USC Viterbi School of Engineering

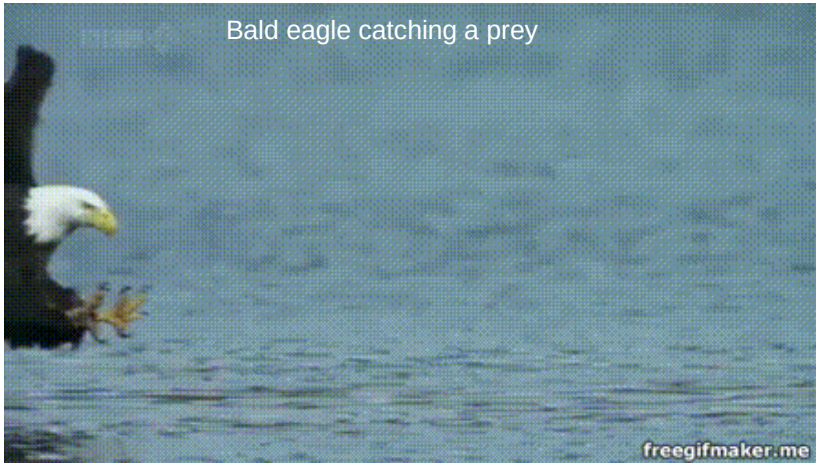
# Mobile Manipulation

*Satyandra K. Gupta*

Director, Center for Advanced Manufacturing
Smith International Professor
Aerospace and Mechanical Engineering Department
Computer Science Department
Viterbi School of Engineering
University of Southern California
https://sites.usc.edu/skgupta/

# What is Mobile Manipulation?

Manipulating an object from a moving platform

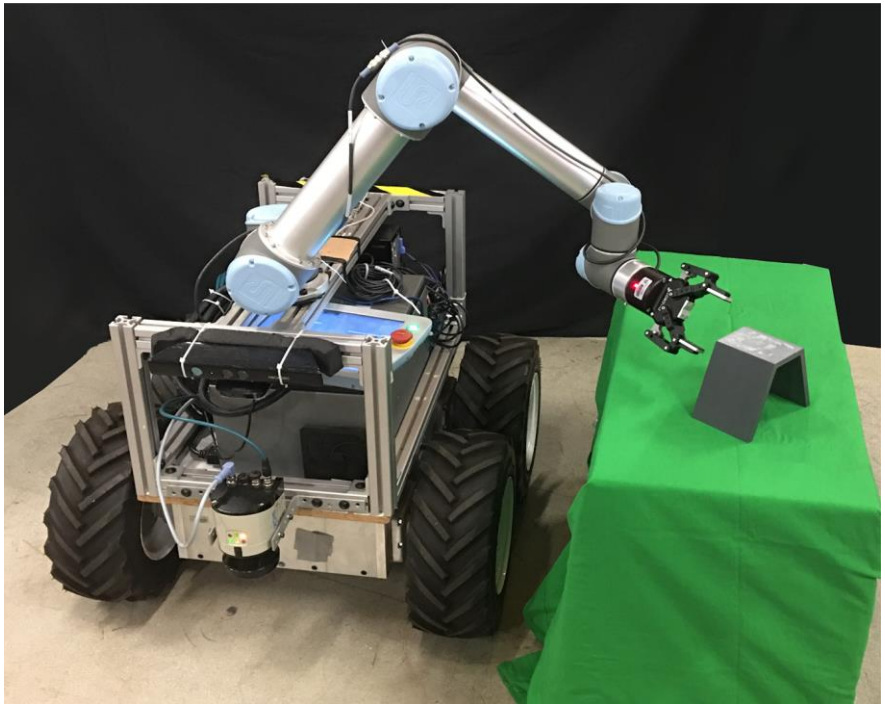




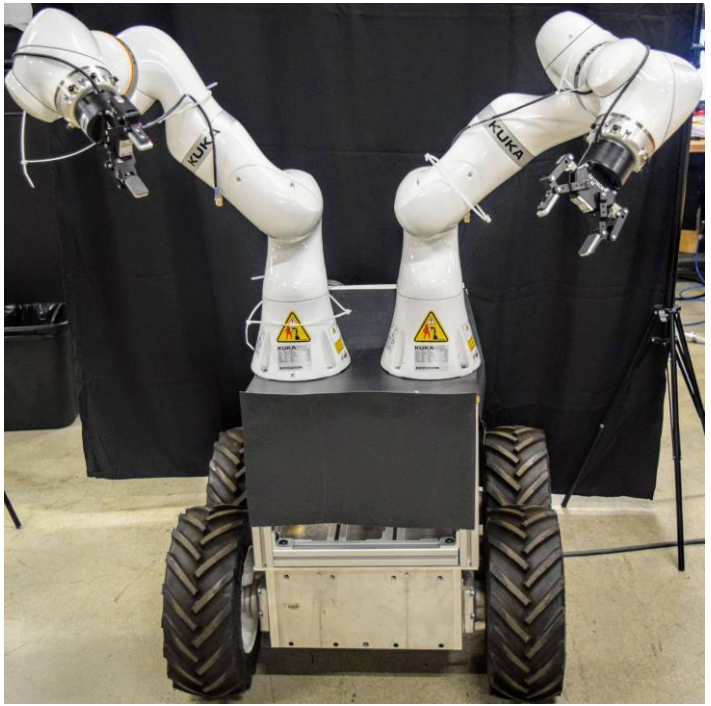
Natural examples of mobile manipulation

USC Viterbi School of Engineering

# Examples of Mobile Manipulators



Mobile Manipulator



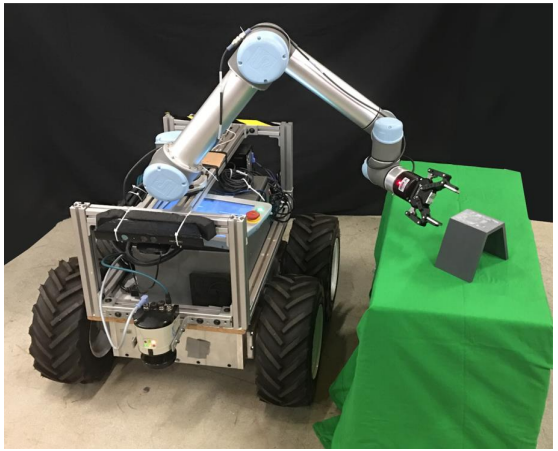

Bimanual Mobile Manipulator

USC Viterbi School of Engineering

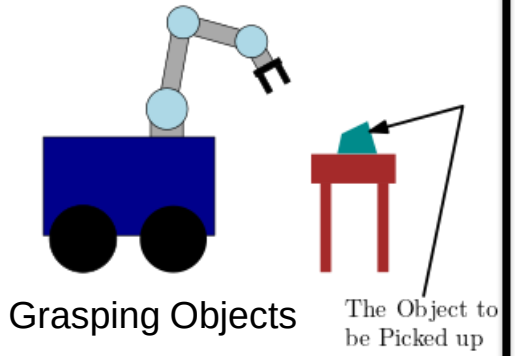
# Objectives

To develop computational foundations for the autonomous operation of mobile manipulators

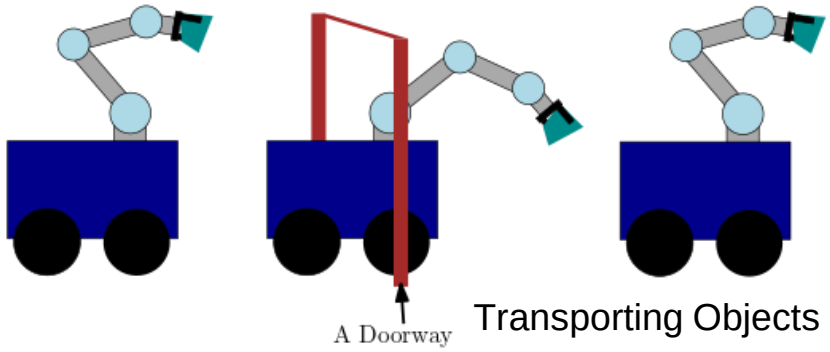
- Automate mobile manipulator motion planning
- Generate smooth motion plans for the robot
- Generate these smooth motion plans quickly



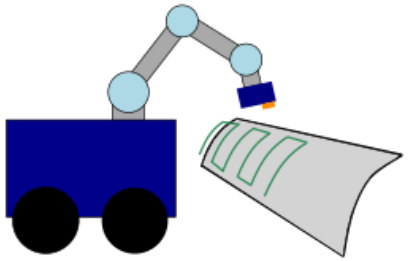
# Required Planning Capabilities




Grasping Objects



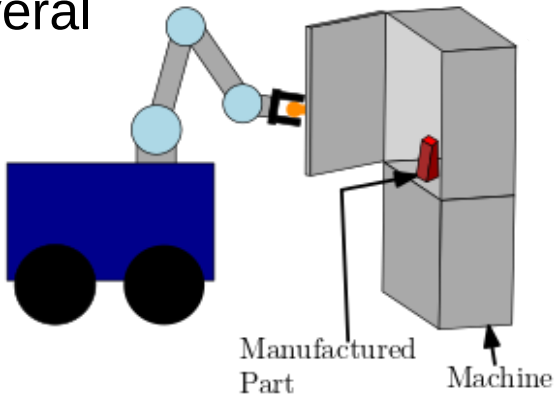

Transporting Objects

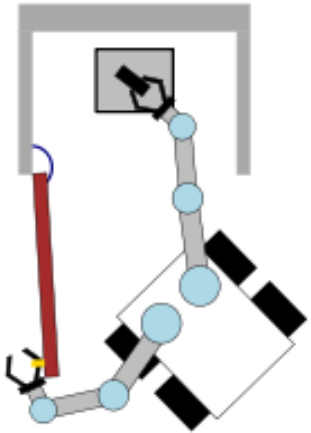


Area Coverage

With these planning capabilities, several complex tasks can be performed;
Example: Machine tending

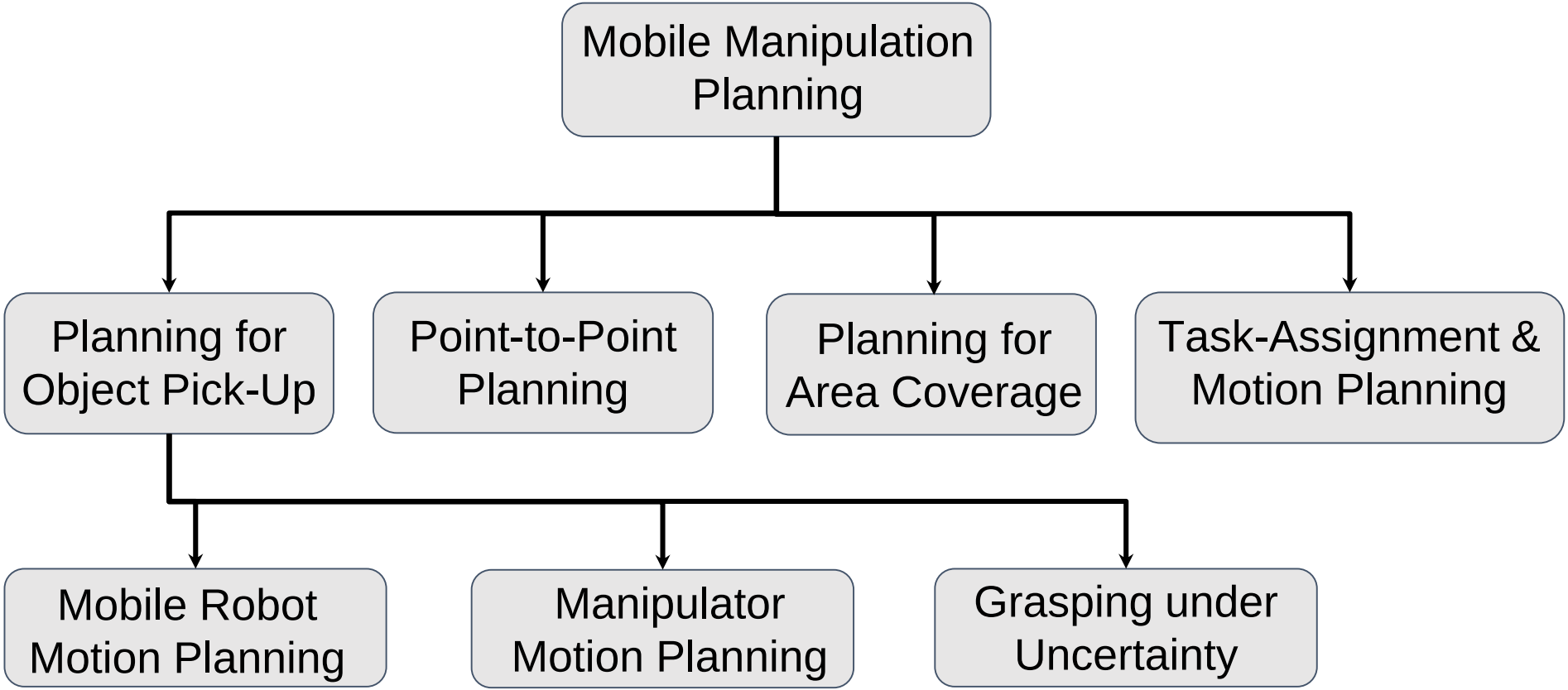
- Opening Door
- Picking up part
- Moving the part to new station



USC Viterbi School of Engineering

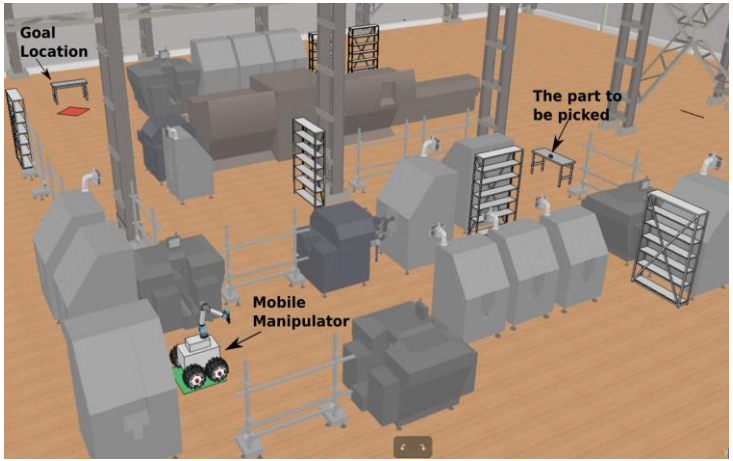
# Overview



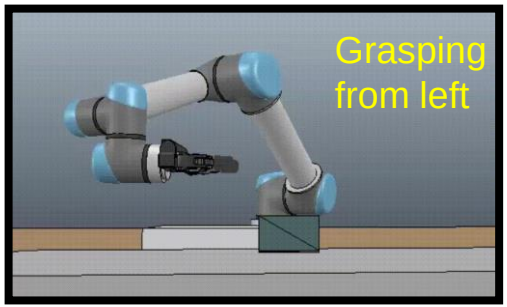
# Time-Optimal Trajectory Planning for Pick-and-Transport Operation with a Mobile Manipulator

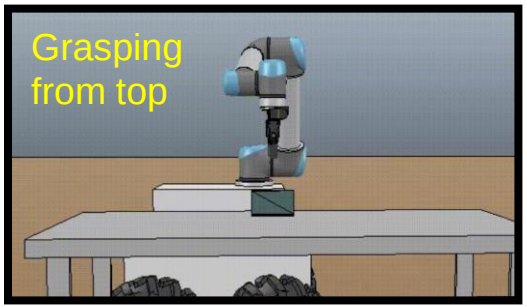
# Motivation & Objective

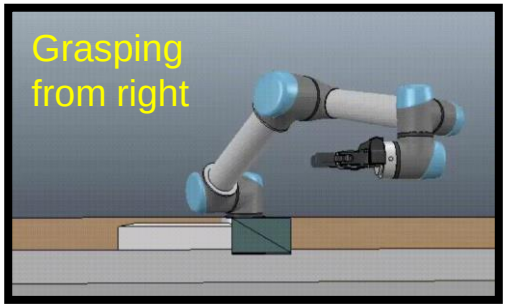
- **Inputs :**
  - Initial and final configurations of the mobile manipulator
  - Static map
  - Part pose
  - Maximum joint rates
  - Grasping strategy
- **Output :**
  - Time-optimal mobile base trajectory for picking up objects from a known location




Example factory setting (generated using V-rep)

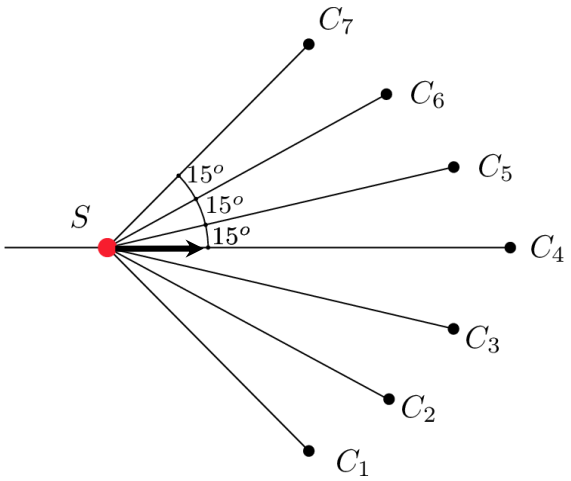


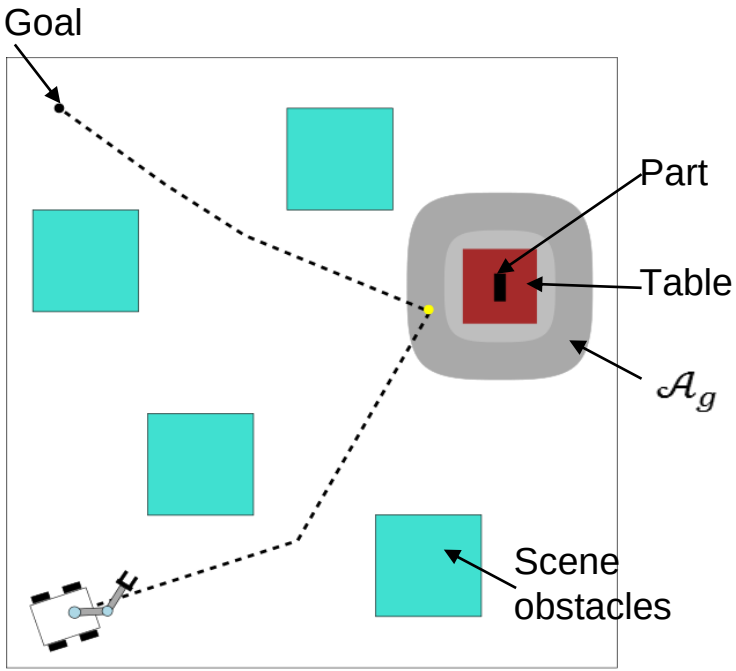



# Overview of Approach

- Hybrid A* based motion planner for the mobile base in 3D ($x$, $y$, $\varphi$)
- Constant-time motion primitives
- Specilized heuristics to guide the search



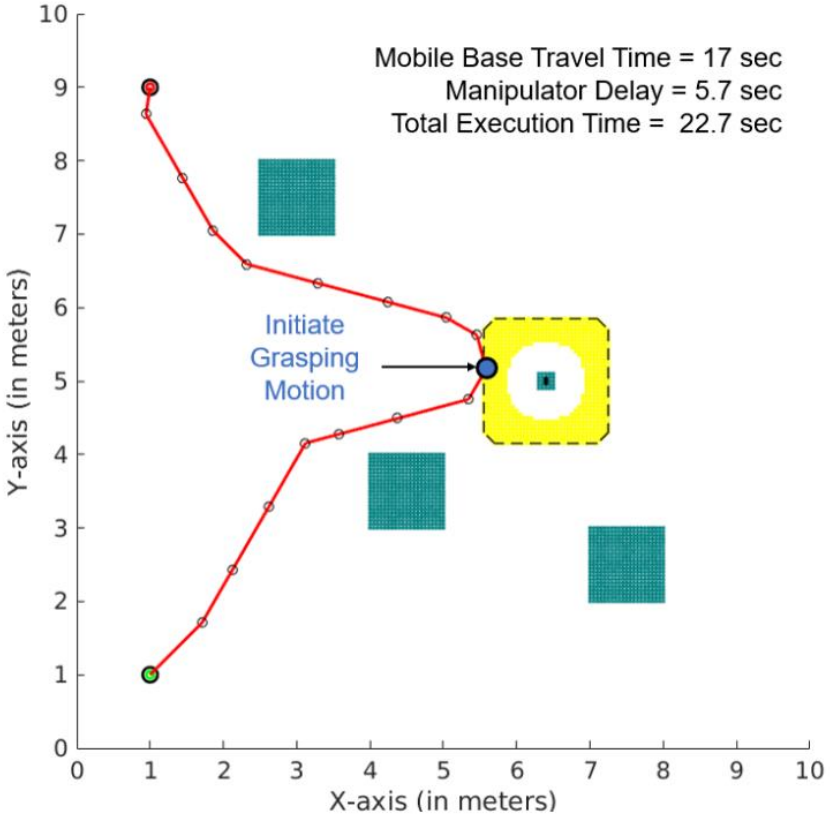

Constant-time motion primitives

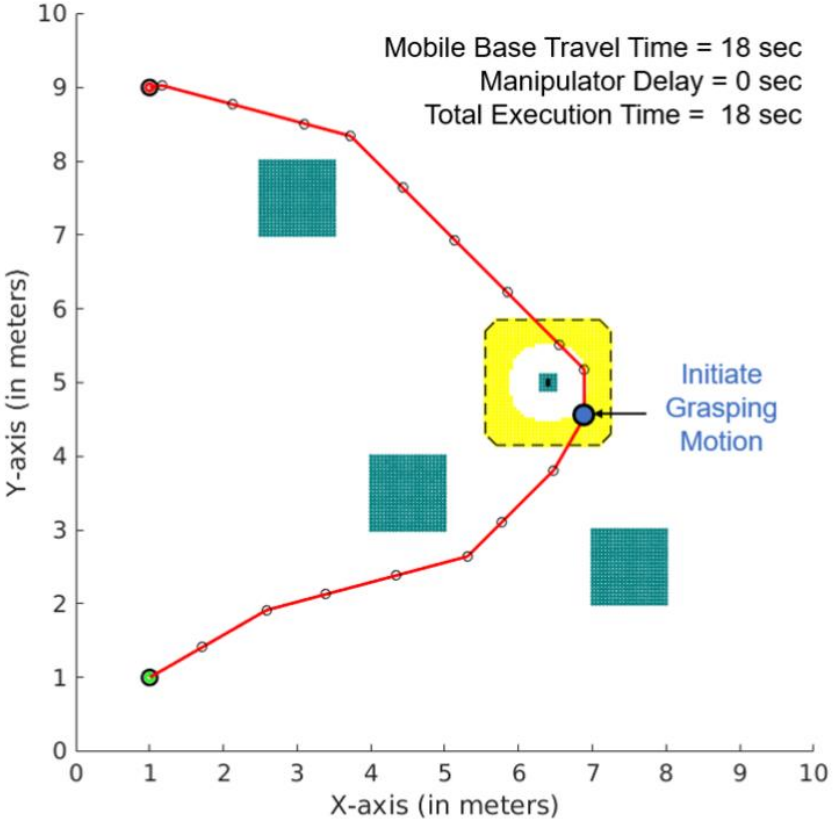



A typical Solution Path

USC Viterbi School of Engineering

# Resulting Trajectories



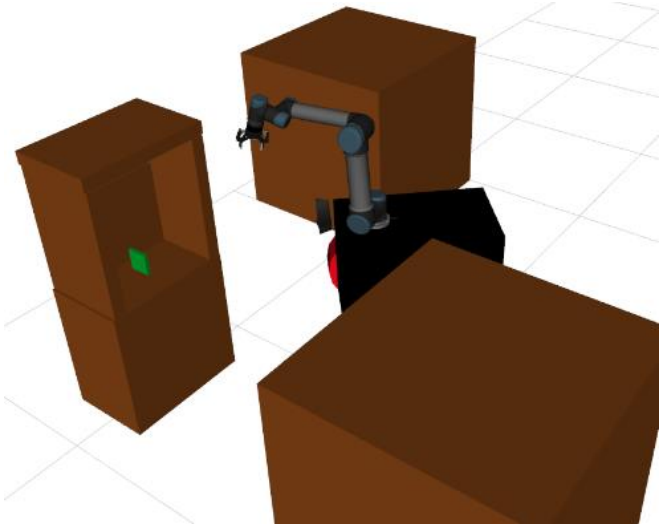



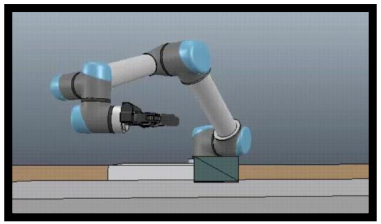

# Manipulator Trajectory Planning on a Moving Mobile base for Part Pick-up and Transport Operations

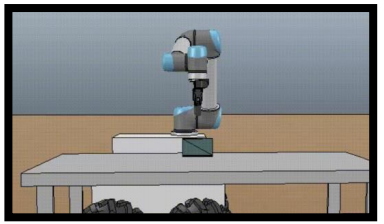
USC Viterbi School of Engineering

# Motivation & Objective

- To determine the manipulator motion for grasping the part while the mobile base moves along a predetermined path
- The manipulator should start moving as late as possible
- The mobile base should slow down only if necessary

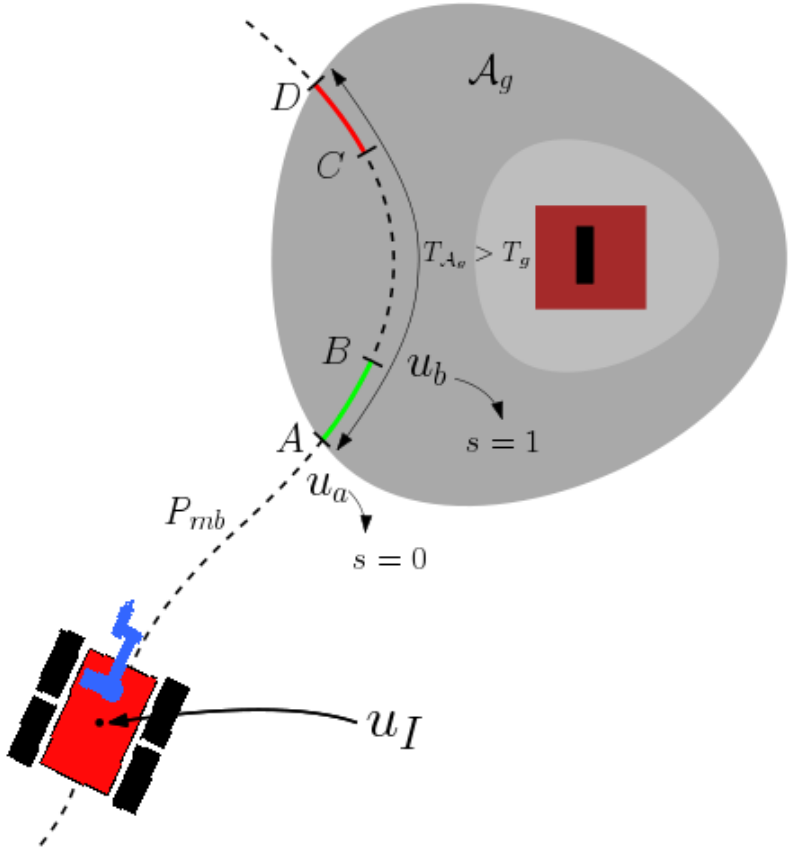






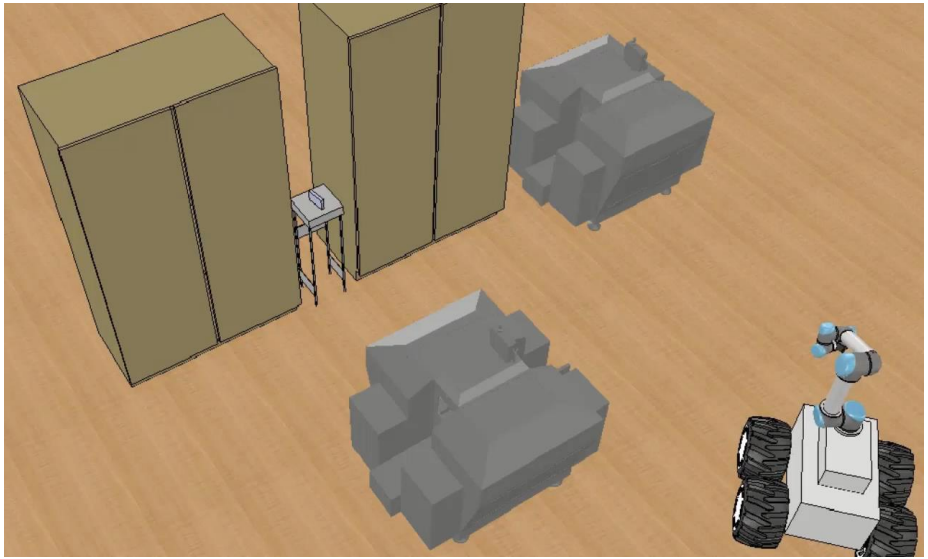
# Overview of Approach

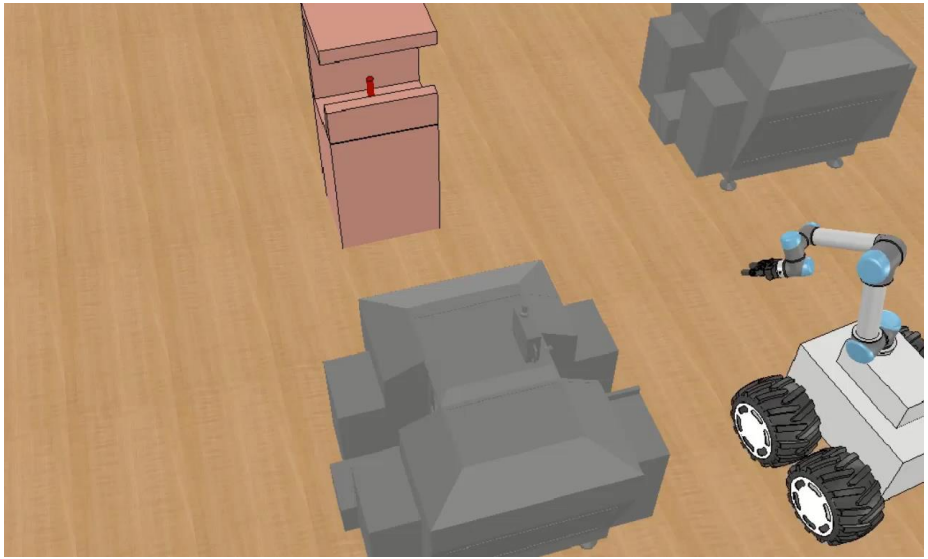
- Baseline method is to grow multiple trees, one for each grasping strategy
- Grow a single tree from the initial configuration
- Connect to a tree at random and that gives a baseline path
- Three techniques to speed up the motion planning, generate smooth paths, select appropriate goal configuration and the grasping strategy were developed



USC Viterbi School of Engineering

# Results

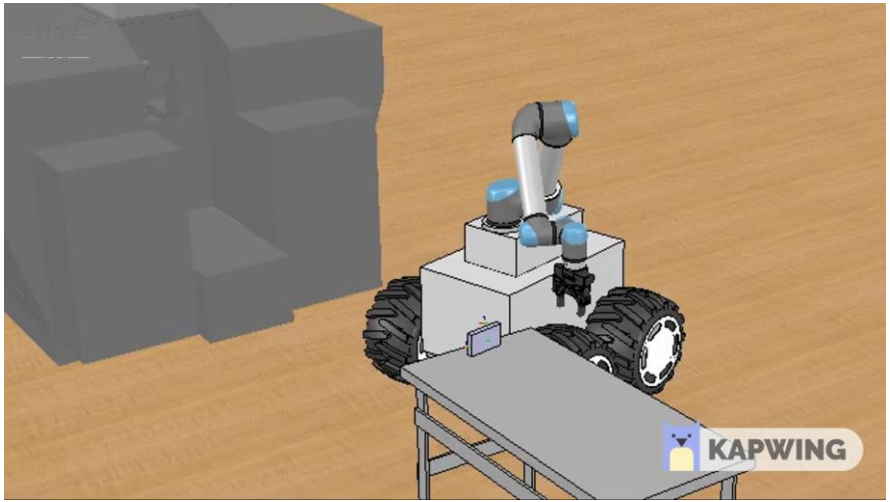


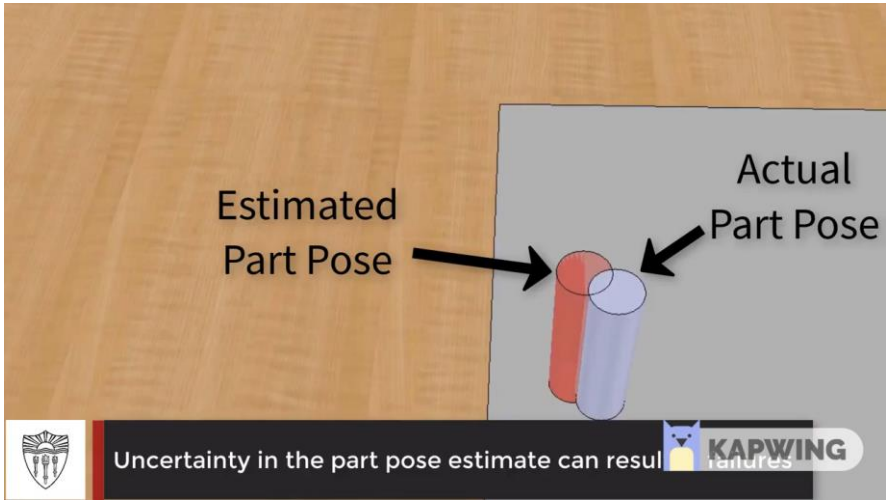


# Accounting for Part Pose Estimation Uncertainties during Trajectory Generation for Part Pick-Up Using Mobile Manipulators

USC Viterbi School of Engineering

# Motivation




To reduce time, mobile manipulators can pick-up objects while the mobile base & gripper are moving

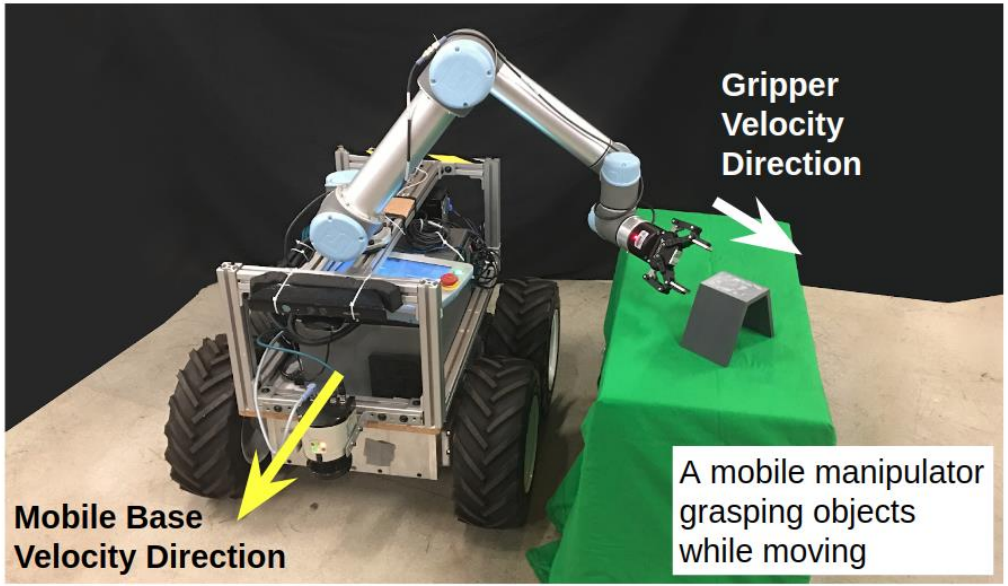
Uncertainty in the pose of the object may result in failure to grasp

USC Viterbi School of Engineering

# Objectives

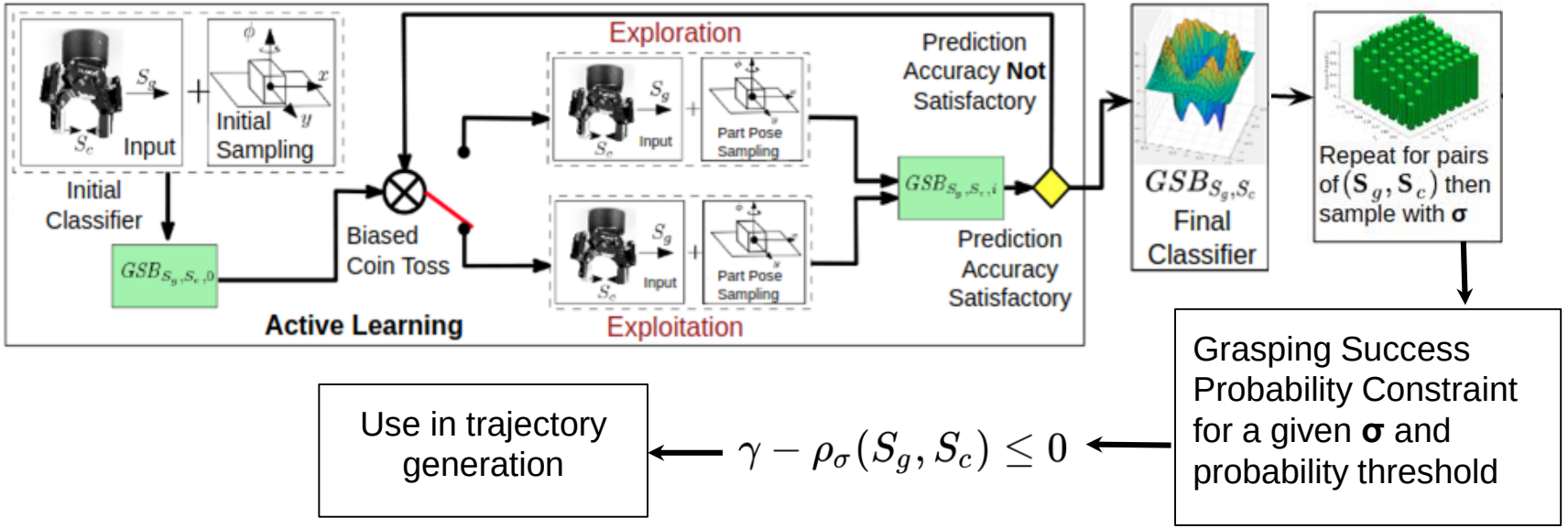
- Compute the gripper speed and the gripper closing speed so that there is a high probability of success in grasping

- Determine the mobile manipulator trajectory so that the gripper moves with that desired speed



# Overview of Approach

- SVM based active learning for determining the meta-model
- Successive refinement based parameter optimization for trajectory generation of the mobile manipulator



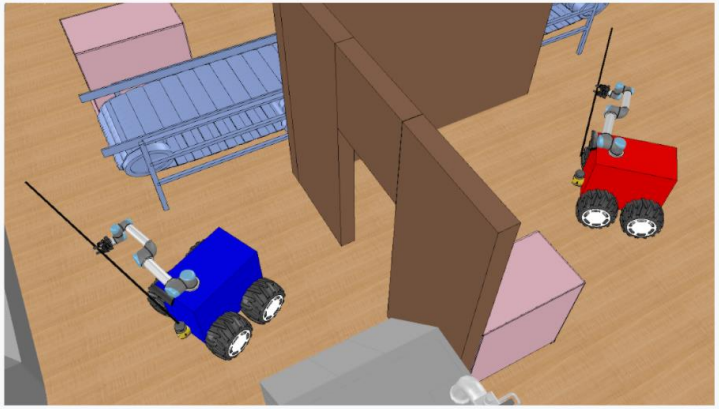
# Accelerating Bi-Directional Sampling-Based Search for Motion Planning of Non-Holonomic Mobile Manipulators

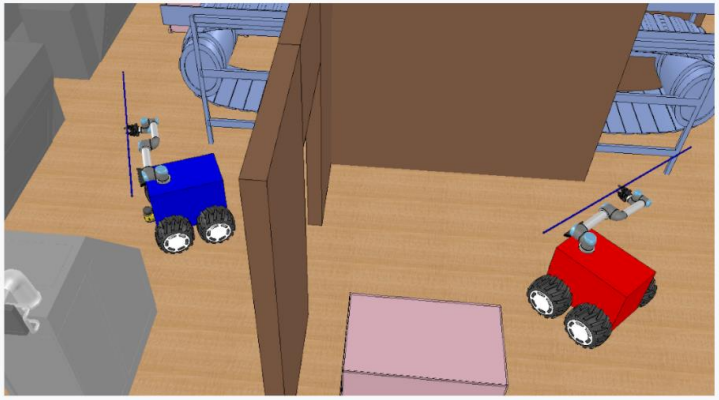
USC Viterbi School of Engineering

# Motivation

- For transportation tasks with a mobile manipulator, the manipulator can be at a “home” position on the mobile base and move after the mobile base arrives at the target location

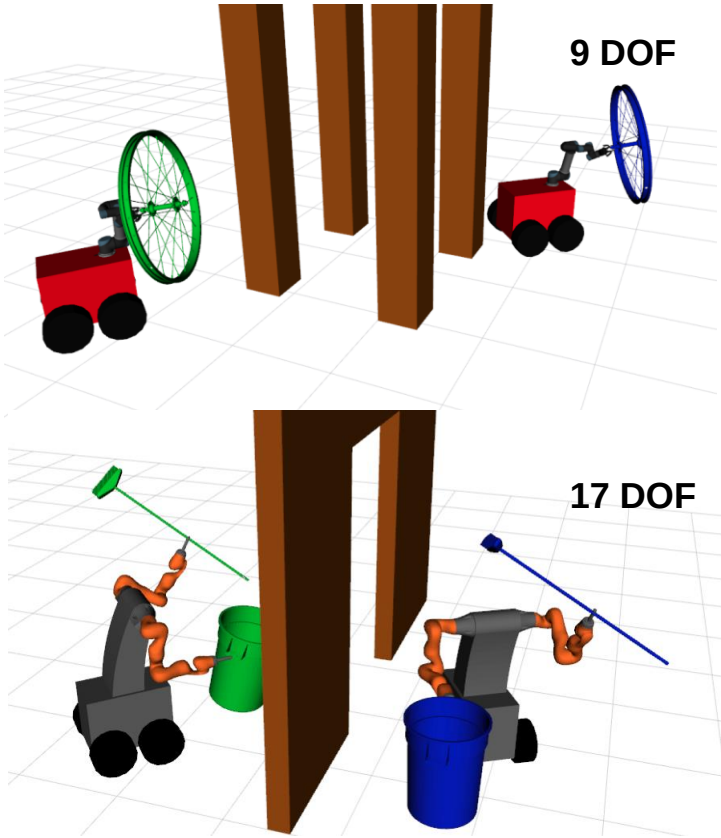
- In cases where large objects are to be transported in narrow passages, there may be a need to simultaneously move the manipulator and the mobile base.





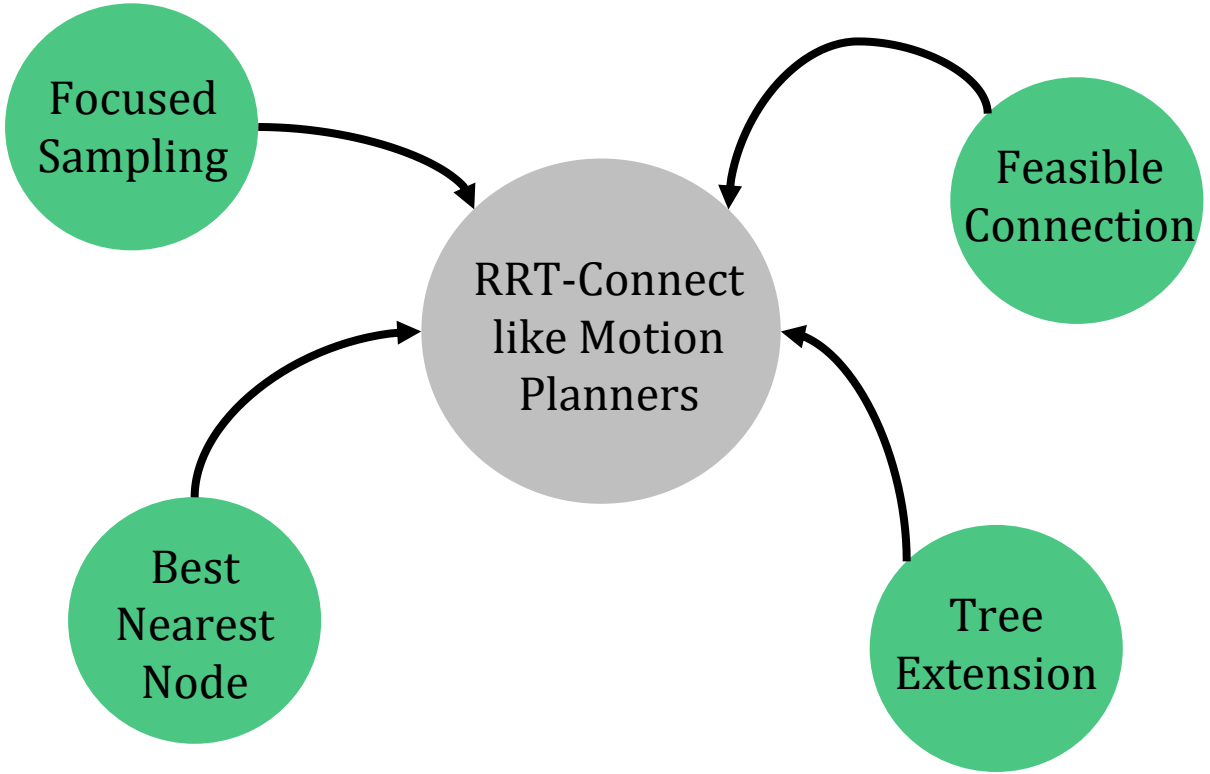
# Problem Statement

- Given:
  - Start and goal configurations of the nonholonomic mobile manipulator (3+n DOF) with a single or multiple manipulators mounted on the mobile base
  - The 3D environment with obstacles
- To Find:
  - A feasible smooth path of the mobile manipulator from the start to the goal configuration that satisfies the nonholonomic motion constraints



USC Viterbi School of Engineering

# Overview



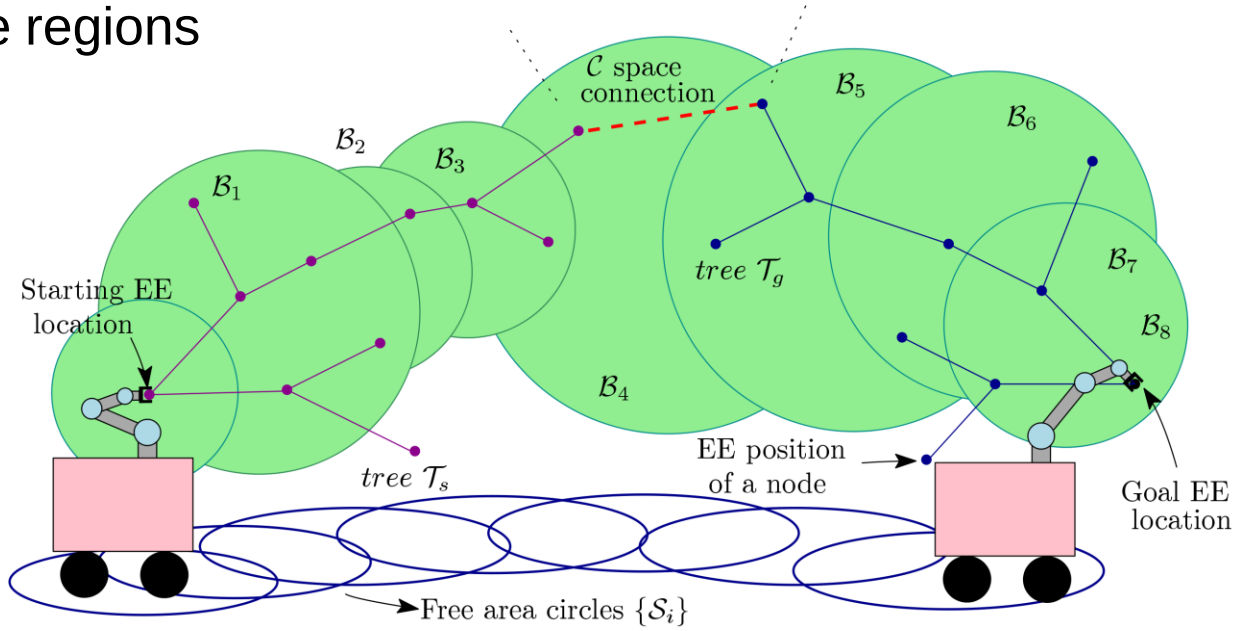
USC Viterbi School of Engineering

# The HS-Bi-RRT Algorithm

- The Hybrid Sampling based Bi-directional RRT (HS-Bi-RRT) is the overall algorithm
- $R_s$ is called the Sampling Ratio, i.e. the percentage of the time the sampling is done in the (3+n)-D Configuration Space (with mobile base in the workspace disks) vs the 6-D pose in Workspace
- To maintain the completeness property, we sample in the entire (3+n)-D Configuration space for a small fraction of time

# The HS-Bi-RRT Algorithm

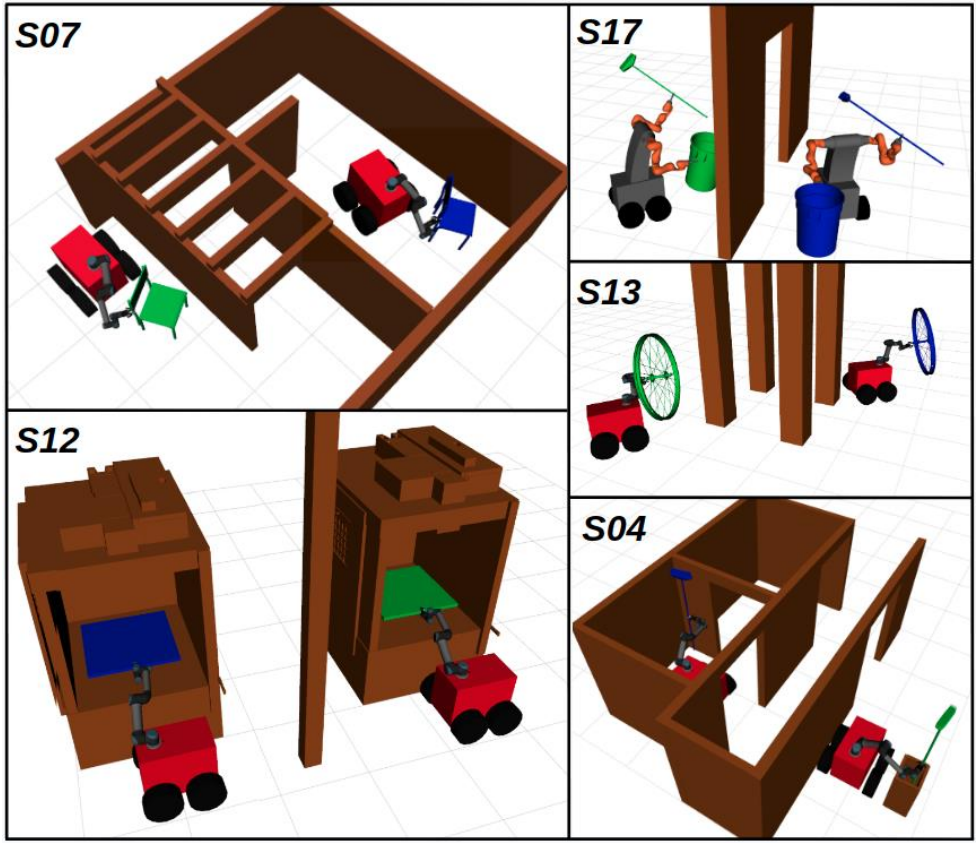
- Sampling is done inside the workspace sphere and the workspace disks from a normal distribution with the mean at their center and a varying standard deviation
- Resulting in Exploration-Exploitation type sampling to grow the trees in appropriate regions



USC Viterbi School of Engineering

# Test Cases

- 20 different test cases: Combination of a variety of large objects being carried in varied environments
- Bi-manual mobile manipulator carrying two different objects in two different hands



USC Viterbi School of Engineering

# Results

- HS-Bi-RRT has been compared with 3 other competing methods. The connection Heuristics are used in all methods:
  1. WD-Bi-RRT: Workspace Disks Bi-RRT, sampling for mobile base only in the disks
  2. WS-Bi-RRT: Workspace Bi-RRT, sampling only in the workspace spheres
  3. Bi-RRT+ : Bi-RRT with sampling in the entire configuration space and the connection heuristics

| Scene | Average Computation Time (s) | | | | Average Path Cost (m) | | | |
|---|---|---|---|---|---|---|---|---|
| | HS-Bi-RRT | WD-Bi-RRT | WS-Bi-RRT | Bi-RRT+ | HS-Bi-RRT | WD-Bi-RRT | WS-Bi-RRT | Bi-RRT+ |
| *S04* | 37.5 | 163.4 | 832.0 | 191.9 | 20.0 | 23.9 | 16.6 | 25.0 |
| *S07* | 32.9 | 175.8 | 727.7 | 200.4 | 26.0 | 29.1 | 20.4 | 49.2 |
| *S12* | 20.9 | 75.7 | 51.0 | 101.5 | 18.7 | 60.3 | 10.2 | 72.6 |
| *S13* | 19.1 | 33.0 | 131.2 | 29.6 | 19.7 | 26.2 | 12.8 | 35.6 |
| *S17* | 77.3 | 302.4 | 802.1 | 399.7 | 76.2 | 89.1 | 51.1 | 99.7 |

USC Viterbi School of Engineering

# Results

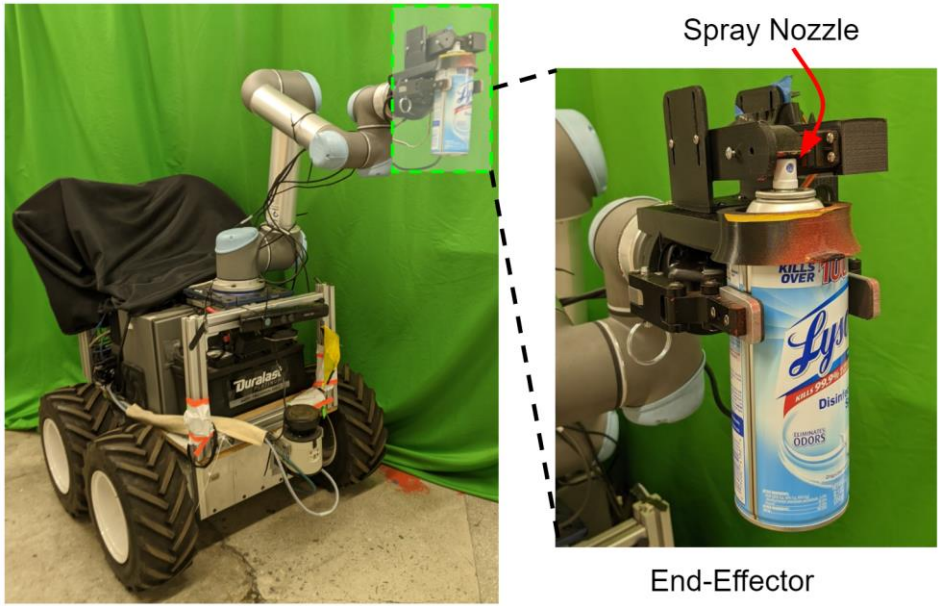
- Without the connection heuristics, there is high failure rates in all methods:
  1. HS-Bi-RRT: 73%
  2. WD-Bi-RRT: 81%
  3. WS-Bi-RRT: 71%
  4. Bi-RRT+ : 90%

- Also, in cases when there is a success, on average the computation time is ~9 times higher as compared to when the connection heuristics are used

# Area Coverage Planning for Spray-Based Surface Disinfection with a Mobile Manipulator

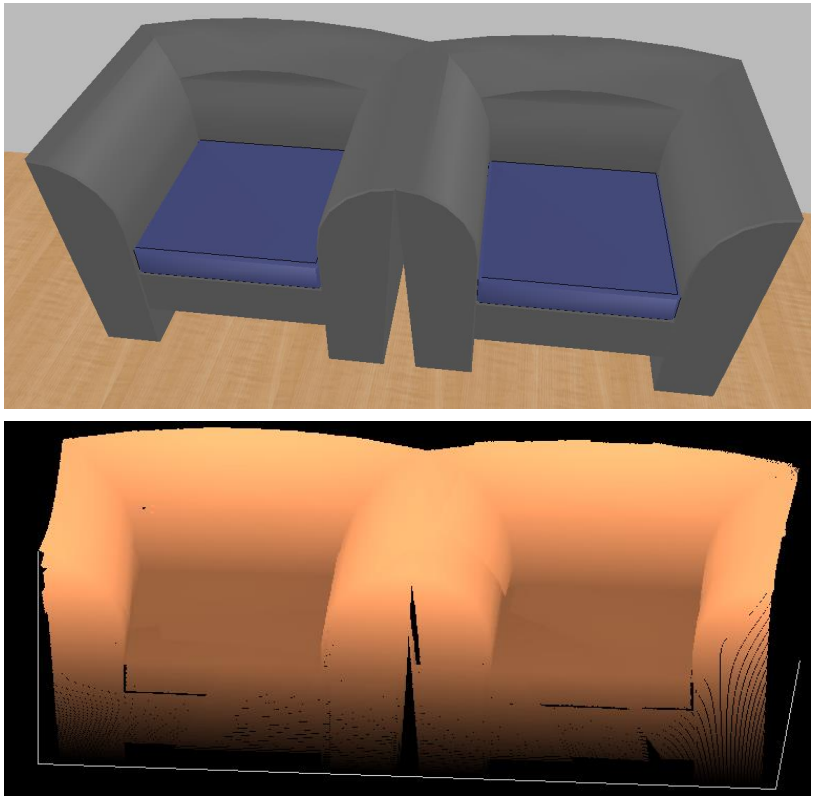
# Motivation

- Manual disinfection can be a time-consuming, risky, labor-intensive, and mundane, and humans may fail to disinfect critical areas due to fatigue
- Mobile manipulators mounted with a spray nozzle at the end-effector can be very effective in spraying disinfectant liquid for deep disinfection of objects and surfaces



# Objective

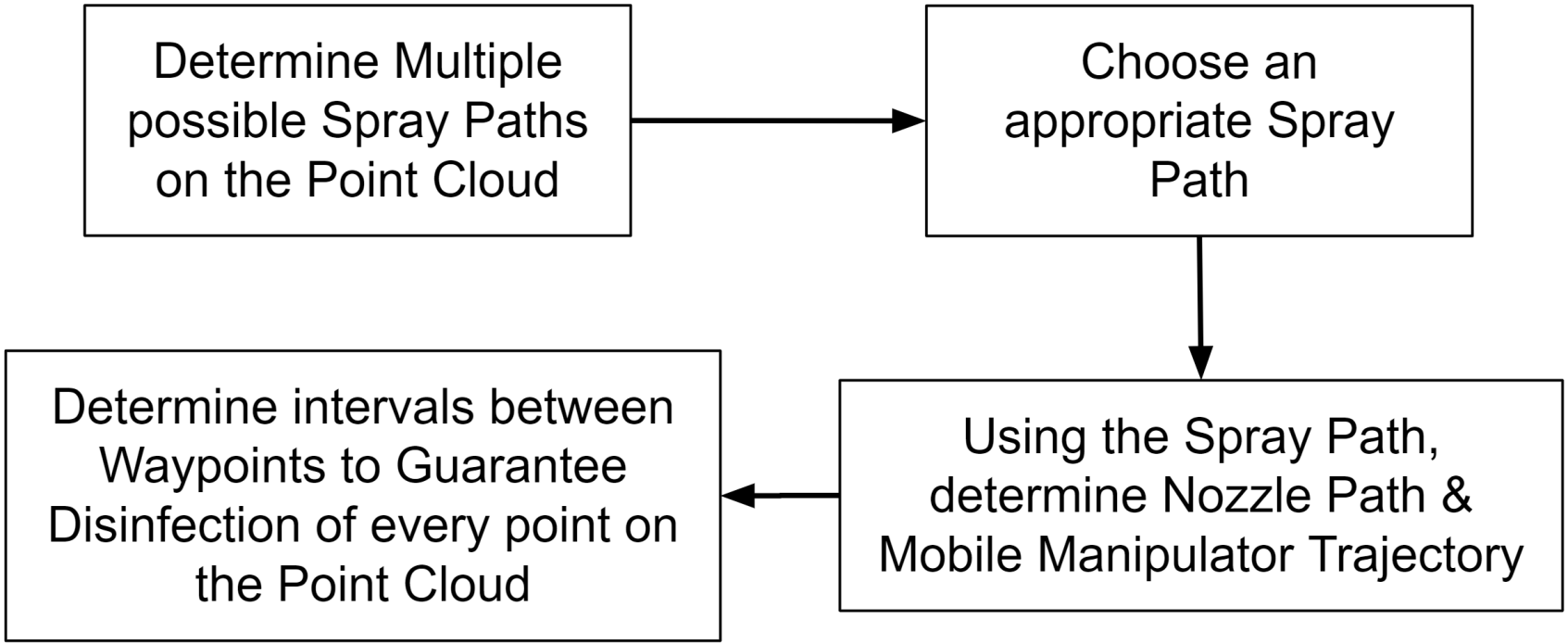
- Given a point cloud, the objective is to:
  - (a) Compute a mobile manipulator trajectory such that the spray nozzle motion results in the entire area of the surface being covered
  - (b) Compute the joint velocities such that each point on the surface receives enough disinfectant to guarantee thorough disinfection



A point cloud of the chairs

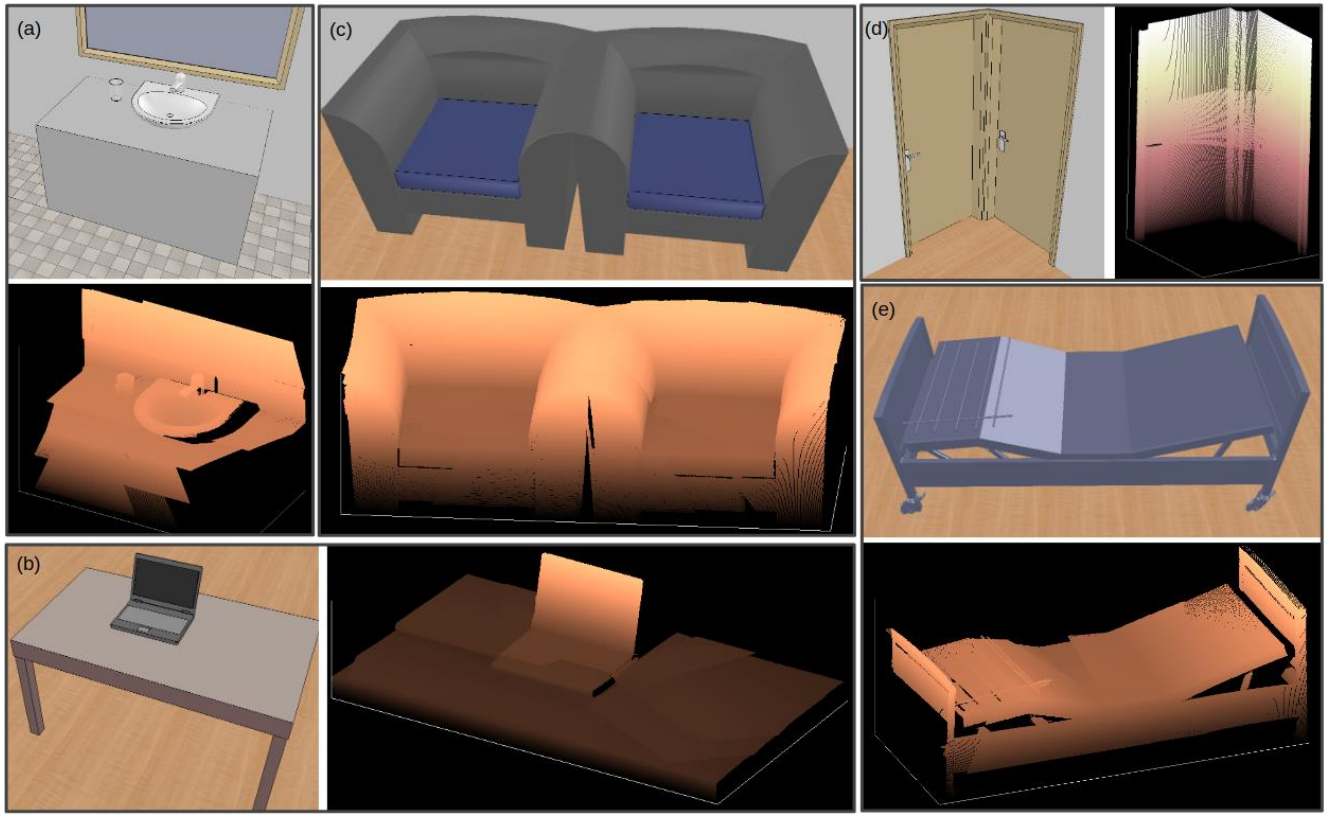
USC Viterbi School of Engineering

# Overview



USC Viterbi School of Engineering

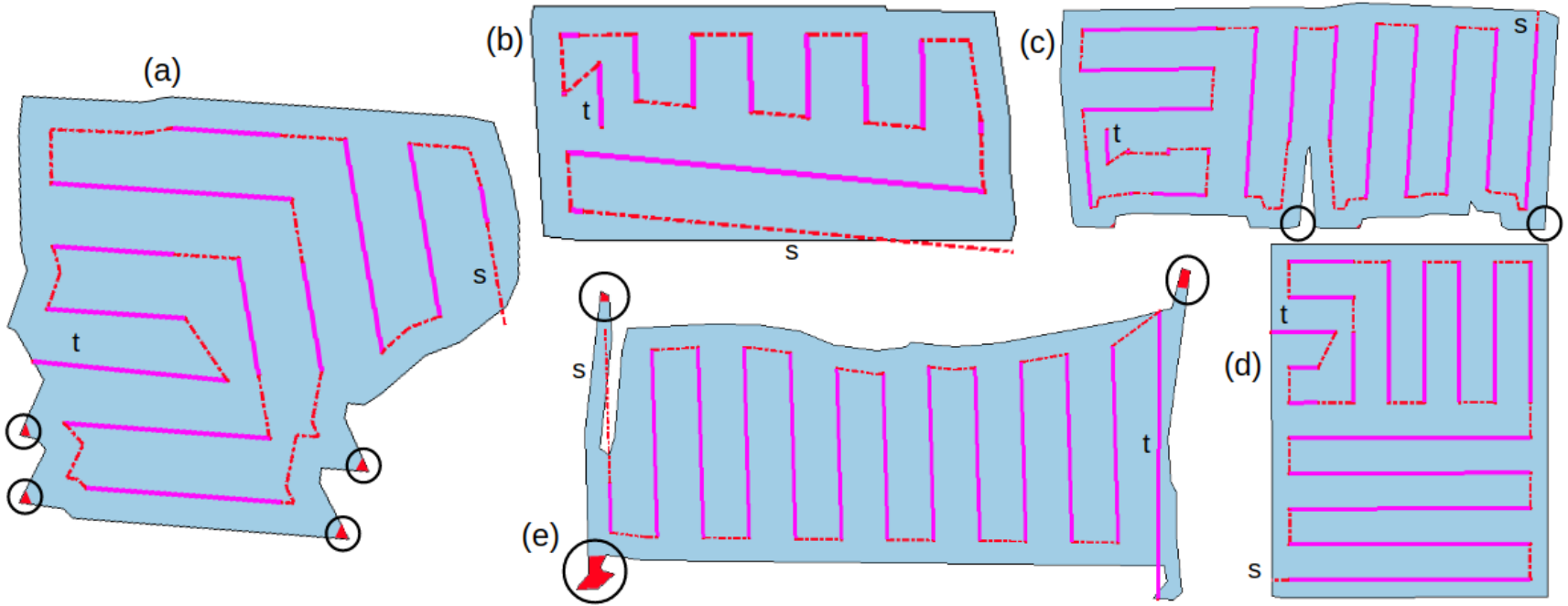
# Results: Test Cases



USC Viterbi School of Engineering

# Results (contd.)

- Spray Paths Generated on the polygon



USC Viterbi School of Engineering

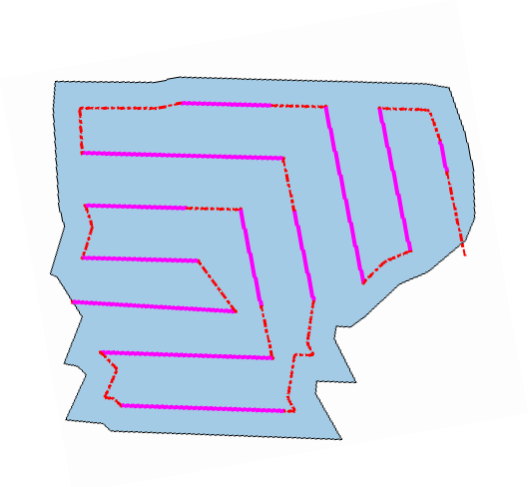
# Results (contd.)

- Spray path generation algorithm performance

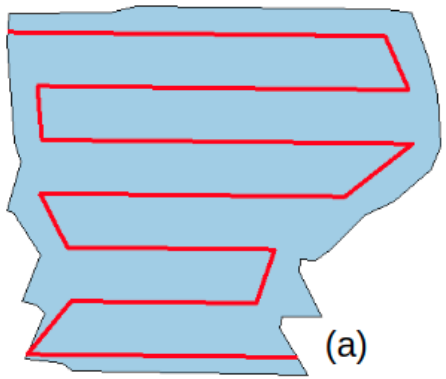
| Test Case | # Spray Polygon Edges | # Polygon Segments Expanded | # Branches Pruned | # Spray Paths Generated | # Grid Points Not Sprayed on | % Area NOT Sprayed | Spray Path Length (m) | Computation time (s) for generating first spray path | Computation Time (s) for last spray path |
|---|---|---|---|---|---|---|---|---|---|
| a | 31 | 688 | 266 | 8 | 67 | 0.59% | 7.66 | 0.05 | 7.6 |
| b | 11 | 153 | 10 | 5 | 0 | 0% | 5.95 | 0.02 | 1.0 |
| c | 32 | 70 | 53 | 5 | 4 | 0.03% | 11.21 | 0.14 | 0.98 |
| d | 5 | 255 | 0 | 5 | 0 | 0% | 13.65 | 0.06 | 1.4 |
| e | 40 | 463 | 257 | 1 | 115 | 0.71% | 10.79 | 0.18 | 5.8 |

# Results (contd.)

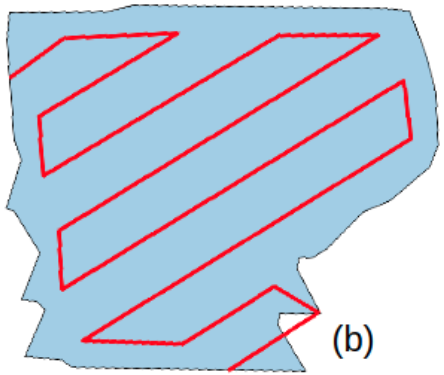
- Benchmarking for test case a: (a) The longest convex edge zig-zag path (b) principle component zig-zag Path and (c) Spiral path



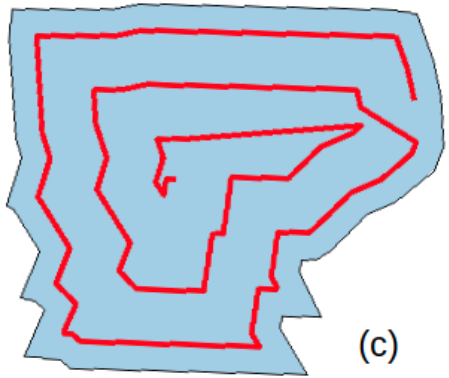
Computed using Spray Path Generation Algorithm




(a) The longest convex edge zig-zag path




(b) Principle component zig-zag path




(c) Spiral path

USC Viterbi School of Engineering

# Results (contd.)

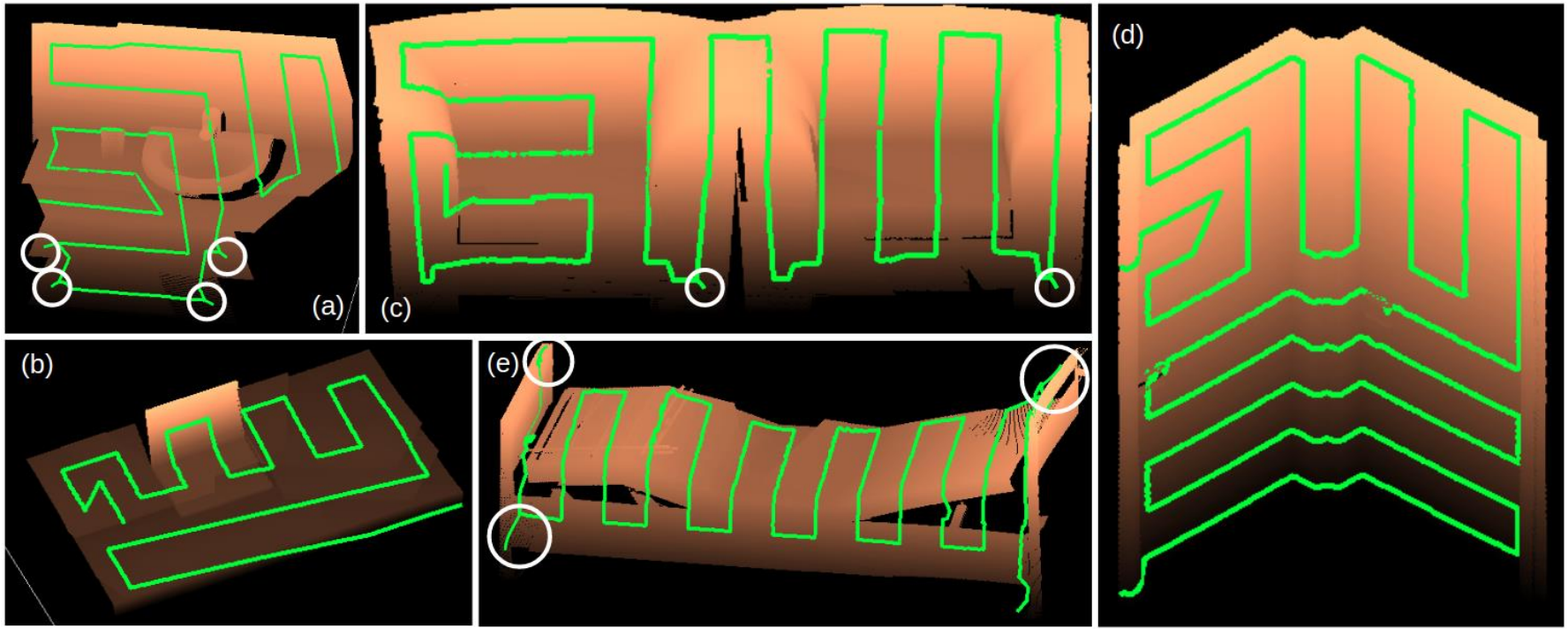
- Spray path generation algorithm performance

| Test Case | Spray Path Generation Algorithm | | | Longest Convex Edge Zig-Zag Path | | | Principle Component Zig-Zag Path | | | Spiral Path | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Path Length (m) | % Area NOT Sprayed | % of Spray Wasted | Path Length (m) | % Area NOT Sprayed | % of Spray Wasted | Path Length (m) | % Area NOT Sprayed | % of Spray Wasted | Path Length (m) | % Area NOT Sprayed | % of Spray Wasted |
| a | 7.66 | 0.59% | 2.11% | 7.37 | 4.22% | 6.16% | 7.25 | 7.93% | 4.55% | 7.32 | 4.67% | 1.71% |
| b | 5.95 | 0% | 1.23% | 5.13 | 3.90% | 1.71% | 5.10 | 3.97% | 1.88% | 5.85 | 0.62% | 1.40% |
| c | 11.21 | 0.03% | 1.11% | 10.41 | 9.46% | 3.04% | 11.66 | 7.34% | 9.69% | 13.13 | 9.56% | 5.28% |
| d | 13.65 | 0% | 0.96% | 12.71 | 3.10% | 2.71% | 12.51 | 2.98% | 2.60% | 13.45 | 3.72% | 1.21% |
| e | 10.79 | 0.71% | 2.15 | 10.50 | 8.72% | 5.27% | 10.93 | 8.82% | 5.61% | 10.65 | 6.32% | 1.94% |

USC Viterbi School of Engineering

# Results (contd.)

- Spray Paths Generated on the point clouds



USC Viterbi School of Engineering

# Results (contd.)

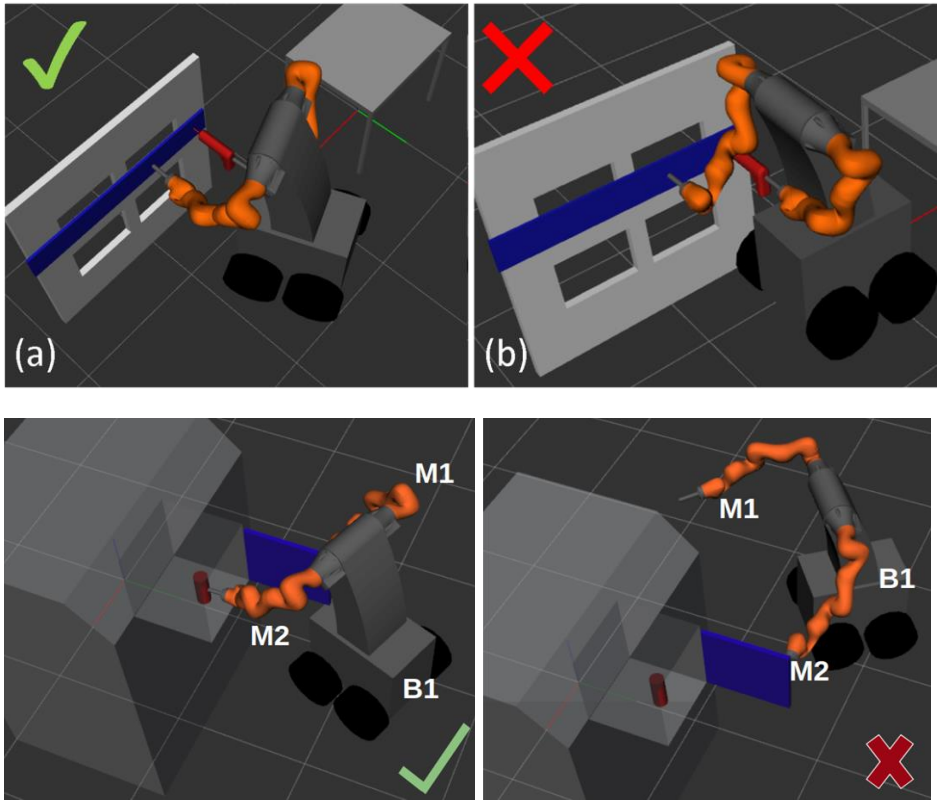
- Results for trajectory execution time before and after time interval determination for thorough disinfection

| Test Case | Trajectory Execution Time (s) Before Retiming | % Points not sprayed enough before retiming | Trajectory Execution Time (s) After Retiming |
|---|---|---|---|
| a | 12.5 | 23.8% | 40.2 |
| b | 14.5 | 68.1% | 38.5 |
| c | 41.4 | 36.7% | 71.2 |
| d | 36.2 | 13.3% | 93.4 |
| e | 24.4 | 24.7% | 48.8 |

# Task Assignment and Motion Planning for Bimanual Mobile Manipulation

# Background

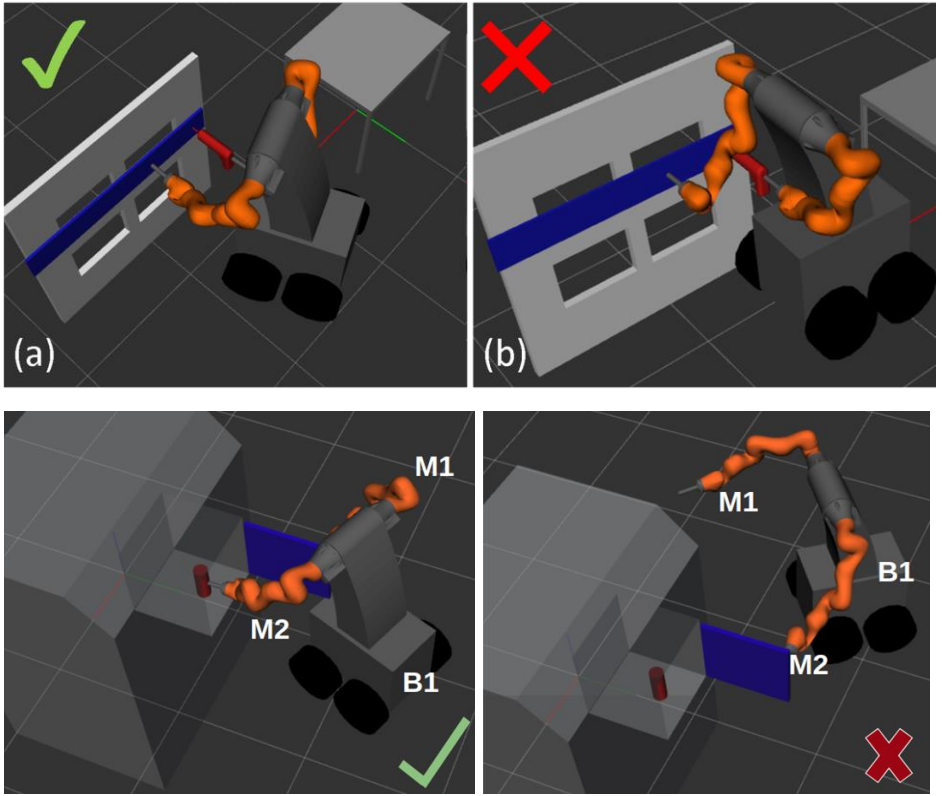
- Existing approaches on Task and Motion planning
  - designed for task assignment do not consider motion planning
  - designed for task and motion planning
    - cannot eliminate infeasible task-agent assignment without invoking motion planners
    - are not suitable for operations where agents need to collaborate and coordinate for complex tasks



USC Viterbi School of Engineering

# Objectives

- For complex tasks, have task-agent assignment for a bimanual mobile manipulator such that the query to the motion planners is done only when we're sure that the task assignment has no conflicts and is physically possible
- In the case where motion planner fails, re-plan only the failed part of the motion

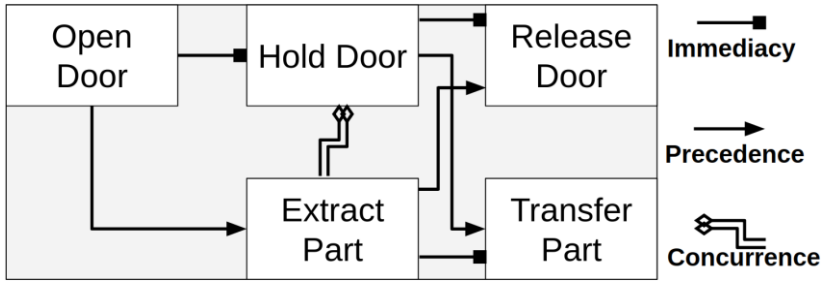


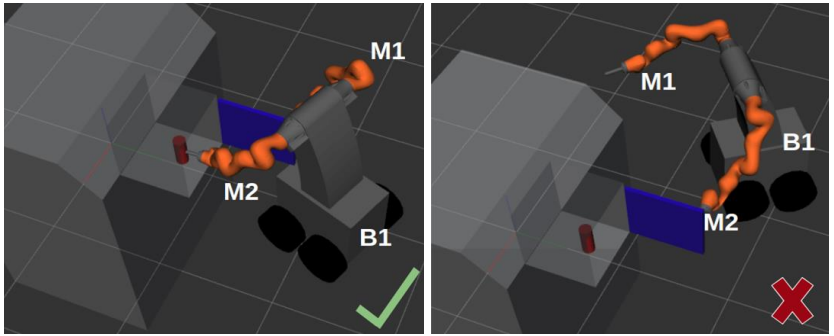
# Problem Statement

- Given a task network (T) for a Bimanual mobile manipulator, our objective is to
  - assign tasks to the agents i.e left arm, right arm and the mobile base,
  - sequence the tasks, and
  - generate continuous configuration space trajectories for each agent

such that

  - there is no conflict in the assignment, and
  - the time span of the operation is minimized

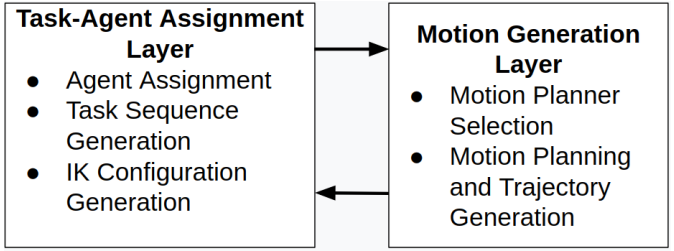


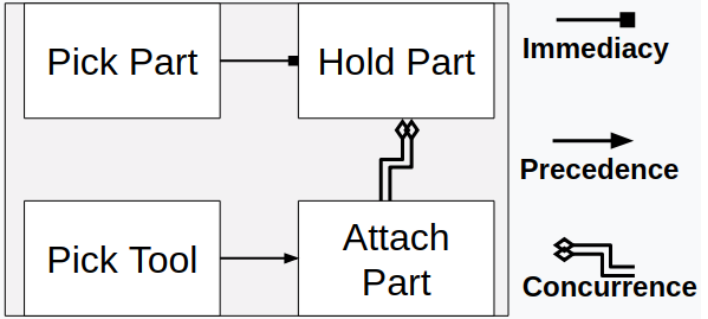



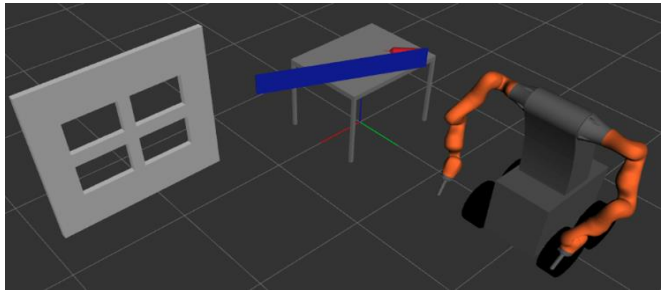
# Basic Idea

- A two-layer architecture for integrating motion planning and task-agent assignment

- Task-Agent Assignment Layer
  - Task Constraint Heuristic
    - based on *immediacy* & *concurrency*
  - Spatial Constraint Heuristic
    - based on *reachability* and existence of *collision free IK*



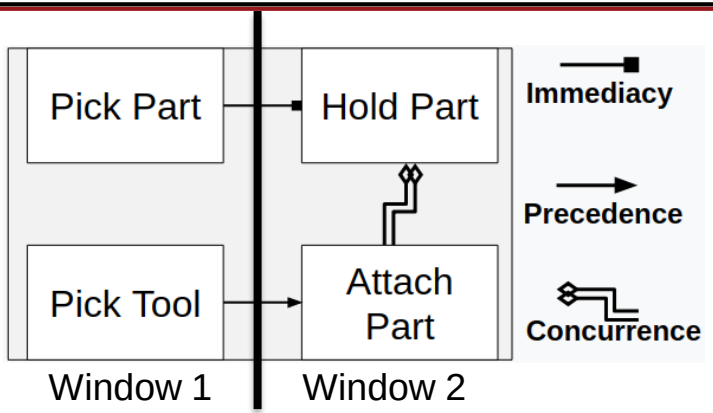
USC Viterbi School of Engineering

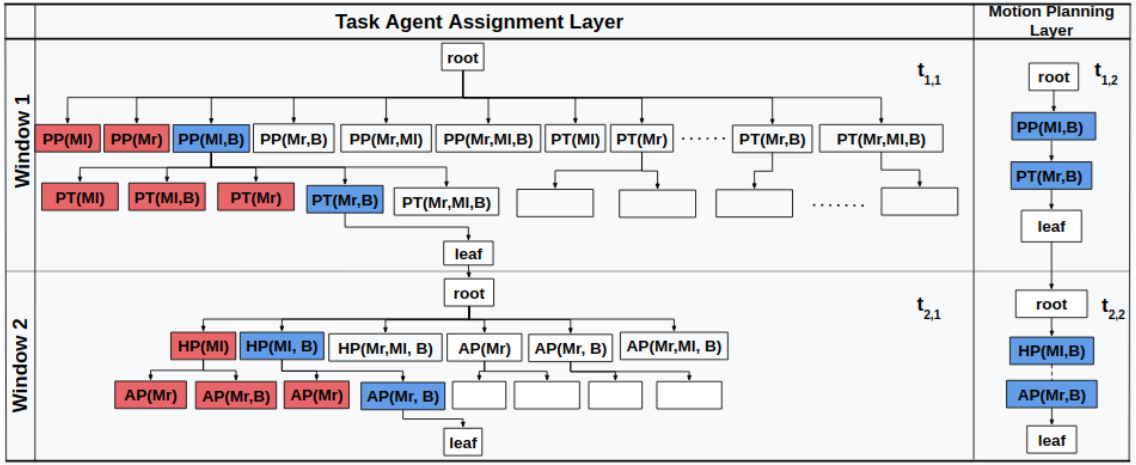
# Overview of Approach

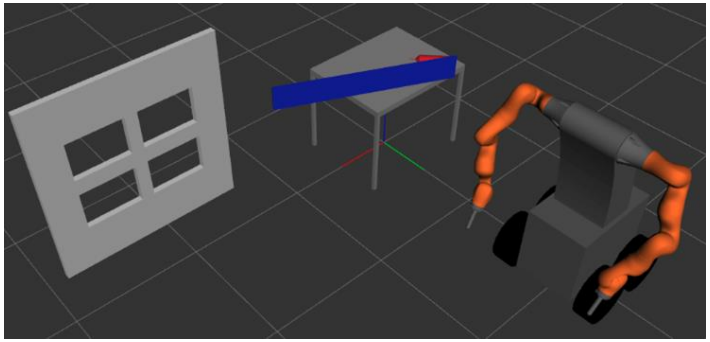
- A search based architecture for task-agent assignment
- Prune the task agent assignment nodes in the search tree based on the task constraints and spatial constraint
- Once the task-agent assignment is complete, move on the motion planning layer
- Cache the motions which succeed and only change those task-agent assignments which fail

# Task Assignment Trees

- Divide the task network into time windows
- Each task can be done by one of the six set of agents

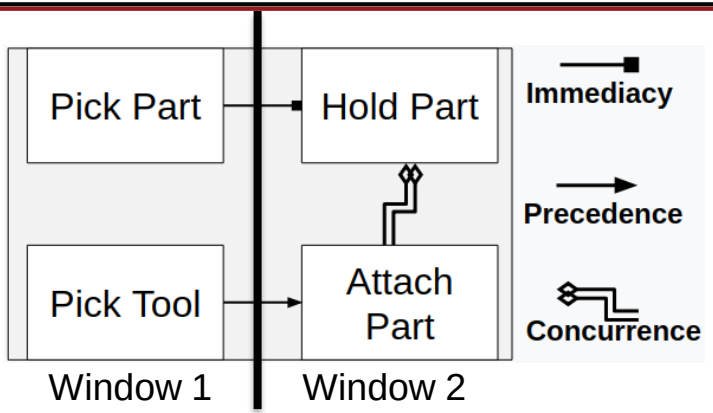


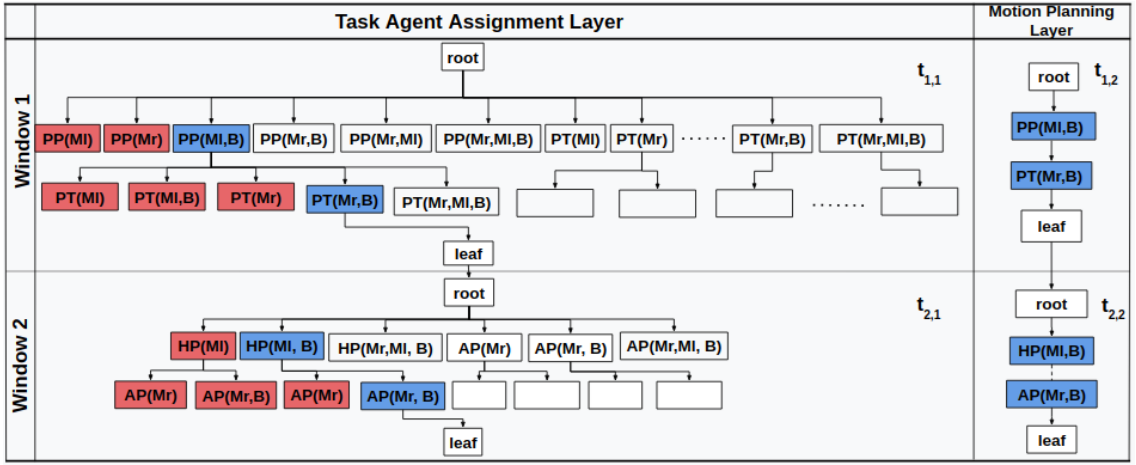



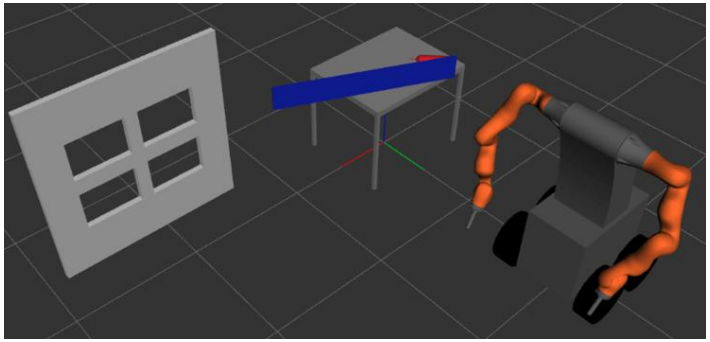



# Node Pruning and Caching

- Prune nodes based on task constraints heuristic, spatial constraints heuristic and previous assignment
- A caching scheme to move from the motion planning layer to task agent assignment layer



USC Viterbi School of Engineering

# Results

- Spatial constraint heuristics and caching scheme help in reducing computation time by 86% on average

| **Task No.** | **With** spatial constraint checking and caching (sec) | **Without** spatial constraint checking and caching (sec) |
|---|---|---|
| 1 | 32.3 | 479.7 |
| 2 | 15.3 | 257.5 |
| 3 | 71.9 | 921.3 |

USC Viterbi School of Engineering

# Publications

https://sites.usc.edu/skgupta/publications/

USC Viterbi School of Engineering

# Videos

https://www.youtube.com/channel/UCO82Tsg5Xc5vP_ZWkax4Wpg