

STAGE 2024

GARRONE GABRIEL

FONCTIONNEMENT DES
INTÉRACTIONS ENTRE
CARLA ET ROS

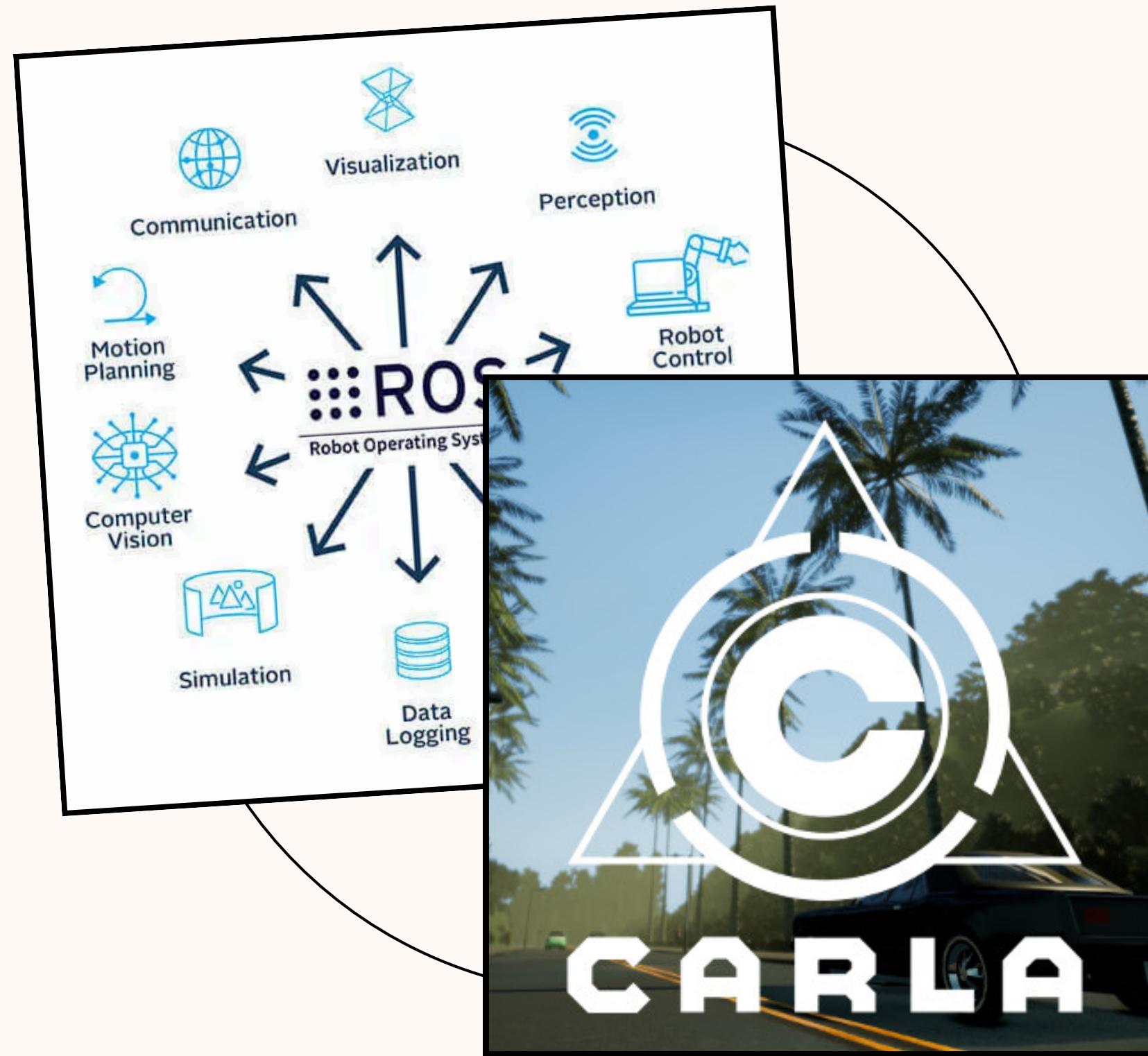
INTRODUCTION

L'intégration de CARLA avec ROS via le CARLA ROS Bridge offre aux développeurs et chercheurs un environnement réaliste et flexible pour tester et développer des applications robotiques.

Ce document explore en détail le fonctionnement de cette intégration, les types de données échangées et les applications potentielles, et la relation entre CARLA et ROS.

- ❖ [DOCUMENTATION D'INSTALLATION](#)
- ❖ [DOCUMENTATION D'UTILISATION](#)





SOMMAIRE

- ❖ VUE D'ENSEMBLE
- ❖ NŒUDS ET TOPICS
- ❖ INTÉGRATION DE CAPTEURS
- ❖ CONTRÔLE DES VÉHICULES

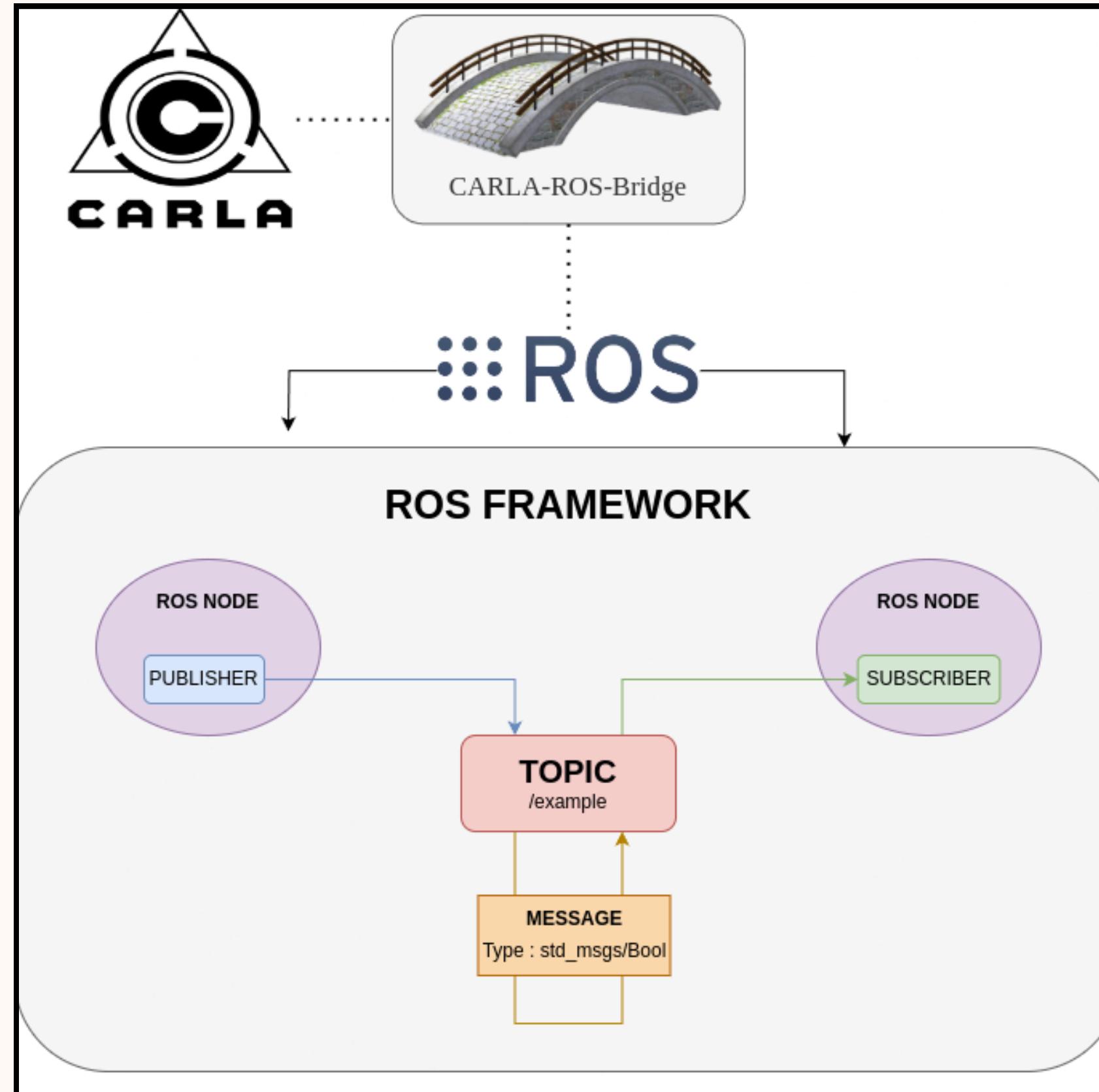
I - VUE D'ENSEMBLE

QU'EST CE QUE LE CARLA ROS BRIDGE ?

Le CARLA ROS Bridge est un composant logiciel qui établit une liaison entre CARLA, une plateforme de simulation de conduite autonome en environnement urbain, et ROS, le Robot Operating System, un framework de développement robotique largement utilisé. Concrètement, le bridge permet une communication bidirectionnelle entre CARLA et ROS, en facilitant l'échange de données et de commandes entre les deux systèmes.

Sur le plan technique, le CARLA ROS Bridge fonctionne en publiant et en écoutant des topics ROS spécifiques, qui servent de canaux de communication pour transmettre des informations telles que la position des véhicules, les données des capteurs et les commandes de contrôle. Les noeuds ROS, quant à eux, jouent un rôle crucial dans cette communication en tant qu'entités logicielles responsables de la publication et de la souscription aux topics, permettant ainsi aux différentes parties de l'écosystème ROS d'échanger des données et de coordonner leurs actions de manière efficace et cohérente.

II - NŒUDS ET TOPICS



MÉCANISME DE COMMUNICATION

Le CARLA ROS Bridge utilise un mécanisme de communication basé sur ROS, le Robot Operating System. Concrètement, ROS utilise un système de publication/souscription, où les différentes parties du système communiquent en publant des données sur des topics et en souscrivant à ces topics pour recevoir les données.

Le CARLA ROS Bridge crée des noeuds ROS qui publient des données sur des topics spécifiques, telles que la position des véhicules, les données des capteurs et les commandes de contrôle. D'autres noeuds ROS peuvent alors souscrire à ces topics pour recevoir les données publiées par CARLA ou pour envoyer des commandes de contrôle à CARLA.

UTILISATION DE NOEUDS ROS

Les noeuds ROS sont des entités logicielles qui exécutent des tâches spécifiques dans un système ROS. Ils peuvent être utilisés pour publier des données sur des topics, souscrire à des topics pour recevoir des données, ou fournir des services pour effectuer des actions spécifiques sur demande. On peut simplement lancer un noeud correspondant à une application à l'aide de la commande **rosrun** ou en utilisant des fichiers de lancement ROS. Par exemple, pour lancer le noeud qui lance le pont entre CARLA et ROS, on peut effectuer les données de capteurs provenant de CARLA, les développeurs peuvent exécuter une commande comme **rosrun carla_ros_bridge carla_ros_bridge.launch**.

Pour créer un nouveau noeud, il faudra définir les fonctionnalités du noeud, telles que les messages qu'il publie sur les topics, les topics auxquels il souscrit pour recevoir des données, ou les services qu'il fournit pour effectuer des actions spécifiques sur demande. On peut notamment créer son propre noeud via un script python. Une fois le code du noeud écrit, il est nécessaire de compiler le package ROS contenant le code du noeud à l'aide de l'outil **catkin_make** et lancer le noeud à l'aide de la commande **rosrun** ou en utilisant des fichiers de lancement ROS.

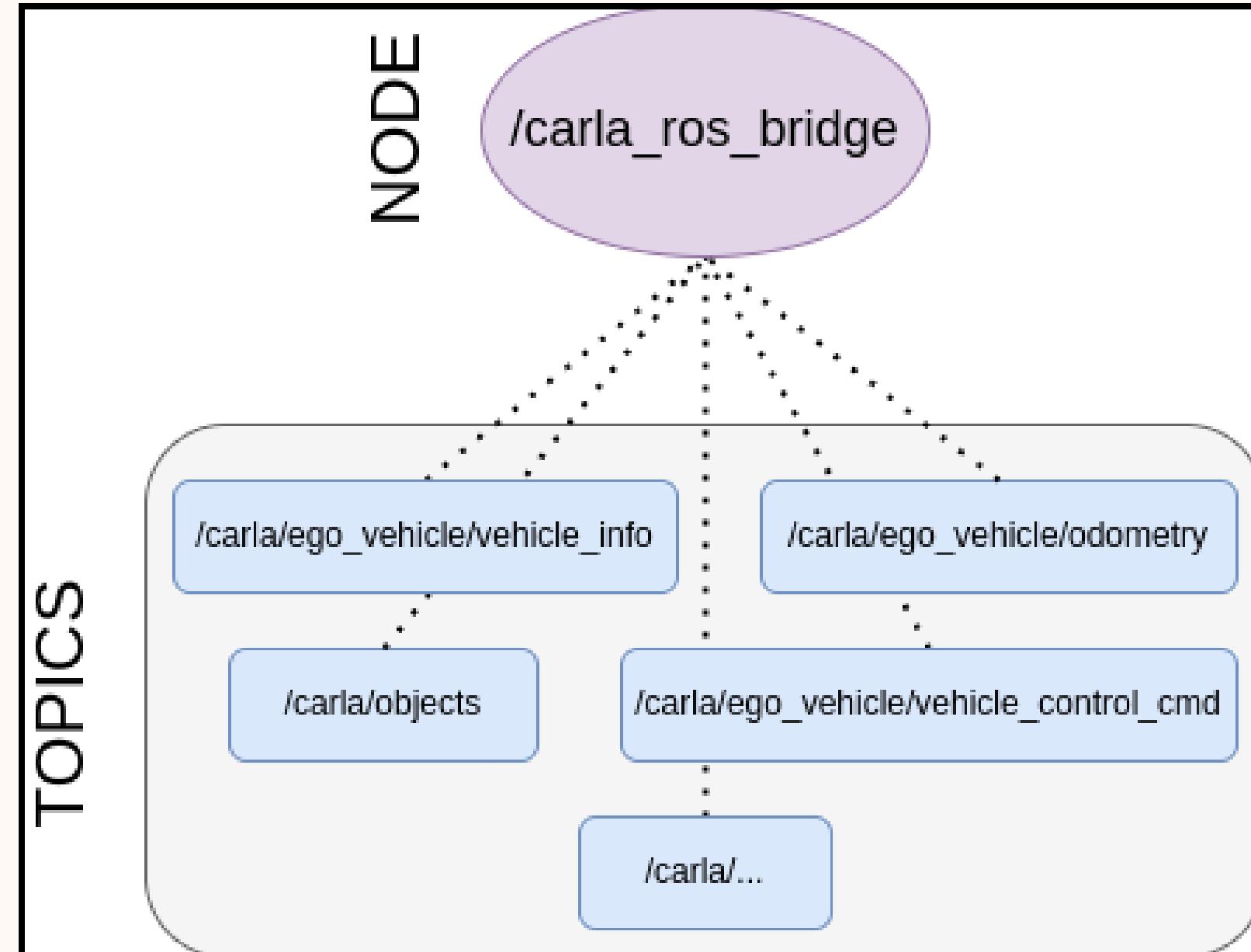
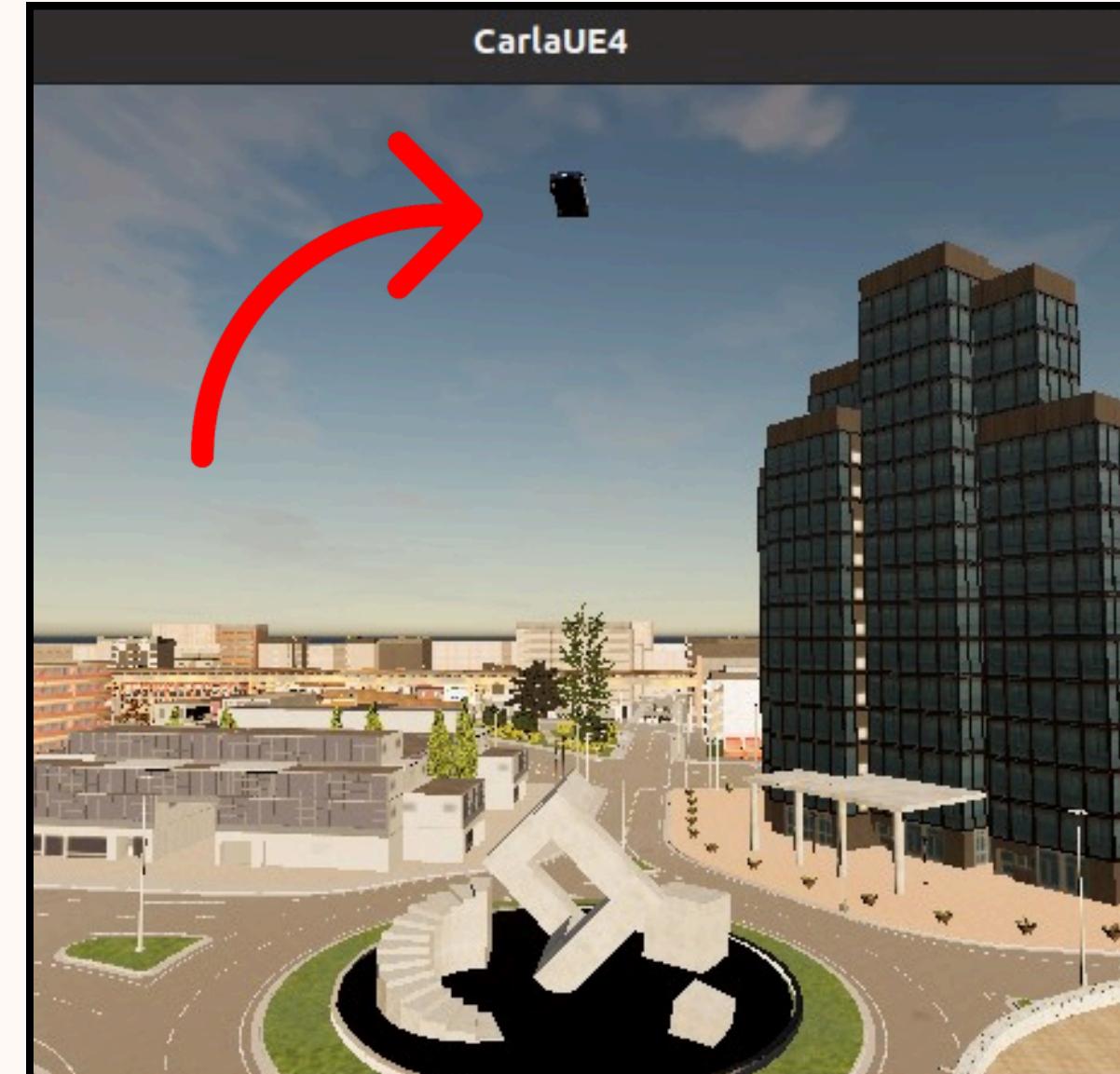


Schéma d'un noeud ROS et de plusieurs de ses topics associés

UTILISATION TOPICS ROS

Les topics ROS sont des canaux de communication utilisés pour transmettre des données entre les différents noeuds ROS. Dans le contexte du CARLA ROS Bridge, des topics sont utilisés pour transmettre des informations telles que la position des véhicules, les données des capteurs et les commandes de contrôle entre CARLA et ROS.

Par exemple, un noeud ROS dans CARLA pourrait publier la position d'un véhicule sur un topic, tandis qu'un autre noeud ROS dans ROS pourrait souscrire à ce topic pour recevoir la position du véhicule et effectuer des calculs ou prendre des décisions en fonction de cette information.



Apparition d'un véhicule via un service

UTILISATION DES SERVICES ROS

En plus de l'utilisation de topics, le CARLA ROS Bridge utilise également des services ROS pour permettre l'interaction entre CARLA et ROS. Les services ROS sont des fonctions appelables qui fournissent des fonctionnalités spécifiques sur demande.

Dans le cadre du CARLA ROS Bridge, des services peuvent être utilisés pour effectuer des actions telles que le contrôle des véhicules, la modification de l'environnement de simulation ou la récupération de données spécifiques à partir de CARLA. On peut alors appeler ces services depuis des noeuds ROS pour interagir avec CARLA de manière dynamique et flexible.

III - INTÉGRATION DES CAPTEURS

LES CAPTEURS AU GLOBAL

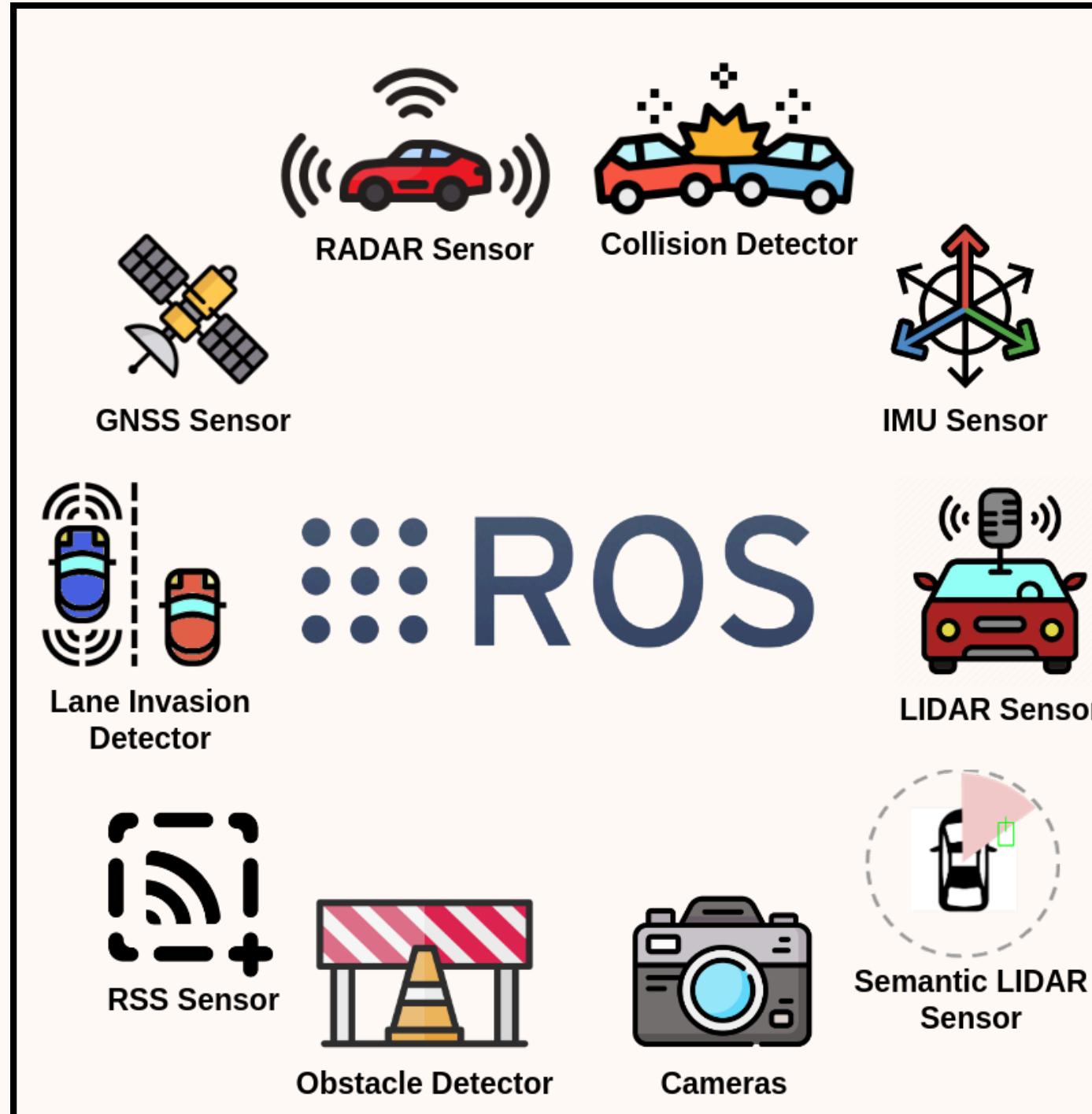


Schéma des capteurs de CARLA intégrés dans ROS

Dans CARLA, une variété de capteurs virtuels tels que des caméras, des LIDAR et des capteurs de profondeur sont disponibles pour collecter des données sur l'environnement de simulation. Ces capteurs sont cruciaux pour la perception et la navigation des véhicules autonomes dans CARLA.

Grâce au CARLA ROS Bridge, ces capteurs virtuels peuvent être facilement intégrés dans des applications ROS. Par exemple, les images capturées par les caméras virtuelles peuvent être publiées sur des topics ROS pour être utilisées dans des algorithmes de vision par ordinateur ou de traitement d'image, tandis que les données des LIDAR peuvent être transmises à ROS pour effectuer des opérations de cartographie ou de localisation.

UTILISATION DES CAPTEURS

Pour utiliser les capteurs virtuels de CARLA avec le CARLA ROS Bridge, il est nécessaire de configurer correctement ces capteurs dans l'environnement de simulation. Cela peut inclure la définition de paramètres tels que la résolution, le champ de vision, la fréquence d'échantillonnage, des caméras virtuelles, la spécification de la portée, de la résolution angulaire, de la fréquence de balayage, et également des capteurs LIDAR. Ces configurations peuvent être effectuées à l'aide de scripts Python dans CARLA ou à travers l'interface graphique de configuration de CARLA.

Une fois que les capteurs virtuels sont correctement configurés dans CARLA, le CARLA ROS Bridge se charge de transmettre les données de ces capteurs à ROS via des topics spécifiques. Par exemple, les images capturées par les caméras virtuelles sont publiées sur des topics ROS dédiés aux images RGB, de profondeur ou de segmentation sémantique. De même, les données des capteurs LIDAR sont transmises via des topics appropriés.

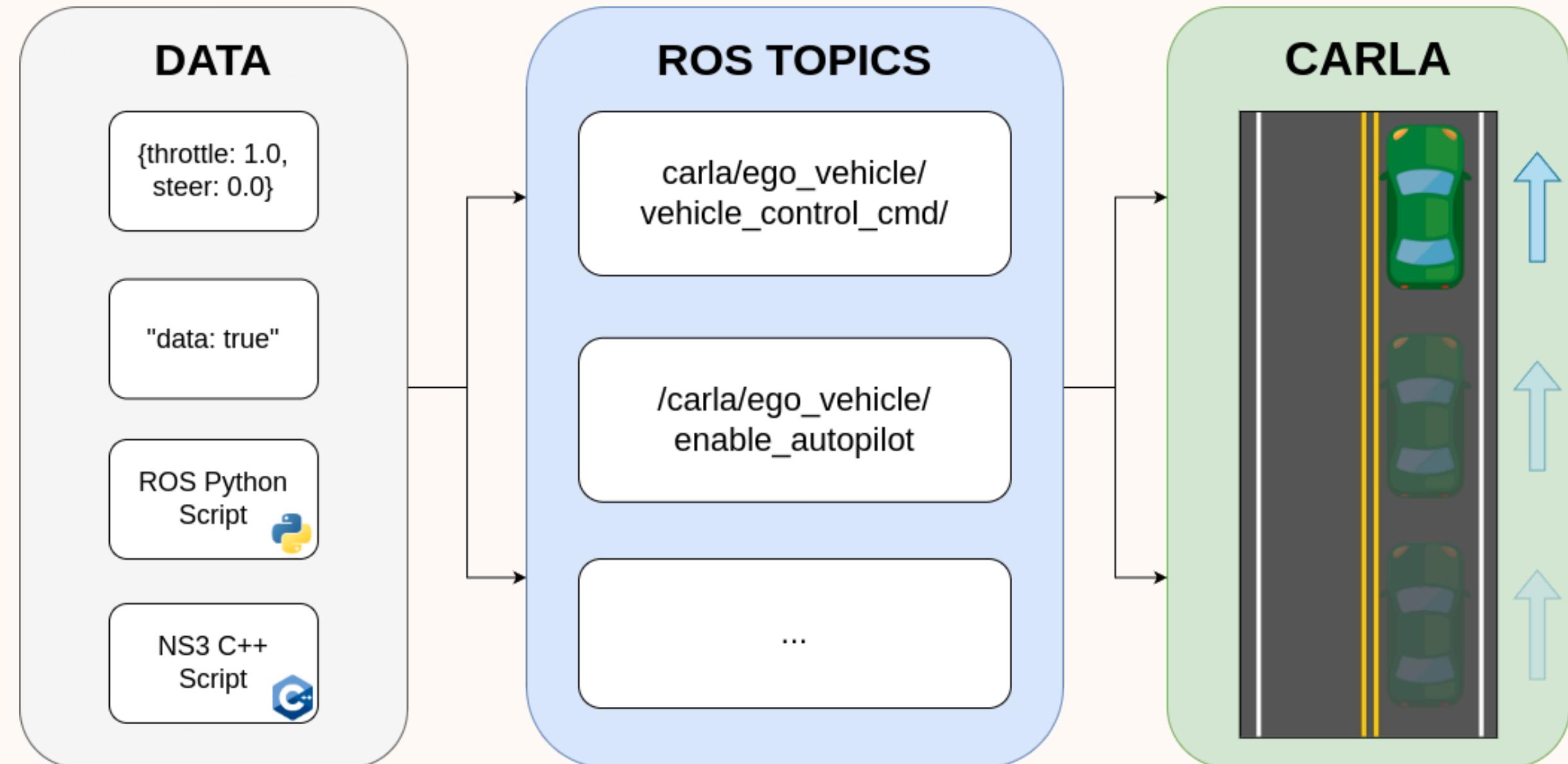
Dans ROS, on peut souscrire à ces topics pour recevoir les données des capteurs et les traiter à l'aide d'algorithmes de perception, de cartographie, de localisation, etc. Cette transmission et ce traitement des données de capteurs dans ROS permettent notamment un analyse significativement meilleure de l'environnement par les véhicules autonomes.

IV - CONTRÔLE DE VÉHICULES

CONTRÔLER VIA LES TOPICS ROS

Il est possible d'utilisation des topics ROS pour contrôler les véhicules dans CARLA. Grâce au CARLA ROS Bridge, on peut publier des commandes de contrôle sur des topics spécifiques, tels que la vitesse, la direction, l'accélération, etc.

Ces commandes sont ensuite transmises à CARLA, où elles sont interprétées par les véhicules simulés pour ajuster leur comportement en temps réel.



APPLICATION N°1 - CARLA SPAWN OBJECT

Le script **carla_spawn_object** est très utile car il permet de générer et de configurer dynamiquement des objets dans l'environnement de simulation CARLA.

Ce script prend en entrée des fichiers JSON qui décrivent les propriétés et les paramètres des objets à créer, tels que leur type, leur position, leur orientation, etc.

Ces fichiers JSON permettent ainsi de spécifier diverses configurations pour les objets, dont l'ajout de capteurs tels que des caméras ou des LIDAR, la définition de trajectoires ou de comportements spécifiques, les positions des objets, et bien plus encore.

```
{  
  "objects":  
  [  
    {  
      "type": "<SENSOR-TYPE>",  
      "id": "<NAME>",  
      "spawn_point": {"x": 0.0, "y": 0.0, "z": 0.0, "roll": 0.0, "pitch": 0.0, "yaw": 0.0},  
      <ADDITIONAL-SENSOR-ATTRIBUTES>  
    },  
    {  
      "type": "<VEHICLE-TYPE>",  
      "id": "<VEHICLE-NAME>",  
      "spawn_point": {"x": -11.1, "y": 138.5, "z": 0.2, "roll": 0.0, "pitch": 0.0, "yaw": -178.7},  
      "sensors":  
        [  
          <SENSORS-TO-ATTACH-TO-VEHICLE>  
        ]  
    }  
    ...  
  ]  
}
```

Exemple type d'un fichier JSON de configuration

APPLICATION N°2 - PUBLICATION SUR UN TOPIC ROS

```
rostopic pub /carla/ego_vehicle/enable_autopilot std_msgs/Bool "data: true"
```

Pour rappel la partie “data: true:” sont les données qu'on publie sur le topic concerné à savoir `/carla/ego_vehicle/enable_autopilot`. Ces données sont également associé au type “`std_msgs/Bool`” : `std_msgs` contient des types de messages courants représentant des types de données primitifs et d'autres constructions de messages de base, dont les booléens.

Le type de message de ce topic est donc booléen (vérifiable avec la commande “`rostopic info`”). Dans notre cas on veut activer le pilote automatique, alors on renseigne « `true` ».

Une fois le mode autopilote activé les capteurs sont essentiels pour permettre au véhicule de fonctionner. En effet les capteurs vont collecter des informations sur l'environnement et permettent au véhicule de prendre des décisions de conduite autonome en fonction de ces données.

```
#!/usr/bin/env python3
import rospy
from carla_msgs.msg import CarlaEgoVehicleControl

def publisher_cmd():
    rospy.init_node('publisher_cmd', anonymous=True)
    pub = rospy.Publisher('/carla/ego_vehicle/vehicle_control_cmd',
                          CarlaEgoVehicleControl, queue_size=10)
    rate = rospy.Rate(10) # 10 Hz

    while not rospy.is_shutdown():
        control_cmd = CarlaEgoVehicleControl()
        control_cmd.throttle = 1.0
        control_cmd.steer = 0.0
        pub.publish(control_cmd)
        rate.sleep()
```

Exemple d'un script python “publisher”

Ce script python publie sur un topic ROS de contrôle de véhicule à savoir `/carla/ego_vehicle/vehicle_control_cmd`.

On envoie la commande de contrôle “`throttle = 1.0`”, qui va faire avancer un véhicule tout droit.