Lecture 9
ROB-GY 7863 / CSCI-GA 3033 7863: Planning, Learning, and Control for Space Robotics

Benjamin Riviere

November 11, 2025

Logistics

- Midterm feedback reports:
  - Thank you for filling these out!
  - **TODO:** discussion
- Project 2 Deadlines:
  - Feedback on Proposals will be this week.
  - Final presentations: December 8th

Space Culture

- We will do a case study based on technical discussion later instead.

Recap

- Convergence of Kalman Filter
- Simultaneous Estimation and Control
- LQG Coding Exercise

Agenda

- Search-Based Planning
  - Local Minima
  - A*, RRT, RRT*
  - Software tools
  - Example: Rover path planning
  - Break
  - Seminar on MCTS for Robotics

Local Minima I

- Recall the optimal control / planning problem (P1):

$$\min_{X,U} \sum_{k=1}^{K} c(x_k, u_k) + d(x_K) \tag{1}$$

$$\text{s.t. } x_k = f(x_{k-1}, u_k), \quad \forall k \in [1, K] \tag{2}$$

- Sequential convexification optimization about linearization point $\bar{X}, \bar{U}$ (P2):

$$\min_{X,U} \sum_{k=1}^{K} x_k^T Q x_k + u_k^T R u_k + x_K^T Q_f x_K \tag{3}$$

$$\text{s.t.} \quad x_k = A_k x_{k-1} + B_k u_k \tag{4}$$

Local Minima II

- The guarantee we previously analyzed is that our algorithm iterates converge to a local minima of P2:

$$Z_{n+1} = Z_n - \alpha \nabla_Z L(Z_n, \lambda_n) \tag{5}$$

$$\lambda_{n+1} = \lambda_n + \beta \nabla_\lambda L(Z_n, \lambda_n) \tag{6}$$

$$\lim_n \nabla_Z L(Z_n) = 0 \text{ (and rest of KKT conditions.)} \tag{7}$$

where $Z = [X, U]$

- **TODO:** show picture of bugtrap local minima.

A* Search

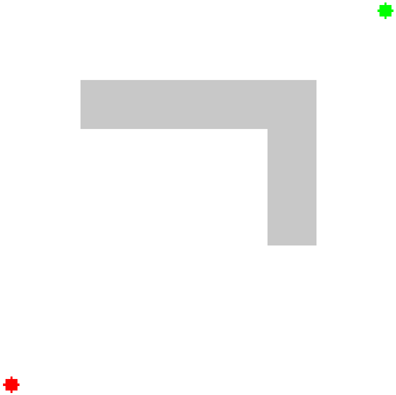


Figure: A* Search

Brief History of A*

- Invented for robot path planning in 1968 (**XX**)
- Refined analysis in 1985 (**XX**)
- Huge impact, e.g. Dijkstra/A* algorithms used in Google Maps (**XX**)
- Some robotics researchers are still using A* and proposing improvements (Saxena et al., 2022)

A* Algorithm

Data Structures and Functions

- Start node and goal node
- O: ("open-set") node frontier
- $d(n_1, n_2)$ is the cost from neighboring node n_1 to node n_2.
- $g(n)$: ("cost-to-come") measures actual cost from start node to node n
- $h(n)$: ("heuristic cost-to-go") guess of cost from node n to goal node.
- $f(n) = g(n) + h(n)$: ("f score")

Pseudocode of one iteration:

- remove node n from open set O with the lowest f score
- if n is the goal, return n.
- else, for each neighbor n' of node n: (i) update cost-to-come $g(n') = \min g(n'), g(n) + d(n, n')$, update f-score, add to open-set.
- **TODO:** draw a few iterations on board

A* Analysis Results

- Assumptions:
  - Admissible: $h(n) \leq h^*(n), \forall n$. Heuristic does not overestimate the true optimal cost to goal.
  - Consistent: $h(n) \leq c(n, n') + h(n')\ \forall (n, n') \in E$. Heuristic obeys triangle inequality between nodes. (consistency implies admissibility)
- Results:
  - Complete: If there exists a path from start to goal, A* will eventually return it.
  - Optimal: The path returned by A* will be a lower cost than all other solutions
  - Efficiency: If two algorithms are given the same heuristic, A* will return the optimal solution in less than or equal number of node expansions.

A* Discussion

- **TODO:** discussion: what are challenges with using A* to control, for example, a rover on Mars?

Rapidly Exploring Random Trees (RRT)

Figure: RRT Search

Brief History of RRT

- First proposed by Lavalle and Kuffner in 1999 (**XX**).
- Hugely influential in Robotics (mobile, manipulation, humanoids), biology drug design, manufacturing, virtual prototyping, verification and validation, computer animation, aerospace.
- RRT*, asymptotically optimal convergence by Karaman and Frazolli in 2011 (Karaman et al., 2011).

RRT Algorithm

Data Structures and Functions

- Start node and goal set
- T: ("tree") tree of nodes
- p: sampling function (e.g. uniform over $[0, 1]^n$)
- d: distance function (e.g. $d(s_1, s_2) = \|s_1 - s_2\|$).
- π: steer function (e.g. $\pi(s, s_g) = -K(s - s_g)$)
- **TODO:** draw a few iterations on board

Pseudocode of one iteration:

- sample a state in the space $s_{\text{sample}} \sim p(S)$
- select the closest node in the tree to the sampled state $s_{\text{select}} = \min_{s \in T} d(s, s_{\text{sample}})$
- steer the selected node to the sampled node: $s_{\text{steer}} = \pi(s_{\text{select}}, s_{\text{sample}})$
- if s_{steer} in goal set, return path from s_{start} to s_{steer}.
- if collision free, add steered node to tree $s_{\text{steer}} \in T$.

RRT Analysis

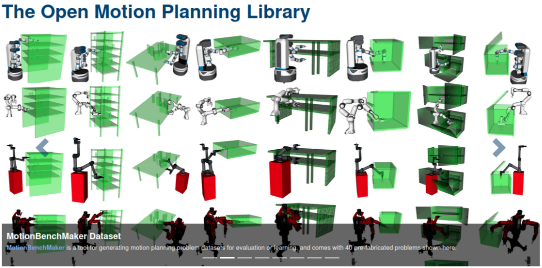
- Probabilistic completeness: With high probability, if there exists a path, RRT will return one eventually.
- Asymtotic optimality (for RRT*): as the number of nodes in the tree goes to infinity, the cost of the path is globally minimized.

Discussion

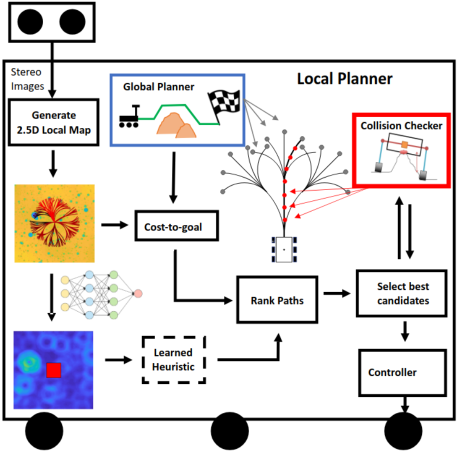
- Probabilistic Road Maps (PRM/PRM*) (Kavraki et al., 2002)
- Fast Marching Trees (FMT) (**XX**)
- **TODO:** discussion: what are issues with RRT?

Modern Software Tools for RRT and Variants

- MoveIt (Ros interface to OMPL and optimization-based approaches like STOMP)
- Open Motion Planning Library (2011) (OMPL)



Case Study: Rover Path Planning



- From (Daftry et al., 2022) and (Abcouwer et al., 2020)
- **TODO:** discuss sensors, navigation strategy, mid and low level planning etc.

Break

- Break

Seminar on MCTS

- Seminar on MCTS