Husky UVG

Utilizing ROS Noetic and Gazebo
for Autonomous Navigation

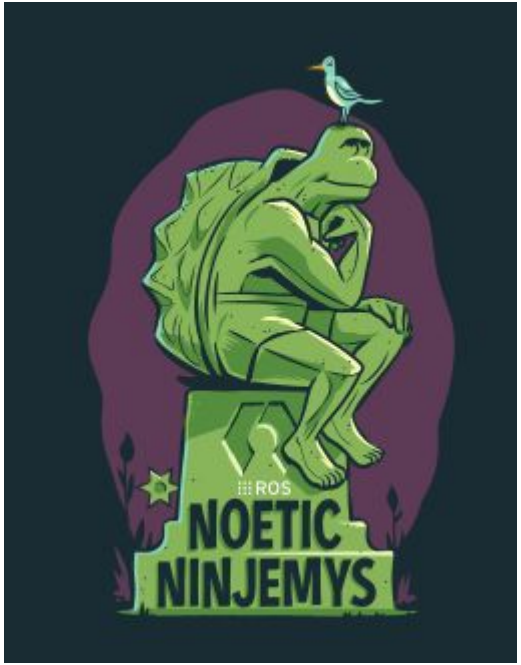
Project Overview

- Install and configure Ros Noetic on Ubuntu 20.04 LTS.
- Setup and control the husky UVG in a simulated environment.
- Create custom worlds in Gazebo.
- Record and execute trajectories using DMP.

ROS Noetic Installation

Resources:- ROS Noetic Installation Wiki

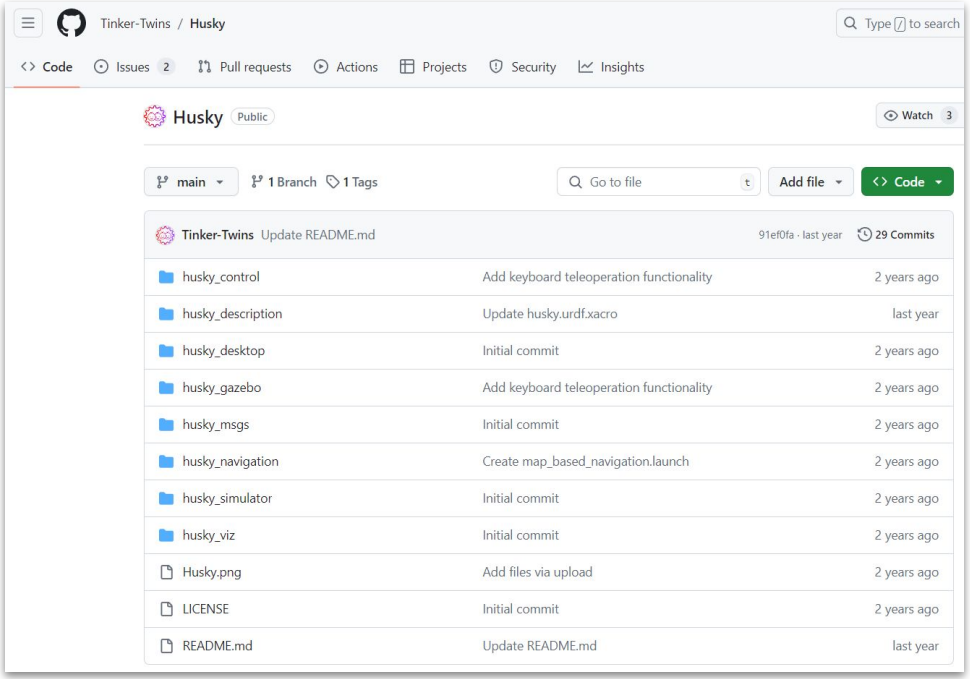
1. Update Ubuntu and install required packages.
2. Configure Ubuntu to accept ROS packages.
3. Install ROS Noetic.
4. Set up the environment.
5. Install additional ROS tools and dependencies.
6. Initialize rosdep.



Installing Husky UVG Dependencies

Resources:- [Tinker Twins Husky GitHub Repository]

1. Install necessary ROS Noetic packages.
2. Set up the Catkin workspace.
3. Clone the Husky repository.
4. Build the packages using `catkin_make`.



Launching and Controlling Husky in Gazebo

- Launch Husky in Gazebo.
  - `roslaunch husky_gazebo husky_playpen.launch`
- Control Husky using keyboard teleoperation.
  - `roslaunch husky_control teleop_keyboard.launch`
- Control Husky using an Xbox controller.
  - `rosrun joy joy_node`
  - `rosrun teleop_twist_joy teleop_node`

Navigation and SLAM

Required Step: Launch playpen and visualization.

- **Map-Less Navigation** is the capability of a robot to navigate without knowing the map. Previous works assume the availability of accurate self-localisation, which is, however, usually unrealistic.
    - roslaunch husky_navigation map_less_navigation.launch
- **Simultaneous localization and mapping (SLAM)** is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of a robot's location within it.
    - roslaunch husky_navigation gmapping.launch
- **AMCL (Adaptive Monte Carlo Localization)** is a localization algorithm commonly used in robotics. It is applied in mobile robots to accurately determine their position and direction in a given environment.
    - roslaunch husky_navigation amcl.launch

Creating Custom Gazebo Worlds

- Open Gazebo and enter Building Editor mode.
  - `gazebo`
- Create and save the building structure.
- Add elements and save the world.
- Create a launch file for the custom world.
- Launch the custom world with Husky.
  - `roslaunch husky_gazebo custom_world.launch`

Recording Husky's Trajectory

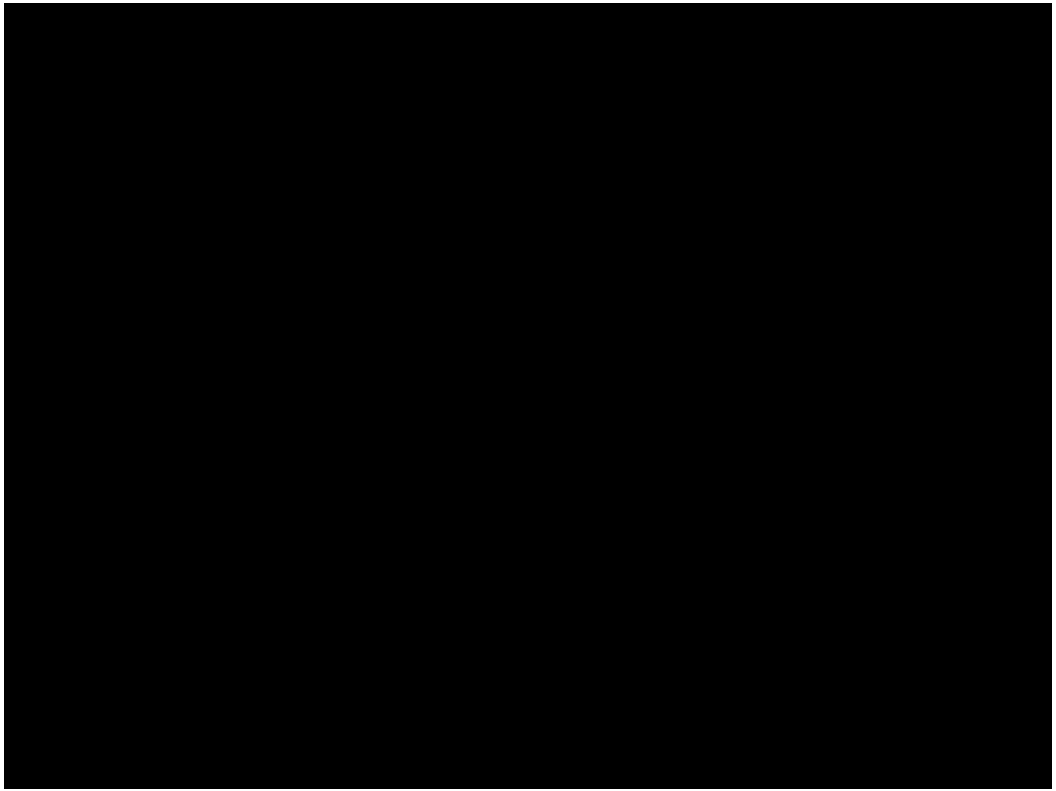
- Create a new package for recording trajectory data.
  - `catkin_create_pkg trajectory_recorder rospy std_msgs`
- Write and execute a Python script to record the trajectory.
- Launch Husky in Gazebo and control it using an Xbox controller.
- Run the node to record the trajectory.
  - `rosrun trajectory_recorder trajectory_recorder.py`

Training and Executing DMP Model

- Train a DMP model using recorded trajectory data.
  - `python3 DMP_trajectory_training.py`
- Generate and publish the trajectory using the trained DMP parameters.
  - `rosrun trajectory_recorder publish_trajectory.py`
- Create a ROS node to execute the trained DMP trajectory in Gazebo.
  - `rosrun trajectory_recorder TrainedDMP_trajectory_execute.py`

Husky Simulation Demonstration

Husky Real Time Demonstration



Challenges and Solutions

- **Challenges Faced:**
  - Installation and configuration issues.
  - Controlling Husky in Gazebo.
  - Recording and accurately reproducing trajectories.
- **Solutions Implemented:**
  - Detailed troubleshooting and use of community resources.
  - Fine-tuning control parameters.
  - Iterative training and testing of DMP models.

Conclusion and Future Work

- **Summary:**
  - Successful installation and configuration of ROS Noetic and Husky UGV.
  - Effective control and navigation in simulated environments.
  - Creation of custom worlds and trajectory execution.
- **Future Work:**
  - Implementation in real-world scenarios.
  - Integration with additional sensors and algorithms.
  - Enhancement of trajectory planning and execution.

“Thank You”