16-782
Planning & Decision-making in Robotics

Case Study: Planning for Mobile Manipulation and Legged Robots

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Two Examples

- Planning for Mobile Manipulation
- Planning for Legged Robots

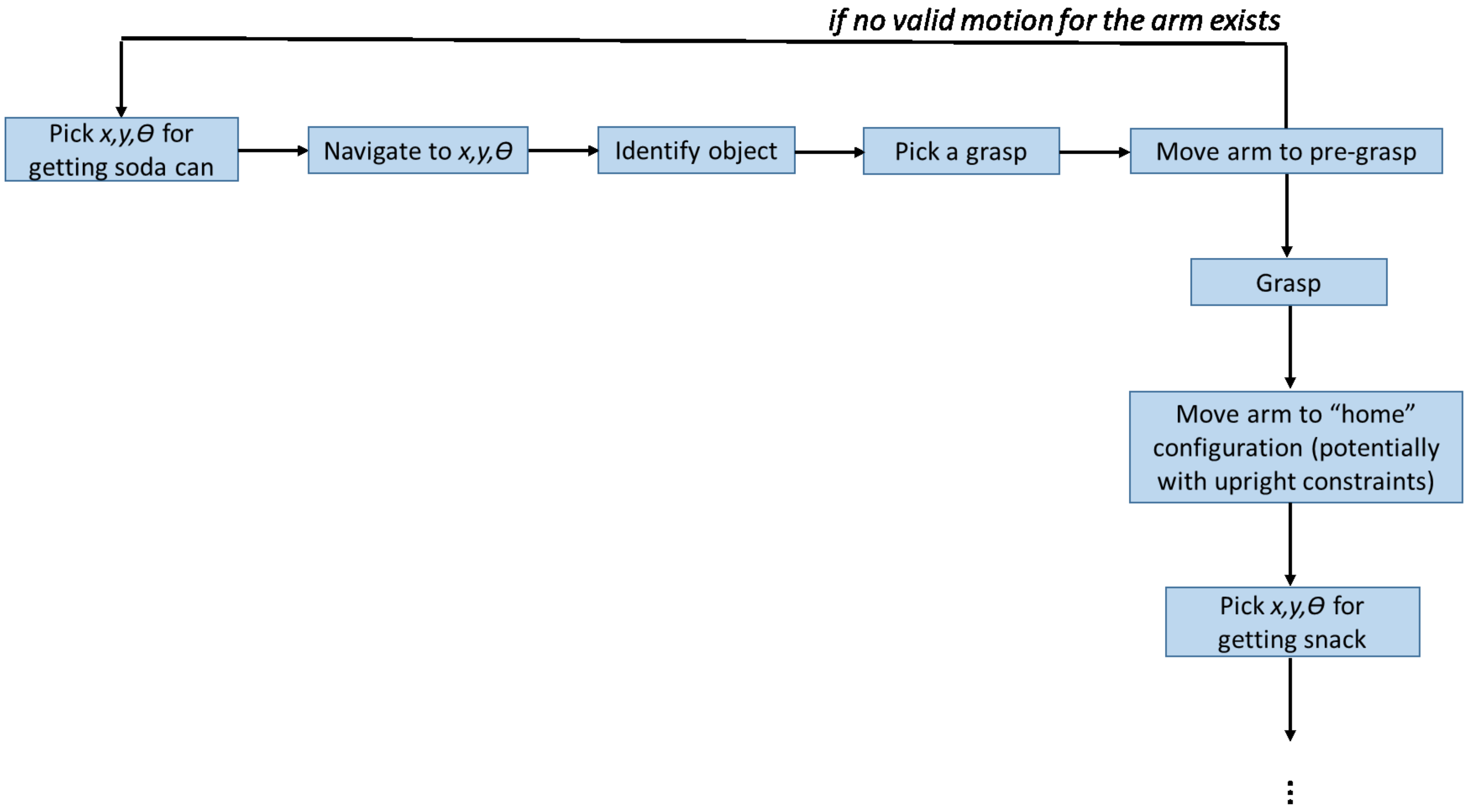
Two Examples

- Planning for Mobile Manipulation
- Planning for Legged Robots

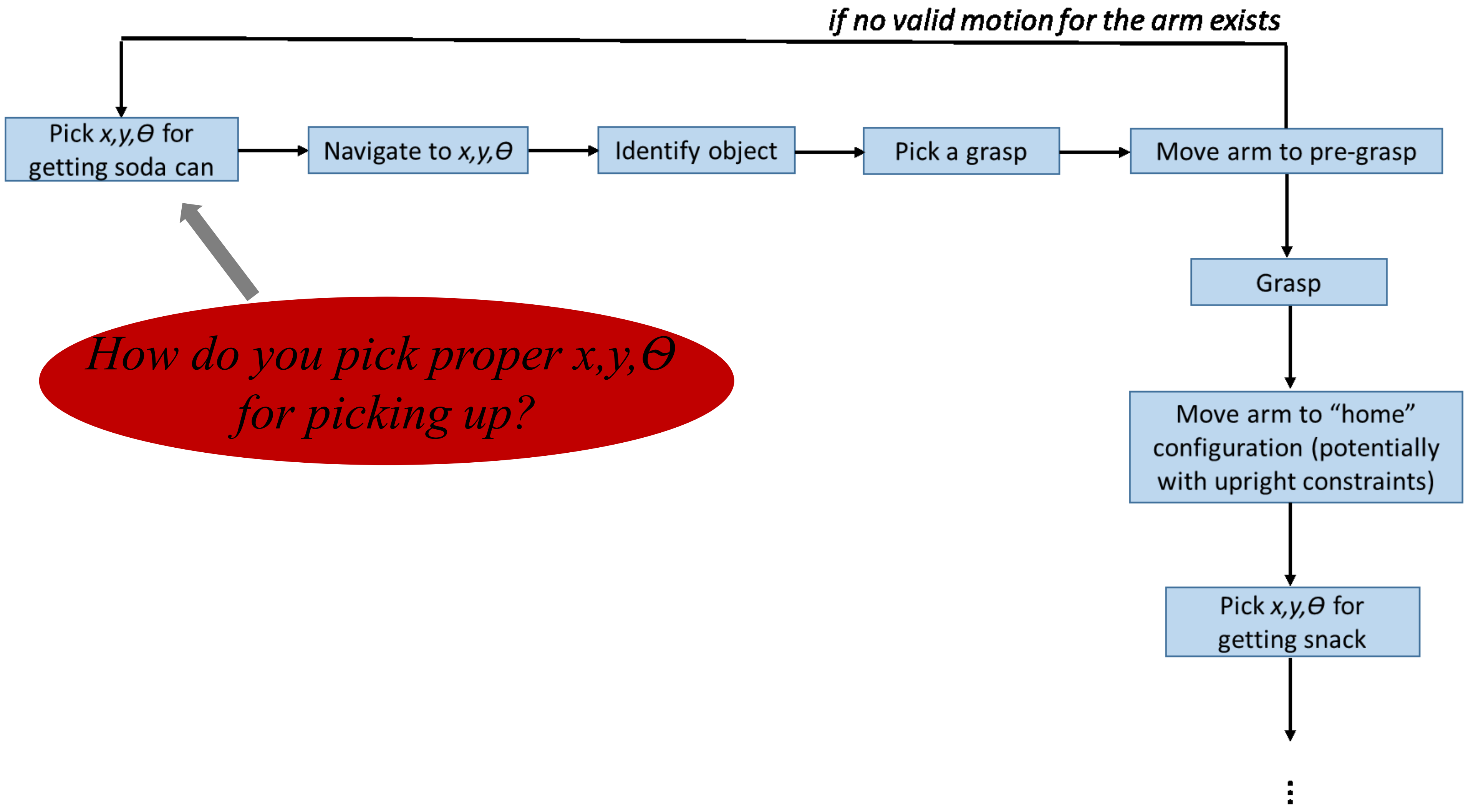
Robotic Bartender Demo ([Phillips et al.])

- Robot takes in a command from User Interface as to what soda can and snack to deliver

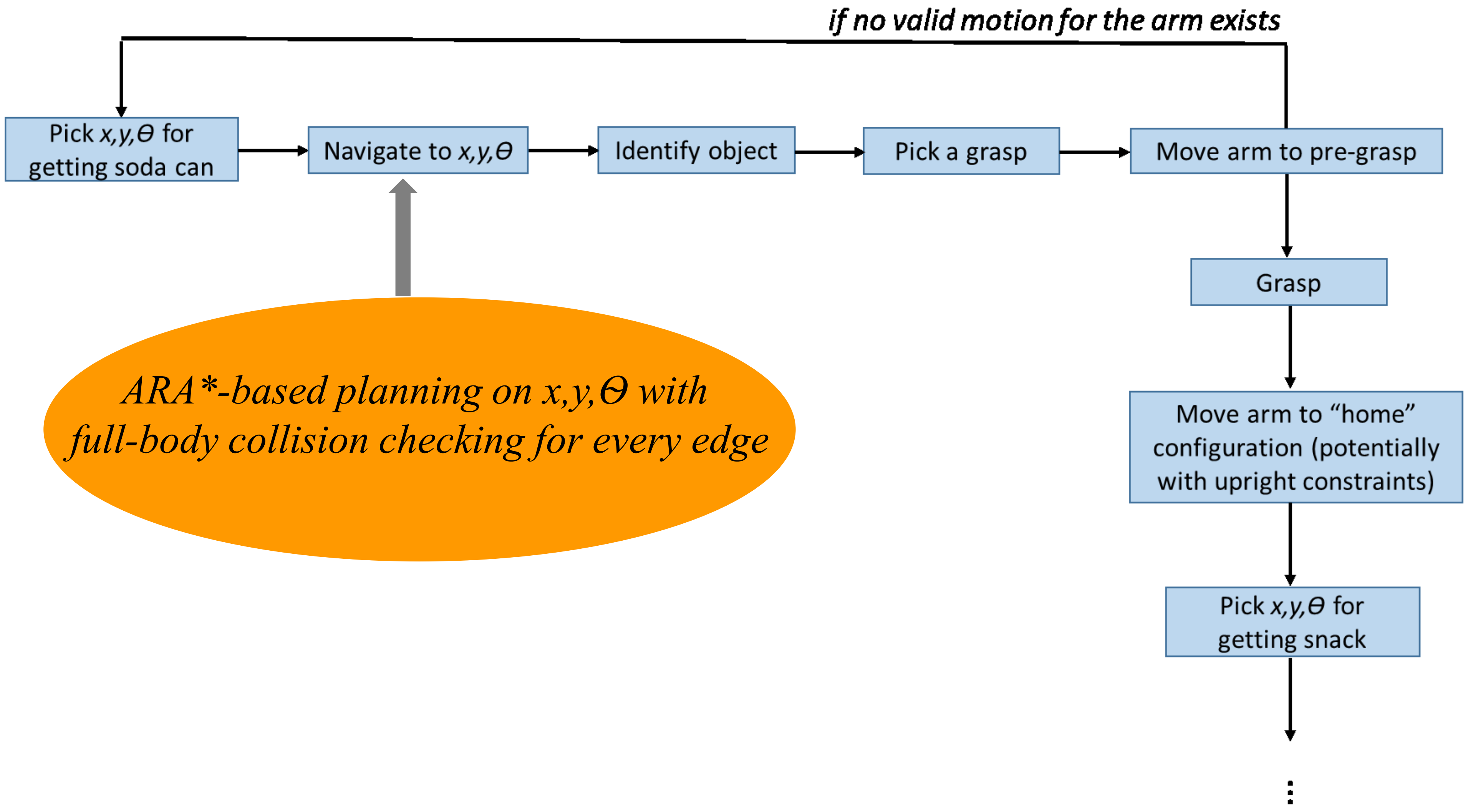
Typical Sequence of Operations (State Machine)


if no valid motion for the arm exists
Pick *x,y,ϴ* for getting soda can
Navigate to *x,y,ϴ*
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to “home” configuration (potentially with upright constraints)
Pick *x,y,ϴ* for getting snack
⋮

Typical Sequence of Operations (State Machine)

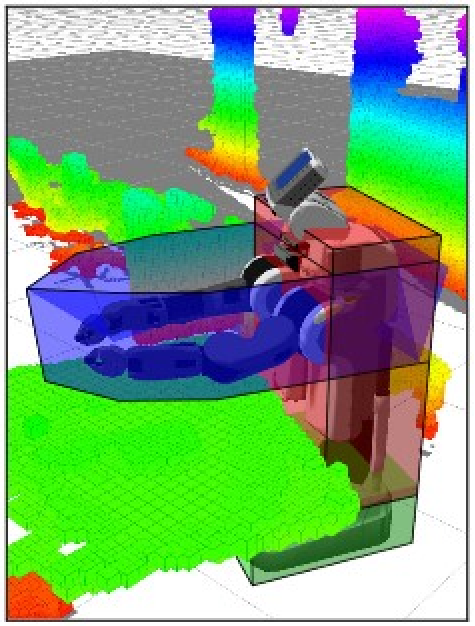

if no valid motion for the arm exists
Pick x,y,Θ for getting soda can
Navigate to x,y,Θ
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick x,y,Θ for getting snack
⋮
How do you pick proper x,y,Θ for picking up?

Typical Sequence of Operations (State Machine)


if no valid motion for the arm exists
Pick x,y,Θ for getting soda can
Navigate to x,y,Θ
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick x,y,Θ for getting snack
⋮
ARA*-based planning on x,y,Θ with full-body collision checking for every edge

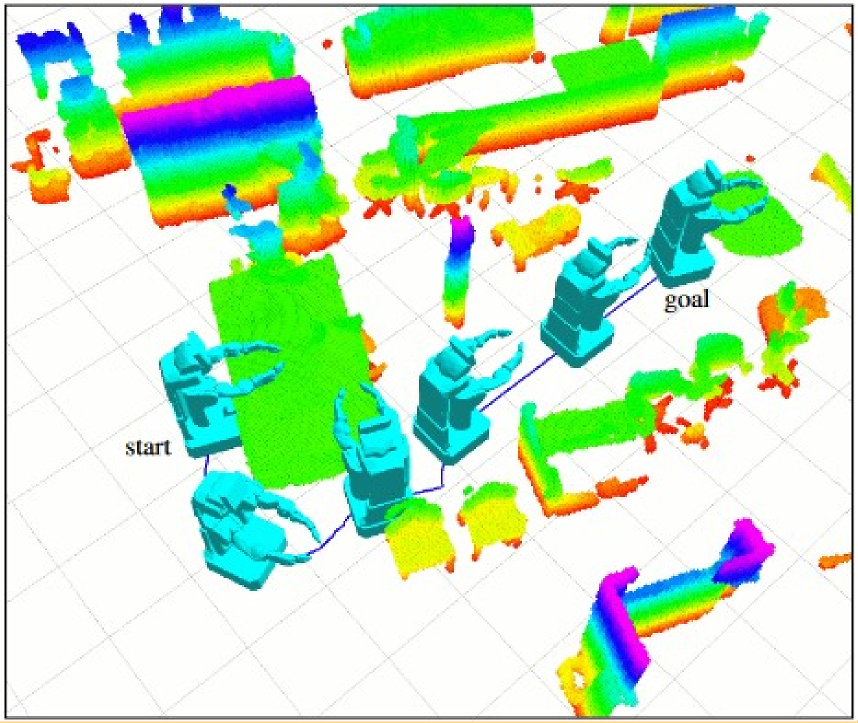
Graph for Navigation with Complex 3D Body [Hornung et al., '12]

- 3D (x,y,θ) lattice-based graph representation for full-body collision checking
  - takes set of motion primitives as input
  - takes N footprints of the robot defined as polygons as input
  - each footprint corresponds to the projection of a part of the body onto x,y plane
  - collision checking/cost computation is done for each footprint at the corresponding projection of the 3D map

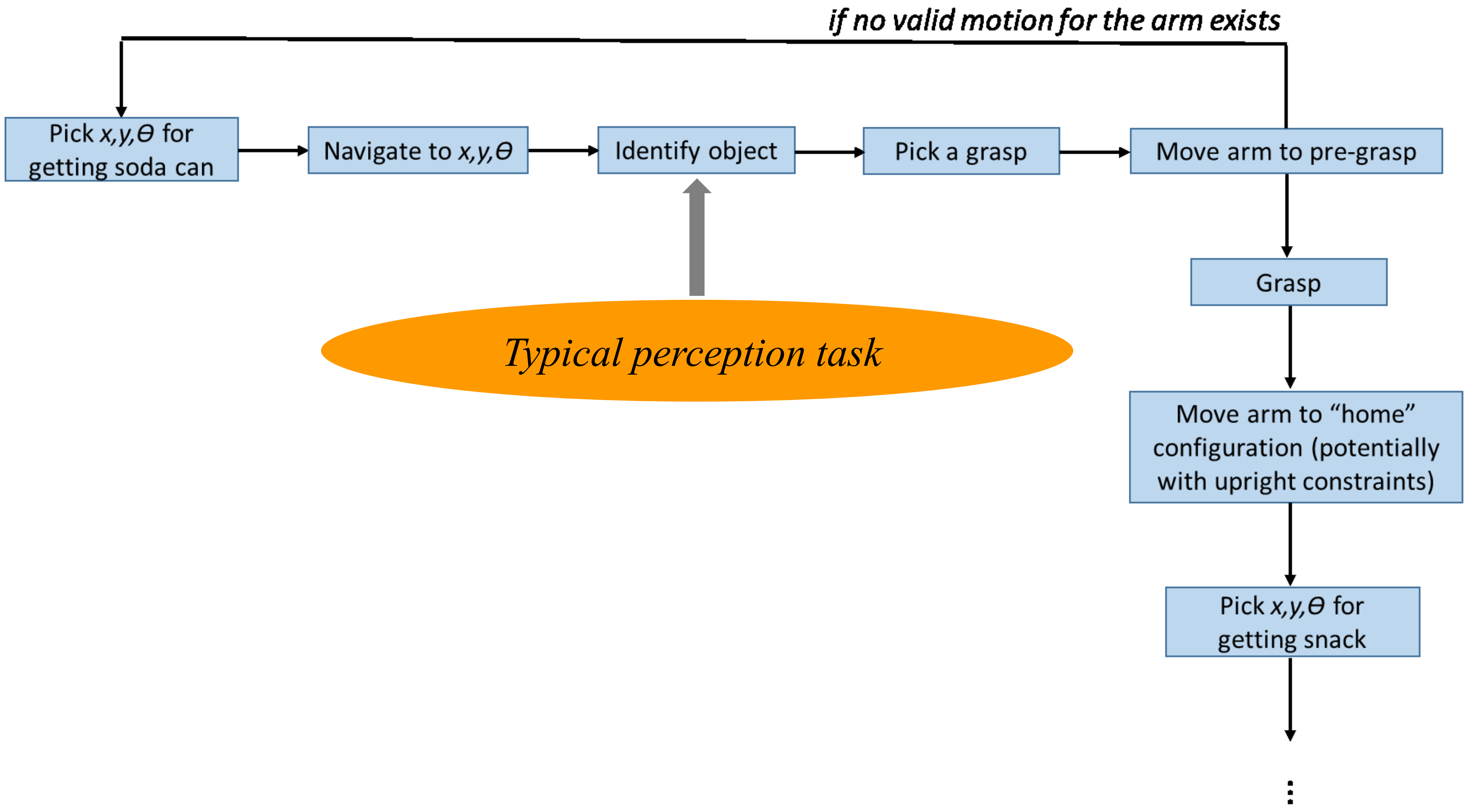


Graph for Navigation with Complex 3D Body [Hornung et al., '12]

- 3D (x,y,θ) lattice-based graph representation for full-body collision checking
  - takes set of motion primitives as input
  - takes N footprints of the robot defined as polygons as input
  - each footprint corresponds to the projection of a part of the body onto x,y plane
  - collision checking/cost computation is done for each footprint at the corresponding projection of the 3D map

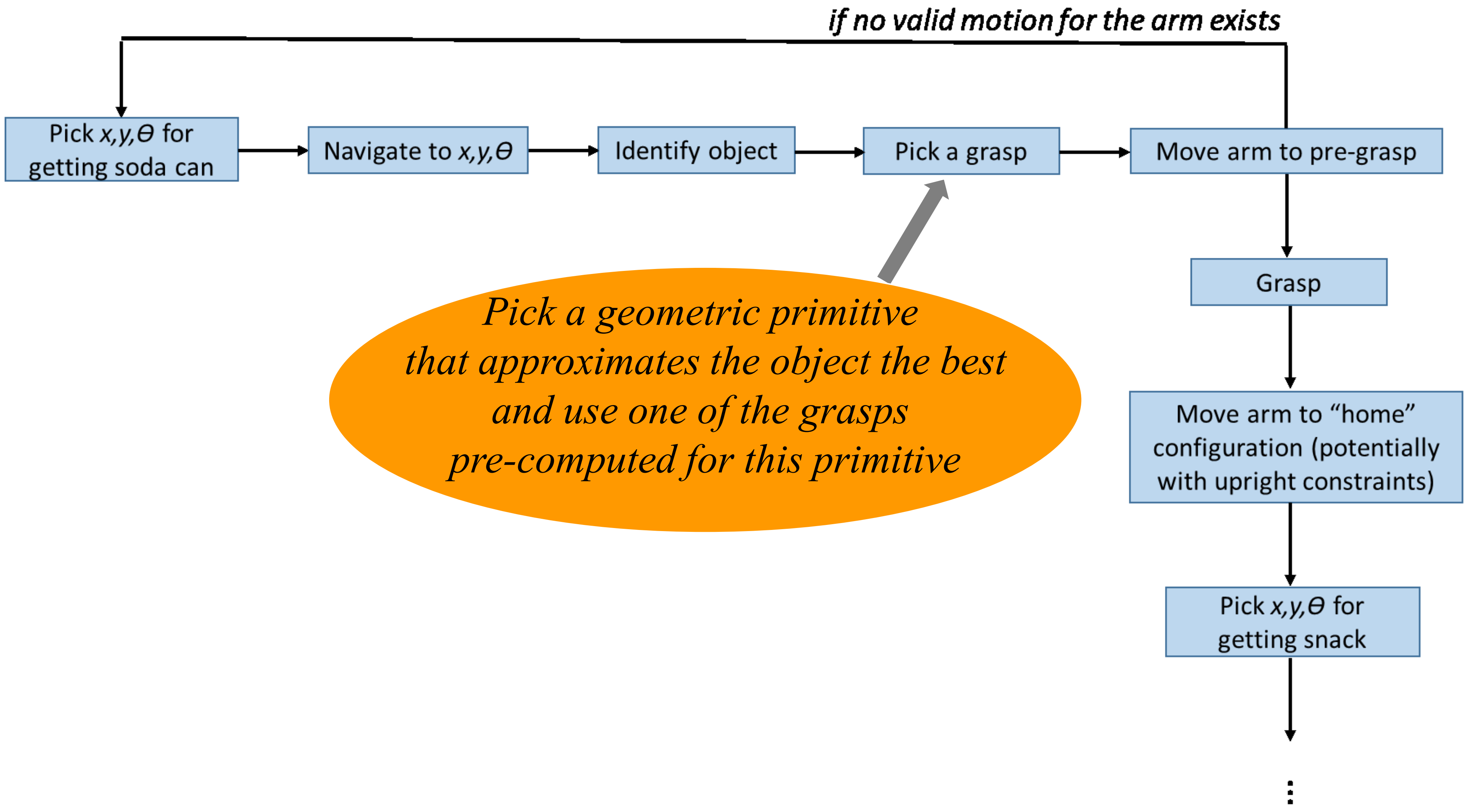


Typical Sequence of Operations (State Machine)



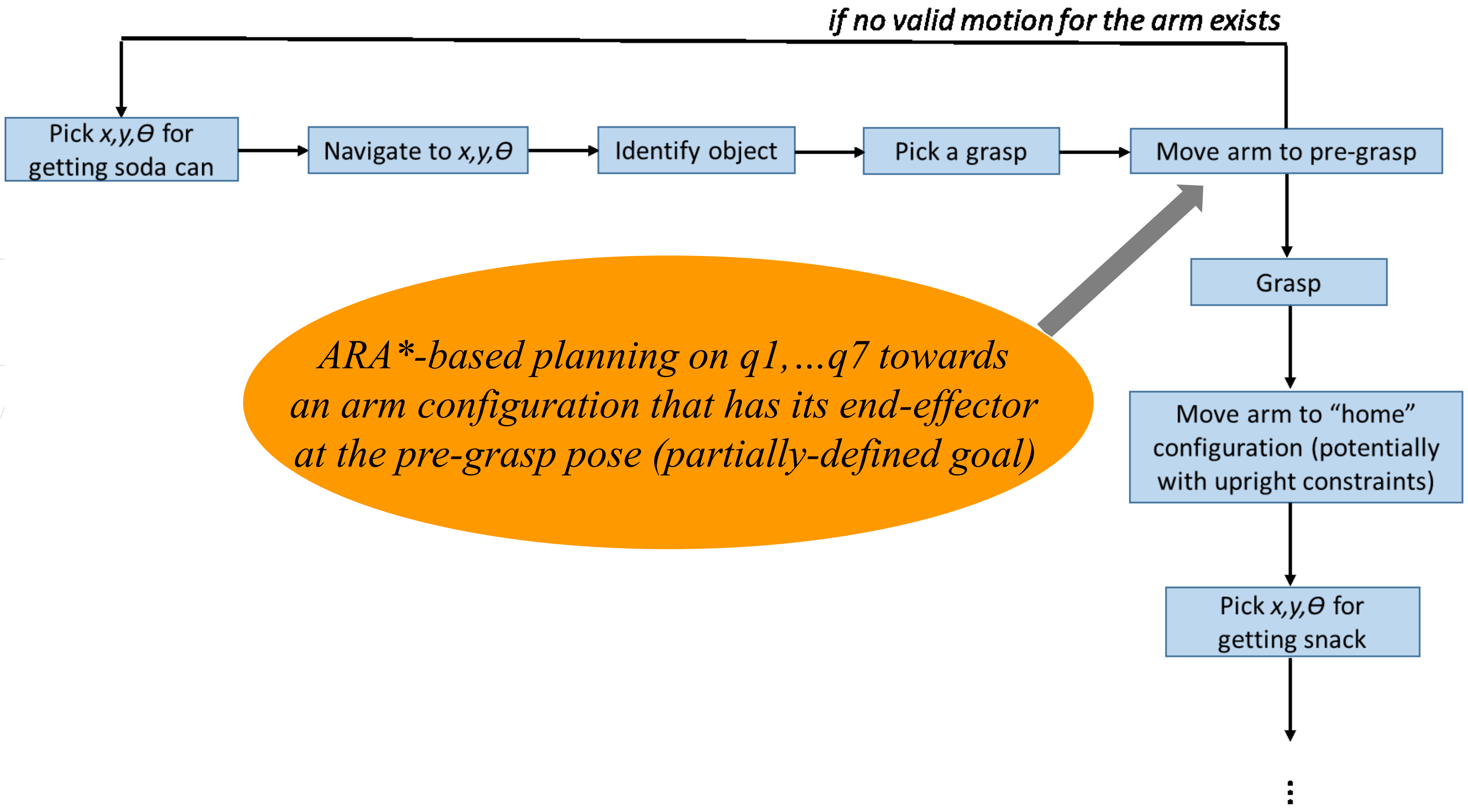
if no valid motion for the arm exists
Pick x,y,Θ for getting soda can
Navigate to x,y,Θ
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick x,y,Θ for getting snack
⋮
Typical perception task

Typical Sequence of Operations (State Machine)



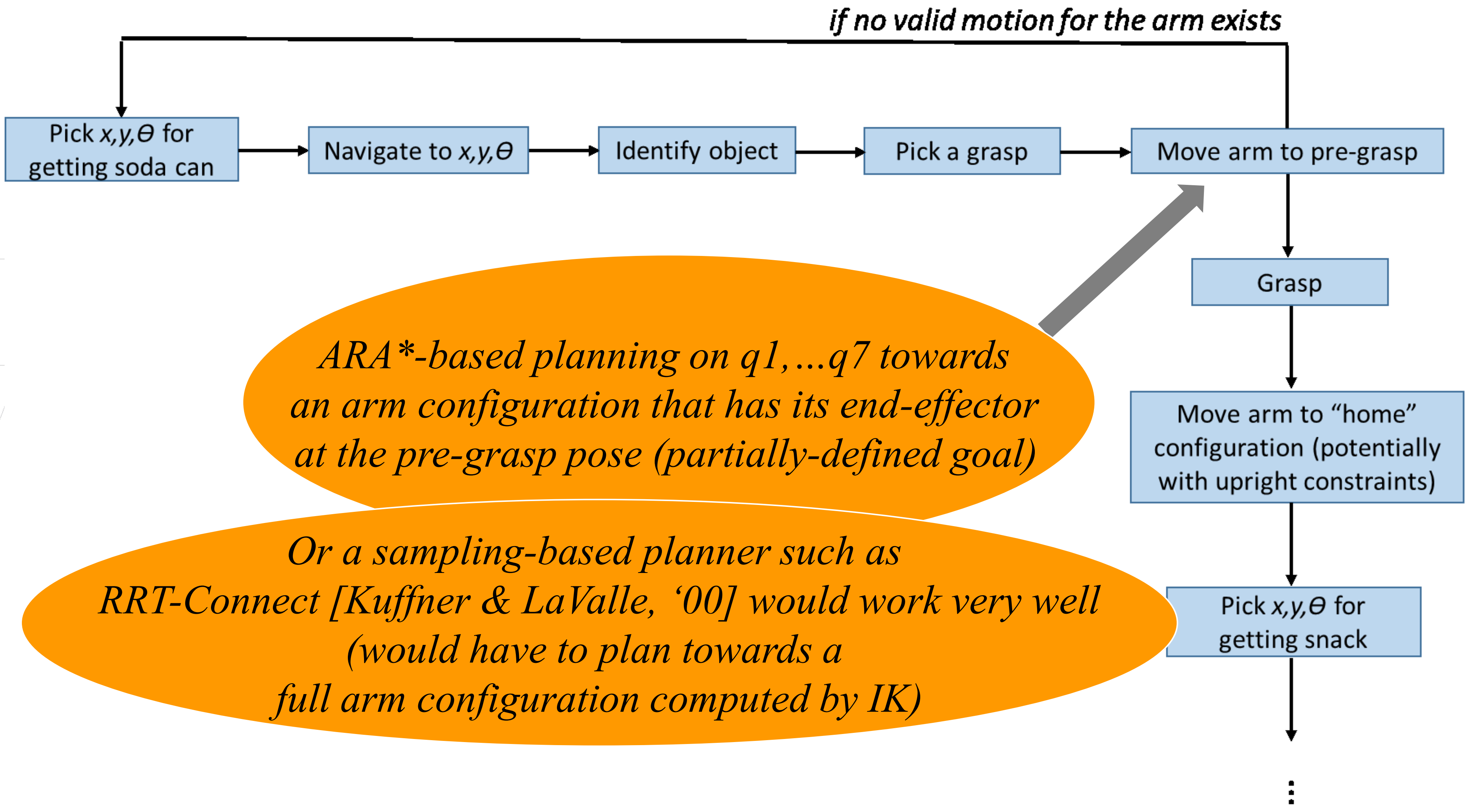
if no valid motion for the arm exists
Pick *x,y,Θ* for getting soda can
Navigate to *x,y,Θ*
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick *x,y,Θ* for getting snack
⋮
Pick a geometric primitive that approximates the object the best and use one of the grasps pre-computed for this primitive

Typical Sequence of Operations (State Machine)



if no valid motion for the arm exists
Pick x,y,Θ for getting soda can
Navigate to x,y,Θ
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick x,y,Θ for getting snack
⋮
ARA*-based planning on q1,…q7 towards an arm configuration that has its end-effector at the pre-grasp pose (partially-defined goal)

Typical Sequence of Operations (State Machine)

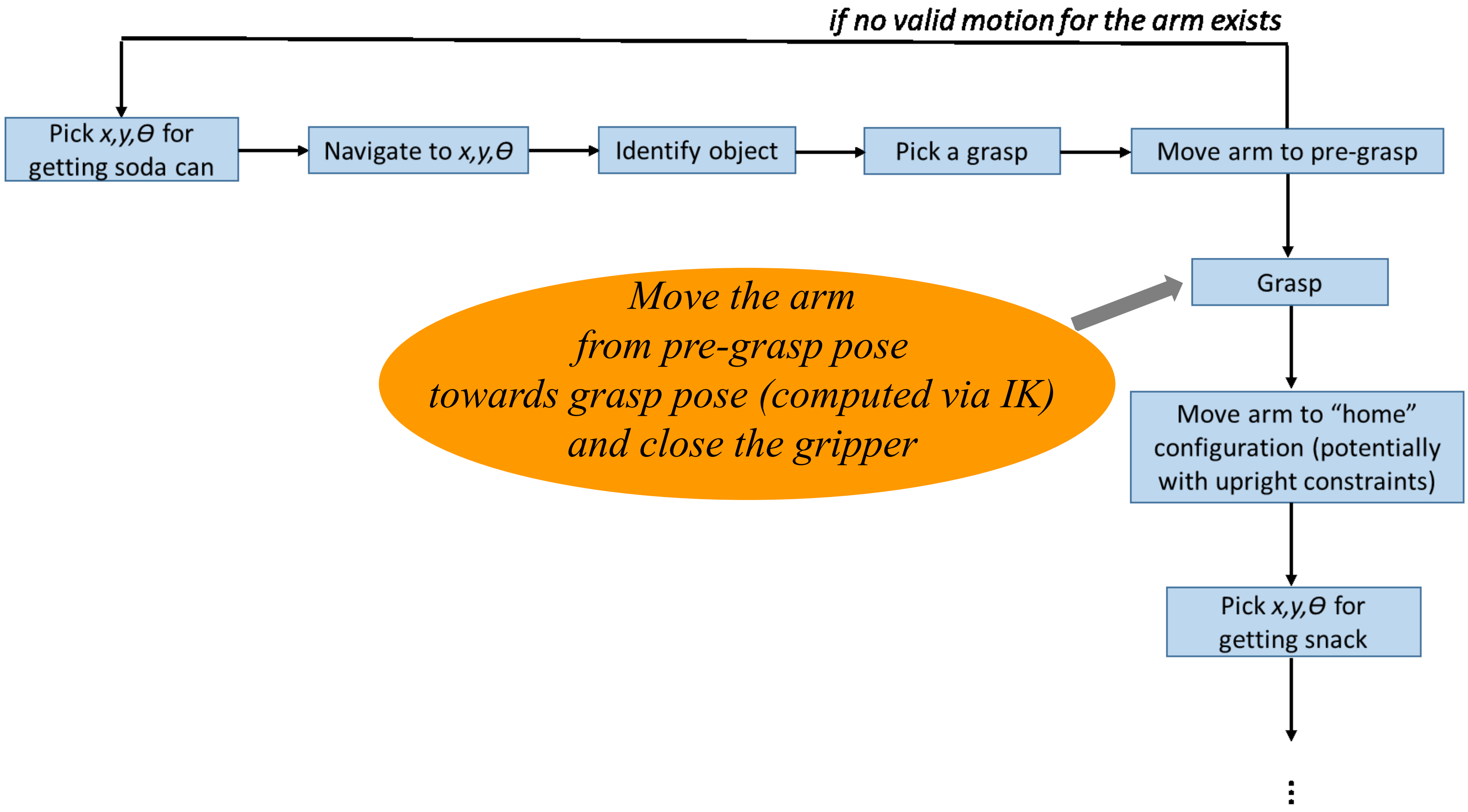


if no valid motion for the arm exists
Pick *x,y,Θ* for getting soda can
Navigate to *x,y,Θ*
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick *x,y,Θ* for getting snack
⋮

ARA-based planning on q1,…q7 towards an arm configuration that has its end-effector at the pre-grasp pose (partially-defined goal)*

Or a sampling-based planner such as RRT-Connect [Kuffner & LaValle, '00] would work very well (would have to plan towards a full arm configuration computed by IK)

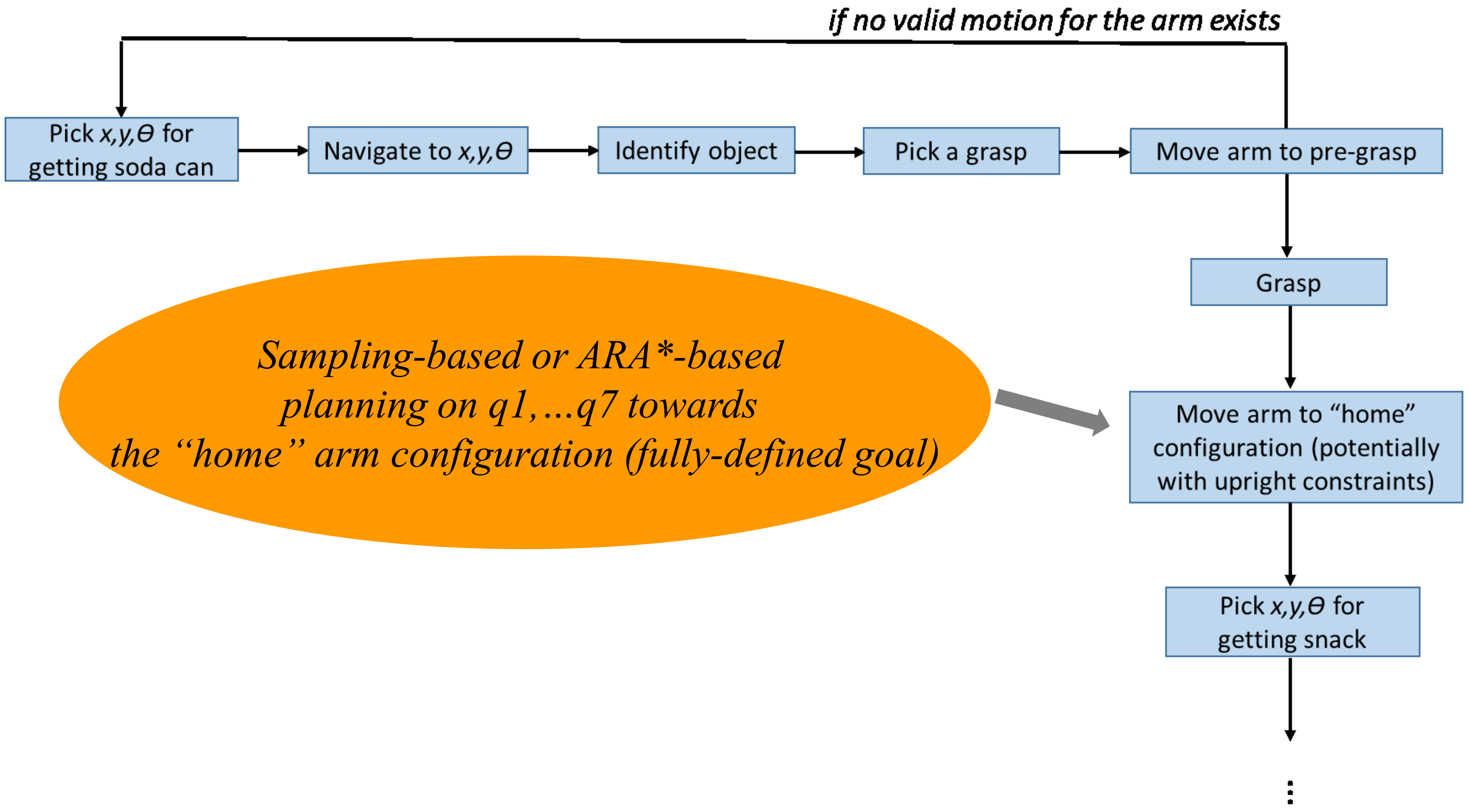
Typical Sequence of Operations (State Machine)



if no valid motion for the arm exists
Pick x,y,Θ for getting soda can
Navigate to x,y,Θ
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick x,y,Θ for getting snack
⋮

Move the arm from pre-grasp pose towards grasp pose (computed via IK) and close the gripper

Typical Sequence of Operations (State Machine)



if no valid motion for the arm exists
Pick *x,y,Θ* for getting soda can
Navigate to *x,y,Θ*
Identify object
Pick a grasp
Move arm to pre-grasp
Grasp
Move arm to "home" configuration (potentially with upright constraints)
Pick *x,y,Θ* for getting snack
⋮

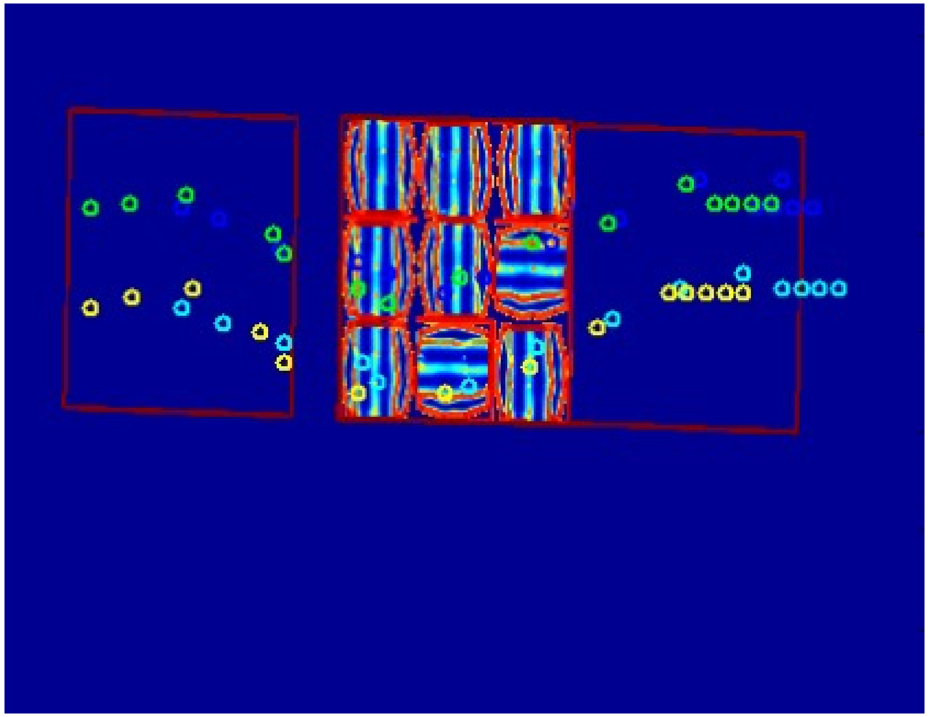
Sampling-based or ARA-based planning on q1,…q7 towards the "home" arm configuration (fully-defined goal)*

Two Examples

- Planning for Mobile Manipulation
- Planning for Legged Robots

Little Dog Demo [Vernaza et al., '09]

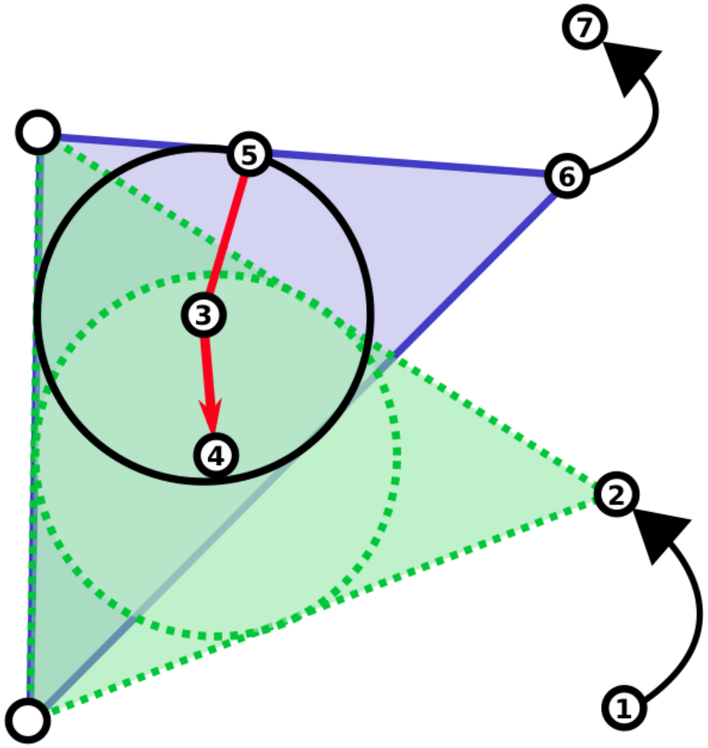
- Little Dog robot needs to traverse a fully-known terrain
- Planning
  - Plans footsteps first with an anytime variant of A*
  - Compute COM of the robot afterwards to support execution



Footstep Planner [Vernaza et al., ‘09]

Assumptions of the planner:

- Only one leg lifted at a time to ensure static stability
- Center of mass shifts during quad-support phase to prevent tipping
- Footholds chosen deliberately to maximize stability



Footstep Planner [Vernaza et al., ‘09]

Planner builds Graph:

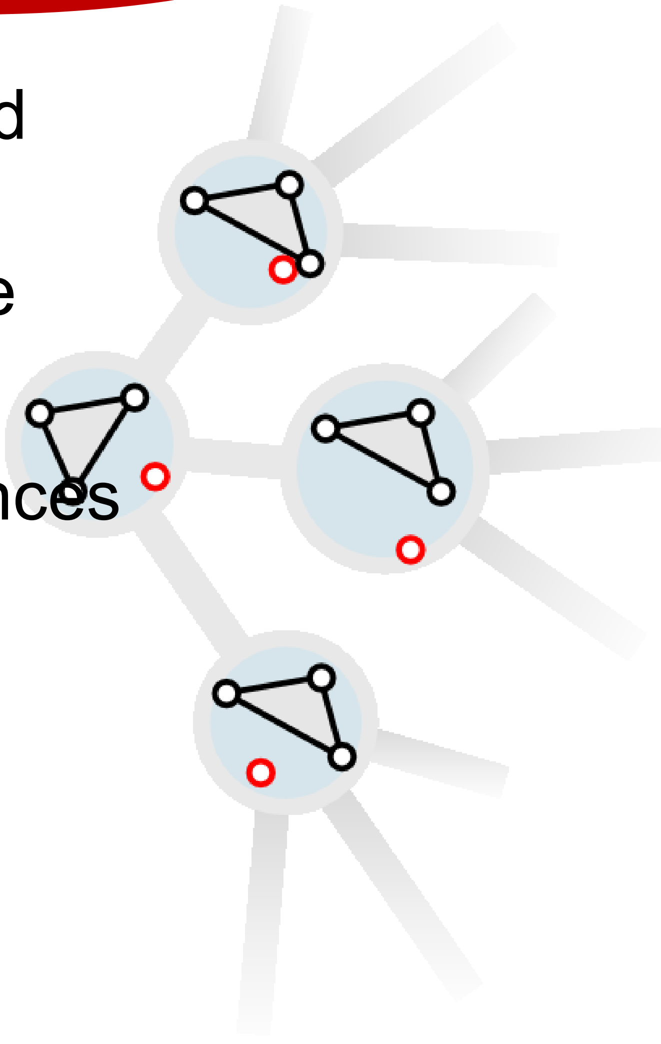
What are states?
What are edges?

Footstep Planner [Vernaza et al., ‘09]

Planner builds Graph:

Implicit or explicit graph?

- **State (stance):** 9-dimensional foothold configuration
  - feet positions and current gait phase
- **Edge**: feasible transition between stances
- **Edge costs** for transitions computed based on risk, anticipated delay



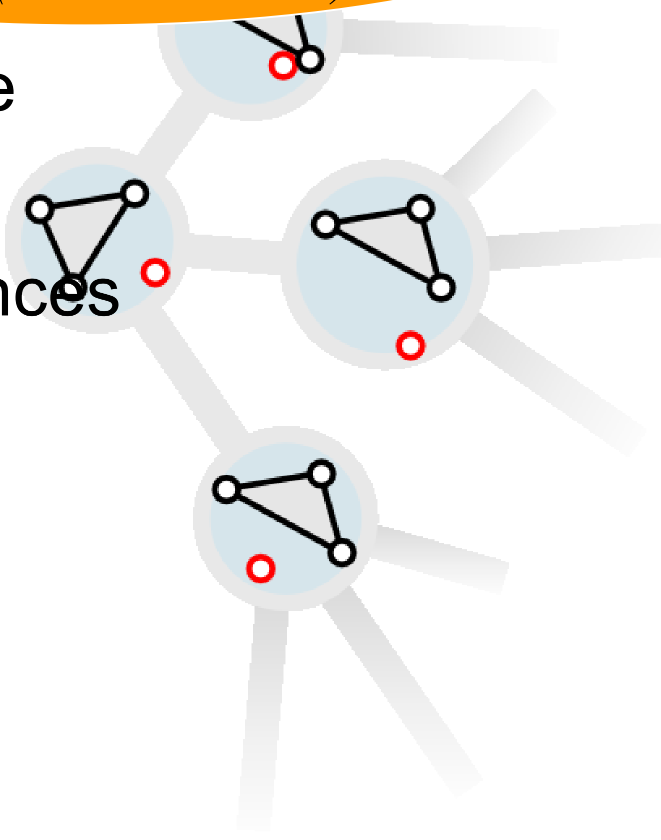
Footstep Planner [Vernaza et al., '09]

Planner builds (<u>implicit</u>) Graph:

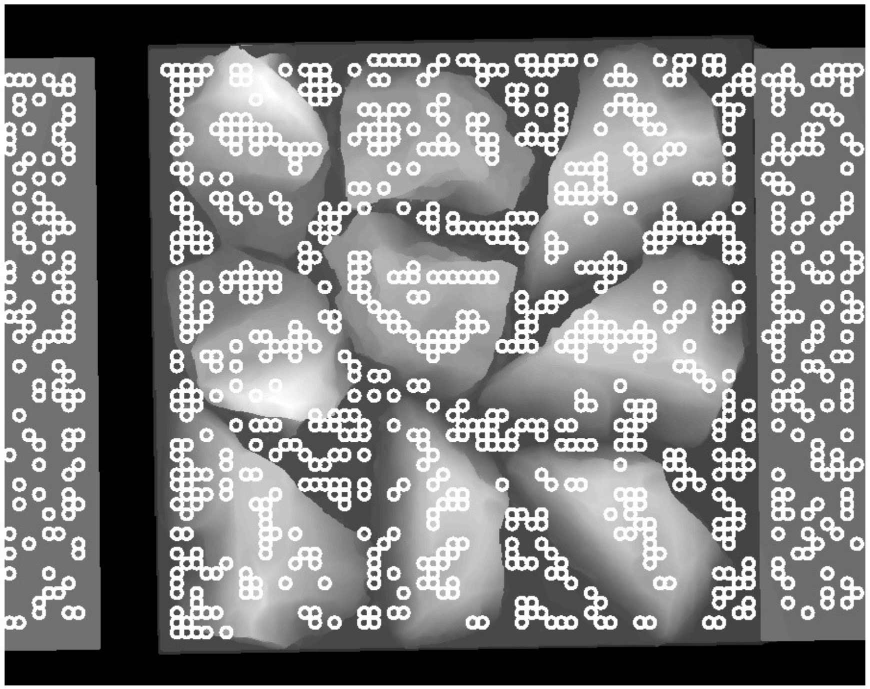
Requires definition of:

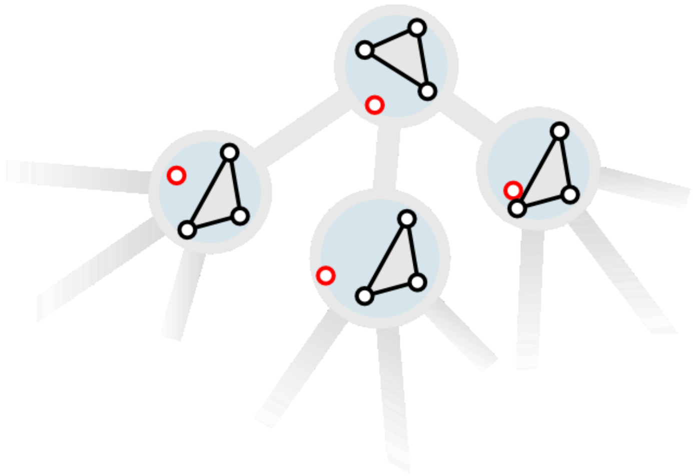
GetSuccessors(state S)
GetCost(state S, state S')

- **State (stance):** 9-dime configuration
  - feet positions and current gait phase
- **Edge**: feasible transition between stances
- **Edge costs** for transitions computed based on risk, anticipated delay



Implementation of *GetSuccessors(s)* Function





- Finite set of good quality candidate footholds selected prior to planning
- Valid stances are kinematically feasible 4-tuples of candidate footholds
- Successors of a given stance computed by:
  - determining reachable candidate footholds that result in a valid stance

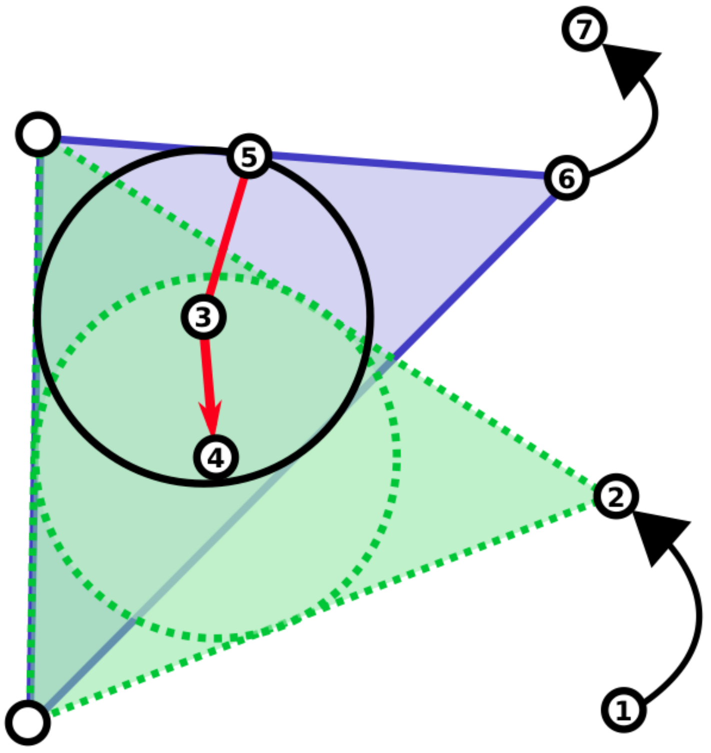
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:

Any thoughts?

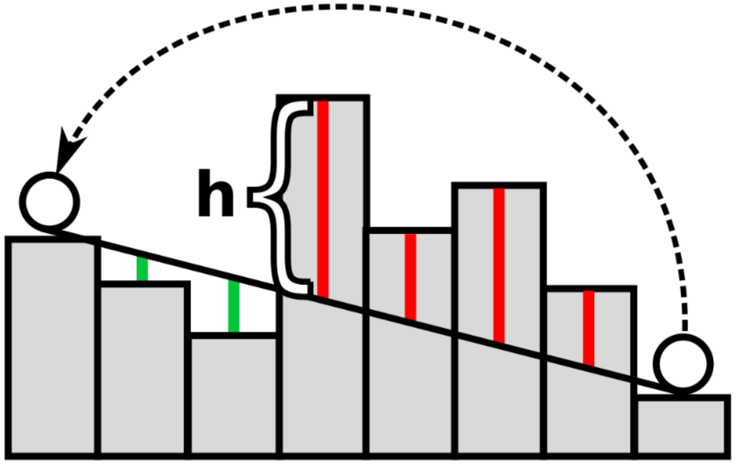
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Fixed cost per step**
    - Minimizes # of steps in the plan
  - **Center of mass travel**
    - Discourages unnecessary motion of COM
  - **Incircle radius**
    - Discourages stances with small incircle radii (distance from point 3 to point 5 in the picture)



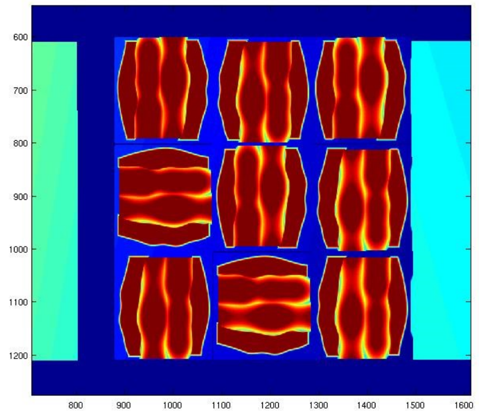
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Collision**
    - Risk of body/foot colliding with terrain
  - **Foot height variance**
    - Encourages robot to stay level



Implementation of *GetCost(s,s')* Function

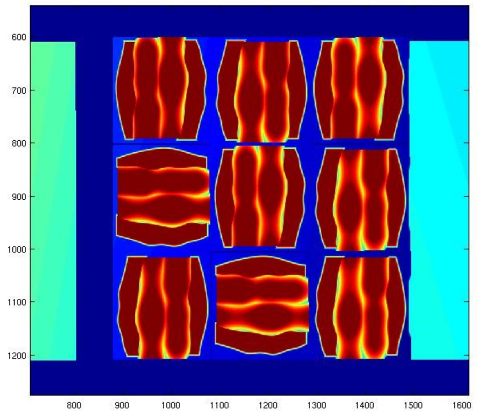
- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to reach next foothold, switch to next support triangle without dragging feet
  - **Terrain slope**
    - Ensures terrain slope supports direction of motion
  - **Terrain cost**
    - Considers slippage potential given terrain



Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to r
      support triangle without
  - **Terrain slope**
    - Ensures terrain slope supports direction of motion
  - **Terrain cost**
    - Considers slippage potential given terrain

Lots of features make up the cost function.
Fine tuning them is not fun ☹

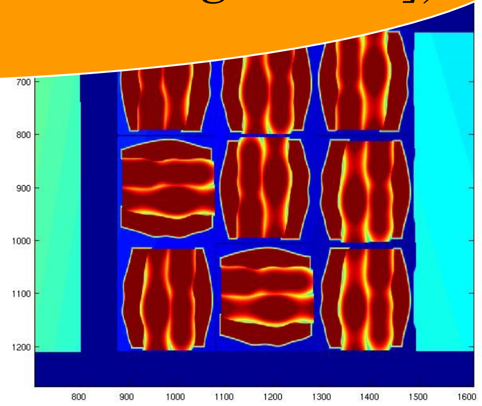


Implementation of *GetCost(s,s')* Function

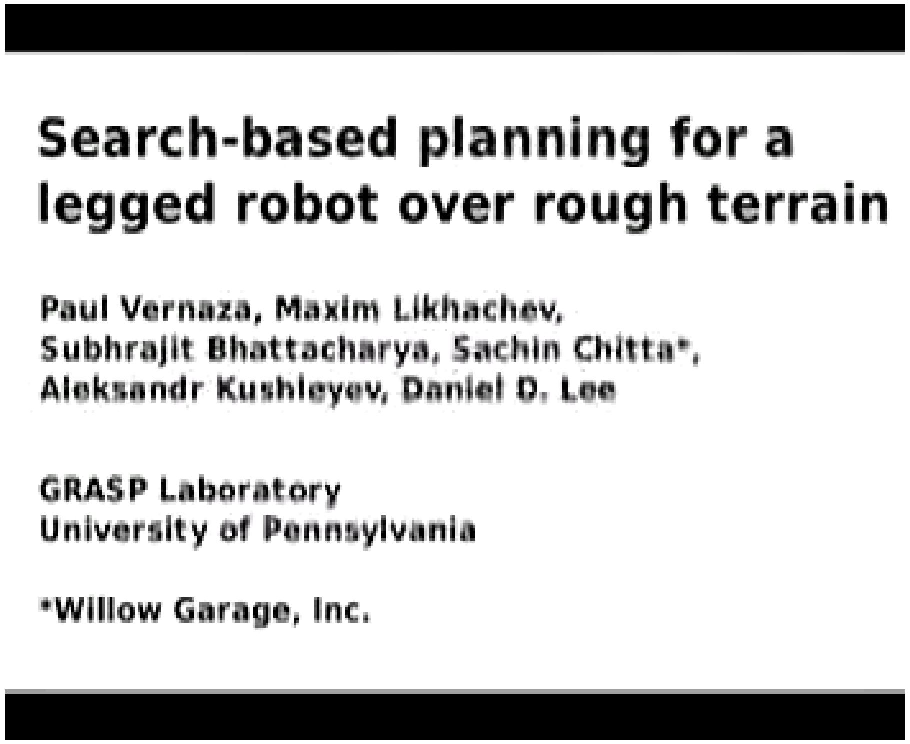
- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to r support triangle without
  - **Terrain slope**
    - Ensures terrain sl directio
  - **Terrain cost**
    - Considers slippage potential given terrain

Lots of features make up the cost function.
Fine tuning them is not fun ☹

There are ways to learn the weights
(e.g., Learning to Search [Ratliff, Silver & Bagnell, '09])



Sometimes smart but often stupid

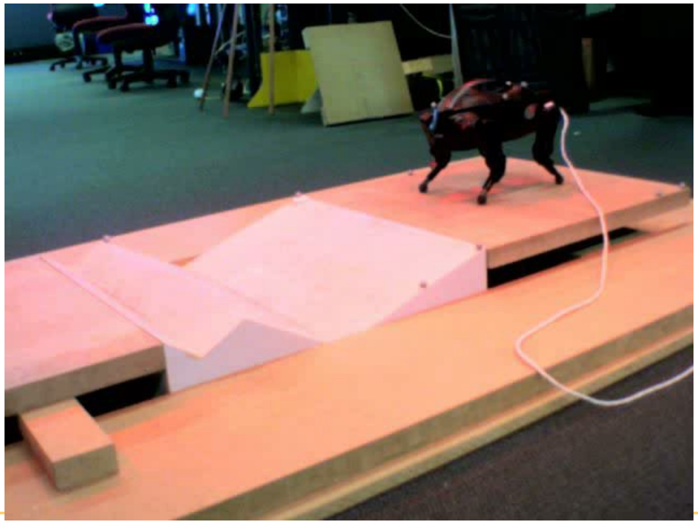


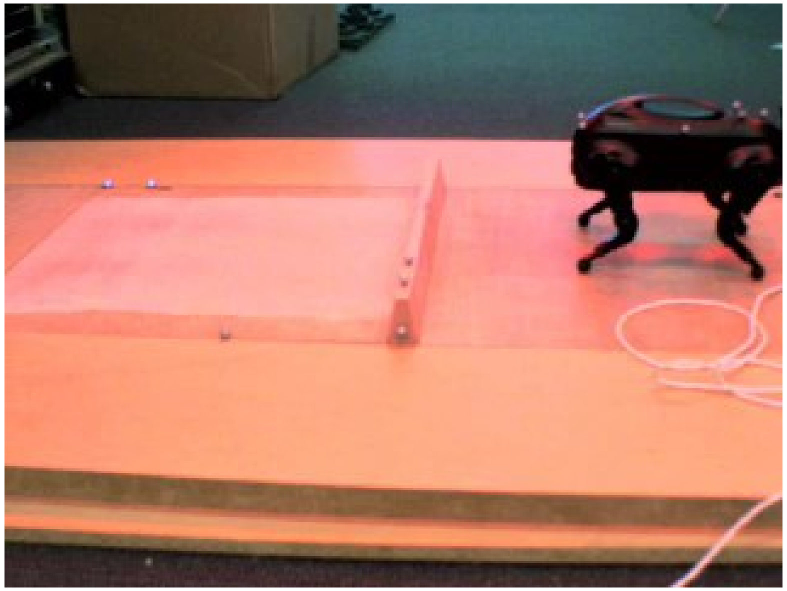
Search-based planning for a legged robot over rough terrain

Paul Vernaza, Maxim Likhachev, Subhrajit Bhattacharya, Sachin Chitta*, Aleksandr Kushleyev, Daniel D. Lee

GRASP Laboratory
University of Pennsylvania

*Willow Garage, Inc.





no footstep planning

What You Should Know…

- General state machine for mobile manipulation
- The dimensionality when planning footsteps for quadrupedal (and bipedal) robots
- Appreciate the complexity of cost components when planning for quadrupedal (and bipedal) robots