SLAM Implementation on Differential Drive Robot using Extended Kalman Filter

ECE T580

Eashan Gallage
Khoa Nguyen
Brandon Xu

Contents

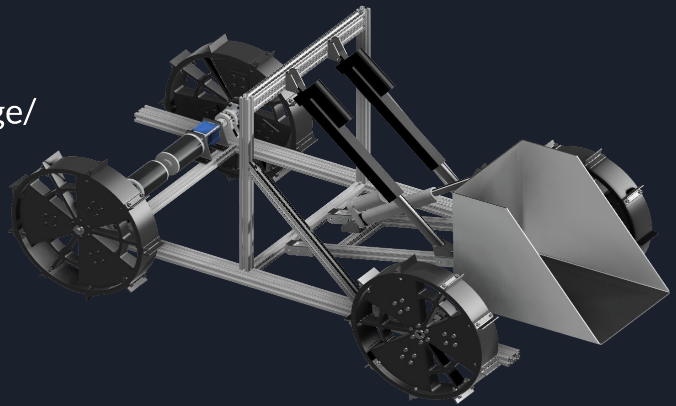
Introduction and Background

Motivation was to implement SLAM using Extended Kalman Filter Localization on a physical robot. The robot used was one slated to participate in the NASA Lunabotics Challenge. The robot uses a differential drive model for locomotion, and is controlled using a Raspberry Pi running ROS2. Robot sensor inputs include odometry, LIDAR, and an IMU.

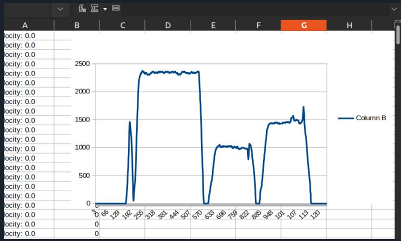
https://www.nasa.gov/learning-resources/lunabotics-challenge/

NASA
EX LUNA, SCIENTIA
LUNABOTICS

DREXEL LUNABOTICS



PID Tuning



The motor provided us with the encoder counts per second and we converted that to linear velocity.

$$F = \frac{encoder\ counts\ per\ second}{encoder\ counts\ per\ revolution * gear\ ratio}$$

$$\omega = 2\pi * F$$

$$V = \omega * radius = \omega * 0.178\ m$$

Proportional relationship between V and encoder counts per second

Hence the PID tuning was done on encoder counts per second

Was very important when syncing the 2 motors

```
container = ComposableNodeContainer(
    name="PhoenixContainer",
    namespace="",
    package="ros_phoenix",
    executable="phoenix_container",
    parameters=[{"interface": "can0"},{"use_container": False}],
    composable_node_descriptions=[
        ComposableNode(
            package="ros_phoenix",
            plugin="ros_phoenix::TalonSRX",
            name="left_wheel_joint",
            parameters=[{"id": 1},
                        {"P": 0.95},
                        {"I": 0.0},
                        {"D":0.0},
                        {"sensor_multiplier":1.0},
                        {"edges_per_rot":4096}],
        ),
        ComposableNode(
            package="ros_phoenix",
            plugin="ros_phoenix::TalonSRX",
            name="right_wheel_joint",
            parameters=[{"id": 3},
                        {"P": 1.2},
                        {"I": 0.0},
                        {"D":0.0},
                        {"sensor_multiplier":-1.0},
                        {"edges_per_rot":4096}],
        ),
    ],
    condition=IfCondition(LaunchConfiguration('use_container')),
    output="screen",

)
```

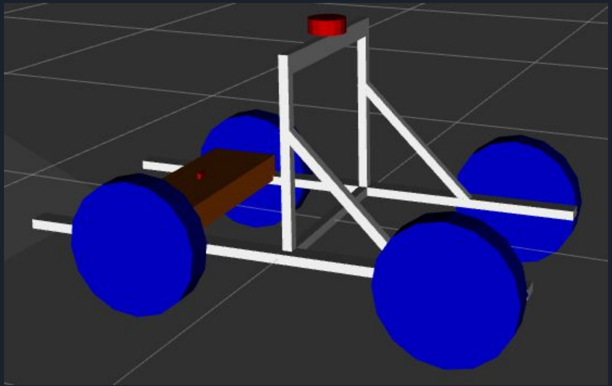
EKF in Gazebo

Robot is approximately modeled in Gazebo using an URDF

Used robot localization package to perform sensor fusion for EKF

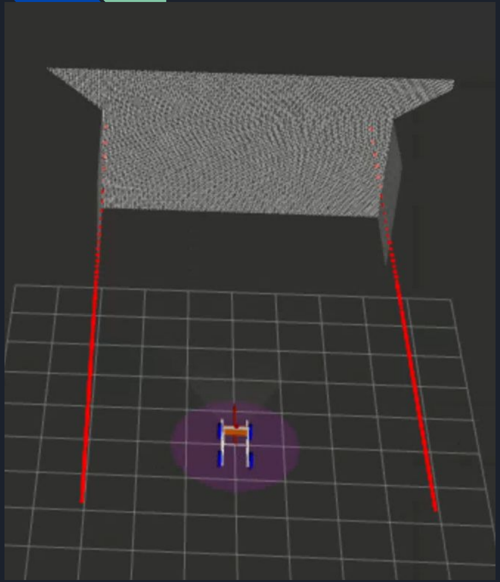
Input to ekf node from Odom and IMU: Linear velocity of x,y and z, yaw velocity/angular velocity around z

https://automaticaddison.com/sensor-fusion-using-the-robot-localization-package-ros-2/

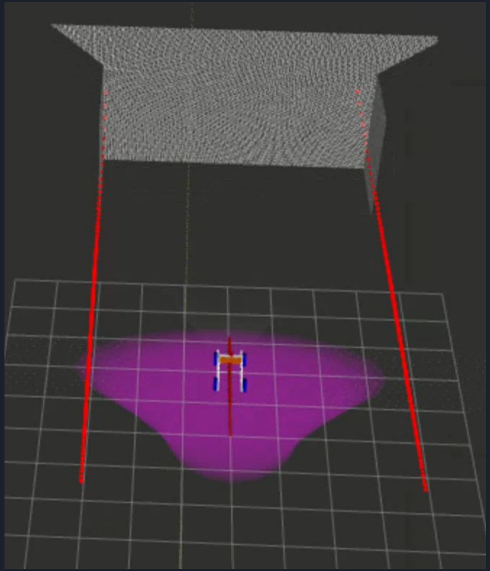


```
/ekf_filter_node
  Subscribers:
    /clock: rosgraph_msgs/msg/Clock
    /diffbot_base_controller/odom: nav_msgs/msg/Odometry
    /imu/data_raw: sensor_msgs/msg/Imu
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /set_pose: geometry_msgs/msg/PoseWithCovarianceStamped
  Publishers:
    /diagnostics: diagnostic_msgs/msg/DiagnosticArray
    /odometry/filtered: nav_msgs/msg/Odometry
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /tf: tf2_msgs/msg/TFMessage
  Service Servers:
    /ekf_filter_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /ekf_filter_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /ekf_filter_node/get_parameters: rcl_interfaces/srv/GetParameters
    /ekf_filter_node/list_parameters: rcl_interfaces/srv/ListParameters
    /ekf_filter_node/set_parameters: rcl_interfaces/srv/SetParameters
    /ekf_filter_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
    /enable: std_srvs/srv/Empty
    /set_pose: robot_localization/srv/SetPose
    /toggle: robot_localization/srv/ToggleFilterProcessing
```

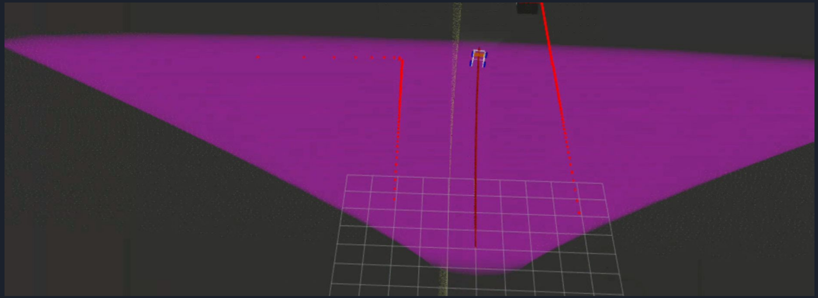
Robot_localization EKF in RVIZ



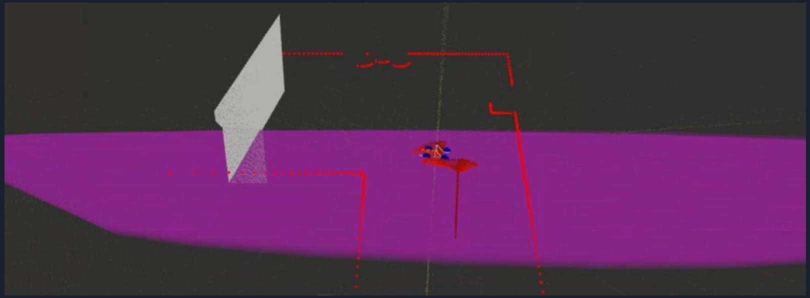
T = 0



T = t1



T = t2

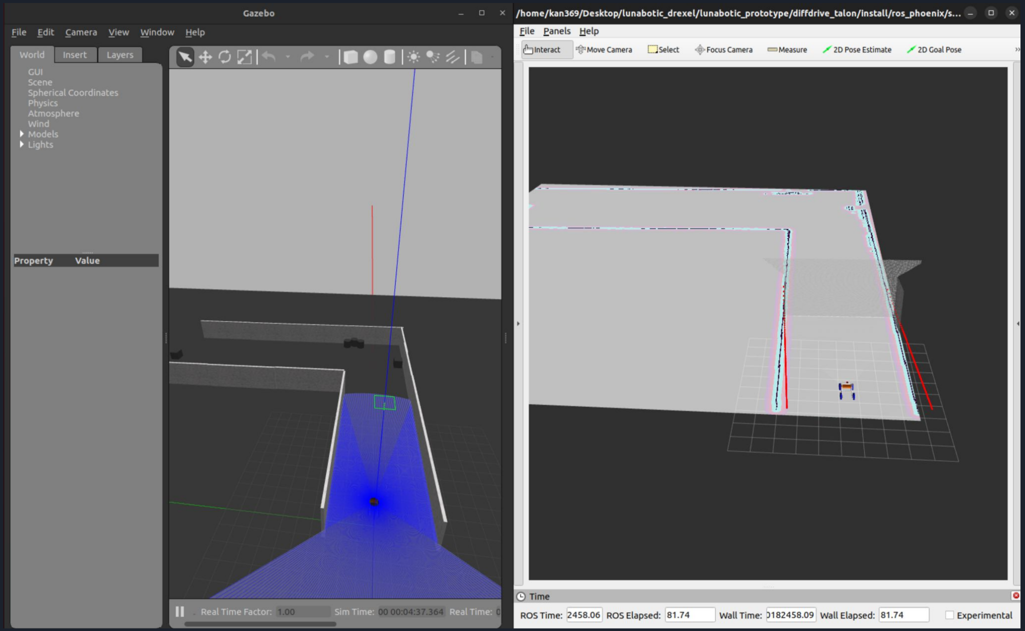


T = t3

t1 < t2 < t3

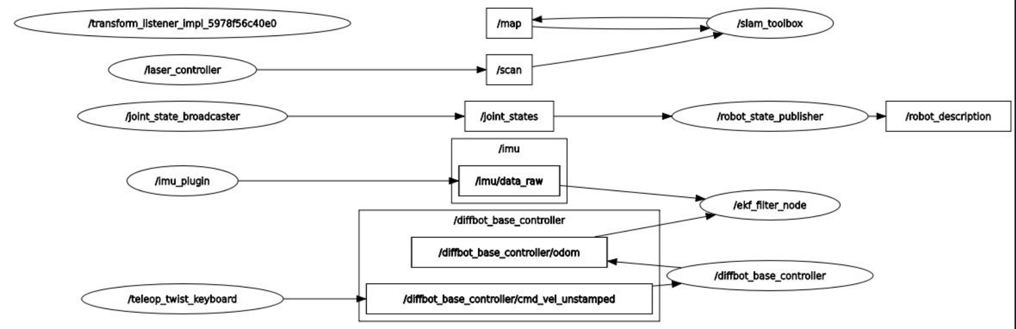
Autonomous Navigation with Nav2 and EKF

- Creating a simulated world in gazebo and evaluating SLAM. EKF and NAV2.

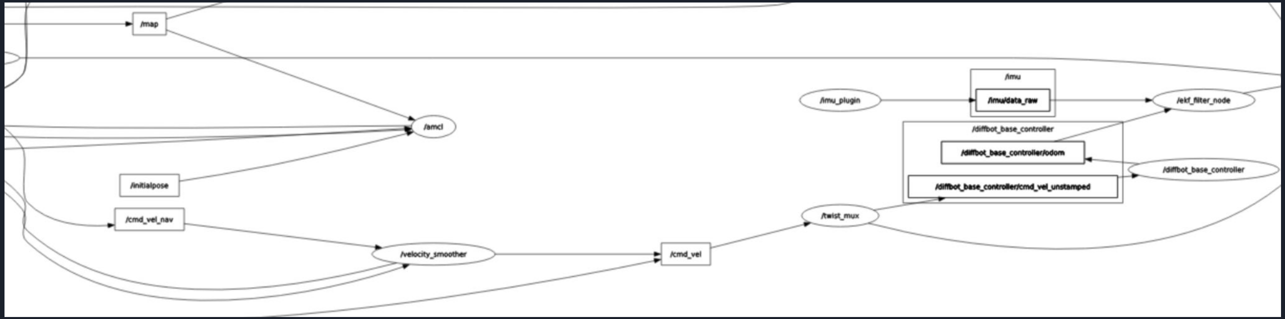


Simulated Autonomous Navigation

RQT_GRAPHs



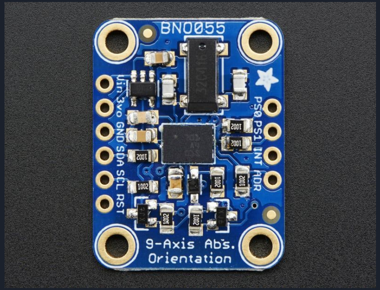
Rqt_graph: During SLAM mapping

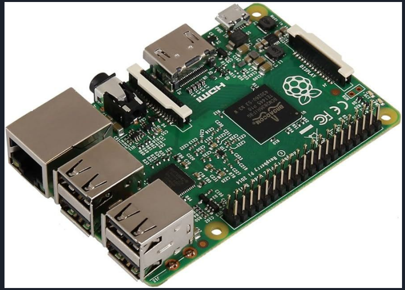


Rqt_graph: During NAV2 implementation

EKF and SLAM in Physical Robot

- BNO055 IMU sensor was used and it communicated to the raspberry PI through Inter-integrated communication (I2C).
- Raspberry PI was used as the main System on Chip in the robot, connected to the developer machine and remote controlled through SSH.
- 2D RPlidar was used for Simultaneous Localization and Mapping, SLAM, of the robot in the unknown environment





EKF and SLAM in Physical Robot

Conclusion

1. The Extended Kalman Filter (EKF) effectively localized the robot's position by utilizing IMU sensor data and wheel odometry from motor encoders, providing precise estimates.
2. The reliability of IMU sensor diminishes during rapid motions.
3. The motor controller demands greater power for angular velocity than what is theoretically expected from the differential-drive controller.
4. Raspberry Pi frequently crashed when attempting simultaneous mapping and localization tasks due to insufficient processing power.
5. To implement Adaptive Monte-Carlo Localization and NAV2, a higher version of the SOC system is required.

Contributions

Eashan: IMU integration, PID tuning, robot localization

Khoa: LIDAR integration, PID tuning, SLAM, robot localization

Brandon: IMU integration, robot localization

Questions?