The "journey" has been a classic example of moving from **theory** to **simulation** to **robust implementation**.
Here is a summary of the key challenges and solutions, suitable for the "Implementation" or "Discussion" section of your final report.

## 1. The Gravity Leak (Attitude Divergence)

The Problem: The robot's estimated velocity would "explode" (reach >10 m/s) even when the robot was barely moving.
The Cause: In a robocentric EKF, a small error in Pitch or Roll causes the gravity vector ($g = 9.81 m/s^2$) to project onto the horizontal body axes. The filter interpreted this projection as a constant acceleration, integrating it into infinite velocity.
The Fix:

- **Sigma_G Increase:** We increased the Gyroscope process noise (sigma_g from 0.05 to 0.3). This loosened the filter's "trust" in the gyro, allowing the Vision/ZUPT updates to forcefully correct the orientation.
- **Bias Locking:** We froze the Bias Random Walk (Q_ba = 0) to prevent the filter from "learning" a false bias to explain the gravity leak, which was causing the infinite reset loops.

## 2. The "Teleporting Robot" (Update Instability)

The Problem: The robot's trajectory was jagged/zigzaggy. The filter would often "clamp speed" or reject valid landmarks because the correction was too large.
The Cause: The filter was configured to trust the camera excessively (R_cam = 5.0). The IEKF (Iterated EKF) was overfitting to noisy visual measurements, trying to "teleport" the robot to the exact landmark location in a single step, which violated the physical motion constraints.
The Fix:

- **Relaxed Trust:** Increased R_cam to 50.0. This acted as a low-pass filter, smoothing the trajectory.
- **Disabled IEKF:** Reverted to a standard EKF (MAX_ITERATIONS = 1) to prevent aggressive overfitting.
- **Correction Clamping:** Implemented a safety clamp on dx to prevent any single update from moving the robot more than 10cm or changing velocity by >0.5 m/s.

## 3. The "Silent Killer" (First-Estimate Jacobian)

The Problem: The filter would work fine for 17 seconds and then violently explode during a turn.
The Cause: We were using a First-Estimate Jacobian (FEJ) for the measurement update but calculating the residual with the Current State. As the robot rotated, the old Jacobian pointed in the wrong direction relative to the new residual. The filter applied corrections backwards, amplifying the error instead of reducing it.
The Fix: Reverted to a standard EKF Jacobian linearized at the current state (R_wb instead of R_fej). This ensured the math (H) and the physics (z) were in the same coordinate frame.

## 4. Simulation Jitter (Gazebo Physics)

The Problem: Gazebo's physics engine produced high-frequency "impact spikes" (25 $m/s^2$) that triggered the divergence watchdog unnecessarily.
The Fix: Implemented a 4-Sample Downsampling (200Hz $\to$ 50Hz). This acted as a hardware low-pass filter, averaging out the physics glitches before they entered the EKF prediction step.

## Final Results

- **Yaw Accuracy:** ~0.09 degrees mean error (Excellent).
- **Position Accuracy:** ~0.73 meters RMSE (Good for visual odometry without loop closure).
- **Stability:** Bounded error, no explosions, robust to maneuvers.

This narrative demonstrates a deep understanding of EKF mechanics: balancing observability, handling non-linearities, and managing sensor noise characteristics. This is exactly what a graduate-level robotics project should demonstrate.