COLUMBIA UNIVERSITY EEME E6911 FALL '25

TOPICS IN CONTROL : PROBABILISTIC ROBOTICS

NAVIGATION

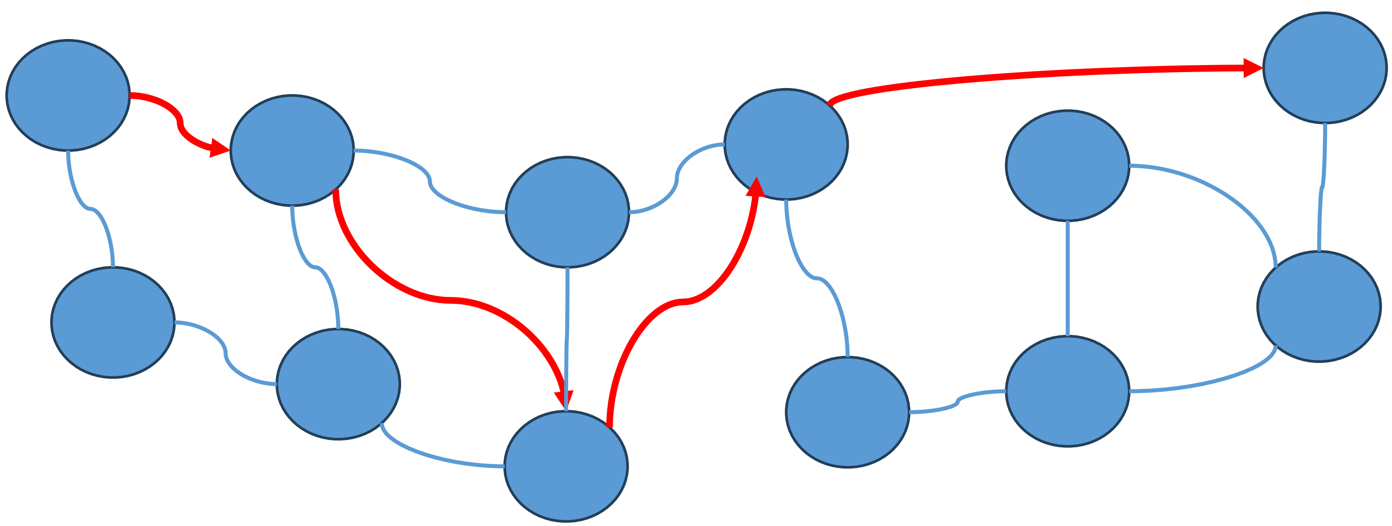
Instructor: Ilija Hadzic

Mobile Robotics Problems

- Localization
- Mapping
- Simultaneous Localization and Mapping (SLAM)
- Navigation
- Exploration

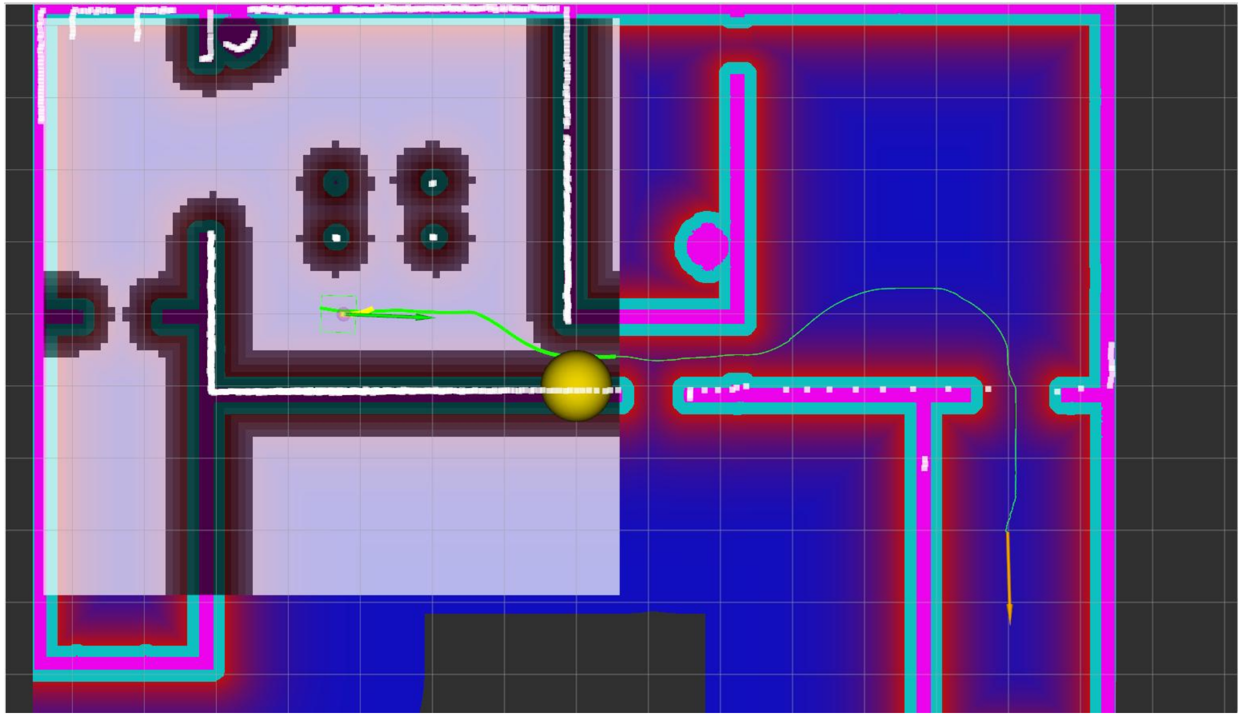
Navigation (Abstract)

- Generate input sequence to:
  - Transition the state from start to goal.
  - Optimize some predefined criteria.



Navigation (Mobile Ground Robots)

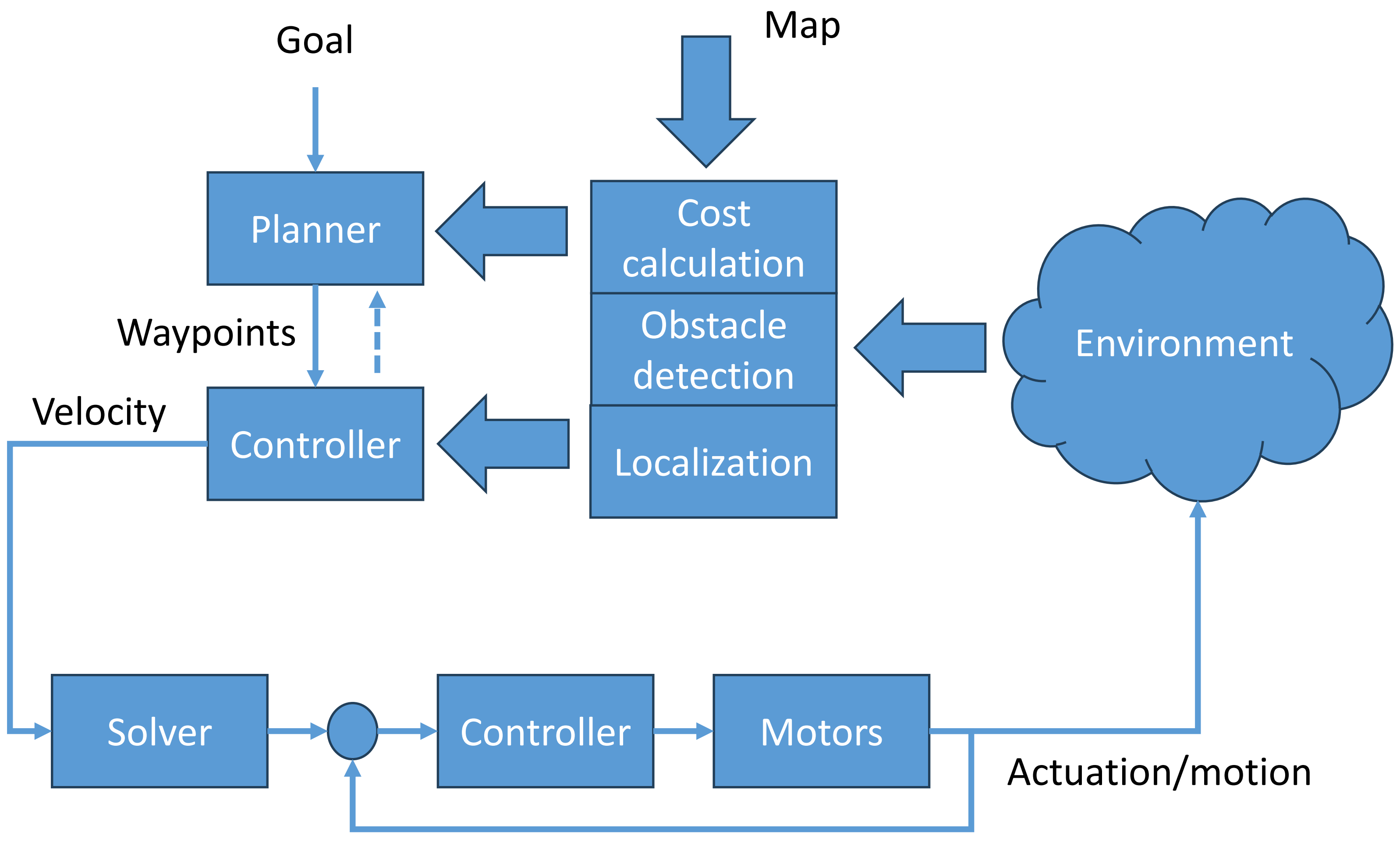
- Find a viable (optimal?) path from start to goal
- Generate controls to follow the path.
- Avoid obstacles



Planning and Control

- Planning:
  - Offline (non-real-time) process.
  - Generates waypoints.
  - Finds (optimal) path to the global goal.
  - Sometimes called Global Planning
- Control:
  - Real-time process.
  - Generates actuation.
  - Seeks to reach next waypoint
  - Split control:
    - Waypoint to velocity (Local Planning)
    - Velocity to actuation (Motion Control)

Planning and Control Loops



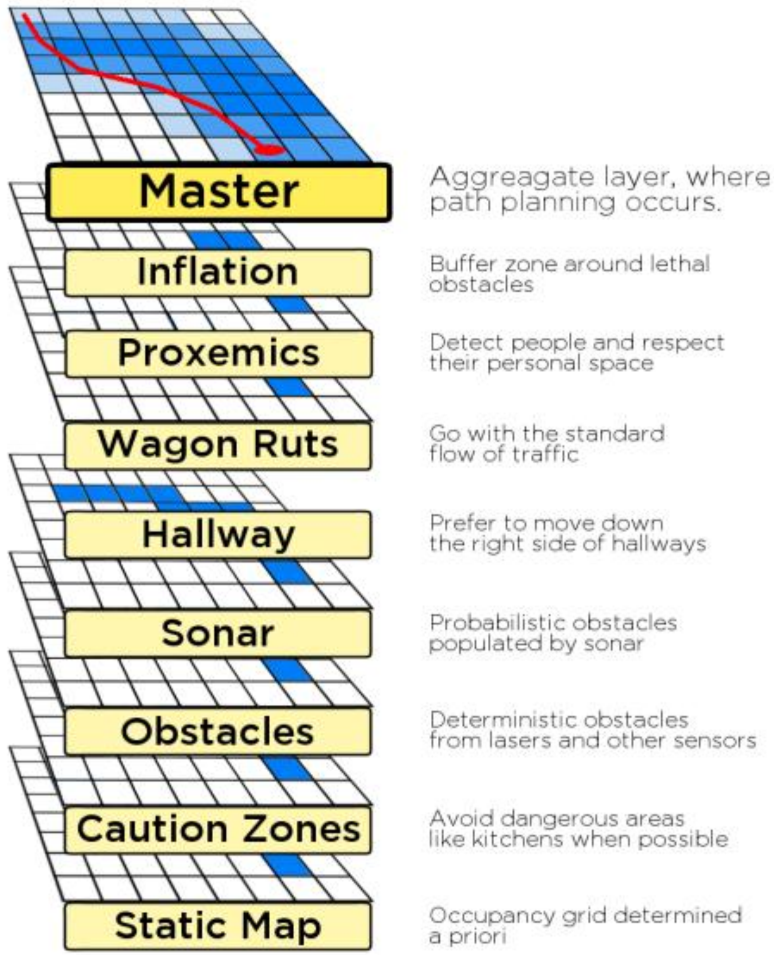
Planning

- Deterministic:
  - Costmap and goal available
  - Find best path
  - Replan if conditions change
  - Examples: A*, Dijkstra
  - Costmap inflation can (somewhat) factor in uncertainty
- Probabilistic:
  - Fully observable Markov Decision Process
    - State is certain, motion is not
    - Example: Value Iteration
  - Partially observable Markov Decision Process
    - Neither state nor motion are certain
    - Examples: QMDP, AMDP, MC-MDP

ROS Navigation Stack

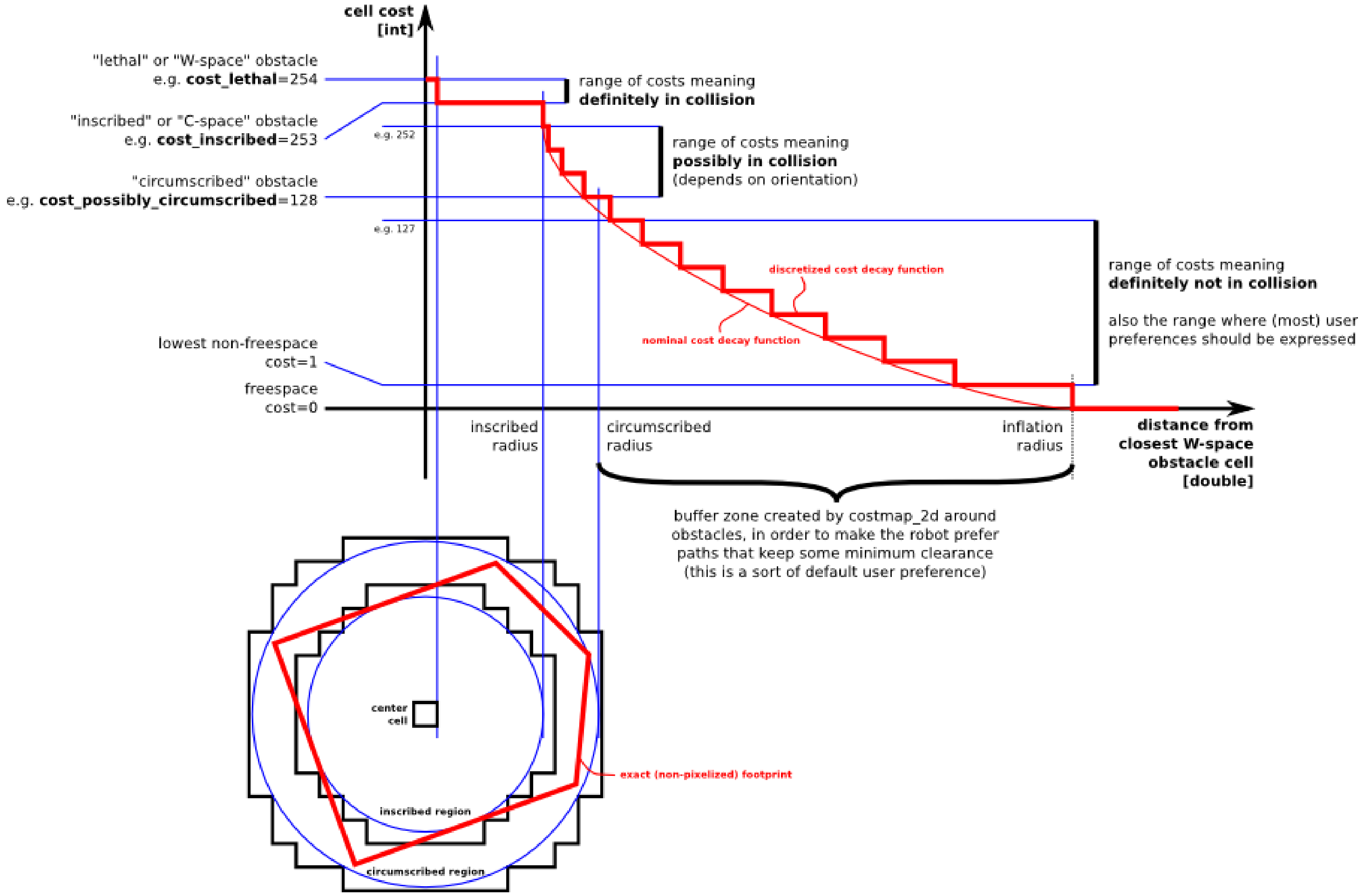
- Global Planner:
  - Dijkstra or A*
  - Input: Goal
  - Input: Layered Cost map
  - Output: Waypoints
- Local Planner:
  - DWA/DWB, MPPI
  - Input: Waypoints
  - Input: Layered Costmap
  - Output: Velocity

Layered Costmap



David Lu, et al., Layered Costmaps for Context-Sensitive Navigation, IROS2014

Inflation Layer



Source: `wiki.ros.org`

A*

$$f(n) = g(n) + h(n)$$

Cost of all previous hops

Heuristic estimate of cost to the goal (e.g, crow-flight distance)

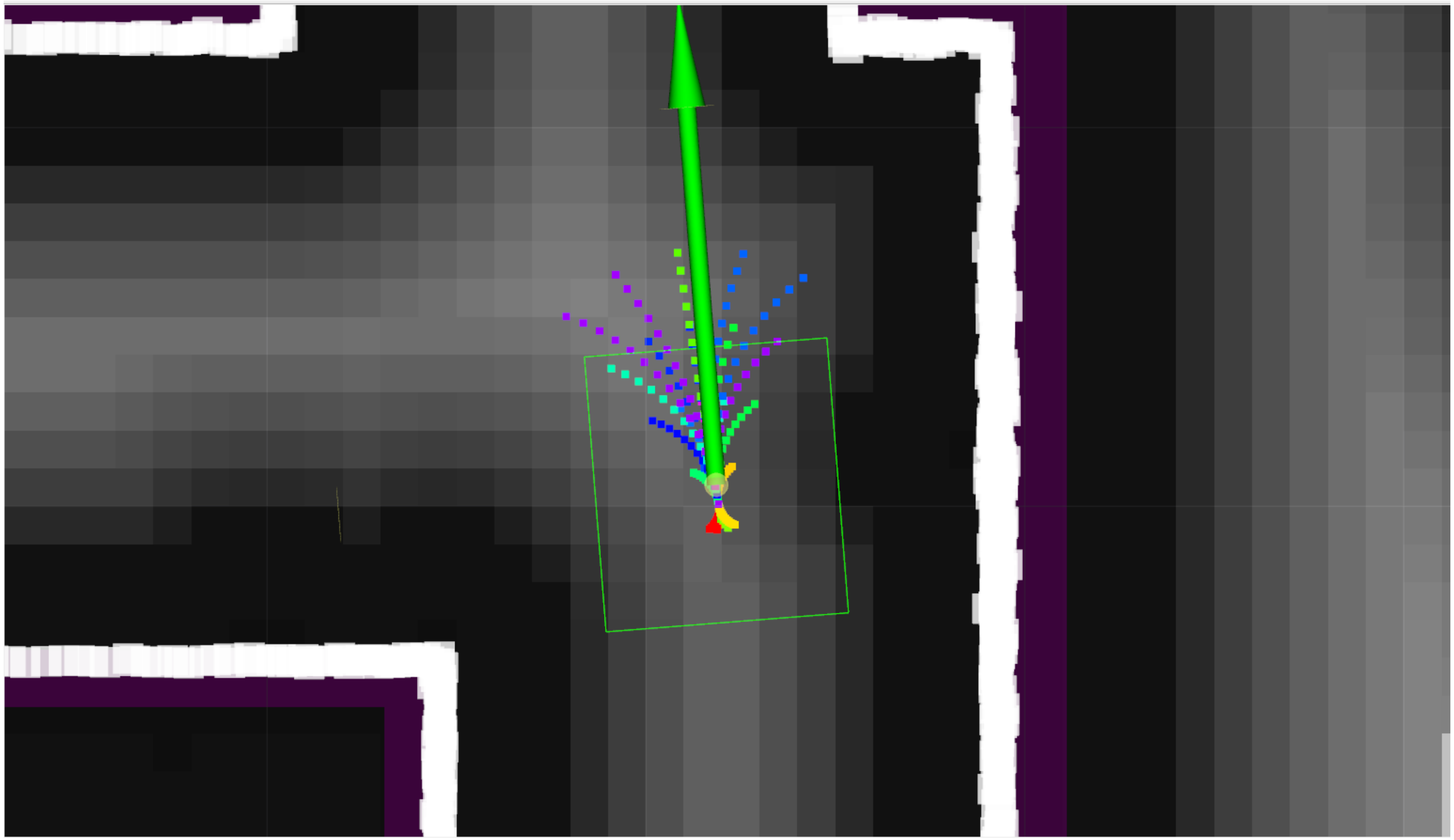
- Priority Queue:
  - Dequeue the node with the lowest cost $f(n)$
  - Expand to the neighbors, calculate new $f(n)$
  - Point to the previous node
  - Place it back in the queue

DWA/DWB

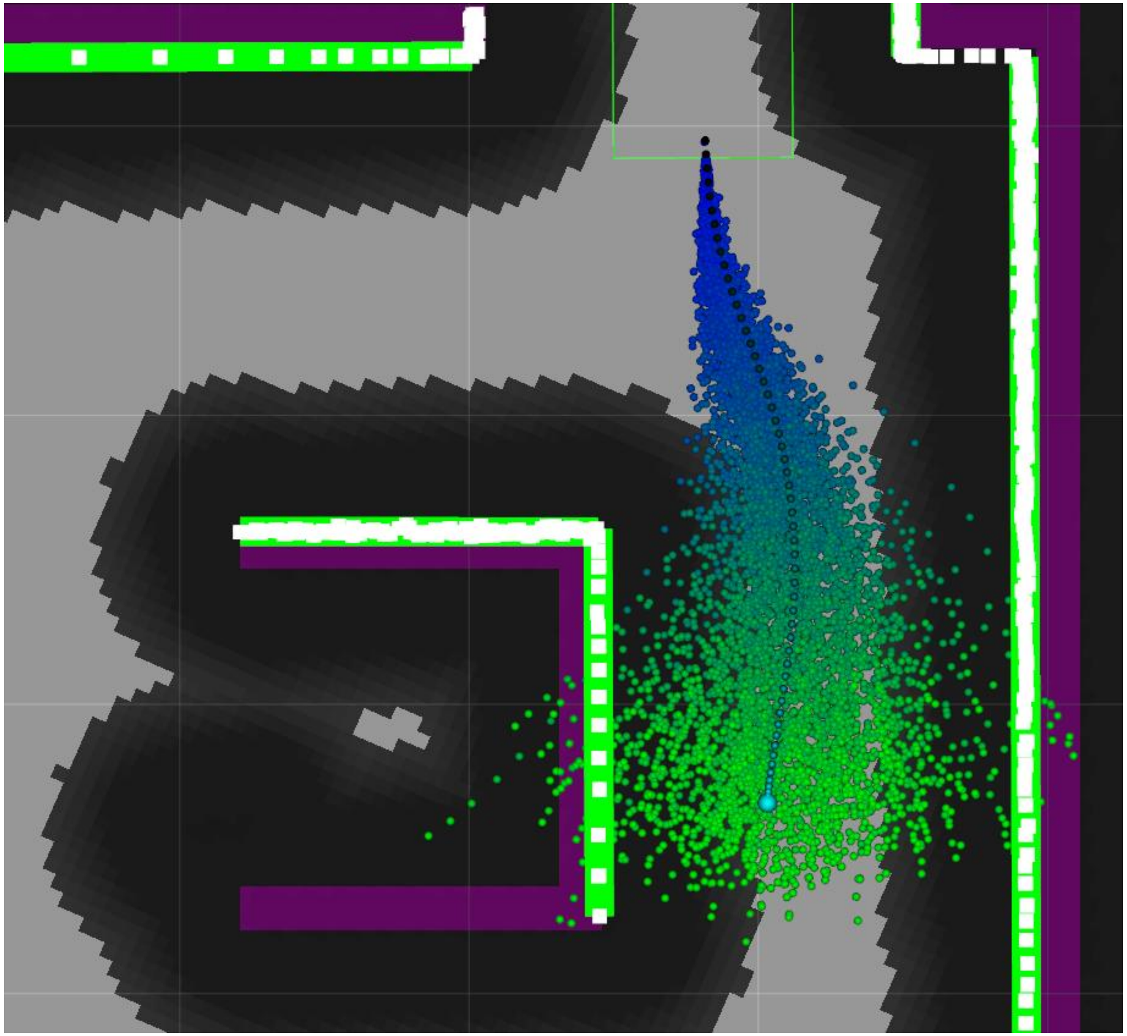
- Main concepts:
  - Admissible velocity:
    - Can be achieved in the current cycle given dynamic constraints
  - Simulation window:
    - How far to project robot velocity in the future
  - Scheduling window:
    - How often to recalculate
  - Exploration grid:
    - Granularity of velocity space to explore
  - Critics:
    - Chain of functions that score the velocity

Fox, et al., The Dynamic Window Approach to Collision Avoidance

Admissible Velocities



MPPI



MPPI

- Model-Predictive Path Integration
- Model-Predictive, but not MPC:
  - MPC optimizes the cost (goal distance, path deviation, ..)
  - Subject to constraints: system dynamics
  - Can provide stability and optimality guarantees
- MPPI:
  - Generate input sequence using Monte-Carlo methods
  - Score the sequence
  - Weight-combine
  - No optimization, just exploration
  - No guarantees, but fits well into navigation stack and works

MPPI

- Nominal input sequence: $u_0, u_1, u_2 \ldots u_n$
- Generate randomized sequences: $u_i^{*(k)} = u_i + \epsilon_{i,k} \, \forall k$
- Roll-out the states using the model: $x_i^{(k)} = f(x_{i-1}^{(k)} u_i^{(k)})$
- Score each state rollout using critics (w_k are weights)
- Weight-combine input sequences
- Alternative (Williams, et al. ICRA2016):
  - Randomize the state rollout instead
  - Score and weight-combine states
  - Find control sequence that corresponds to combined state

`https://ieeexplore.ieee.org/document/7487277`