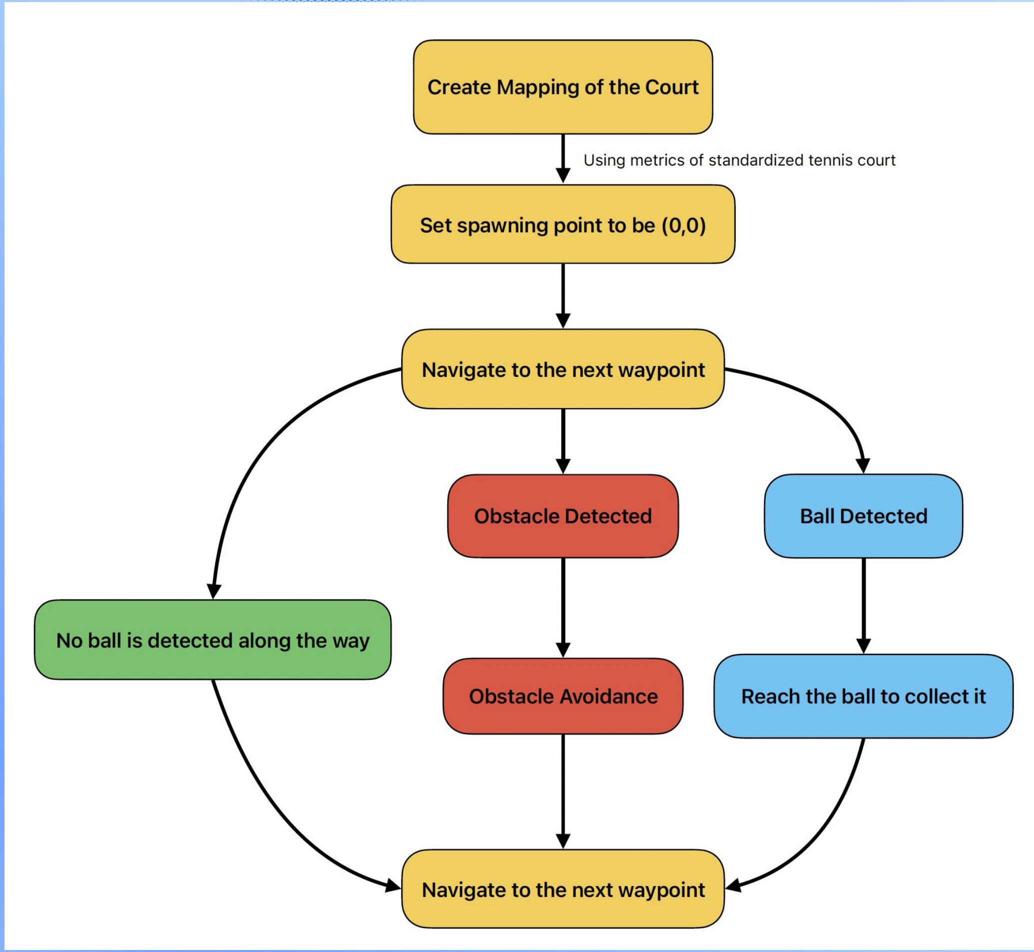
TENNIS BUDDY

BIG IDEAS

Tennis Buddy is capable of:

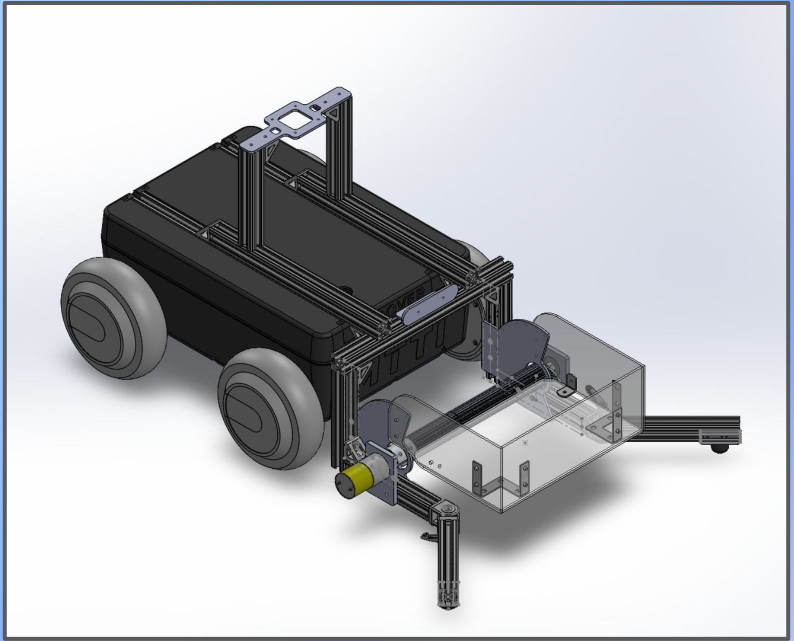
- Autonomously navigating around the court
- Detect and collect tennis balls
- Avoid fixed and dynamic obstacles around the court (net, ball holders, players)
- Ease player experience and maximize court time


Create Mapping of the Court
Using metrics of standardized tennis court
Set spawning point to be (0,0)
Navigate to the next waypoint
No ball is detected along the way
Obstacle Detected
Ball Detected
Obstacle Avoidance
Reach the ball to collect it
Navigate to the next waypoint

Hardware Overview

Robot Base: RoverRobotics MITI 65
Sensor: RPLidar S2/ IMU/ RealSense D435i
Compute: NVIDIA Jetson Nano
Front Ball Collector Module:

- **Aluminum-extrusion frame** with side plates for structural support
- **Motor-driven roller** to pull tennis balls into the robot
- **Hopper** for storing collected balls
- **Mounts for camera and LiDAR** for perception on the intake line
- **Front casters** that support the module and improve stability
- **Bolted interface** to the RoverRobotics base for easy removal/adjustment



Software Architecture

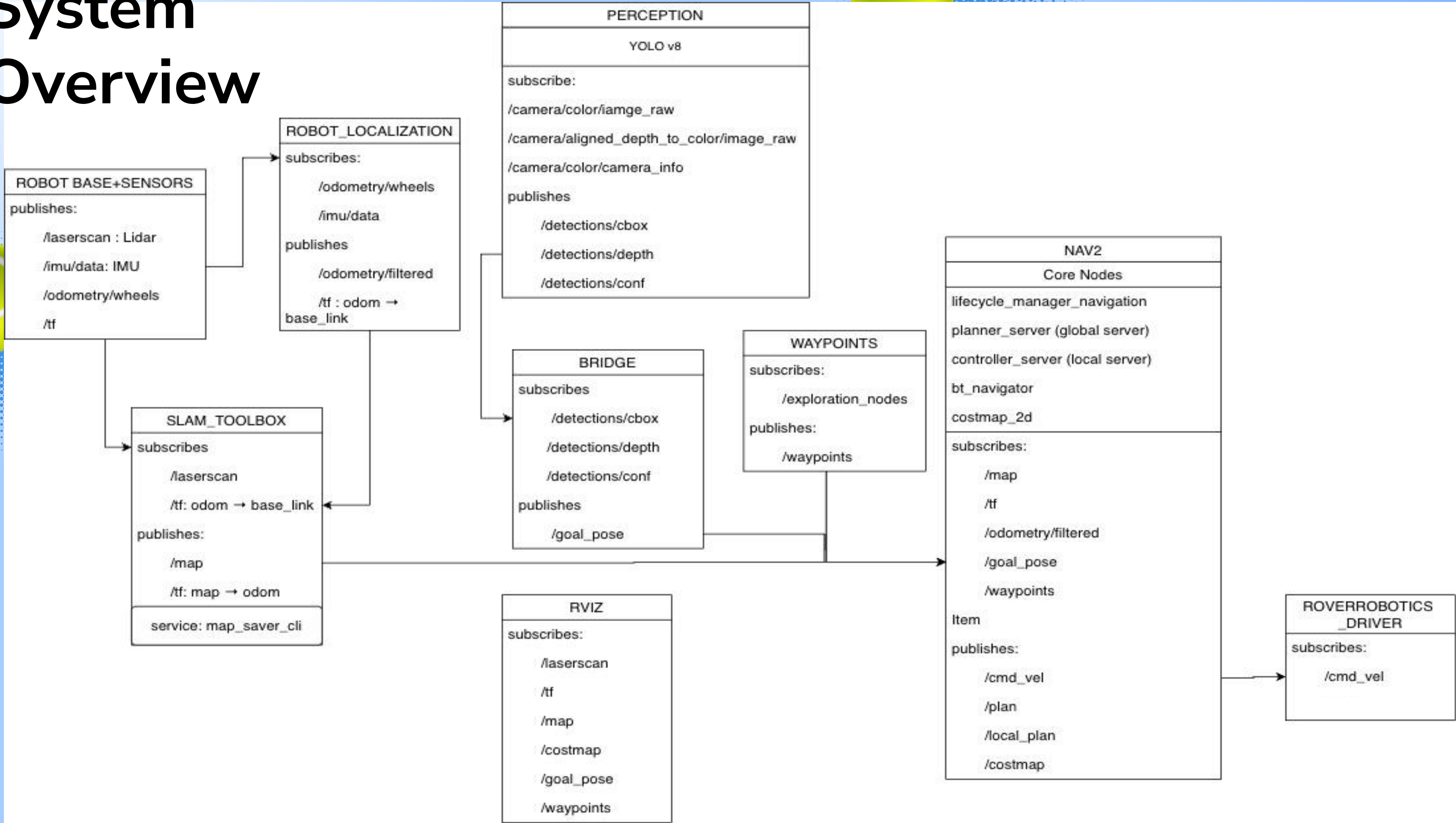
Goal: Enable Tennis Buddy to map the court, navigate safely:

- Follow predefined waypoints
- Reach and collect tennis balls detected by the camera.

Primary Processes:

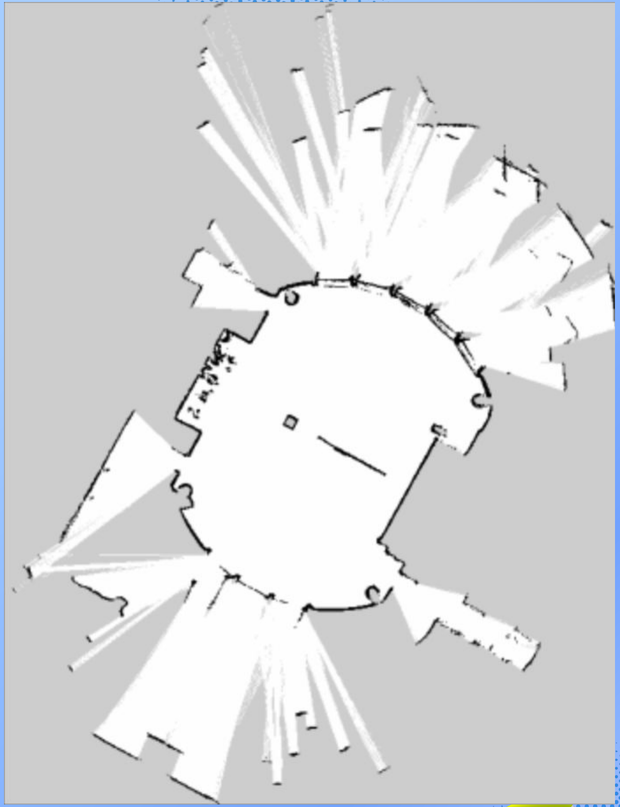
Map Generation: Mapping the testing space using IMU, Lidar, SLAM
Autonomy: path planning (Nav2)
Task Logic: State Machine Driven
Perception: RealSense Camera, YOLOv8 trained on tennis balls
InterNode Communication: ROS2 Msgs on: ball positions, ball-to-goal bridge, exploration with waypoints

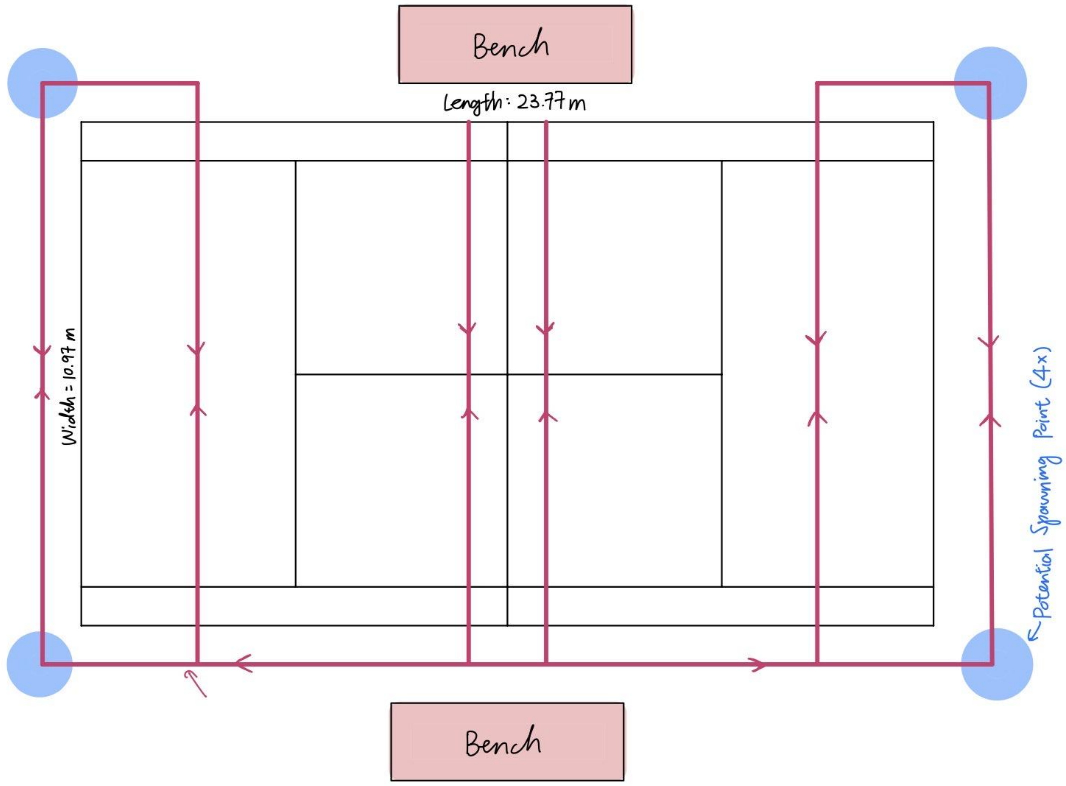
System Overview



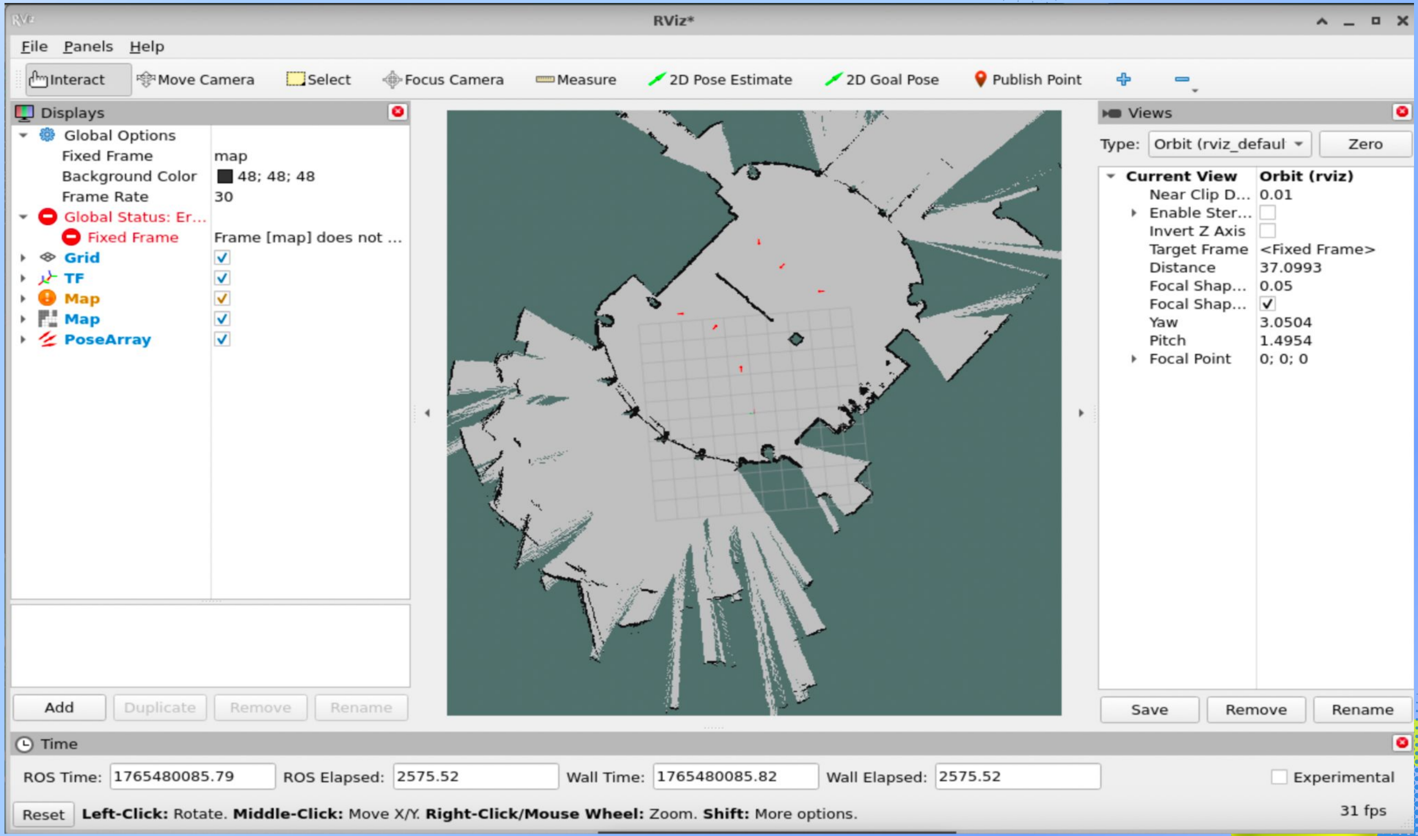
Mapping

The robot drives around a space once at startup to establish an initial map.
For each future run, the robot can be run from the corner of the court without any user calibration.




Bench
Length: 23.77 m
Width: 10.97 m
Potential Spawning Point (4x)
Bench

RVIZ: Visualization and Debugging



Planning

Bridge

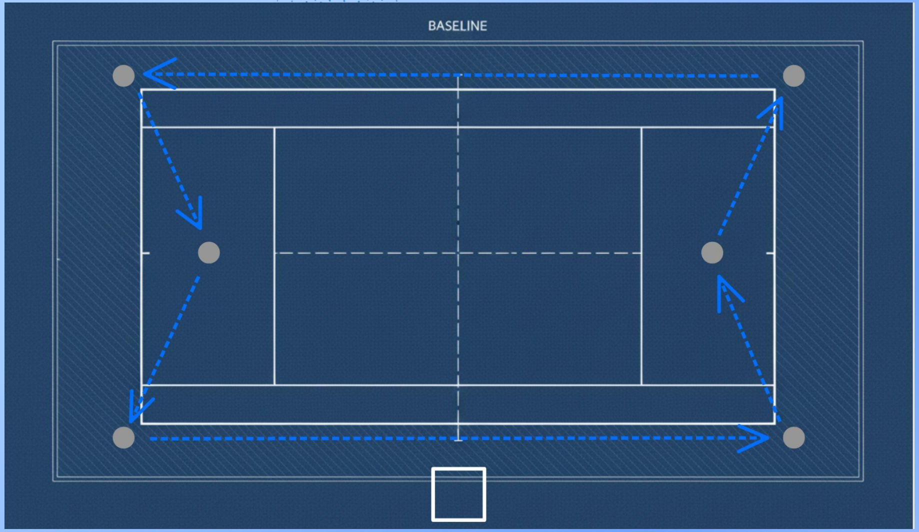
Subscribes the ball depth in camera_link

Transforms ball position into the global map frame using TF (camera_link → base_link → odom → map)

Outputs: ball positions (goal_pose) in map frame

Waypoints

waypoints -> scan -> ball_collector -> waypoints



The Goal: Move around the court and scan at each point to detect balls and collect them.

System Util Choices

CONTROLS/ NAVIGATION

To maximize the capacities of the Jetson, controls and navigation processing is done on the CPU.

[83%@1344,79%@1344,76%@1344,80%@1344,89%@1344,90%@1344] GR3D_FREQ 0% cpu@50.625C soc2@47.843C soc0@48.718C gpu@50.343C tj@50.687C soc1@50.687C VDD_IN 7644mW/7685mW VDD_CPU_GPU_CV

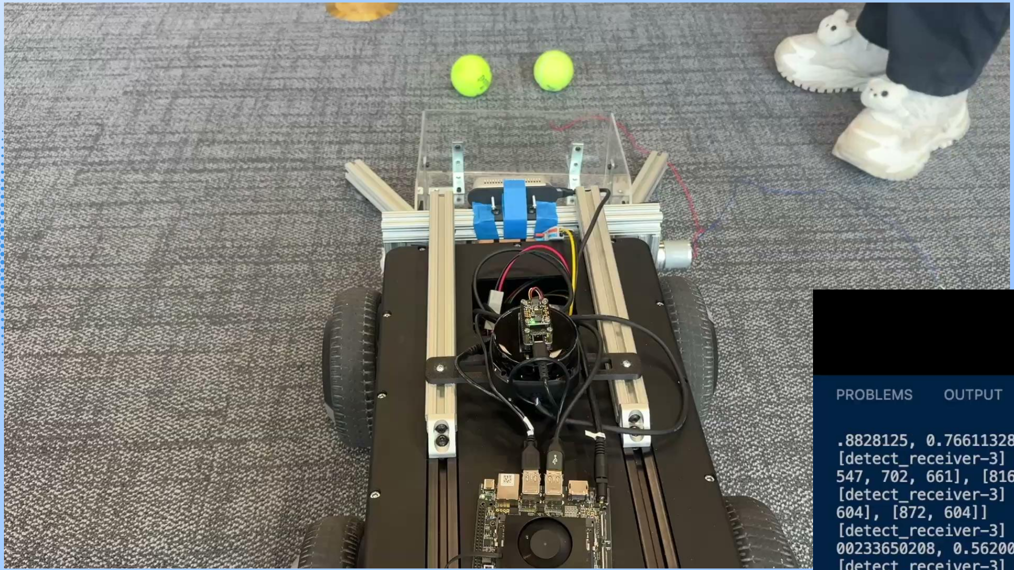
DETECTION MODEL

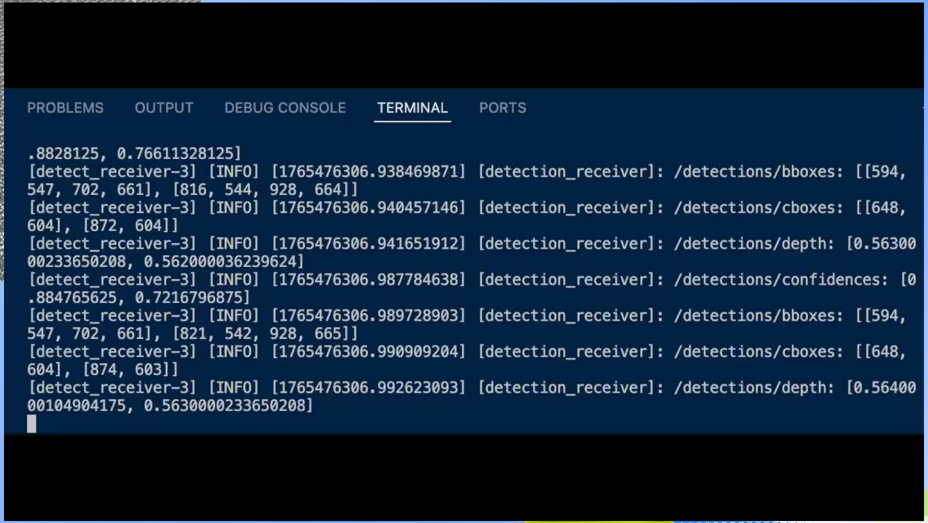
The model was moved to the GPU to improve inference speed and also allow CPU processing for the navigation pipeline.

RAM 4695/7621MB (lfb 5x4MB) SWAP 0/3810MB (cached 0MB) CPU [33%@1497,44%@1497,61%@1497,48%@1497,80%@1344,21%@1344] GR3D_FREQ 98% cpu@53.218C soc2@50.562C soc0@51.531C gpu@54.125C

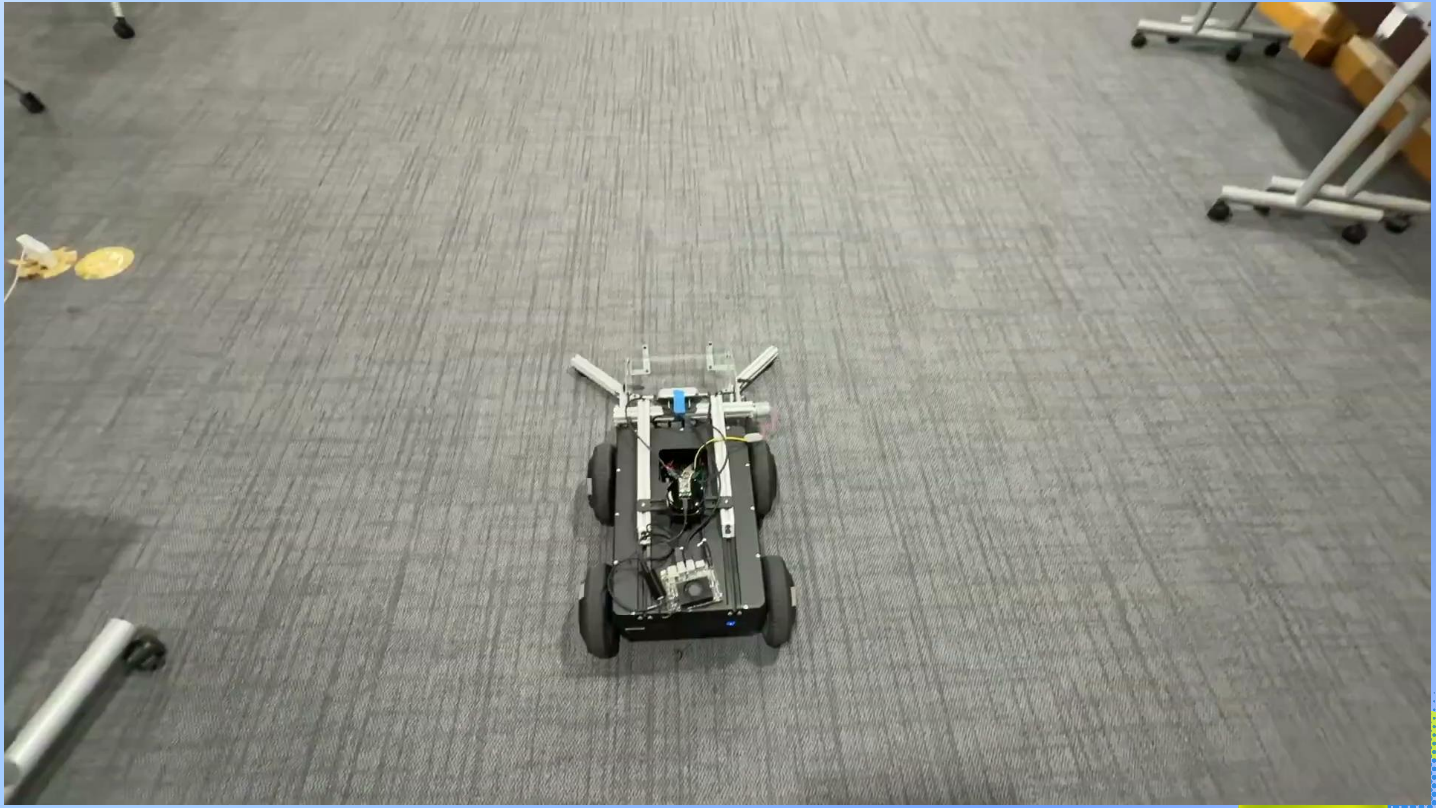
Conclusion: A major choke-point was that all systems were using the CPU, and the perception pipeline was causing behavioural issues for the planning and controls pipeline.

Perception




PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
.8828125, 0.76611328125]
[detect_receiver-3] [INFO] [1765476306.938469871] [detection_receiver]: /detections/bboxes: [[594, 547, 702, 661], [816, 544, 928, 664]]
[detect_receiver-3] [INFO] [1765476306.940457146] [detection_receiver]: /detections/cboxes: [[648, 604], [872, 604]]
[detect_receiver-3] [INFO] [1765476306.941651912] [detection_receiver]: /detections/depth: [0.5630000233650208, 0.562000036239624]
[detect_receiver-3] [INFO] [1765476306.987784638] [detection_receiver]: /detections/confidences: [0.884765625, 0.7216796875]
[detect_receiver-3] [INFO] [1765476306.989728903] [detection_receiver]: /detections/bboxes: [[594, 547, 702, 661], [821, 542, 928, 665]]
[detect_receiver-3] [INFO] [1765476306.990909204] [detection_receiver]: /detections/cboxes: [[648, 604], [874, 603]]
[detect_receiver-3] [INFO] [1765476306.992623093] [detection_receiver]: /detections/depth: [0.5640000104904175, 0.5630000233650208]

Basic Methodology



Pre-Demo

Demo

Why Ball Collecting So Hard?

- Yolo accuracy degrades when map is larger, and when the robot is dynamical moving (with only one camera)
- FOV with one camera is limiting
- Nav2 path planning success rate decreases in tennis map (obstacles with two narrow passages aside).

Future

Next Steps:

- Move to Jetson AGX Orin (flashed)
- Add dense waypoint coverage of the court
- Exploration mode without a pre-loaded map
- User interaction & configuration tools
- Player aware operation modes (no players/ players on court)
- Increase number of cameras

Further Features for Expansion:

- Holders for lifting camera and LiDAR
- Return balls to players
- Support for all types of tennis courts

THANKS!