Robot Localization with ROS: EKF-based Sensor Fusion Implementation

Mini-Project 1 Progress Report

Nicholas Birch de la Calle (IST1116701)
Antonio Maria Trigueiros de Aragão Moura Coutinho (IST1196837)
Gabriel Badan (IST1116537)
Janaína da Silva Pacheco (IST1117233)

Instituto Superior Técnico

September 18, 2025

Outline

Team Workflow & Scope for This Week

- All teammates use macOS; VMs made Wi-Fi connection to the robot tricky.
- We split work to move faster:
  - **Group A:** Fix connectivity to the real robot (prep for mapping in Step 2).
  - **Group B:** Use the dataset (.bag) and implement Step 1 (EKF with `robot_localization`).
- Collaboration: shared notes, common checklist, quick pair-debug sessions on TF and timing.
- **This presentation:** Focus on Step 1 progress with the dataset. Step 2 will follow next week.

Project Objectives

Main Goal

Implement and test robot self-localization using Extended Kalman Filter (EKF) based localization with ROS `robot_localization` package

- **Platform:** TurtleBot3 Waffle Pi with real sensor data
- **Sensors:** IMU, wheel odometry, laser scanner
- **Method:** EKF-based sensor fusion
- **Data:** Pre-recorded rosbags with ground truth from Motion Capture System
- **Framework:** ROS Noetic environment

Key Learning Outcomes

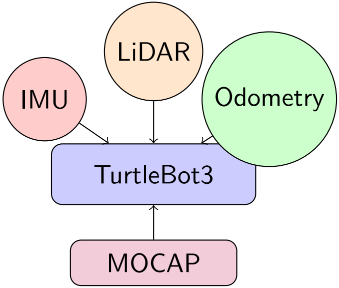
Understanding Bayesian filtering, ROS navigation stack, and practical sensor fusion implementation

Dataset Information

extbfWhat's inside (kept simple):

- Wheel odometry (`/odom`)
- IMU measurements (`/imu`)
- Laser scans (`/scan`)
- Ground truth in TF (`mocap` → `mocap_laser_link`)
- Camera topics are present but not used this week

We read the bag with simulated time and use the helper TF script from the dataset repo.



How We Ran It

- Start ROS core and enable simulated time.
- Play the dataset with `--clock`.
- Use the helper script to connect ground-truth TF to a fixed frame.
- Start the EKF node and view topics in RViz (Fixed Frame: `odom`).

Kalman Filter in Two Minutes

- **Goal:** Combine noisy sensor readings to estimate the robot's pose.
- **Predict:** Use motion (odometry) to predict where we are now.
- **Update:** Correct that prediction with another sensor (e.g., IMU).
- **Trust:** Weight each sensor by how noisy it is (uncertainty/covariance).
- **Result:** A smoother, more reliable trajectory than any single sensor.

What We Accomplished

- Set up ROS workspace and installed required packages.
- Downloaded and played the dataset with simulated time.
- Connected frames and launched the EKF node.
- Displayed raw and filtered trajectories in RViz.

EKF Configuration (Conceptual)

- Node: `ekf_localization_node` at 5 Hz.
- Frames: `odom` (world), `base_footprint` (base), world frame = `odom`.
- Inputs: `/odom` (wheel odometry) enabled for planar position (x, y).
- IMU to be added next (orientation & angular velocity) with tuned covariances.
- Static TFs: `base_footprint` ↔ `base_link`, `base_link` ↔ `imu_link`.

Problem-Solving Approach

1. Environment Variables

Issue: `TURTLEBOT3_MODEL is not set`
Solution: Set `TURTLEBOT3_MODEL=waffle_pi` and persist it in shell config.

2. Time Synchronization

Issue: Timing messages and data inconsistencies
Solution: Proper simulation time configuration before any node launch

3. Frame Connectivity

Issue: Transform errors between sensor frames
Solution: Static transform publishers to connect missing frame links

Status and Next Steps

- **Status:** Dataset playback and EKF are running; frames connected; RViz shows trajectories.
- **Next:** Add IMU to the EKF, tune parameters, and compare filtered vs raw path.
- **Coming up:** Solve VM networking to run gmapping with the real robot (Step 2).

Immediate Next Steps

1. EKF Configuration Debugging

- Add IMU data to EKF configuration
- Tune sensor noise parameters
- Verify frame ID consistency across all sensors
- Enable additional state variables if needed

2. Enhanced Configuration

- Integrate IMU orientation (roll, pitch, yaw) and angular rates.
- Calibrate covariances, start conservative then refine.
- Consider enabling velocities in the state if needed.

RViz Trajectory Comparison

- **Fixed Frame:** `odom`.
- **Raw odometry:** `/odom` (displayed as a Path or Odometry, e.g., red).
- **Filtered odometry:** `/odometry/filtered` (e.g., green).
- Keep **100** points; use distinct colors and small arrowheads for clarity.
- Observation: filtered path is smoother and more stable than raw odometry.

Wrap-up

- We have a working pipeline with the dataset and EKF.
- We understand the basics of Kalman filtering (predict + update).
- Next, we'll add IMU, tune it, and move to mapping with the real robot.

Final Remarks

Progress Summary

Successfully established a complete working environment for EKF-based robot localization with real sensor data. The foundation is solid for completing the sensor fusion implementation.

Next Session Goals

1. Complete EKF parameter tuning
2. Achieve filtered odometry output
3. Perform quantitative evaluation against ground truth
4. Prepare for Phase 2: SLAM with gmapping

Thank you for your attention!

Questions?