UNIVERSITY OF MINNESOTA

Department of Computer Science and Engineering

CSCI 5551 | **Introduction to Intelligent Robotic Systems** | **Fall 2024**

Assignment 2 and Extra Credit 2

Apurv Kushwaha
kushw022@umn.edu

Problem

Problem 1: TurtleBot3 Obstacle Avoidance

- Use a TurtleBot3 in Gazebo with LiDAR data from topic `/scan`.
- Implement obstacle detection and automatic avoidance.
- Robot is controlled by keyboard teleop, but when an obstacle is detected in front, the robot should override teleop and rotate to the right until the path is clear.
- Provide a launch file to start simulation, teleop, and the custom nodes.

Problem 2: Kinova Gen3 Robot Manipulator

- Use the Kinova Gen3 arm in Gazebo.
- Plan and execute five different end-effector trajectories using MoveIt.
- Save the trajectories to a file and later load and execute them again.

Extra Credit 2: LiDAR Ranges Publisher

- Subscribe to the TurtleBot3 Gazebo `/scan` topic.
- Extract the `ranges` array and publish it as `float32[]` on a new topic `/ranges`.

System Setup

This assignment was developed and tested on:

- **Host OS:** Ubuntu 24.04 LTS
- **ROS Version:** ROS1 Noetic (running in Docker container)
- **Docker Image:** `osrf/ros:noetic-desktop-full`

Note: The submitted `catkin_ws` folder is compatible with any ROS1 Noetic environment (natively on Ubuntu 20.04 or via Docker).

Required ROS Packages

- `turtlebot3`, `turtlebot3_simulations`, `turtlebot3_msgs`
- `ros_kortex` (Kinova manipulator packages)
- `moveit_commander`

Setup and Execution Instructions

Workspace Setup

```
# Extract the submitted ZIP file
unzip catkin_ws.zip

# Build the workspace
cd catkin_ws
catkin_make
source devel/setup.bash

# Set TurtleBot model
export TURTLEBOT3_MODEL=burger
```

Running Problem 1: TurtleBot Obstacle Avoidance

Single command to launch everything:

```
roslaunch hw_pkg hw2.launch
```

Keyboard Controls:

- `w` – Move forward
- `x` – Move backward
- `a` – Turn left
- `d` – Turn right
- `s` or `spacebar` – Stop

Expected Behavior: Robot moves based on keyboard input. When an obstacle is detected within 0.5m in a 30-degree cone (±15°) directly in front, the robot automatically stops and turns right until the path is clear.

Running Problem 2: Kinova Trajectories

Terminal 1 – Launch Gazebo:

```
roslaunch kortex_gazebo spawn_kortex_robot.launch gripper:=robotiq_2f_85
```

Wait for Gazebo and RViz to fully load.

Terminal 2 – Collect Trajectories:

```
rosrun hw_pkg trajectory_collection.py __ns:=my_gen3
```

This will plan and execute 5 trajectories, then save them to `trajectories.pkl`.

Terminal 2 – Execute Saved Trajectories:

```
rosrun hw_pkg trajectory_execution.py __ns:=my_gen3
```

The robot arm will execute all 5 saved trajectories in sequence.

Verifying Extra Credit

To verify the `/ranges` topic is publishing:

```
rostopic echo /ranges
```

You should see an array of LiDAR range measurements being published.

Solution

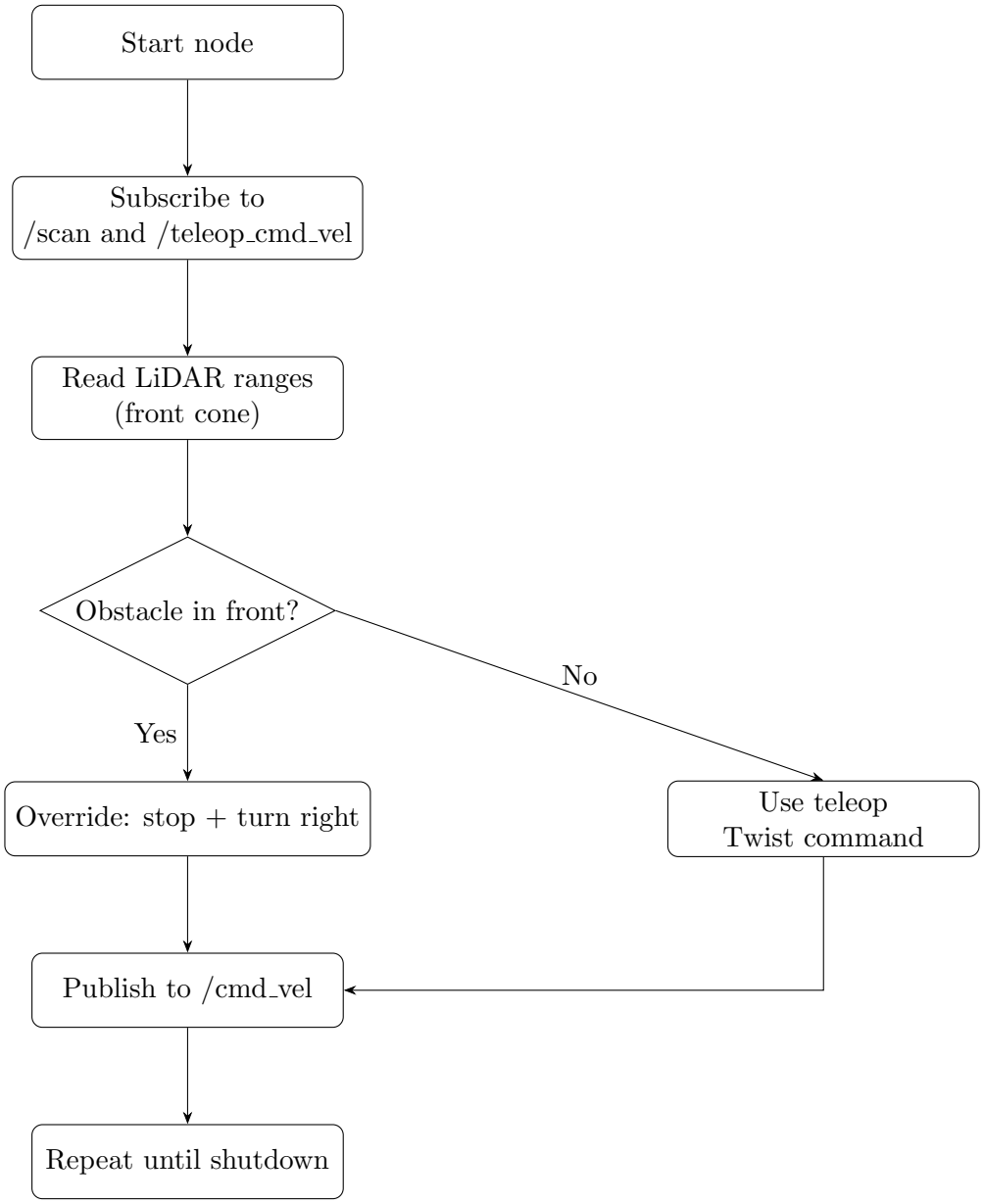
All code is implemented in a ROS1 Noetic catkin workspace. A custom package `hw_pkg` contains scripts and launch files for the assignment.

Problem 1: TurtleBot3 Obstacle Avoidance

Nodes Implemented

- `obstacle_detection.py`:
  - Subscribes to `/scan` (`sensor_msgs/LaserScan`).
  - Copies `msg.ranges` into a `std_msgs/Float32MultiArray` and publishes on `/ranges`.
  - Used mainly for debugging and to satisfy the extra credit requirement.
- `obstacle_avoidance.py`:
  - Subscribes to `/scan` and a teleop velocity topic (`/teleop_cmd_vel`).
  - Publishes velocity commands to `/cmd_vel`.
  - Checks a 30-degree cone in front of the robot (±15° from center) and sets a flag if the minimum valid distance is below 0.5 m.
  - If an obstacle is detected, it overrides teleop and commands the robot to stop and rotate right; otherwise, it forwards the teleop command.

Flow Diagram: TurtleBot Obstacle Avoidance

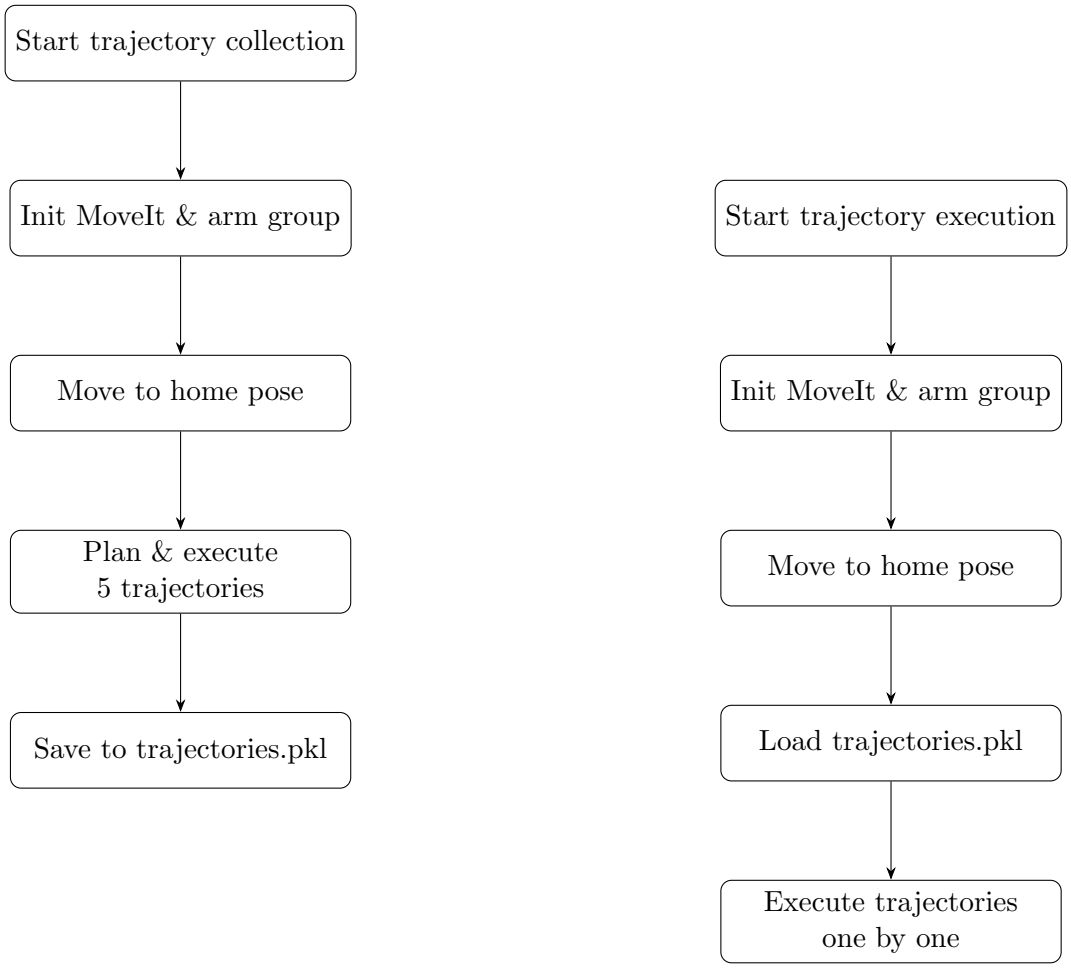


Problem 2: Kinova Gen3 Robot Manipulator

Nodes Implemented

- `trajectory_collection.py`:
  - Uses `moveit_commander` to control the Kinova arm group (`"arm"`).
  - Moves the robot to a known home position.
  - Plans and executes five trajectories:
    1. Home → Vertical
    2. Vertical → Home
    3. Home → Custom Pose 1
    4. Custom Pose 1 → Custom Pose 2
    5. Custom Pose 2 → Home
  - Stores the planned `RobotTrajectory` objects in a Python list and saves them to `trajectories.pkl` using `pickle`.
- `trajectory_execution.py`:
  - Re-initializes MoveIt for the same arm group.
  - Moves the robot to the home pose to match the starting conditions.
  - Loads `trajectories.pkl` and executes each stored trajectory in sequence.

Flow Diagram: Kinova Trajectory Workflow



Extra Credit 2

The extra credit requirement to publish `float32[]` ranges on `/ranges` is satisfied by the `obstacle_detection.p` node:

- **Input:** `/scan` (`sensor_msgs/LaserScan`)
- **Output:** `/ranges` (`std_msgs/Float32MultiArray` containing `ranges`)

The topic can be verified with:

```
rostopic echo /ranges
```

Results

- **Problem 1:** Successfully demonstrated TurtleBot3 obstacle avoidance with automatic right turn override when obstacles are detected within 0.5m.
- **Problem 2:** Successfully collected and executed all 5 Kinova arm trajectories using `moveit_commander`.
- **Extra Credit:** `/ranges` topic publishing verified with `rostopic echo`.