On Use of Nav2 Smac Planners

Steve Macenski, Samsung Research America
Steve Macenski

Senior Technical Lead - Samsung Research

- Your Friendly Neighborhood Navigator!
- ROS Technical Steering Committee Member
- Navigation Working Group & Project Lead
- Developed 50+ ROS & ROS 2 Packages

Simbe Robotics (2017 - 2019)
Samsung Research America (2019 - Present)
Overview

Background

Nav2 Smac Planners

In The Wild
Background - Path Planning

“How Do I Get There?”

Finding a route through an environment
- Could be feasible, but not definitionally
- Could be a trajectory, but not definitionally

Paired with a Local Trajectory Planner
- Costly traj. planning locally
- Path / route planning globally

When is it not needed?
- Predefined routes or ‘teach and repeat’
- Simplistic environments / following tasks
Background - Path Planning

Major Classes of Path Planners

Search
- Dijkstra’s, A*, State Lattice, SBPL

Sampling
- RRT & Variants, OMPL

Optimization and Smoothing
- Gradient Descent on Objective Functions
Background - Path Planning

Major Classes of Path Planners

Search
- Dijkstra’s, A*, State Lattice, SBPL

Sampling
- RRT & Variants, OMPL

Optimization and Smoothing
- Gradient Descent on Objective Functions

* Admittedly, this is a bit of a strawman example, but we only have 15 min!
Background - Path Planning

Major Classes of Path Planners

Search
- Dijkstra’s, A*, State Lattice, SBPL

Sampling
- RRT & Variants, OMPL

Optimization and Smoothing
- Gradient Descent on Objective Functions
What Options Did We Have in ROS (1)?

Navigation Stack
- NavFn
- Global Planner

The Community
- DLux Global Planner
- Voronoi Planner
- SBPL Lattice Planner

→ Plenty of circular diff and omni options
Background - ROS Planning

What Options Did We Have in ROS (1)?

Navigation Stack
- NavFn
- Global Planner

The Community
- DLux Global Planner
- Voronoi Planner
- SBPL Lattice Planner

→ Plenty of circular diff and omni options

No teach & repeat or pre-defined route following? (though not the topic of today’s talk)
What Options Did We Have in ROS (1)?

Navigation Stack
- NavFn
- Global Planner

The Community
- DLux Global Planner
- Voronoi Planner
- SBPL Lattice Planner

→ Plenty of circular diff and omni options

No teach & repeat or pre-defined route following? (though not the topic of today’s talk)

Where’s the support for non-circular robots?
Background - ROS Planning

What Options Did We Have in ROS (1)?

Navigation Stack
- NavFn
- Global Planner

The Community
- DLux Global Planner
- Voronoi Planner
- SBPL Lattice Planner

→ Plenty of circular diff and omni options

No teach & repeat or pre-defined route following? (though not the topic of today’s talk)

Where’s the support for non-circular robots?

What about Ackermann or Legged robots?
Overview

Background

Nav2 Smac Planners

In The Wild
Smac Planner - Overview

Cost-Aware A*-Based Planning Framework

Feature-Packed Templated A* Search Algorithm

Multiple Node Types, Creating 3 Unique Planners

Enables Non-Circular, Legged, Ackermann, Diff & Omni

Optimized for High Performance; Drop-In Replacement

93% Unit Test Coverage, Used in Production Today

```cpp
SmacAStar<NodeT>

while not queue.empty()
    node = getNextNode()
    if node->wasVisited()
        continue
    node->visited()
    node = tryAnalyticExpansion(node)
    if isGoal(node) or withinApproachTol(node)
        return backtrace(node)
    for neighbor in node->getNeighbors()
        g = node->getAccumulatedCost() + node->getTraversalCost()
        if g < neighbor->getAccumulatedCost()
            neighbor->setAccumulatedCost(g)
            neighbor->parent = node
            addNode(g + neighbor->getHeuristicCost(), neighbor)
```
Support for New / Modern Robot Types

Alternative for Circular or Small Robots (2D)

Non-Circular or Large Diff / Omni (Lattice / Hybrid)
- Where a circular assumption is not possible

Legged or Ackermann (Lattice / Hybrid)
- Curvature constrained, kinematically feasible
- Arbitrary models for bespoke systems

Nav2 Supports All Major Robot Types
Smac Planner - Some Important Technical Deets

Cost-Aware Obstacle Heuristic
- Steers towards solution, away from obstacles
- Uses cost, not just binary obstacles
- Respect user behavioral constraints
- Higher quality path before smoothing

Search Penalty Functions
- Reverse, Change Direction, Non-Straight, Cost

Analytic Expansions
- Finds exact & feasible paths to the goal
Smac Planner - 2D A*

Circular Diff / Omni

Performance: 20 - 200 ms

Simple Grid-A* with Smoothing

Point-Cost Collision Checking

Moore Search Model

L2 Distance Heuristic

Consistent Behavior in Heterogeneous Fleet

\[ \text{cost}_{\text{trav.}} = L_{\text{prim}} \times (1 + \frac{w_{\text{cost}} \times \text{cost}_{i,j}}{\text{cost}_{\text{max}}}) \]
Smac Planner - Hybrid-A*

Ackermann, Legged

Performance: 20 - 60 ms

SE2 Pose Collision Checking

Kinematically Feasible

Dubins or Reed-Shepp Search Model
  ● Dynamically adjusted
  ● + Analytic Expansions

Heuristic is the Maximum of:
  ● Precomputed Kine-Distance Window
  ● Cost-Aware Obstacle Heuristic
Smac Planner - State Lattice

Non-Circular Diff / Omni, Arbitrary

Performance: **30 - 65 ms**

SE2 Pose Collision Checking

Kinematically Feasible

Minimum Control Set Search Model
- Generated offline
- + Analytic Expansions

Heuristic is the Maximum of:
- Precomputed Kine-Distance Window
- Cost-Aware Obstacle Heuristic
Minimum Control Set Generator

Creates set of primitives to describe motion model
- In a structured & principled lattice pattern
- Primitives smoothly transition from one to another

For each cell + heading in a wavefront:
→ Create a curvature minimizing trajectory
→ Check if a similar traj. can be constructed from set
→ If not, add it to the set
Repeat until a wavefront adds no new primitives
Overview

Background

Nav2 Smac Planners

In The Wild
Configuration

See Nav2 Docs For Full List

Create Potential Fields

Cache Obstacle Heuristic
- For consecutive replanning in static spaces
- Less than 10 ms replans typical

Cost Penalty
- Cost: Penalizes higher cost areas*
- Reverse: Penalizes going in reverse
- Change: Penalizes not continuing last action
- Non-Straight: Penalizes non-straight actions
- Rotation: Penalizes pure rotations (Lattice only)

* Shared with cost-aware obstacle heuristic
In The Wild
Repository, Documentation, and Issue Tracker:
https://github.com/ros-planning/navigation2
https://discourse.ros.org/c/navigation
https://navigation.ros.org

Interested in Getting Involved?

Steve Macenski
Senior Technical Lead - Samsung Research
s.macenski@samsung.com
stevenmacenski@gmail.com
"His path-planning may be sub-optimal, but it's got flair."
Configuration

See Nav2 Docs For Full List

Termination Conditions
● Max Planning Time / Iterations
● Planning Tolerance / Iterations on Approach

Downsample Costmaps

Analytic Expansion
● Ratio
● Maximum Length

Angular Quantizations
Smac Planner - Features

Optimized
- Carefully selected data structures
- Tons of precomputation and caching

Dynamic Graph Creation
- Constructs graph on expansion
- No run-time lookups
- **NEW** ~25% speed-up with Robin Hood Hashing

Costmap Downsampling

Smoothing Options
- Basic gradient descent
- Collision & curvature constrained

Approximate Paths W/In Tolerance

Collision Checking
- Checks if center cost is less than min possibly inscribed
  - If not, checks full SE2 footprint
  - If circular, checks center inflated costs

Analytic Expansions
- Uses motion model to find exact path to goal
- Computed more frequently closer to the goal
- Significantly speeds up on approach to goal