Basavaraj Navalgund

+919845268868 | basavarajnavalgund97@gmail.com

Bengaluru, IN

LinkedIn | GitHub | Portfolio

Personal Statement

I'm an experienced professional, who mastered both hardware and software skills throughout a journey of building personal projects, and working in the industry. I am passionate about the autonomous systems field including ROS Robots and Autonomous Driving. I consider myself to be a strong software engineer who is capable of researching, designing, and developing of software solutions.

Education

KLE Technological University — Hubli, IN

Bachelor of Engineering in Electronics and Communication — *Aug. 2016 – June 2020*

Technical Skills

C++, Python, ROS, Gazebo, Image Processing, GitHub, Linux, Arduino, Continuous Integration

Experience

Software Engineer — Sep. 2021 – Present

Mercedes-Benz Research and Development India — *Bengaluru, IN*

- Software Development in the Sensor Fusion team for Autonomous Driving
- Good knowledge of simulation tools, software debugging, and analysis of measurement recorded during vehicle test drives
- Experience in Atlassian tool chain - JIRA, Bitbucket, Confluence etc
- All software was based on ROS framework, and was written in C++ or Python

ROS Developer — Apr. 2021 – Sep. 2021

Conigital Group — *WFH*

- Software Development, Validation, and Simulation for Level 3+ Autonomous Vehicle (AV)
- Modelled sensors on the simulation environment and configured them on the vehicle to create a digital twin
- Performed virtual validation and testing of path planning and control modules, which generates trajectories for the vehicle and calculates control inputs (steering, acceleration, and deceleration!) to execute those trajectories using CARLA with Autoware simulator
- Created ROS package to establish the communication between Aslan and ConOP application via vehicle telemetry messages (GPS, IMU, and Odometry)
- Mapped topics published and subscribed on ROS, that serves as the middleware for the AV stack
- All software was based on ROS framework, and was written in C++ or Python

Graduate Engineer — Oct. 2020 – Feb. 2021

Mando — *Bengaluru, IN*

- Software development for ADAS applications
- Image Processing algorithm development, and testing
- All software was written in Python

Software Engineer Intern — Feb. 2020 – June 2020

Continental — *Bengaluru, IN*

- Software Development for Autonomous Mobile Robot (AMR)
- Worked primarily as a ROS developer with a focus on SLAM and Navigation stack for Holonomic drive with 4 mecanum-wheeled AMR
- Interfaced multiple ROS packages and coded nodes in C++ and Python to autonomously map an environment, localize itself, and navigate to pick-up and drop-off objects in a manufacturing plant
- Expanded my knowledge in both the hardware and software side of engineering especially in Linux, C++, Python, and ROS

Projects

- **lidar_robot_mapping** (ROS, Autoware, Gazebo, C++)
  Hands-on Experience with RS–LiDAR-16, OS2–64, and OS2–128 to build an occupancy grid of controlled environment. Fusion of slam_toolbox with Autoware package based NDT Mapping to get odom_to_baselink transformation and, accurate 2D SLAM
- **t265_robot_navigation** (ROS, Gazebo, C++) [code]
  Using Intel Realsense D435 and T265 to build an occupancy grid and autonomously navigate around using movebase
- **Workplace Service Robot** (ROS, Gazebo, C++, Python) [code]
  Autonomous Mobile Robot (AMR), a holonomic drive with 4 mecanum-wheels. It autonomously maps an environment, localizes itself, and navigate to pick-up and drop-off objects in a simulated environment
- **Sensor fusion-based Tracking** (C++) [code]
  Designed and implemented a Kalman filter, Extended Kalman filter, and Unscented Kalman filter for object tracking
- **Vehicle Localization** (C++) [code]
  Implemented a real-time particle filter to estimate the position and orientation of a moving vehicle
- **Path Planning** (C++) [code]
  Implemented a simple real-time path planner in C++ to navigate a car around a simulated highway scenario, including other traffic, given waypoint, and sensor fusion data
- **Model Predictive Control** (C++) [code]
  Implemented a Model Predictive Control (MPC) to drive a car in a game simulator. The server provides reference waypoints via websocket, and we use MPC to optimize the actuators (steering and throttle), simulate the vehicle trajectory, and minimize the cost like cross-track error.