Optimisation de la trajectoire sur un circuit de sport automobile

Aloïs DENOISEUX

Épreuve de TIPE

2023-2024

Introduction

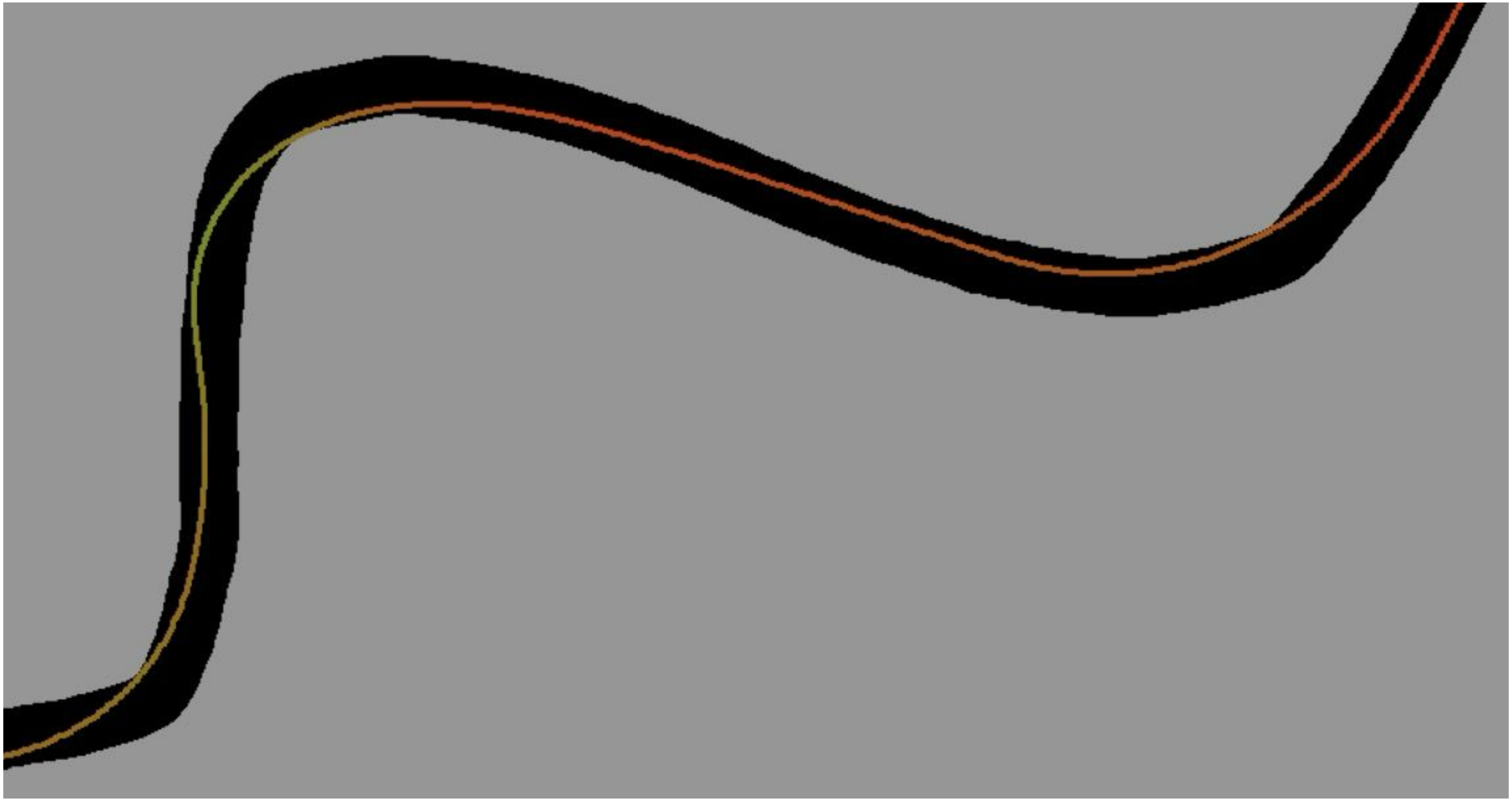
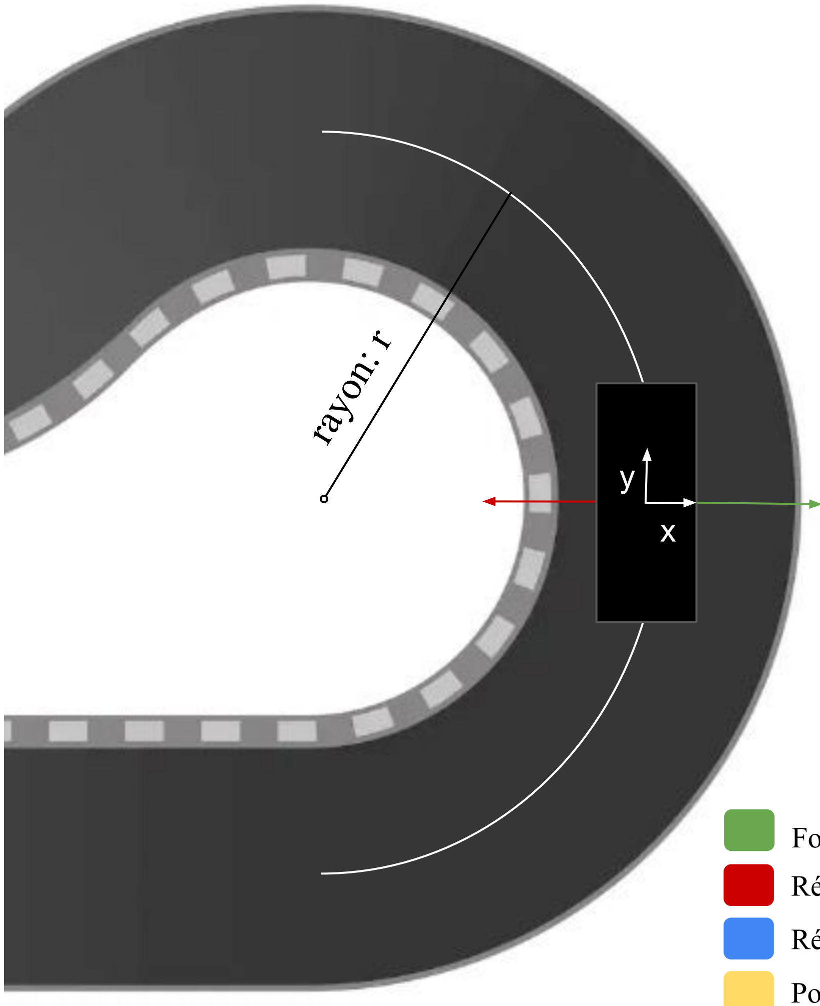


Illustration: Un exemple de trajectoire[4]

1. Introduction
2. Théorie
3. Modélisation
4. Résultats

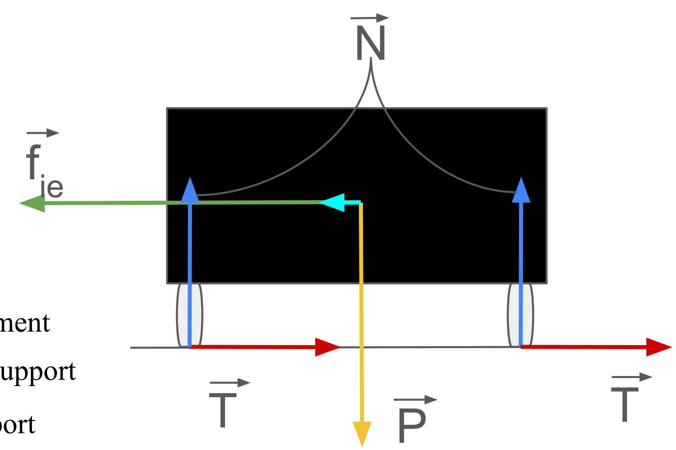
Théorie - Modèle choisi



Vu de haut

Hypothèses:

- Frottements pneu-sol selon e_y négligeables
- R_T supposé galiléen
- Trajectoire localement circulaire uniforme
- Vitesse constante pendant dt

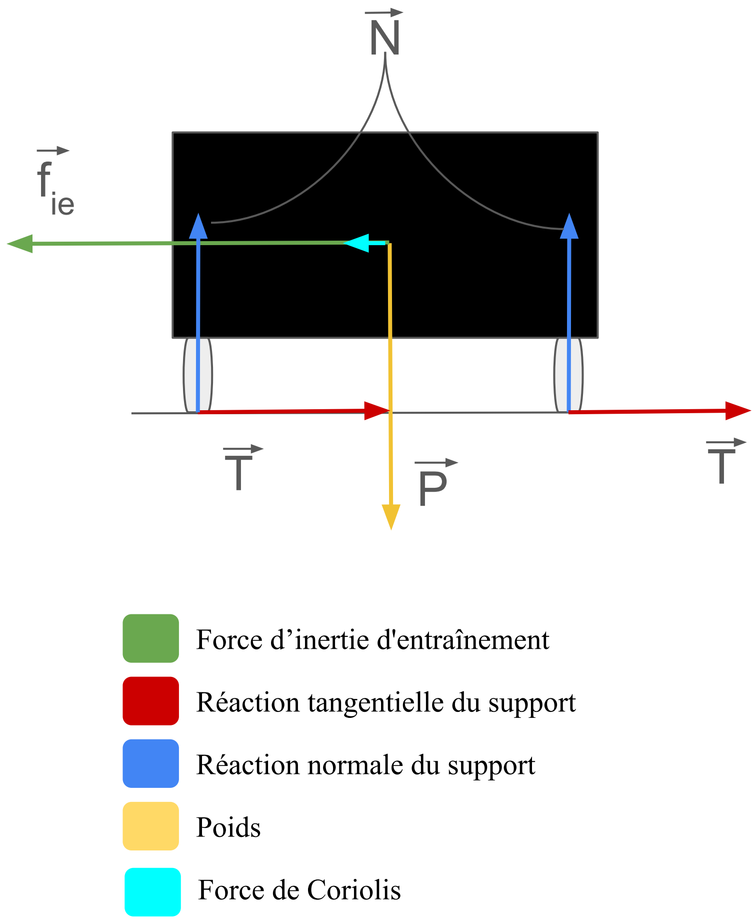


Vu de face

- Force d'inertie d'entraînement
- Réaction tangentielle du support
- Réaction normale du support
- Poids
- Force de Coriolis

On applique un PFD sur {voiture} dans $R_{voiture}$ non galiléen, en rotation uniforme autour de R_T galiléen

$$m \cdot \vec{a} = \vec{T} + \vec{f}_{ie} + \vec{P} + \vec{N} + \vec{f}_{ic}$$



Et, en utilisant la Loi de Coulomb dans le cas limite d'adhérence:

$$||\vec{N}|| = f \cdot ||\vec{T}||$$

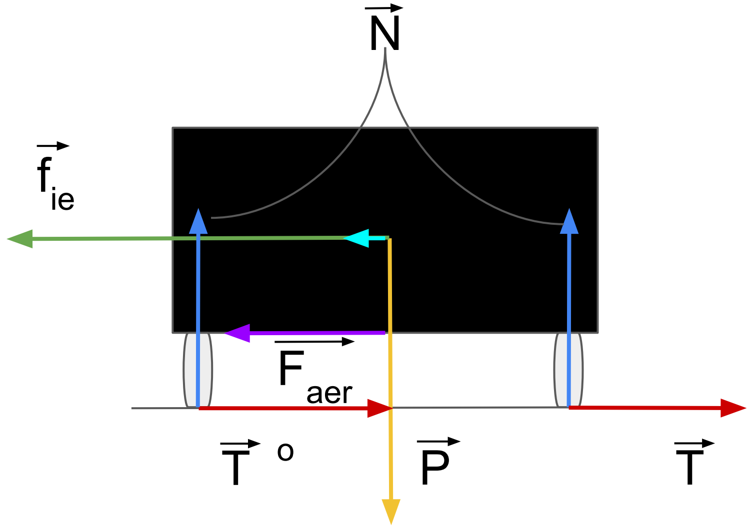
On obtient en projetant:

$$v_{\max} = \sqrt{f_d \cdot g \cdot r}$$

Force supplémentaire liée au profil de la voiture

$$F_{\text{aero-max}} \approx 3 \times 10^4\,\mathrm{N}$$

$$F_{\text{aero}} = m_{\text{sup}} \times g$$



En reprenant nos calculs:

$$v_{\max} = \sqrt{\left(\frac{m_{\text{sup}}+m}{m}\right) \cdot g \cdot f_d \cdot r}$$

- Force d’inertie d'entraînement
- Réaction tangentielle du support
- Réaction normale du support
- Poids
- Force de Coriolis
- Force aérodynamique

“DRS”: Système de réduction de traînée, +1.5s

“KERS”: Système de récupération d’énergie au freinage, +0.6s

Différents types de pneus (Soft, Medium, Hard, Intermediate, Wet)

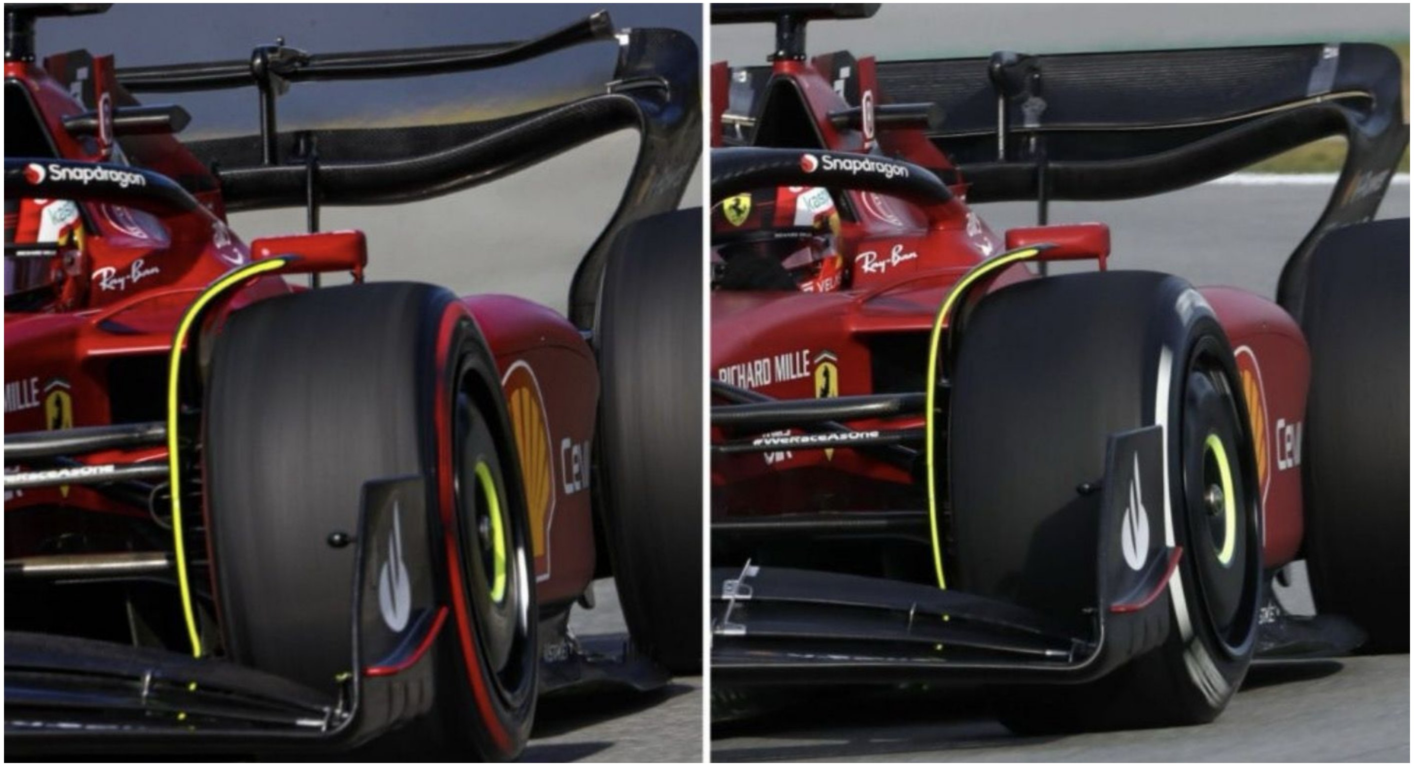


Illustration: Prashant Bhadauria[12]

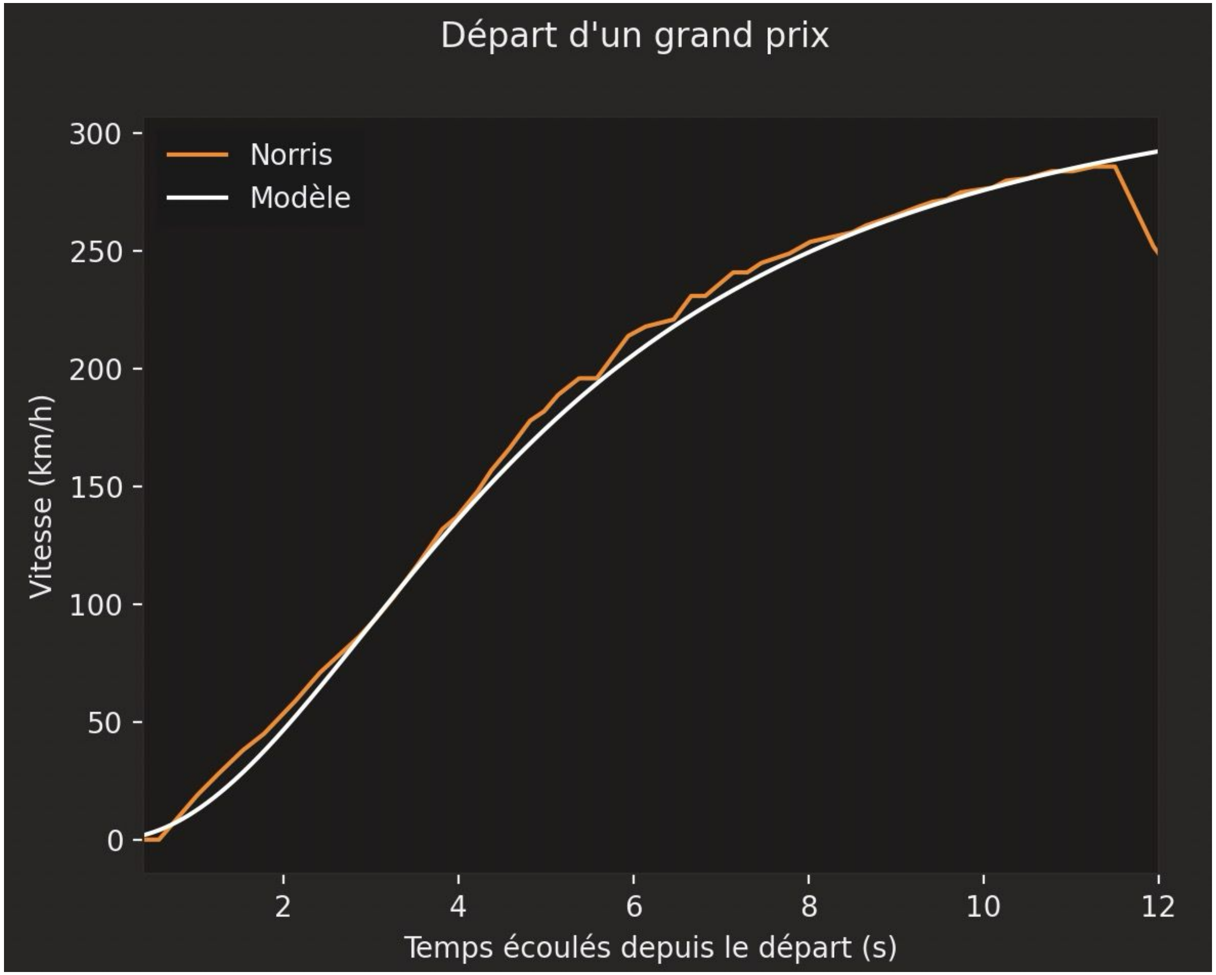
Objectifs:

- Modélisation suffisamment réaliste pour obtenir des trajectoires précises
- Cohérence du modèle avec la théorie précédente (~2s de retard par tour, vitesse aux virages cohérente)

Limites:

- Configuration fixe entre les courses, valide pour une unique écurie (McLaren F1 Team)
- Contrôles restreints à: accélération, freins et direction.

Modélisation - Paramètres physiques



On effectue une régression pour modéliser la vitesse:

- Courbe sigmoïde symétrique
- Moyenne des courbes pour tous les circuits

$$f(t) = a + \frac{d-a}{1+\left(\frac{t}{b}\right)^c}$$

$$a = 336, \quad b = 4.81, \quad d = 0, \quad c = 2.08632$$

Modélisation - Réseau de neurones

- Réseau de neurones entraîné par évolution génétique, similaire à l'évolution humaine (apprentissage par renforcement)

- 26 paramètres d'entrée, 2 de sortie

- 78 neurones, soit ~2100 poids ajustables

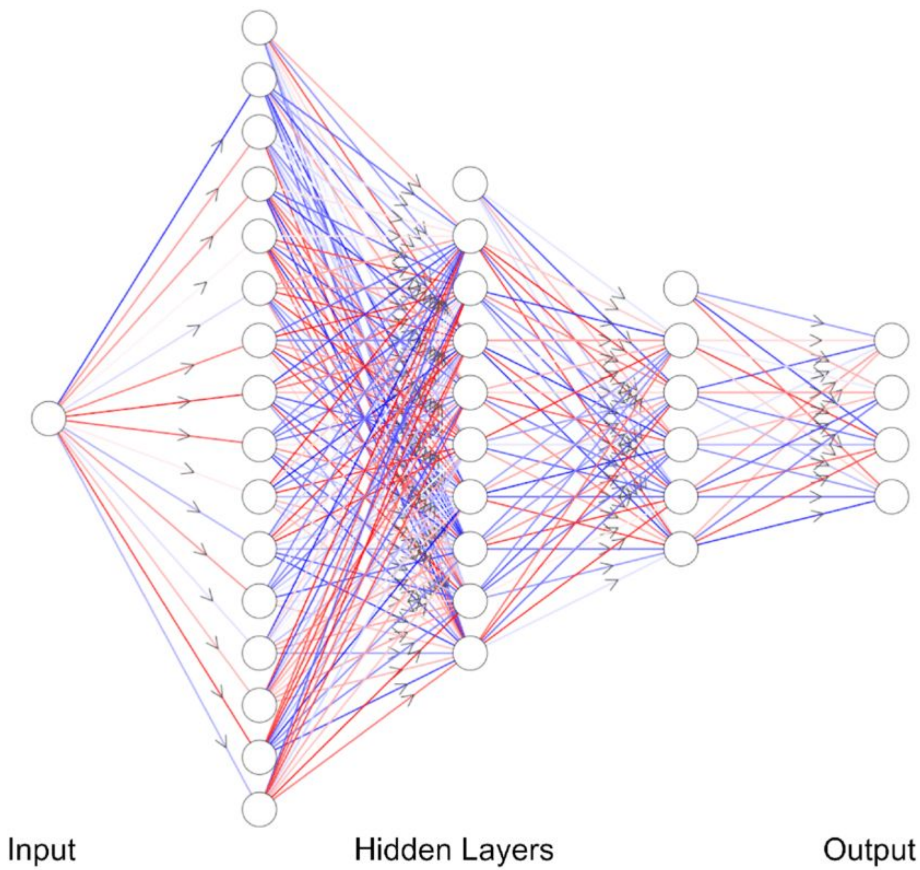
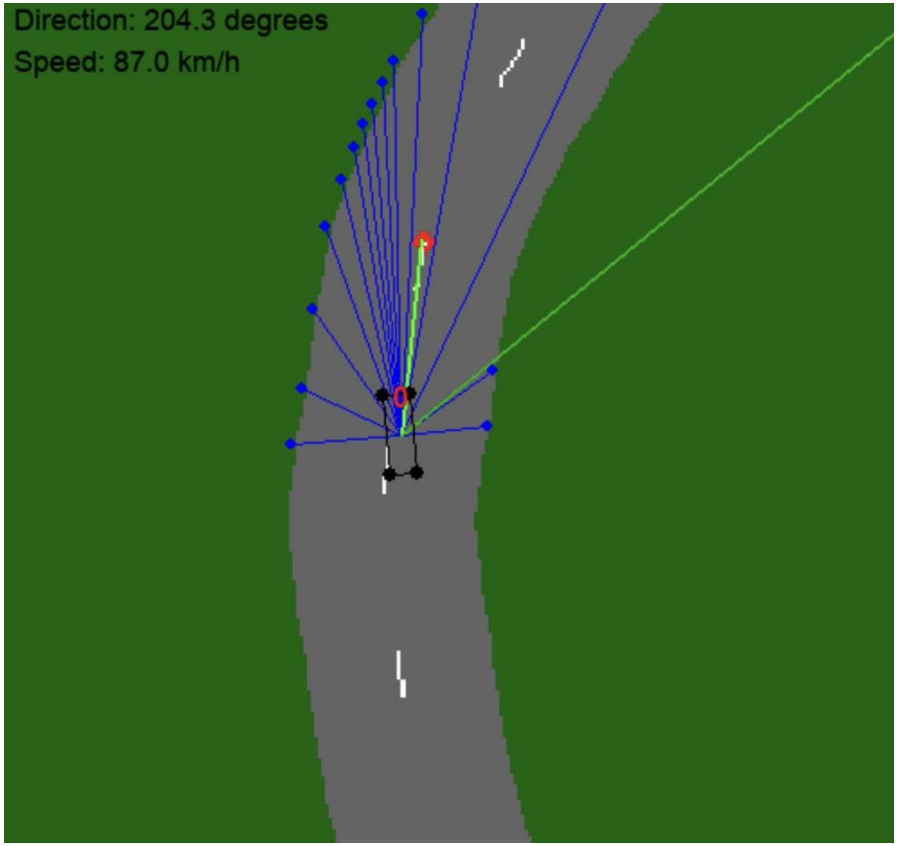
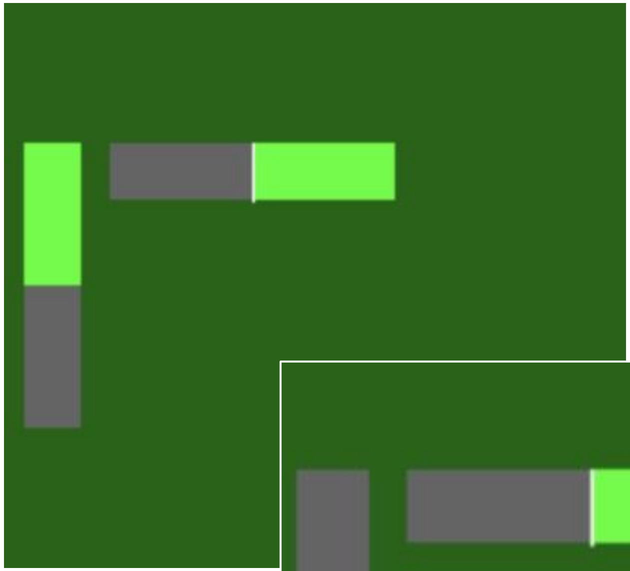


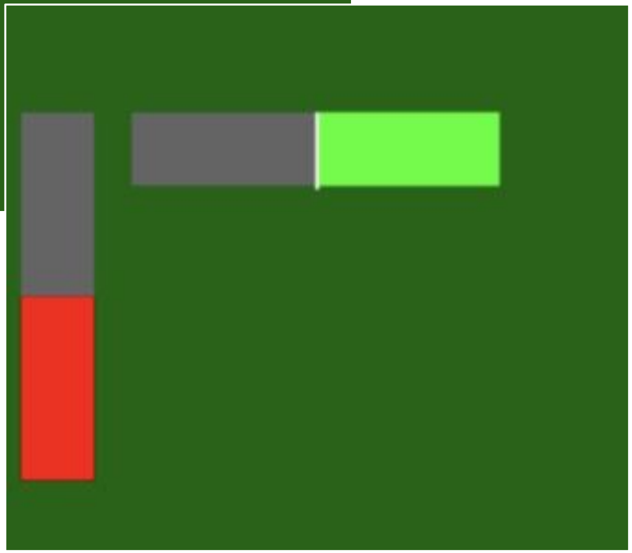
Illustration: Réseau de neurone [5]

Entrées

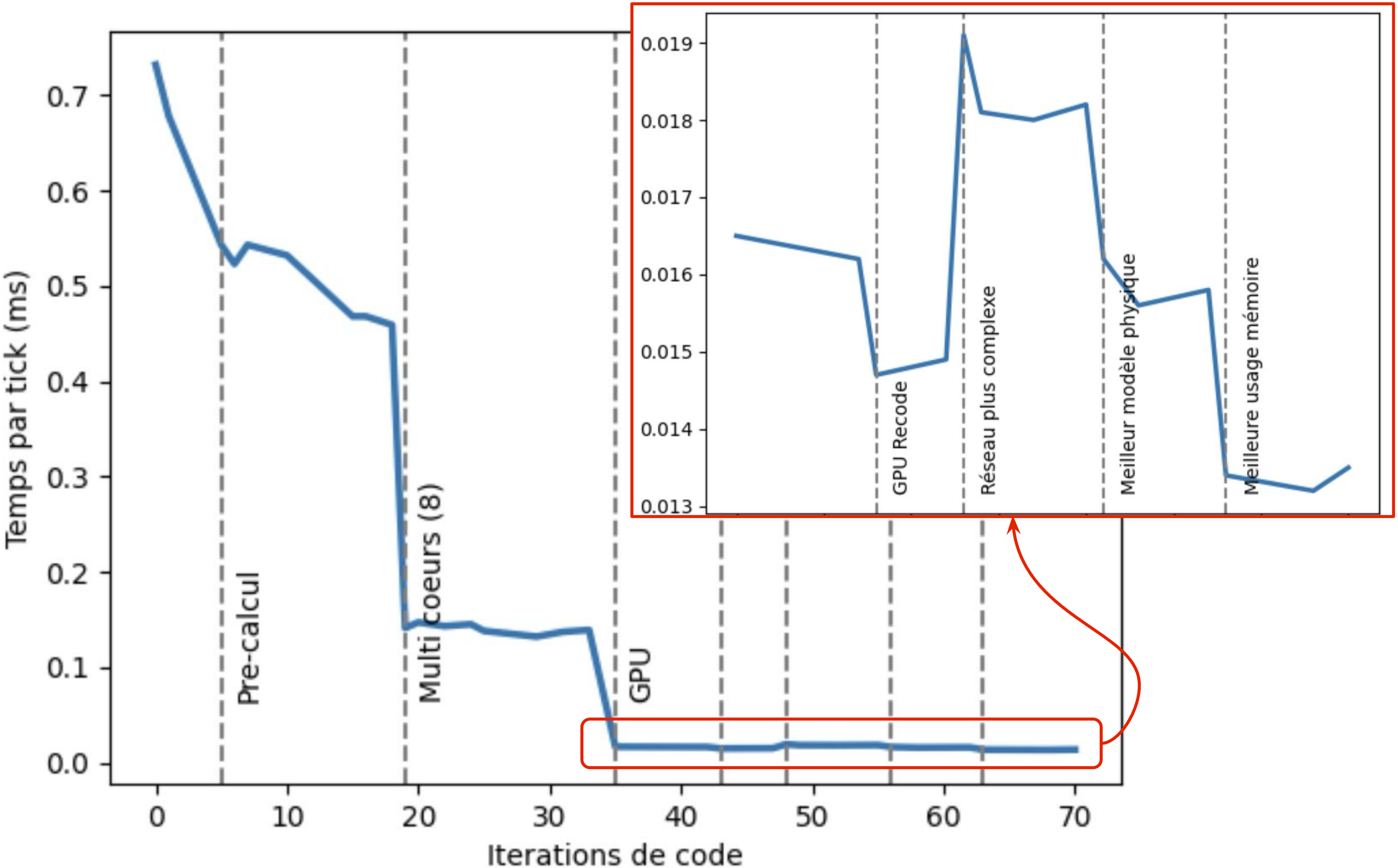


Sorties



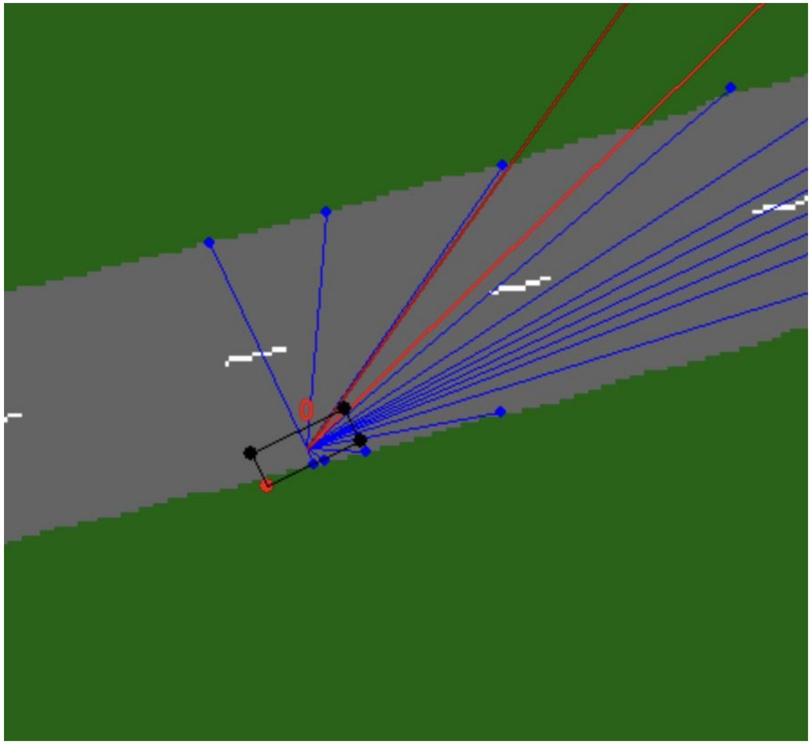


Modélisation - Performance



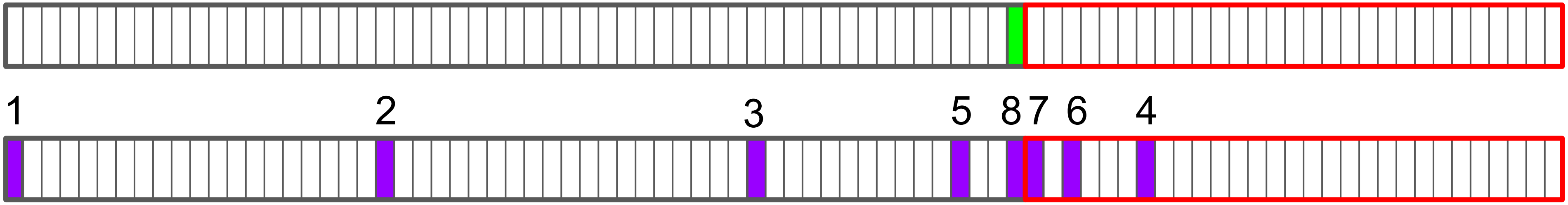
GPU - Graphics Processing Unit - Carte Graphique

- Extrêmement parallélisable
- Point faible: conditions de courses
- Rend le code ~10x plus rapide



Case recherchée

Configuration initiale du circuit

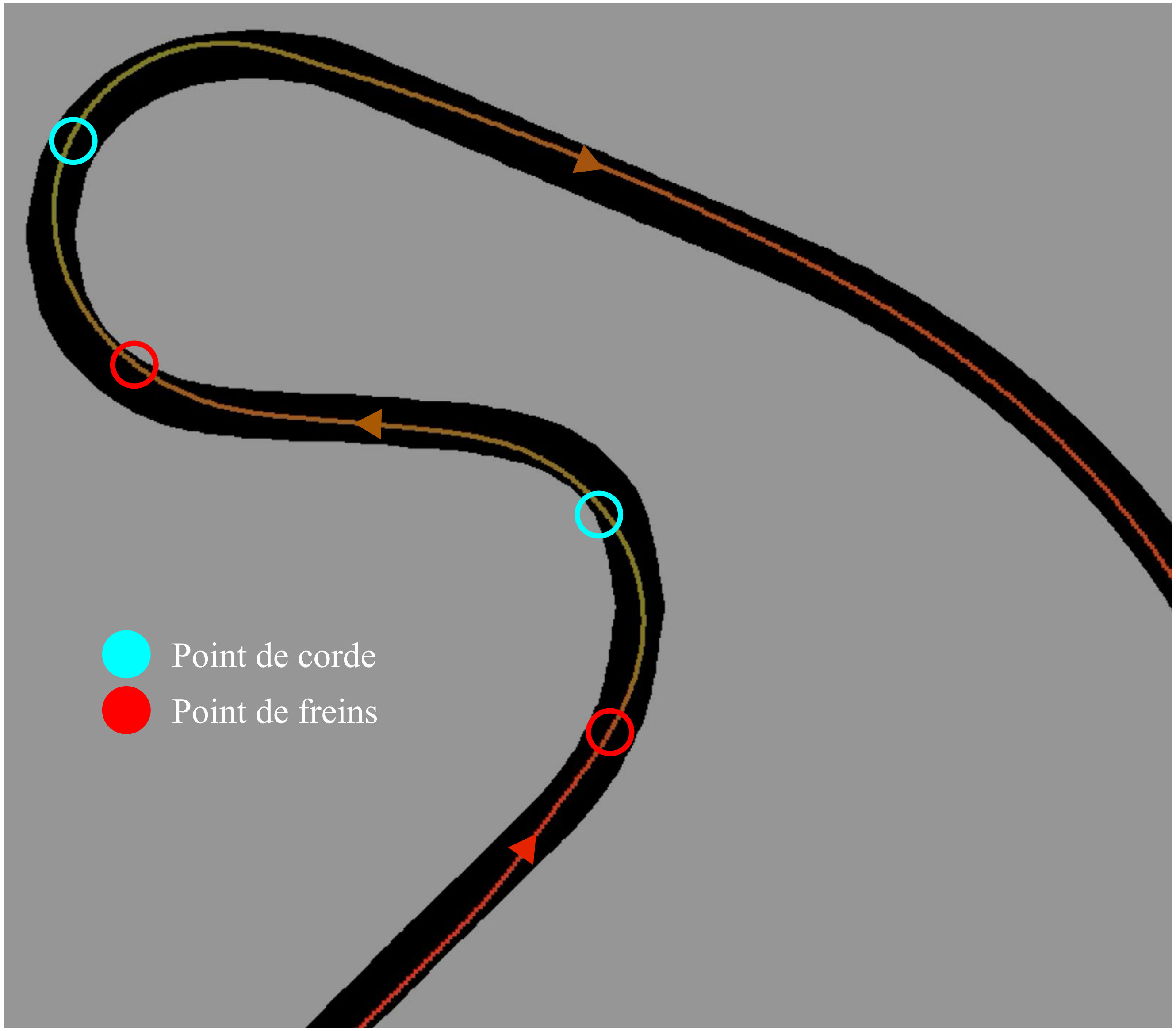


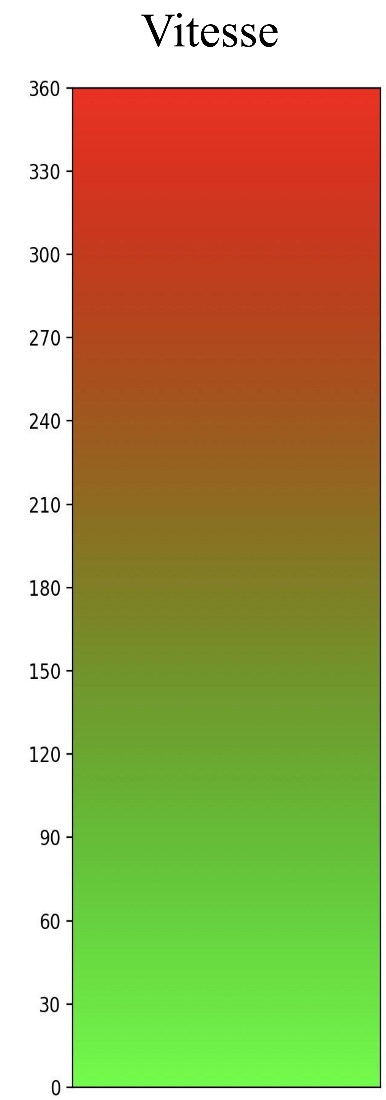
Cases explorées

Exploration du GPU

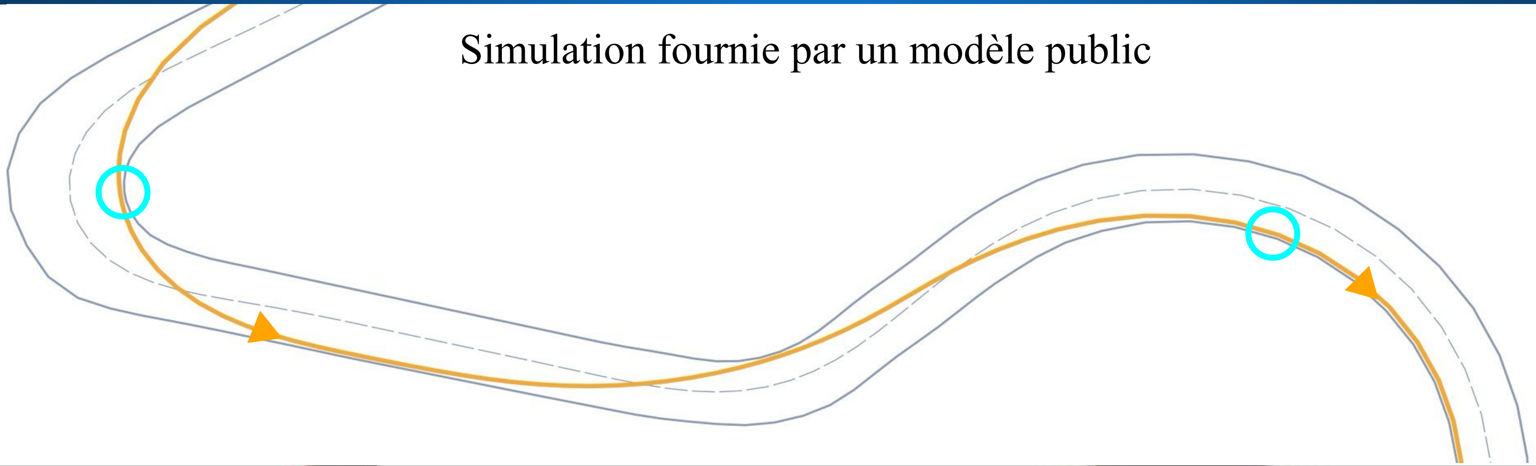
1. Introduction
2. Théorie
3. Modélisation
4. Résultats

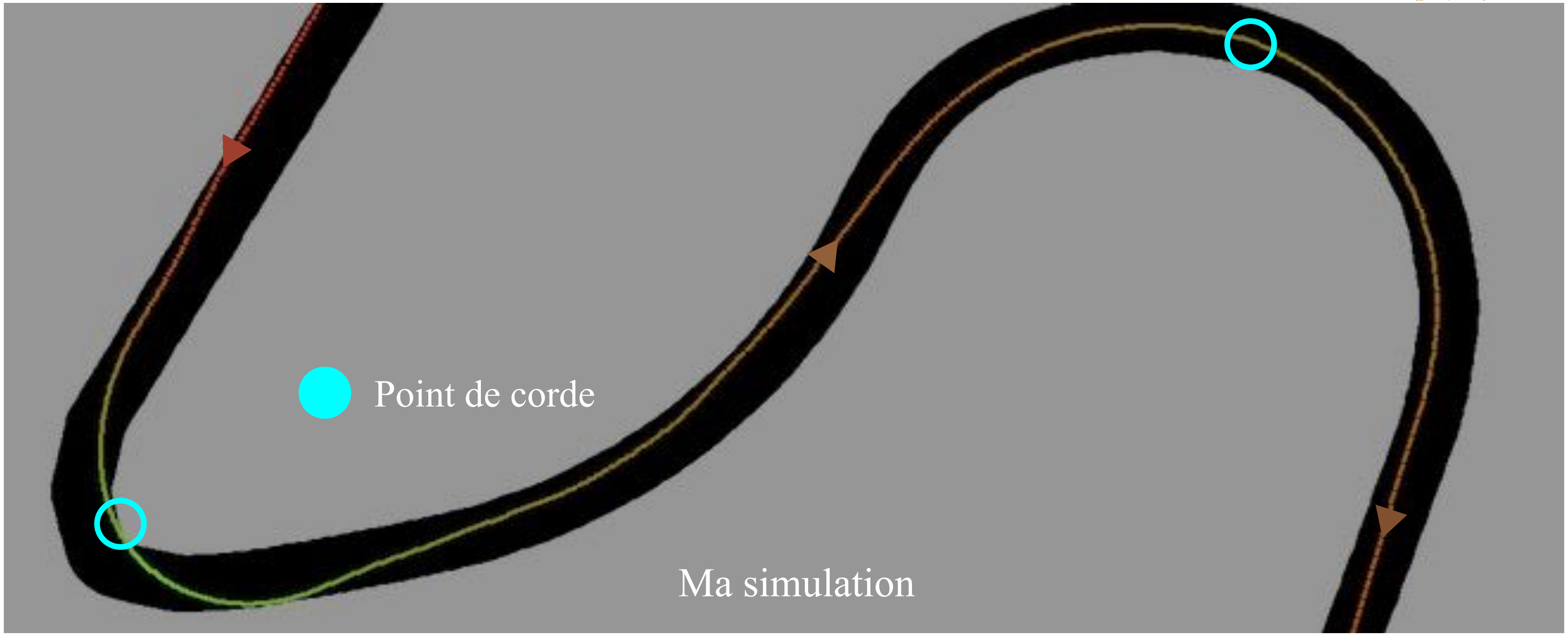
Résultats - Visualisation (1)



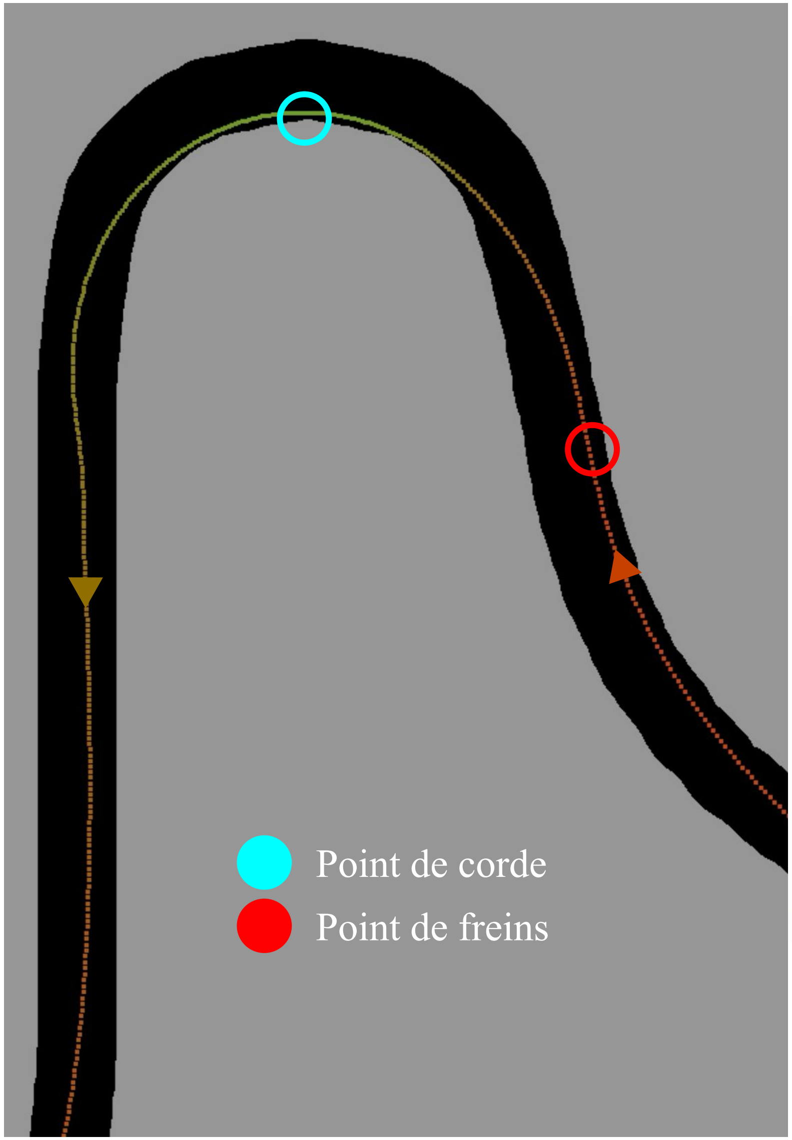


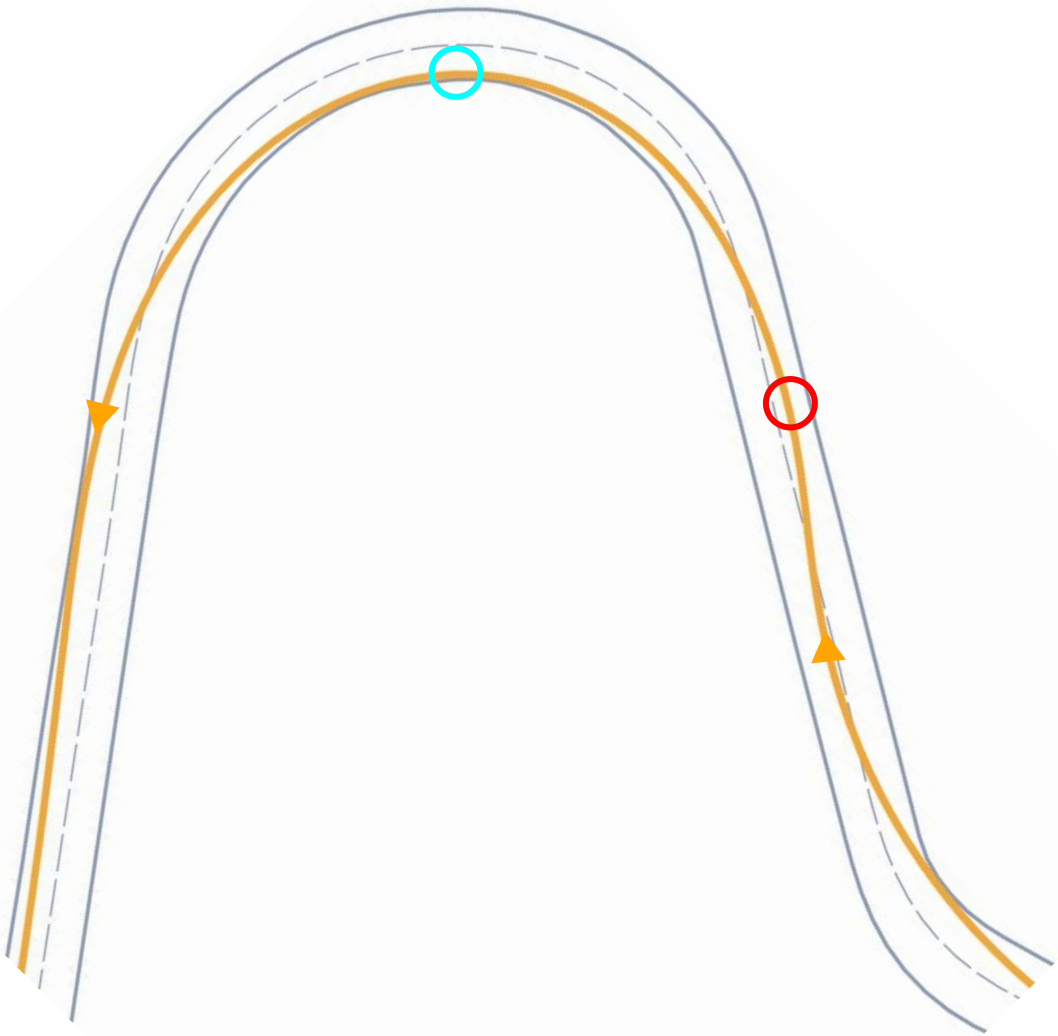
Simulation fournie par un modèle public





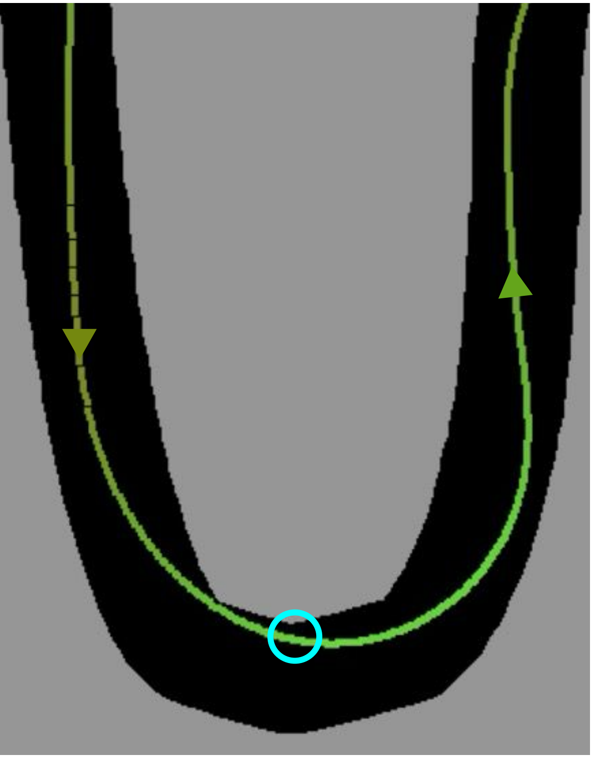
Résultats - Comparaison (2)





Résultats - Comparaison à des vrais pilotes de F1

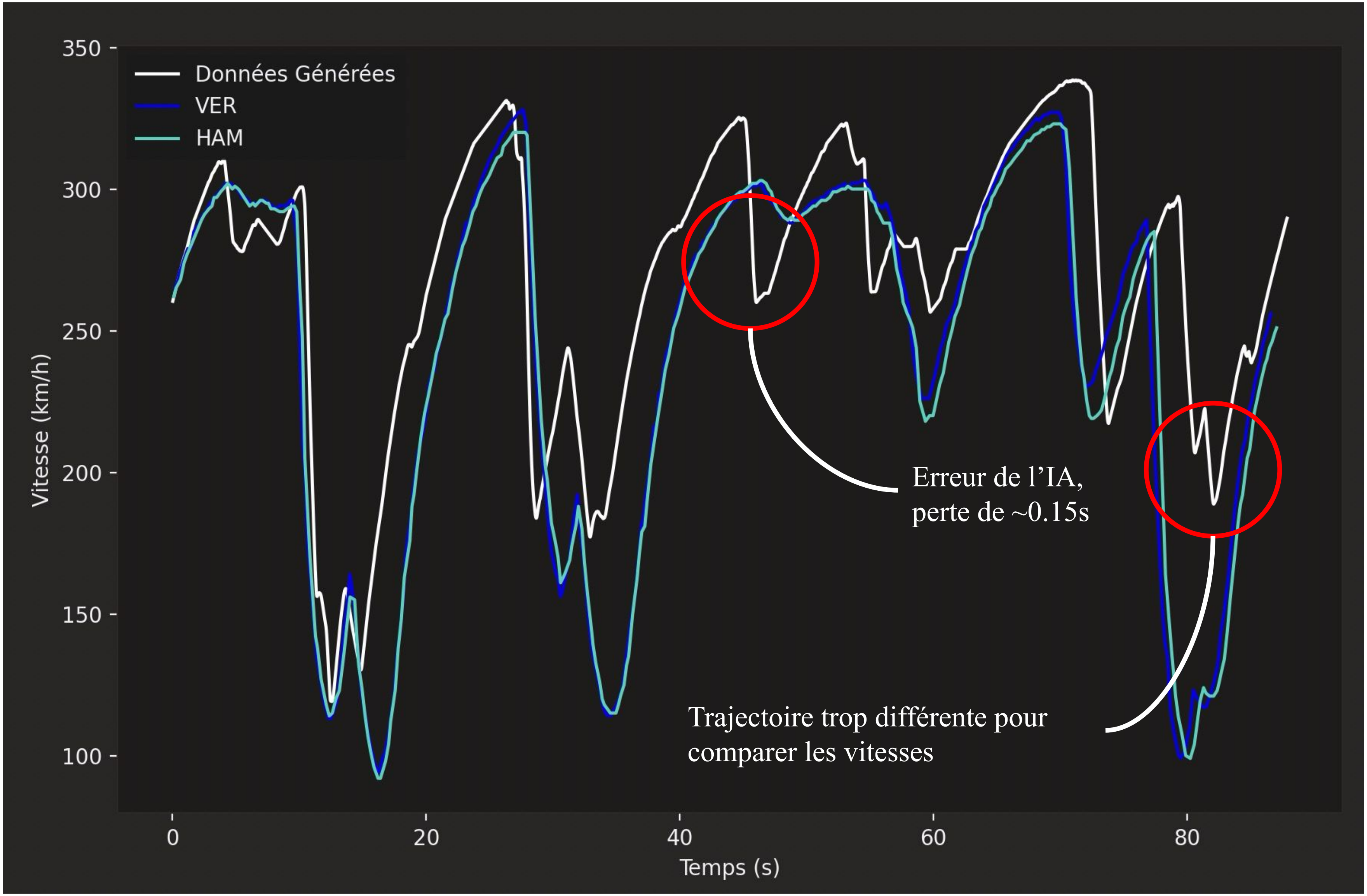
Mon modèle



Point de corde

Pilotes de Formule 1

Résultats - Comparaison à des pilotes de F1



Résultats finaux

Résultats sur tous les circuits:
Écart moyen: 3.1s
Écart relatif moyen: 3.4%

En excluant Monaco et le Hungaroring
Écart moyen: 1.6s
Écart relatif moyen: 1.8%

Circuit	Générations	Temps d'entrainement (Heures CPU)	Delta	À convergé?
melbourne	6526	413.22	-2.867s	Oui
monaco	6259	404.67	-8.567s	Oui
cota	5658	385.44	-1.167s	Oui
abu-dhabi	6054	398.11	-2.933s	Oui
redbullring	5126	368.42	-1.583s	Oui
spa	2415	147.46	-4.033s	Oui
silverstone	2410	153.68	+1.867s	Oui
hungaroring	4991	364.1	-8.817s	Oui
spain	2845	162.66	-0.4s	Oui
imola	5156	369.38	-1.5s	Oui
brazil	5067	366.53	+5.133s	Oui
mexico	4005	198.34	-2.9s	Oui
monza	5441	378.5	-3.733s	Oui
singapoure	9121	669.58	+1.483s	Oui
bahrain	4885	360.7	-0.533s	Non

Conclusion

- Modèle suffisamment précis pour obtenir des trajectoires réalistes et réelles

- Pour aller plus loin, on pourrait:
  - Ajouter plusieurs voitures et des collisions
  - Rendre les paramètres évolutifs (pluie, pneus utilisés, condition du circuit, température)
  - Donner plus de contrôle à l'IA (équilibre des freins, système de réduction de traînée), pour être plus proche d'une situation réelle

Références

Références:

1. *(PDF) optimization of driver and chassis of FWD racing car for faster cornering*. Available at: https://www.researchgate.net/publication/301322219_Optimization_of_driver_and_chassis_of_FWD_racing_car_for_faster_cornering
2. Davidbonde *et al.* (2021) *Track racing - setup and riding style*, *esk8.news*. Available at: https://forum.esk8.news/t/track-racing-setup-and-riding-style/9172?page=9
3. Williams, D. (2016) *3.2: Fundamentals: The optimal racing line*, *Virtual Racing School (VRS)*. Available at: https://virtualracingschool.com/academy/iracing-career-guide/second-season/the-optimal-racing-line/
4. *How to drive the perfect racing line - fast through Apex & Exit* (2022) *Driver61*. Available at: https://driver61.com/uni/racing-line/
5. *(PDF) Image Analysis and machine learning in Digital Pathology: Challenges and opportunities*. Available at: https://www.researchgate.net/publication/304823681_Image_Analysis_and_Machine_Learning_in_Digital_Pathology_Challenges_and_Opportunities
6. PIER LUCA LANZI : Searching for the optimal racing line using genetic algorithms : Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, IEEE
7. YING XIONG : Racing Line Optimization : Masters Thesis, Shanghai Jiao Tong University with Massachusetts Institute of Technology, 2010
8. ARAVINDPAI PAI : Analyzing Types of Neural Networks in Deep Learning : https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-indeep-learning/
9. S. RISUCHAT : Development of a car physics engine for games : Masters Thesis, Bournemouth University, 2012
10. FÉDÉRATION INTERNATIONALE DE L'AUTOMOBILE : FIA 2023 Sporting Regulations - Formula 1 : https://www.fia.com/sites/default/files /fia_2023_formula_1_sporting_regulations_-_issue_2_-_2022-09-30.pdf
11. MARCO GADOLA, DAVID VETURRI, DANILO CAMBIAGHI : A Tool for Lap Time Simulation : Research Paper, University of Brescia, 1996

Références:

11. VINCENT BARRA, LAURENT MICLET, ANTOINE CORNUÉJOLS : Apprentissage artificiel - 4e édition, Concepts et algorithmes : 2021, ISBN-13 978-2-416-00104-8

12. Collection de Pistes de course F1: Vecteur Gratuite (no date) *Freepik*. Available at: https://fr.freepik.com/vecteurs-libre/collection-pistes-course-f1_2775878.htm#query=circuit%20f1&position=4&from_view=keyword&track=ais_user&uuid=ba681c3c-33ee-489b-b17b-a5f480b33beb (Accessed: 23 May 2024).

12. Anatomy of an overtake using DRS: A data analysis perspective, Prashant Bhadauria

13. Carrera Go, https://noscollections.ddns.net/carrerago/?p=656

Annexe

Une configuration simplifiée

- 3 points importants:

  - Point de virage
  - Point de corde
  - Point d'accélération

- On ne considère qu'un unique virage dans ce modèle

- Plusieurs trajectoires possibles

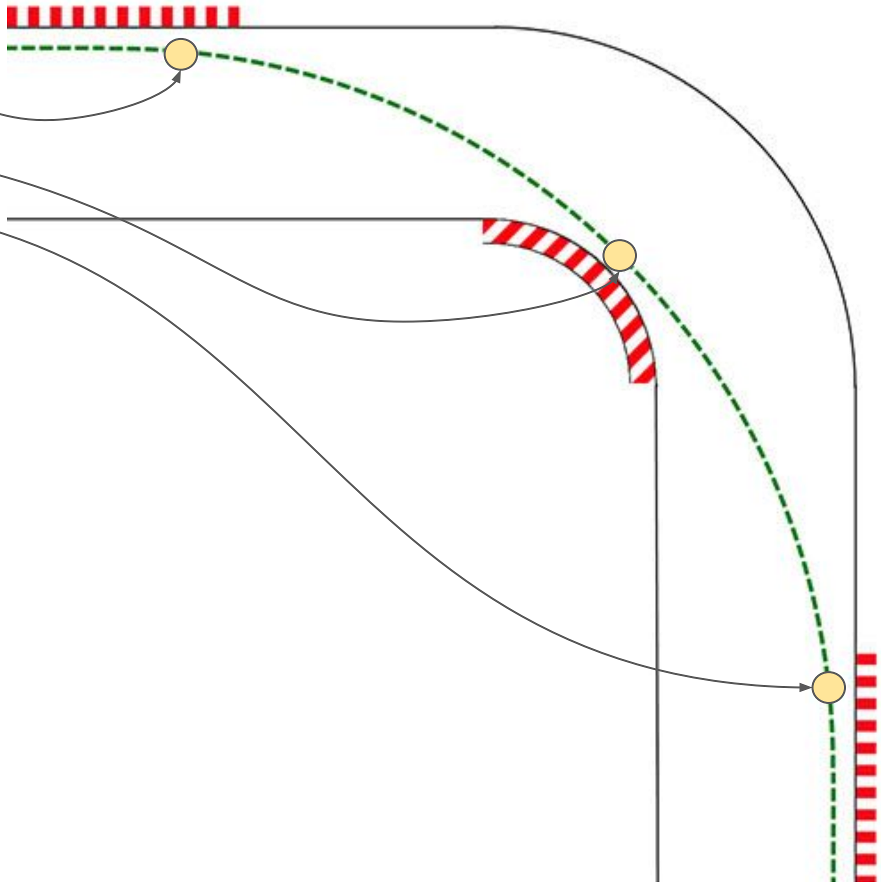


Illustration: Trajectoire intuitive sur un seul virage[3]

Limites de cette modélisation théorique simple

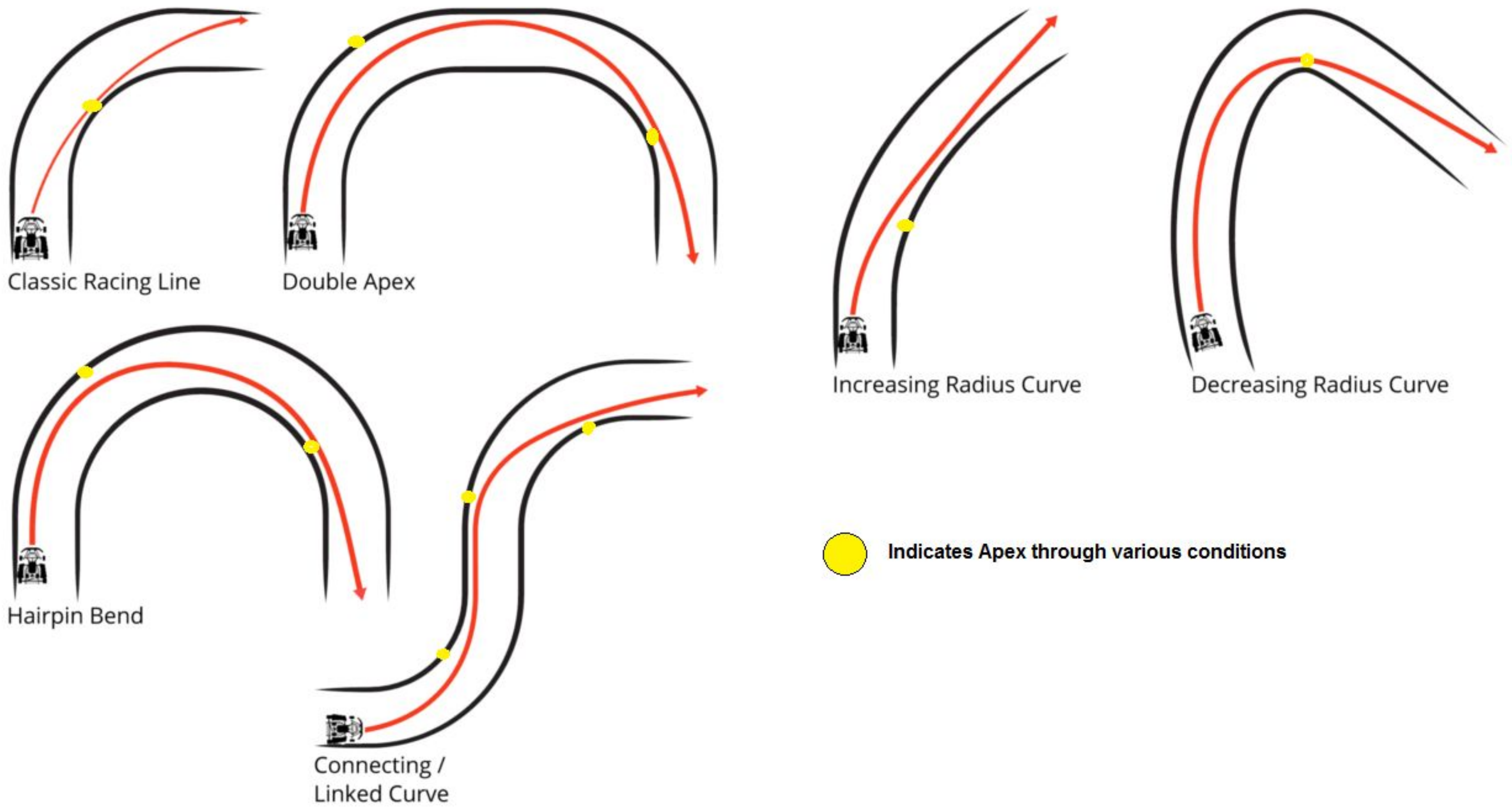


Illustration: Différents types de trajectoires[2]

D’autres approches non abordées

- Courbes de Bézier/splines
- Résolution analytique par la vitesse maximale
- Résolution par intelligence artificielle et apprentissage (méthode choisie)

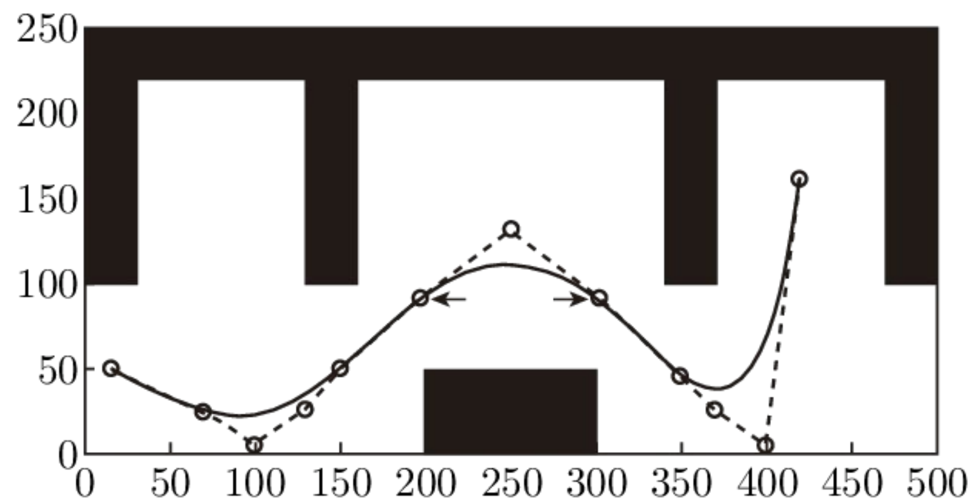


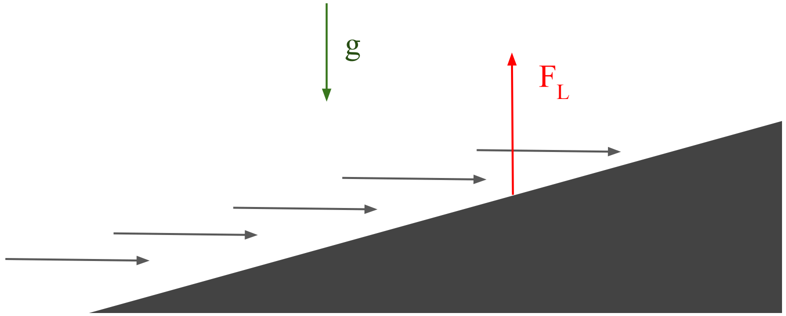
Illustration: Courbes de Bézier

Estimation des effets aérodynamiques

Pression dynamique: $$q = \frac{1}{2}\rho C^2$$

Coefficient de portée: $$C_L \approx 2\pi \sin(\alpha)$$

Force d'appui: $$F_L = q \cdot S \cdot C_L$$



Expressions utiles: $$\alpha = \tan^{-1}\left(\frac{H}{L}\right) \quad S = L \times W \quad \sin(\tan^{-1}(x)) = \frac{x}{\sqrt{1+x^2}}$$

Ainsi: $$F_L = \left(\frac{1}{2}\rho C^2\right) \cdot (L \times W) \cdot \left(2\pi \sin\left(\tan^{-1}\left(\frac{H}{L}\right)\right)\right) \text{ et } \quad m_{eq} = \frac{F_L}{g}$$

En simplifiant: $$m_{eq} = \frac{\rho C^2 LW\pi \frac{H}{\sqrt{H^2+L^2}}}{g}$$

Estimation des effets aérodynamiques - Calcul du coefficient de porté

$$L = \rho V_\infty \Gamma$$

Théorème de Kutta-Joukowski

$$\Gamma = 2 V_\infty c \alpha$$

c: longueur de corde

$$C_L = \frac{L}{\frac{1}{2}\rho V_\infty^2 S}$$

Coefficient de portée

$$C_L = \frac{2\Gamma}{V_\infty S}$$

$$C_L = m\alpha$$

Avec m une constante, qui peut être trouvée empiriquement

Différents réseaux de neurones

- Deux catégories de réseaux:
  - Propagation avant
  - Réseau récurrent

- Méthodes d'apprentissage:
  - Apprentissage supervisé
  - Apprentissage non supervisé
  - Apprentissage par renforcement

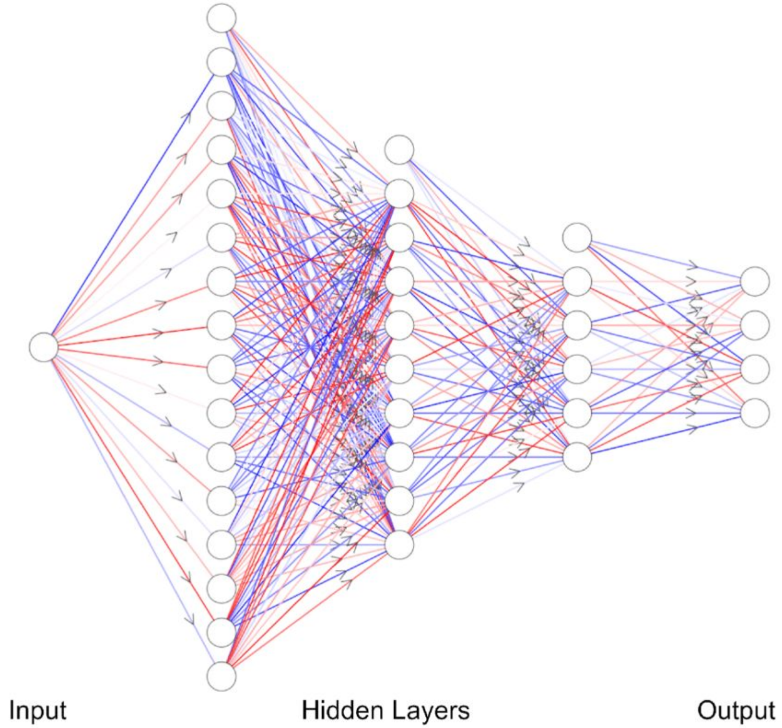


Illustration: Réseau de neurone à propagation avant [5]

L'apprentissage génétique

1. Évaluation de la population

2. Choix des survivants
    a. Choix aléatoire
    b. Mélange de différents agents
    c. Choix pondéré au score

3. Début de la prochaine génération

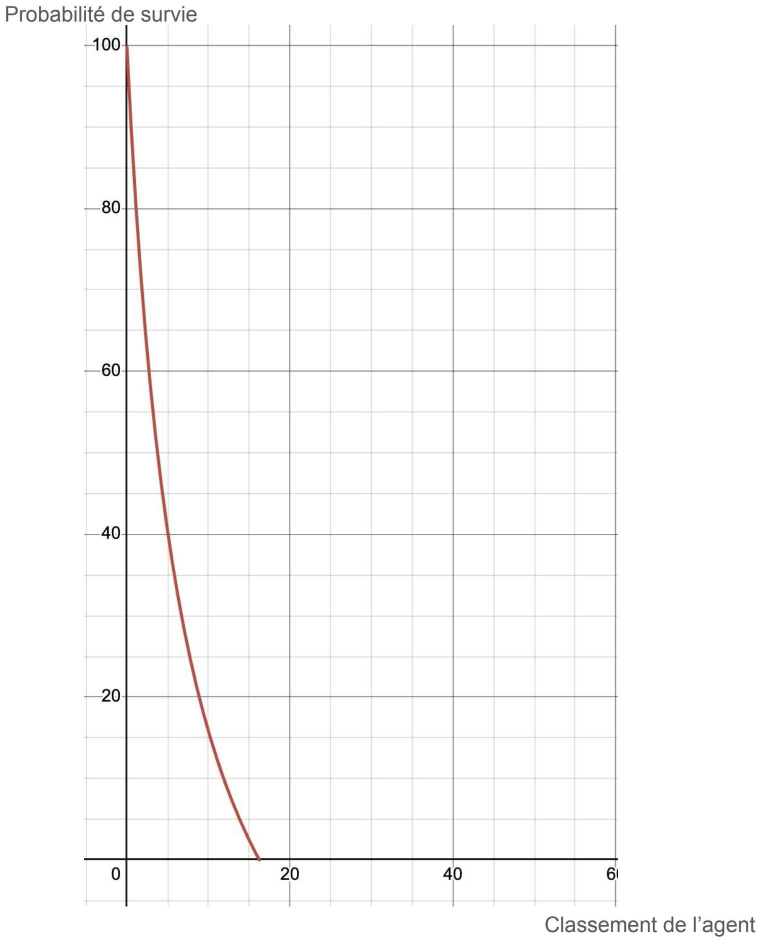
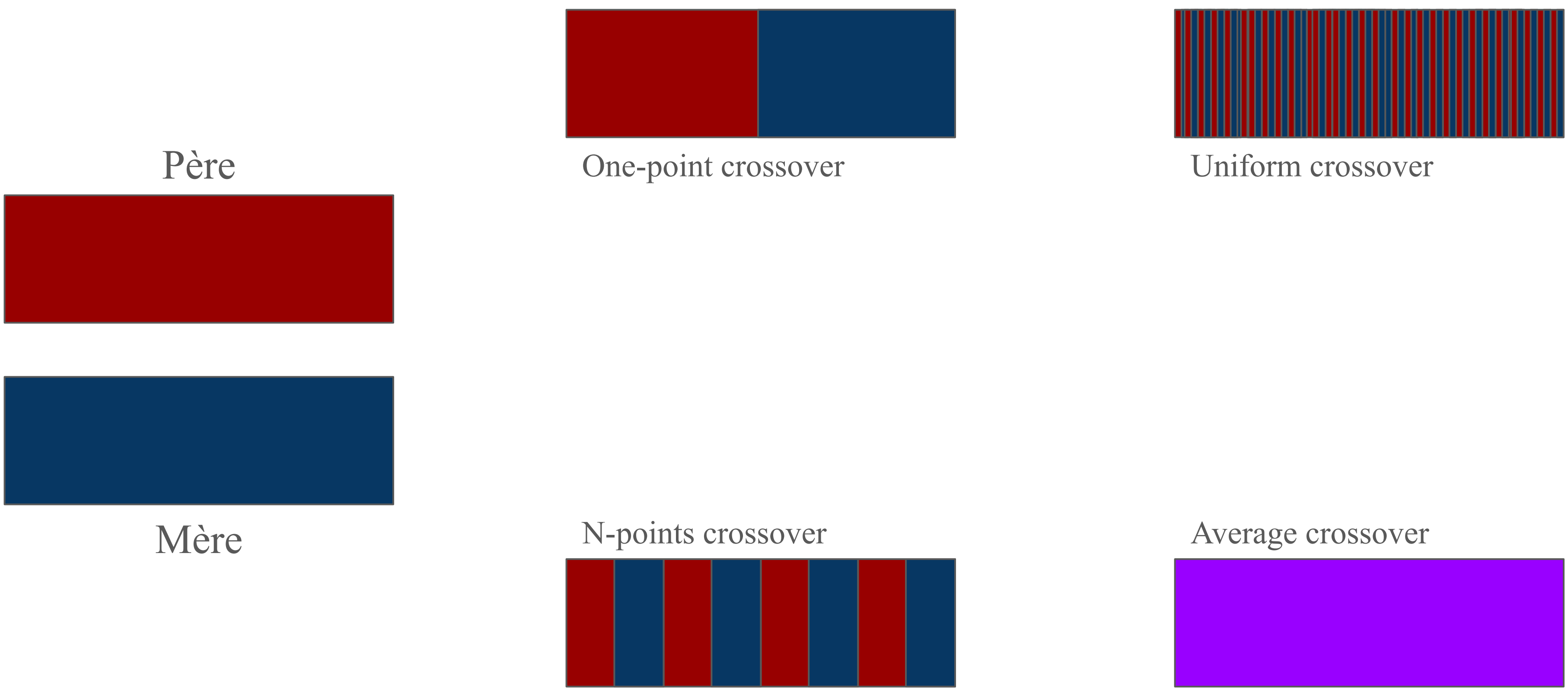
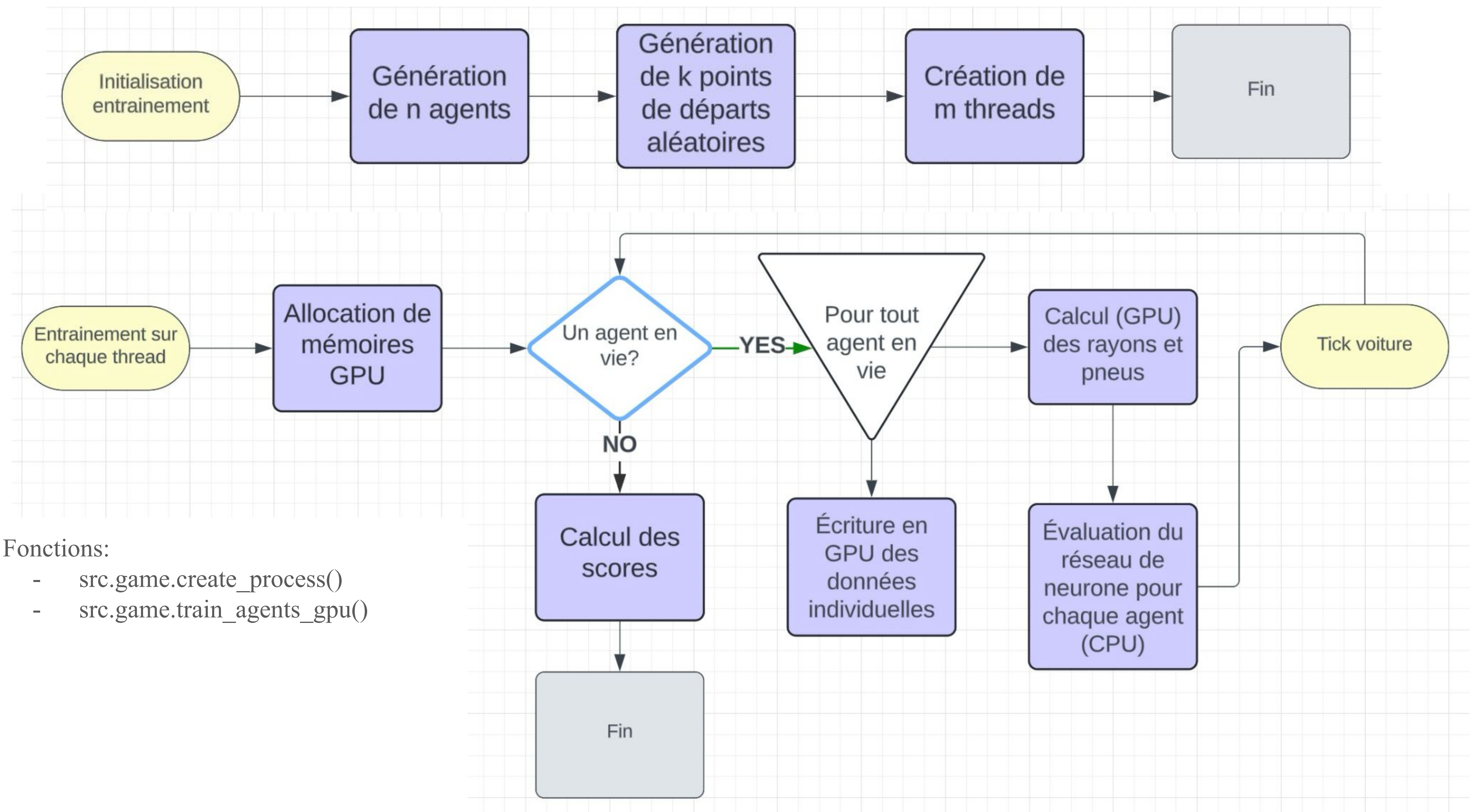


Illustration: Choix pondéré au score

Mélange des agents

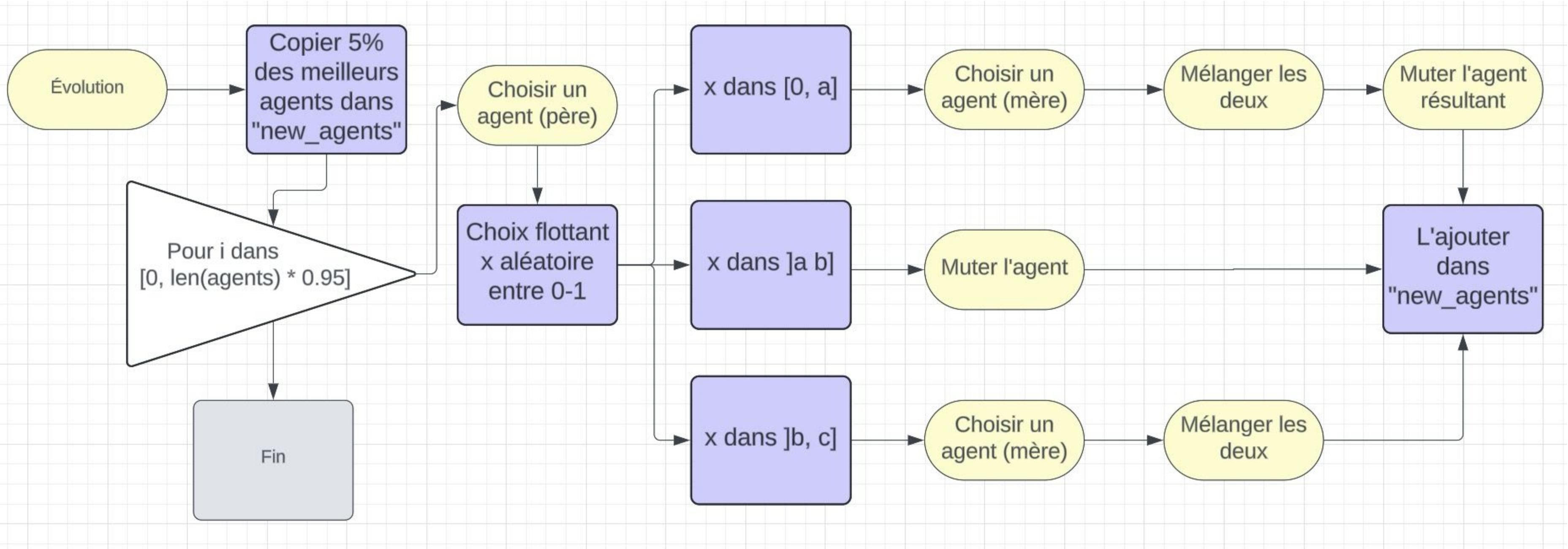




Fonctions:

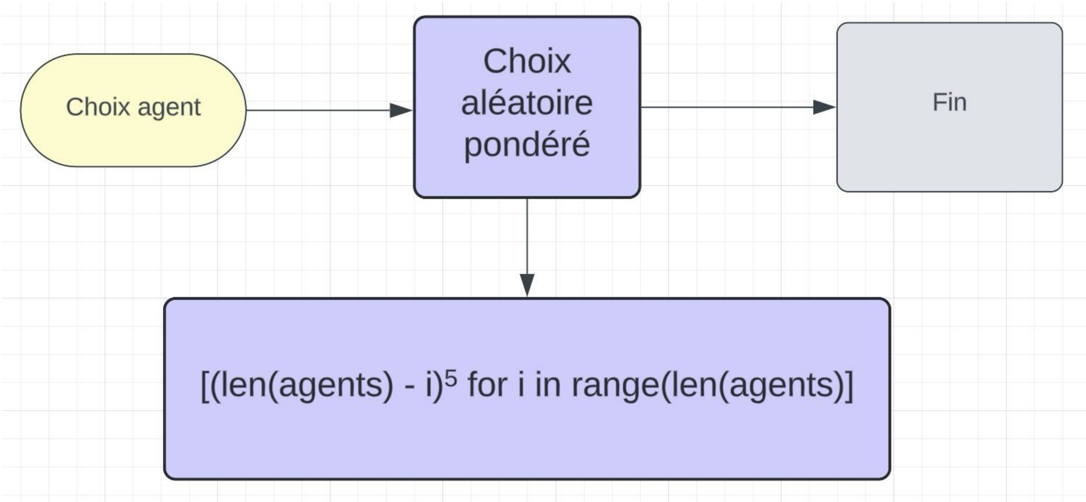
- src.game.create_process()
- src.game.train_agents_gpu()

Réseau de neurone - Évolution 1



Fonctions:

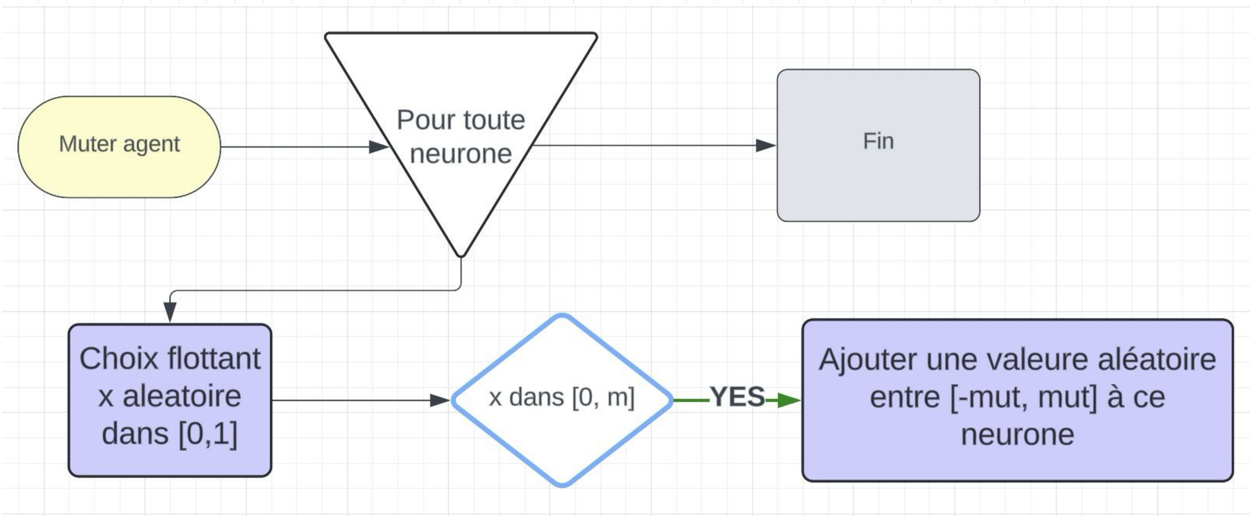
- src.environment.next_generation()
- src.environment.mutate()
- src.environment.crossover()
- src.environment.linear_weighted_selection()



m: Paramètre fixé, choix empirique

Fonctions:

- src.environment.linear_weighted_selection()
- src.agent.mutate()

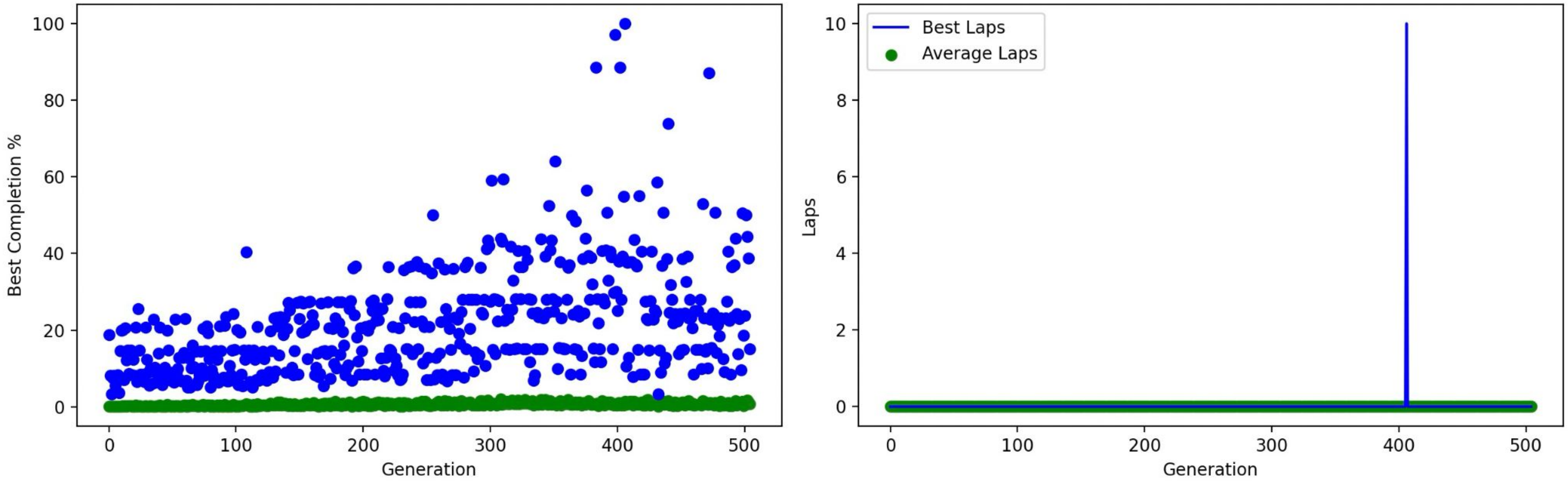


Réseau de neurone - Évaluation des agents

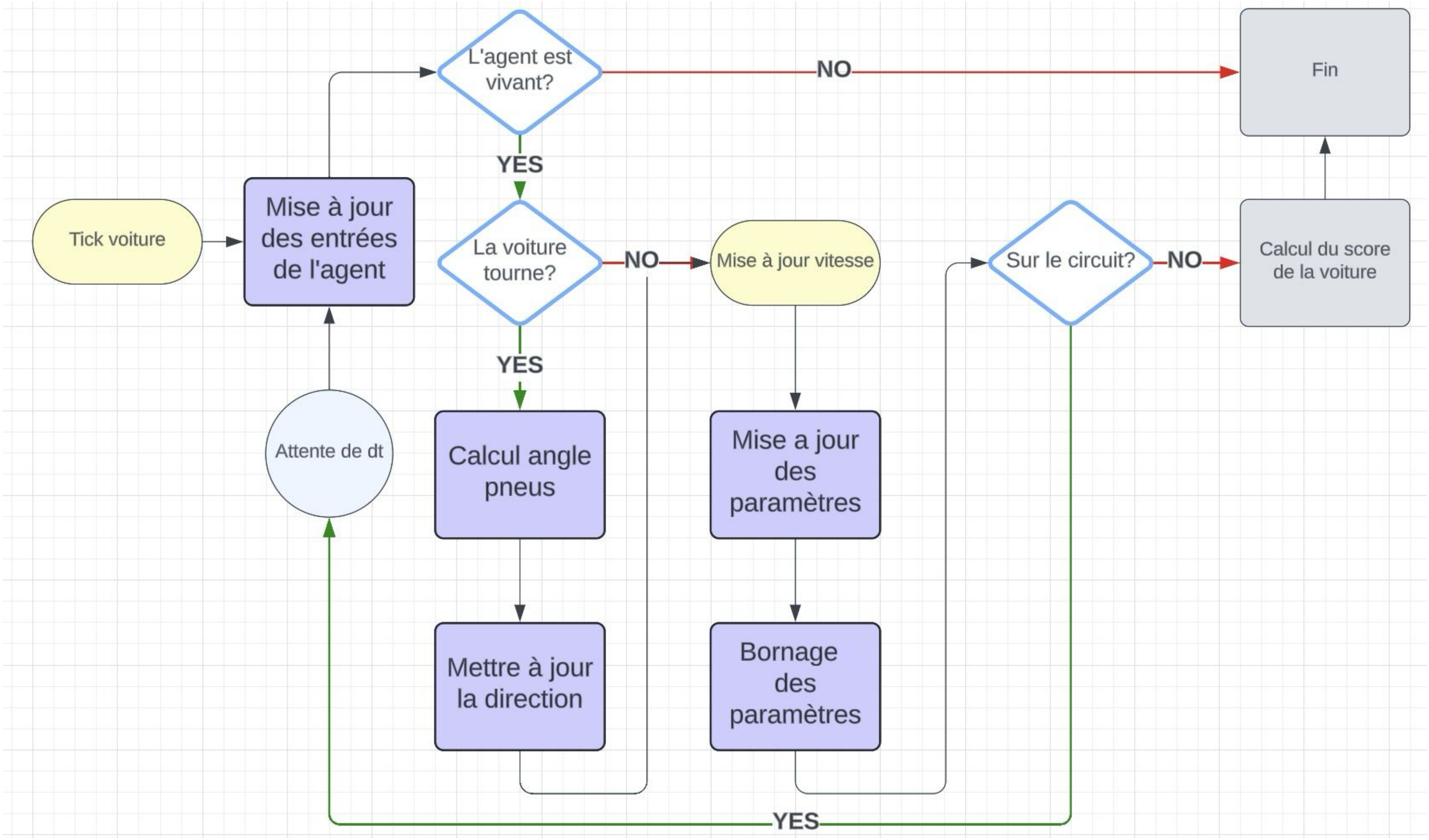
2 méthodes d'évaluation:

- Avant que l'agent soit capable de faire un tour complet ⇒ Fonction de score
- Après que celui-ci finisse un tour complet ⇒ Temps du tour

```
ranked_agents = sorted(self.agents, key=lambda x: (x.car.laps, x.car.score, -x.car.lap_time), reverse=True)
```

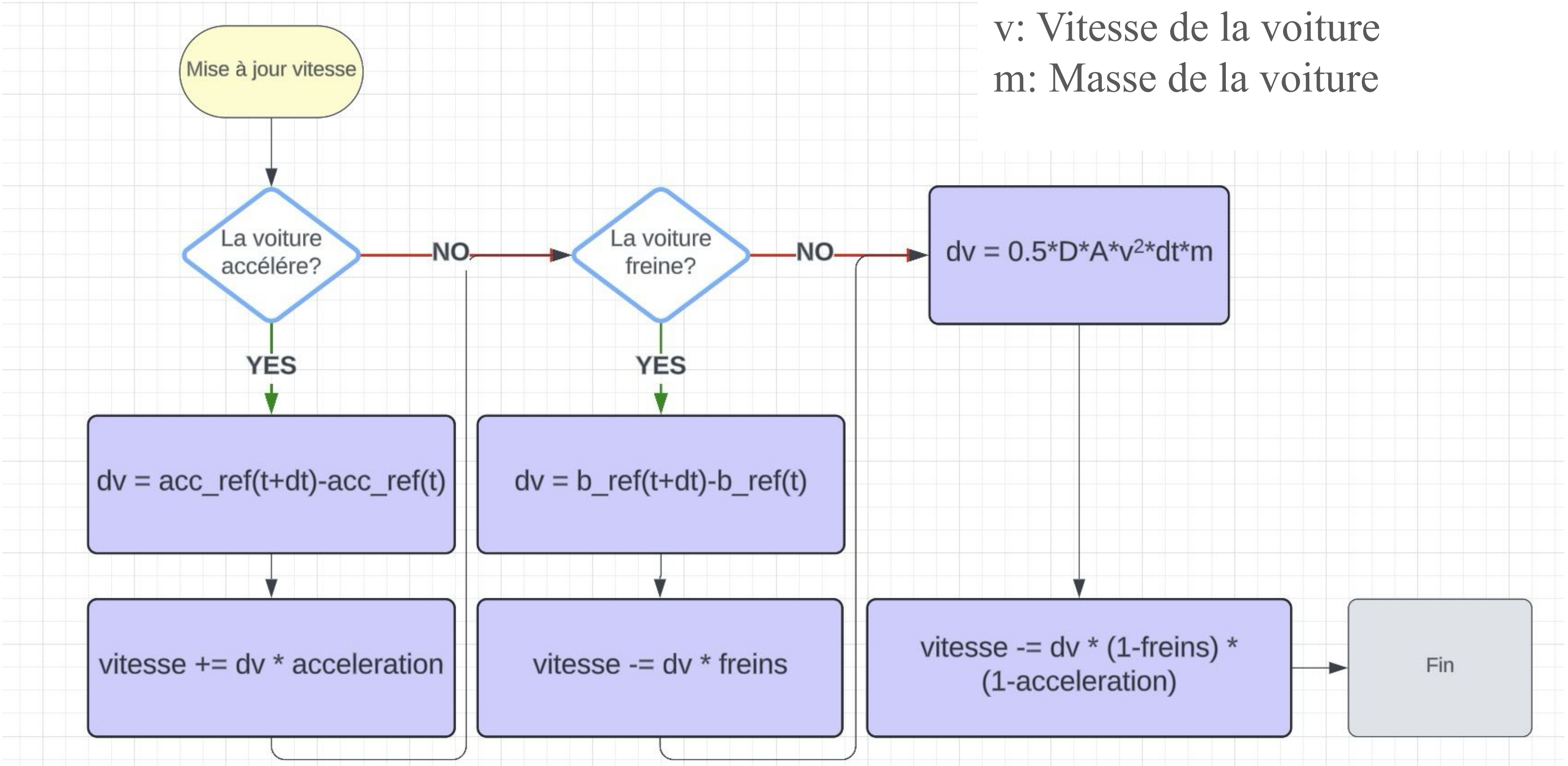


Modélisation physique - Voiture 1

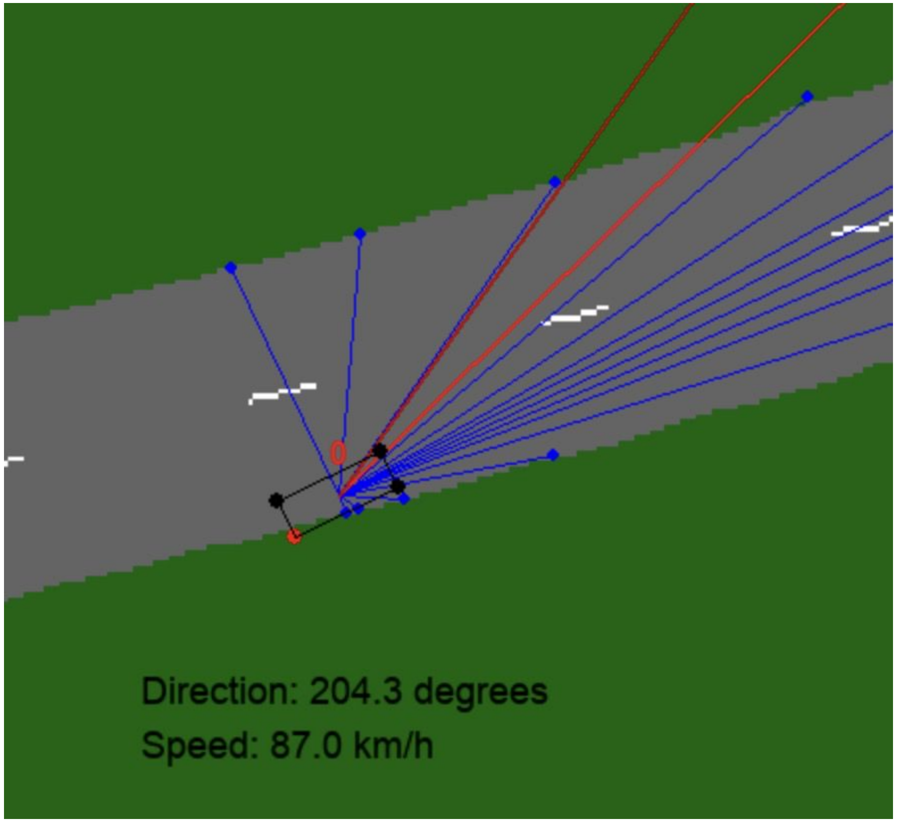


Modélisation physique - Voiture 2

D: Viscosité dynamique
A: Surface de contact
v: Vitesse de la voiture
m: Masse de la voiture



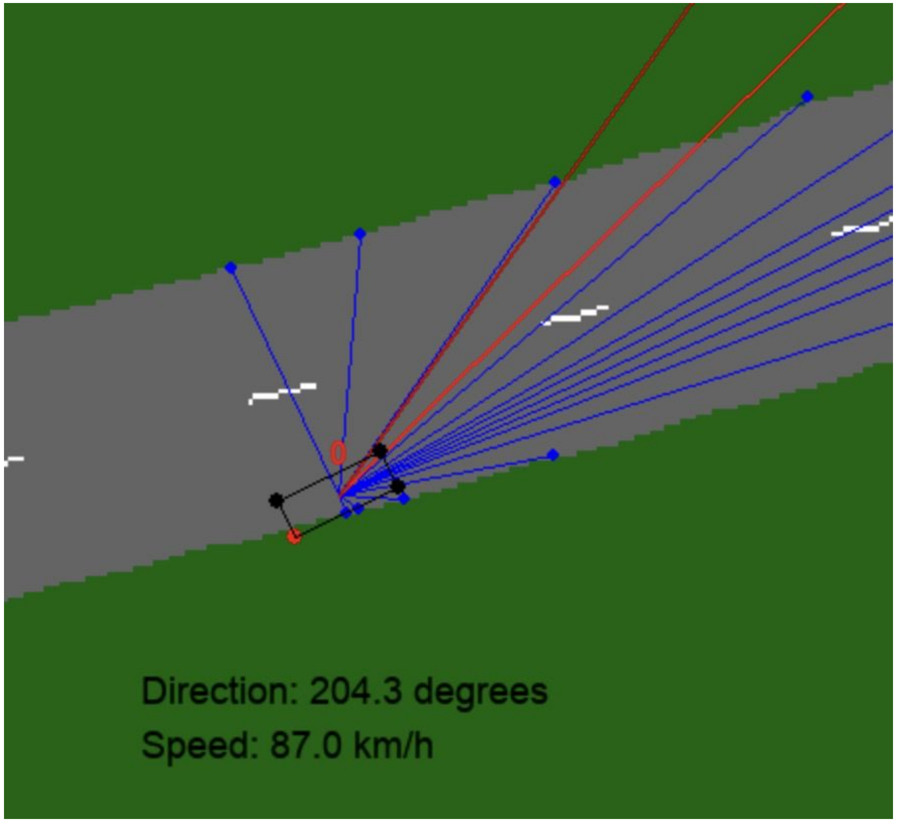
- **Bleu:** Distance à la bordure, pour différents angles relatif à la direction de la voiture (15)
- **Rouge**: Distances et angles par rapport aux prochains virages (2)
- **Ronds noir:** Roues de la voiture sur la route
- **Ronds rouge:** Roues de la voiture en dehors de la route
- Paramètres physiques



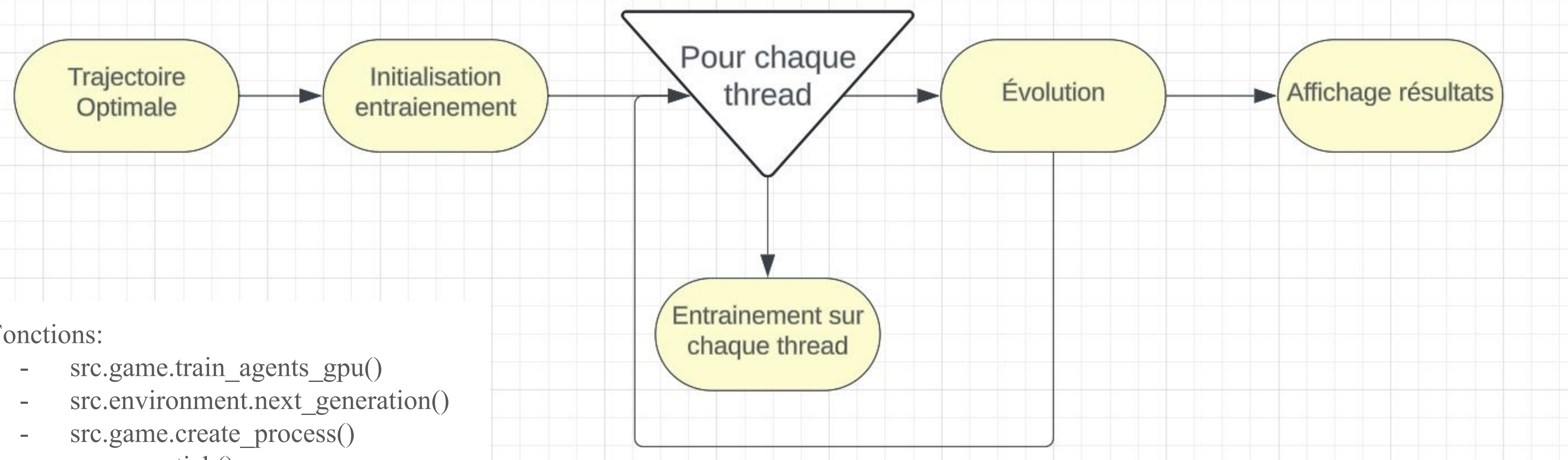
Total: 26 Entrées

Obtentions des entrés

- **Bleu:** Ray tracing (GPU, binary search)
- **Rouge:** Calcul des prochains virages (Algorithme de RANSAC et suivis de ligne, GPU)
- **Noir:** Vérification matricielle (CPU)



Implémentation pour notre problème



Fonctions:

- src.game.train_agents_gpu()
- src.environment.next_generation()
- src.game.create_process()
- src.game.tick()

Obtention des circuits et limitations

- Google Maps ou images de circuits: résolution trop faible

- Format EPS (Vectoriel) transformés en matricielle en 5000x5000

Circuit exclus:

- Suzuka: circuit contenant 2 composantes connexes, compliqué à implémenter

- Circuits avec trop de variations de hauteur ou de virages en inclinaisons

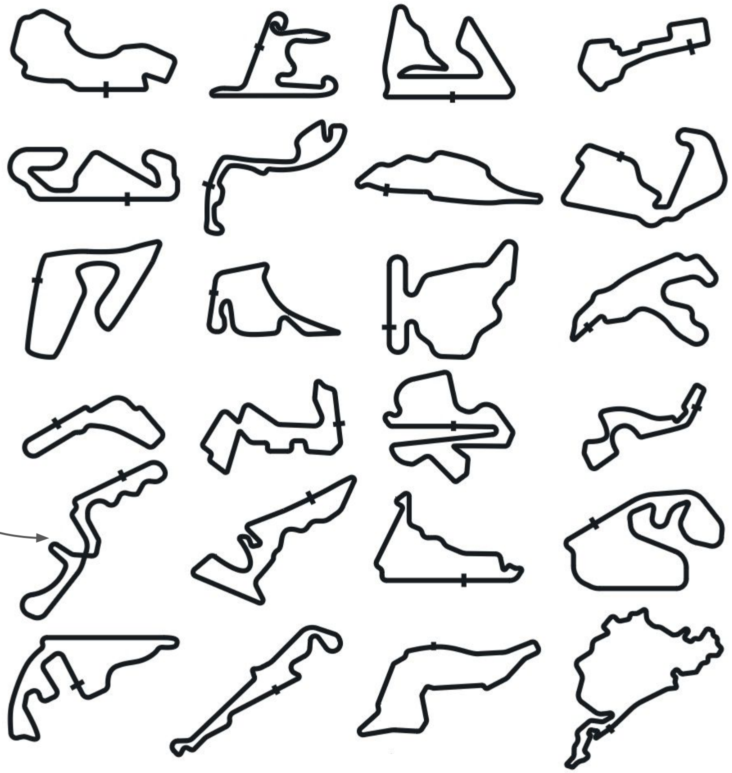
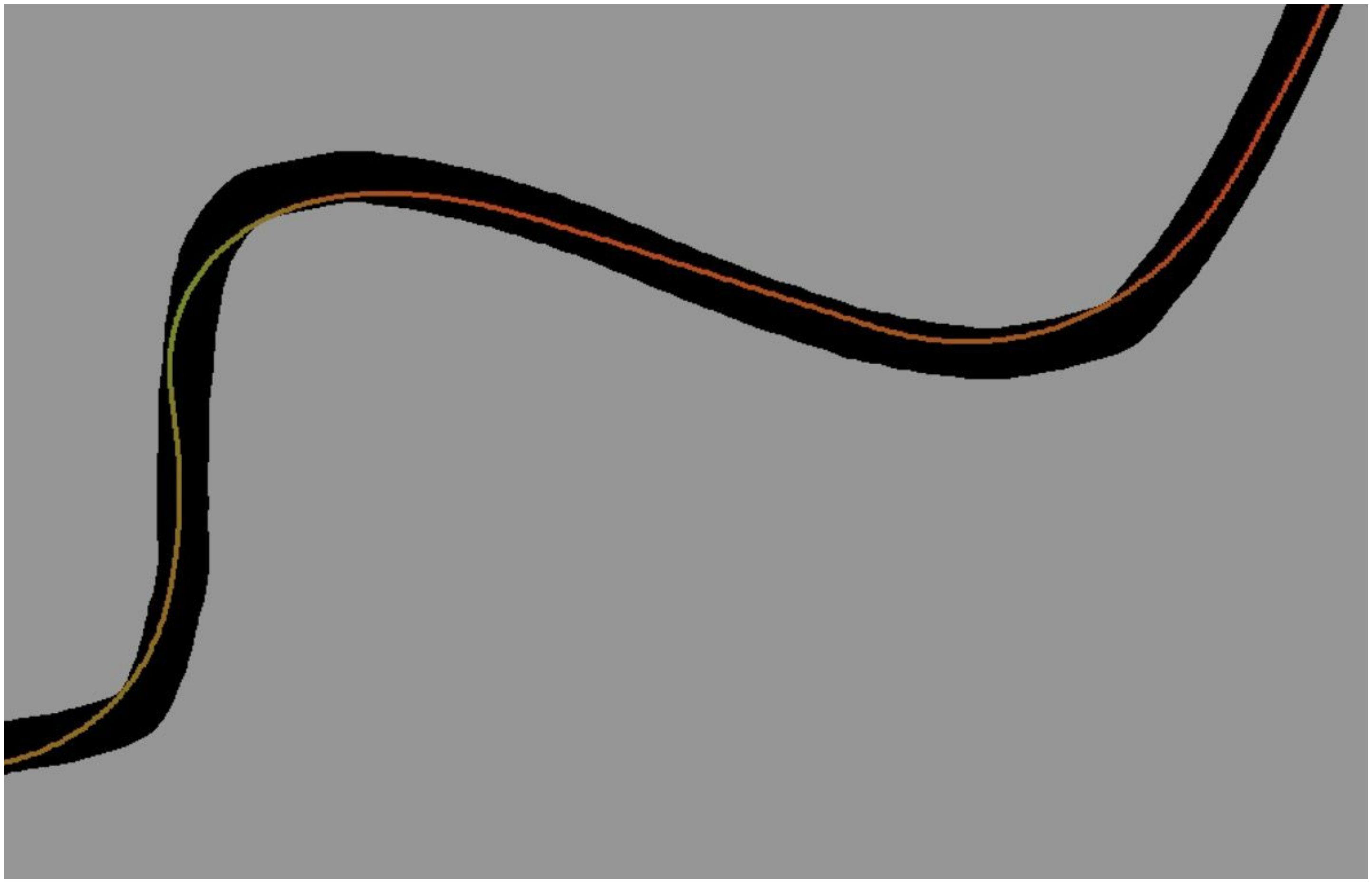
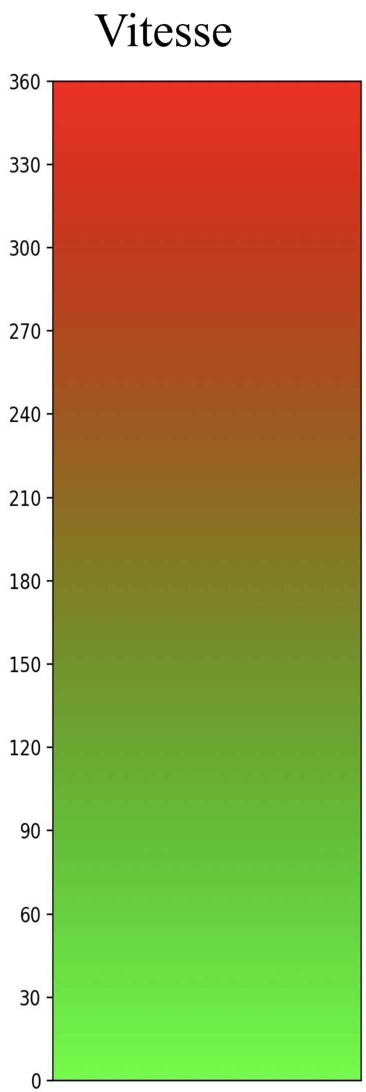


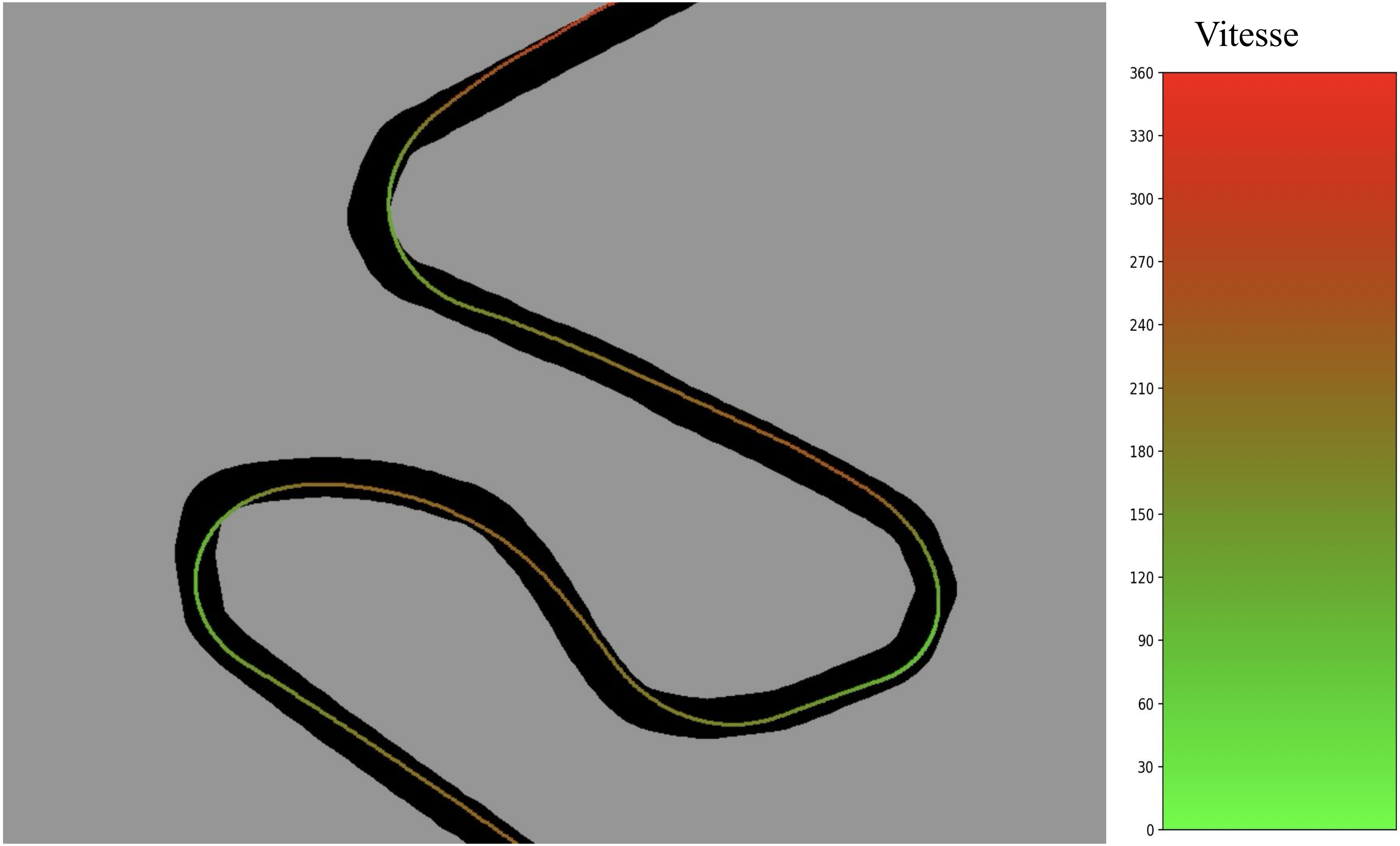
Illustration: Source utilisé pour les circuits[12]

Visualisations supplémentaires - 1



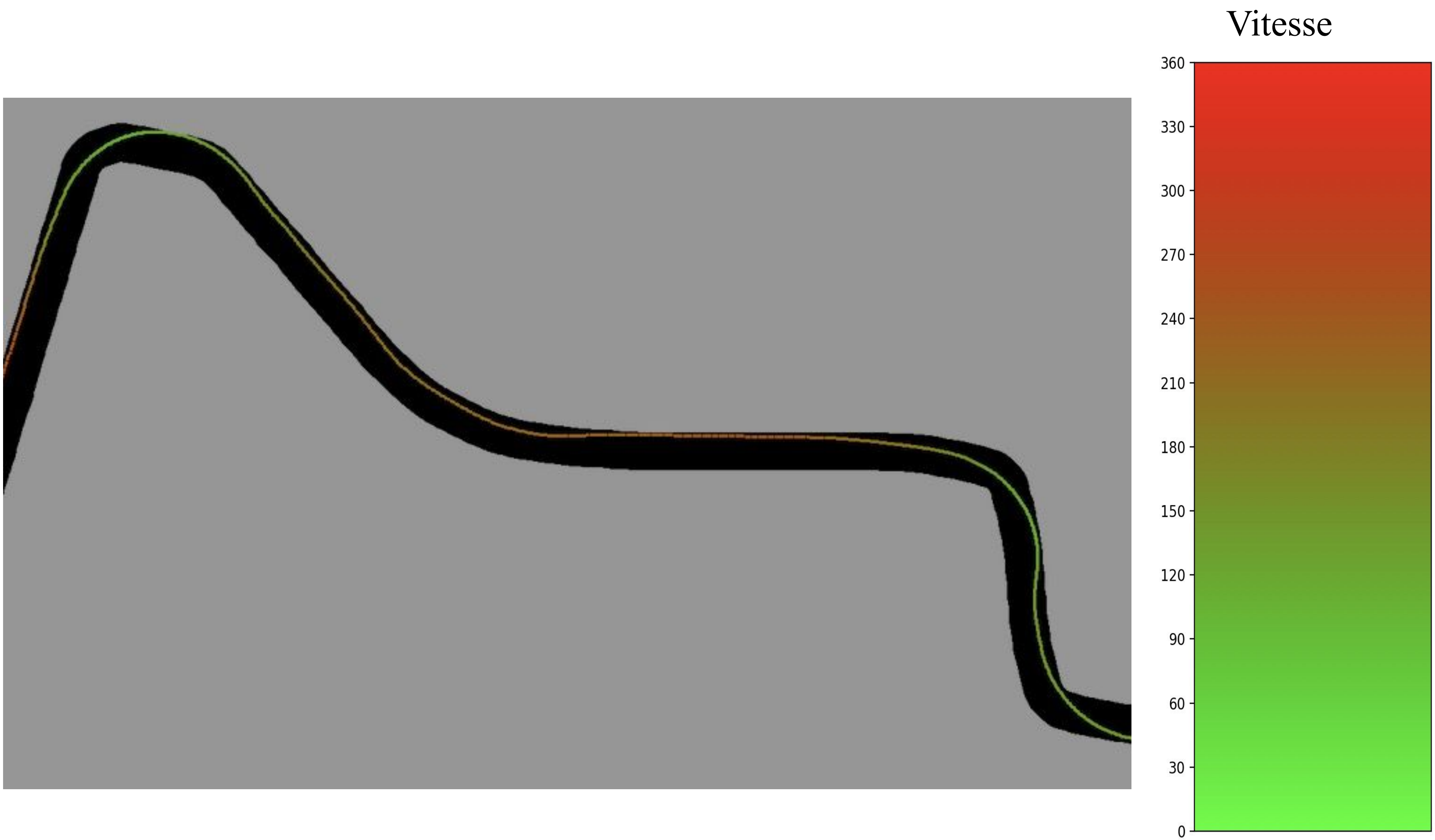


Visualisations supplémentaires - 2

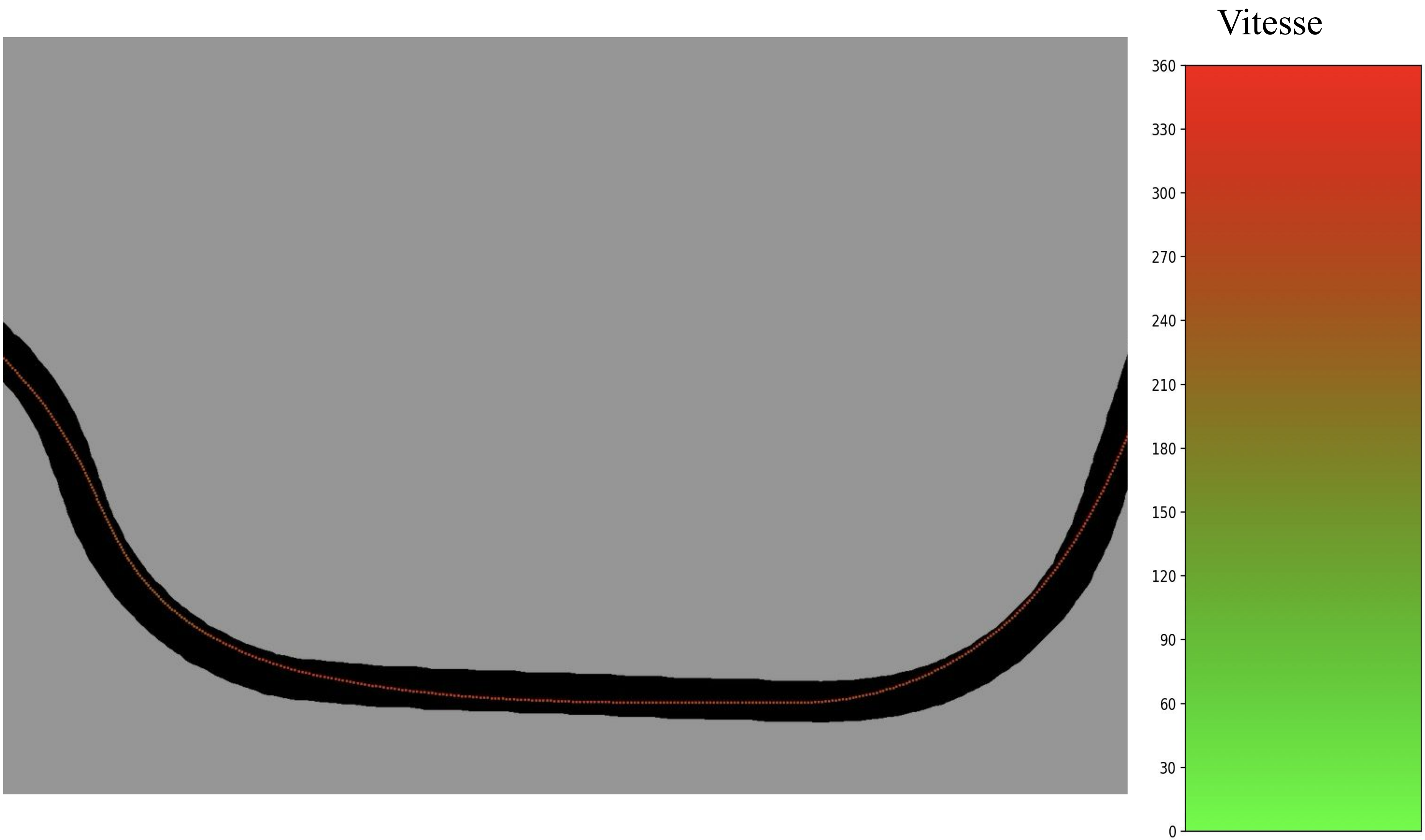


Visualisations supplémentaires - 3

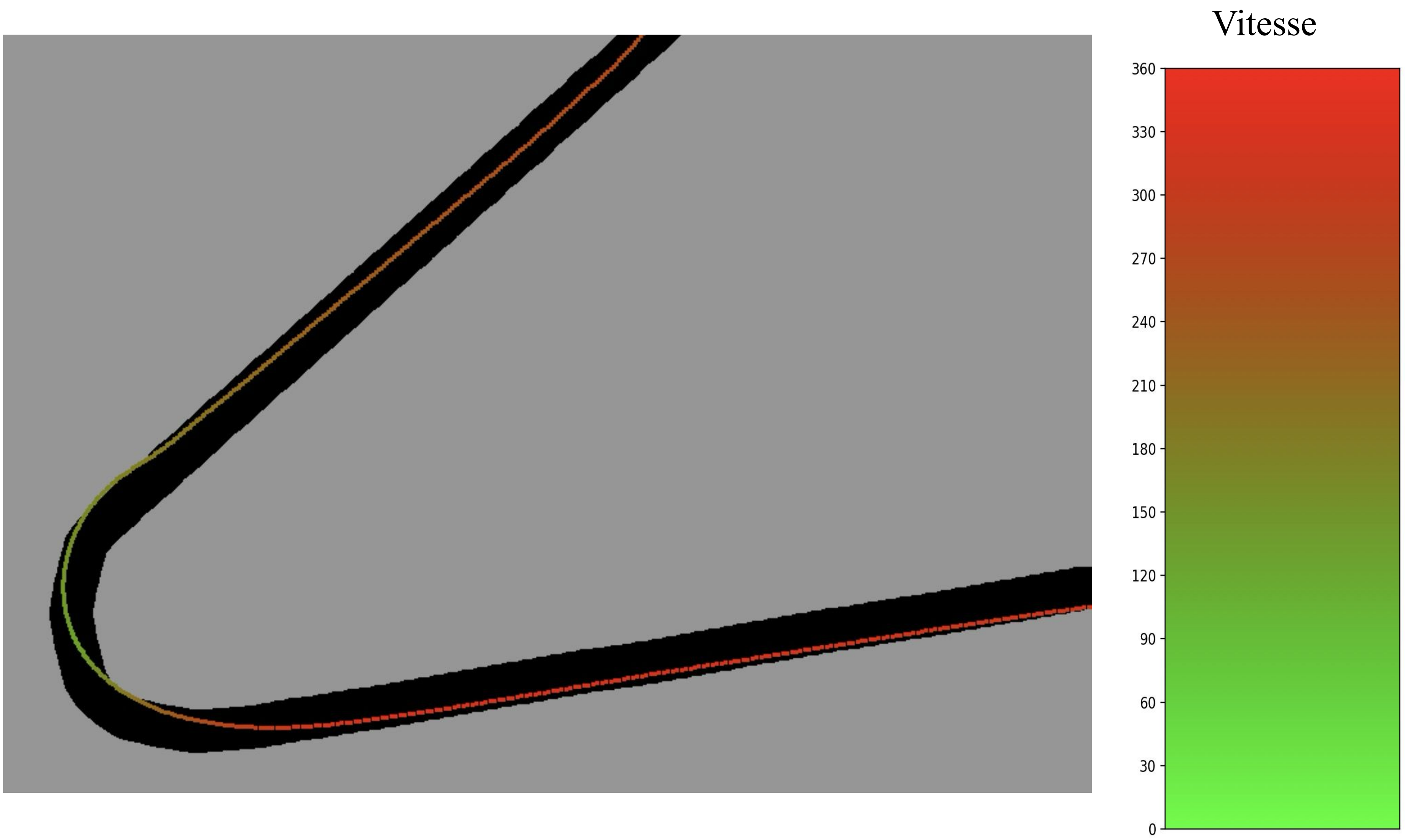
Visualisations supplémentaires - 4



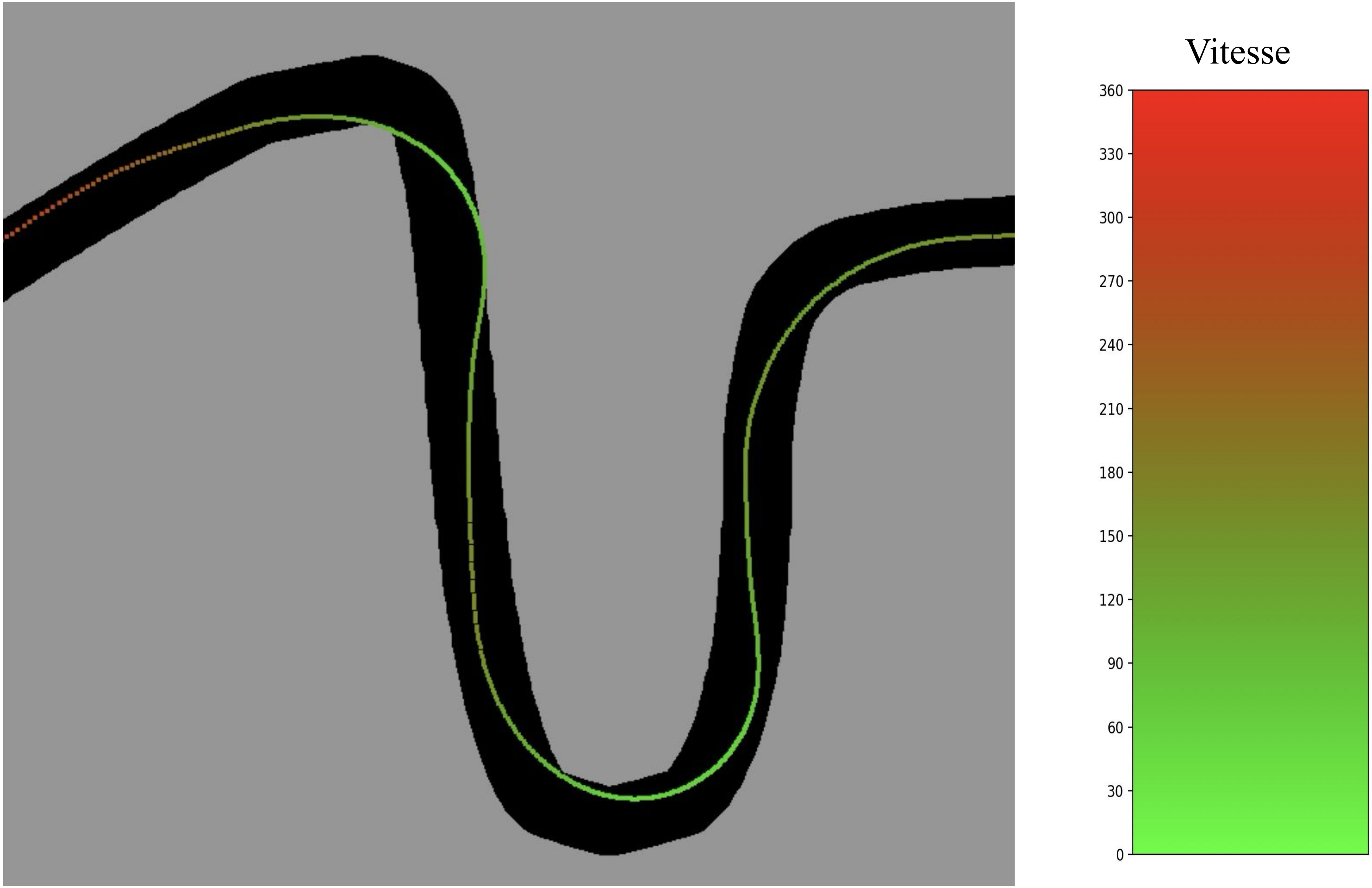
Visualisations supplémentaires - 5

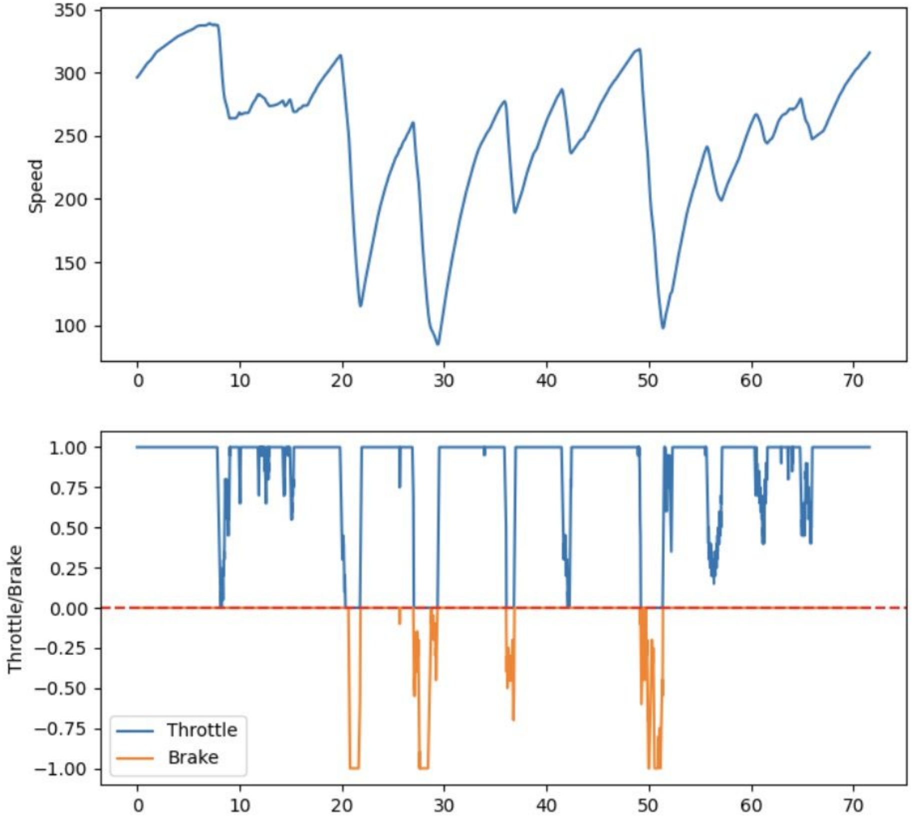


Visualisations supplémentaires - 6

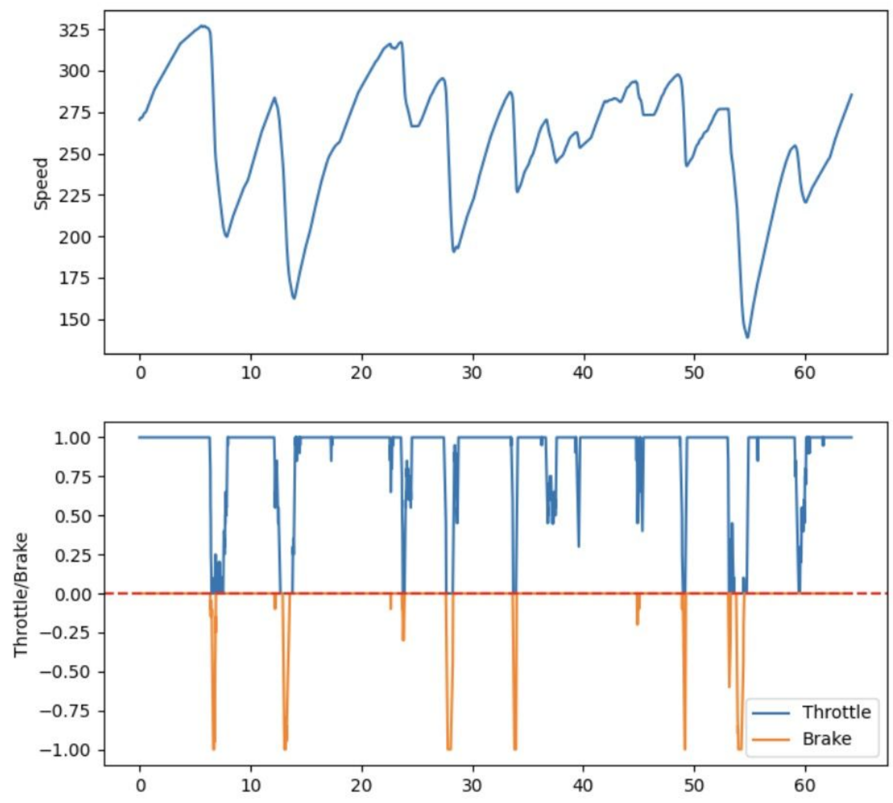


Visualisations supplémentaires - 7



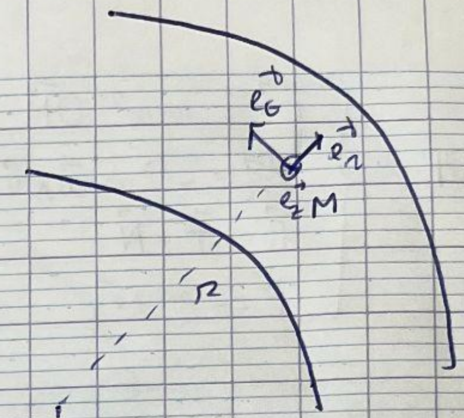


Circuit: Barcelona-Catalunya (Espagne)

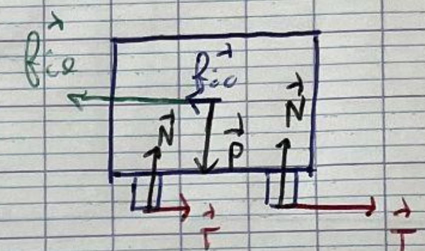


Circuit: Hungaroring (Hongrie)

BDF : $\begin{cases} \vec{P} = -mg\vec{e_z} \\ \vec{f_{ic}} = -2m\vec{\omega} \wedge \vec{v_{R_v}} \\ \vec{f_{ie}} = m\omega^2 \vec{OM} \\ \vec{R} = \vec{T} + \vec{N} \end{cases}$



Avec R_v référentiel de la voiture, en rotation uniforme autour de R_T supposé galiléen.



Donc, $\vec{v_{R_v}} = \vec{0}$

Loi de Coulomb pour le cas limite de l'adhérence :

$$\|\vec{T}\| = f\|\vec{N}\|$$

D'après le PFD dans R_v non galiléen, sur {voiture}

$$m\vec{a} = \vec{P} + \vec{R} + \vec{f_{ie}}$$

$\vec{e_r}$: $$0 = -fmg + m\omega^2 r$$

$$\Leftrightarrow \quad m\frac{v_{max}^2}{r} = fmg$$

$$\Leftrightarrow \quad \boxed{v_{max} = \sqrt{fmg}}$$

Avec $\vec{F_{aero}}$, $\vec{F_{aero}} = -m_{sup}\, g\, \vec{e_z}$

$$m\vec{a} = \vec{P} + \vec{R} + \vec{f_{ie}} + \vec{F}, \qquad \|\vec{T}\| = f(\|\vec{N}\| + \|\vec{F}\|) = fg(m + m_{sup})$$

D'où $$v_{max} = \sqrt{\left(\frac{m+m_{sup}}{m}\right) f g}$$