Building an Autonomous Testbed for Planning and Control Algorithms on a modified RC Car

Aditi Chandrashekar

Under the mentorship of Soon-Jo Chung, John Lathrop, and Ben Rivière

8/24/2023

INDY Autonomous Challenge (IAC)

Autonomous vehicles face several obstacles to commercialization

1. Safety/ Reliability (Need Edge-Case testing/ solutions)
2. Expense
3. Public Opinion

INDY Autonomous Challenge
A racing series for fully autonomous race cars.

Mission/ Goal

Designing Planning & Control Algorithms for IAC in collaboration with KAIST.

An integral step the design process is testing.

There are two options:

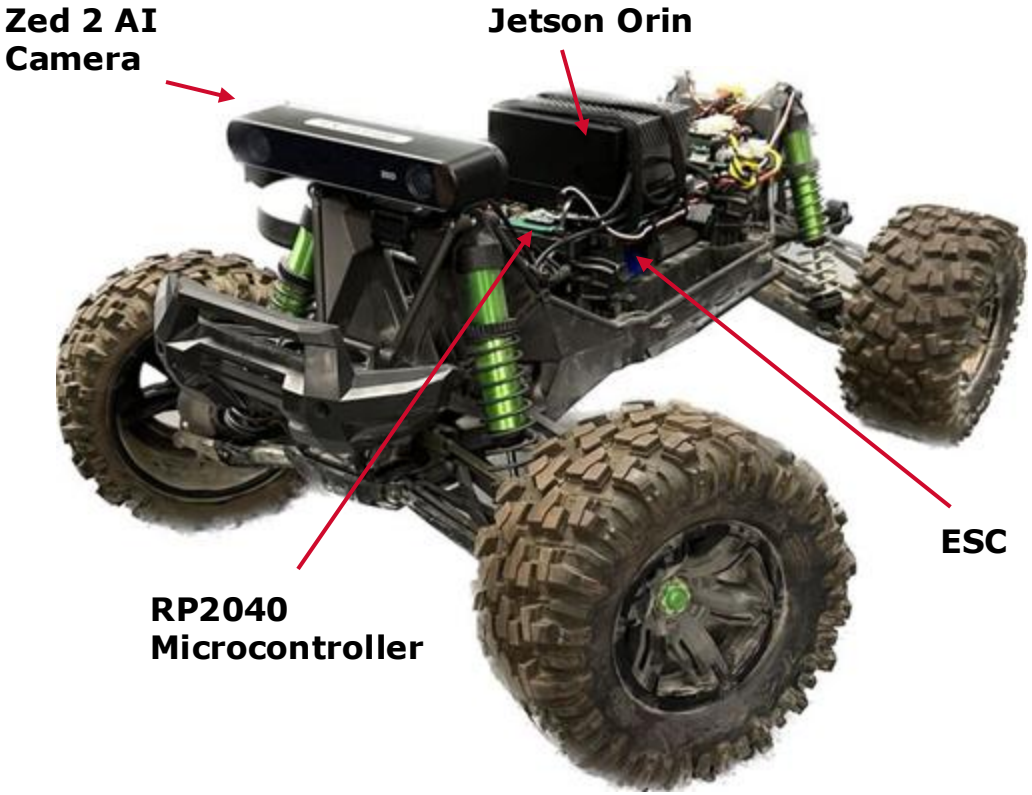
1. Testing in simulation
2. **Testing on a physical robot**

Goal: Create an autonomous testbed for Control and Planning algorithms on an RC Car.

Traxxas X-Maxx

Hardware Stack

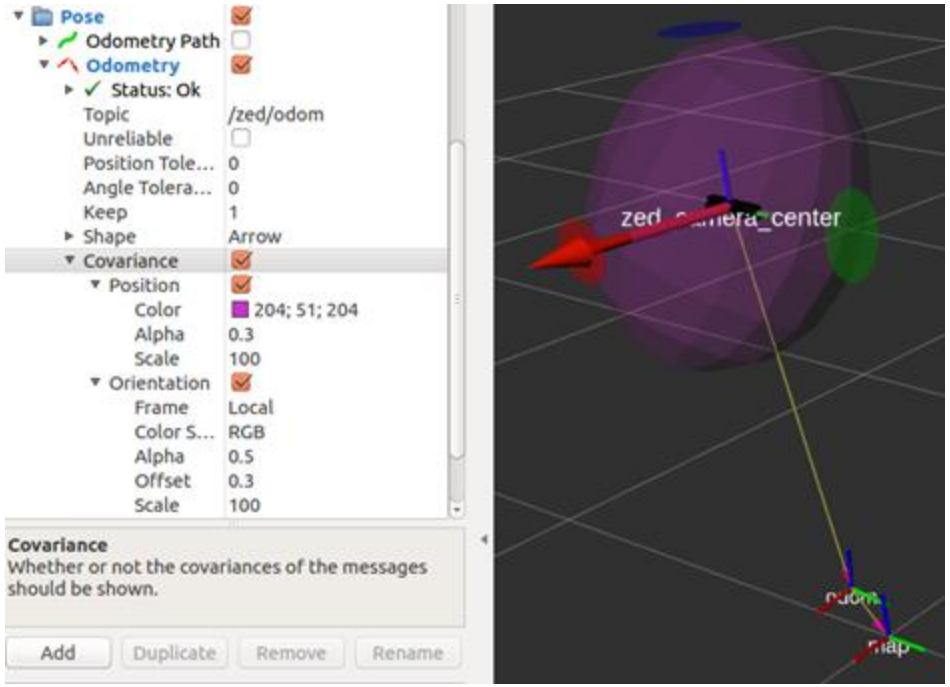
- NVIDIA Jetson Orin installed for onboard computing.
- Zed 2 AI camera integrated for perception.
- Steering servo and ESC preserved from original car.
- RP2040 microcontroller used to receive command messages.

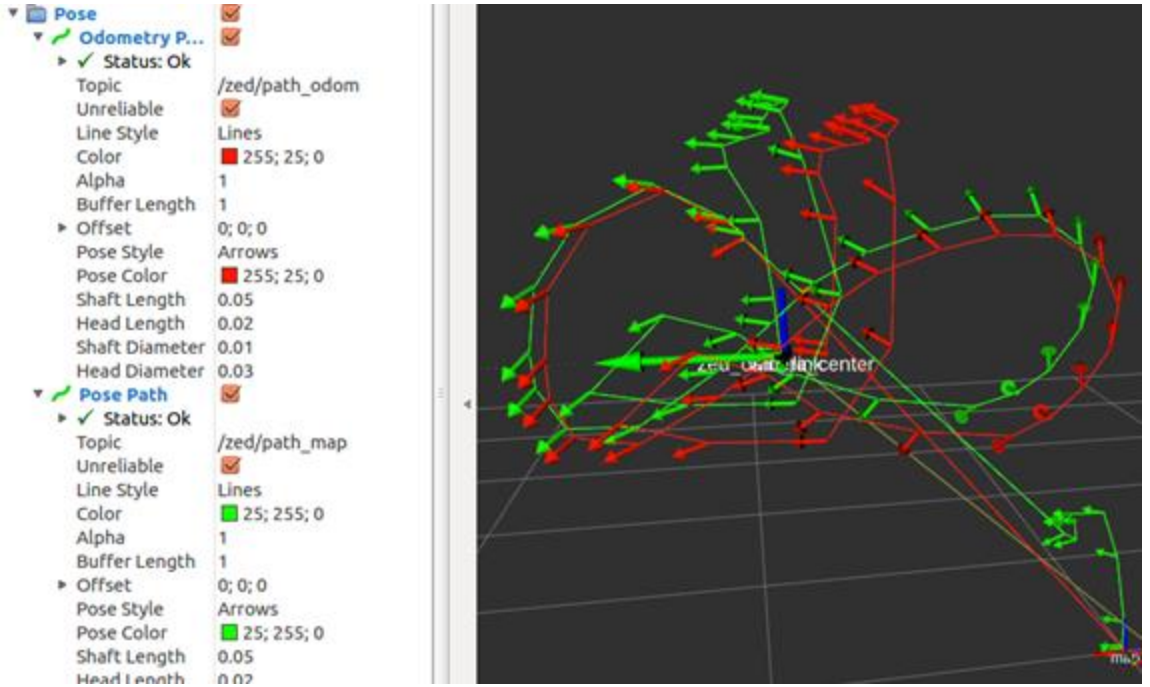


Hardware Stack

- Jetson Orin: Ubuntu 22.04, ROS1, ZED SDK 4.0
- RP2040 microcontroller:
  - Receives velocity and angular position in range [-1, 1]
  - Outputs PWM signal
- Steering: (-30, 30) degrees

Perception: The ZED Camera

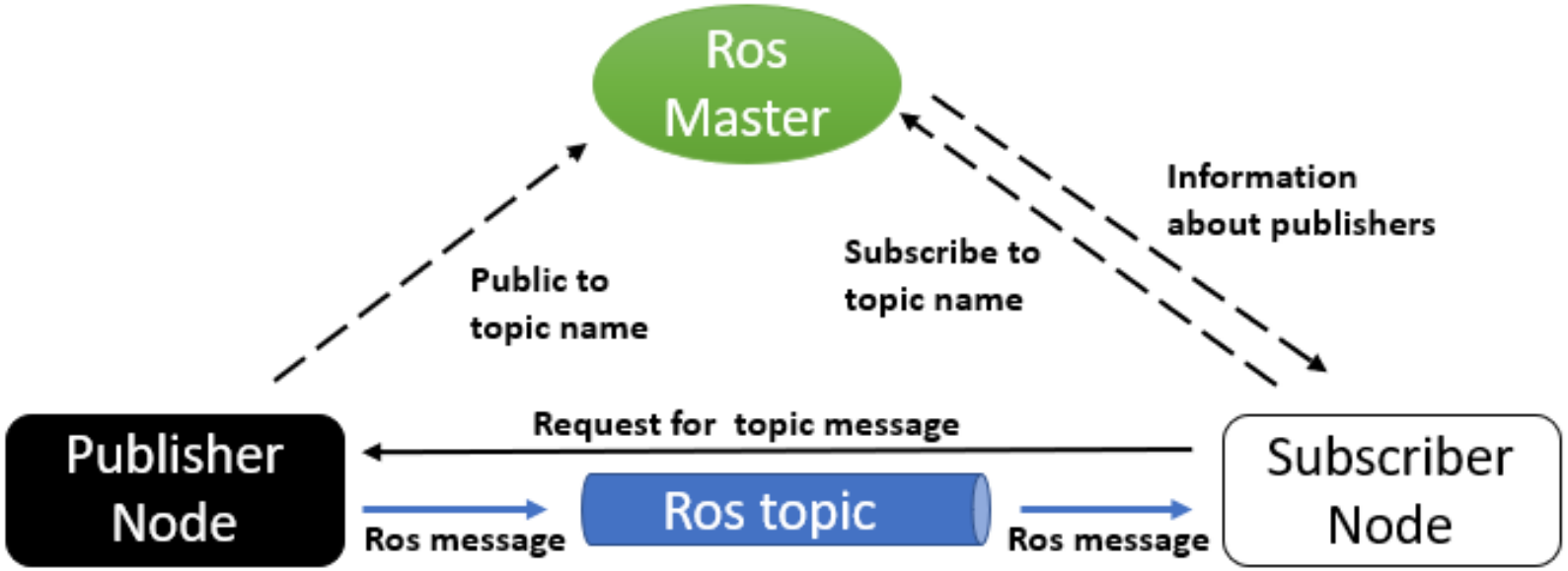




Plot of Odometry Path Data. Credit: Stereolabs

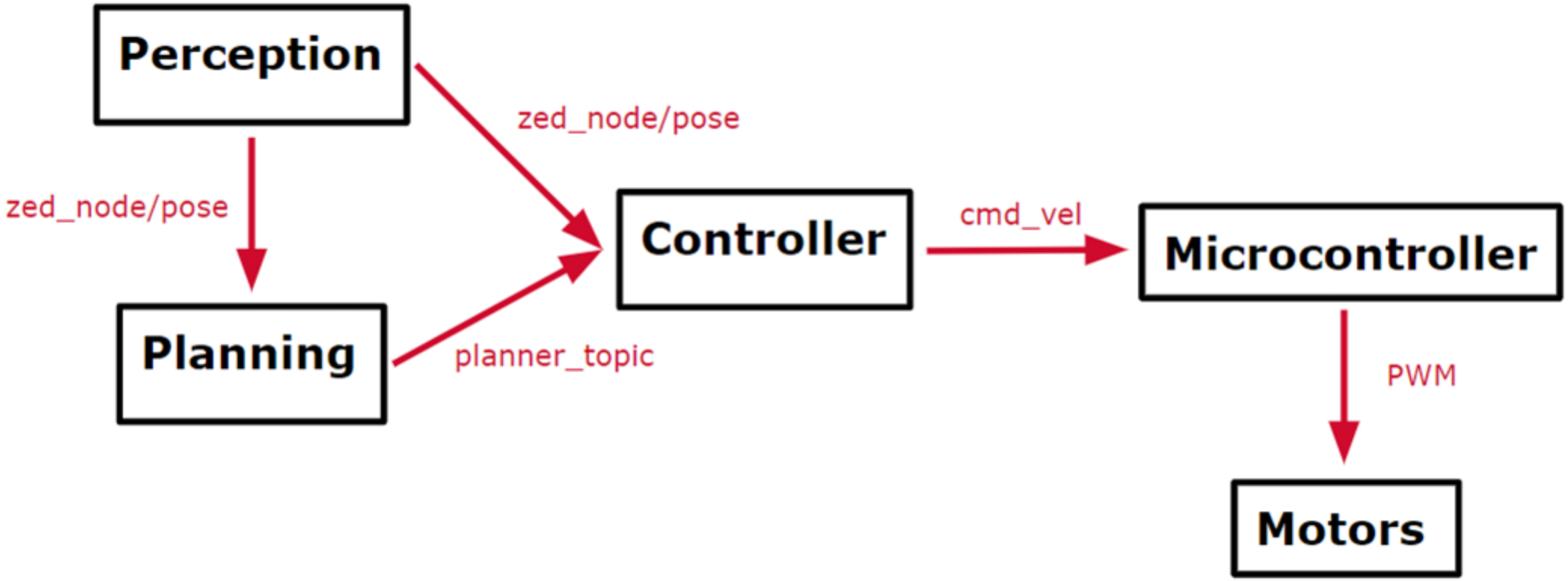
Robot Operating System (ROS)

A set of libraries that allows a distributed network of components to communicate.



Note diagram taken from https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system%28ros%29/

Software Stack



Software Stack - Motion Planning

Planning algorithm must output a series of Pose objects corresponding to planned path in the **map/ odometry frame**.

Pose: (x, y, z) position, orientation quaternion

- Algorithms are interchangeable

Need to simulate vehicle dynamics to plan path.

- **Dubin's Vehicle**

Dubin’s Vehicle

Dubin’s Paths

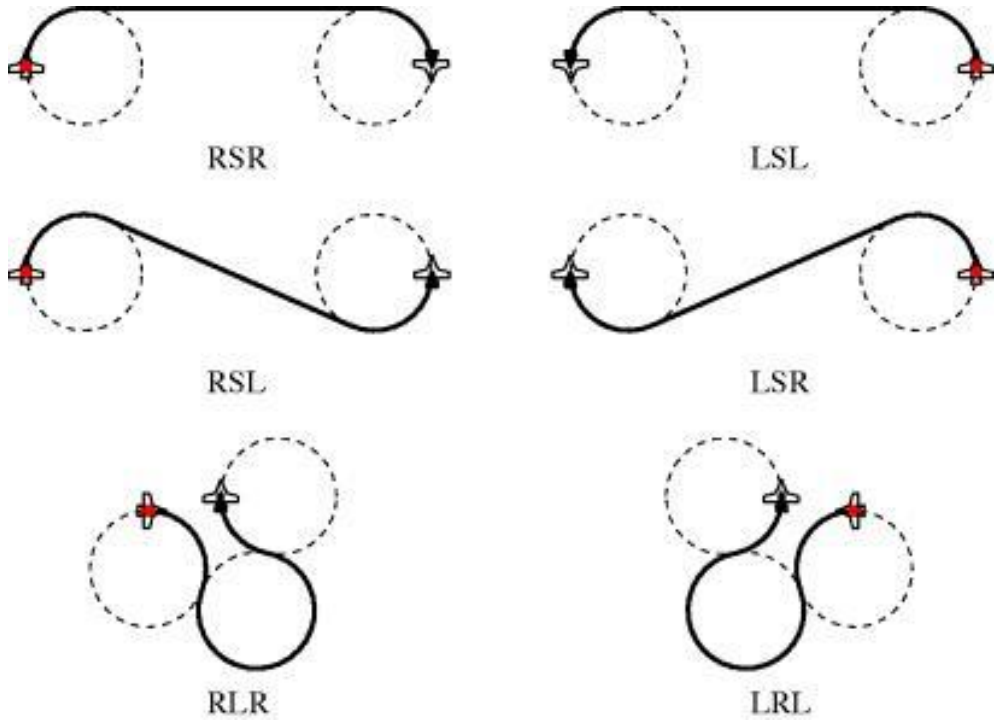
Family of curves describing shortest paths for a wheeled vehicle to navigate between two points while respecting the vehicle's minimum turning radius

Dubin's Vehicle

Kinematic car model with velocity and steering angle inputs.

Advantages

- Efficiency and simplicity: computationally efficient to calculate.
- Optimality: guarantee the shortest possible trajectory between two points for Dubin’s Vehicles



Note figure taken from https://github.com/ESchouten/Dubins

Software Stack - Controller

Using a simple two-dimensional control law to correct for error between perceived pose and desired pose.

Linear Velocity is proportional to the square root of displacement.

- Motion planning → **Odometry** Frame
- Position information → **Odometry** frame
- Commands to Publish → **Robot/ Body** Frame

Need to transform positions into **Body Frame**!

$$\tan\phi = -\frac{2L}{r_1}\sin\theta_1$$

L → Distance between front and rear wheels

r_1 → Distance between current and desired positions

θ_1 → Heading Error

ϕ → Optimal Steering Angle

Continuing Work

Continue testing controller

- Incorporate full perception stack into planning

Integrate more sophisticated planning algorithms

- Monte Carlo Tree Search (MCTS)
- Goal Oriented Recursive Path-Planning (GORP)

Upgrade controller

- Nonlinear Magic Controller

Acknowledgements

Many thanks to John Lathrop, Ben Rivière, Matt Anderson, and Professor Soon-Jo Chung for their mentorship.

Thanks to the Aerospace Corporation and Dr. Stupian for funding my work this Summer.