

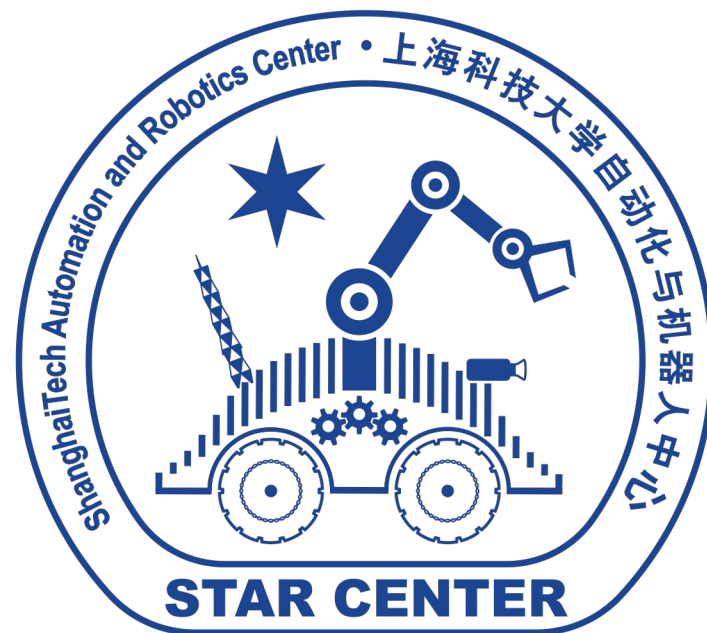


上海科技大学
ShanghaiTech University

CS289: Mobile Manipulation Fall 2026

Sören Schwertfeger

ShanghaiTech University



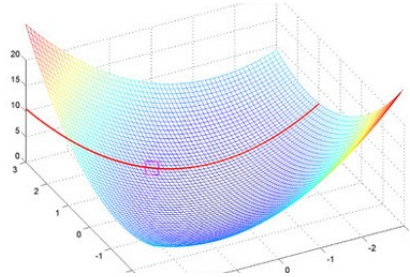
Outline

- Arm planning
- OMPL
- Mobile Robotics

Path Planning: Overview of Algorithms

1. Optimal Control

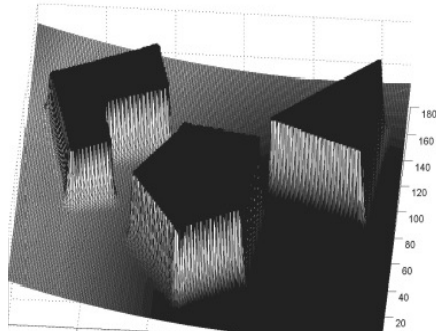
- Solves truly optimal solution
- Becomes intractable for even moderately complex as well as nonconvex problems



Source:
<http://mitocw.udsm.ac.tz>

2. Potential Field

- Imposes a mathematical function over the state/configuration space
- Many physical metaphors exist
- Often employed due to its simplicity and similarity to optimal control solutions



3. Deterministic Graph Search

- Identify a set edges between nodes within the free space
- Breath/ Depth First Search
- Dijkstra
- A*, D*



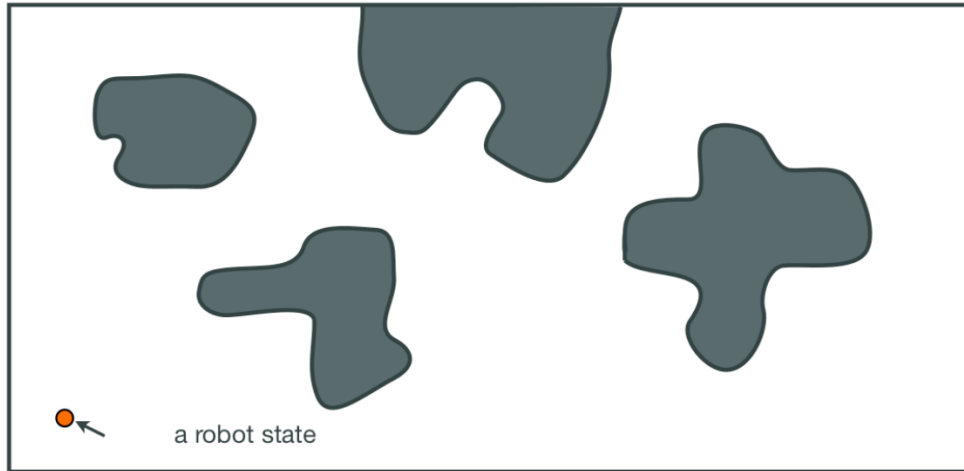
4. Sampling-based Motion Planning

- Randomized Graph Search
- Rapidly exploring Random Tree RRT

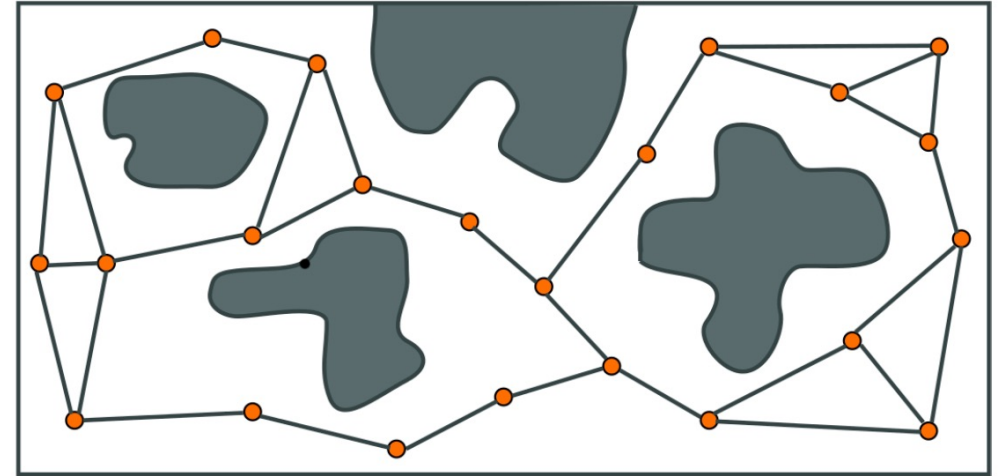
Sampling-based Motion Planning

- No need to reason over the entire continuous state space -> no explicit state space representation!
- Well suited for high-dimension search space -> robot arm planning (often 6DoF or more)
- Sample state space of robot (configuration space)
 - for collisions
 - and constrains
 - pose constraints on state (e.g. point gripper up)
 - differential constraints on edges between state:
 - edge represents the motion between two states
 - edges are only added if they adhere to the constraints (e.g. maximum speed; maximum acceleration)
- Probabilistic Roadmap
 - One of first sampling-based motion planner

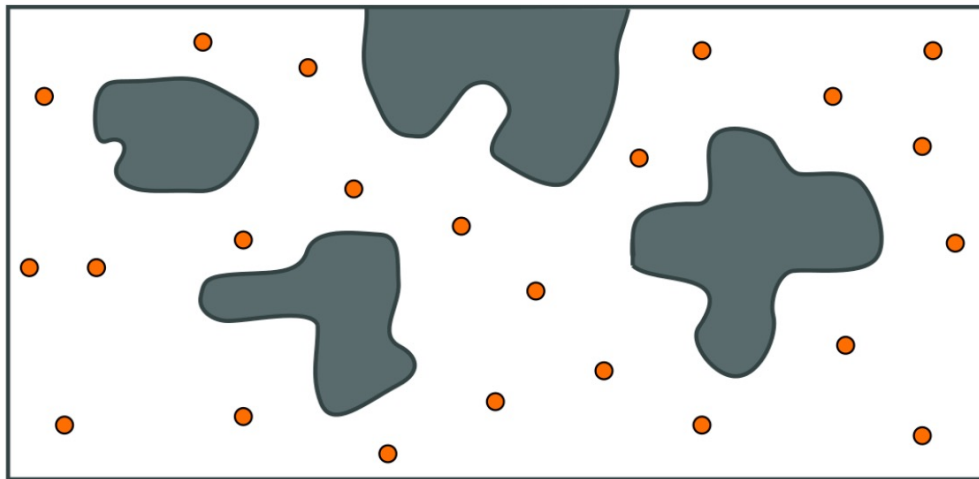
Probabilistic Roadmap



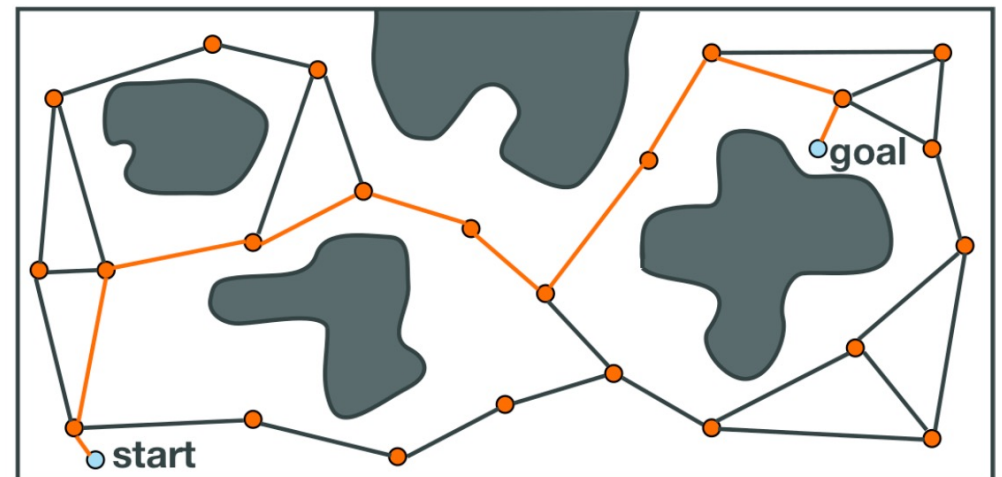
1) Single State (e.g. robot; configuration)



3) Connect close samples with straight path



2) Uniform random samples of free state space



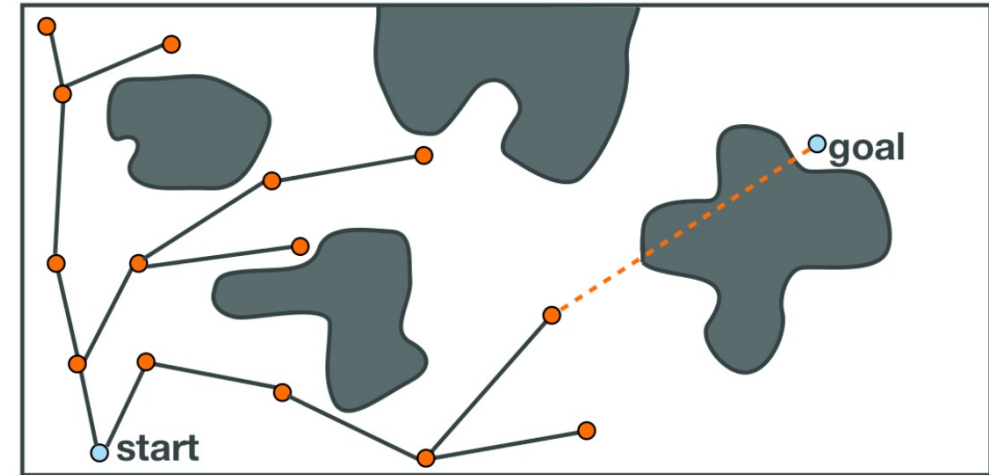
4) Use graph search (e.g. A*) to plan a path

Probabilistic Roadmap vs Tree-based Planner

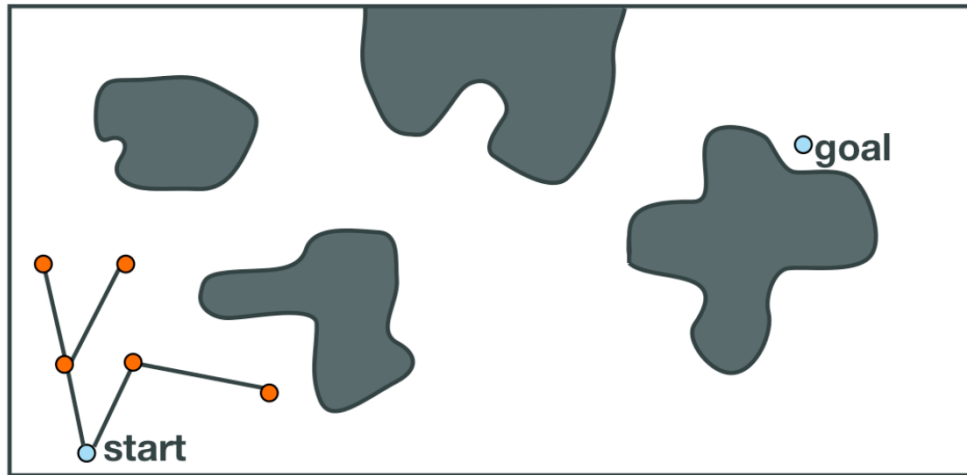
- Probabilistic Roadmap
 - Goal is used only at last step -> pre-compute graph
 - Need samples distributed over entire space (don't know where goal is)
 - Too big for high dimension search spaces
 - But: fast planning for small dimension search spaces (e.g. 2D or 3D robot motion planning)
- Tree-based Planning
 - Directed edges -> can encode control commands
 - E.g.: Move up needs more power than move down -> motion is directed
 - Easier to represent differential constraints (e.g. max acceleration of joints)
 - Expand tree on the fly
 - Sometimes with bias towards goal

Tree-based Planner

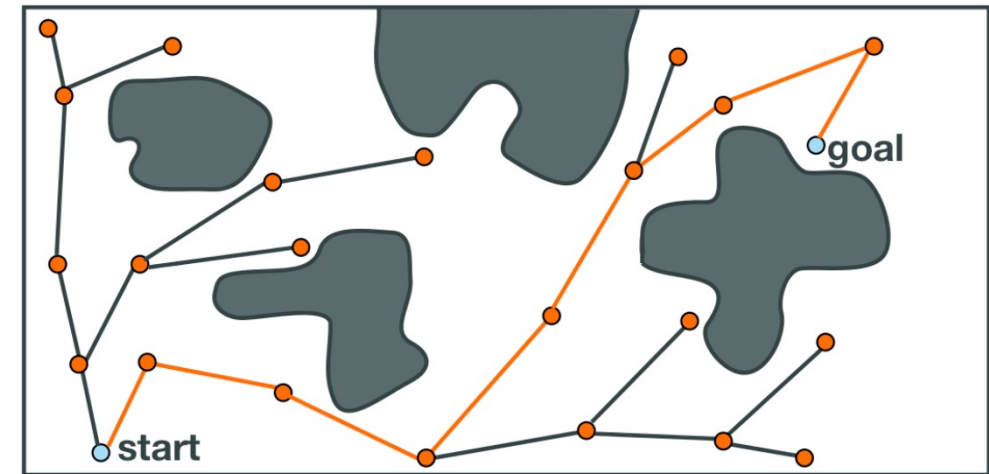
- Tree:
 - no cycles
 - Directed edges
- Grow from start state
- Expansion heuristic to select next state in tree



2) Goal is close to a node, but cannot be connected because collision check of edge fails!



1) First few samples expanded randomly using expansion heuristic



3) Goal is close to a node and can be connected

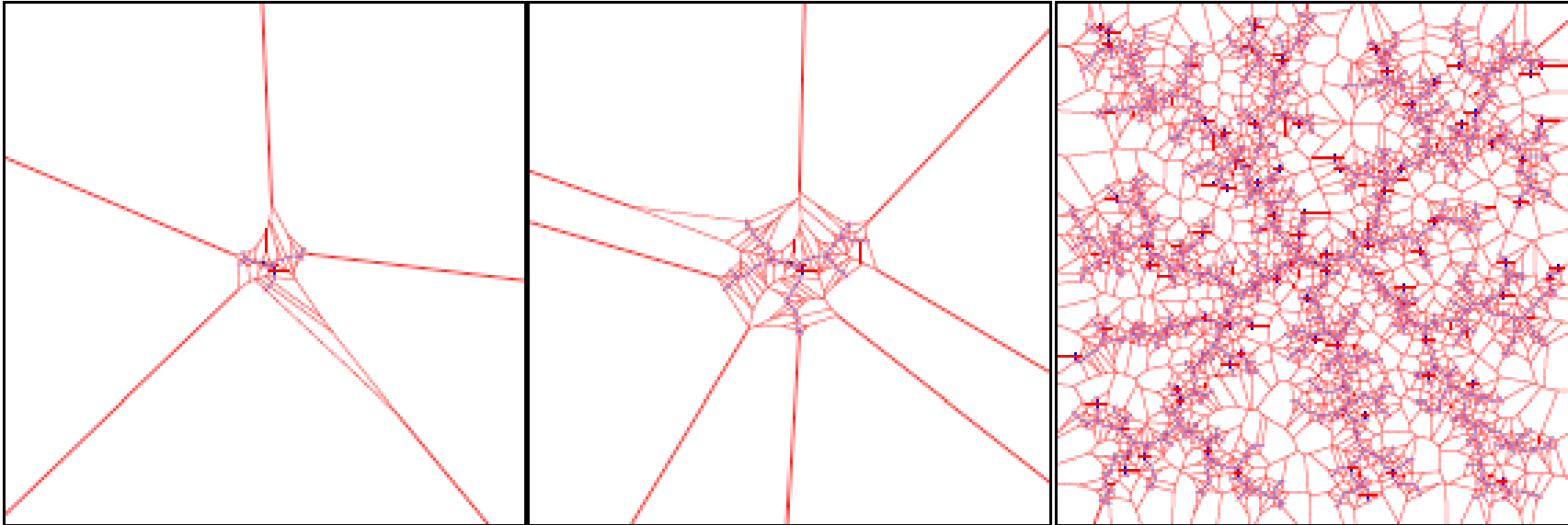
Rapidly Exploring Random Tree: RRT

- Most famous/ widely used planner concept for Robot Manipulations
- Monte-Carlo based method with
- Bias towards “big empty space” (largest Voronoi region)

Kuffner, J. J., & LaValle, S. M. (2000, April). RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*. IEEE.

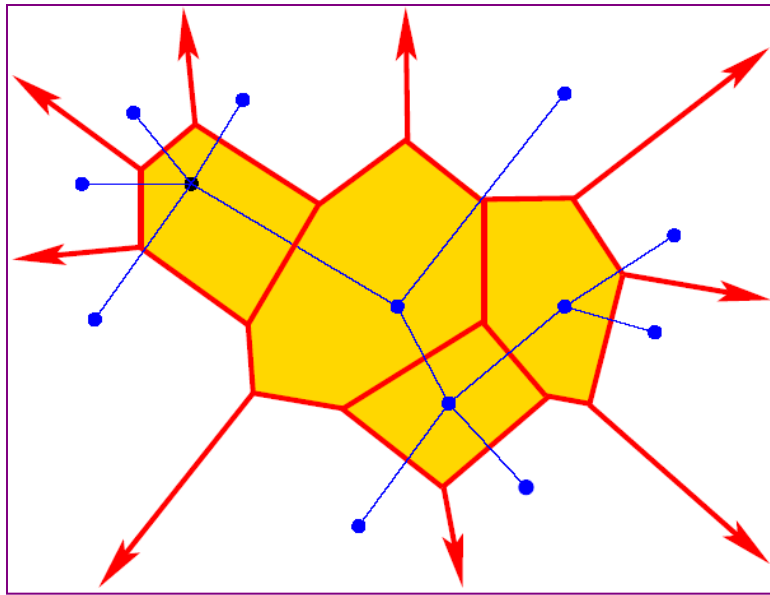
Animation from iteration 0 to 10,000

Why “Rapidly Exploring”?

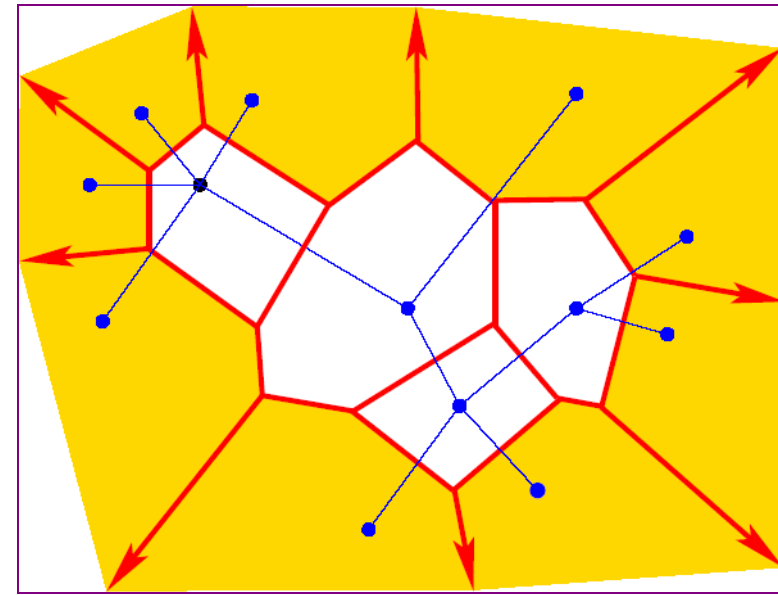


- What is the probability that a vertex will be extended?
 - Proportional to the area of its Voronoi region
- If just choose a vertex at random and extend, then it would act like random walk instead
 - Biased towards start vertex

Refinement vs. Expansion



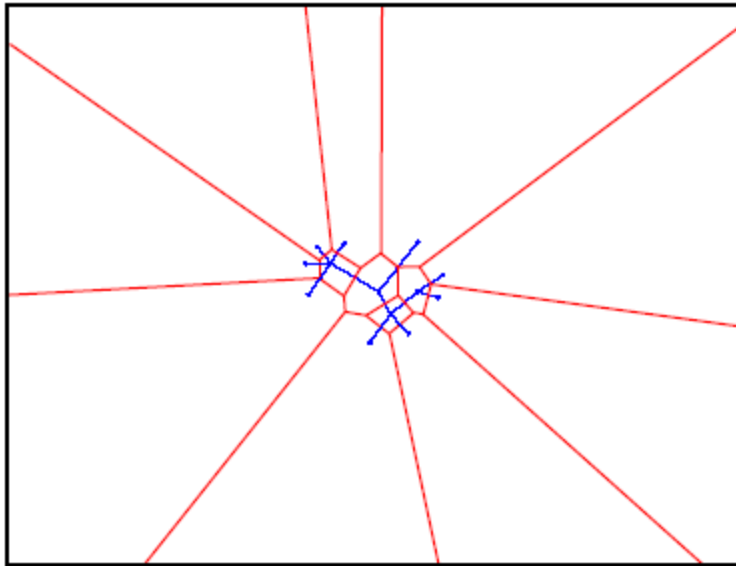
refinement



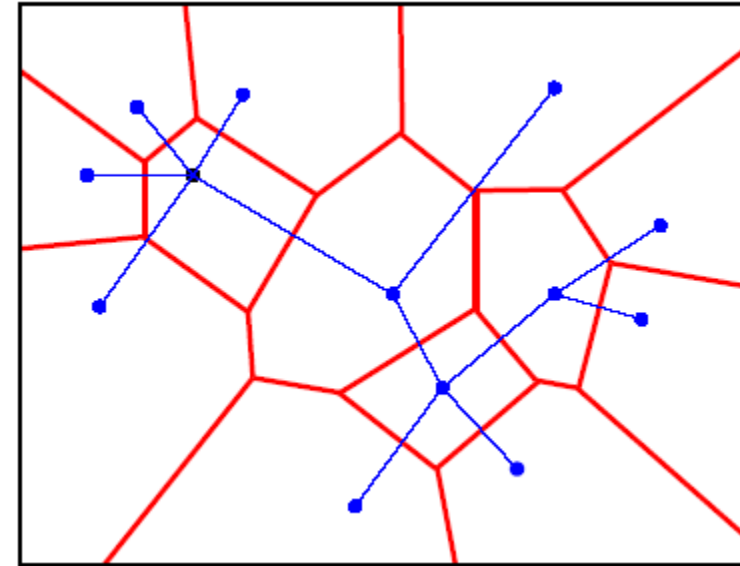
expansion

Where will the random sample fall? How to control the behavior of RRT?

Determining the Boundary



Expansion dominates



Balanced refinement and expansion

The tradeoff depends on the size of the bounding box

Problem

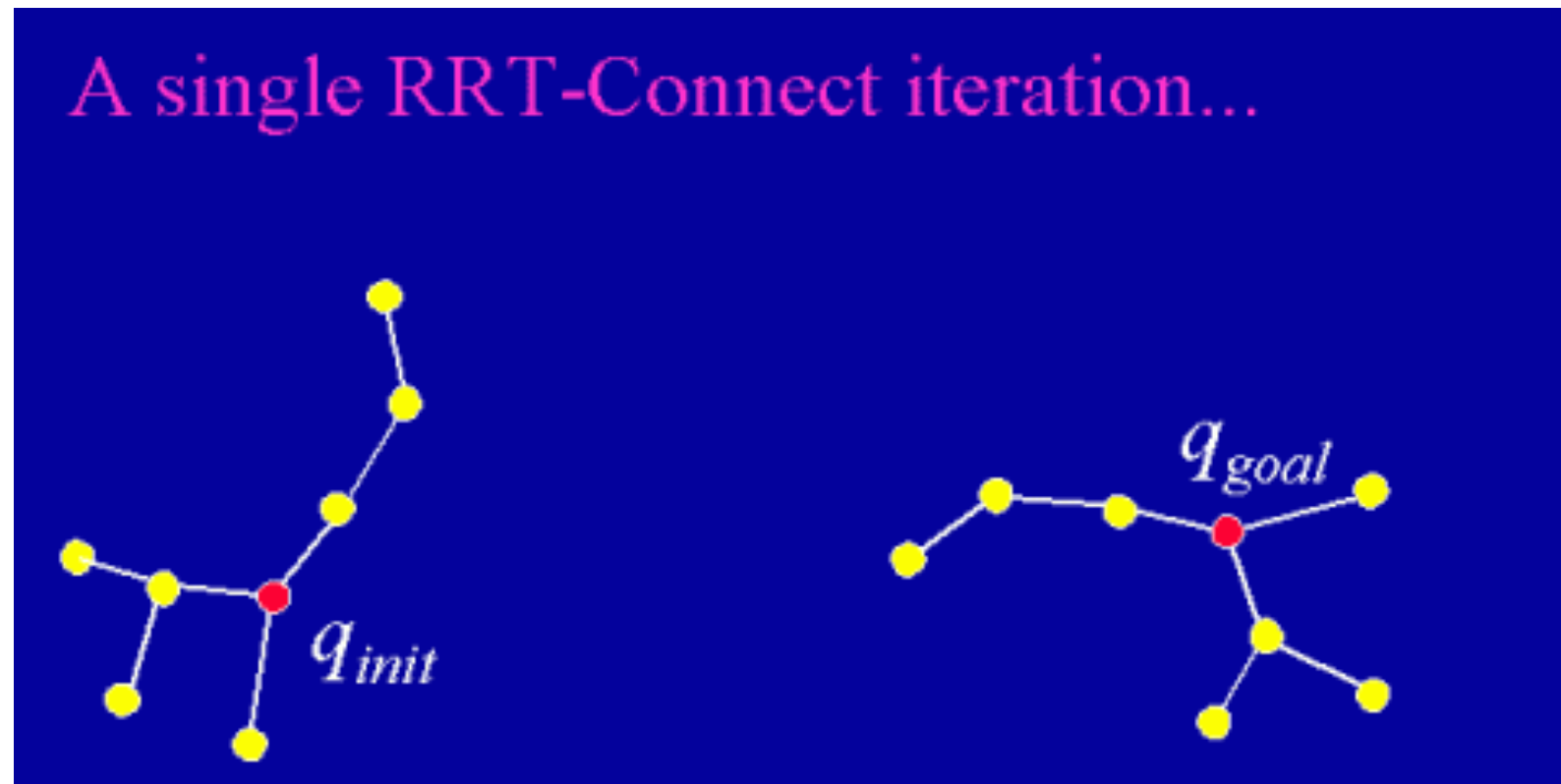
- Non-biased RRT explores in all directions
 - Not aimed at the goal
- Use a small percentage of targets to be the goal or its neighborhood

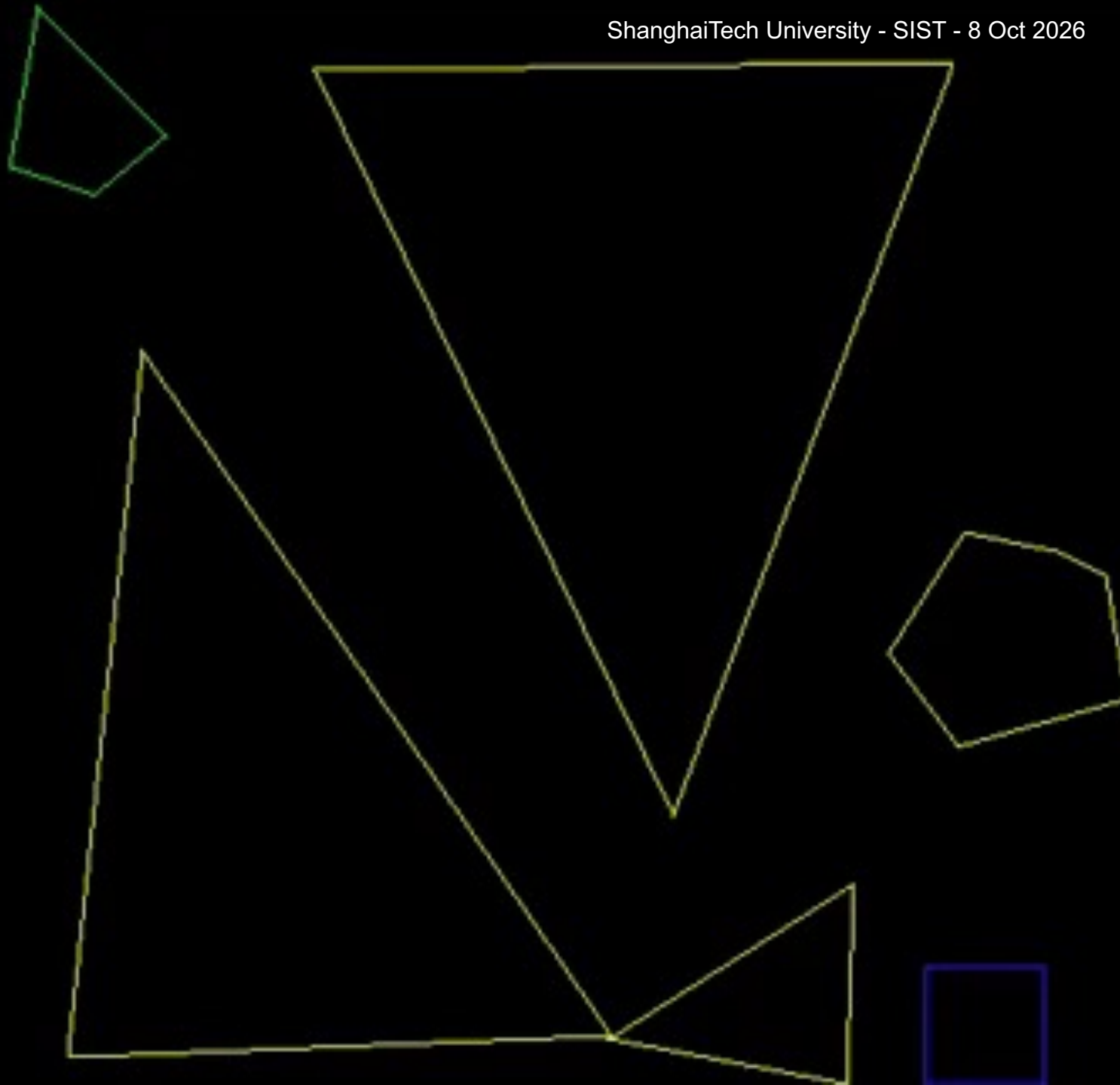
How Far to Extend?

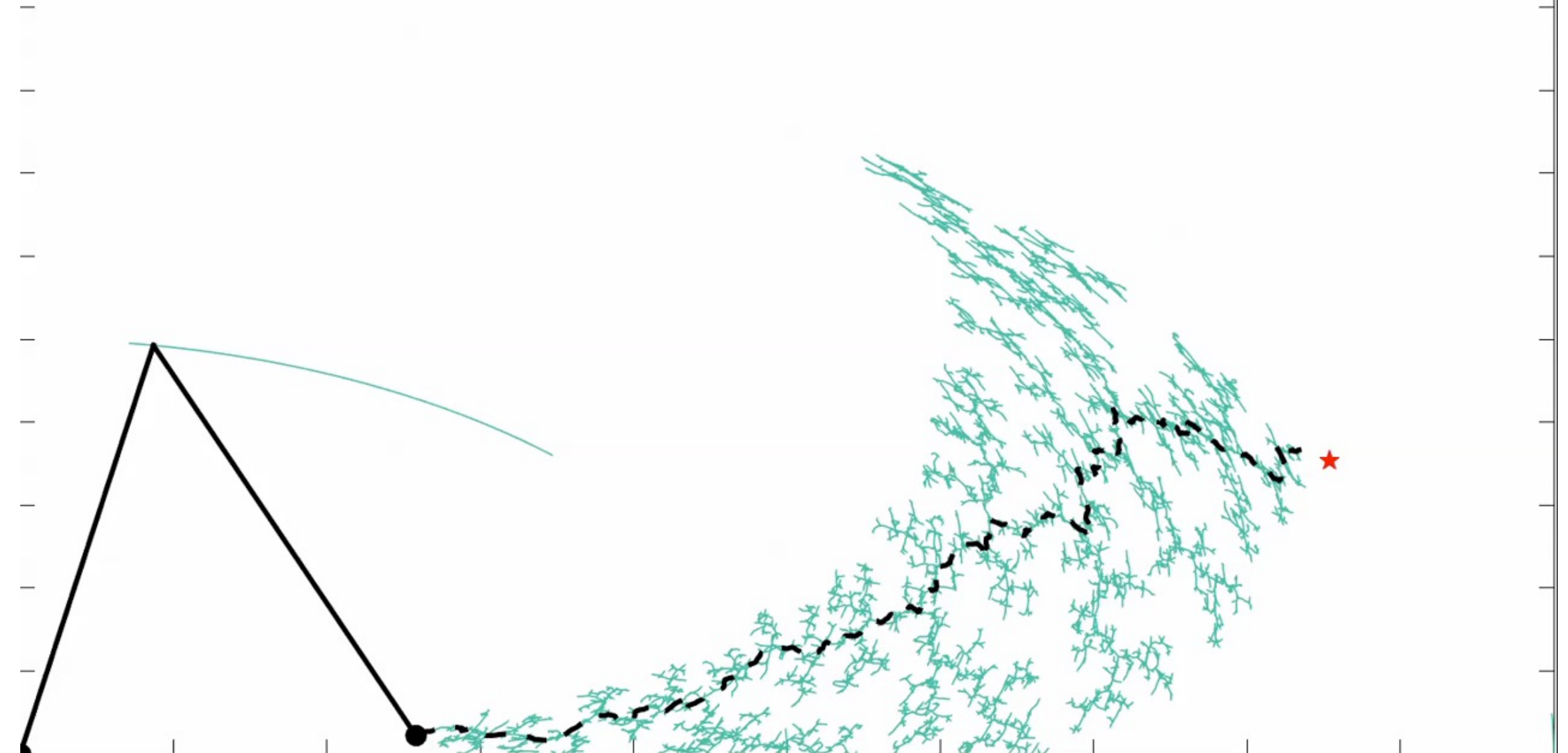
- Use distance measures from collision detection
 - Big steps when far away

RRT connect

- Grow two trees from start and goal – connect when close

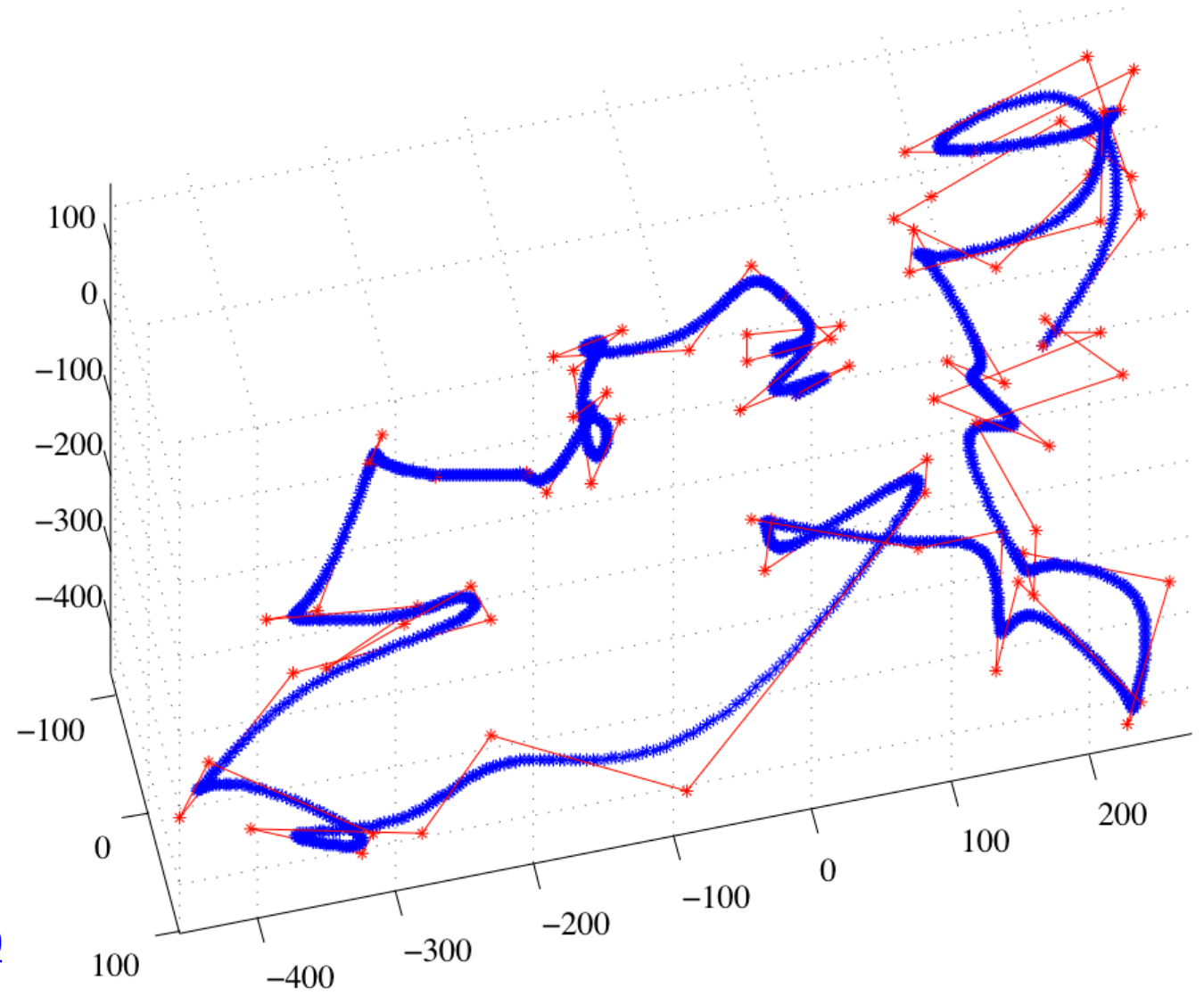






Path smoothing

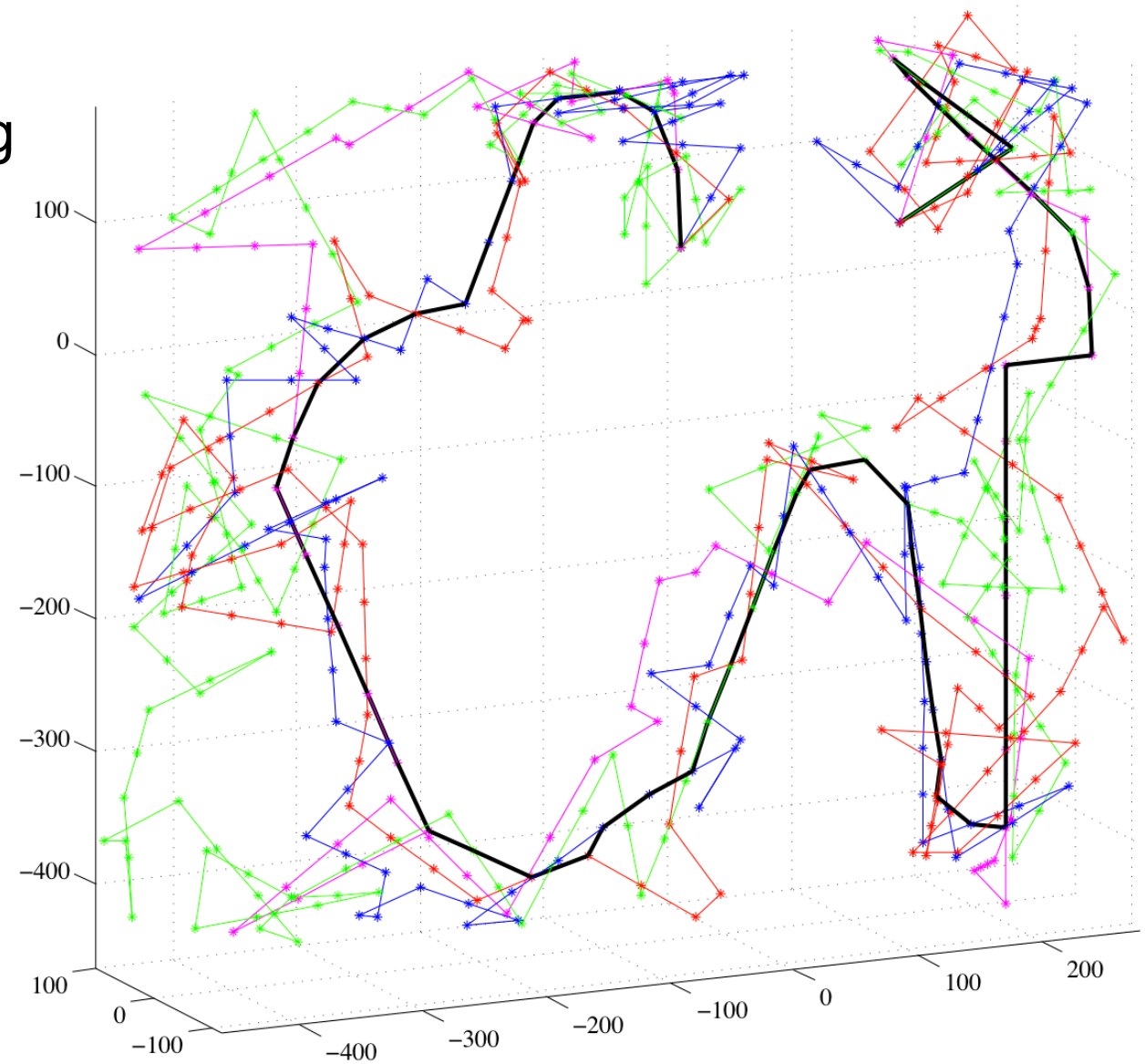
- Path in tree not smooth ->
- Jerky motion and high forces ->
- Path smoothing needed
- E.g. use B-splines



Images from this and next slide from:
https://ompl.kavrakilab.org/gallery.html#gallery_comp450

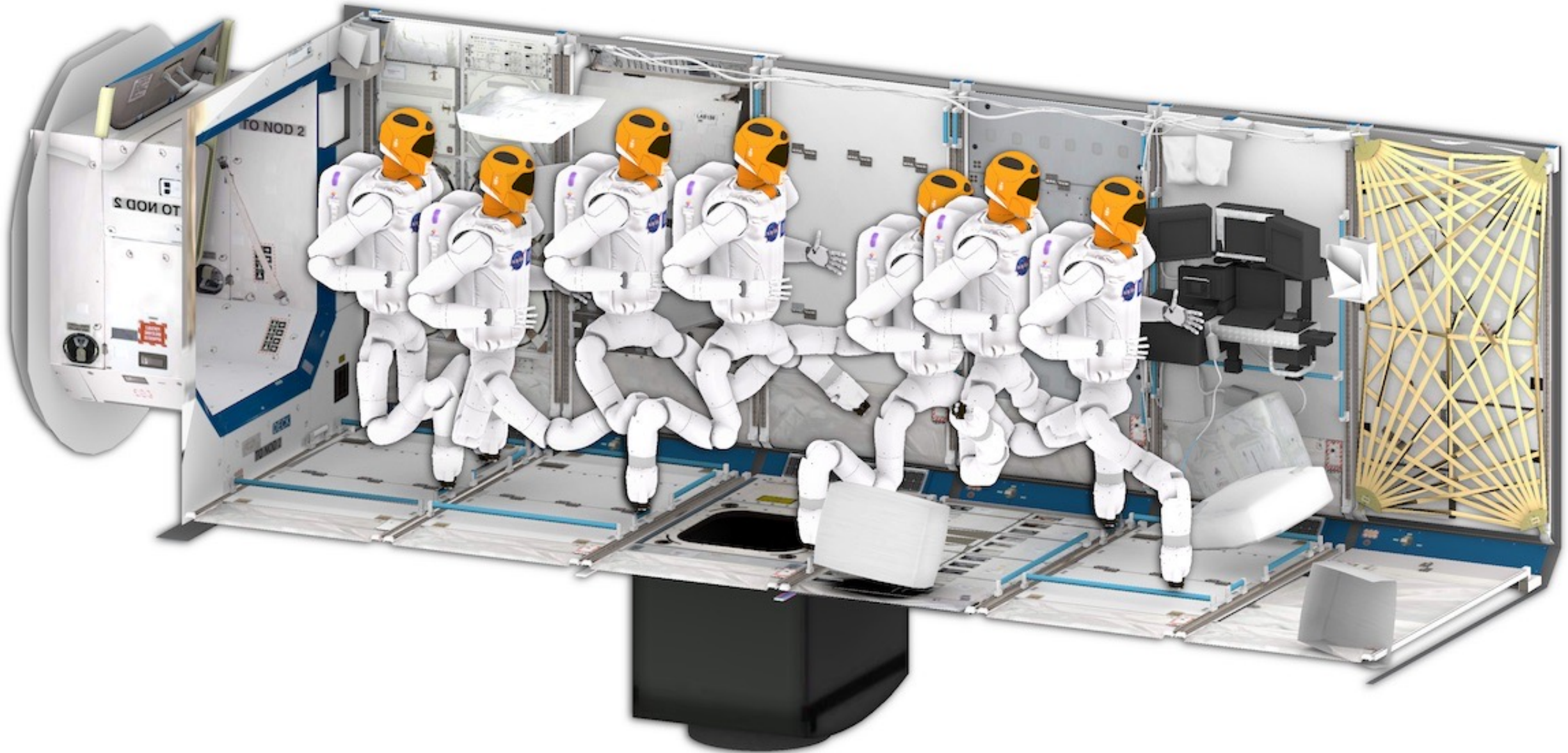
Path Shortening

- Path generated by RRT often too long
- Optimizing/ shortening path desirable
- Example: path shortening using hybridization

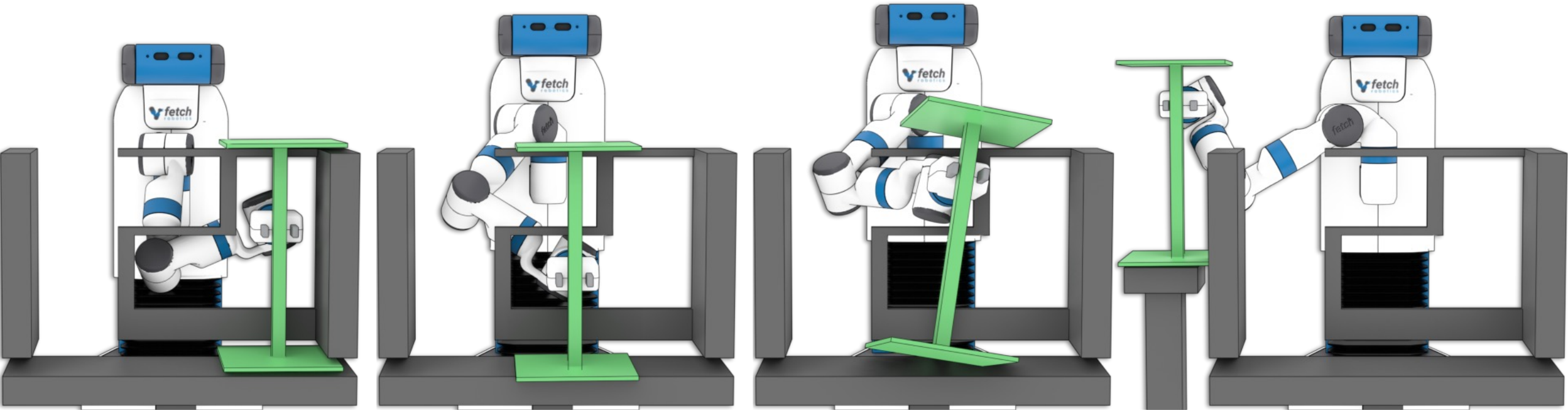


OMPL: Open Motion Planning Library

- <https://ompl.kavrakilab.org/>
- Used in many systems
 - Movit -> ROS; OpenRAVE; CoppeliaSim; MORSE; ...
- Limited to motion planning algorithms ->
 - Collision detection; visualization; environment specification external, e.g. MoveIt!
- Written in C++ with python bindings
- Includes framework to Benchmark Planners -> select best planner for your problem/ robot! <https://plannerarena.org/>

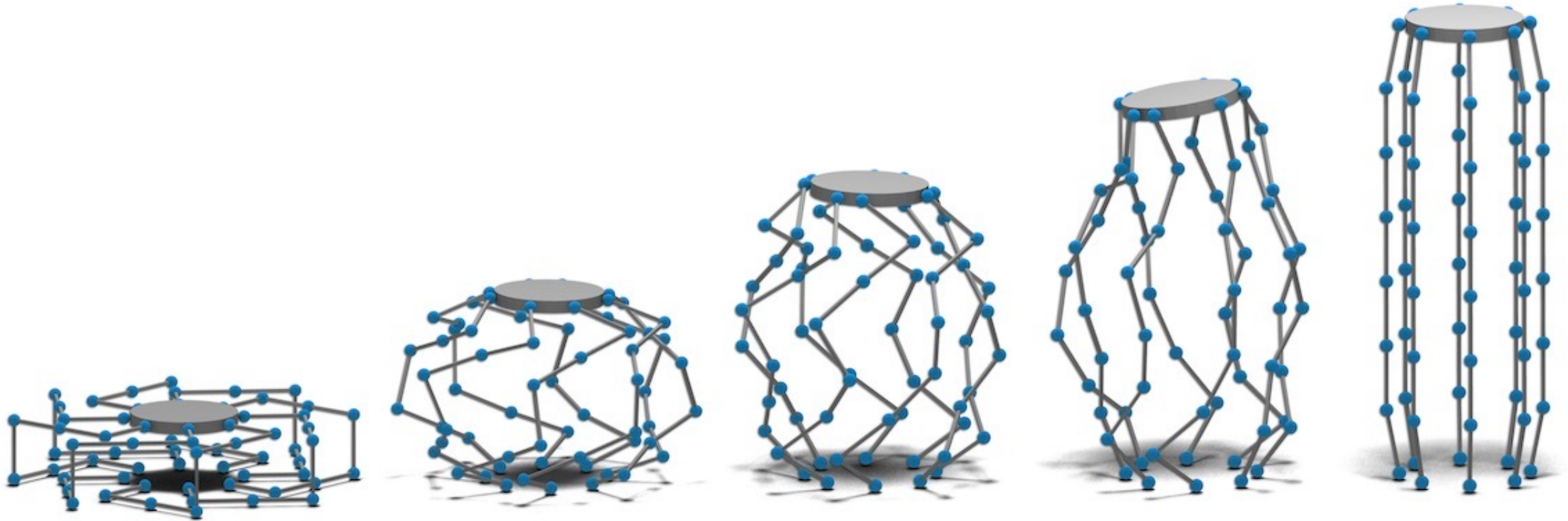


OMPL is the default planning library in MoveIt and has been used for many robots. Here, OMPL is used to plan footsteps for NASA's Robonaut2 aboard the International Space Station.



Multi-modal Motion Planning

OMPL has been used in multi-modal planners to achieve complex robot behavior, such as removing this barbell object from a puzzle box.



Constrained Motion Planning

OMPL supports planning for kinematically constrained robots. This parallel manipulator has over 150 degrees of freedom, but feasible motions can still be computed in seconds.

OMPL Foundations (1)

- StateSampler
 - Uniform and Gaussian sampling of state space configurations
- NearestNeighbors
 - Perform nearest neighbor search along samples
 - E.g. Geometric Near-neighbor Access Trees (GNAT) (similar to kd-tree), linear searches
- StateValidityChecker
 - Check a single state for
 - Collision
 - Constraints

OMPL Foundations (2)

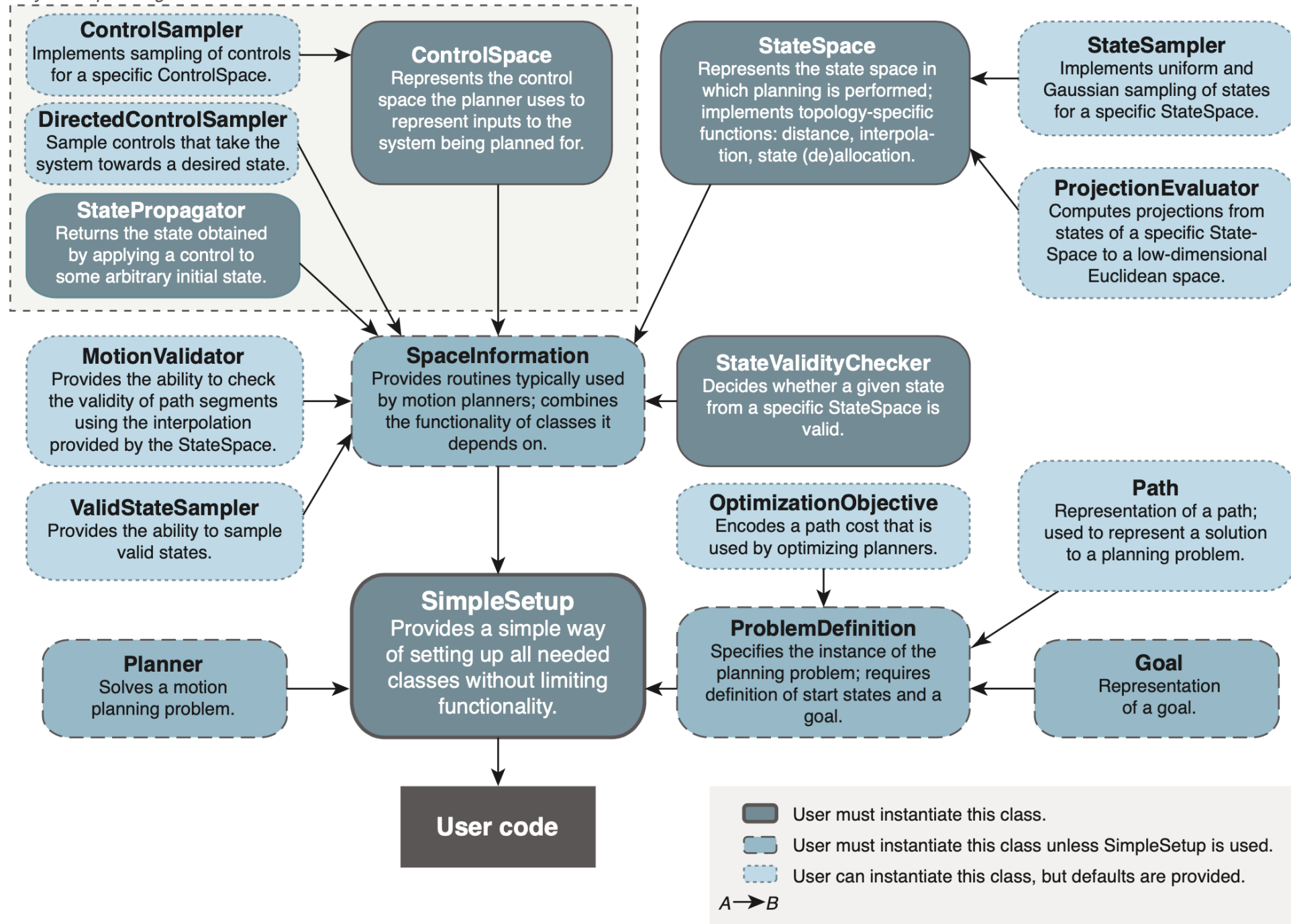
- MotionValidator
 - Check if motion between two states is valid
 - Collisions
 - Constraints
 - E.g.: DiscreteMotionValidator: interpolated movement between 2 states
 - Check finite number of states (using StateValidityChecker) between the 2 states
- OptimizationObjective
 - Some planners support optimization, e.g.: Minimize path length
- ProblemDefinition
 - Planning query with start, goal, optimizationObjectives
 - Goal can be single configuration or region surrounding a state

OMPL Foundations (3)

- StatePropagation
 - Given: state, control, duration
 - Calculate: new state
 - Used for control-based planning
- Steer
 - Also member of StatePropagator Class
 - Given: start and end state (close to each other!)
 - Calculate: needed control; duration; success
 - Used to save control commands in edge/ path/ trajectory

OMPL

only when planning with differential constraints



OMPL Available Planners: Geometric Planners (1)

- Only geometric and kinematic constraints
- Multi-query planners
 - First build roadmap, afterwards plan quickly for many queries
 - Probabilistic Roadmap Methods (PRM): LazyPRM; PRM*; LazyPRM*
 - SPArse Roadmap Spanner algorithm (SPARS); SPARS2
- Single-query planners
 - RRTs: RRT Connect; RRT*; Lower Bound Tree RRT; Parallel RRT; Lazy RRT; ...
 - Search Tree with Resolution Independent Density Estimation (STRIDE)
 - Path-Directed Subdivision Trees (PDST)
 - Fast Marching Tree algorithm (FMT*); Bidirectional FMT*
 - ...

OMPL Available Planners: Geometric Planners (2)

- Multi-level planners
 - Multiple levels of abstraction
 - Rapidly-exploring Random Quotient space Trees (QRRT); QRRT*
 - Quotient-Space Roadmap Planner (QMP): generalized RPM; QMP*
- Optimizing planners:
 - Have a star * after the name
 - Provide optimality guarantee, even though sampling based
 - Optimization objectives:
 - shortest path;
 - Minimum path clearance;
 - Minimize mechanical work;
 - General state cost integral; (user-based optimization)

OMPL Available Planners: Control-based Planners

- Differential constraints $\rightarrow$ need control-based planner!
- State propagation instead of interpolation for motion/ edge
- RRT
- Sparse Stable RRT
- Expansive Space Trees (EST)
- Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE)
- Path-Directed Subdivision Trees (PDST)

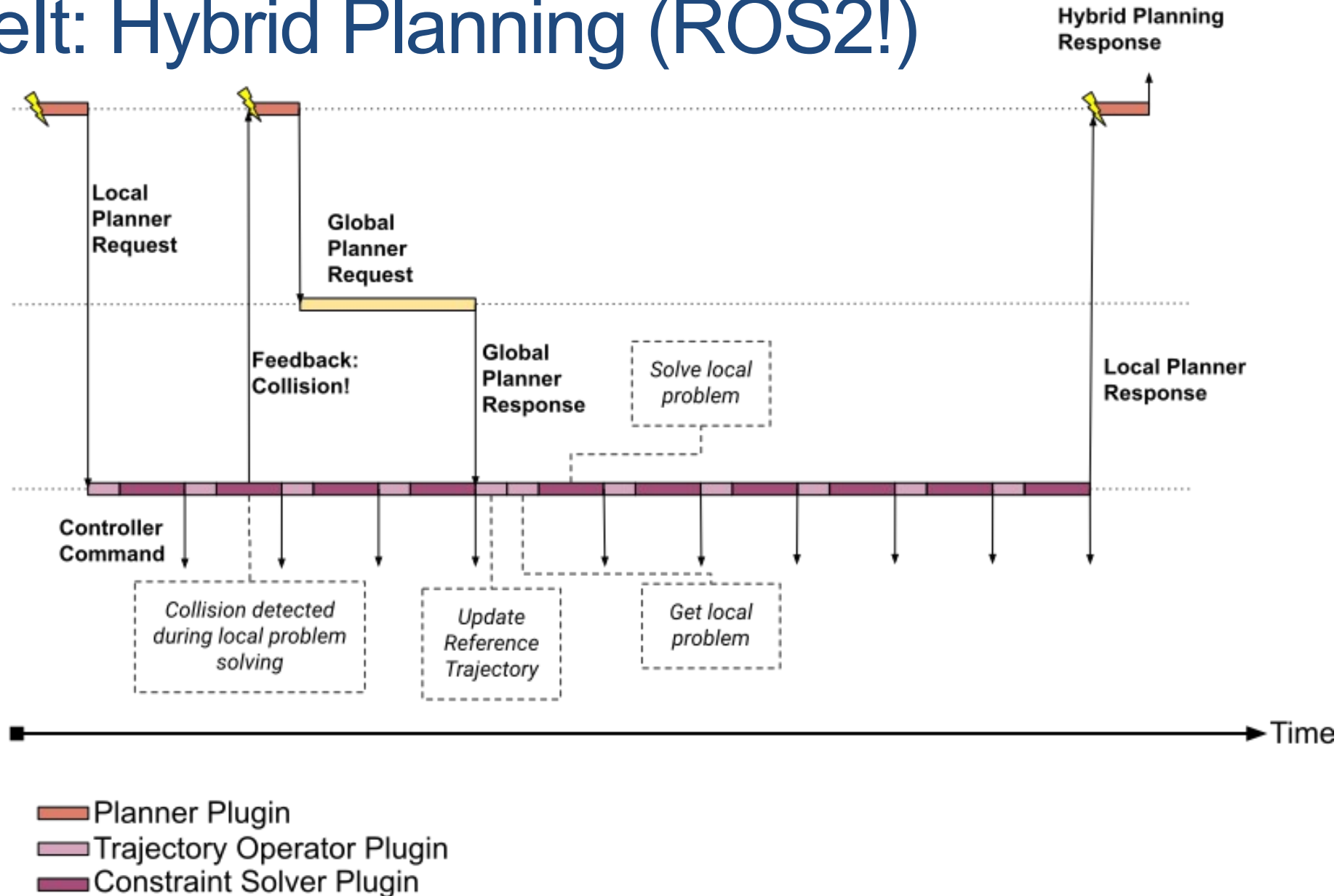
OMPL Available Planners: Multilevel-based Planners

- Very high-dimensional state space ->
 - Use Multi-level abstractions to simplify state space ->
 - Find solutions faster
-
- Rapidly-exploring Random Quotient space Trees (QRRT)
 - QRRT*
 - Quotient-Space Roadmap Planner (QMP)
 - QMP*
 - Sparse Quotient space roadmap planner (SPQR)*

Movelt ROS 2 Planners

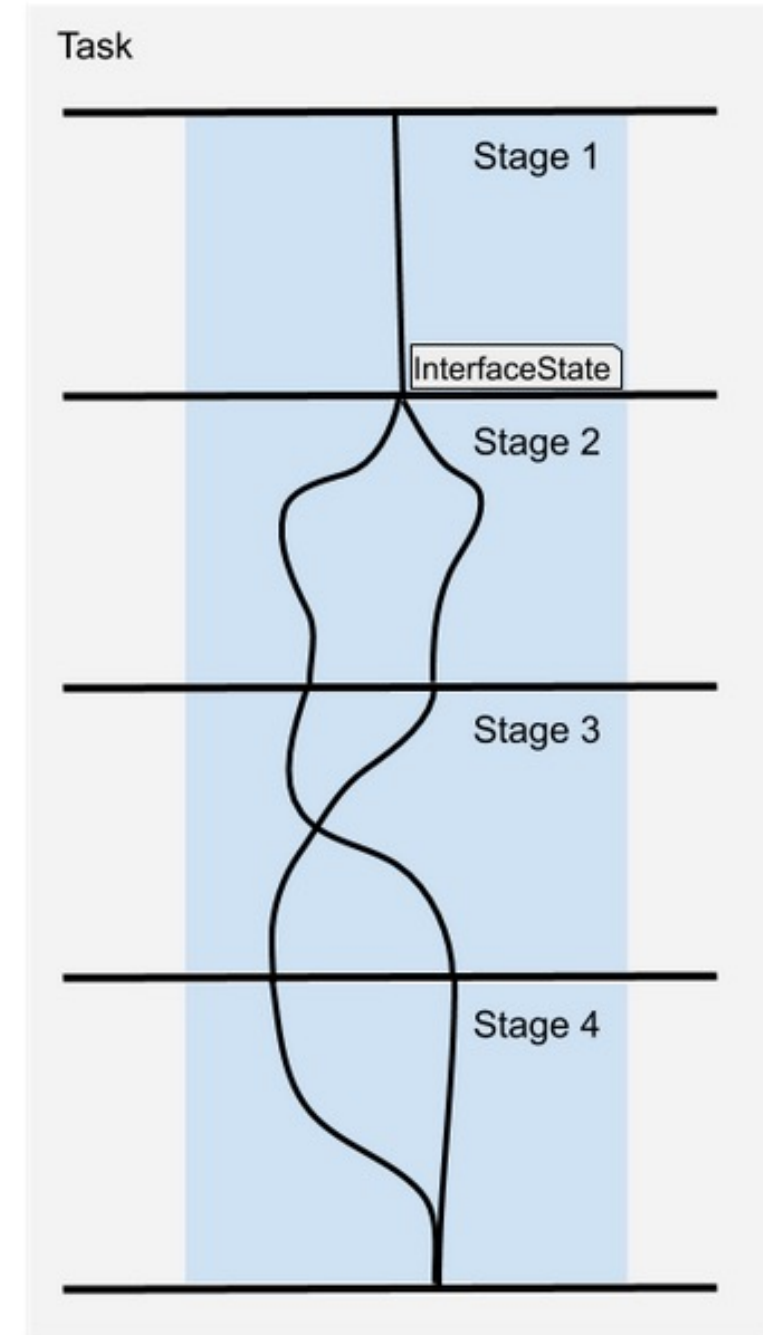
- OMPL
- Pilz Industrial Motion Planner
 - Deterministic generator for circular and linear motions
 - Supports blending multiple motion segments together using a Movelt capability.
- Stochastic Trajectory Optimization for Motion Planning (STOMP)
 - Optimization-based planner based on “Policy Improvement with Path Integrals” (partially supported)
- Search-Based Planning Library (SBPL)
 - Search based planning in discrete space (work in progress)
- Covariant Hamiltonian Optimization for Motion Planning (CHOMP)
 - gradient-based trajectory optimization
 - covariant gradient and functional gradient approaches
 - motion planning algorithm based entirely on trajectory optimization
 - Start with infeasible, naïve trajectory -> use optimization to pull out of collision (work in progress)

Movel: Hybrid Planning (ROS2!)

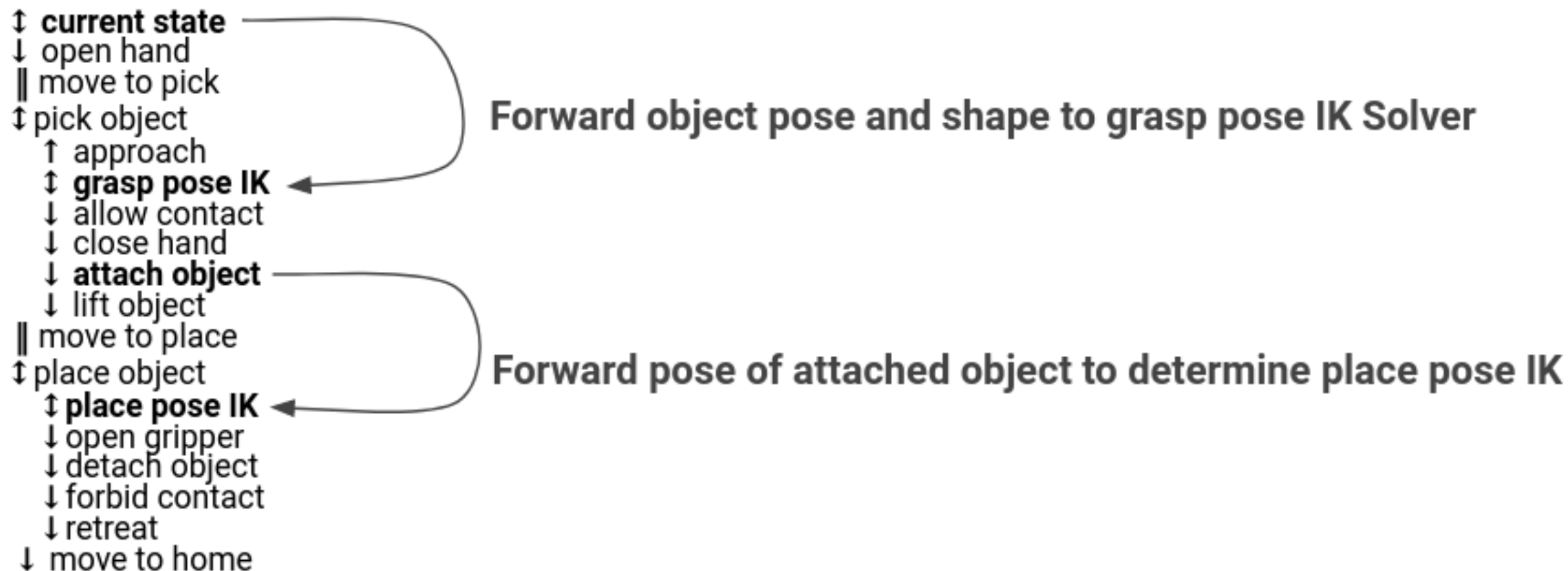


MoveIt Task Constructor MTC in ROS2

- Break down complex planning tasks to multiple interdependent subtasks
- MTC uses MoveIt to solve the subtasks
- Stages:
 - Generators: no input; compute results and send in both directions; e.g. GeneratePose (e.g. current or goal)
 - Propagators: get solution, do something, transmit result; e.g. Move Relative
 - Connectors: get inputs but do not provide output: solve for feasible trajectory between the start and goal states
- Wrappers: encapsulate another stage to modify or filter the results
- Parallel containers: combine a set of stages to allow planning alternate solutions



MTC: Pick and Place



MTC: Pick and Place

File Panels Help

Interact Move Camera Select Key Tool

Displays

- Global Options
- Global Status: Ok
- Grid ☒
- MarkerArray ☒
 - Status: Ok
 - Marker Topic /rviz_visua...
 - Queue Size 100
 - Namespaces
- Trajectory ☒
- PlanningScene ☒
- Motion Planning Tasks ☒

Add Duplicate Remove Rename

RvizVisualToolsGui

Next Continue Break Stop

Reset

Motion Planning Tasks

Task Tree

Name		
▼ Motion Planning Tasks		
▼ pick_place_task	10	0
▼ applicability test	1	0
↑ current state	1	0
↓ open hand	1	0
↓ move to pick	13	0
▼ pick object	14	0
↑ approach object	14	2
▼ grasp pose IK	101	4
↑ generate grasp pose	25	0
↓ allow collision (hand,object)	17	0
↓ close hand	17	0
↓ attach object	17	0
↓ allow collision (object,support)	17	0
↓ lift object	17	0
↓ forbid collision (object,surface)	17	0
↓ move to place	10	0
▼ place object	11	0
↑ lower object	15	1
▼ place pose IK	22	6
↑ generate place pose	340	0
↓ open hand	17	0
↓ forbid collision (hand,object)	17	0
↓ detach object	17	0
↓ retreat after place	11	6

Properties

31 fps

File Panels Help

Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point + -

Motion Planning Tasks



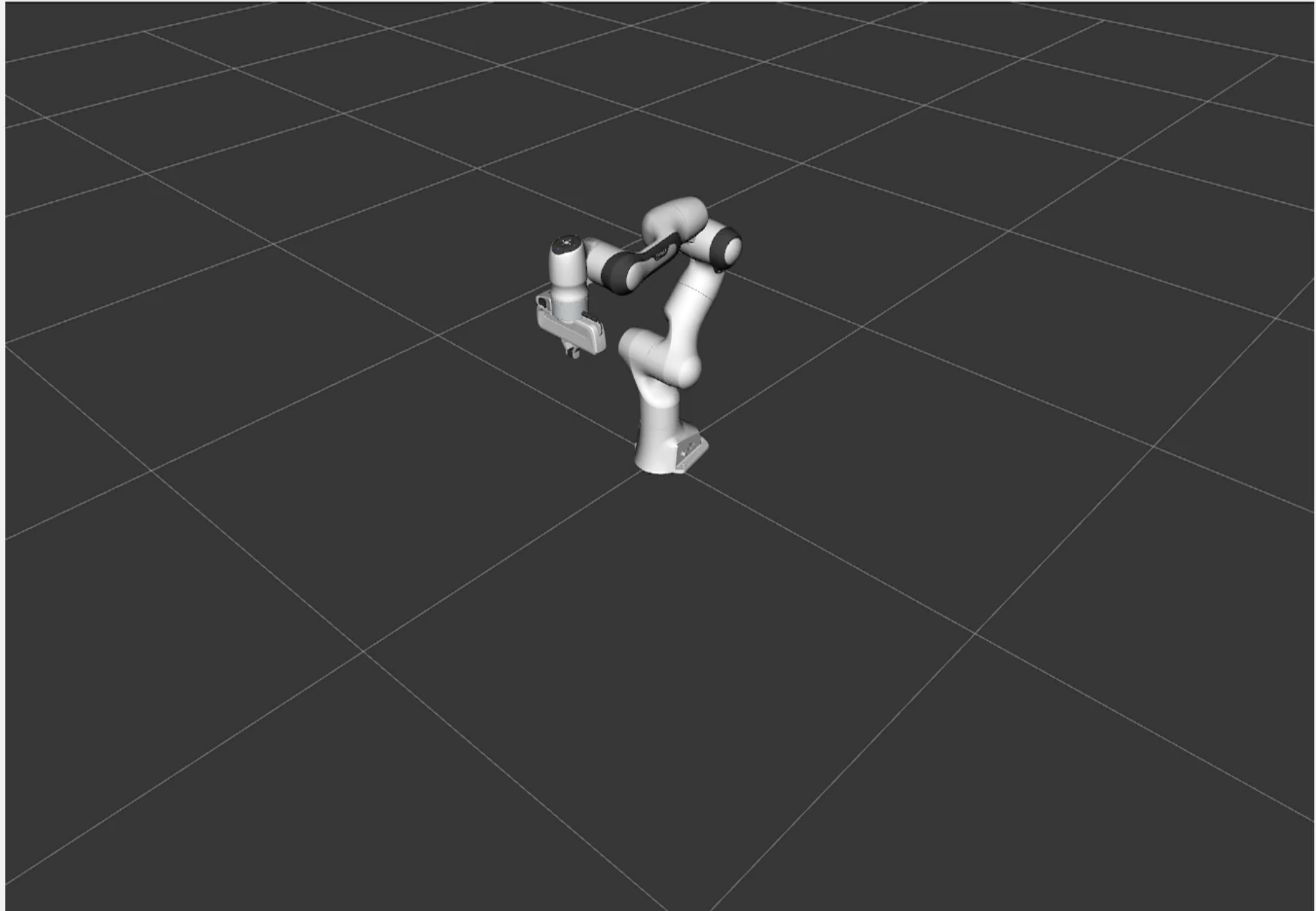
Task Tree

name ✓ ✗ time

Motion Planning Tasks

Exec

Properties



Learning-based Motion Planning

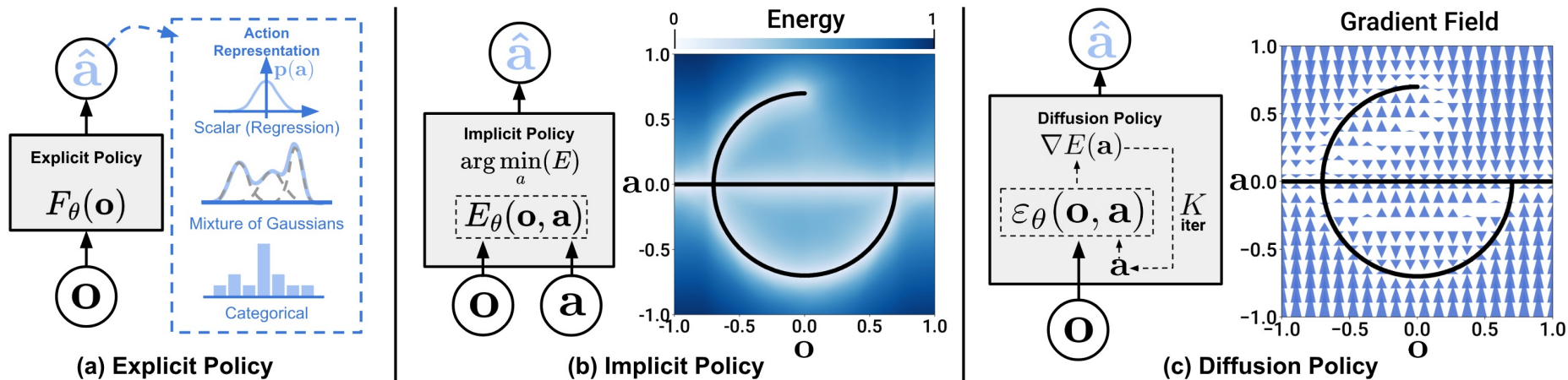
- Not (by default) available in MoveIt (yet)
- Supervised methods (30+ cited)
 - End-to-end: sensor data to motion ...
 - Module replacement: integrate with/ parallel to classical methods
 - E.g. improve RRT; Predict trajectory; improve collision checking
- Reinforcement Learning based Motion Planning (30+ cited)
 - value-based, policy-based, and actor-critic (AC)
 - Also end-to-end and module-replacement

Wang, J., Zhang, T., Ma, N., Li, Z., Ma, H., Meng, F., & Meng, M. Q. H. (2021). A survey of learning-based robot motion planning. *IET Cyber-Systems and Robotics*, 3(4), 302-314.

<https://ietresearch.onlinelibrary.wiley.com/doi/pdfdirect/10.1049/csy2.12020>

Learning-based Approaches

- Use Neural Networks to control/ plan arm motion E.g.:
- Visuomotor Policy Learning via Action Diffusion
 - generate robot behavior by representing a robot's visuomotor policy as a conditional denoising diffusion process <https://diffusion-policy.cs.columbia.edu/>
 - Transformer-based or CNN-based
 - predict robot arm's end-effector pose trajectory

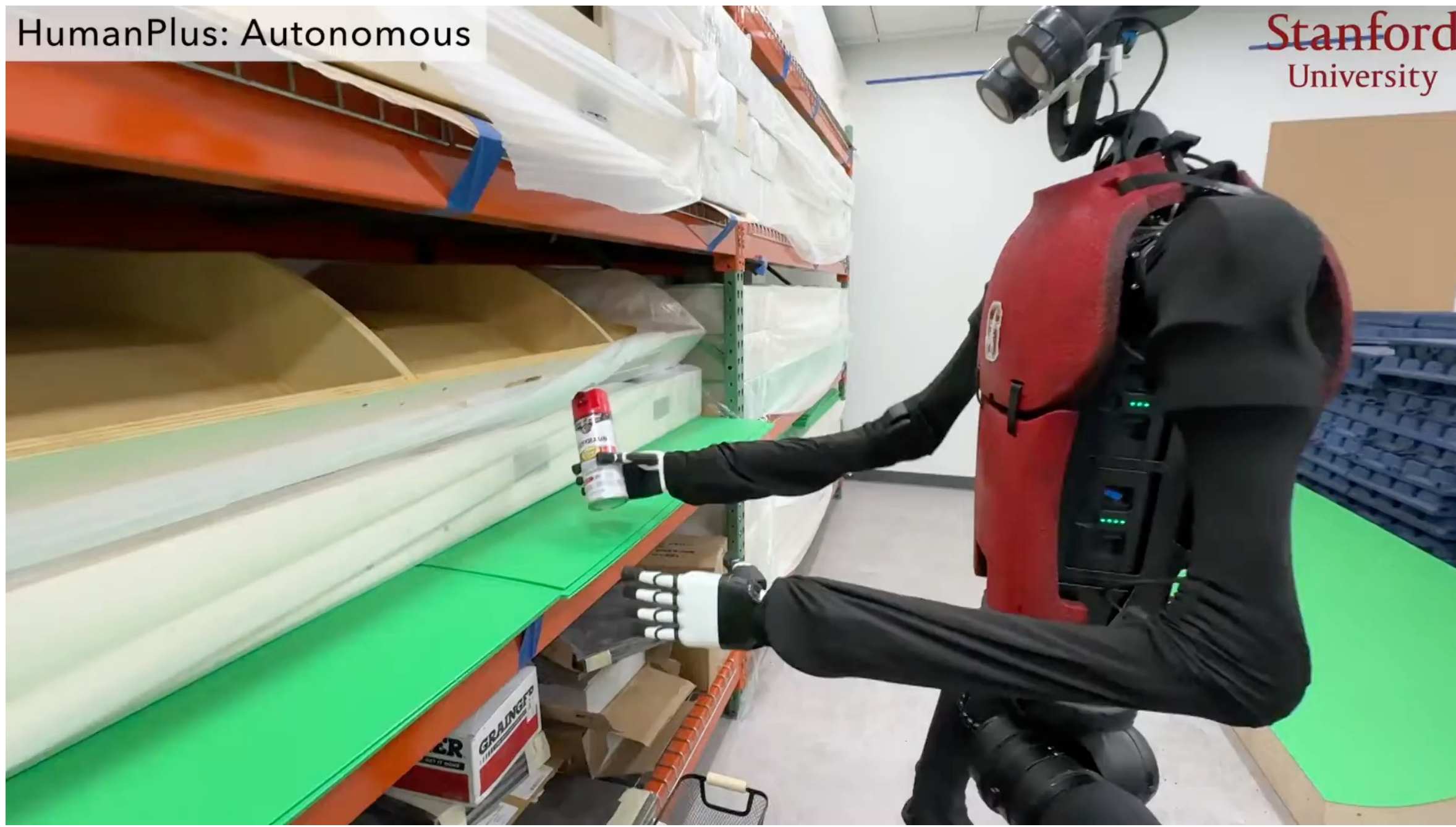


Learning-based Approaches

- Use Neural Networks to control/ plan arm motion E.g.:
- HumanPlus Humanoid Shadowing and Imitation from Humans
CoRL 2024
 - input: RGB & last action & robot proprioception info;
 - output: next action
 - using transformer-based network to predict the trajectory of each joint's pose.
 - Train via shadowing human operators
- <https://humanoid-ai.github.io/>

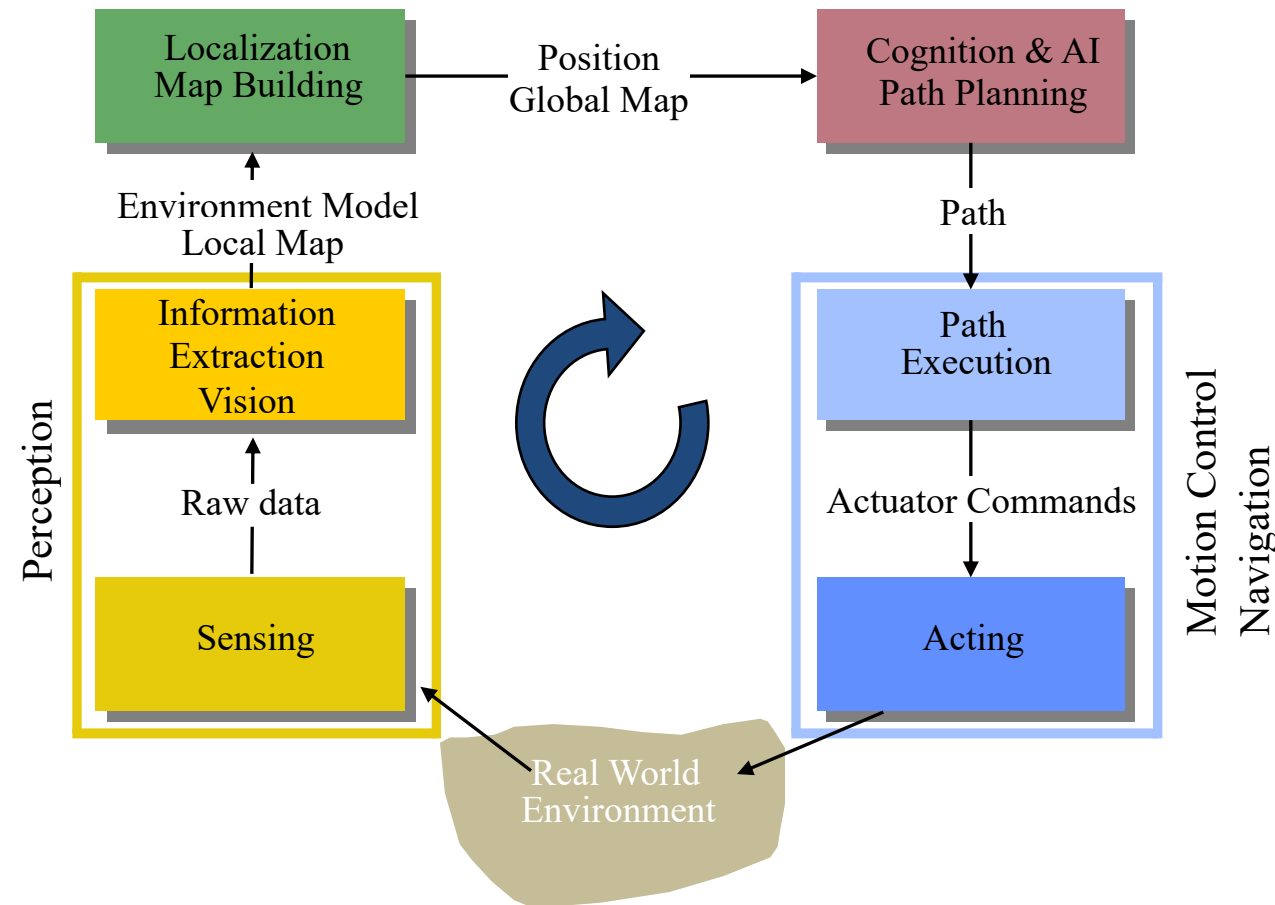
HumanPlus: Autonomous

Stanford
University



MOBILE ROBOTICS

General Control Scheme for Mobile Robot Systems



Mobile Robot Kinematics

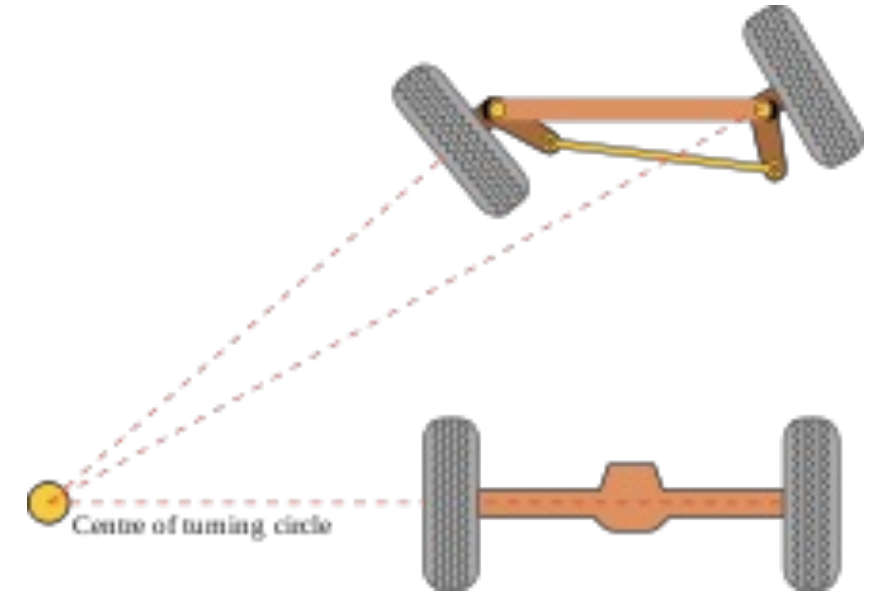
- Aim
 - Description of mechanical behavior of the robot for *design* and *control*
 - Similar to robot manipulator kinematics
 - However, mobile robots can move unbound with respect to its environment
 - there is no direct way to measure the robot's position
 - Position must be integrated over time
 - Leads to inaccuracies of the position (motion) estimate
 - > *the number 1 challenge in mobile robotics*

Differential Drive Robots



Ackermann Robot

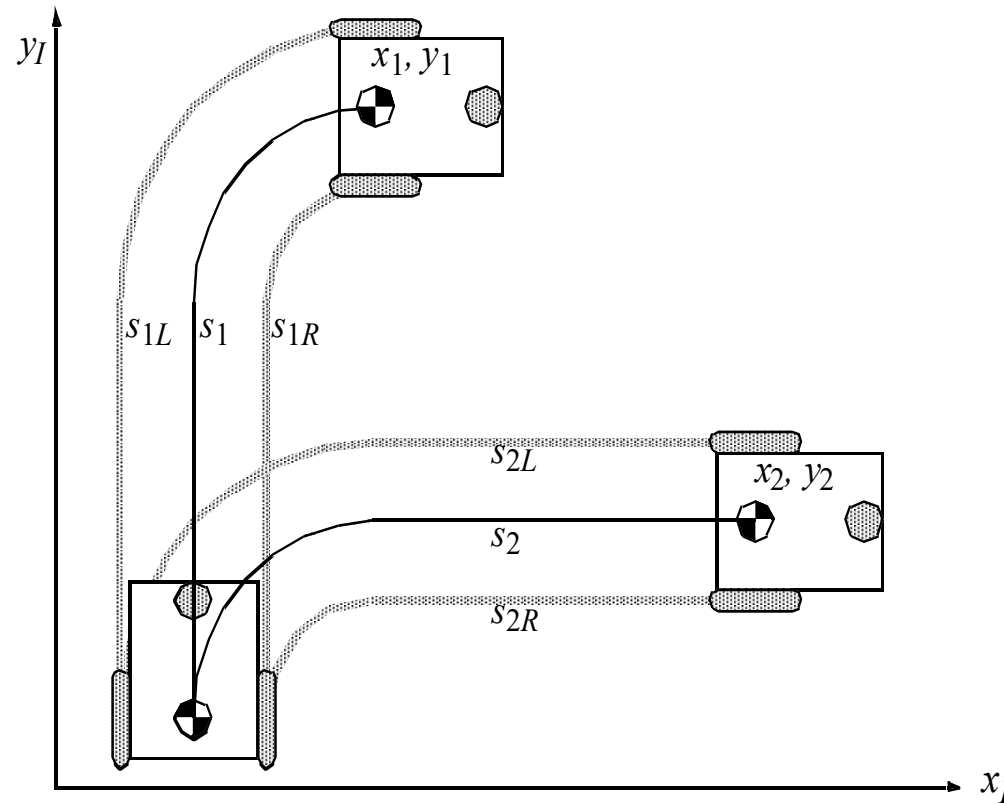
- No sideways slip than differential drive during turning 😊
- Cannot turn on the spot 😞



Mobile Robot Kinematics: Non-Holonomic Systems

$$s_1 = s_2; s_{1R} = s_{2R}; s_{1L} = s_{2L}$$

$$\text{but: } x_1 \neq x_2; y_1 \neq y_2$$



- Non-holonomic systems
 - differential equations are not integrable to the final pose.
 - the measure of the traveled distance of each wheel is not sufficient to calculate the final position of the robot. One has also to know how this movement was executed as a function of time.

MAPS & MAPPING

Map Representation: what is “saved” in the map

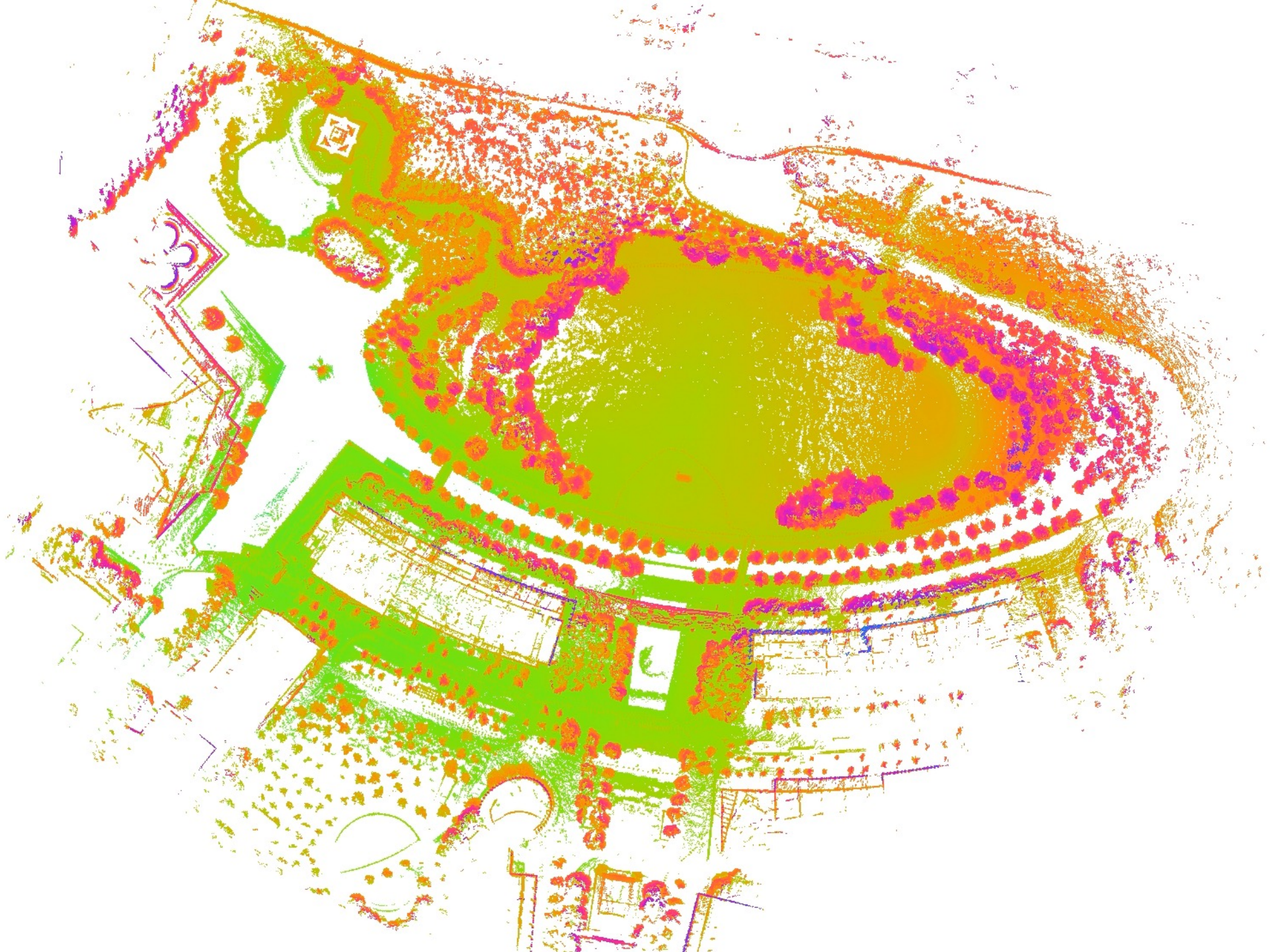
- Points (surface of objects, buildings): 2D or 3D
 - What: x,y or x,y,z coordinates;
Optional: intensity; maybe RGB; maybe descriptor; temperature; ...
 - From range sensors (laser, ultrasound, stereo, RGB-D): dense
 - From cameras (structure from motion; feature points): sparse
 - Variant: kd-tree
- Grid-map: 2D or 3D
 - Option: probabilistic grid map
 - Option: elevation map
 - Option: cost map
 - Option: Truncated Signed Distance Field
 - Option: Normal Distributions Transform (NDT)
 - Variant: Quad-tree; Oct-tree
- Higher-level Abstractions
 - Lines; Planes; Mesh
 - Curved: splines; Superquadrics
- Semantic Map
 - Assign semantic meaning to entities of a map representation from above
 - E.g. wall, ceiling, door, furniture, car, human, tree, ...
- Topologic Map
 - High-level abstraction: places and connections between them
- Hierarchical Map
 - Combine Maps of different scales. E.g.:
 - Campus, building, floor
- Pose-Graph Based Map
 - Save (raw) sensor data in graph, annotated with the poses; generate maps on the fly
- Dynamic Map
 - Capture changing environment
- Hybrid Map
 - Combination of the above

RGBD-Inertial Trajectory Estimation and Mapping for Ground Robots

Zeyong Shan, Ruijian Li and Sören Schwertfeger



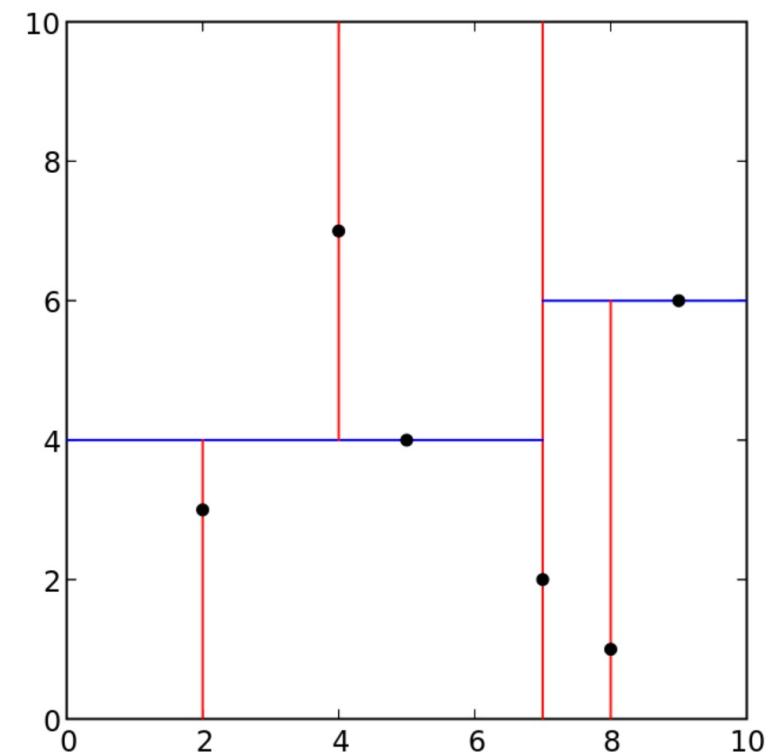
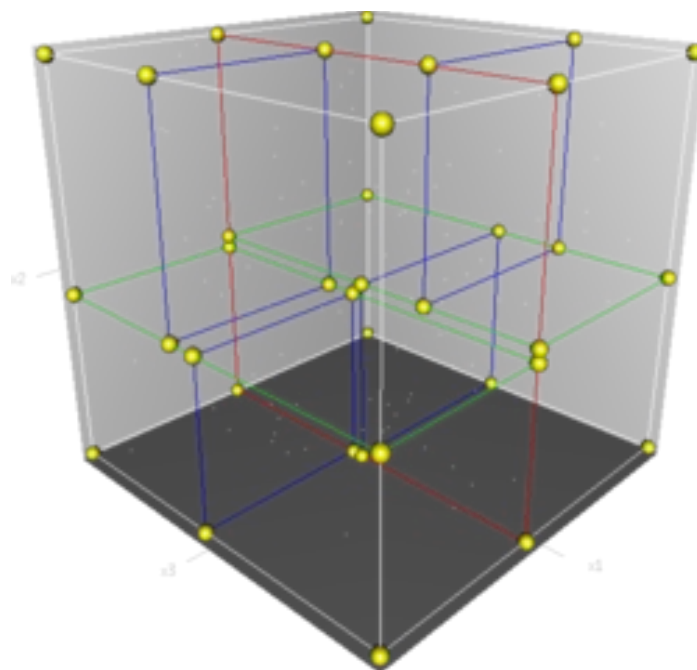
Source code: <https://github.com/STAR-Center/VINS-RGBD>



k-d tree

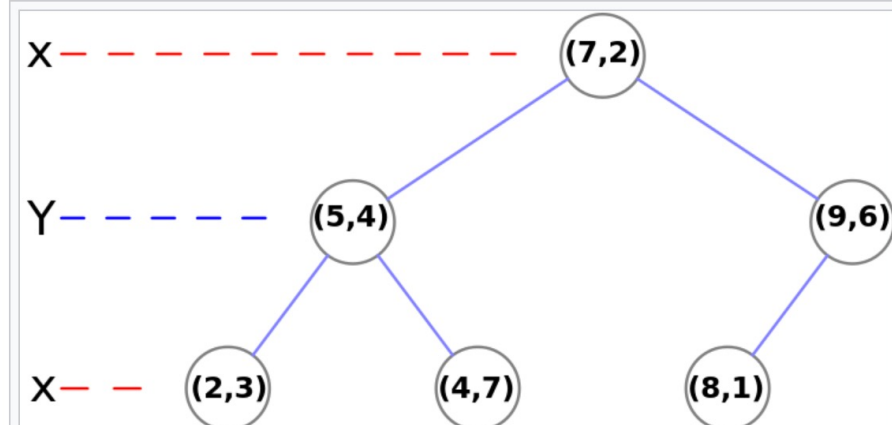
- k-dimensional binary search tree
- Robotics: typically 3D or 2D
- Every level of tree:
 - For a different axis (e.g. x,y,z,x,y,z,x,y,z) ($\Rightarrow$ split space with planes)
 - Put points in left or right side based on median point (w.r.t. its value of on the current axis) $\Rightarrow$
 - Balanced tree
- Fast neighbor search $\rightarrow$ ICP!

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$



k-d tree decomposition for the point set

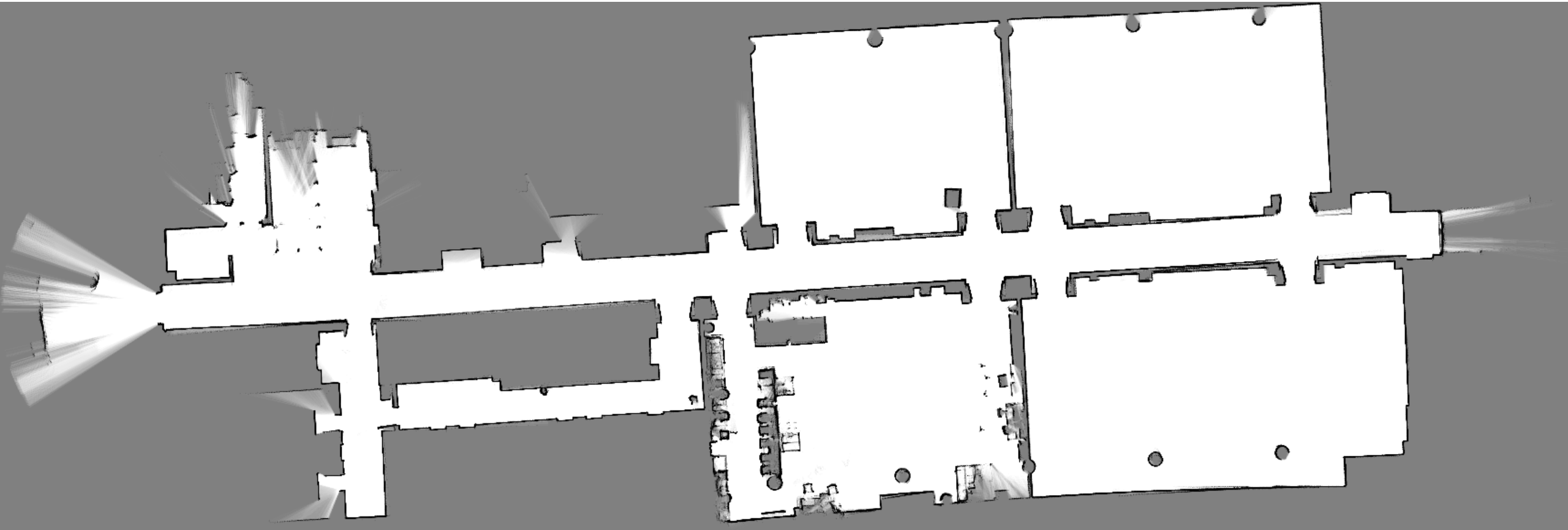
$(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)$.



The resulting k-d tree.

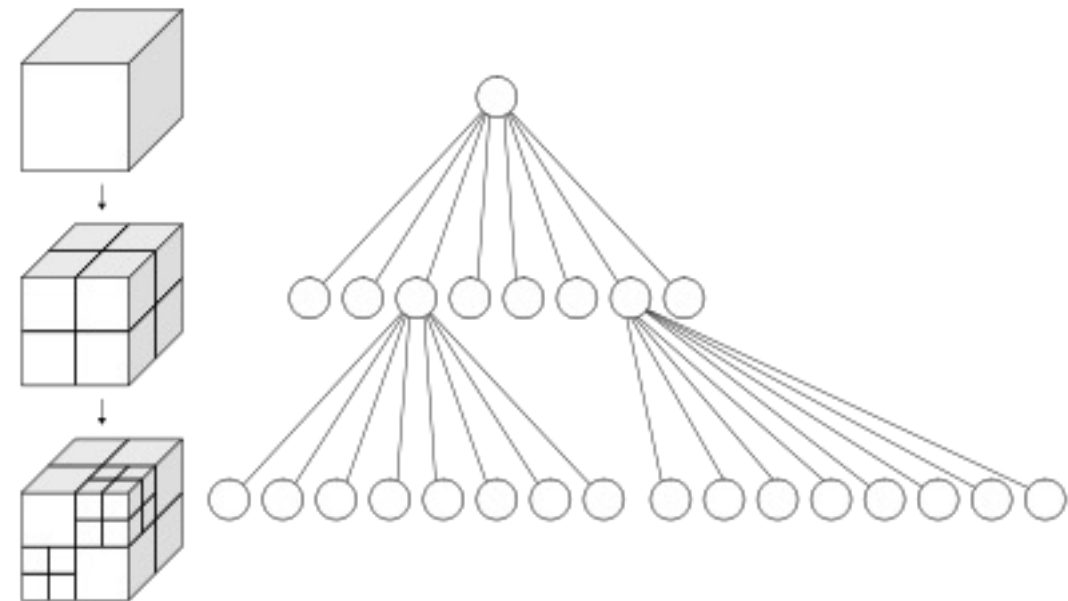
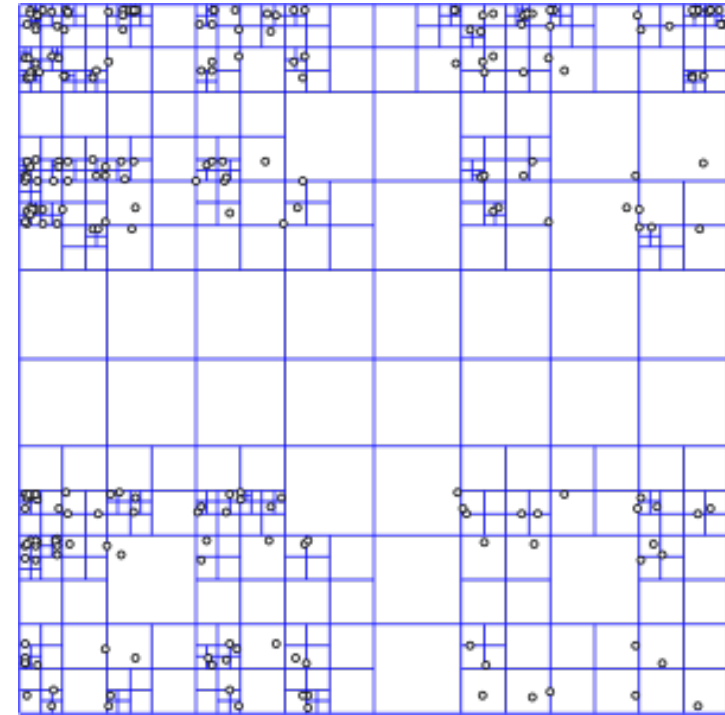
Probabilistic grid map

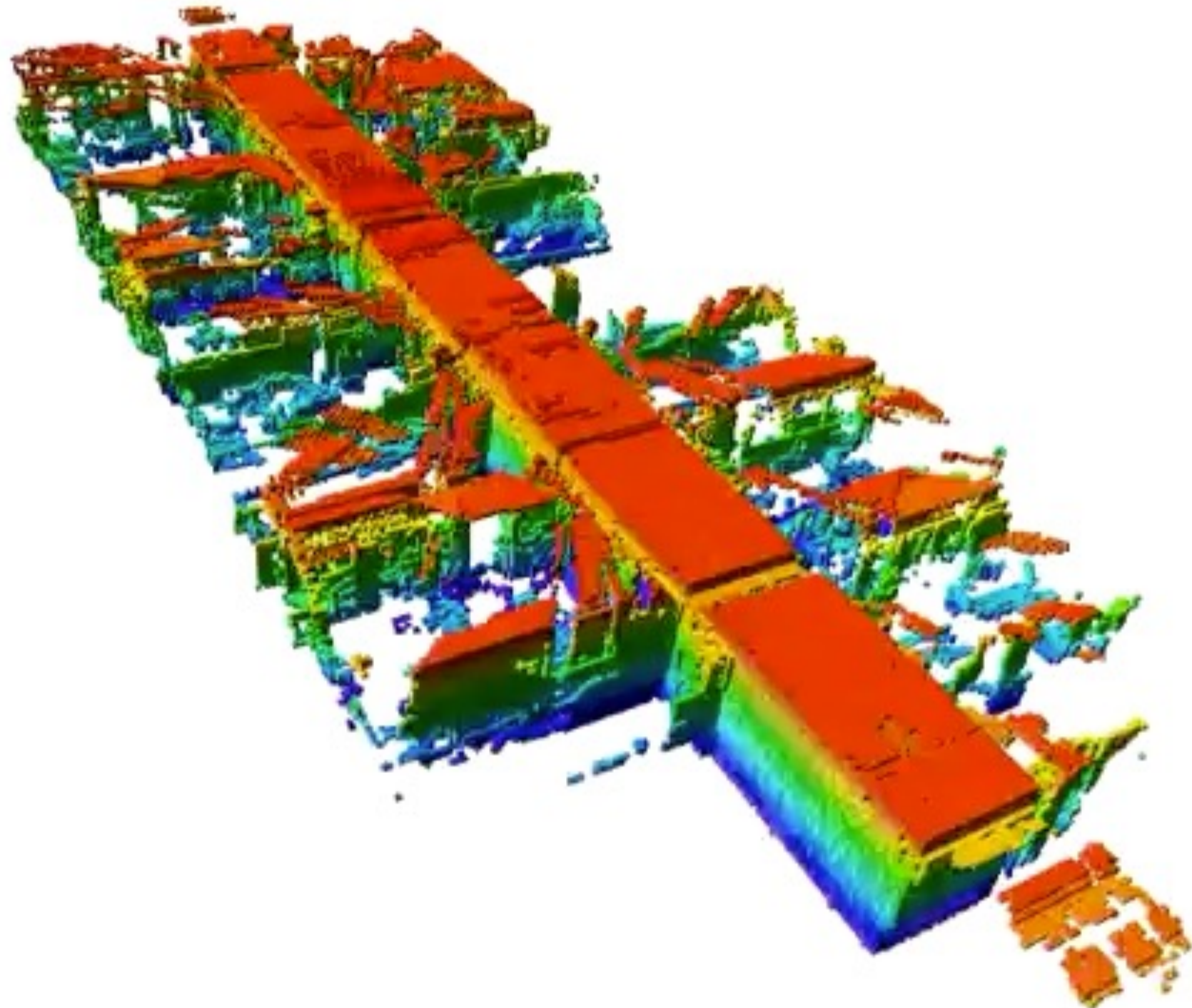
- 1: occupied; 0: free; 0.5: unknown
- Need error model of sensor (and of localization) to properly update cells with a scan
- Can remove dynamic (moving) objects (by observing the free space multiple times)



Adaptive Cell Decomposition

- Quad-tree
 - 2D map/ grid is recursively divided into 4 smaller cells
 - Only cells with different values/ points get divided further =>
 - Compact representation of big space (if many cells stay merged)
- Oct-tree
 - 3D grid divided into 8 smaller cells
 - Very compact! (There is lots of free space!)
- OctoMap: probabilistic oct-tree!
 - <http://octomap.github.io>
 - Good support in ROS (e.g. MoveIt!)





Semantic Information

- Assign labels to data
- Segmentation: automatically group data (e.g. points) according to their semantic class
- Even save just very high level data; e.g. room at (x,y); Eiffel tower; ...
- Applications:
 - Human Robot Interaction (“go to kitchen”)
 - Scene understanding
 - Navigation (detect road; detect door)
 - Localization
 - ...

Semantic Map

- Semantic Segmentation
 - In room (e.g. detect furniture & objects)
 - Outdoors

