Proprioceptive Vegetation Navigation

David Seong, Weichen Qiu, Benny Zhu

04/23/2025

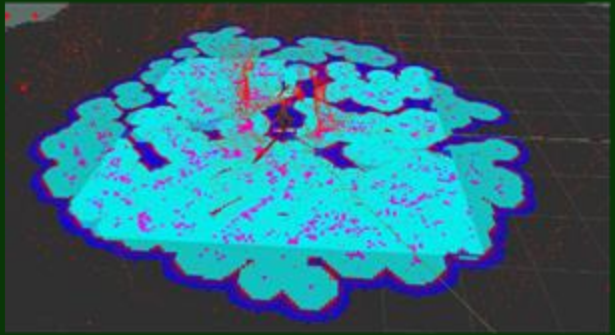
[1] Motivation

- Vegetated areas have lots of grass, bushes, etc. those *look* like obstacles BUT are traversable

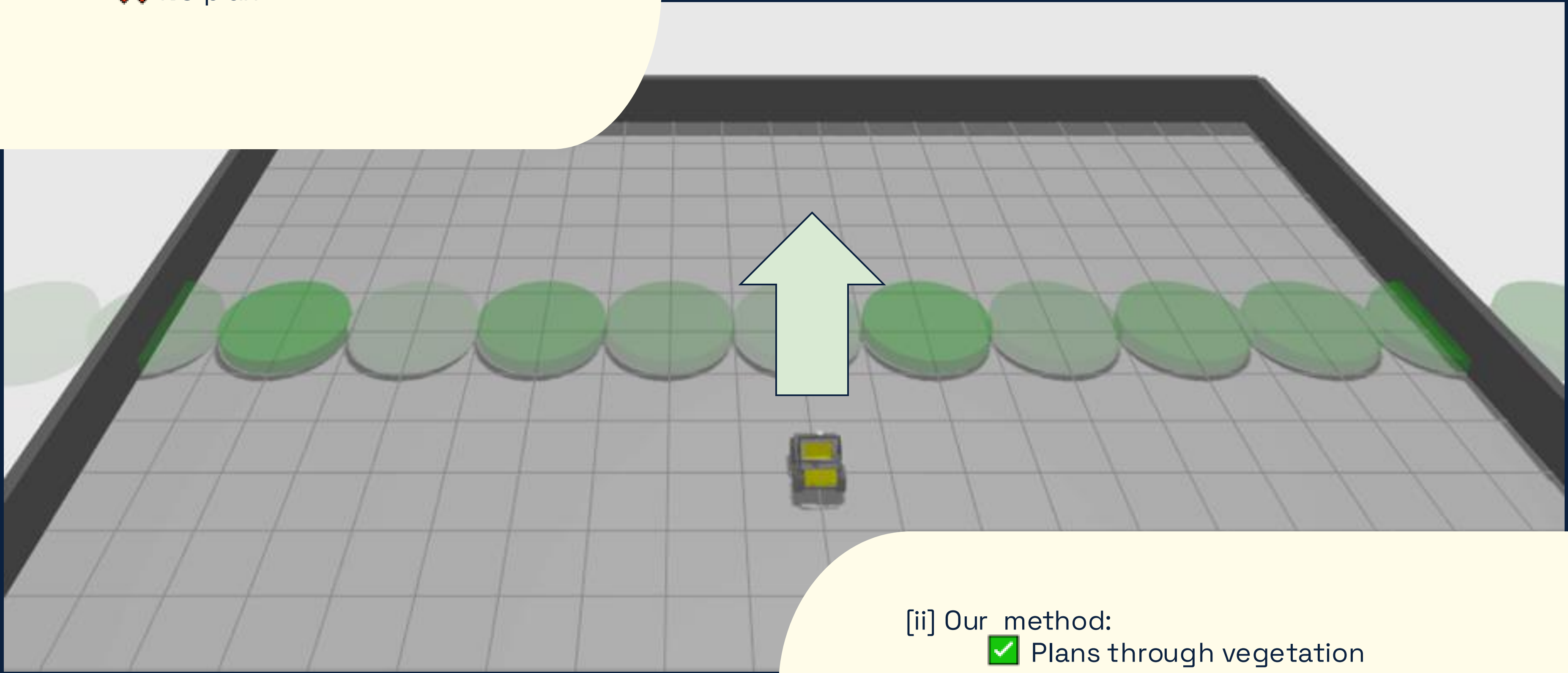
- **Traditional methods:**
  - All obstacles are <u>untraversable</u>

vs.

- **Our method:**
  - Assume all obstacles are <u>traversable</u> with initial guess on cost (i.e. cost(trees) > cost(grass))
  - Update initial guess upon collision



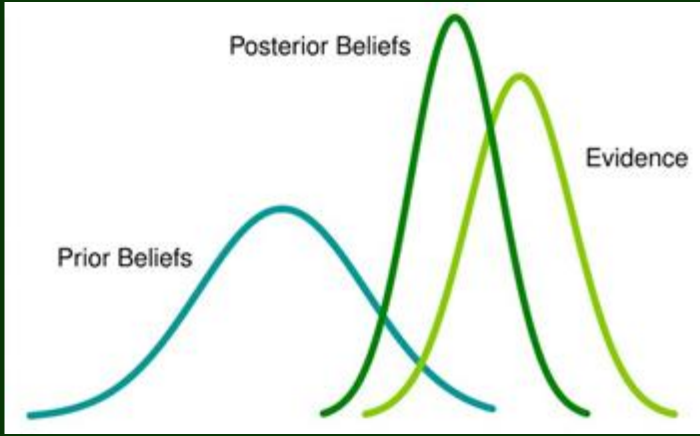
[i] Traditional methods:
❌ NO plan



[ii] Our method:
✅ Plans through vegetation

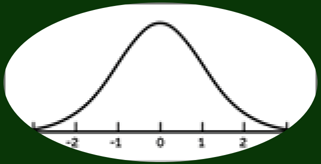
[2] The math

- **State:** $(x, y, \theta, costmap)$
- **Goal state:** Goal pose and updated costmap
- **Cost Function:** $f(state) = g(state) + \varepsilon * h(state)$
- **Heuristic:** euclidean distance
- **Map Cost:** desired_velocity - actual_velocity
    - Where desired_velocity=1m/s
    - If equal, 1-1 = 0 low cost
    - And 1-0 = 1 high cost



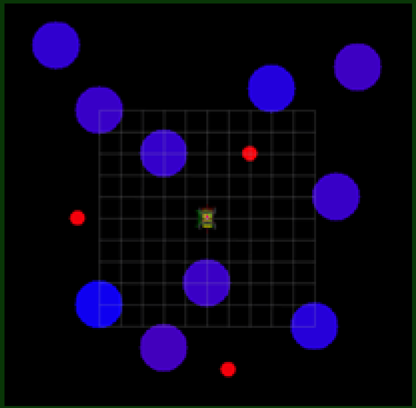
- **Bayesian Updating:**
- $\mu\_post = (\mu\_prior \cdot P\_prior + \mu\_obs \cdot P\_obs) / (P\_prior + P\_obs)$
    - Where post=posterior, obs=observation
    - $P\_prior = 1/(\sigma\_prior^2)$
    - $P\_obs = 1/(\sigma\_obs^2)$

[3] The algorithm

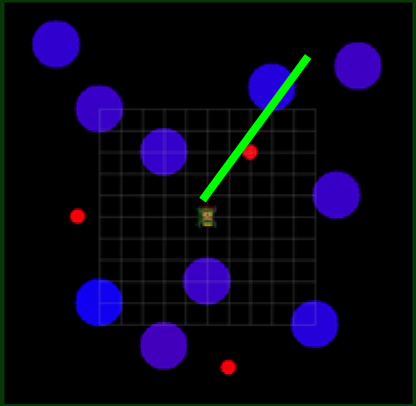


Prior cost knowledge

Gazebo obstacle locations



Initial costmap



Plan to goal

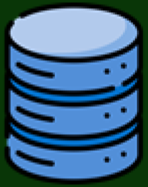
A*, D* Lite

[3] The algorithm

Collision with
vegetation

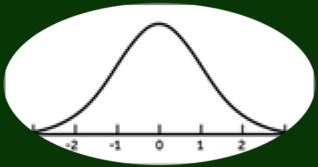
Measure cost
from velocity

*Bayesian
updating*

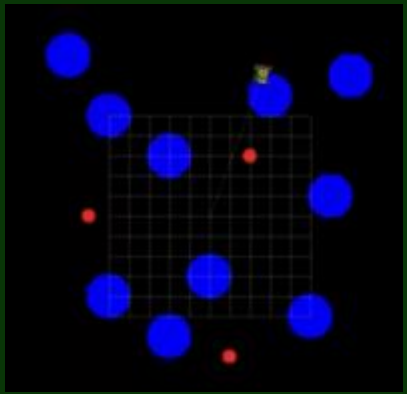


Update
database

Update the
obstacle's cost
& set stddev=0



Update costmap:
re-samples for
"similar" obstacles



Re-plan until
goal is reached

[4] Results

- Implemented an **end-to-end simulation** in ROS2 and Gazebo with Husky robot
    - Demonstrated successful planning and traversal through vegetation
- Developed resistance zones **contact model** using proprioceptive sensor
    - Applied specific velocity reductions in vegetated zones
    - Experimented with solid obstacle collision (e.g. stl trees, friction patches)
- Integrated custom **costmap** plugin into nav2 stack
- Experimented with A* and D* Lite **planning algorithms**

	D* Lite	**A***
Planning time	6000ms	6000ms
States expanded	23101	25462
Path Length	52	50