**Concepts clés du projet Autonomous Vision driven LIMO**

Ce document présente de façon détaillée, exhaustive et sourcée les cinq briques conceptuelles exploitées dans notre projet :

1. **Initiation à Gazebo**
2. **URDF (Unified Robot Description Format)**
3. **Fichiers de lancement ROS 2 (Launch)**
4. **Programmation Python**
5. **Programmation orientée objet (POO) en Python**

**1. Initiation à Gazebo**

Gazebo est un simulateur 3D open-source majeur en robotique, développé initialement par l’équipe de l’ETH et Stanford et aujourd’hui maintenu par Open Robotics. Il permet de tester des algorithmes de contrôle, de perception et de navigation sans risquer le matériel réel. Gazebo intègre plusieurs moteurs physiques (ODE, Bullet, DART) et un moteur de rendu graphique OGRE, garantissant à la fois réalisme et performance ([classic.gazebosim.org](https://classic.gazebosim.org/tutorials?tut=guided_b1&utm_source=chatgpt.com), [gazebosim.org](https://gazebosim.org/docs/latest/getstarted/?utm_source=chatgpt.com)).

Les trois composants fondamentaux de Gazebo sont :

* **World** : décrit l’environnement simulé (terrain, obstacles, propriétés physiques).
* **Model** : englobe robots et objets, définis via SDF ou URDF/XACRO.
* **Plugin** : DLLs C++ chargées par Gazebo pour étendre les fonctionnalités (capteurs, contrôleurs, interactions) ([classic.gazebosim.org](https://classic.gazebosim.org/tutorials?tut=guided_b1&utm_source=chatgpt.com), [articulatedrobotics.xyz](https://articulatedrobotics.xyz/tutorials/ready-for-ros/gazebo/?utm_source=chatgpt.com)).

L’**intégration ROS 2** s’effectue grâce au package gazebo\_ros, qui synchronise l’horloge de simulation (/clock), offre des services de spawn/delete d’entités et mappe directement les capteurs simulés vers des topics ROS standard tels que /scan (LiDAR) ou /camera/image\_raw (caméra) ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html?utm_source=chatgpt.com), [docs.ros.org](https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo/Gazebo.html?utm_source=chatgpt.com)). Cette passerelle garantit que les mêmes nœuds ROS utilisés sur le robot réel fonctionnent sans modification majeure en simulation.

**2. URDF (Unified Robot Description Format)**

L’URDF est un format XML officiel de ROS pour décrire la structure cinématique et les propriétés visuelles/dynamiques d’un robot. Il définit :

* **Links** (liens) : volumes rigides avec inertie, collision et visuel.
* **Joints** : connexions articulées (fixed, revolute, continuous, prismatic), décrivant degrés de liberté et coordonnées relatives des links.
* **Materials** : aspects esthétiques pour le rendu. ([wiki.ros.org](https://wiki.ros.org/urdf/Tutorials?utm_source=chatgpt.com), [wiki.ros.org](https://wiki.ros.org/urdf?utm_source=chatgpt.com)).

Pour éviter la redondance dans les robots complexes, on utilise XACRO : un préprocesseur qui introduit macros, propriétés et conditions, facilitant la maintenance et la génération paramétrique d’URDF. Par exemple, on peut définir la taille d’une roue une seule fois et la réutiliser sur chaque lien de roue via :

<xacro:property name="wheel\_radius" value="0.15"/>

...

<geometry><cylinder radius="${wheel\_radius}" length="0.05"/></geometry>

Sans XACRO, chaque valeur devrait être dupliquée manuellement dans l’URDF ([wiki.ros.org](https://wiki.ros.org/urdf/Tutorials?utm_source=chatgpt.com), [wiki.ros.org](https://wiki.ros.org/urdf?utm_source=chatgpt.com)).

Enfin, ROS fournit des outils de visualisation (rviz2) et de conversion URDF→SDF pour Gazebo (gz sdf -p model.urdf > model.sdf), garantissant l’interopérabilité entre simulateur et framework ROS 2 ([docs.ros.org](https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo/Gazebo.html?utm_source=chatgpt.com), [wiki.ros.org](https://wiki.ros.org/urdf_tutorial?utm_source=chatgpt.com)).

**3. Fichiers de lancement ROS 2 (Launch)**

Les **launch files** orchestrent le démarrage coordonné de multiples nœuds et processus ROS 2, remplaçant les scripts shell lourds. Écrits en Python, ils exploitent l’API launch et launch\_ros pour déclarer :

* **Nodes** à lancer (package, exécutable, arguments, remappings)
* **Parameters** à injecter dans la configuration des nœuds
* **GroupActions**, timers et conditions pour des scénarios complexes ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html?utm_source=chatgpt.com), [docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials.html?utm_source=chatgpt.com)).

Un exemple minimal :

from launch import LaunchDescription

from launch\_ros.actions import Node

def generate\_launch\_description():

return LaunchDescription([

Node(

package='limo\_description',

executable='spawn\_entity.py',

arguments=['-entity', 'limo\_pro', '-file', 'urdf/limo\_pro.urdf.xacro'],

output='screen'

)

])

Ce code lance un nœud ROS 2 qui invoque le plugin Gazebo pour apparaître dans la simulation ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html?utm_source=chatgpt.com), [docs.ros.org](https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo/Gazebo.html?utm_source=chatgpt.com)).

Les avantages principaux sont :

1. **Programmabilité** (boucles, if/else, import de fonctions)
2. **Modularité** (inclusion conditionnelle de sous-launch files)
3. **Réutilisation** via ament\_index\_python pour découvrir dynamiquement les chemins de packages ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials.html?utm_source=chatgpt.com), [docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html?utm_source=chatgpt.com)).

**4. Programmation Python**

Python, grâce à sa syntaxe claire et son vaste écosystème (NumPy, OpenCV, PyTorch), est omniprésent pour la vision, les scripts de test et la gestion des pipelines ROS 2.

**Bonnes pratiques** courantes :

* Respect de PEP 8 (indentation, noms de variables en snake\_case, 79 caractères max.)
* Isolation des dépendances dans un venv ou conda
* Documentation via docstrings au format Google ou NumPy
* Tests unitaires avec pytest et couverture minimale de 80 % ([Gist](https://gist.github.com/ruimaranhao/4e18cbe3dad6f68040c32ed6709090a3?utm_source=chatgpt.com), [appacademy.io](https://www.appacademy.io/blog/python-coding-best-practices/?utm_source=chatgpt.com)).

Un template de script ROS 2 en Python commence souvent par :

import rclpy

from rclpy.node import Node

from sensor\_msgs.msg import Image

class PerceptionNode(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_('perception\_node')

self.sub = self.create\_subscription(Image, '/camera/image\_raw', self.callback, 10)

def callback(self, msg):

# traitement de l'image

self.get\_logger().info('Image reçue')

def main(args=None):

rclpy.init(args=args)

node = PerceptionNode()

rclpy.spin(node)

node.destroy\_node()

rclpy.shutdown()

Cette structure standard facilite l’intégration rapide de tout module Python dans l’écosystème ROS 2 ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials.html?utm_source=chatgpt.com), [Gist](https://gist.github.com/ruimaranhao/4e18cbe3dad6f68040c32ed6709090a3?utm_source=chatgpt.com)).

**5. Programmation orientée objet (POO) en Python**

La POO permet d’encapsuler données et comportements dans des **classes**, améliorant la lisibilité et la maintenabilité du code. Les quatre piliers sont :

1. **Encapsulation** : attributs privés/protégés via \_attribut ou \_\_attribut.
2. **Abstraction** : interfaces claires (méthodes publiques vs privées).
3. **Héritage** : factorisation de code commun entre classes parent/enfant.
4. **Polymorphisme** : même interface, différentes implémentations ([realpython.com](https://realpython.com/python3-object-oriented-programming/?utm_source=chatgpt.com), [w3schools.com](https://www.w3schools.com/python/python_oop.asp?utm_source=chatgpt.com)).

**Exemple** – base d’un capteur :

class Sensor:

def \_\_init\_\_(self, name: str):

self.name = name

def read(self):

raise NotImplementedError

class LidarSensor(Sensor):

def \_\_init\_\_(self, name: str, angle\_res: float):

super().\_\_init\_\_(name)

self.angle\_res = angle\_res

def read(self):

# code de lecture du LiDAR

return [0.0] \* int(360 / self.angle\_res)

Ici, LidarSensor hérite de Sensor, partage l’attribut name et implémente read(), profitant de l’abstraction pour garantir une API uniforme ([realpython.com](https://realpython.com/python3-object-oriented-programming/?utm_source=chatgpt.com), [datacamp.com](https://www.datacamp.com/tutorial/python-oop-tutorial?utm_source=chatgpt.com)).

L’utilisation de classes améliore la **testabilité**, car on peut instancier des objets avec des arguments fictifs pour les tests unitaires (pytest + fixtures) et simuler facilement des sous-composants via des **mocks**.

**Références**

1. N. Koenig & A. Howard, *Design and use paradigms for Gazebo, an open-source multi-robot simulator*. IROS 2004. ([classic.gazebosim.org](https://classic.gazebosim.org/tutorials?tut=guided_b1&utm_source=chatgpt.com))
2. Gazebo Tutorials, *Beginner: Overview*. Gazebo Classic tutorial, version 7.0. ([gazebosim.org](https://gazebosim.org/docs/latest/getstarted/?utm_source=chatgpt.com))
3. URDF Tutorials, ROS Wiki. ([wiki.ros.org](https://wiki.ros.org/urdf/Tutorials?utm_source=chatgpt.com), [wiki.ros.org](https://wiki.ros.org/urdf?utm_source=chatgpt.com))
4. ROS 2 Launch documentation, ROS 2 Jazzy Tutorial. ([docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials.html?utm_source=chatgpt.com), [docs.ros.org](https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Creating-Launch-Files.html?utm_source=chatgpt.com))
5. A Guide of Best Practices for Python, GitHub Gist. ([Gist](https://gist.github.com/ruimaranhao/4e18cbe3dad6f68040c32ed6709090a3?utm_source=chatgpt.com))
6. 8 Python Best Practices Every Developer Should Know, App Academy. ([appacademy.io](https://www.appacademy.io/blog/python-coding-best-practices/?utm_source=chatgpt.com))
7. Object-Oriented Programming (OOP) in Python, Real Python. ([realpython.com](https://realpython.com/python3-object-oriented-programming/?utm_source=chatgpt.com))
8. Python OOP tutorial, W3Schools. ([w3schools.com](https://www.w3schools.com/python/python_oop.asp?utm_source=chatgpt.com))