Performance Evaluation:

Car behaviour optimization

Macéo Ottavy, Mathieu Longatte, Louison Mocq

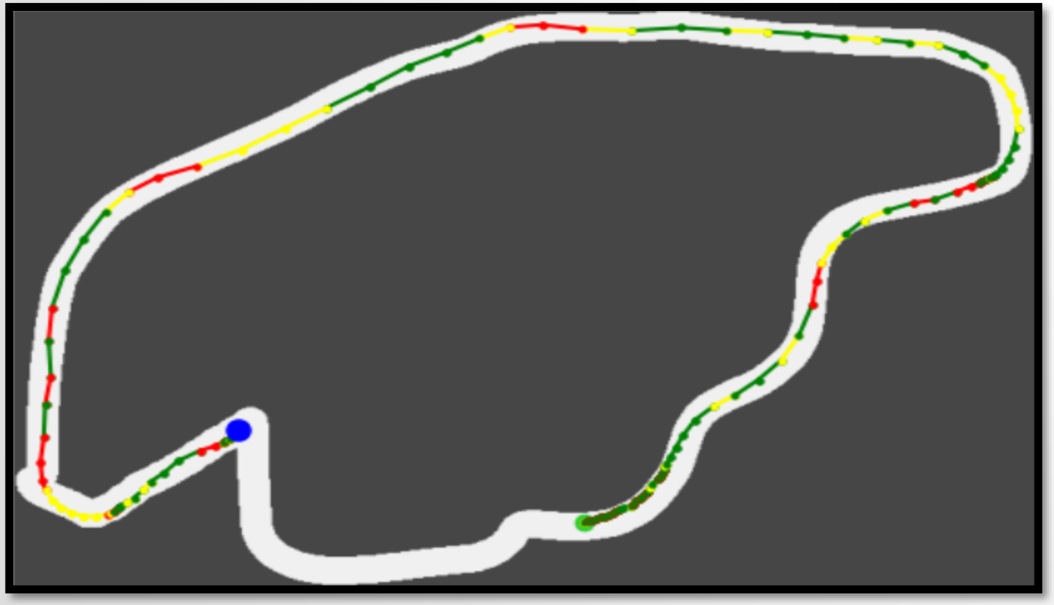
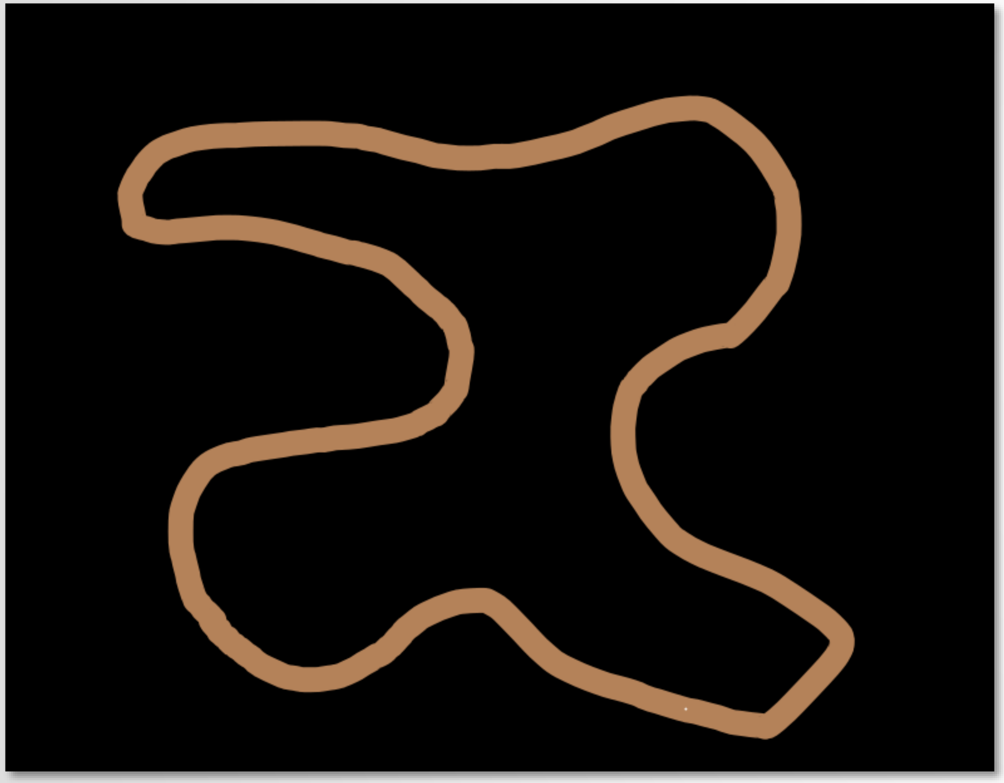


Table of content

1) Racing Car Environment

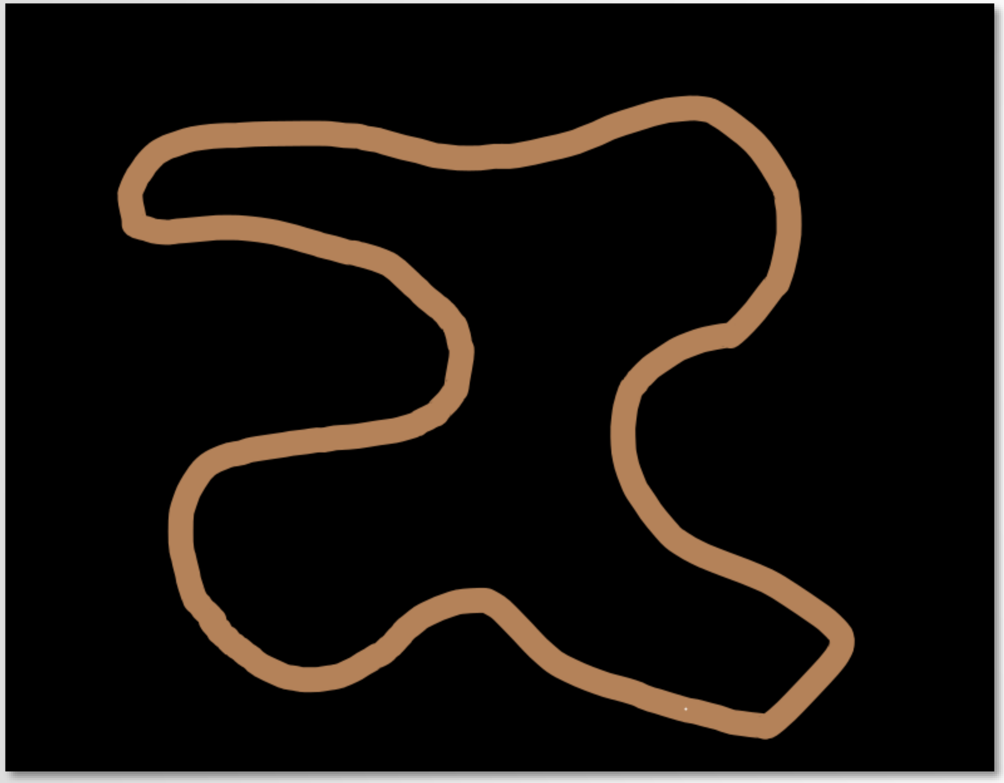
1) Tracks



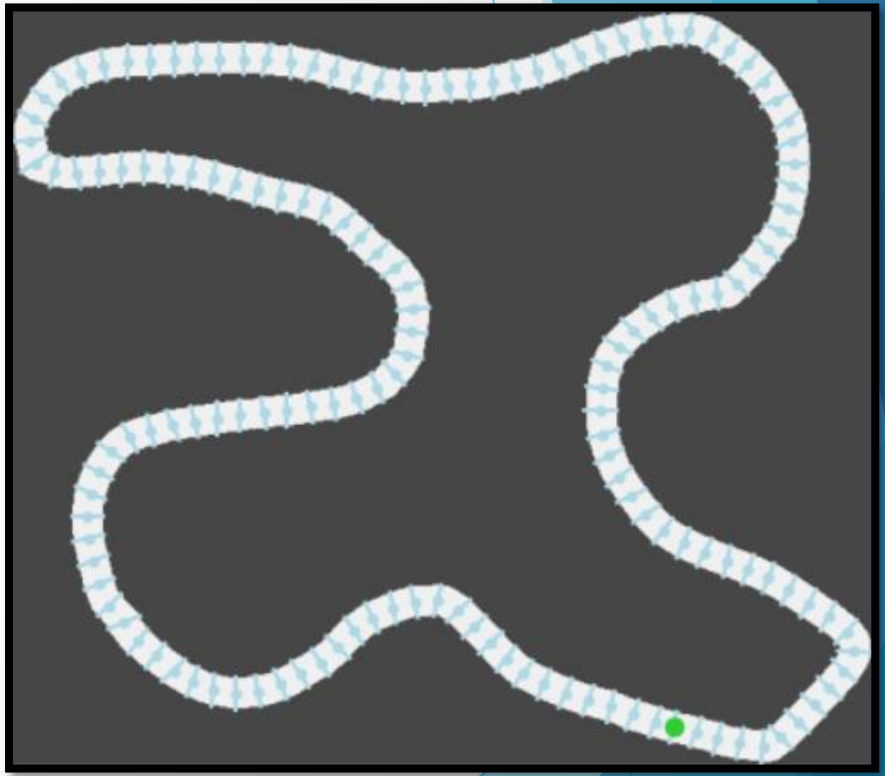
Track_06.png

1) Racing Car Environment

1) Tracks



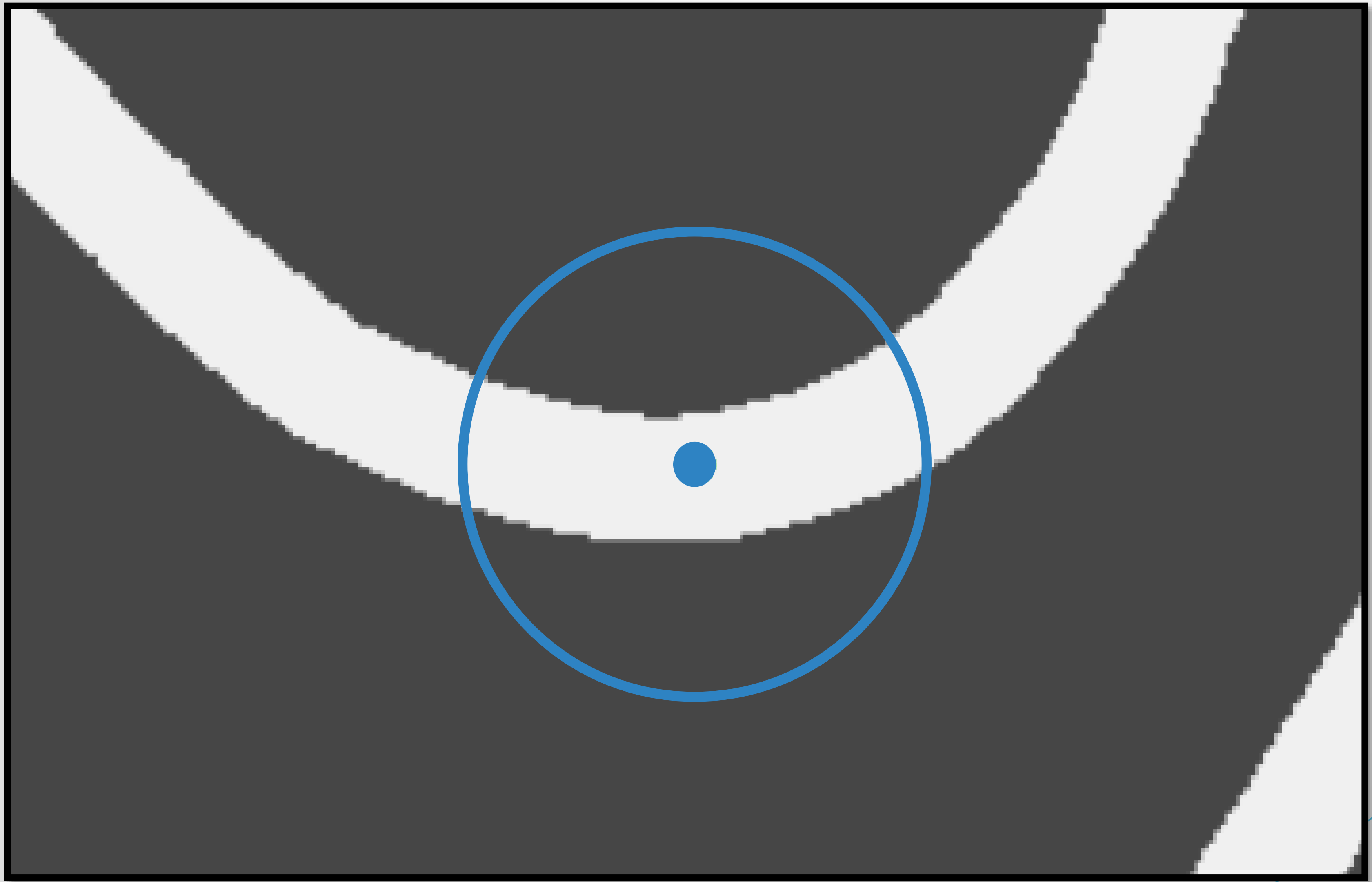
Track_06.png



Track_06_computed

1) Racing Car Environment

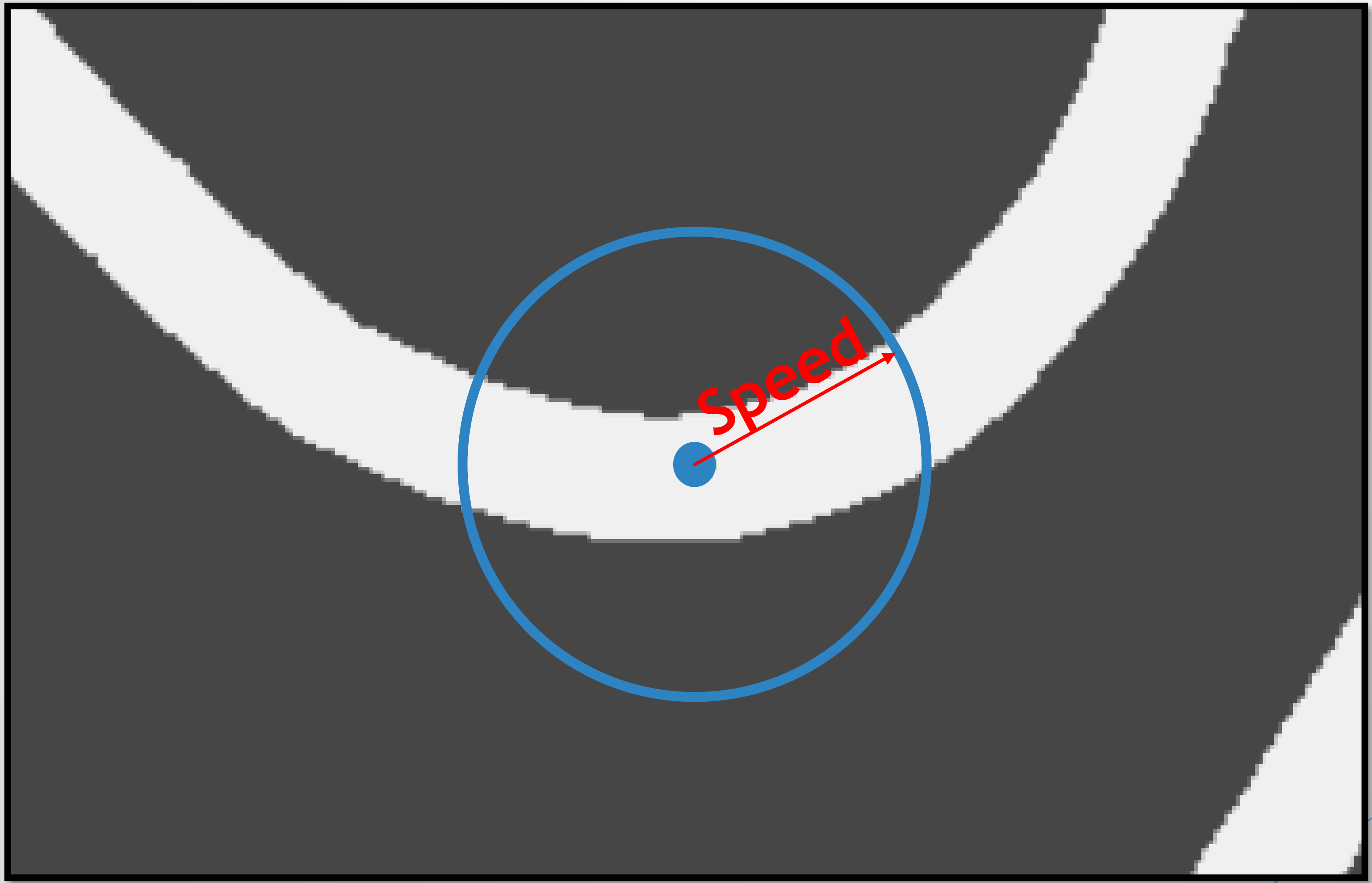
2) Car’s physics



Atributes:

1) Racing Car Environment

2) Car's physics

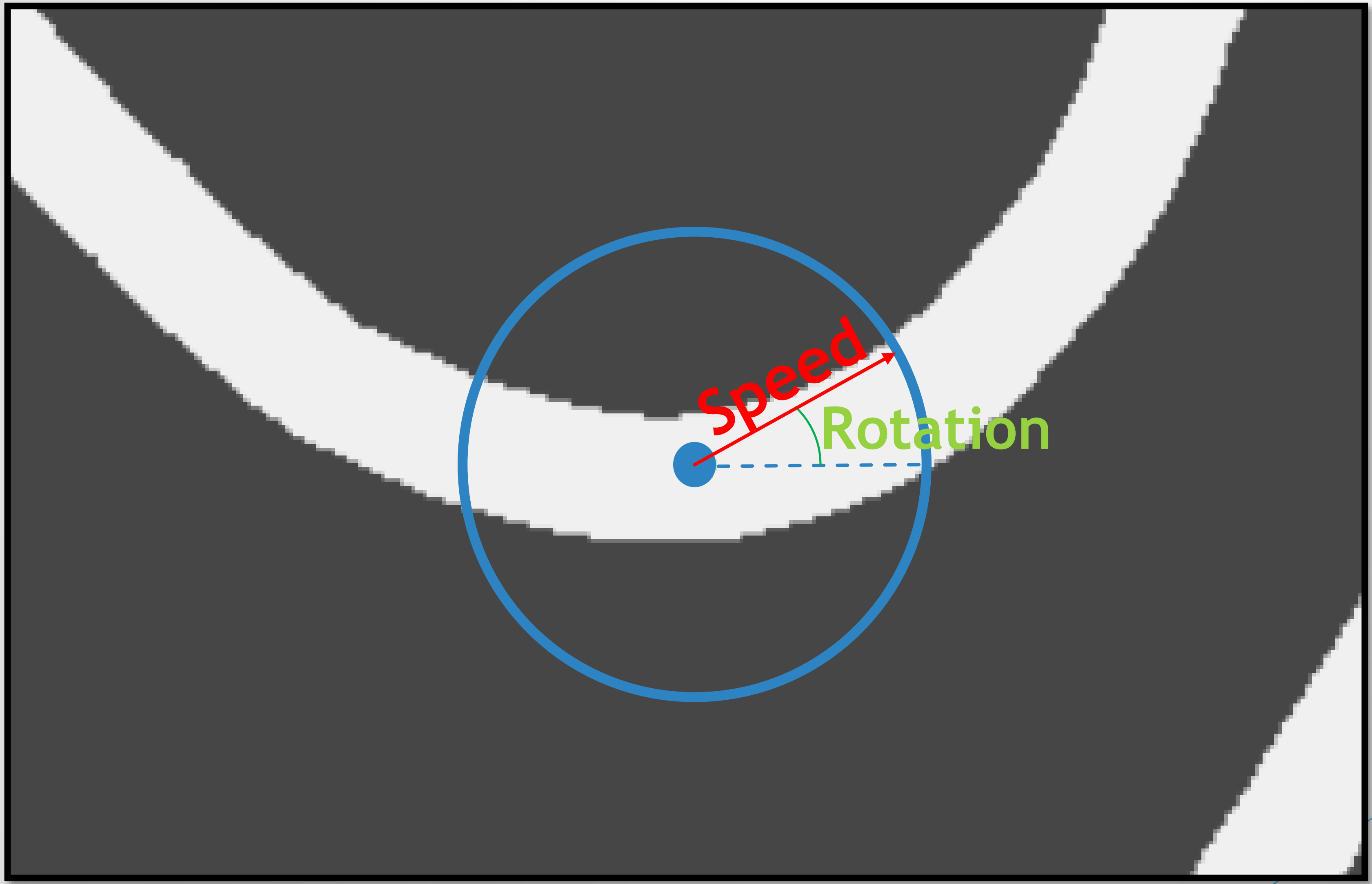


Atributes:

- Speed in [0, MaxSpeed]

1) Racing Car Environment

2) Car's physics

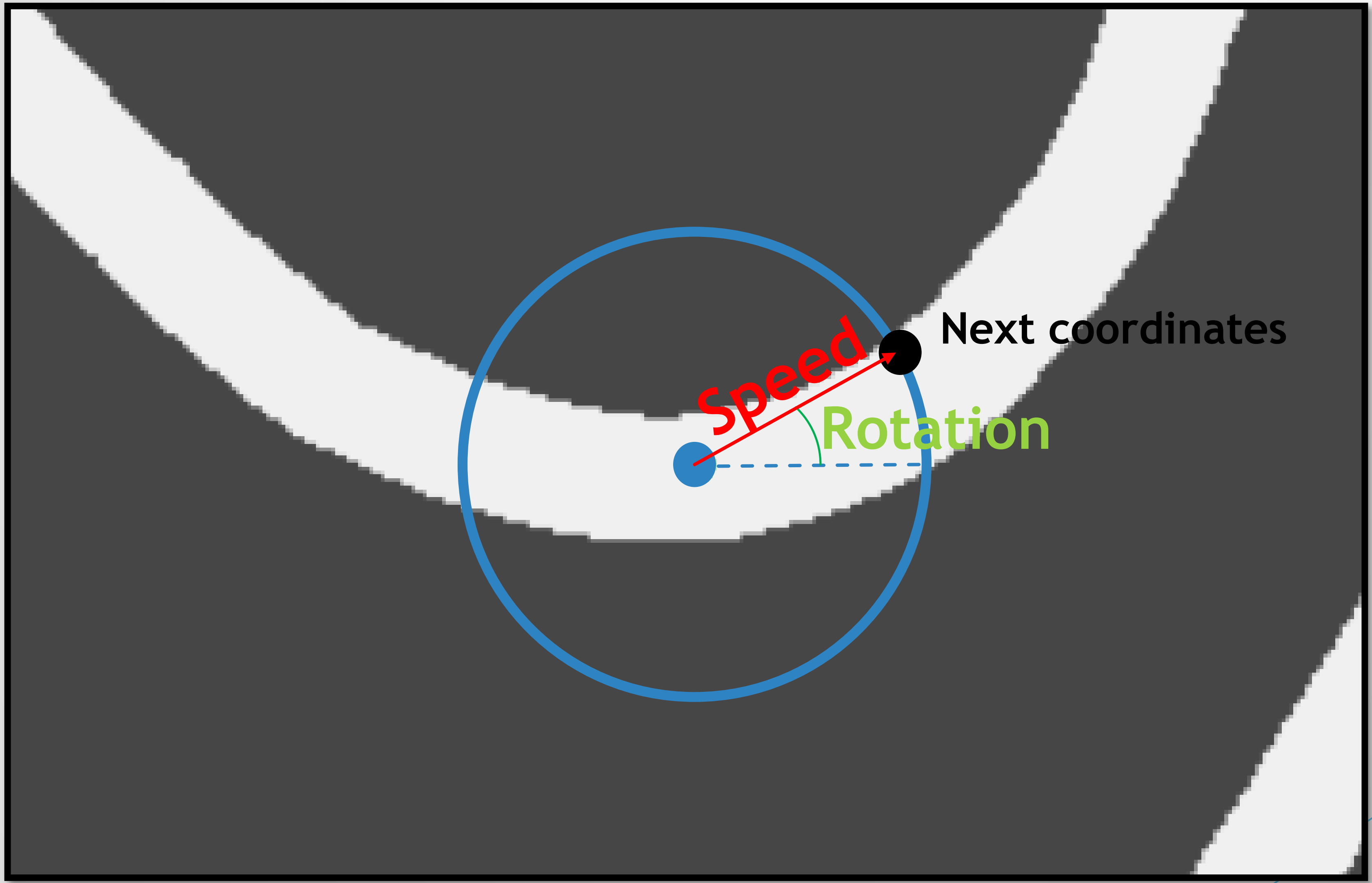


Atributes:

- Speed in [0, MaxSpeed]
- Rotation in [0, 360]

1) Racing Car Environment

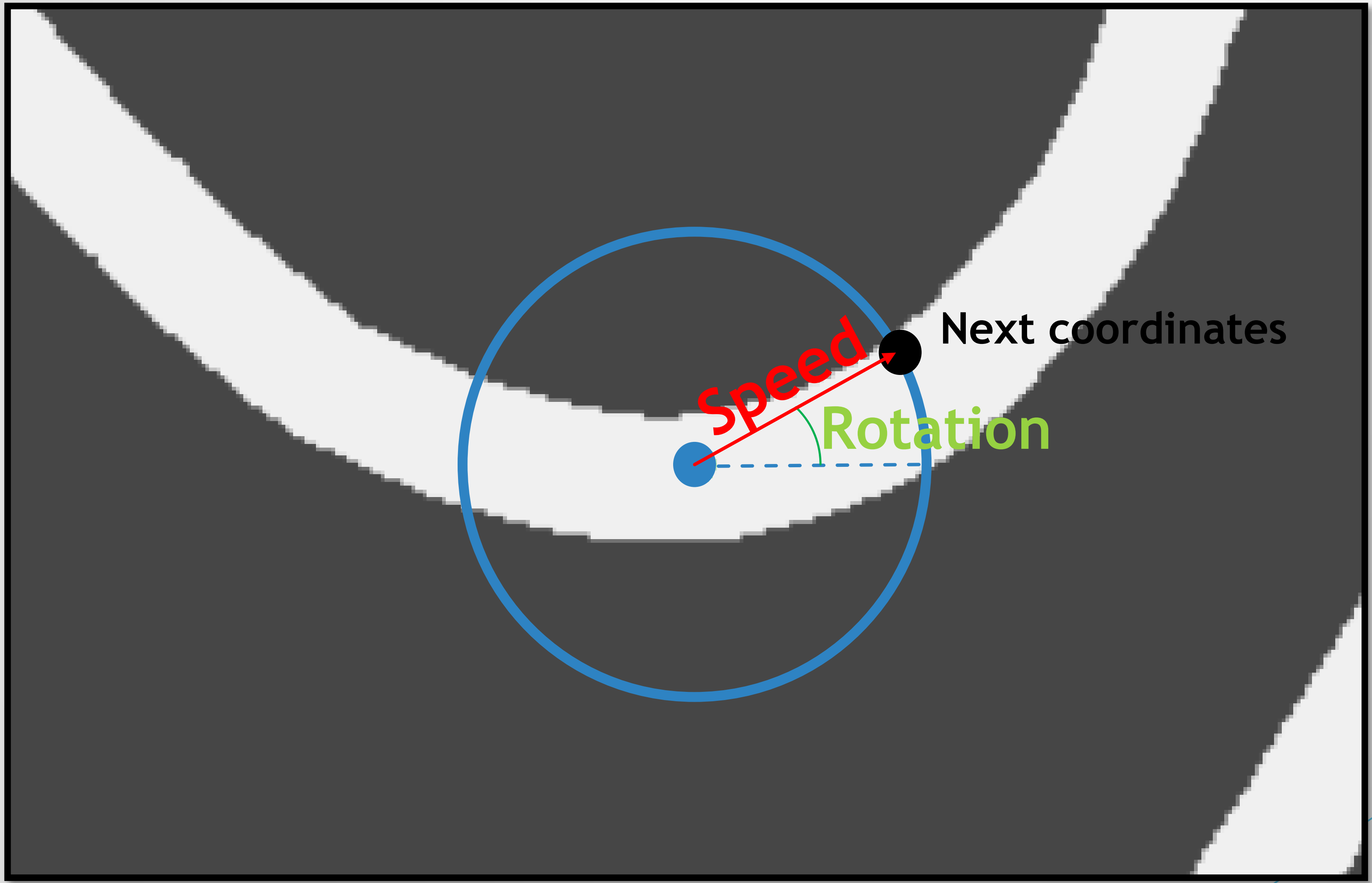
2) Car's physics



Atributes:
- Coordinate
- Speed in [0, MaxSpeed]
- Rotation in [0, 360]

1) Racing Car Environment

2) Car's physics

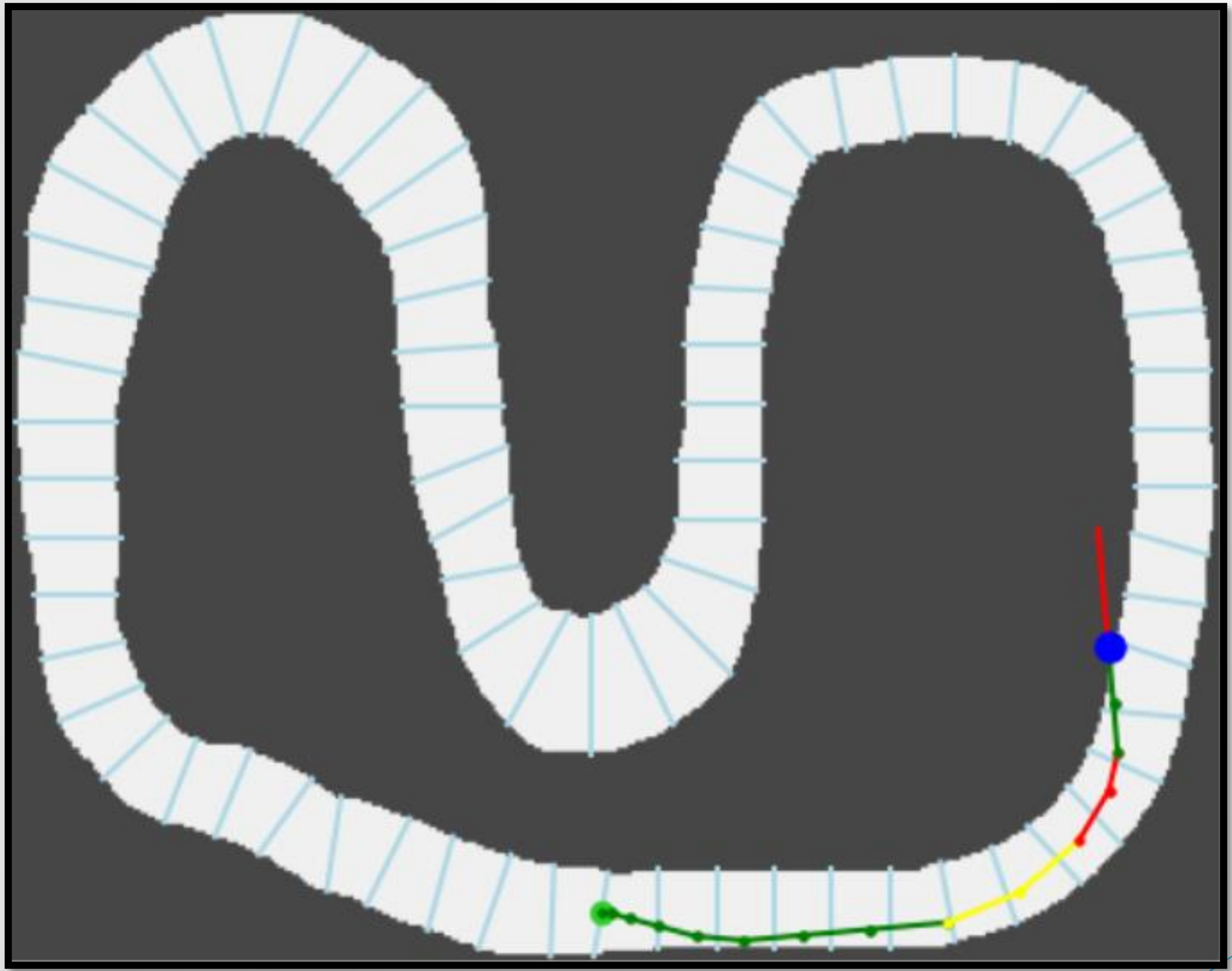


Atributes:
- Coordinate
- Speed in [0, MaxSpeed]
- Rotation in [0, 360]

Actions:
- Accelerate
- Brake
- Turn

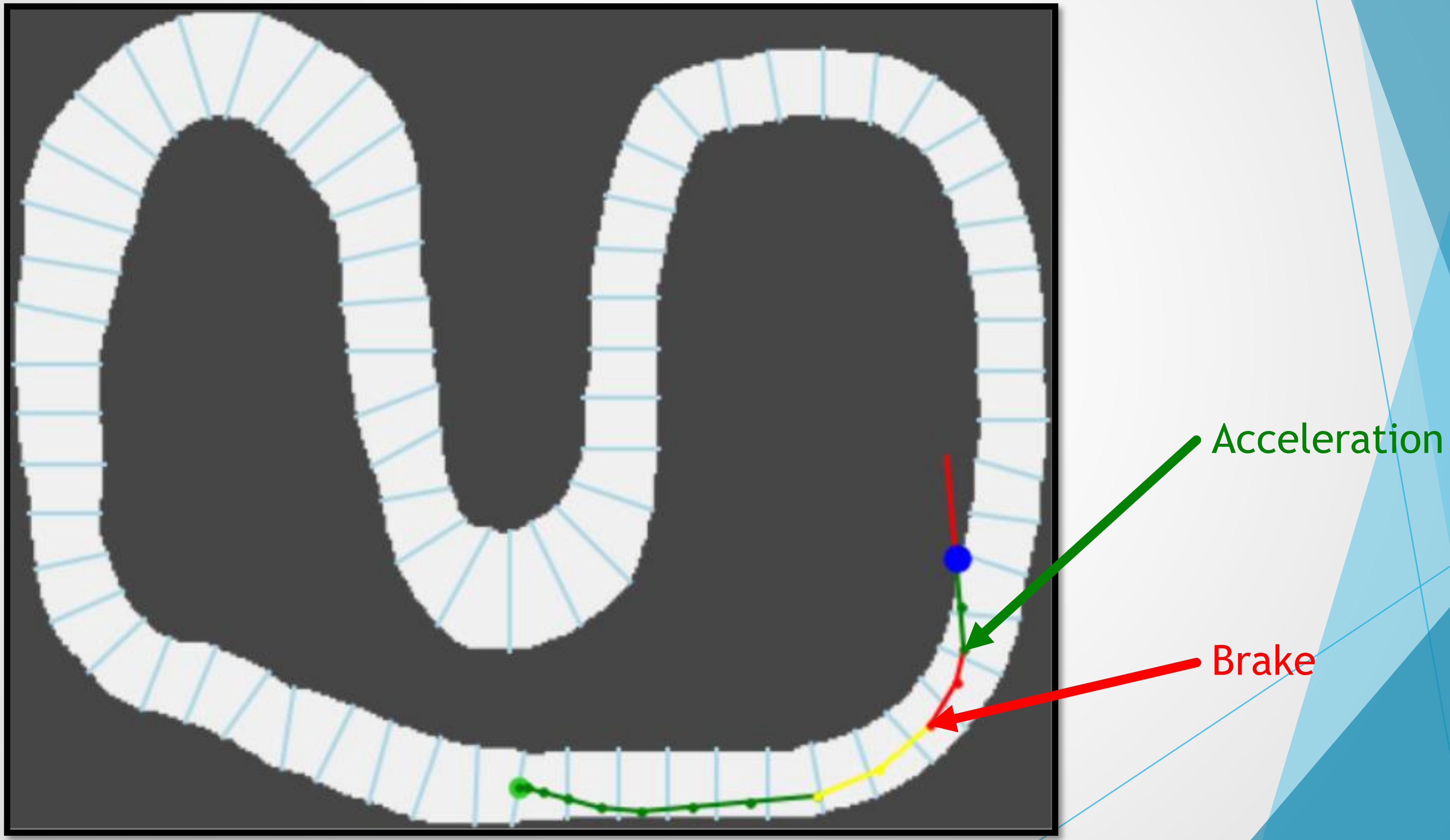
1) Racing Car Environment

3) Example



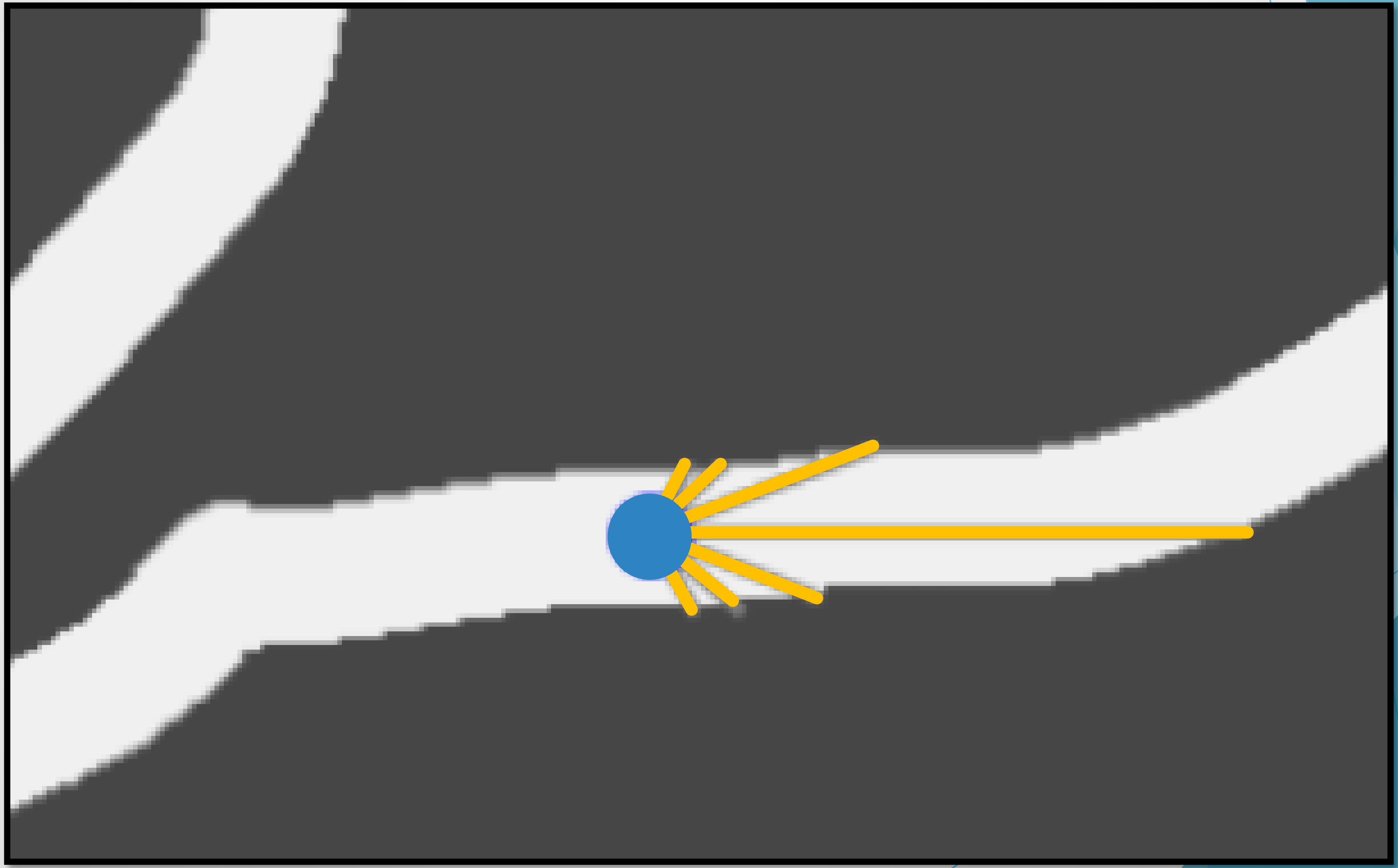
1) Racing Car Environment

3) Example



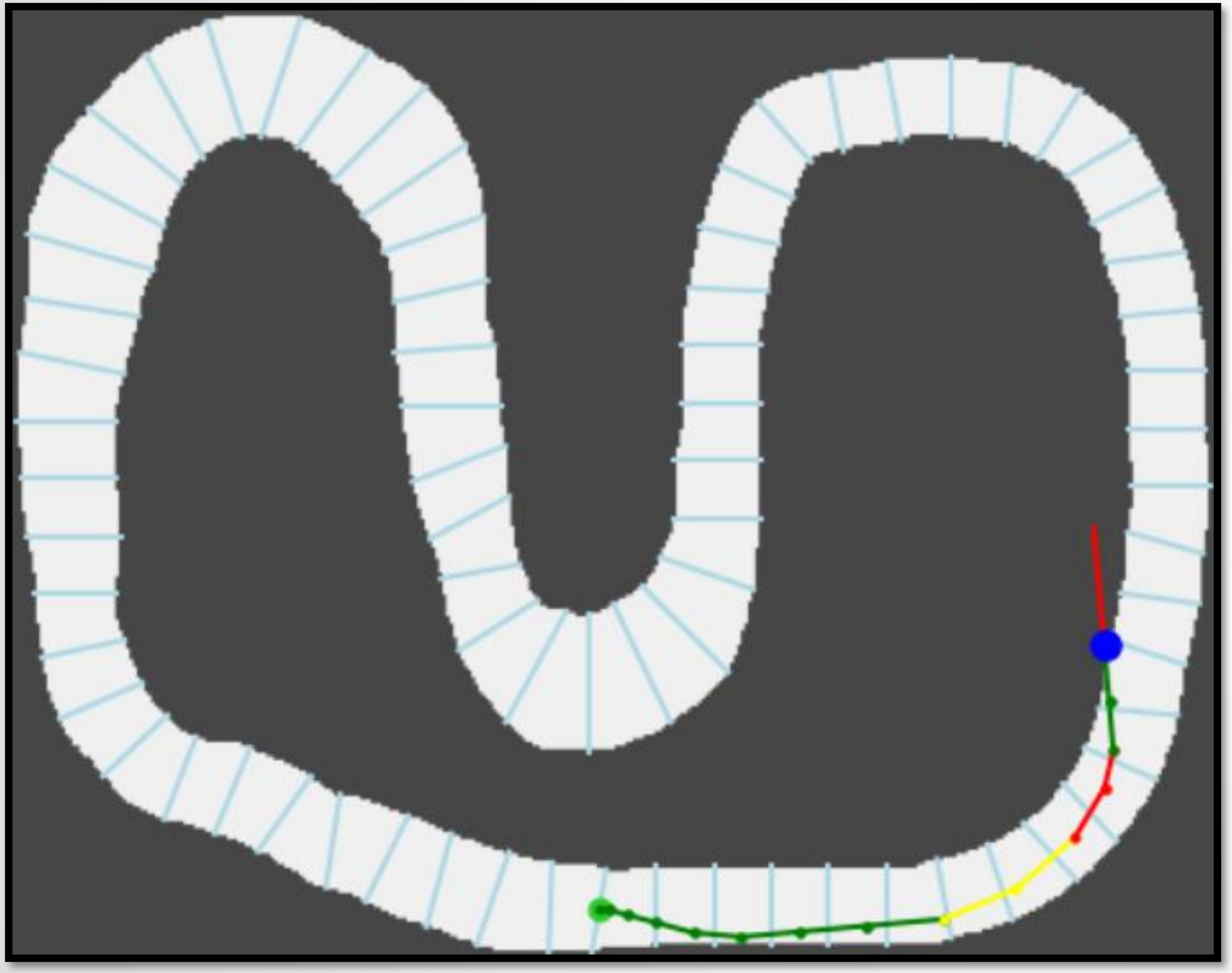
1) Racing Car Environment

4) Interaction with the environment



1) Racing Car Environment

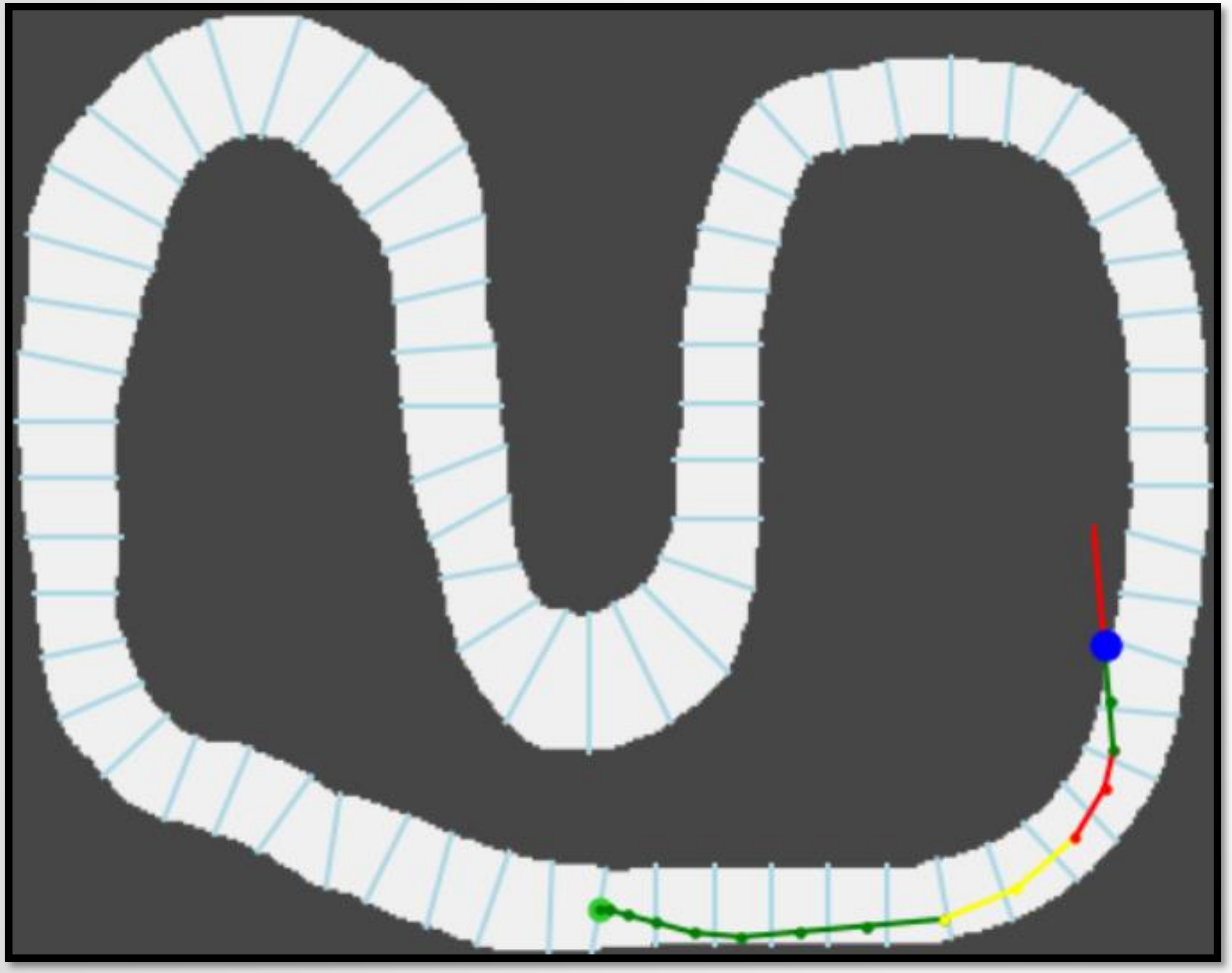
4) Reward



R = [10, 1, 2, 12, 13, 13, 13, 14, 24, 14, 13, 12, 13, -497]

1) Racing Car Environment

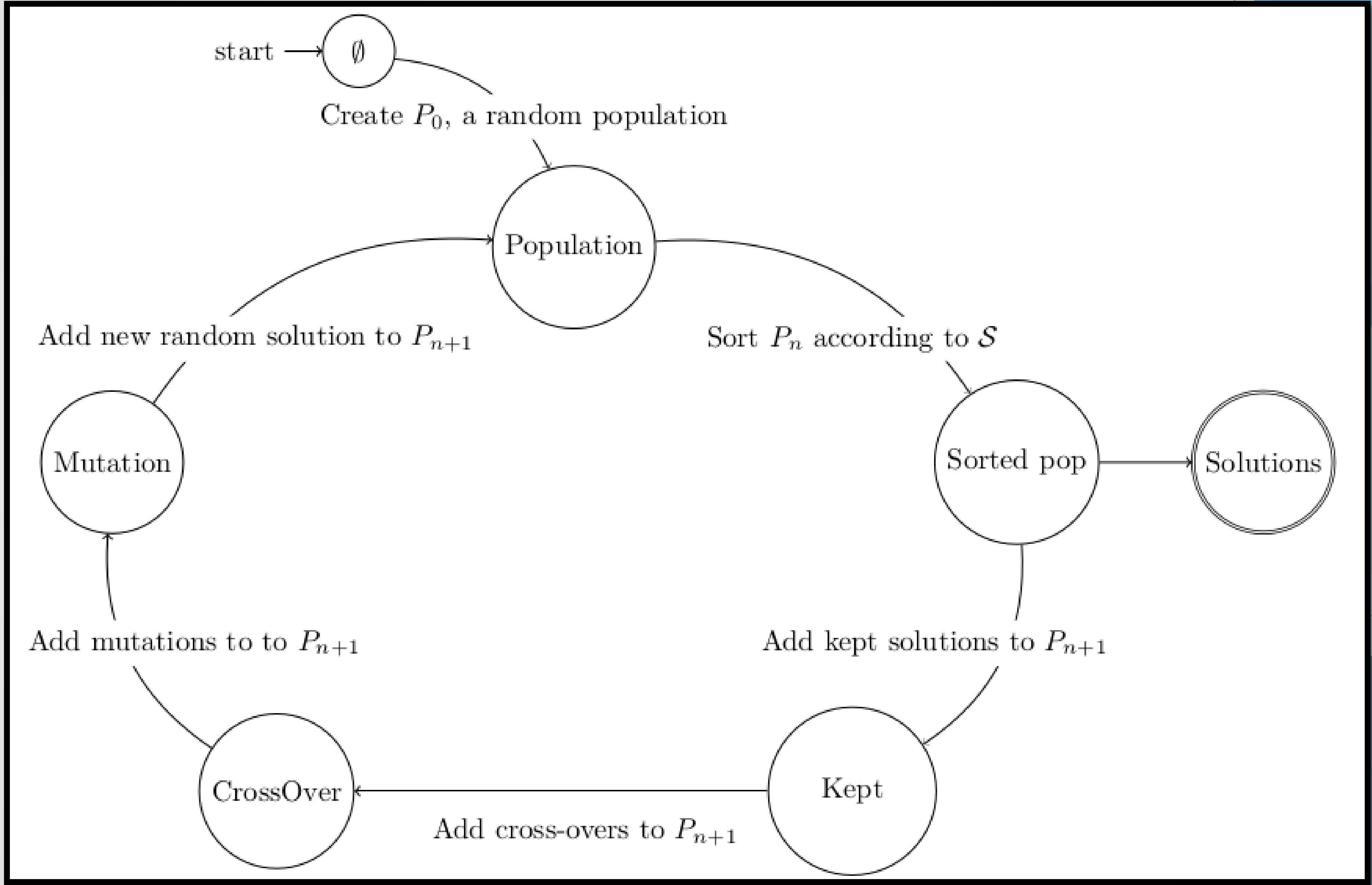
4) Reward



R = [10, 1, 2, 12, 13, 13, 13, 14, 24, 14, 13, 12, 13, -497]

Total reward: $\sum_{r \in R} r = -343$

2) Genetic Algorithm



3) Q-Learning

1) Formula

Score of a Policy π:

$$Q_\pi(s,a) = \sum_{i=0}^{+\infty} \gamma^i r_i$$

3) Q-Learning

1) Formula

Score of a Policy π:

$$Q_\pi(s,a) = \sum_{i=0}^{+\infty} \gamma^i r_i$$

Bellman Equation:

$$Q^*(s,a) = r + \gamma \max_{a' \in A} Q^*(s',a)$$

3) Q-Learning

1) Formula

Score of a Policy π:

$$Q_\pi(s,a) = \sum_{i=0}^{+\infty} \gamma^i r_i$$

Bellman Equation:

$$Q^*(s,a) = r + \gamma \max_{a' \in A} Q^*(s',a)$$

Error:

$$|Q_\pi(s,a) - (r + \gamma \max_{a' \in A} Q_\pi(s',a))|^2$$

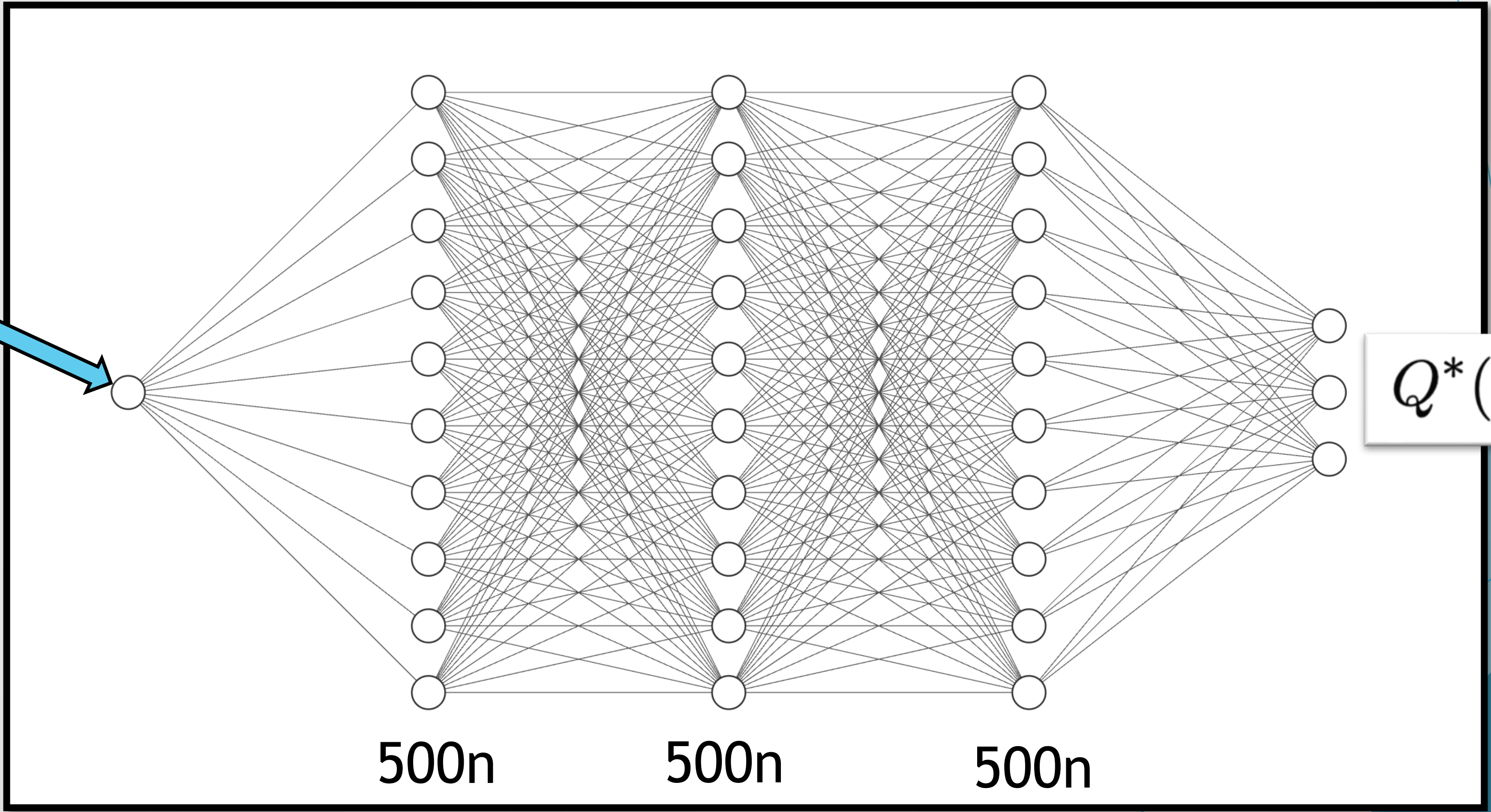
3) Q-Learning

2) Neural network

- Goal : Approximate Q^*

300000 Parameters

State of the Car



$Q^*(s, a)$

3) Q-Learning

3) High level workflow

- Choose an action
- Observe the reward
- Compute the error given by the Bellman equation.
- Change the model’s parameters to minimize this error.

- Remark: We use techniques to stabilize the training
  (mini-batches, ReplayBuffer, epsilon-greedy policy)

4) Simulation

1) Objectives

- Compare the different models used to train the car

- Evaluate the best hyper-parameters for the Deep Q model.

- Analyse the performance of the best train.

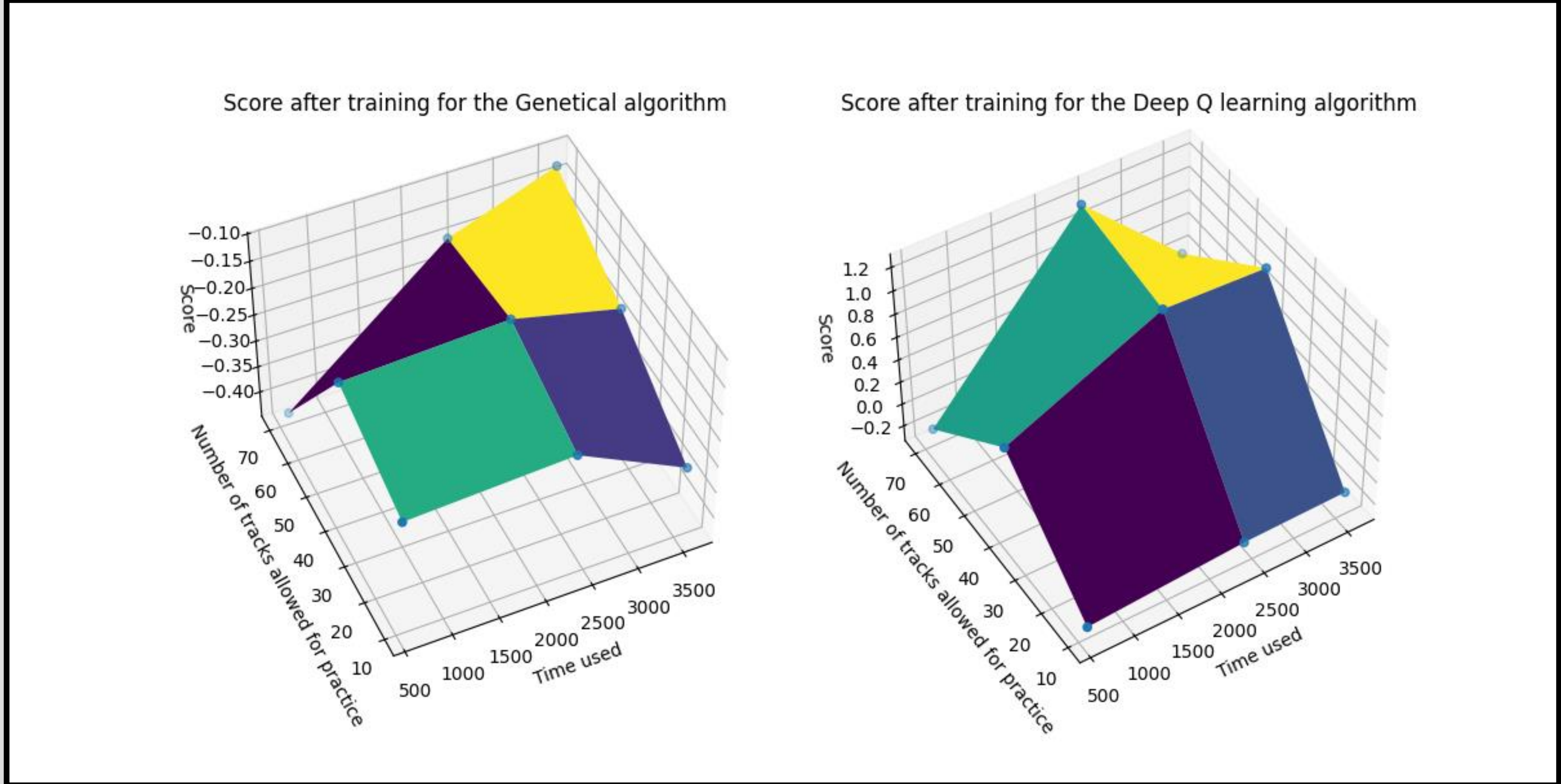
4) Simulation

2) How to compares the models?

- ✓ Same number of CPU/GPU (Grid5000)
- ✓ We chose to change the number of training tracks and the training time
- ✓ We chose to evaluate on the average reward and the volatility of the trained car

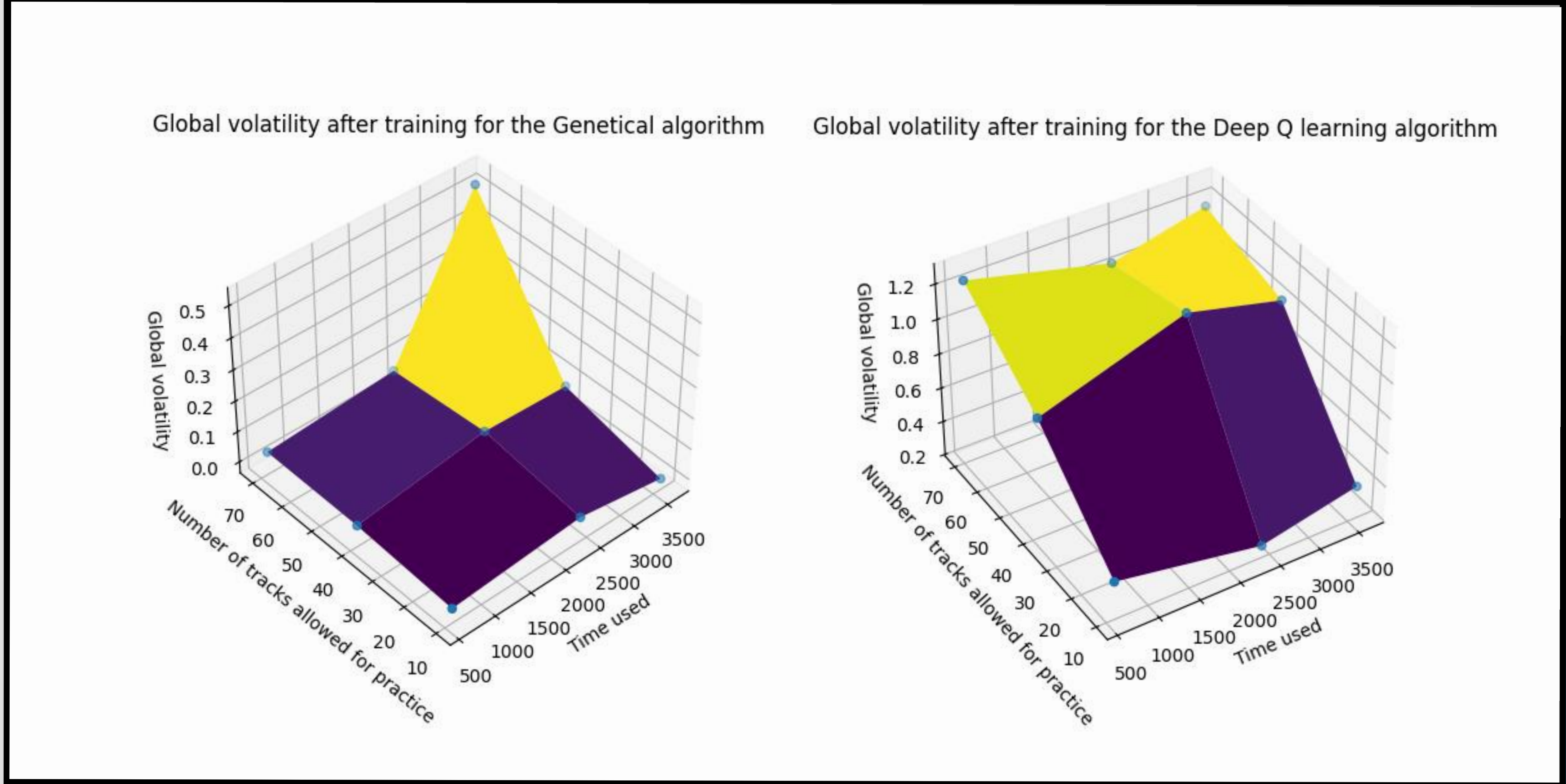
5) Evaluation

1) Q-Learning VS Genetic Algorithms: Reward



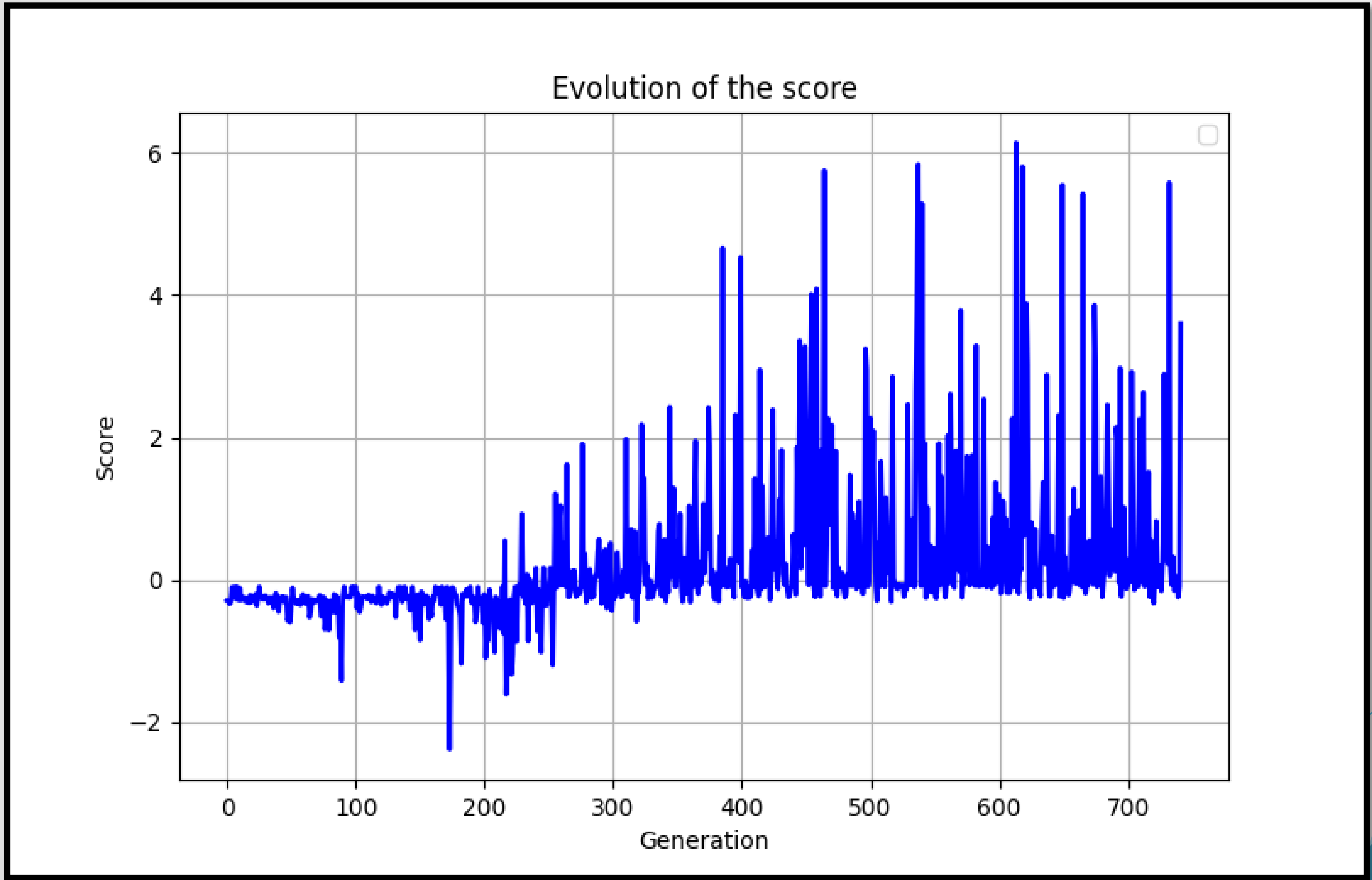
5) Evaluation

2) Q-Learning VS Genetic Algorithms: Volatility



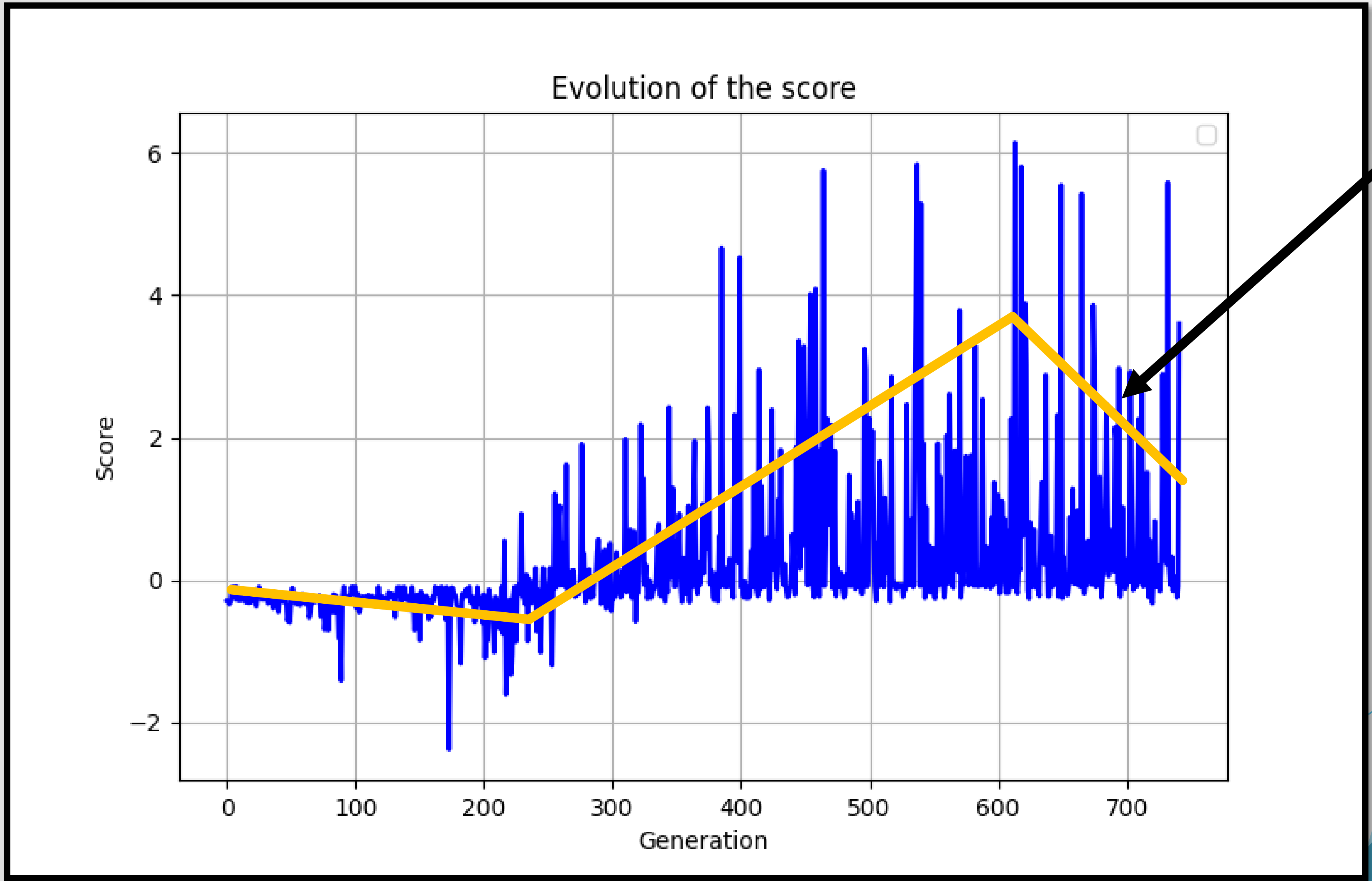
5) Evaluation

3) Q-Learning Rewards



5) Evaluation

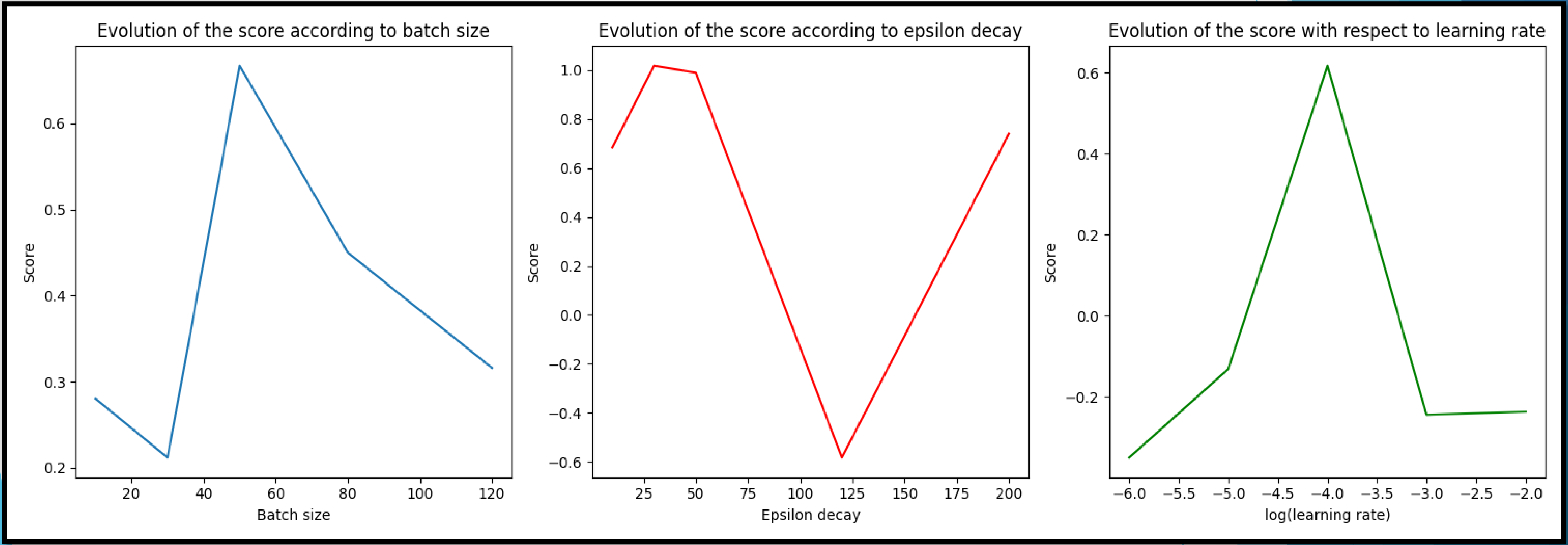
3) Q-Learning Rewards



Overfitting

5) Evaluation

4) Best hyper-parameters



6) Demonstration

7) Conclusion

Results:

- Deep Q-Learning outperforms Genetic Algorithms
- Impressive generalization of the model to new tracks

7) Conclusion

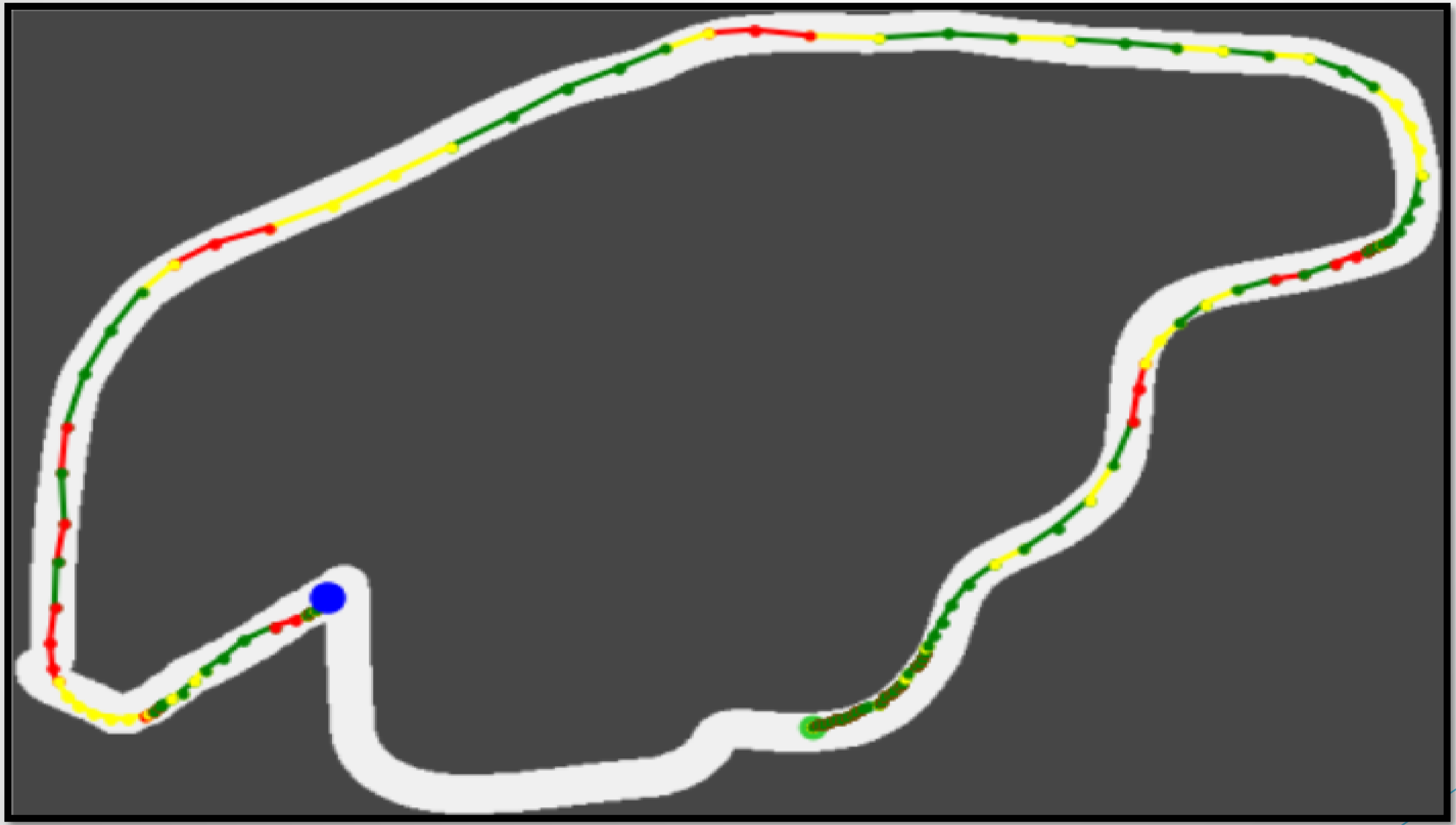
Results:

- Deep Q-Learning outperforms Genetic Algorithms
- Impressive generalization of the model to new tracks

Limits, Future Work:

- Explore the dependence on other hyperparameters
- Complexify the physics of the car
- Overfitting

Thanks you for your attention



Macéo Ottavy, Mathieu Longatte, Louison Mocq