grande envergure, mobilisant plusieurs compétences clés : la reconnaissance, la prédiction et la planification. Les agents d'intelligence artificielle utilisent des capteurs et des modèles IA (par exemple les réseaux de neurones) pour identifier les éléments les entourant au sein de l'environnement, prédire les états futurs afin de prendre des décisions immédiates, puis intégrer ces informations pour planifier des stratégies de conduite sûres et efficaces, évitant les dangers et atteignant leur destination

L'apprentissage par renforcement, une approche IA, combiné à des architectures neuronales profondes (par exemple les réseaux convolutifs) permet d'entraîner des agents autonomes dans le but de réagir efficacement dans des environnements complexes, notamment à l'aide de simulateurs réalistes comme CARLA, qui reproduisent des situations de circulation urbaine, avec feux tricolores, panneaux et circulation dense.

Problématique

Un véhicule autonome doit non seulement éviter les accidents, mais aussi proposer une conduite fluide et efficace, dans des conditions de circulation dynamiques :

— suivre la route et rester dans la voie,

— adapter sa vitesse aux autres véhicules,

— éviter les collisions, y compris dans les situations de trafic dense ou de bouchons,

— éventuellement dépasser ou changer de voie pour réduire la durée du trajet.

Les travaux en conduite autonome se concentrent souvent sur la détection d'obstacles et le suivi de voie à partir des capteurs, mais la qualité de conduite (confort, efficacité, gestion de bouchons) est parfois négligée. Un agent autonome "intelligent" doit non seulement assurer la sécurité, mais aussi adopter des comportements *raisonnables* : profiter d'une seconde voie libre, éviter de rester bloqué derrière une file de véhicules très lents, gérer des conducteurs imprudents, etc.

Dans ce projet, nous nous focalisons sur un scénario central : **apprendre à une voiture à conduire en autonomie dans un environnement urbain dense**, incluant la présence d'un **bouchon** sur une portion de route, avec des autres véhicules au comportement éventuellement imprudent.

Les aspects feu rouge et panneau stop sont considérés comme des **bonus** facultatifs (si le temps le permet).

Objectif du projet

L'objectif est de concevoir et implémenter un **agent de conduite autonome** (un seul véhicule d'intérêt, appelé *voiture ego*) basé sur l'apprentissage par renforcement (RL), capable de :

— se déplacer de manière autonome sur des routes urbaines,

— adapter sa vitesse à la densité de trafic (faible, moyenne, forte),

— gérer un environnement comprenant un **bouchon** (embouteillage),

— de changer de voie pour gagner du temps,

— éventuellement respecter des feux rouges et panneaux stop (bonus).

La tâche est décomposée en plusieurs versions progressives :

V0 – Conduite autonome simple : conduire en ligne droite / suivre une route (tout droit, virages, route à gauche, etc.), avec **adaptation automatique de la vitesse** en fonction des véhicules devant, dans un environnement de faible à moyenne densité. Pas de changement de voie.

V1 – Changement de voie sur double voie : même objectif que V0, mais sur une route à **double voie dans le même sens**. L'agent peut décider de changer de voie (par exemple pour dépasser un véhicule lent) afin de réduire le temps de trajet.

VB1 (bonus) – Densité moyenne + conducteurs imprudents : extension de V1 dans un environnement où une partie des autres véhicules est **imprudente** (distance de sécurité réduite, variations de vitesse, respect imparfait des règles). L'agent doit rester sûr et efficace malgré ce comportement.

VB2 (bonus) – Feu rouge et stop : permettre à la voiture (V0, V1 ou VB1) de **détecter et respecter la signalisation** de type feu rouge et panneau stop, à partir de capteurs embarqués (caméra, etc.) : s'arrêter au bon endroit, repartir au feu vert, respecter l'arrêt au stop.

Méthodologie générale

Le projet s'appuie sur **l'apprentissage par renforcement**. L'agent interagit avec l'environnement CARLA en boucle :

- à chaque instant t, l'agent observe l'état (ou une observation partielle) via les capteurs (caméra, vitesse, distance aux autres véhicules, etc.) ;
- il choisit une **action** (accélérer, freiner, tourner, changer de voie, etc.) ;
- l'environnement renvoie un **état/observation** mis à jour et une **récompense** r_t ;
- l'agent ajuste sa politique pour maximiser la récompense cumulée.

Observations, actions et récompense

Pour chaque version (V0, V1, VB1, VB2), vous devrez définir :

- **les observations** issues des capteurs :
  - internes : vitesse actuelle, angle par rapport à la route, etc. ;
  - externes : distances aux autres véhicules (LIDAR, RADAR) et/ou images caméra frontale,
  - éventuellement informations dérivées de waypoints (orientation de la route, existence d'une voie à gauche/droite), à discuter.
- **les actions** :
  - accélérer / maintenir / freiner,
  - modifier l'angle de direction (gauche/droite),
  - pour V1 et VB1 : changer de voie (vers gauche/droite) ou rester dans la voie.
- **la fonction de récompense**, par exemple :
  - récompense positive pour l'avancement sans collision,
  - pénalités pour les collisions, les trajets très lents ou les arrêts inutiles,
  - bonus pour avoir dépassé un bouchon ou réduit la durée du trajet,
  - pour VB2 : pénalités pour passage au feu rouge / non-respect du stop.

Les choix d'observations, d'actions et de récompense devront être discutés avec l'enseignant, notamment pour V0 et V1.

Algorithme RL

Vous êtes libres de choisir l'algorithme d'apprentissage par renforcement, en justifiant vos choix et à condition 'en expliquer clairement le principe et la motivation.

Simulateur CARLA

Le projet s'appuiera sur le simulateur open-source **CARLA** (version 0.9.13 de préférence si ressources limités), qui fournit :

- des cartes urbaines (Town01, Town03, etc.) avec routes, intersections, signalisation,
- des capteurs embarqués : caméras RGB, LIDAR, GPS, IMU, radar, etc.,
- une API Python pour :
  - contrôler un véhicule ego (accélération, freinage, direction),
  - gérer des véhicules non contrôlés (NPC) via le *Traffic Manager* (densité, agressivité, respect des feux),
  - lire les données des capteurs.

Concepts clés de CARLA

- **Client** : programme Python qui se connecte au serveur CARLA via IP/port.
- **Monde** : état courant de la simulation (acteurs, météo, carte).
- **Acteurs** : véhicules, piétons, capteurs, feux, panneaux, etc.
- **Blueprints** : modèles pour créer des acteurs avec des attributs configurables.
- **Waypoints** : points 3D orientés sur les voies, permettant de récupérer la structure de la route (orientation, carrefour, voies à gauche/droite).
- **Capteurs** : objets attachés à la voiture ego qui renvoient des données :
  - caméras (RGB, profondeur, segmentation),
  - capteurs de collision, détecteurs d'obstacle,
  - GNSS, IMU, radar, LIDAR, etc.

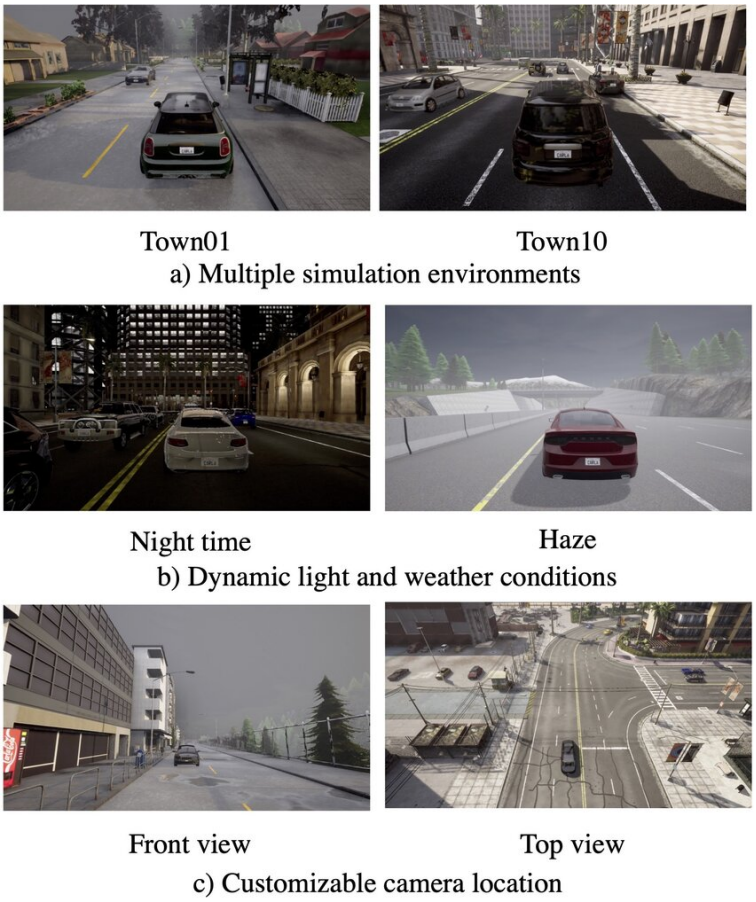

FIGURE 1 – Environnements différents de CARLA

Densité de trafic et imprudence des véhicules

Pour simuler le **trafic dense** et le **bouchon**, vous utiliserez :

- le **Traffic Manager** pour :
  - générer un grand nombre de véhicules NPC,

  - régler la densité (nombre de véhicules) sur la carte,
  - contrôler la vitesse moyenne (voitures plus lentes),
  - ajuster la distance de sécurité.
- pour VB1, certains véhicules pourront être paramétrés comme **imprudents** :
  - distance très faible au véhicule de devant,
  - légère augmentation de vitesse par rapport à la limite,
  - éventuellement comportement plus agressif.

Visualisation (Pygame)

Une fois CARLA actif (serveur), vous pouvez utiliser une **visualisation légère** basée sur `pygame` dans votre client Python :

- récupérer les images d'une caméra attachée à la voiture ego,
- les afficher dans une fenêtre `pygame` en temps réel,
- visualiser la conduite de l'agent sans ouvrir l'interface complète d'Unreal Engine.

Cela permet de déboguer et de démontrer le comportement de l'agent de manière plus simple et portable.

Notes – Contraintes RL et capteurs

Lors de la phase d'apprentissage et lors des tests, l'agent ne doit se baser que sur les **données issues des capteurs embarqués** (et éventuellement sur des informations dérivées de la structure de la route via les waypoints, si cela est discuté avec l'enseignant). Les informations dites "omniscientes" (comme la liste complète de tous les véhicules ou la position exacte des feux) peuvent être utilisées pour le calcul de la **récompense**, mais pas directement comme observations de l'agent.

Vous êtes libres de :

- choisir les capteurs utilisés (caméra seule, caméra + LIDAR, etc.),
- concevoir la fonction de récompense en fonction des objectifs (sécurité, temps de trajet, confort),
- choisir l'algorithme RL, en expliquant les raisons de vos choix.

Versions fonctionnelles

Le développement doit être progressif :

V0 – Conduite autonome simple

- Environnement avec faible ou moyenne densité de véhicules.
- Voiture ego conduite par un agent RL :
  - suit la route,
  - adapte sa vitesse pour éviter les collisions,
  - ne change pas de voie.
- Définir et implémenter :

— un vecteur d'observation (vitesse, distance devant, angle par rapport à la route, etc.),
— un ensemble d'actions (accélérer, freiner, tourner avec un certain angle etc.),
— une fonction de récompense.
— entraînement de l'agent RL

V1 – Changement de voie sur double voie

— Même environnement, mais avec une route à **double voie dans le même sens**.
— L'agent peut décider de :
  — rester dans sa voie,
  — changer de voie (vers la gauche ou la droite si disponible).

VB1 (bonus) – Densité moyenne + imprudence

— Même principe que V1, mais :
  — densité de trafic moyenne à forte,
  — certains véhicules NPC ont un comportement agressif / imprudent.
— L'agent doit se montrer **robuste** face à ces comportements (éviter les collisions, adapter sa vitesse, changer de voie au bon moment).

VB2 (bonus) – Feu rouge / stop

— Ajouter la gestion de la signalisation :
  — détection de feux rouges et panneaux stop via caméra (ou autre capteur),
  — décision de s'arrêter ou de continuer,
  — respect du code (pénalités fortes pour passage au rouge ou non-respect du stop).
— Possibilité de traiter ce bonus en fin de projet, si V0 et V1 fonctionnent.

Organisation du projet

Le projet se déroule sur **4 jours**, dont le dernier demi-jour consacré à la soutenance. Chaque groupe (3–4 étudiants) travaille sur l'ensemble des versions (V0, V1, bonus éventuels).

Plan de travail (4 jours)

Jour	Durée	Tâches principales	Livrables / objectifs
Jour 1	7h	Prise en main de CARLA, lancement du simulateur, découverte des capteurs (caméra, capteurs internes), contrôle d'un véhicule, réglage de la densité de trafic (bouchon simple)	Scripts de base (contrôle manuel / autopilot, génération de trafic, visualisation Pygame éventuelle)

Jour 2	7h	Définition des observations, actions et fonction de récompense pour V0. Implémentation et entraînement de l'agent RL. Discussion avec l'enseignant des choix de modélisation (en particulier la récompense).	Code poussé sur le dépôt, notes sur la fonction de récompense
Jour 3	7h	Extension vers V1 : ajout du changement de voie. Re-définition/ou ajustement des observations, actions et récompense. Tests et débogage.	Agent V1 fonctionnel dans un scénario simple, démonstration interne, code et configuration documentés
Jour 4	3,5h (matin)	Finalisation des fonctionnalités et Préparation de la soutenance (diapositives, scénario de démonstration, synthèse des résultats).	Scénario de démonstration prêt, dépôt Git à jour, diaporama de soutenance
Jour 4	3,5h (après-midi)	Soutenance (présentation + démonstration) et questions avec le jury.	Présentation orale, démonstration, évaluation finale

Structure du projet

Une structure de projet recommandée (par exemple avec `uv`) :

- `voiture_autonome/`
  - `rl_agents/` // Implémentations RL (V0, V1, bonus)
  - `envs/` // Environnements CARLA (scénarios, bouchon, densité)
  - `visualisation/` // Scripts Pygame, outils de debug
  - `utils/` // Fonctions utilitaires (capteurs, logging, etc.)
- `config/` // Paramètres d'entraînement, scénarios
- `tests/` // Tests simples
- `README.md` // Instructions de lancement

Rendus et soutenance

Au fil des jours, des livrables intermédiaires sont attendus :

- scripts de contrôle du véhicule et de génération de trafic,
- implémentation des agents RL (V0, V1, bonus éventuels),
- configuration des scénarios de test (densité, bouchon, imprudence),
- support de présentation pour la soutenance.

La soutenance finale (environ 25 minutes de présentation + 15–20 minutes de questions) devra inclure :

- une présentation du problème et des hypothèses,
- la description des observations, actions et récompenses,

— les choix d'algorithmes RL,
— une démonstration de la conduite autonome (V0 et V1 au minimum),
— une discussion sur les résultats et les limites.

L'évaluation portera sur :

— Qualité du code et fonctionnement de l'agent : **55 %**
— Qualité du rapport / documentation technique : **10 %**
— Qualité de la soutenance orale et de la démonstration : **35 %**