Autonomous Road Inspection Challenge

The challenge theme is to build an autonomous mobile robot software system for an AgileX LIMO robot to survey roads, detect, localise, and quantify road defects (e.g., potholes), and present results on a map. Each student will implement their own individual solution. Any steering mode may be used (differential, ackermann, or mecanum). The robot will be placed at a designated starting point prior to each run. The robot must be fully autonomous making its own decisions about how to proceed.

The core requirements you **must** implement are

- Autonomous road surveying
  - You may choose a strategy, for example:
    - Generate waypoints and then follow them.
    - Integrating exploration algorithms (e.g., explore_lite).
    - Random reactive exploration.
- Potholes detection and localisation
  - Detect potholes center locations (x, y), their contours and areas (can be computed by calculating the bounding bax around the contour) as seen in Figure 1 using OpenCV and/or ML methods.

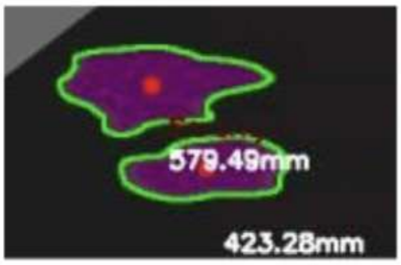

Figure 1: Pothole detection with contours

- Reporting and visualising
  - Publish the potholes positions on the map using visual markers in Rviz2 as seen in Figure 2 and also save center locations with their areas to CSV/JSON file.
  - Report the number of potholes detected to the terminal.

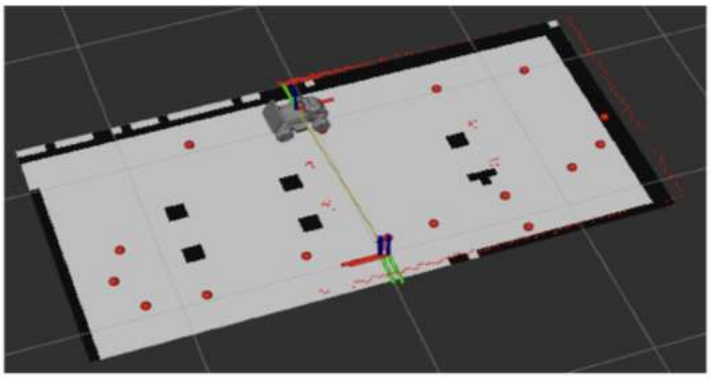

Figure 2: Marked pothole center locations in red in Rviz2.

Simulated Environment & ROS 2 Template Workspace

A Gazebo LIMO simulation that replicates the real challenge environment featuring the potholes highlighted in pink as shown in Figure 3. Assessment template ROS 2 package is provided within the KV6022 github repo workspace. Clone or pull the latest changes of the repo into your machine. You are required to use and extend the given `KV6022_assessment` template package and Gazebo simulation environment to test and develop your solution.

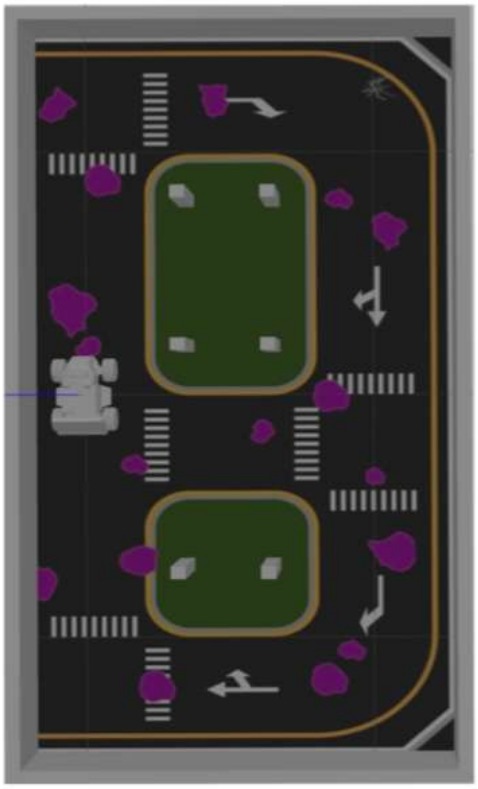

Figure 4: Simulation driving area

`KV6022_limo_ros2` workspace contains:

- Gazebo road inspection simulation world package (`limo_gazebosim`)
- Limo robot description package (`limo_description`)
- The package to be extended (`KV6022_assessment`). The following helper scripts in the package:
  - Navigation parameter file in (`/params/nav2_params.yaml`)
  - Occupancy grid map in `/maps` folder with 20 mm resolution.
  - An "example_opencv_detector" node
  - An "example_waypoint_follower", "example_nav_through_poses" and "example_nav_to_pose" node

These examples are a good starting point for your assessment. You may choose whether to use the provided example node implementations, however, all nodes you develop or use must be placed in the `KV6022_assessment` package.

How to Run Provided Workspace

1. Install the Navigation2 packages:

```
sudo apt install ros-humble-navigation2 ros-humble-nav2-bringup ros-humble-tf-transformations
```

2. Set up an alternative DDS (Cyclone DDS or Zenoh). Due to some issues with Fast DDS and Navigation in ROS2, it has been recommended to use Cyclone DDS or Zenoh instead. Install one of the following:

   Cyclone DDS:

```
sudo apt install ros-humble-rmw-cyclonedds-cpp
```

   Zenoh :

```
sudo apt install ros-humble-rmw-zenoh-cpp
```

3. Configure ROS 2 to use the chosen DDS. You need to set the "*RMW_IMPLEMENTATION*" environment variable in your *.bashrc* file.

   For switching to Cyclone DDS:

```
echo export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp >> ~/.bashrc
```

   For Zenoh:

```
echo export RMW_IMPLEMENTATION=rmw_zenoh_cpp >> ~/.bashrc
```

After editing *.bashrc*, close the terminal and open up a new one. Build and source your workspace.

If using Cyclone DDS, proceed directly to running the simulation.

If using Zenoh, start the Zenoh router in a separate terminal:

```
ros2 run rmw_zenoh_cpp rmw_zenohd
```

4. Run the simulation:

```
ros2 launch limo_gazebosim limo_gazebo_assessment.launch.py
```

5. Run the navigation node (high-resolution maps with adjusted parameters)

```
ros2 launch KV6022_assessment limo_navigation.launch.py
```

6. Run the object detector node:

```
ros2 run KV6022_assessment example_opencv_detector
```

7. Run the waypoint follower node:

```
ros2 run KV6022_assessment example_waypoint_follower
```

Deliverables

1. **In-person demo:** A ~5-minute presentation and live demonstration of your robotic system, supported by up to 5 slides.
   - Cover the following:
     - **Problem:** Clearly describe the problem and why it matters.
     - **Approach:** Explain and justify your method/solution.
     - **Result:** Highlight key outcomes.
2. **The report:** A written report documenting your project (structure provided in Report Marking Rubric).