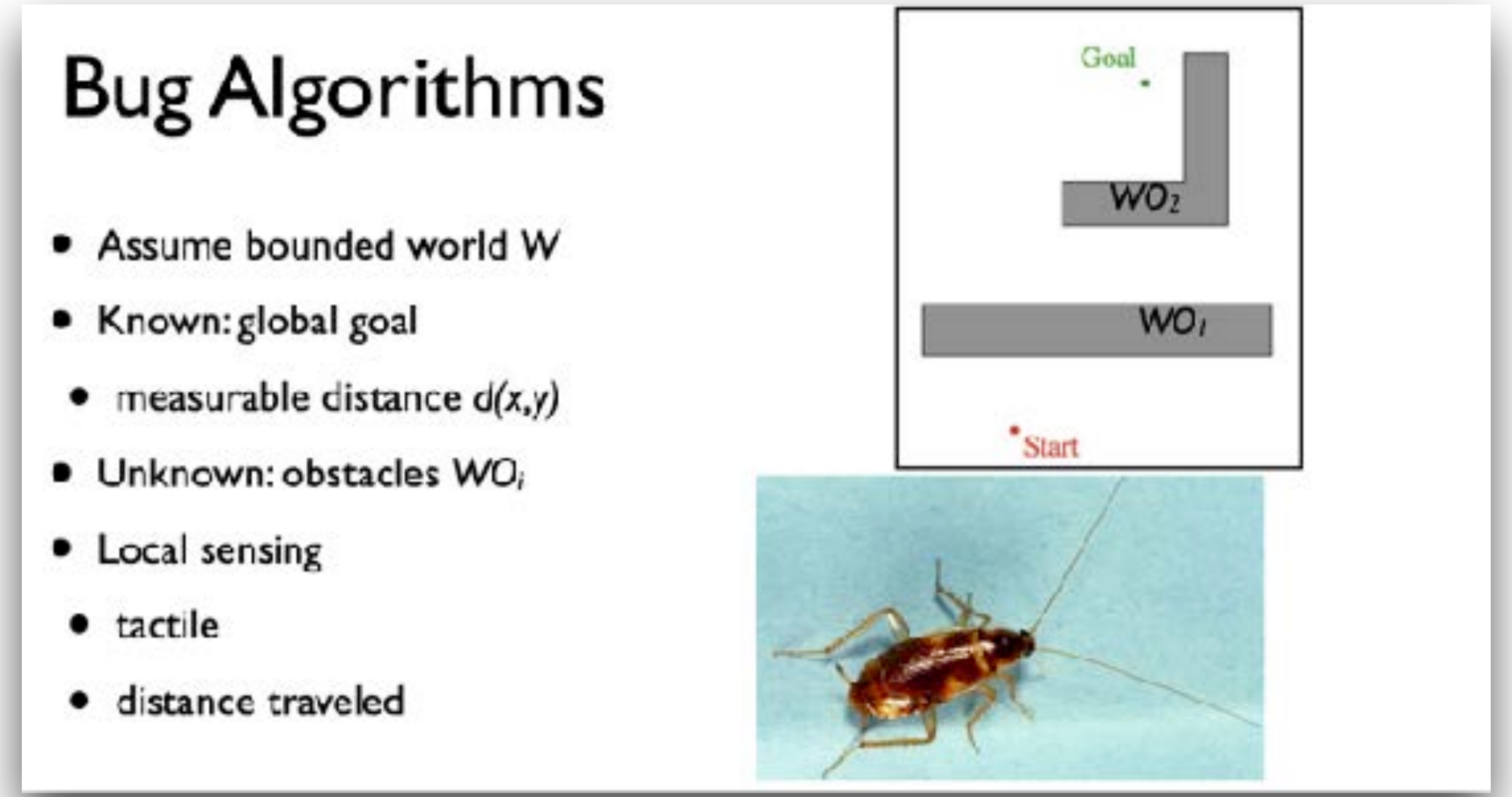
Lecture 11
Planning - III - Configuration Spaces

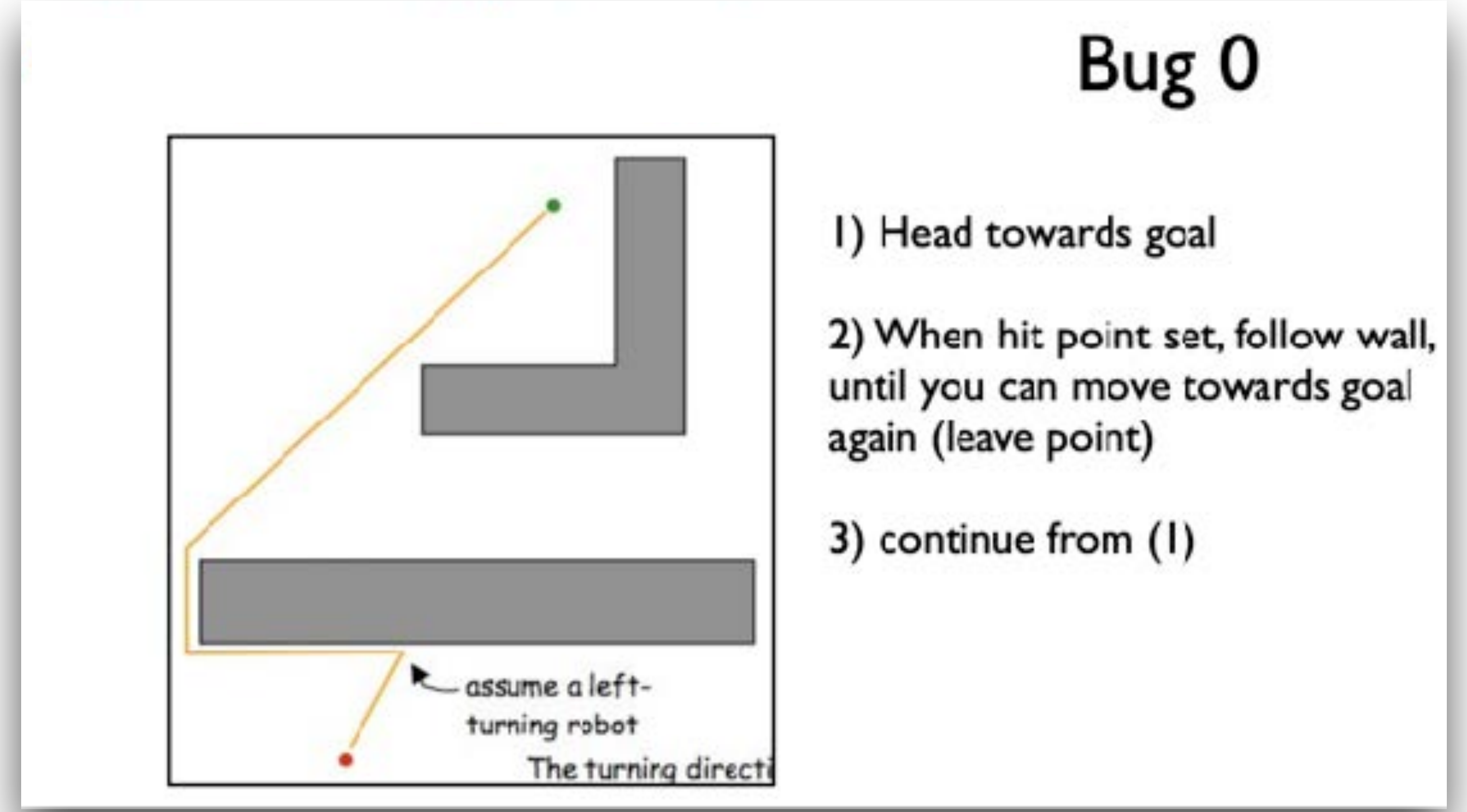
Course Logistics

- **Quiz 5 was posted yesterday and was due today at noon.**
- Project 4 was posted on 02/14 and will be due on 02/28.
  - Start early!

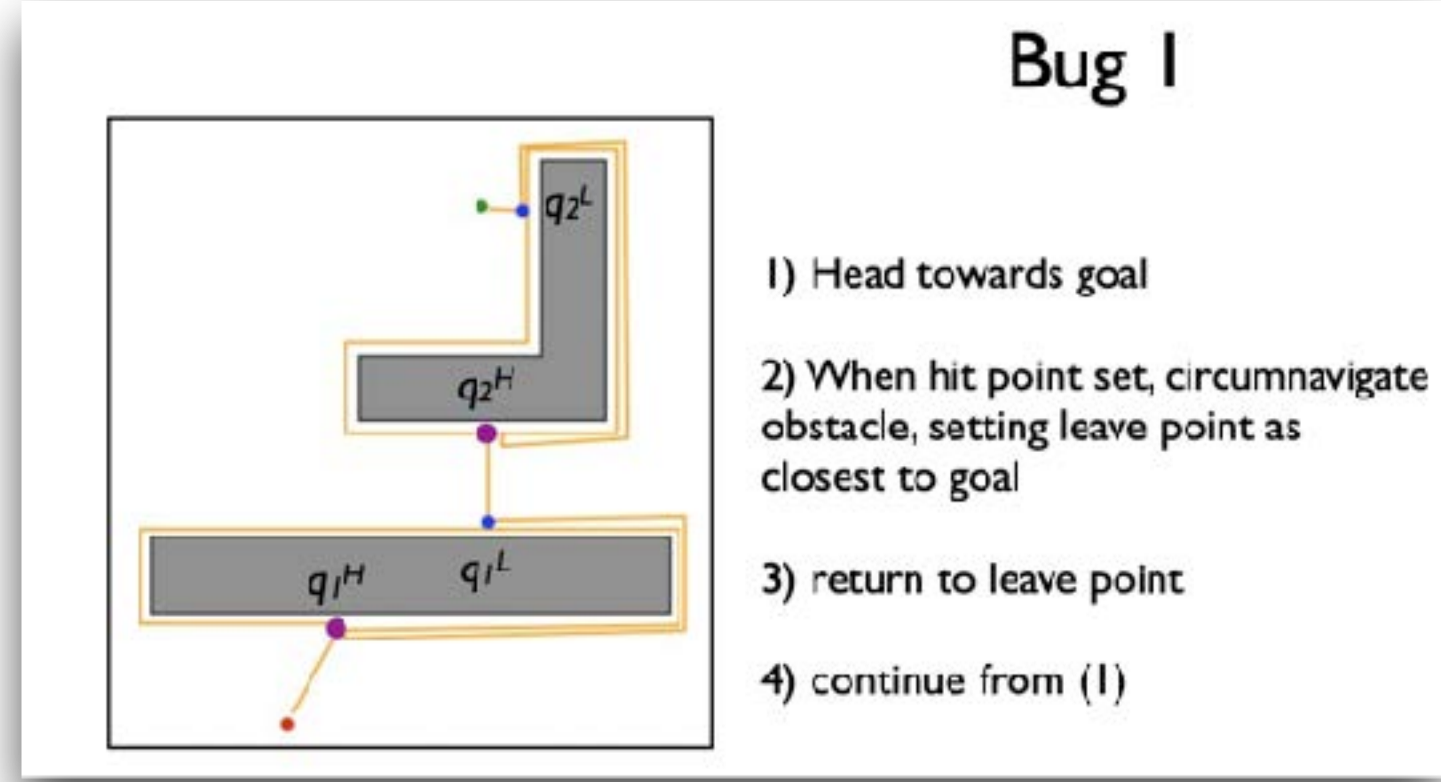
Previously



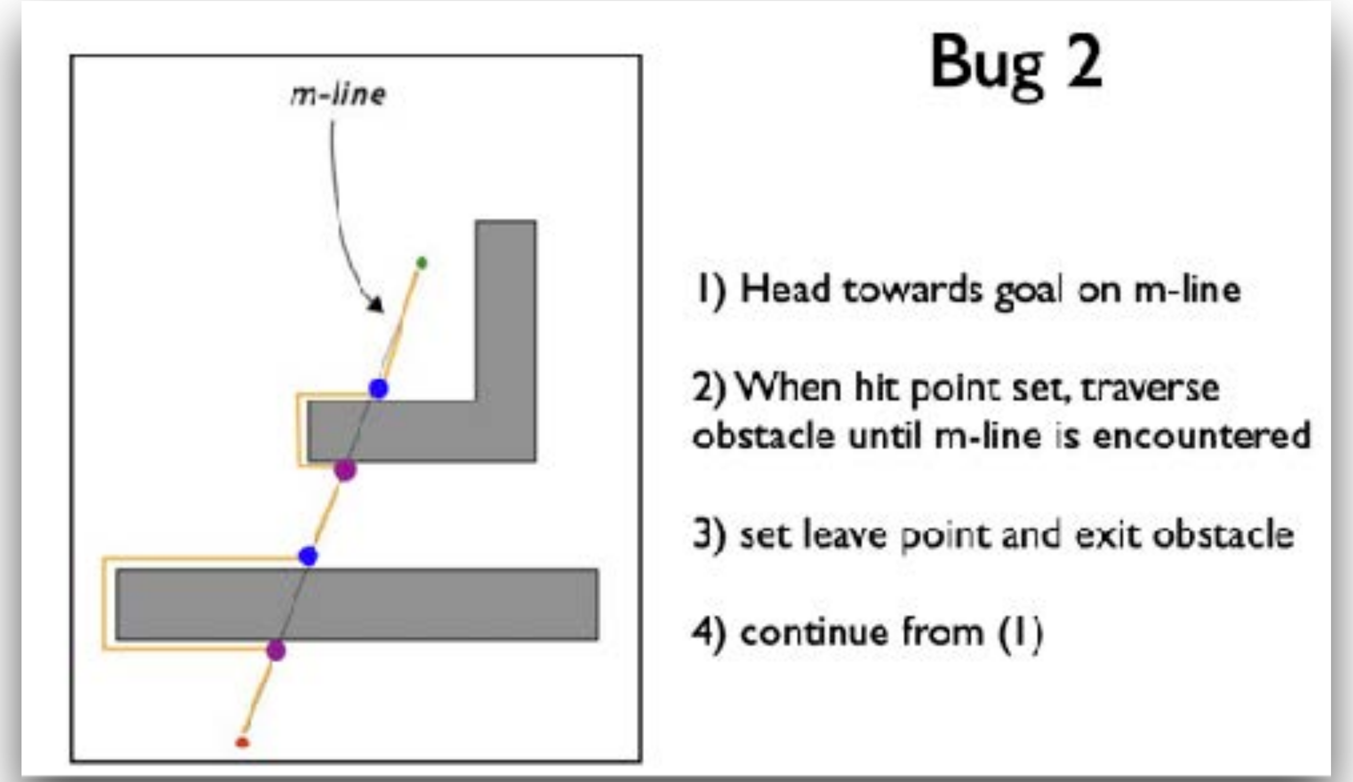
Bug Algorithms
Assume bounded world W
Known: global goal
measurable distance d(x,y)
Unknown: obstacles WO_i
Local sensing
tactile
distance traveled
Goal
WO_2
WO_1
Start



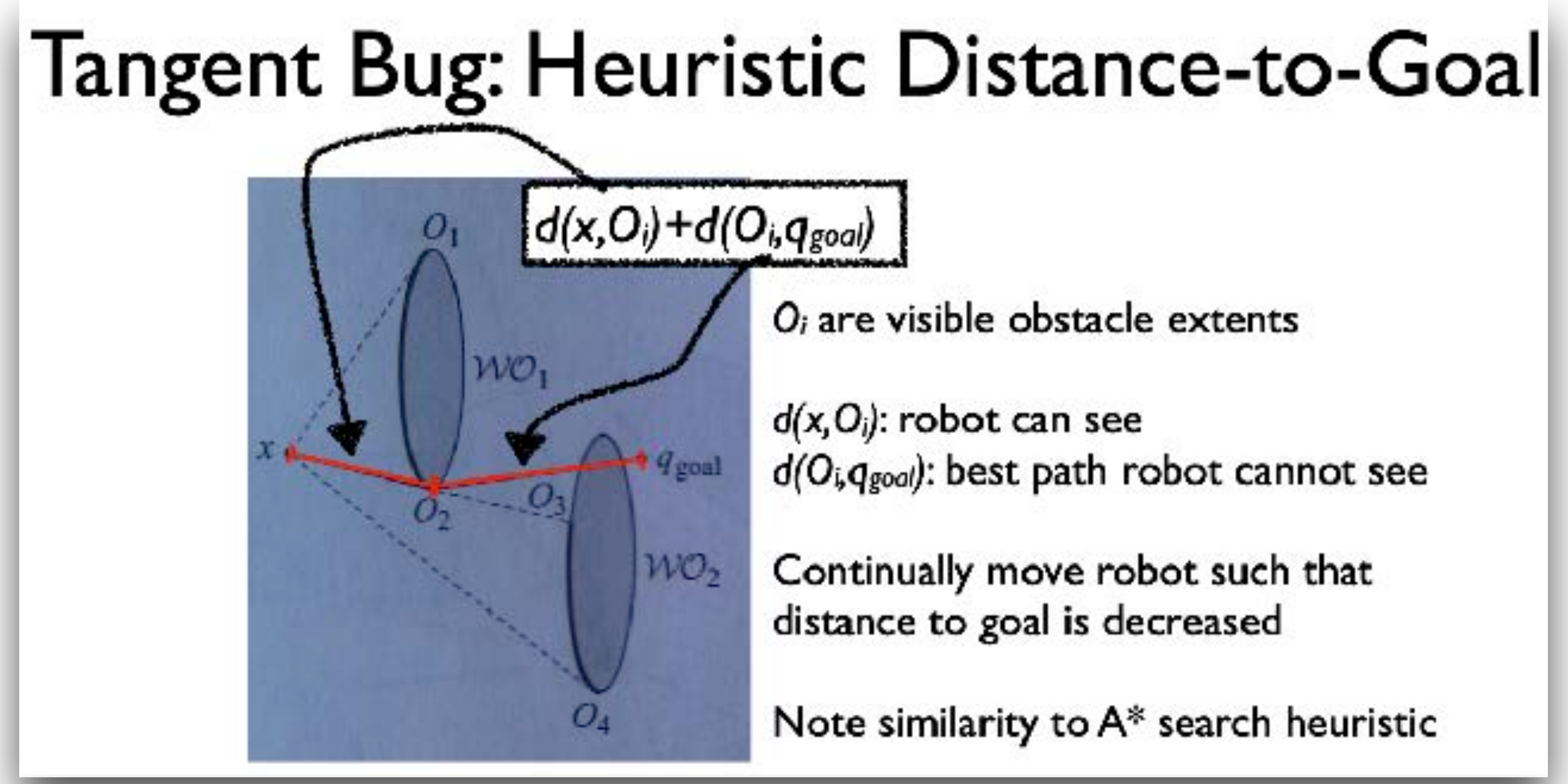
Bug 0
1) Head towards goal
2) When hit point set, follow wall, until you can move towards goal again (leave point)
3) continue from (1)
assume a left-turning robot
The turning directi



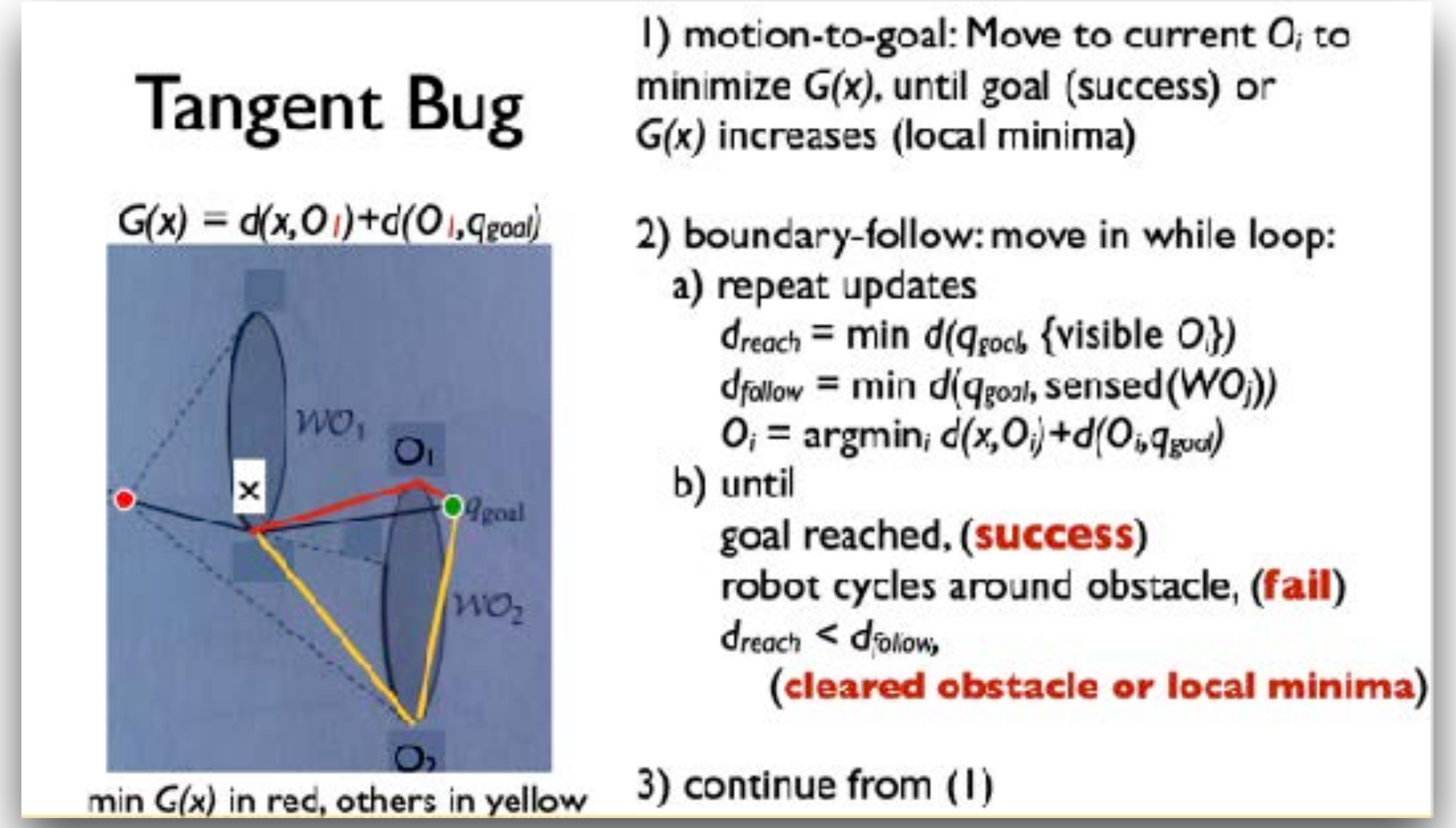
Bug 1
1) Head towards goal
2) When hit point set, circumnavigate obstacle, setting leave point as closest to goal
3) return to leave point
4) continue from (1)
q2^L
q2^H
q1^H
q1^L



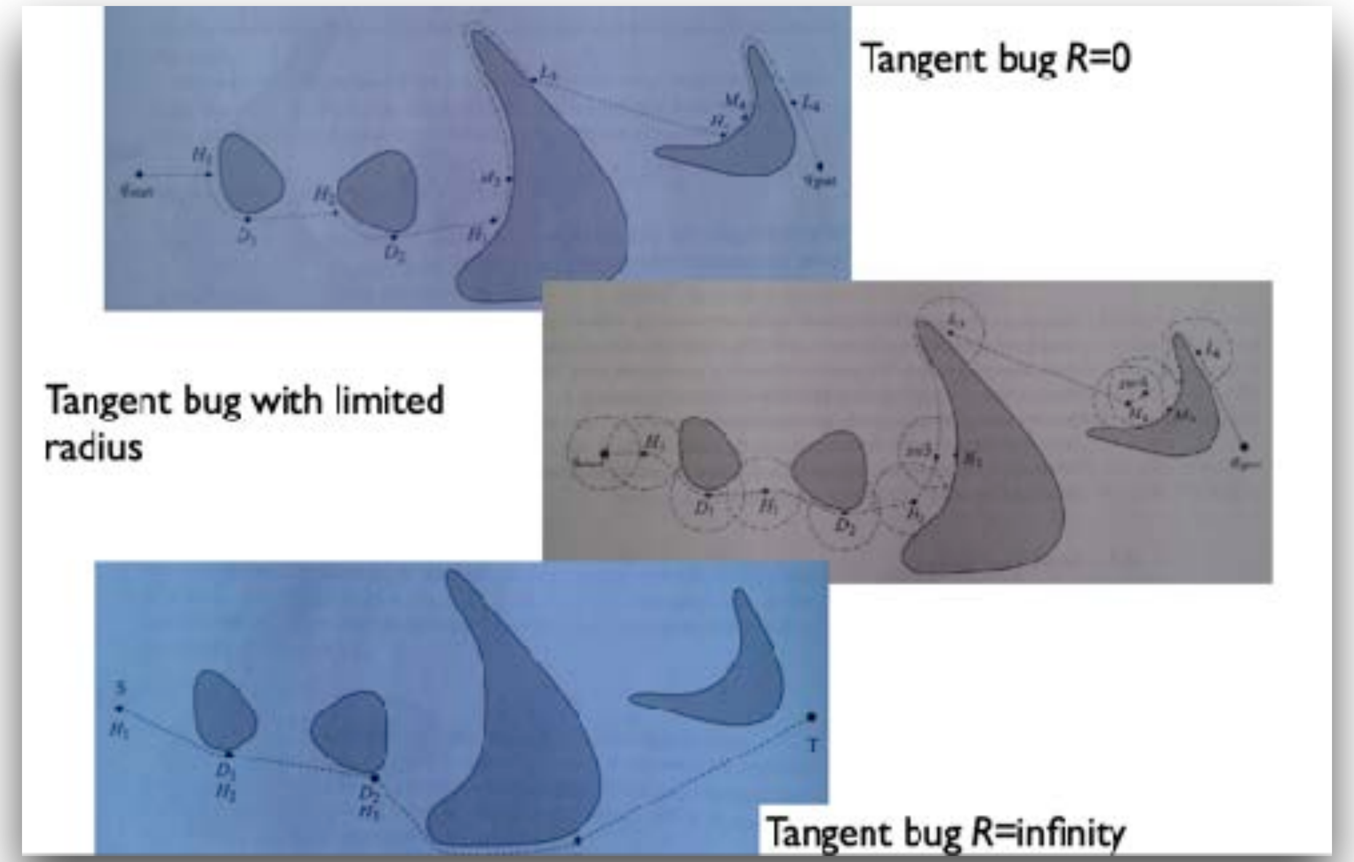
Bug 2
m-line
1) Head towards goal on m-line
2) When hit point set, traverse obstacle until m-line is encountered
3) set leave point and exit obstacle
4) continue from (1)



Tangent Bug: Heuristic Distance-to-Goal
d(x,O_i)+d(O_i,q_goal)
O_i are visible obstacle extents
d(x,O_i): robot can see
d(O_i,q_goal): best path robot cannot see
Continually move robot such that distance to goal is decreased
Note similarity to A* search heuristic
O_1
WO_1
x
O_2
O_3
q_goal
WO_2
O_4



Tangent Bug
G(x) = d(x,O_i)+d(O_i,q_goal)
min G(x) in red, others in yellow
1) motion-to-goal: Move to current O_i to minimize G(x), until goal (success) or G(x) increases (local minima)
2) boundary-follow: move in while loop:
a) repeat updates
d_reach = min d(q_goal, {visible O_i})
d_follow = min d(q_goal, sensed(WO_j))
O_i = argmin_i d(x,O_i)+d(O_i,q_goal)
b) until
goal reached, (success)
robot cycles around obstacle, (fail)
d_reach < d_follow,
(cleared obstacle or local minima)
3) continue from (1)
WO_1
O_1
q_goal
WO_2
O_2



Tangent bug R=0
Tangent bug with limited radius
Tangent bug R=infinity

Search Bounds:

Bug 1

Bounds on path distance, assuming
D: distance start-to-goal
P_i: obstacle perimeter

Best case: D

Worst case: $D + 1.5\sum_i P_i$

Bug 2

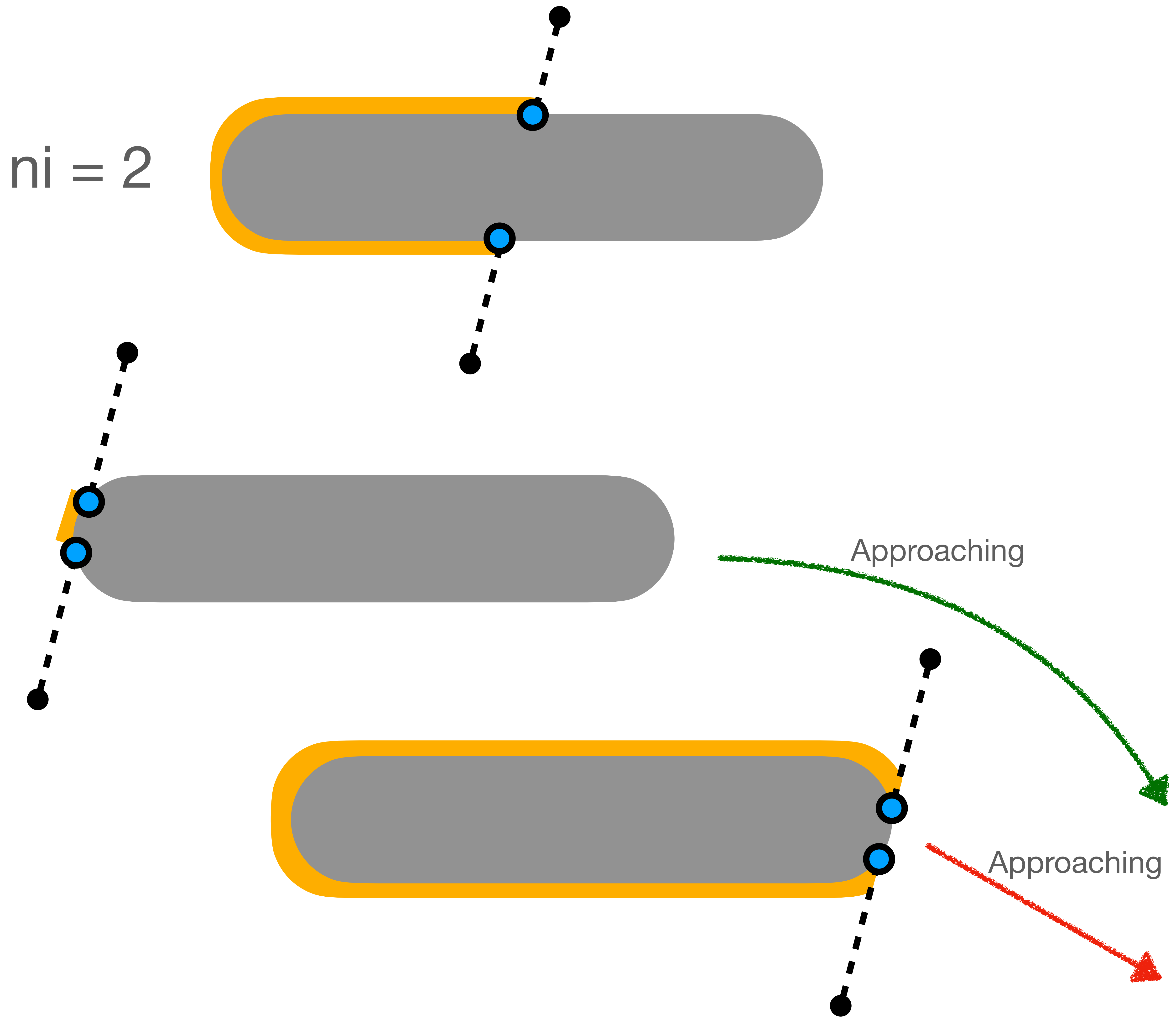
Bounds on path distance, assuming

- D: distance start-to-goal
- P_i: obstacle perimeter
- n_i: number of m-line intersections for WO_i

Best case: D

Worst case: $D + \sum_i (n_i/2)P_i$

Search Bounds:



Bug 2

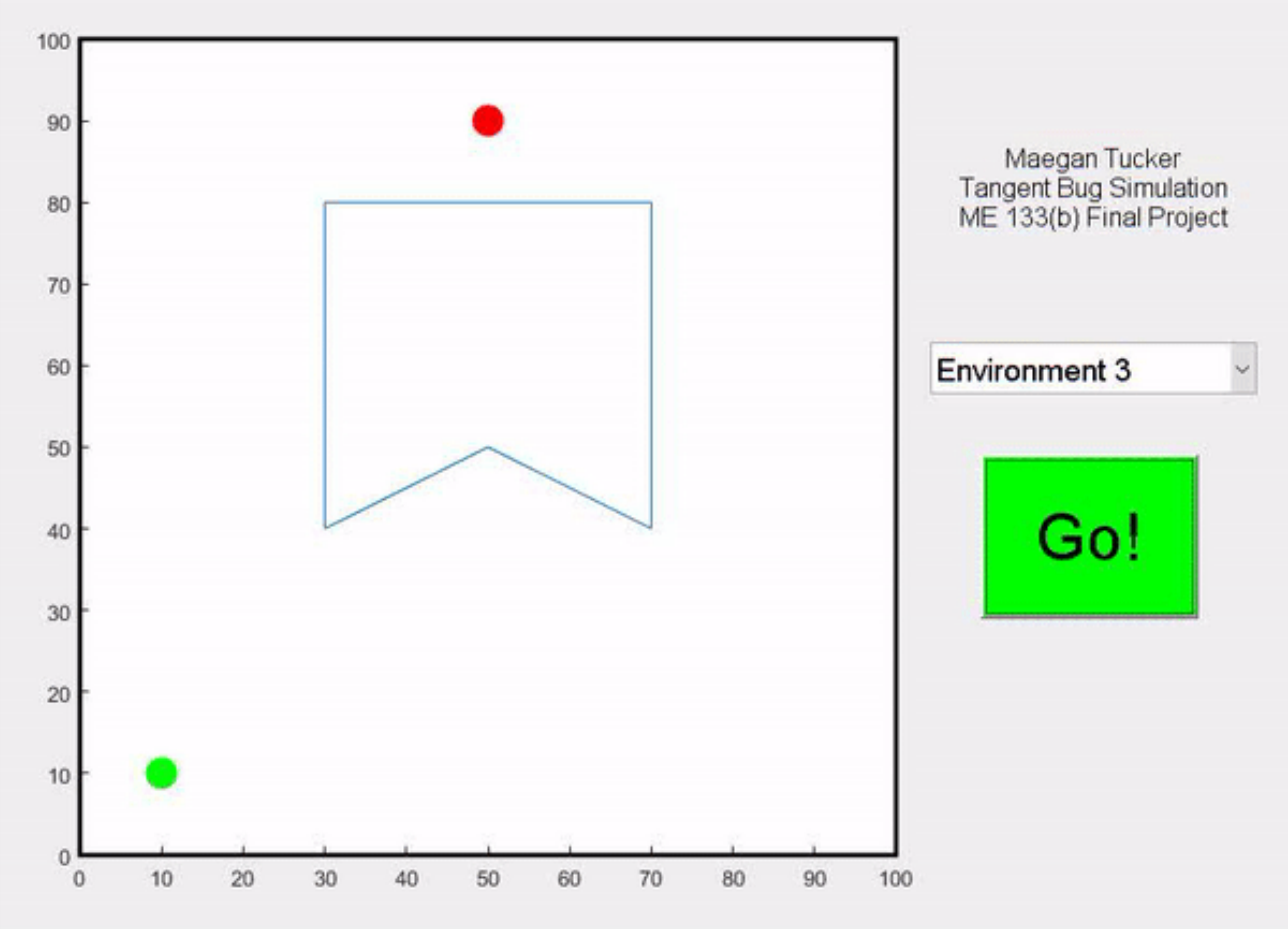
Bounds on path distance, assuming

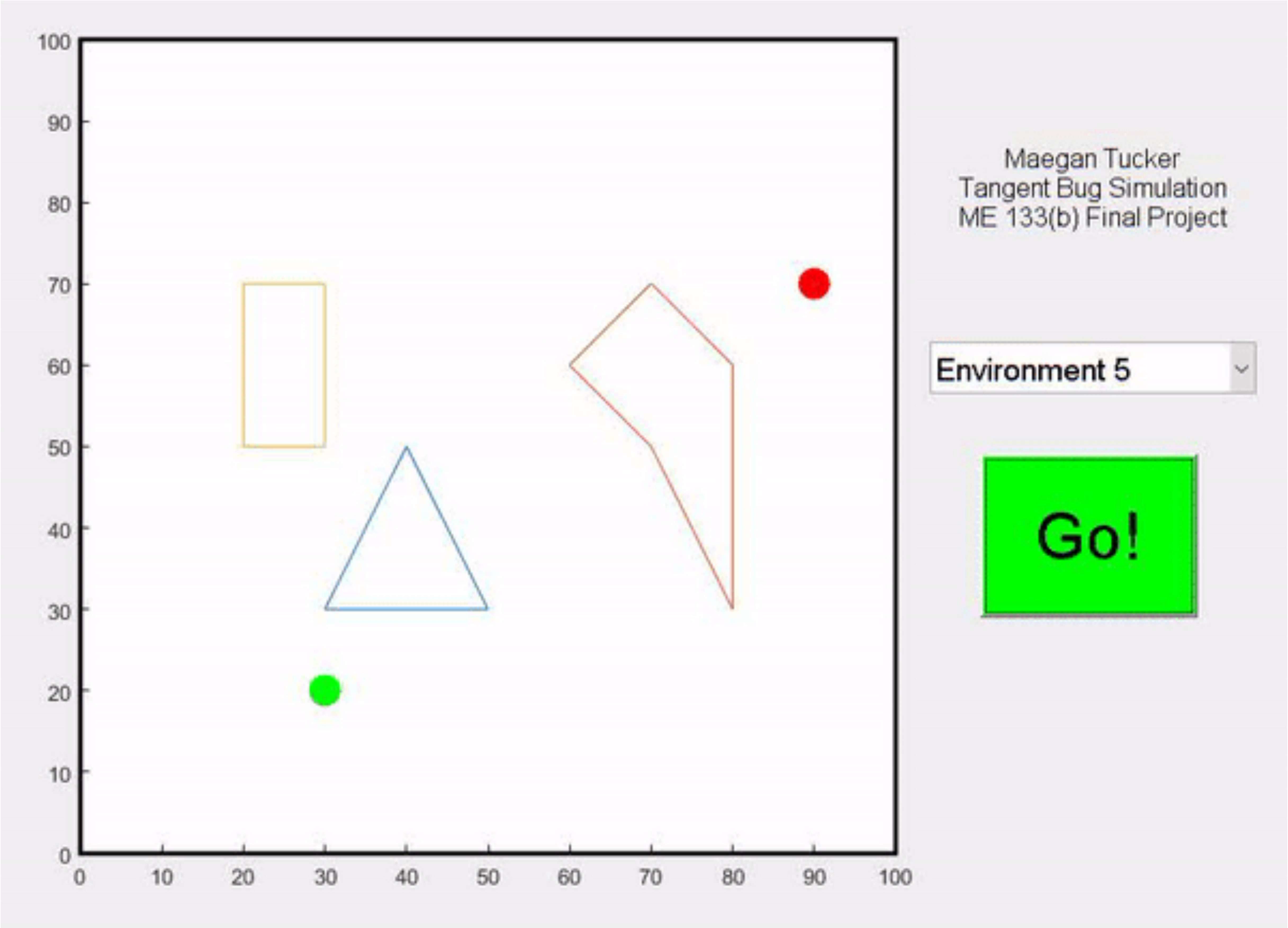
- D: distance start-to-goal
- P_i: obstacle perimeter
- n_i: number of m-line intersections for WO_i

Best case: D

Worst case: $D + \sum_i (n_i/2)\, P_i$

Tangent Bug

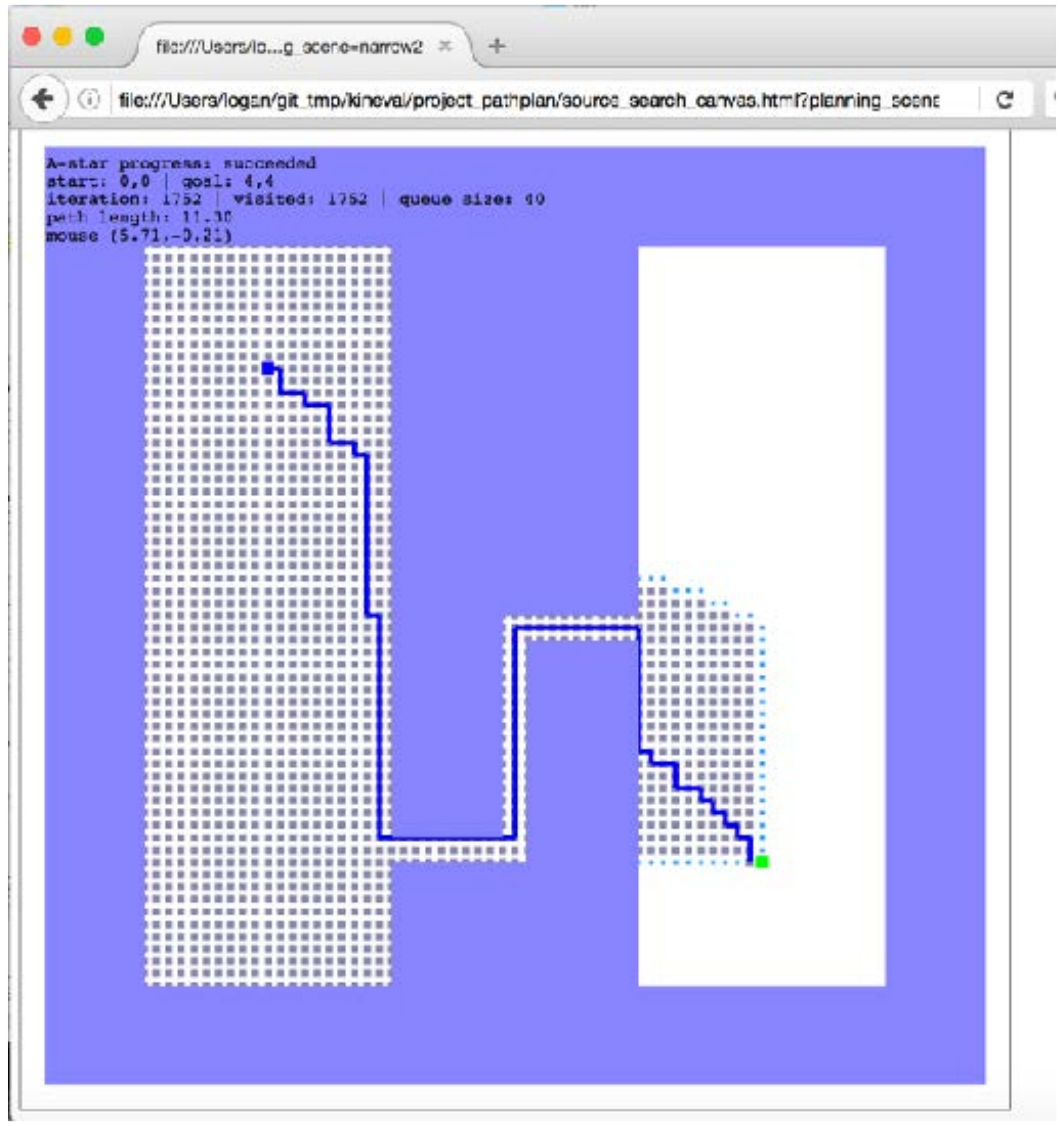




https://maegantucker.com/projects/2018-04-01-me133b/

Will our current search methods apply to this robot?

2D Path Planning



N-dimensional Motion Planning

Will our current search methods apply to this robot?

<u>Assumptions:</u>

- Known graph of traversability
    - How big is this graph? How was this graph built?
- Known localization and map/obstacles
    - How do we detect collisions?
    - Is our robot just a point in workspace?
- Known link geometry
    - Does robot geometry change wrt. configuration?

Configuration Spaces

seibertron.net

more than meets the eye

Configuration Space
(or C-space)

- C-space (Q) is the space of all possible configurations (q) of a system
  - kinematics: geometry of possible configurations, without respect to physics
  - dynamics: evolution of configurations over time wrt. physics

- Each degree of freedom (q_i) is a dimension of C-space

- The span of C-space is constrained by obstacles (QO_i), joint limits, etc.

Consider some examples of configuration spaces

Configuration Space

- Consider a robot d=21 DOFs, where each DOF can take 1 of n=10 angular values
- How many configurations?

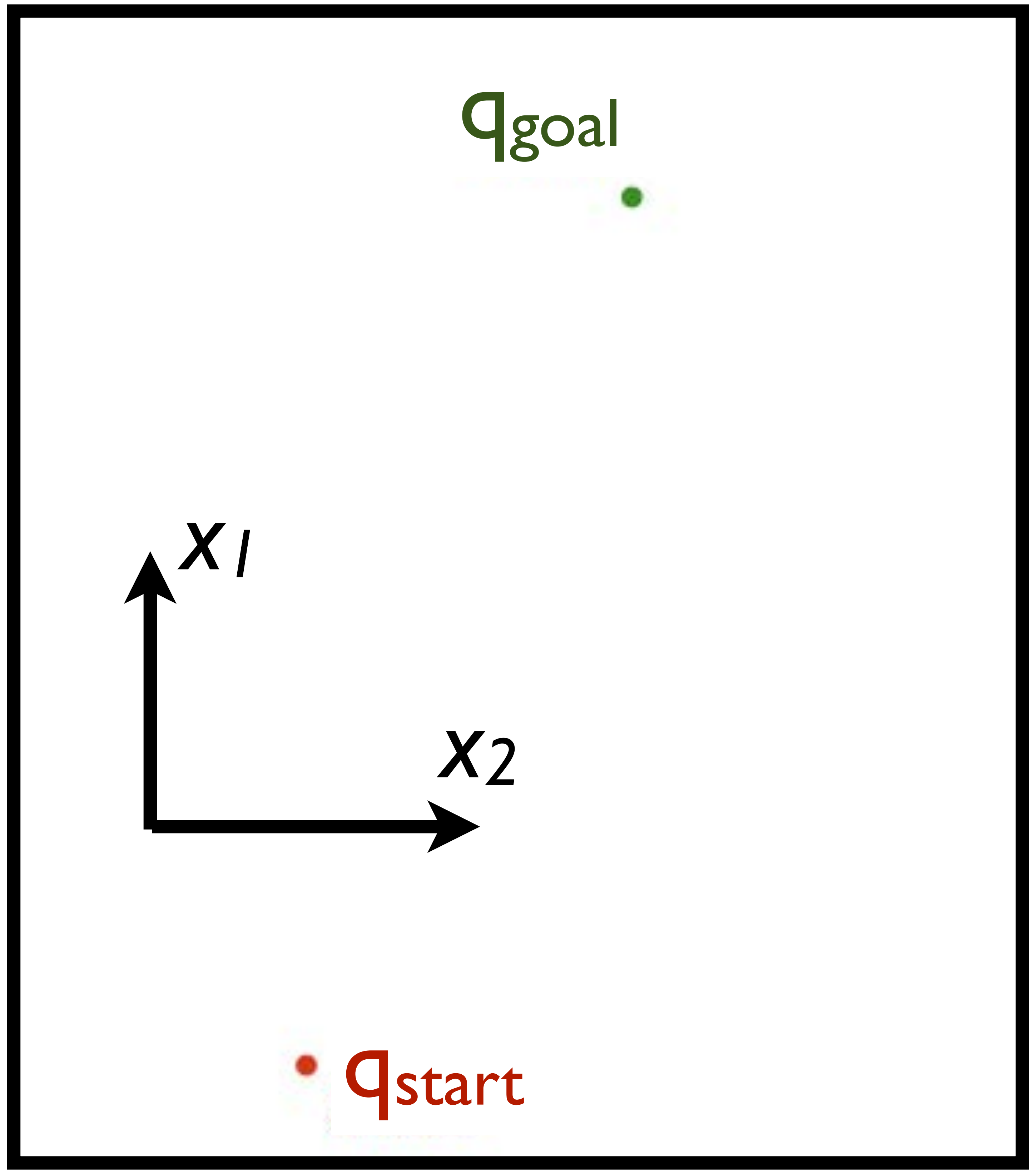
[redacted]

Configuration Space

- Consider a robot d=21 DOFs, where each DOF can take 1 of n=10 angular values
- How many configurations?
  - 10^{21}, n^d in general
- **"Curse of dimensionality"**
  - exponential growth of C-space wrt. number of DOFs
- Obstacles also create discontinuities and nonlinearities in C-space

C-space examples

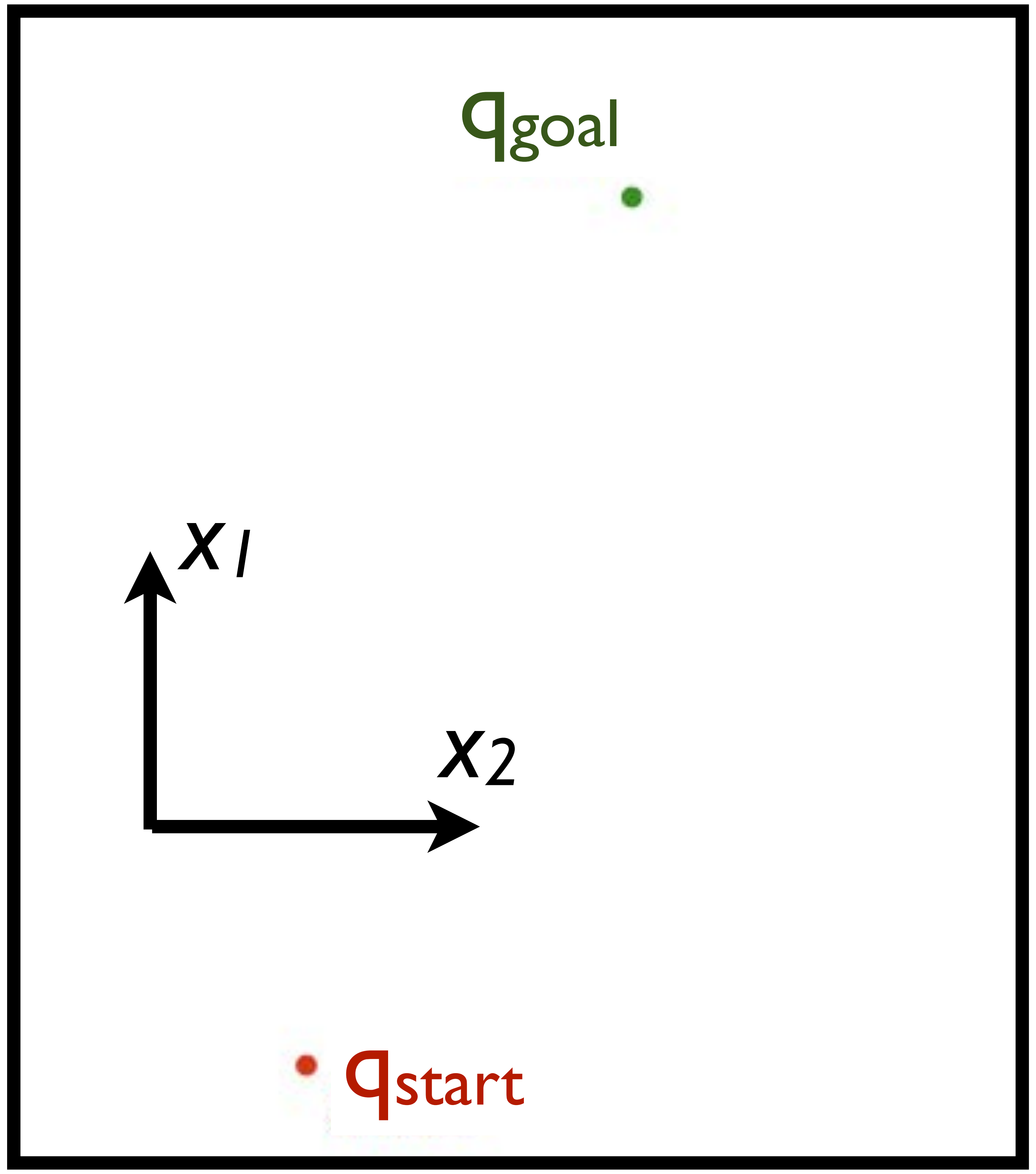
- How many configurations are in the C-space of a planar point robot in a bounded rectangular world?



C-space examples

- How many configurations are in the C-space of a planar point robot in a bounded rectangular world?
  - DOFs: 2, $\{x_1,x_2\}$
  - Number of poses is infinite
  - C-space: $\Re^2$

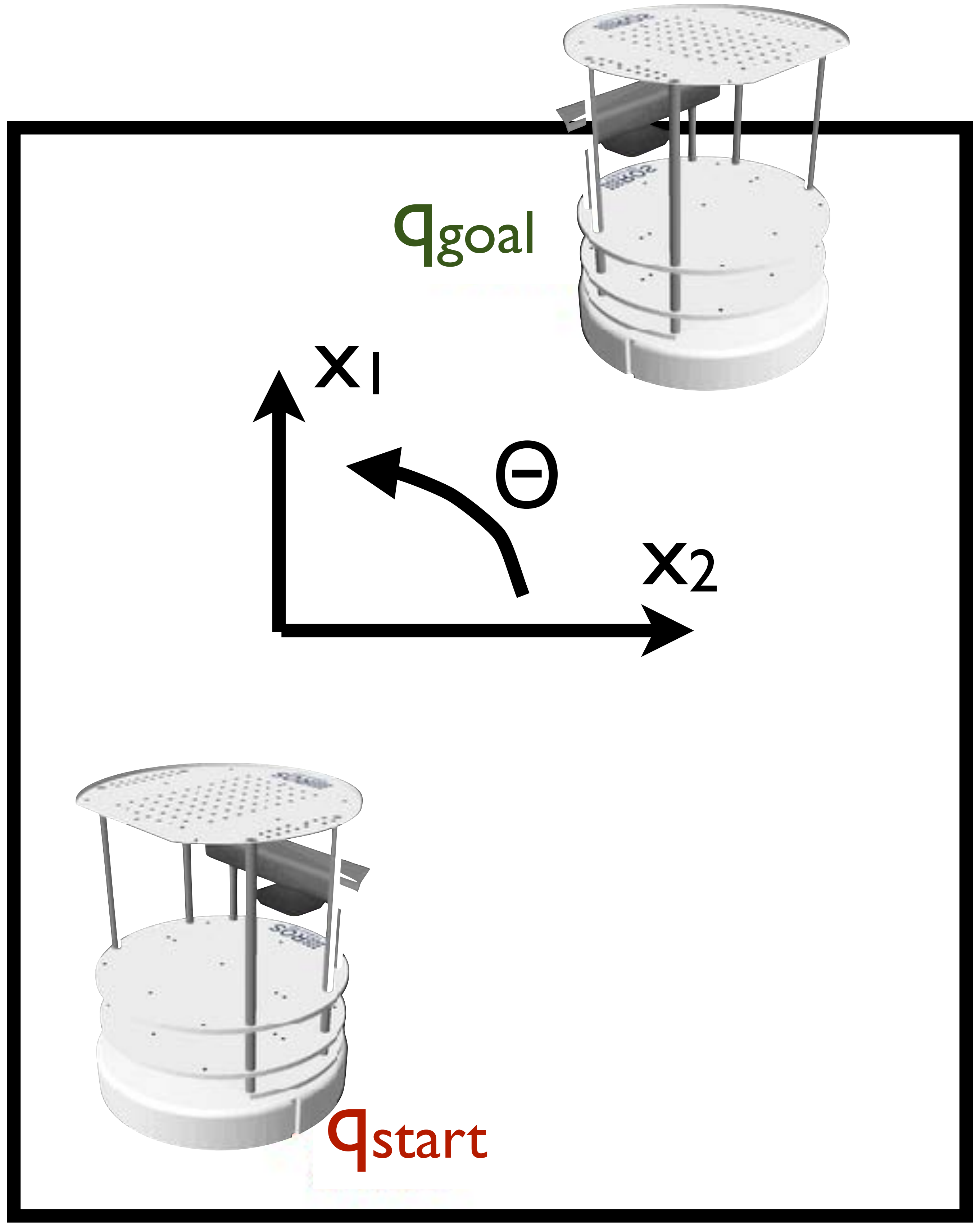
$\boxed{\text{Toplogically, this C-space is a homeomorphism of } \Re^2}$



C-space examples

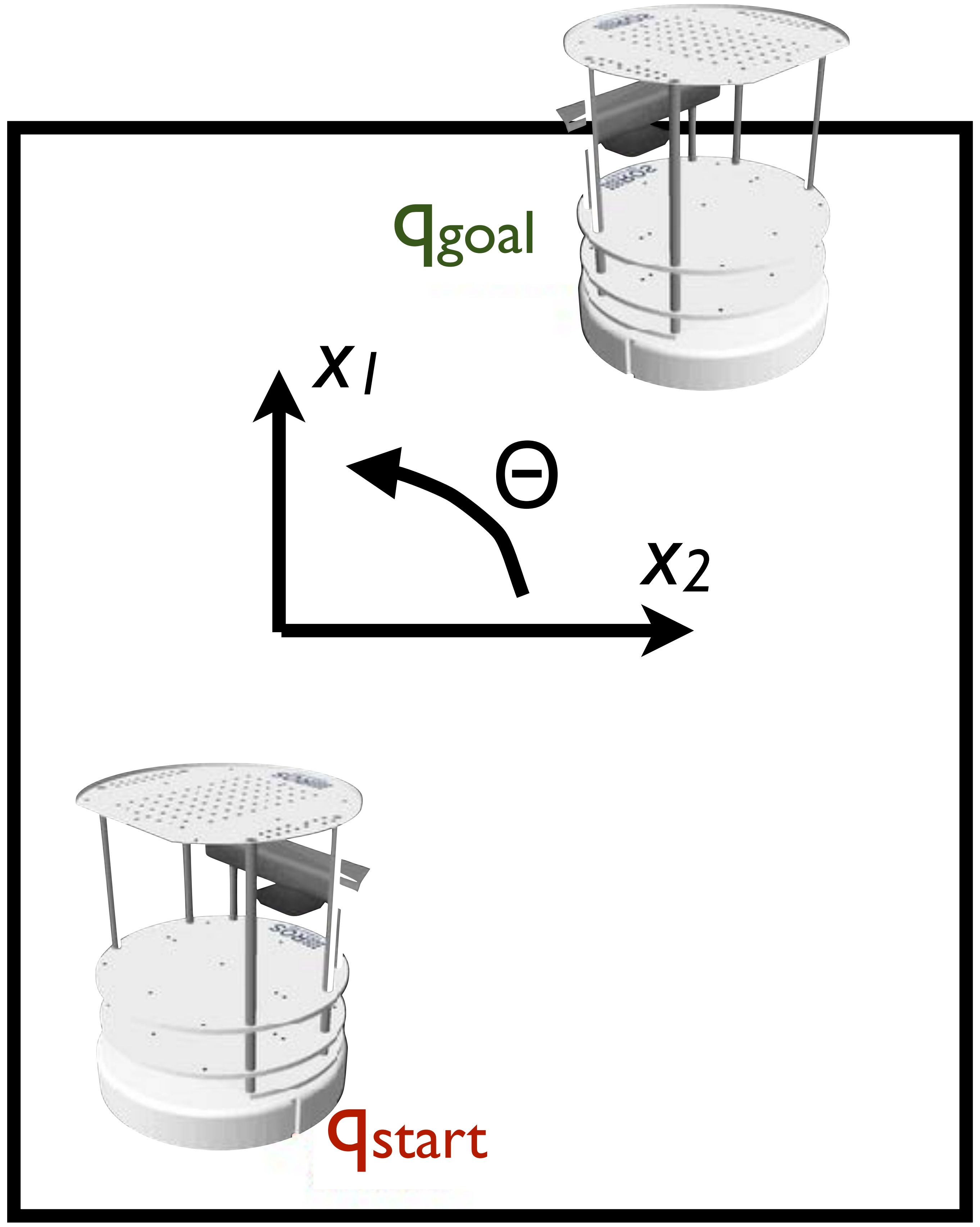
- What is the C-space of a Turtlebot?

[redacted]



C-space examples

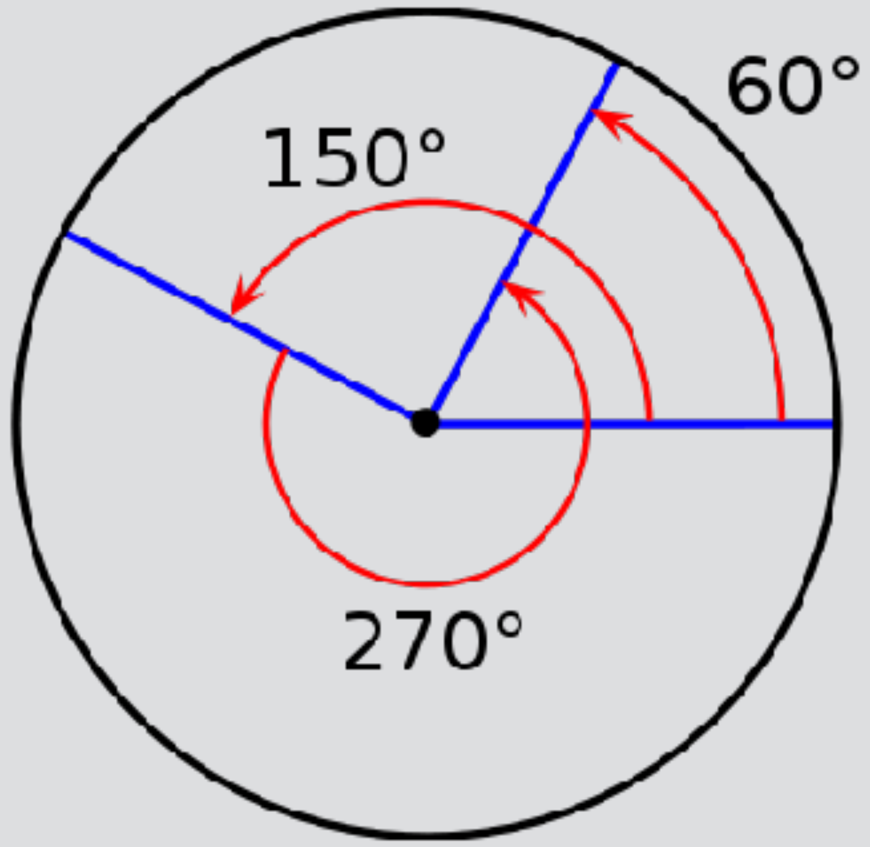
- What is the C-space of a Turtlebot?
  - DOFs: 3, $\{x_1, x_2, \Theta\}$
  - C-space: $\Re^2 \times S^1$



S^1 is the 1-sphere
group of 1D rotations

S^n is the n-sphere

$S^1 \times S^1 \neq S^n$



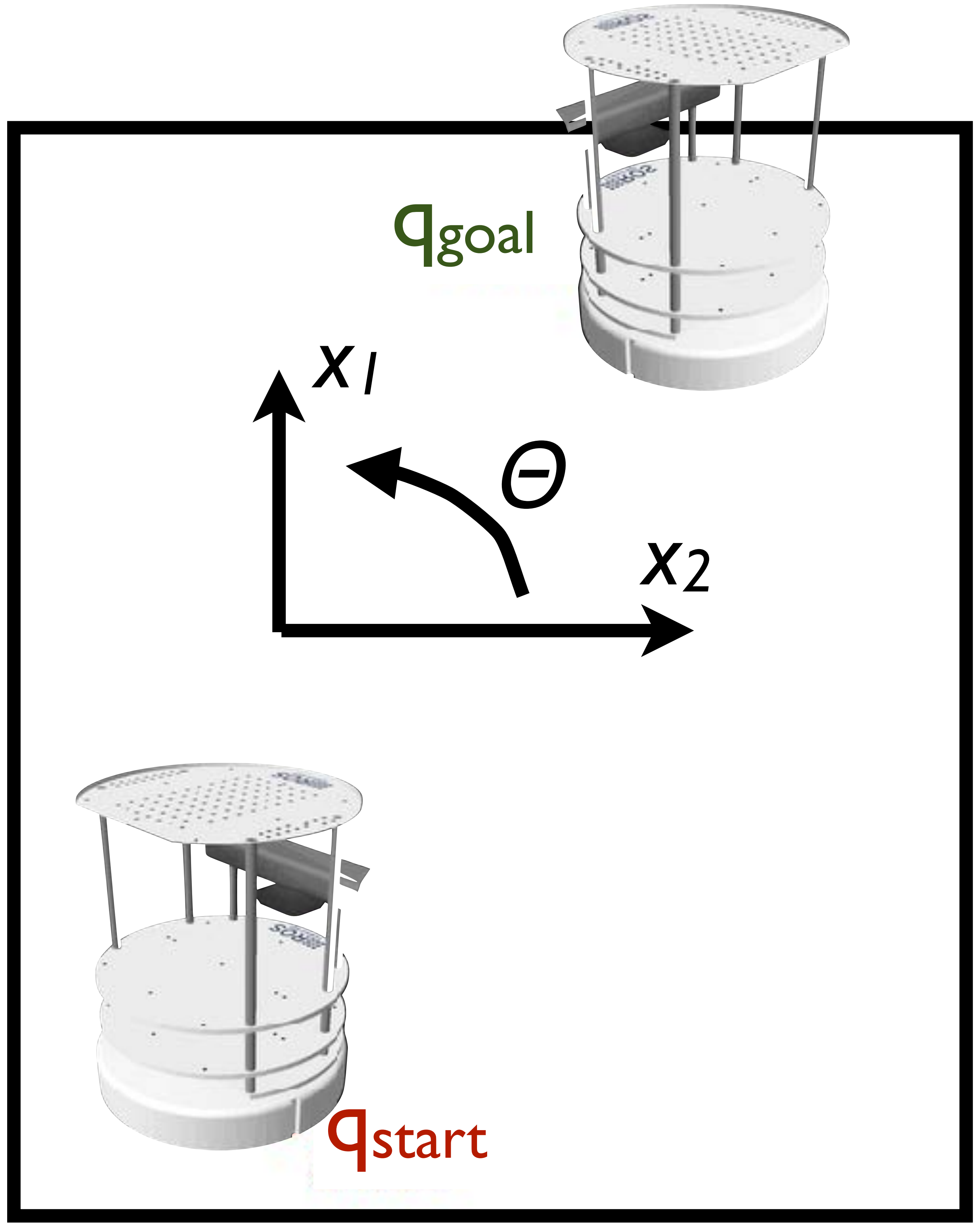
C-space examples

- What is the C-space of a Turtlebot?
  - DOFs: 3, $\{x_1, x_2, \Theta\}$
  - C-space: $\Re^2 \times S^1$

2D translation　　　rotation in 2D

$\Re^2 \times S^1$ is also known as the *SE(2)* group.

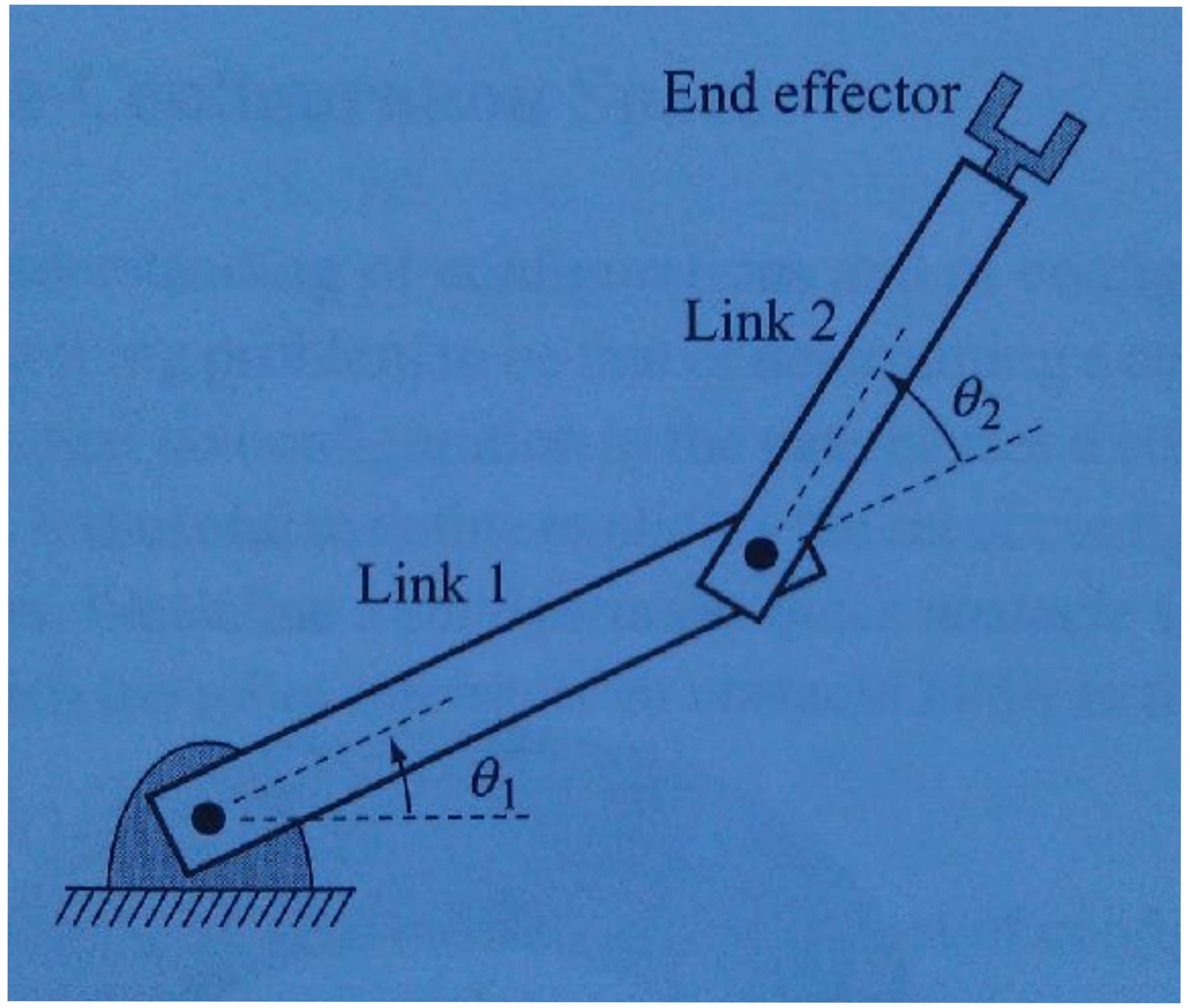
Group of homogeneous transformations in 2D



Slide borrowed from Michigan Robotics autorob.org

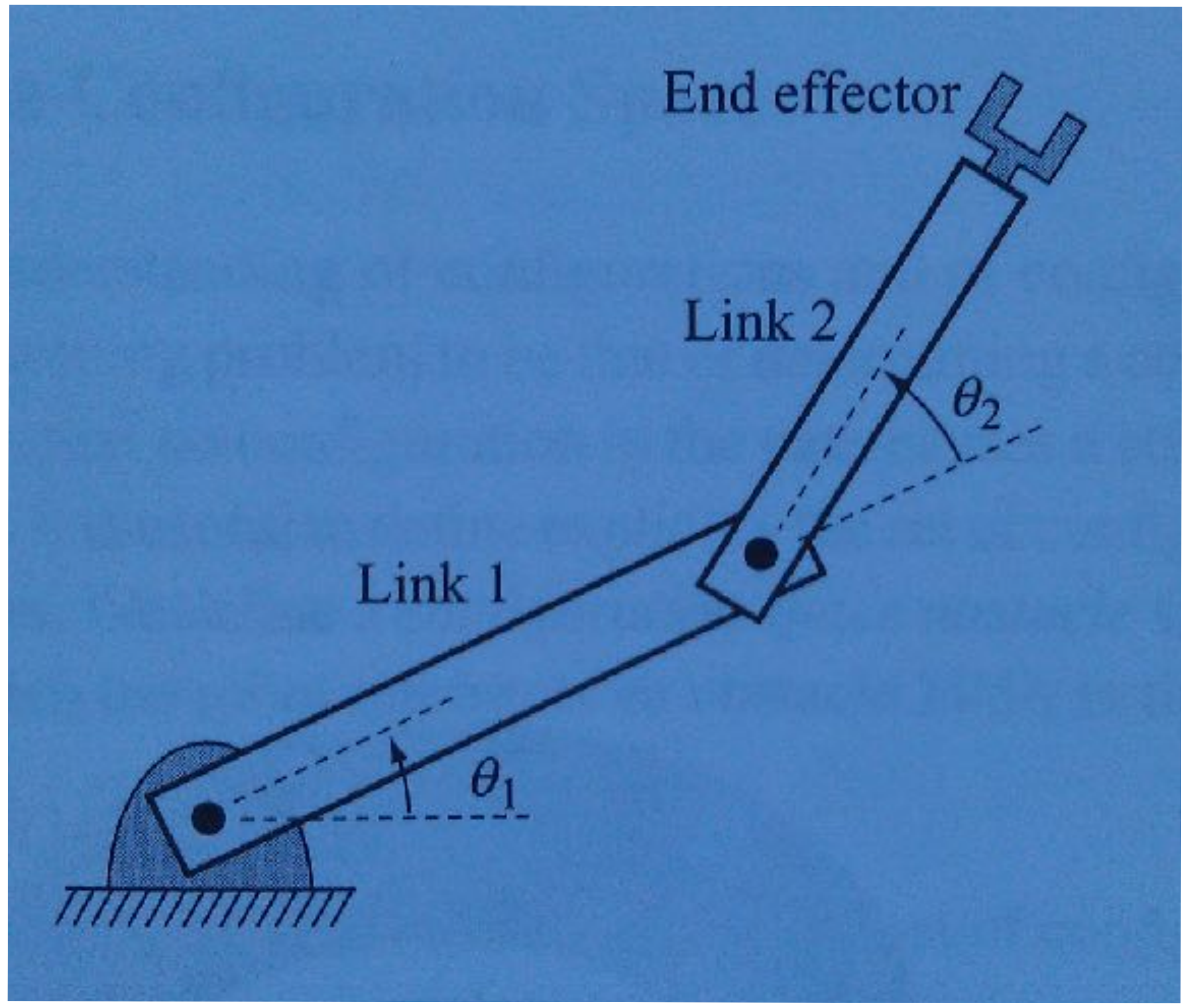
C-space examples

- What is the C-space of a planar arm with 2 rotational joints?



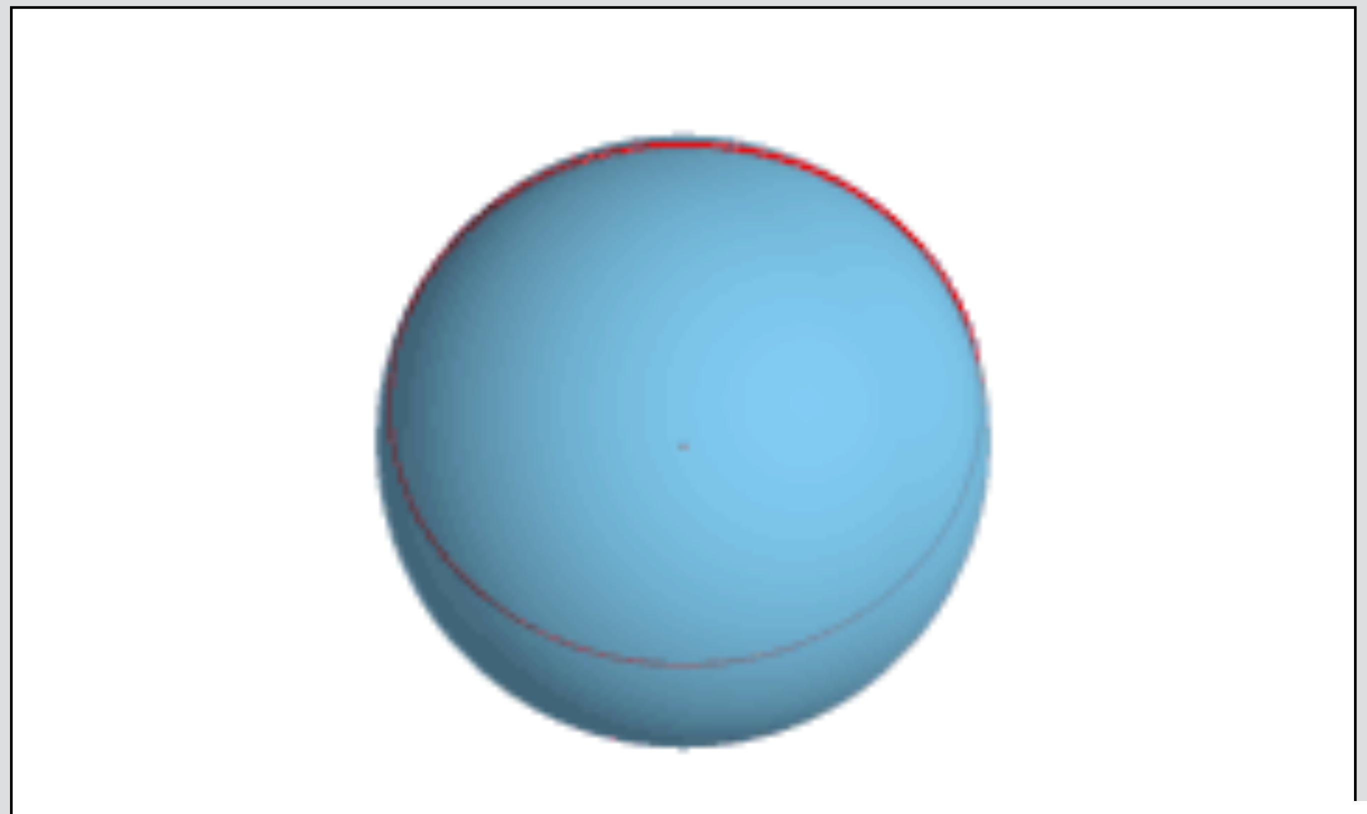
C-space examples

- What is the C-space of a planar arm with 2 rotational joints?
  - DOFs: [redacted]
  - C-space: [redacted]

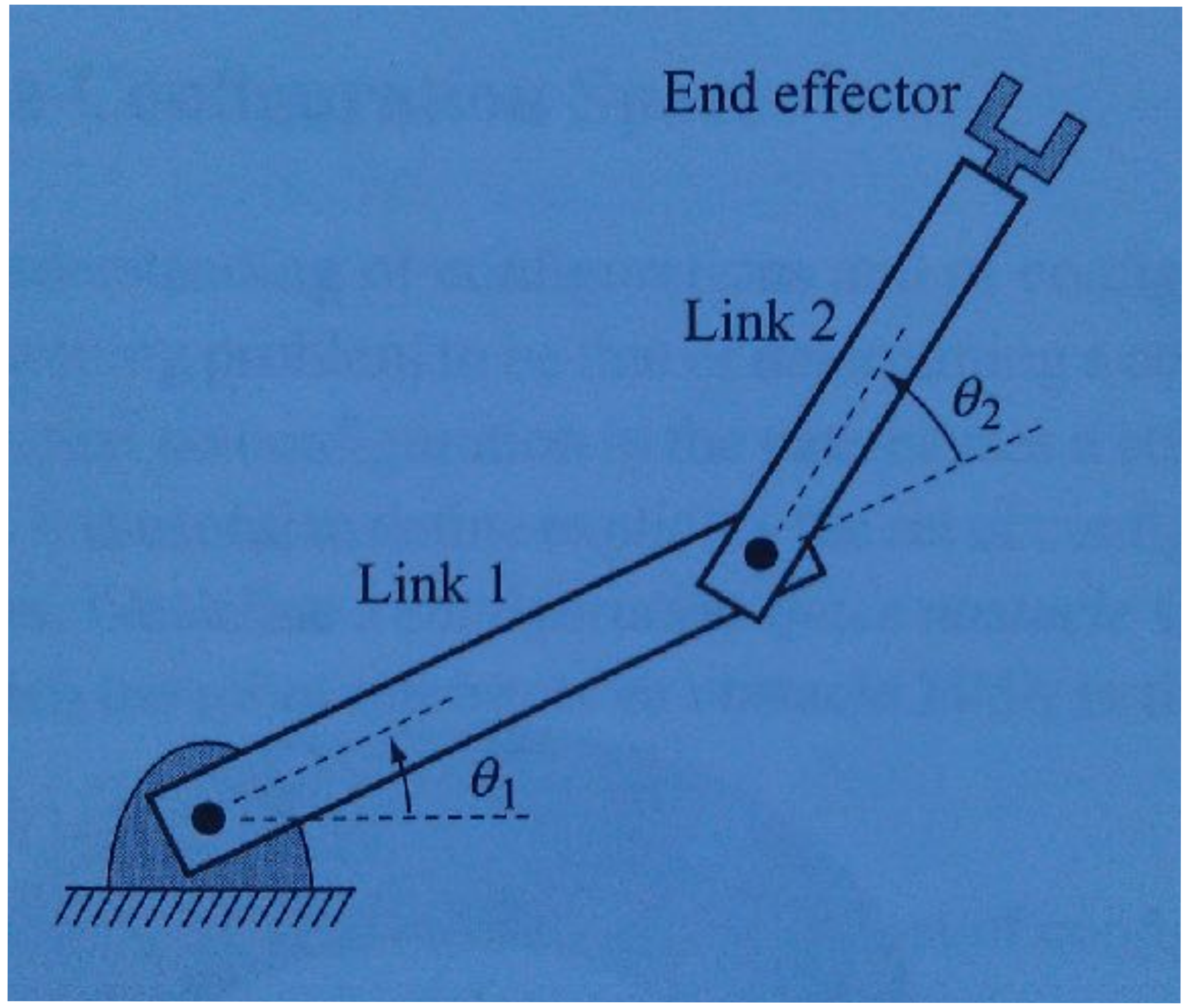


C-space examples

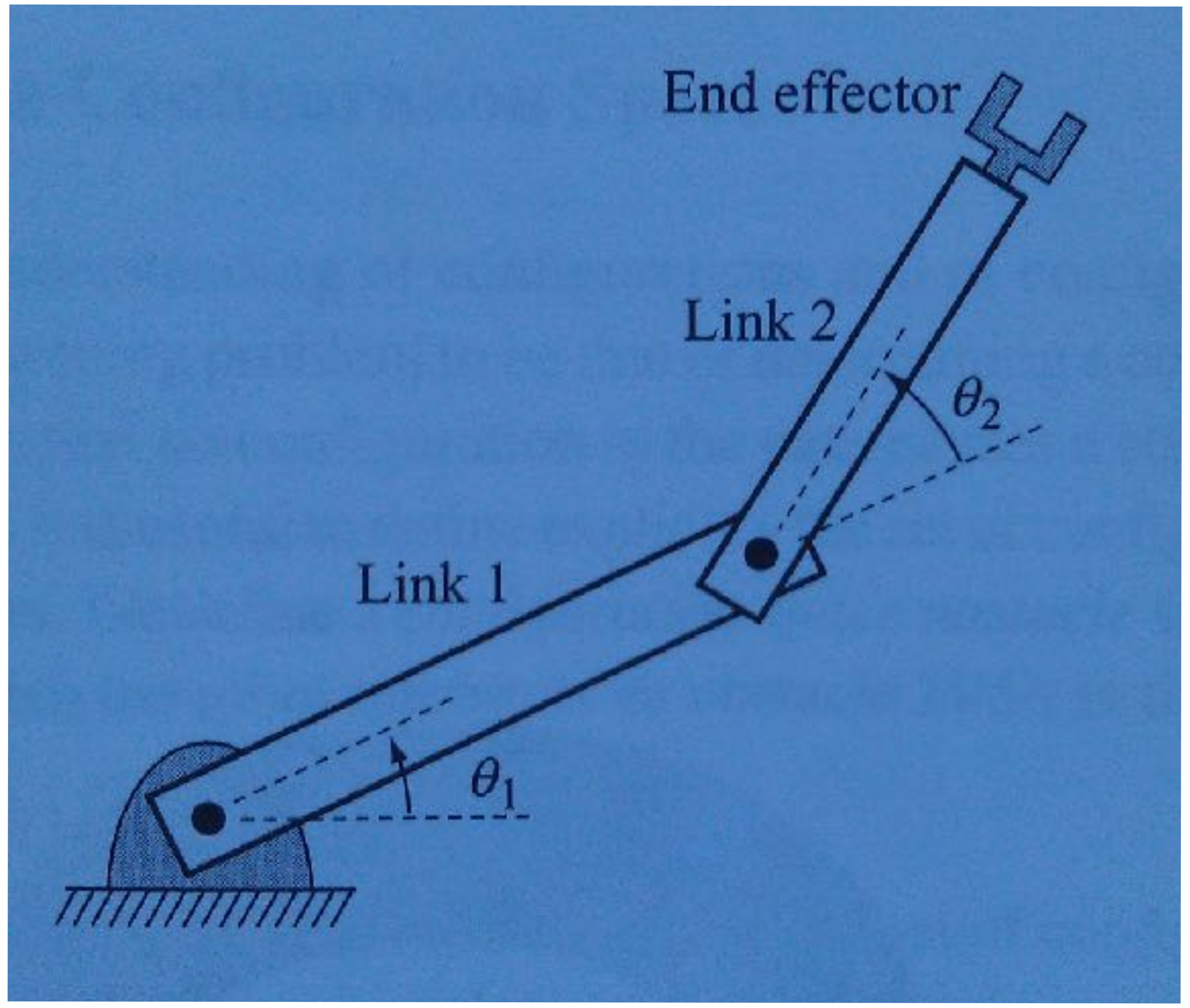
- What is the C-space of a planar arm with 2 rotational joints?
  - DOFs: 2, $\{\Theta_1, \Theta_2\}$
  - C-space: $\Re^2$ or S^2 or $S^1 \times S^1$?



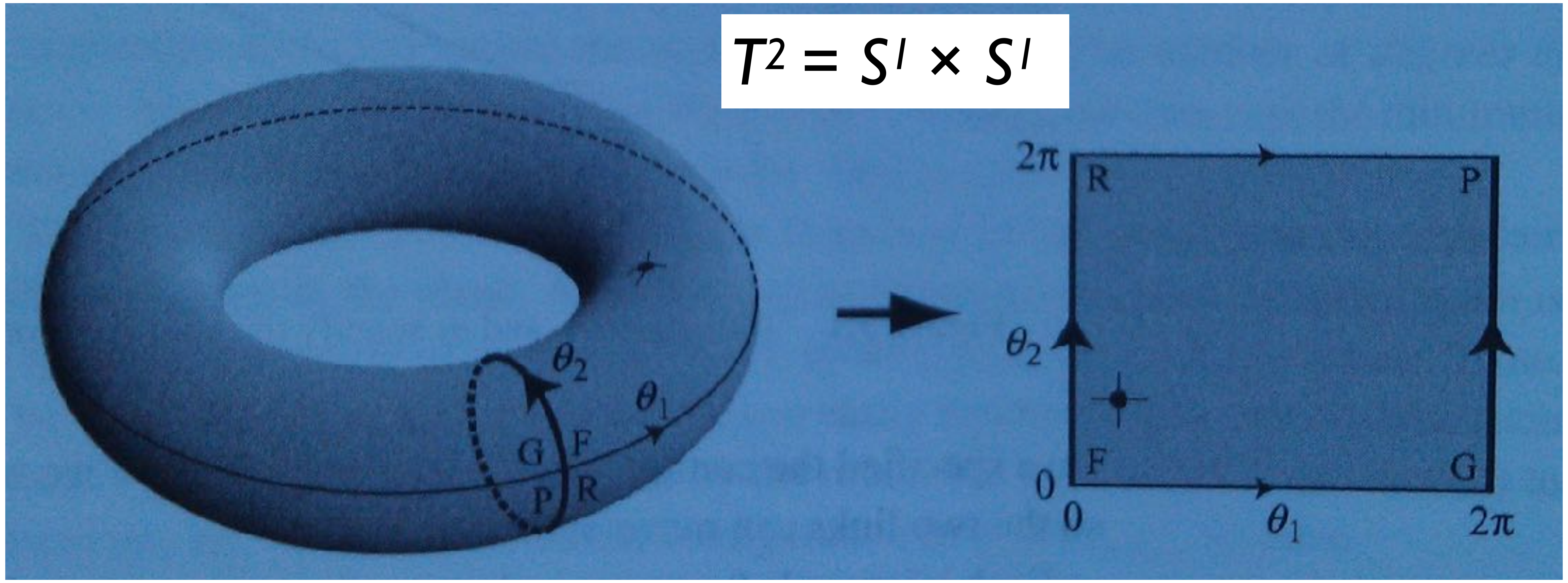
$S^1 \times S^1 = S^2$ when torus axis on surface



T^2 Torus Group



Space must fuse on each DOF where $2\pi = 0$



T^n is the torus group for an N-D rotational system

$$\mathbb{T}^n = \underbrace{S^1 \times S^1 \times \cdots \times S^1}_{n}$$

C-space examples

- What is the C-space of a Barrett WAM arm with 4 rotational joints, not including fingers of gripper?

[redacted]

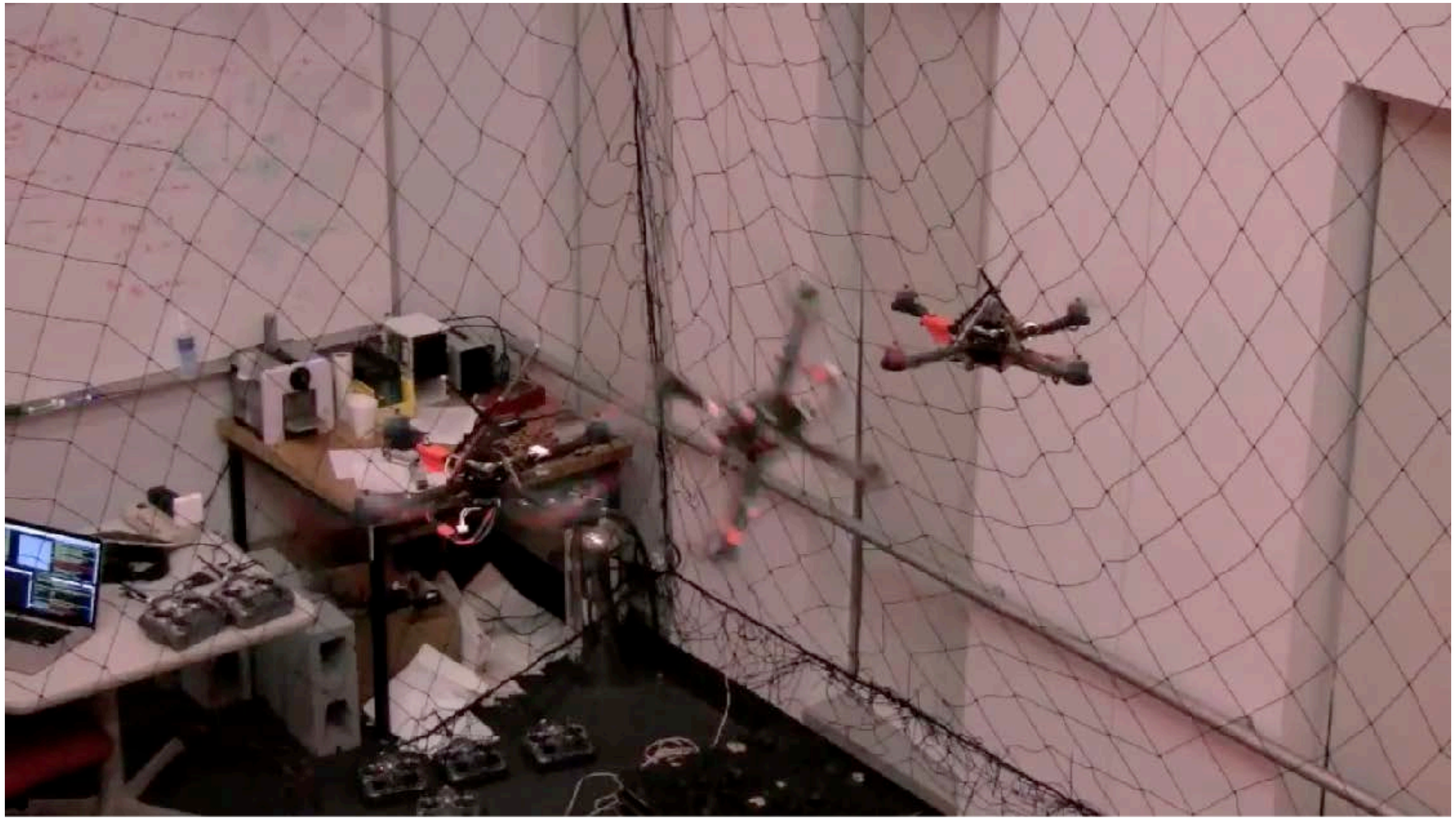
C-space examples

- What is the C-space of a Barrett WAM arm with 4 rotational joints, not including fingers of gripper?
  - DOFs: 4
  - C-space: T^4

Slide borrowed from Michigan Robotics autorob.org

C-space examples

- What is the C-space of a quad rotor helicopter?



V. Kumar et al. (2010) - UPenn - https://www.youtube.com/watch?v=MvRTALJp8DM

Drone Racing League (2017) https://www.youtube.com/watch?v=NpgGQCv642o

IROS 2017 Autonomous Drone Racing Competition

https://www.youtube.com/watch?v=y1DvYkPCnmM

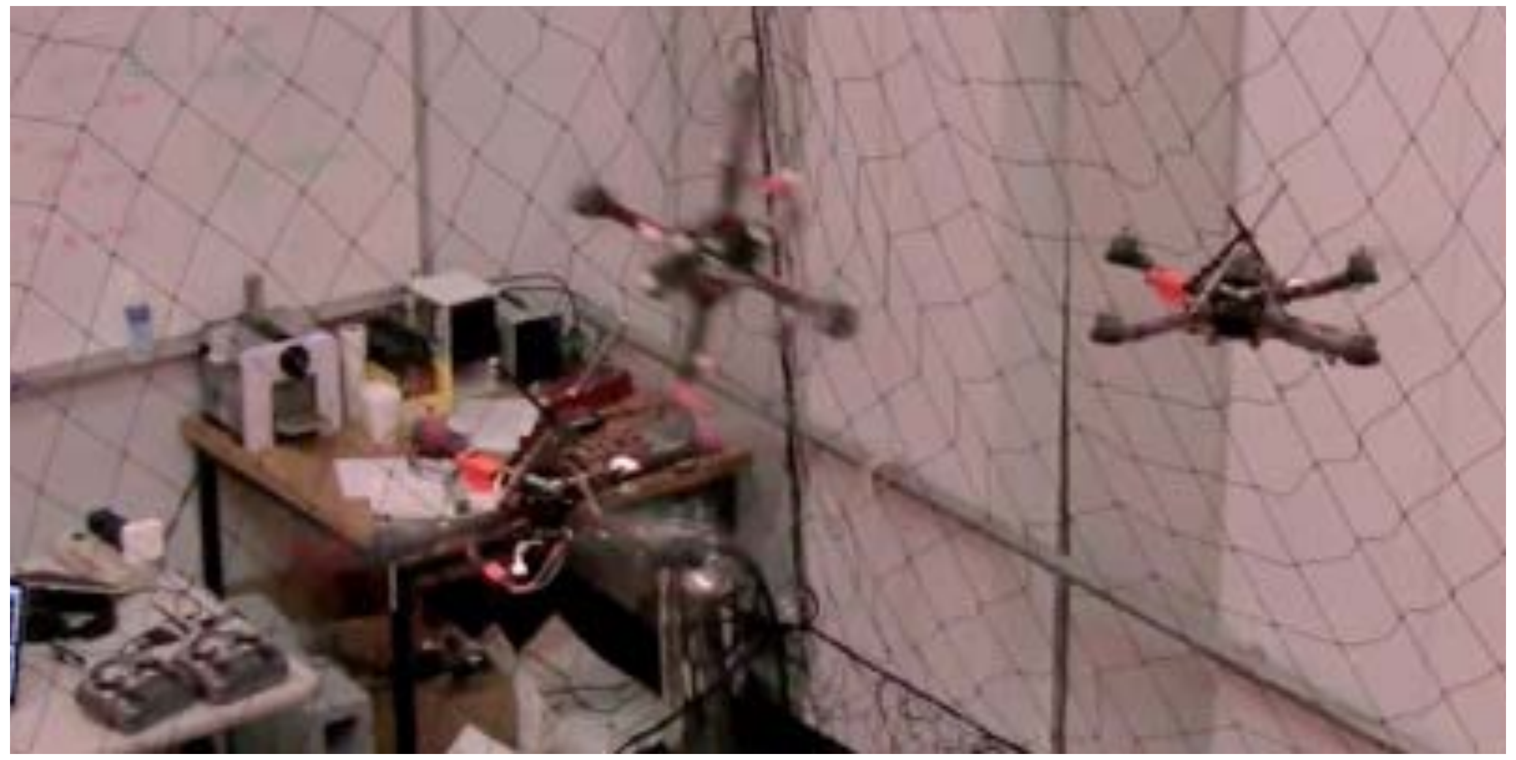
C-space examples

- What is the C-space of a quad rotor helicopter?
- DOFs: 6
- C-space: *SE(3)*,
  - or $\Re^3 \times SO(3)$

Group of homogeneous transformations in 3D

3D translation

3D rotation



SE(3) combines:
$\Re^3$: 3D translation and
SO(3): 3D rotation

$SO(3) = S^1 \times S^1 \times S^1$

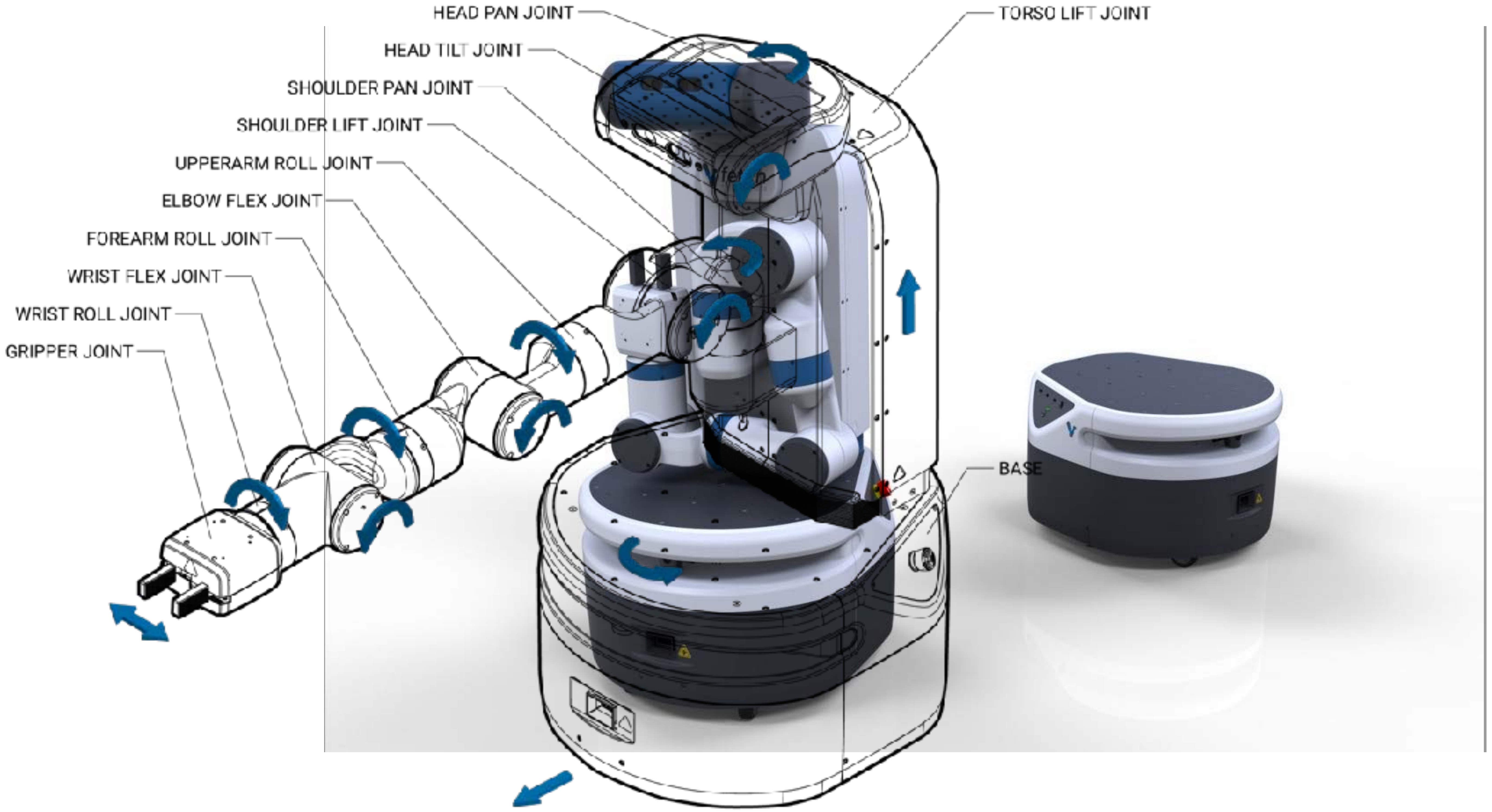
V. Kumar et al. - UPenn

C-space examples

- What is the C-space of a Fetch robot, not including grippers?

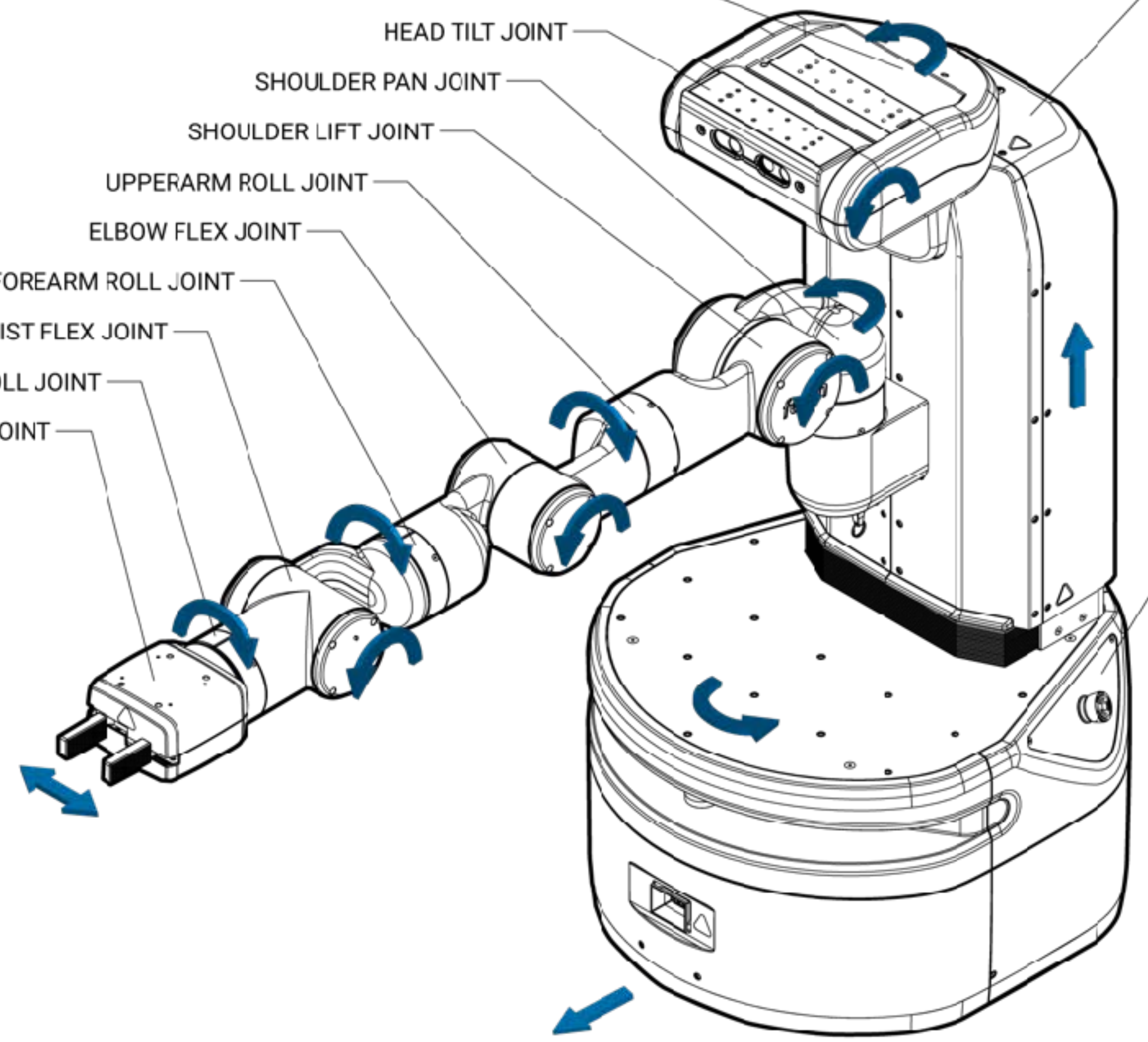
C-space examples

- What is the C-space of a Fetch robot, not including grippers?



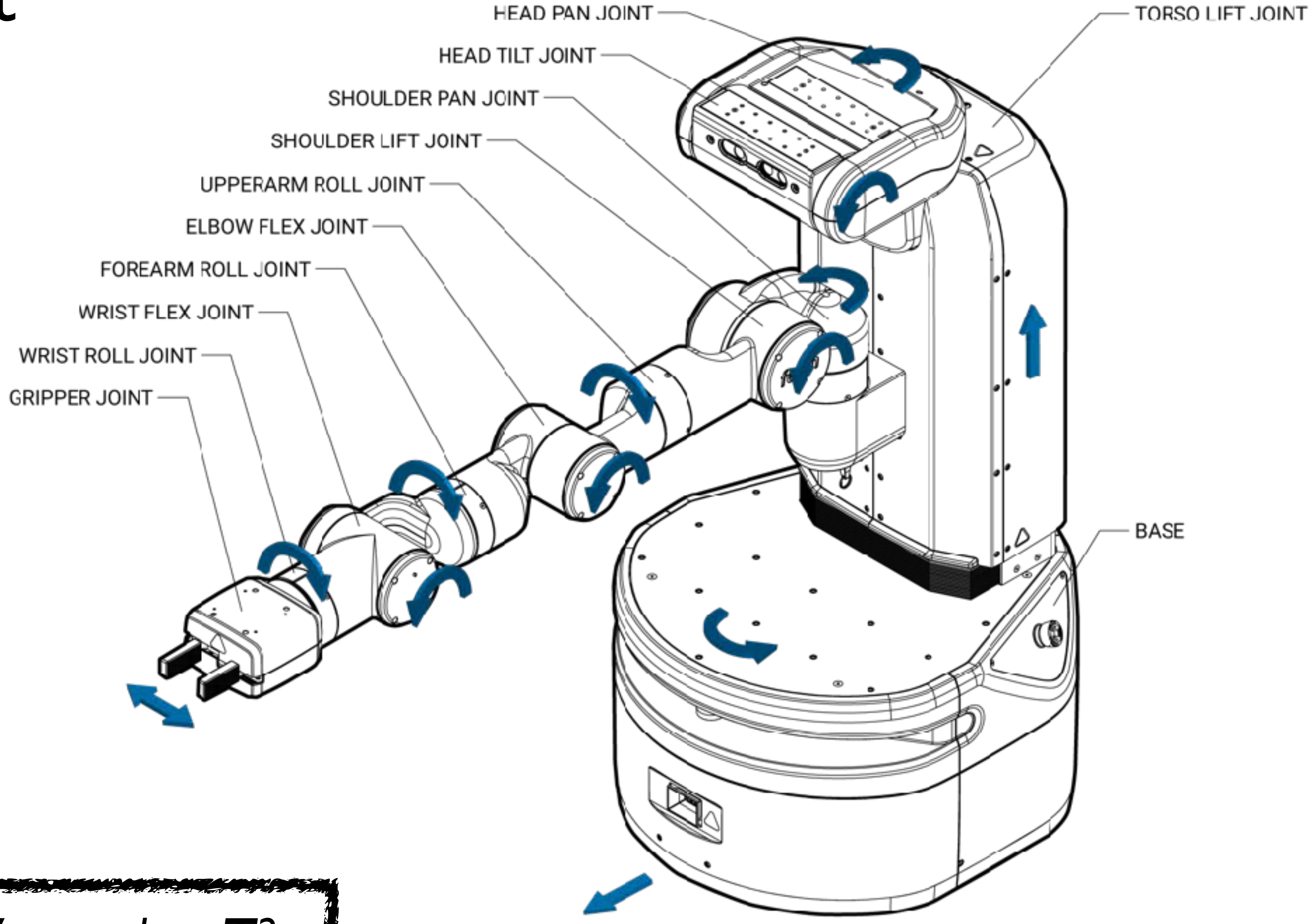
C-space examples

- What is the C-space of a Fetch robot, not including grippers?



C-space examples

- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: $SE(2)$
  - 7 in arm: T^7
  - 1 in the spine: $\Re^1$
  - 2 in neck: T^2

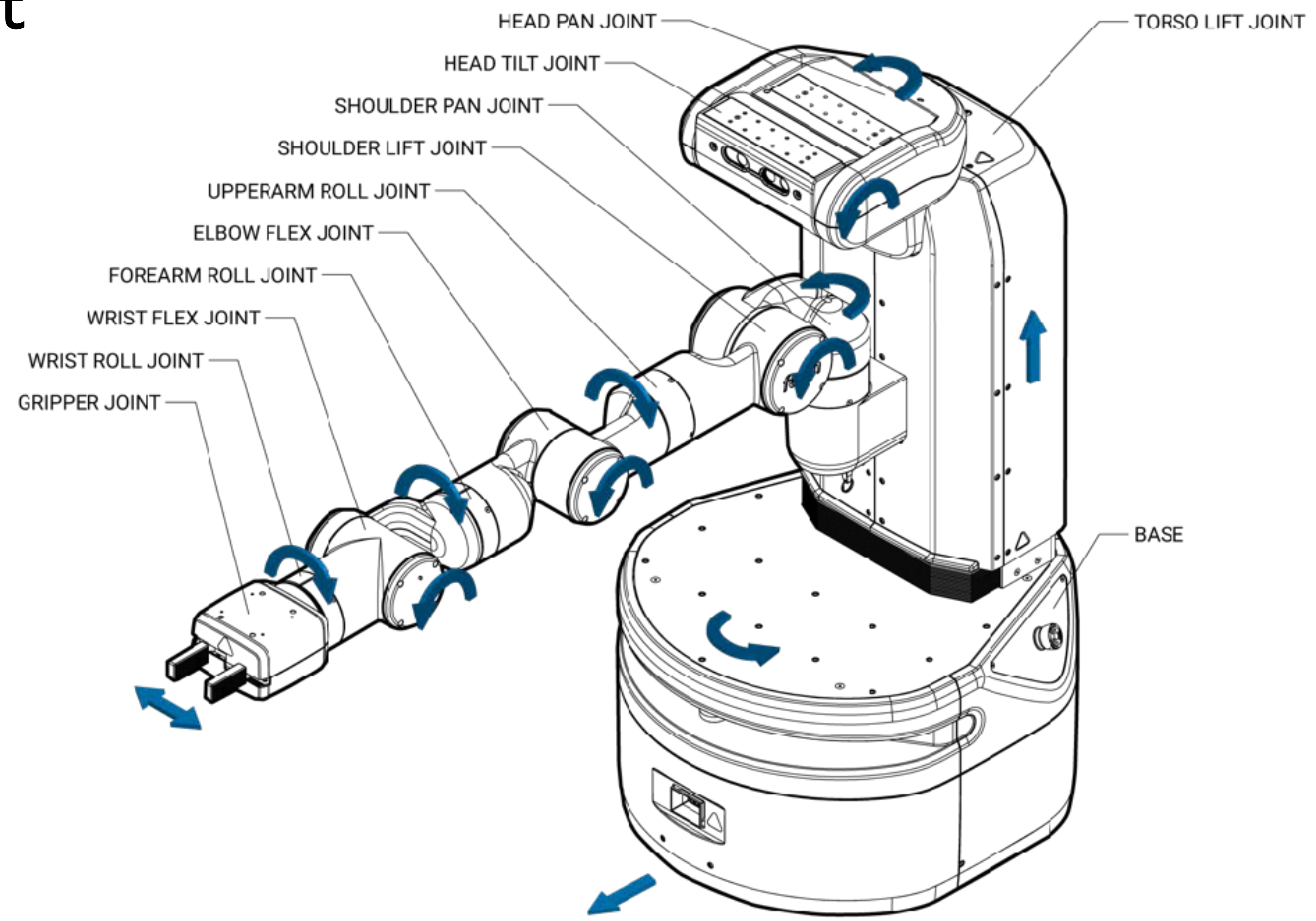


C-space: $SE(2) \times T^7 \times \Re^1 \times T^2$

Did we get this wrong?

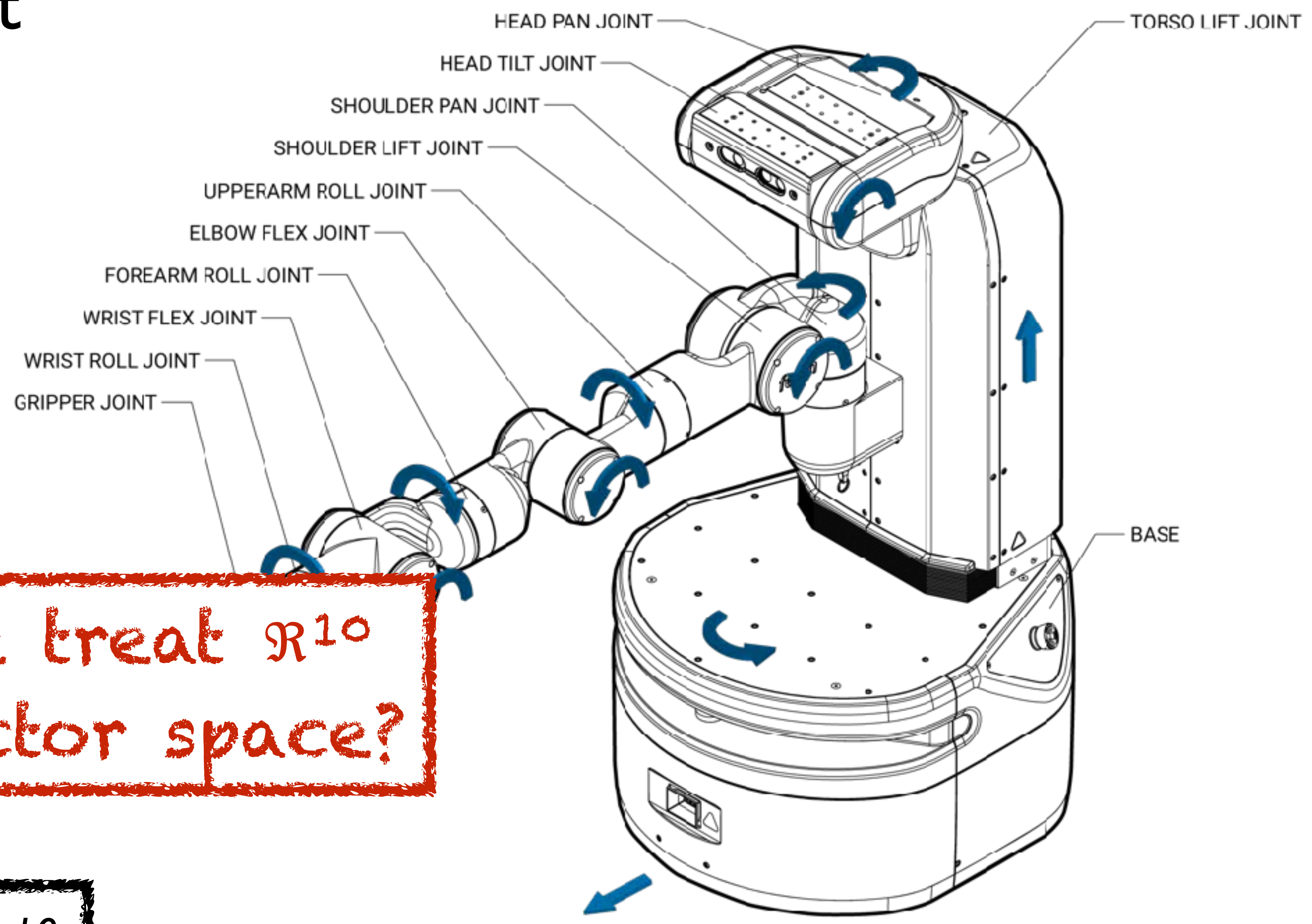
- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: $SE(2)$
  - 7 in arm: $\cancel{T^7}$
  - 1 in the spine: $\Re^1$
  - 2 in neck: $\cancel{T^2}$

Consider joint limits



C-space with joint limits

- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: $SE(2)$
  - 7 in arm: $\Re^7$
  - 1 in the spine: $\Re^1$
  - 2 in neck: $\Re^2$

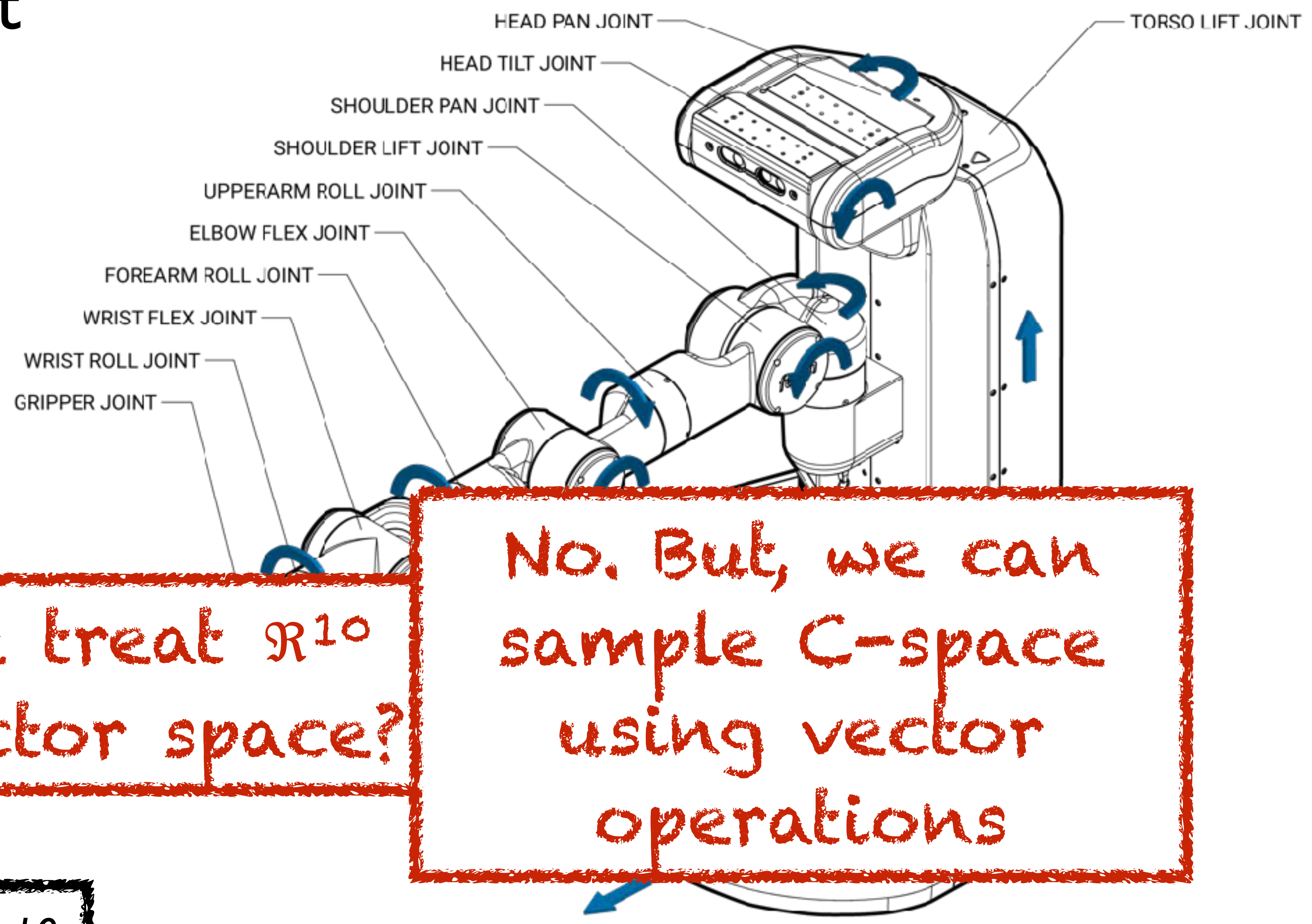


Can we treat $\Re^{10}$ as a vector space?

C-space: $SE(2) \times \Re^{10}$

C-space with joint limits

- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: *SE(2)*
  - 7 in arm: $\Re^7$
  - 1 in the spine: $\Re^1$
  - 2 in neck: $\Re^2$



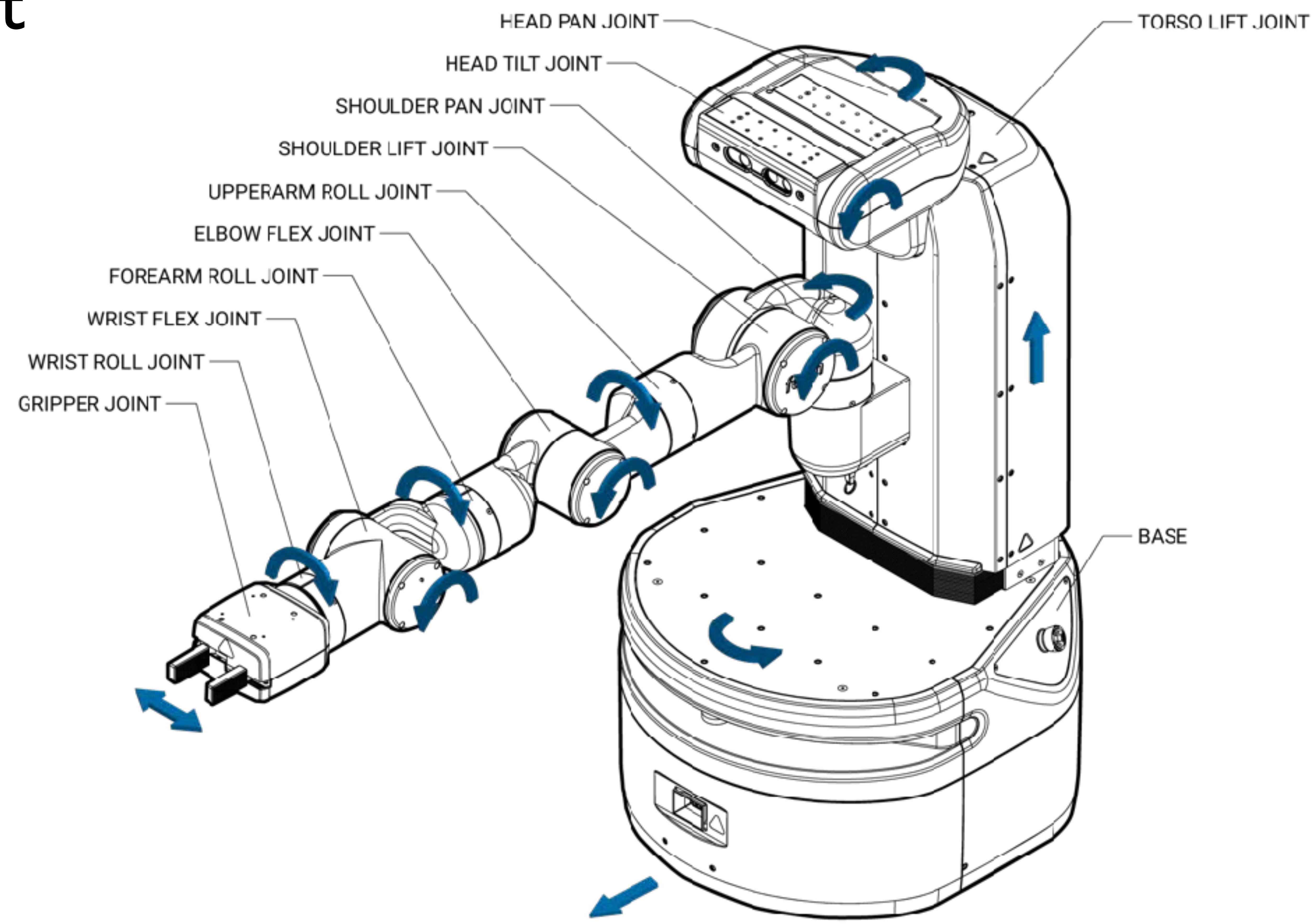
Can we treat $\Re^{10}$ as a vector space?

No. But, we can sample C-space using vector operations

C-space: $SE(2) \times \Re^{10}$

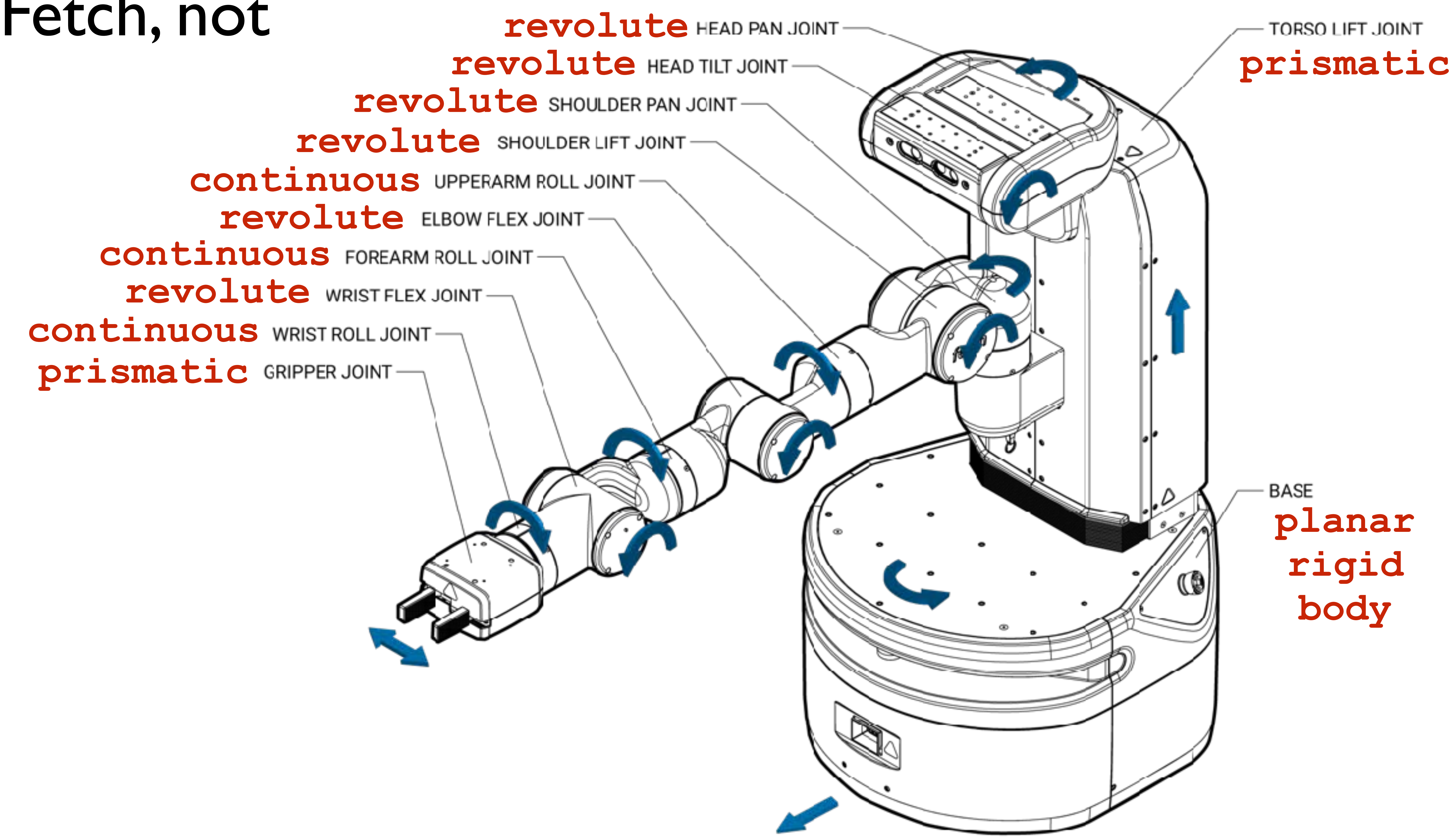
Still not quite right…

- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: *SE(2)*
  - 7 in arm: $\cancel{\Re^7}$
  - 1 in the spine: $\Re^1$
  - 2 in neck: $\Re^2$

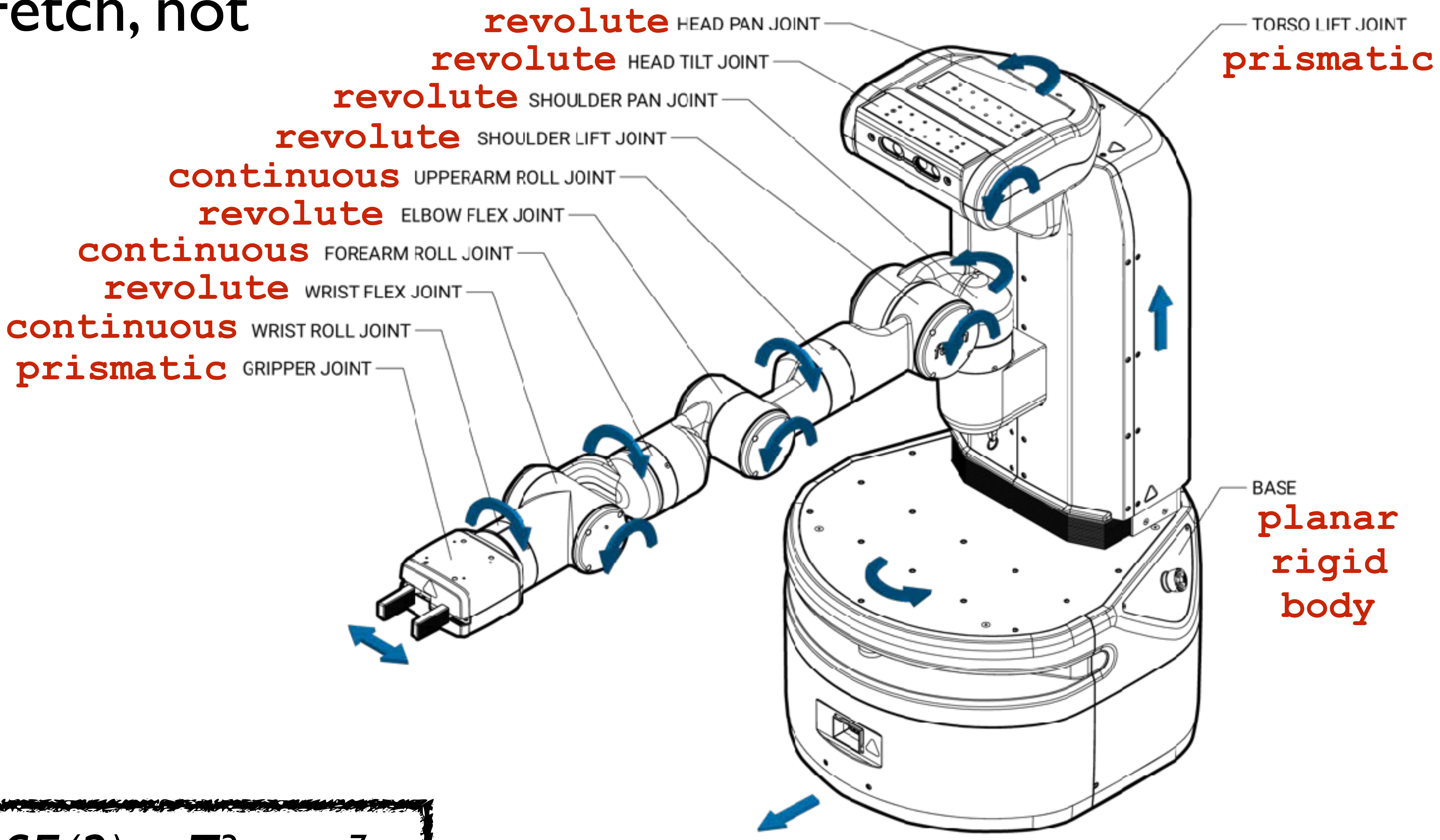


Still not quite right…

- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: *SE(2)*
  - 7 in arm: $\cancel{\Re^7}$
  - 1 in the spine: $\Re^1$
  - 2 in neck: $\Re^2$



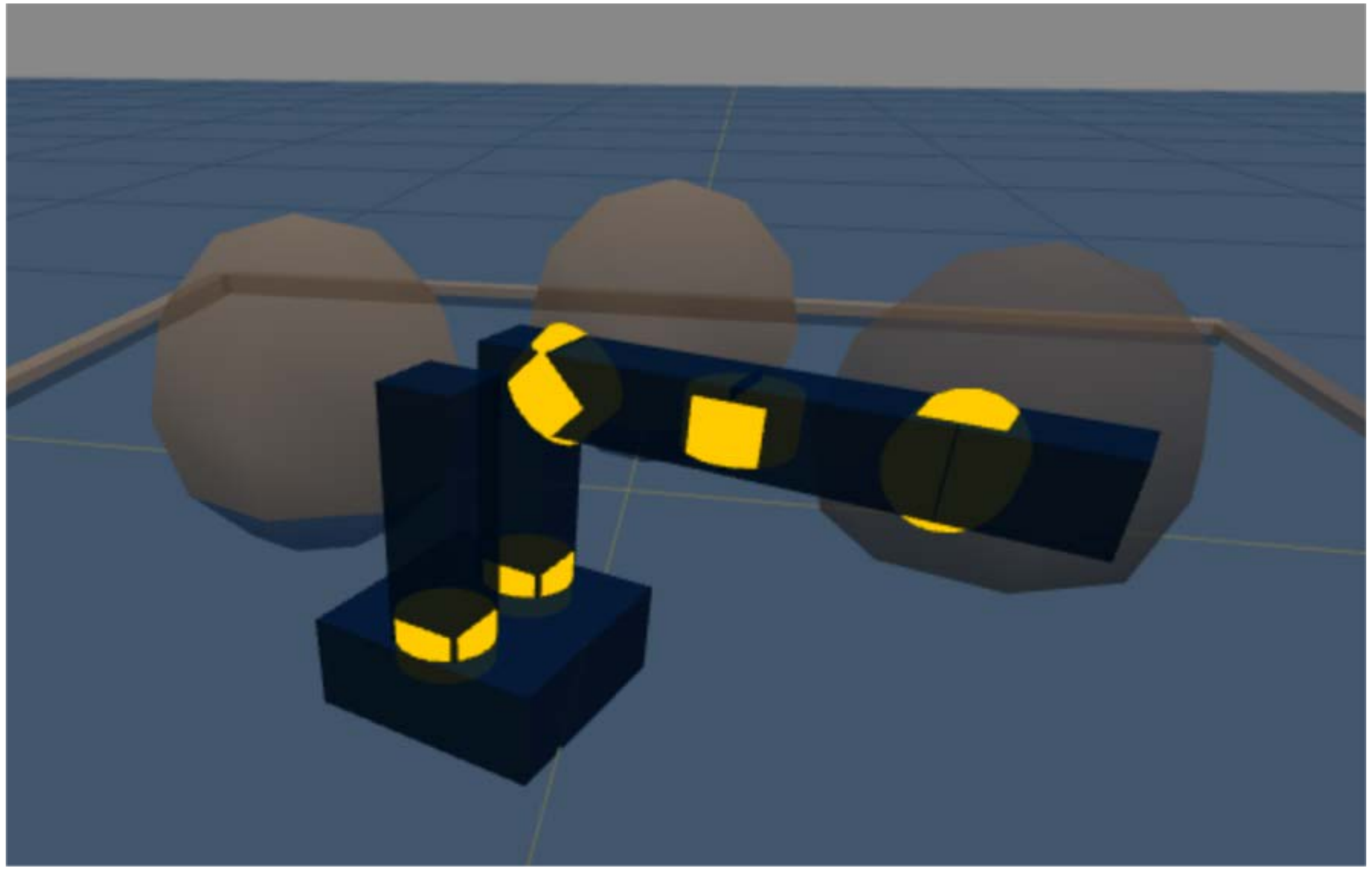
- What is the C-space of a Fetch, not including grippers?
- DOFs: 13
  - 3 in base: $SE(2)$
  - 3 continuous: T^3
  - 1 prismatic: $\Re^1$
  - 6 revolute: $\Re^6$



C-space: $SE(2) \times T^3 \times \Re^7$

C-space examples

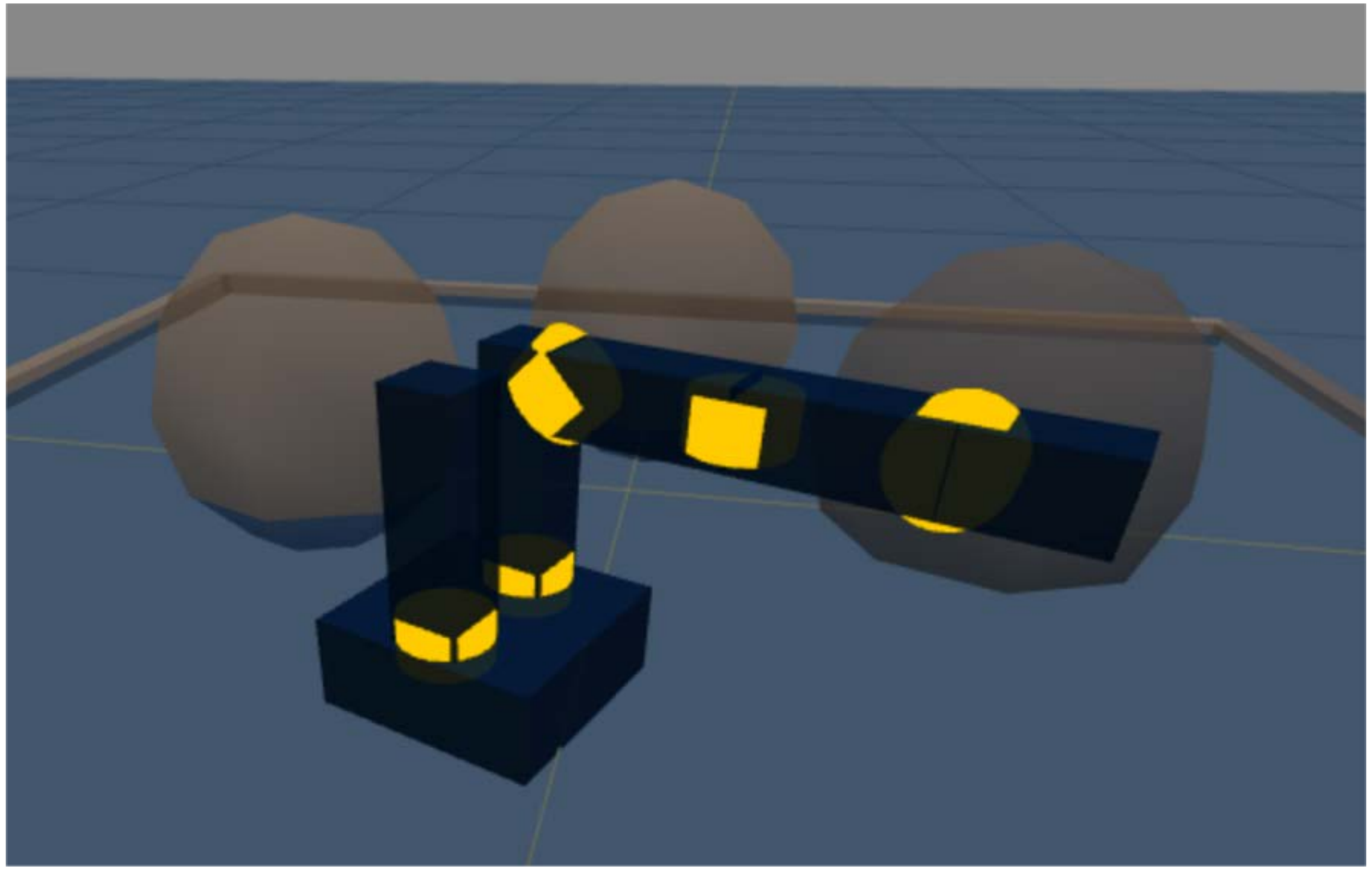
- What is the C-space of a MR2?



C-space examples

- What is the C-space of a MR2?
- DOFs: 14
  - 3 in base: $SE(2)$
  - 5 in arms: T^5

$\boxed{\text{C-space: } SE(2) \times T^5}$



C-space examples

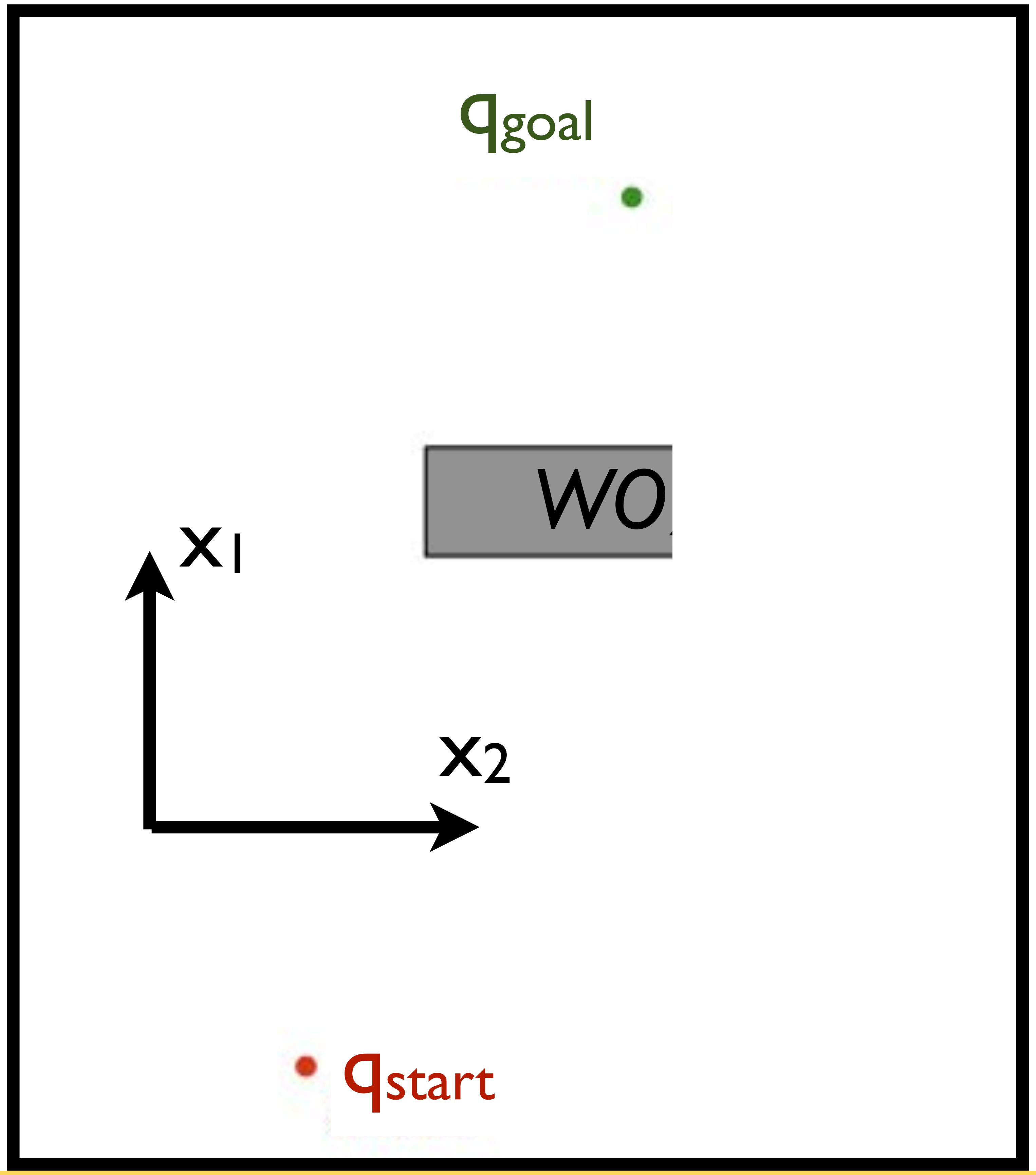
- What is the C-space of a Robonaut 2 on the International Space Station?

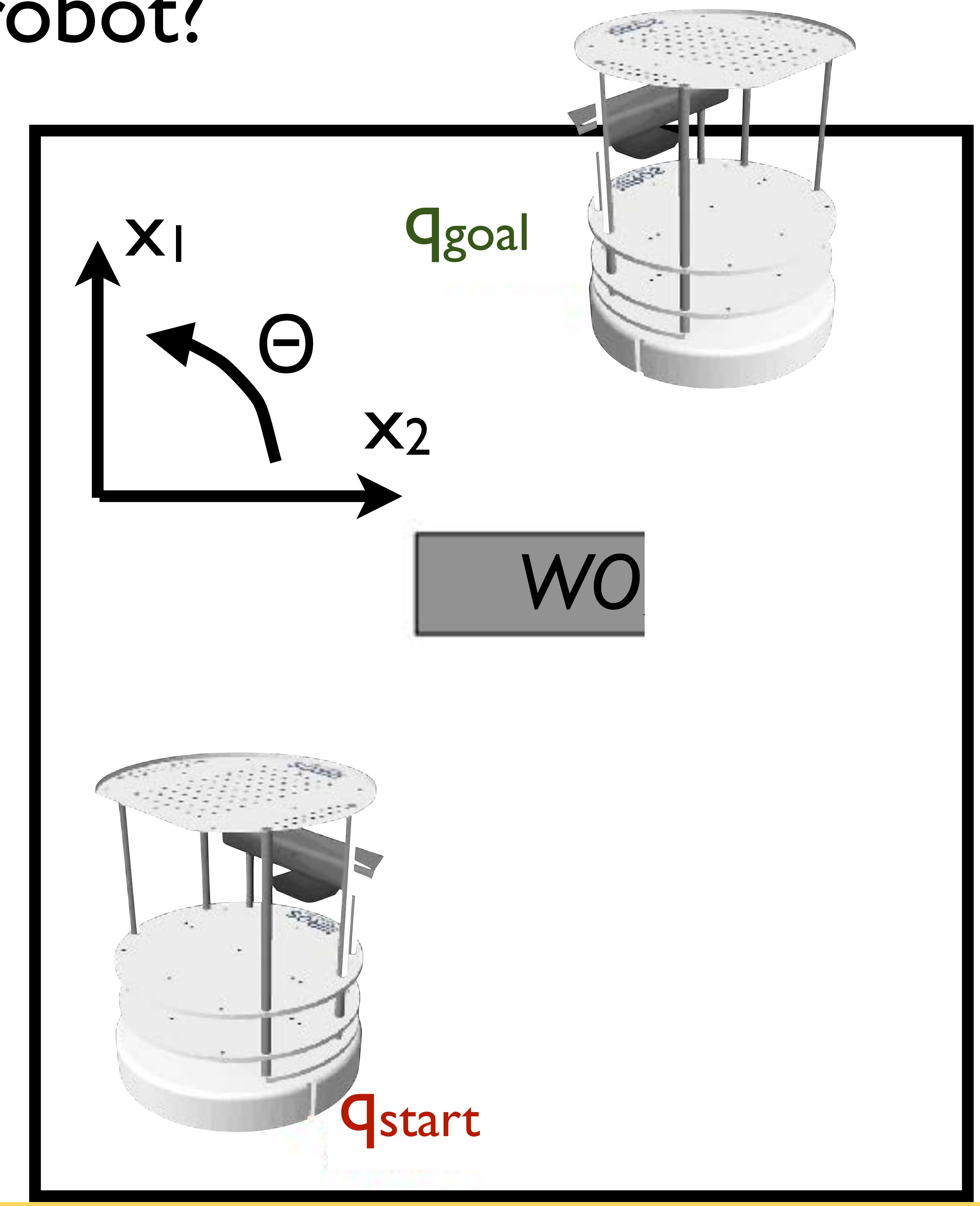
- What is the C-space of a PR2?

What about the robot's physical geometry?

Configuration v. Workspaces

- Other than rotation, how is the Turtlebot different than the point robot?

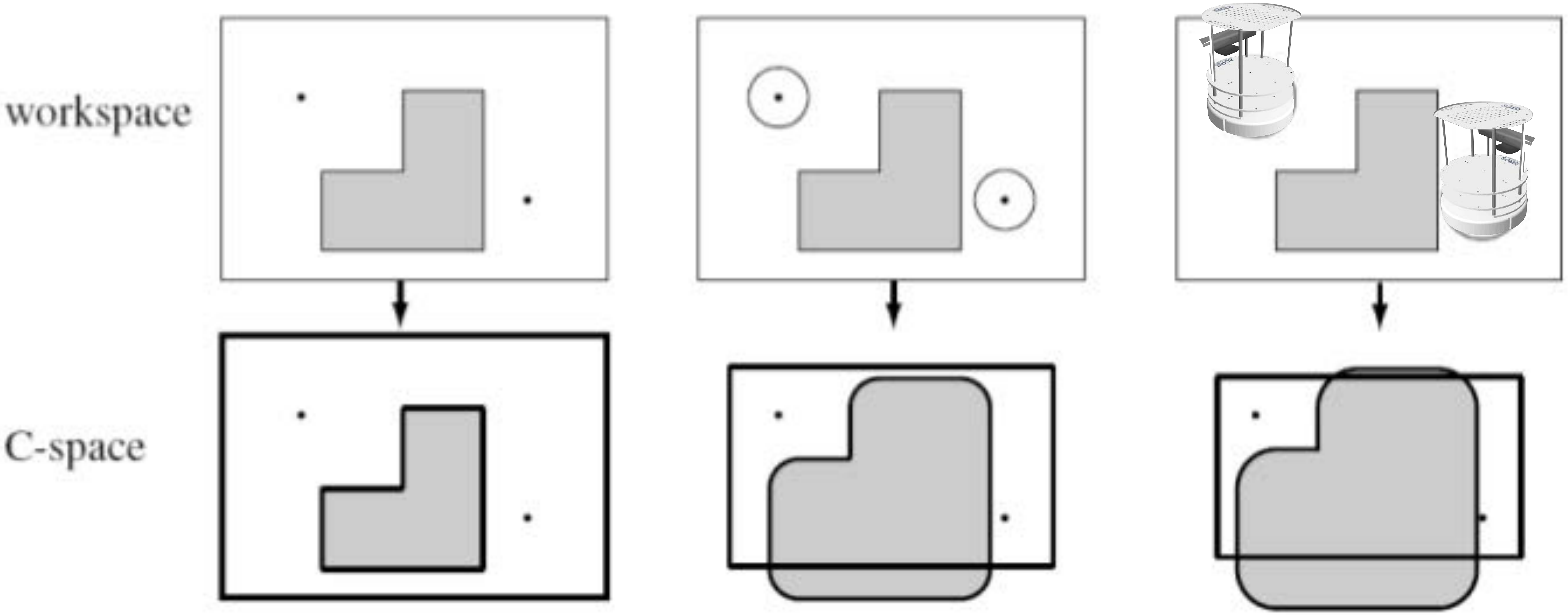




Slide borrowed from Michigan Robotics autorob.org

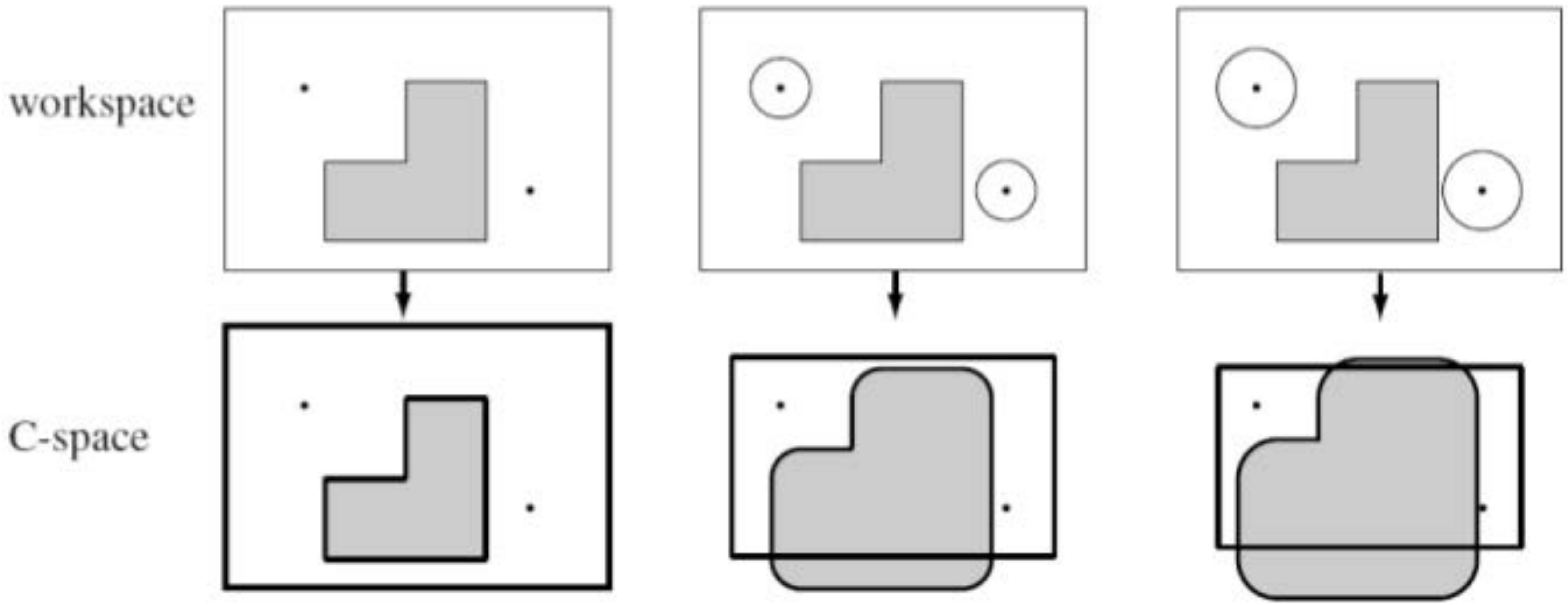
Robot Geometry

- Turtlebot is larger than a point, having a circular radius in the robot's planar workspace
- As this radius increases, the C-space shrinks

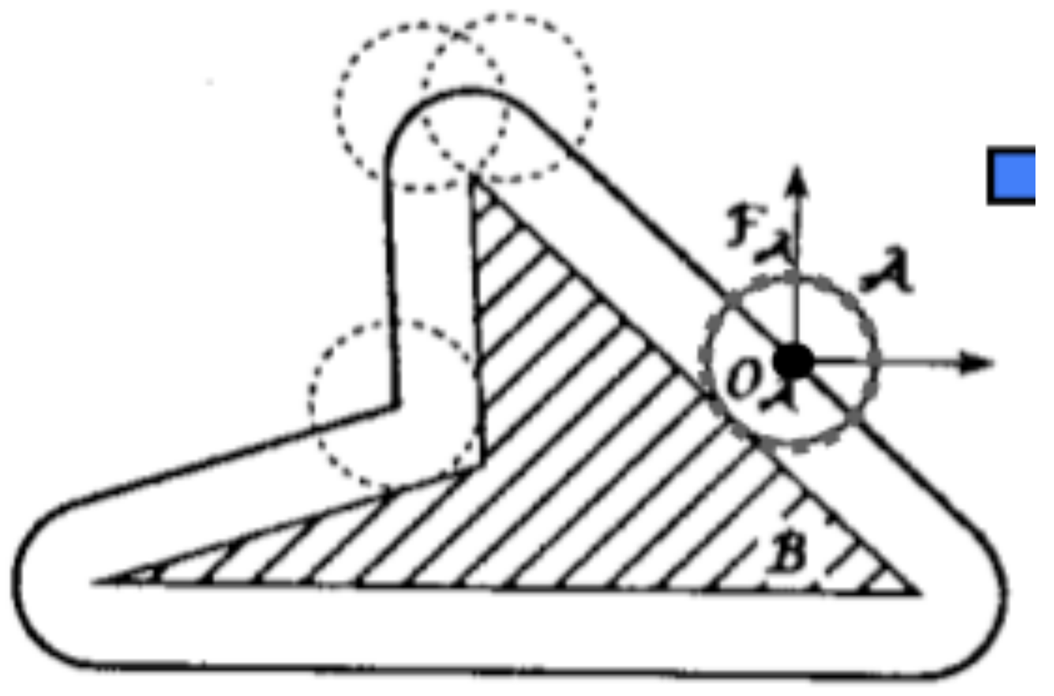


Robot Geometry

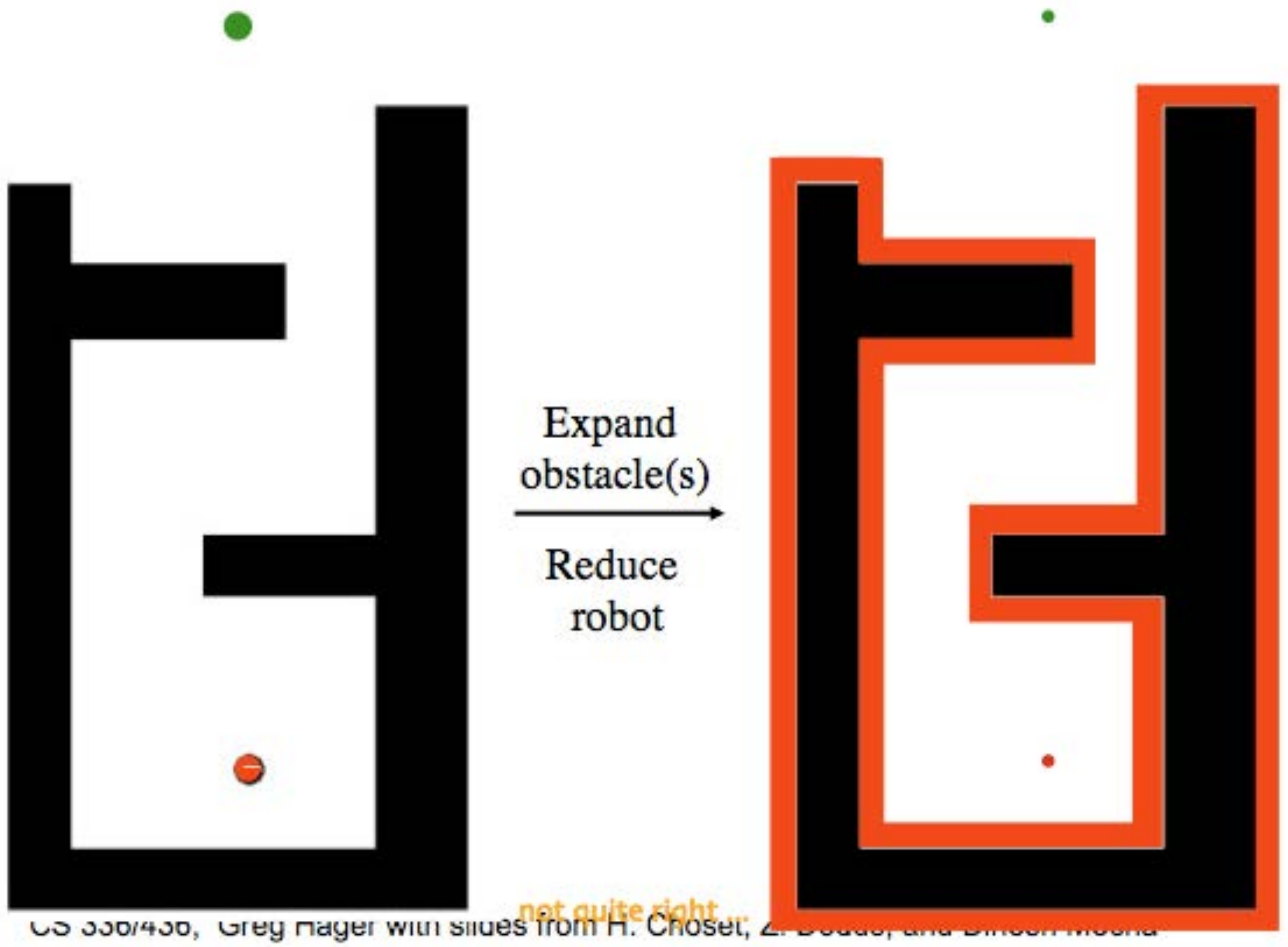
- Turtlebot is larger than a point, having a circular radius in the robot's planar workspace
- As this radius increases, the C-space shrinks



Conversion to point robot C-space

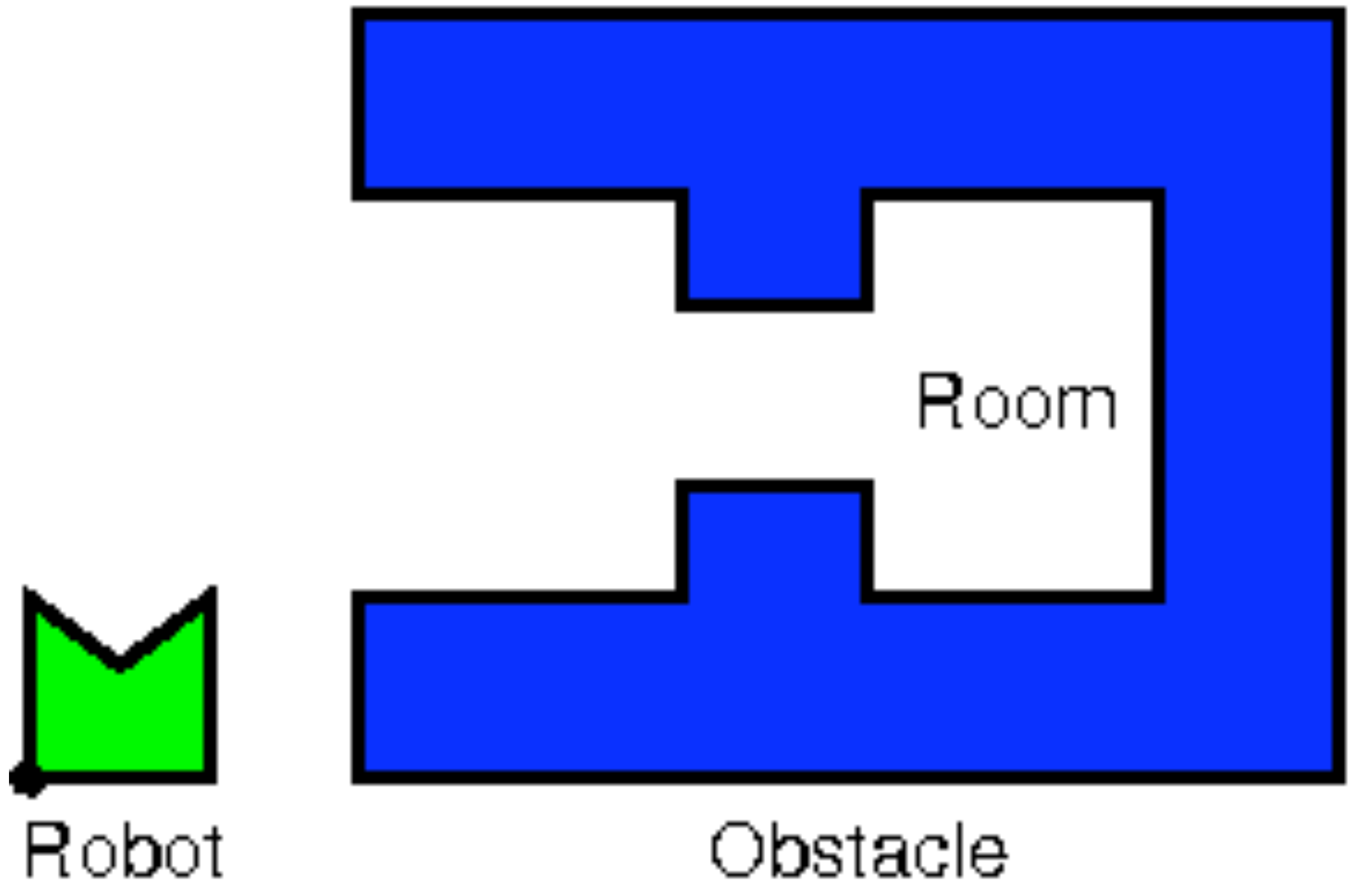


- Workspace for robot can be converted to point robot C-space
- Expand obstacles by tracing robot geometry along boundary
- Computable by Minkowski sum

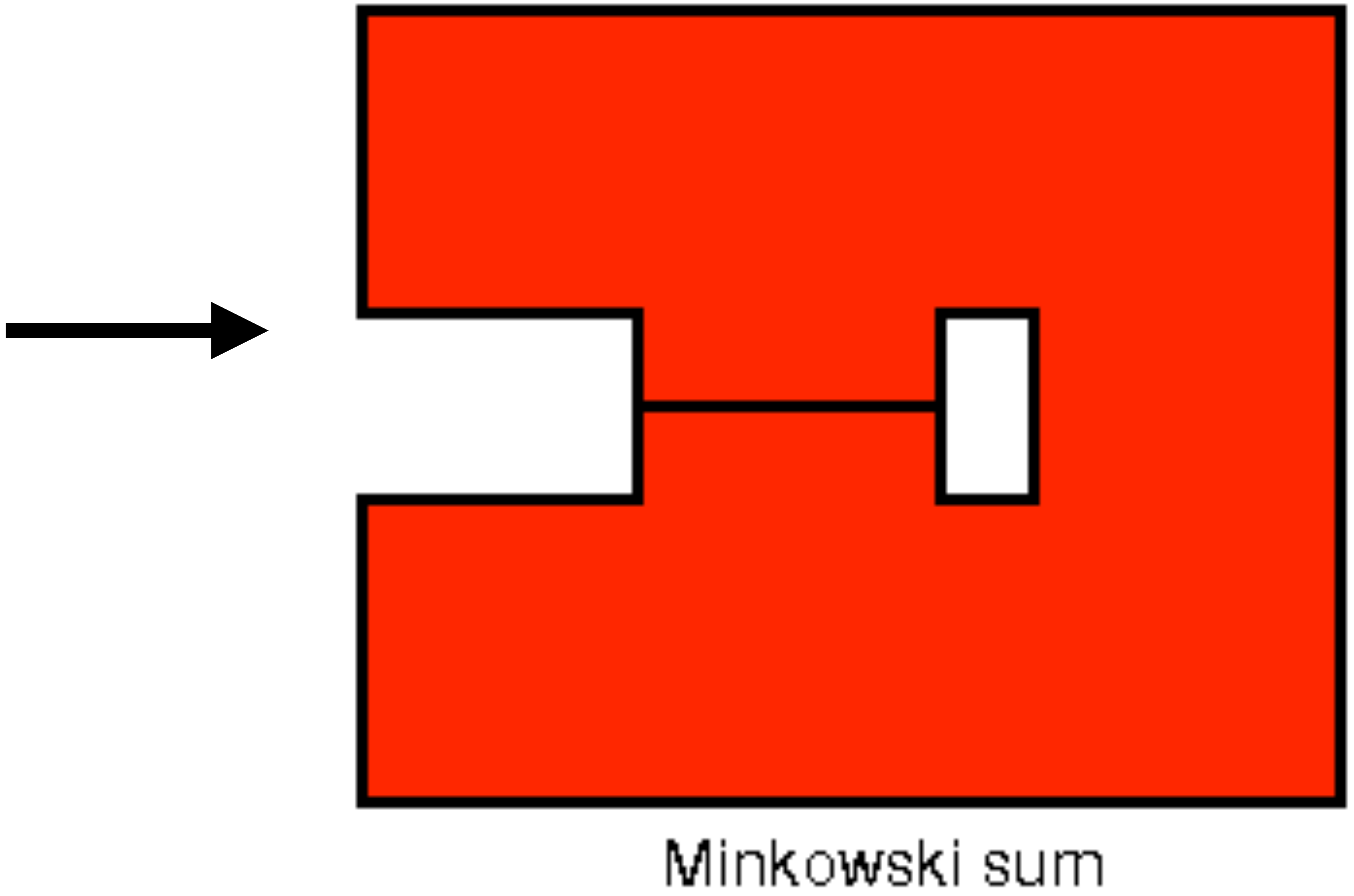


Minkowski Planning

Given

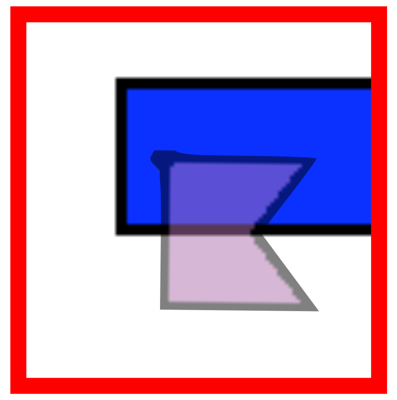


Compute

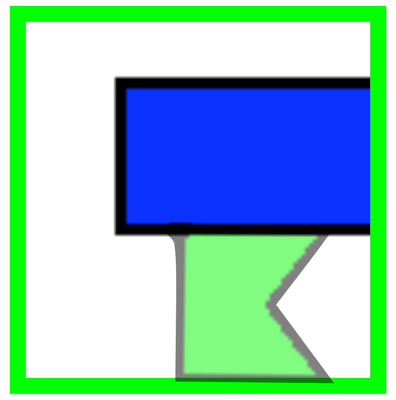


Slide borrowed from Michigan Robotics autorob.org

Minkowski Planning

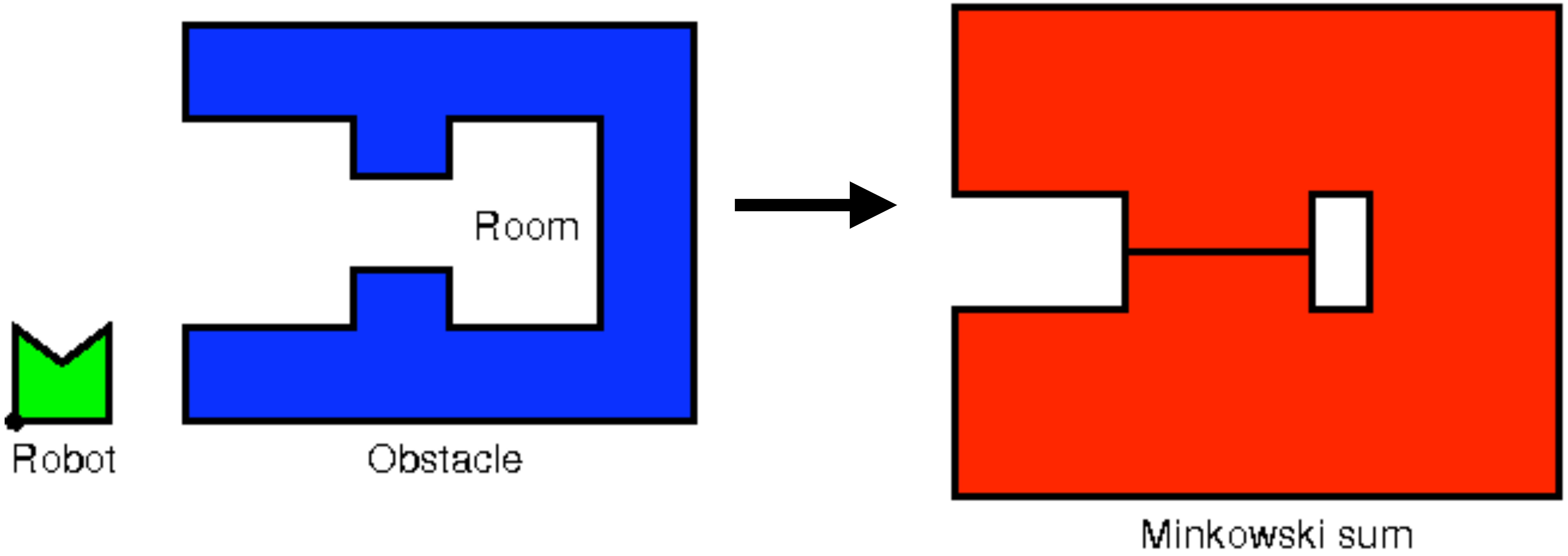


Occlude location if robot intersects obstacle



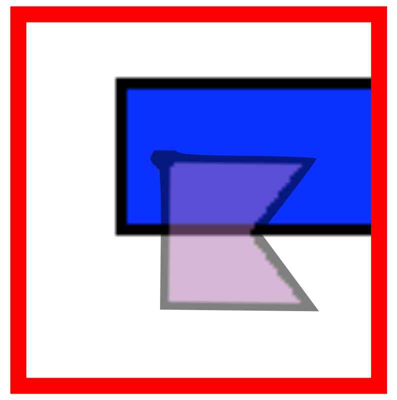
Leave location free if robot non-intersecting with object

Minkowski sum: identify non intersecting robot locations

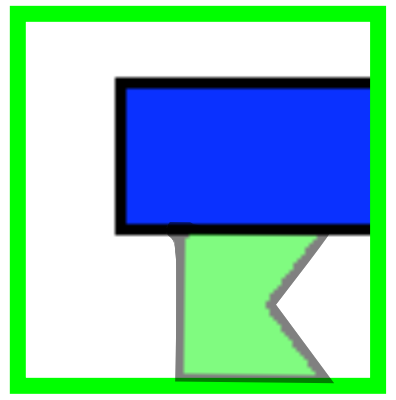


Slide borrowed from Michigan Robotics autorob.org

Minkowski Planning



Occlude location if robot intersects obstacle

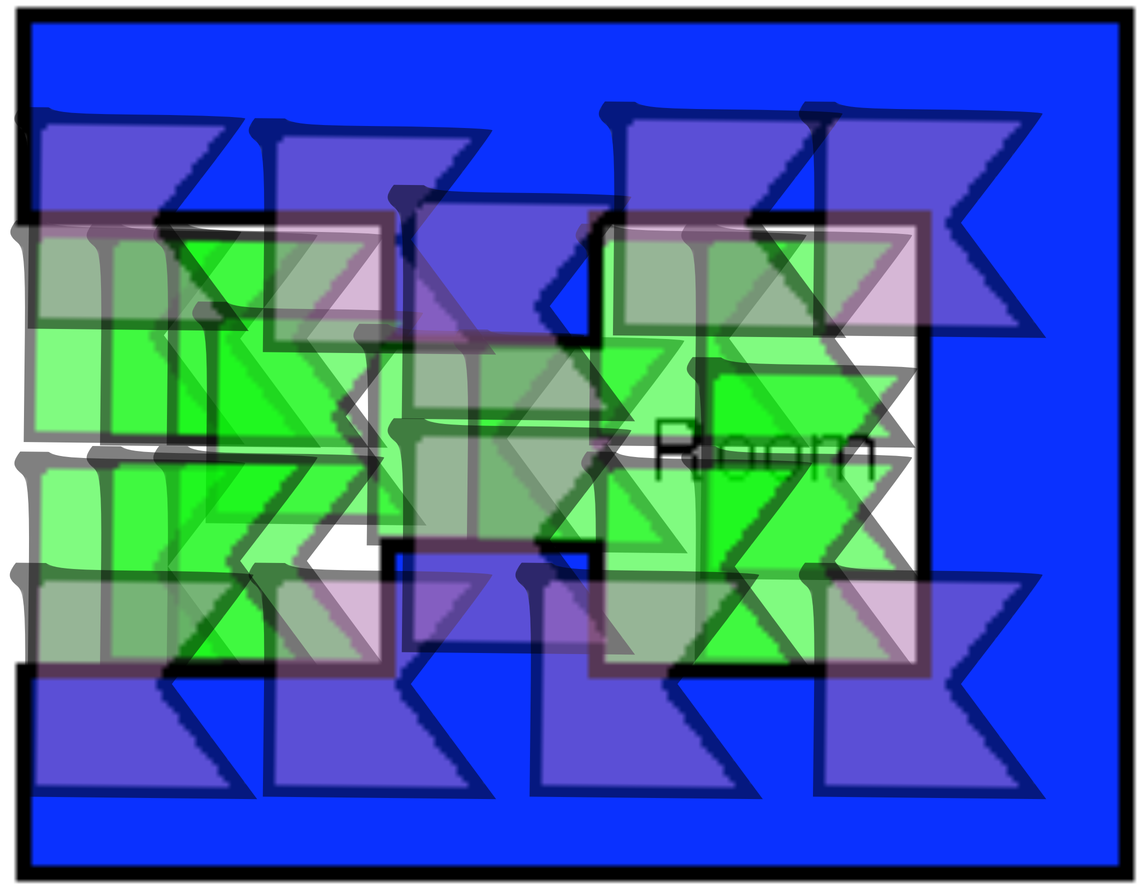


Leave location free if robot non-intersecting with object

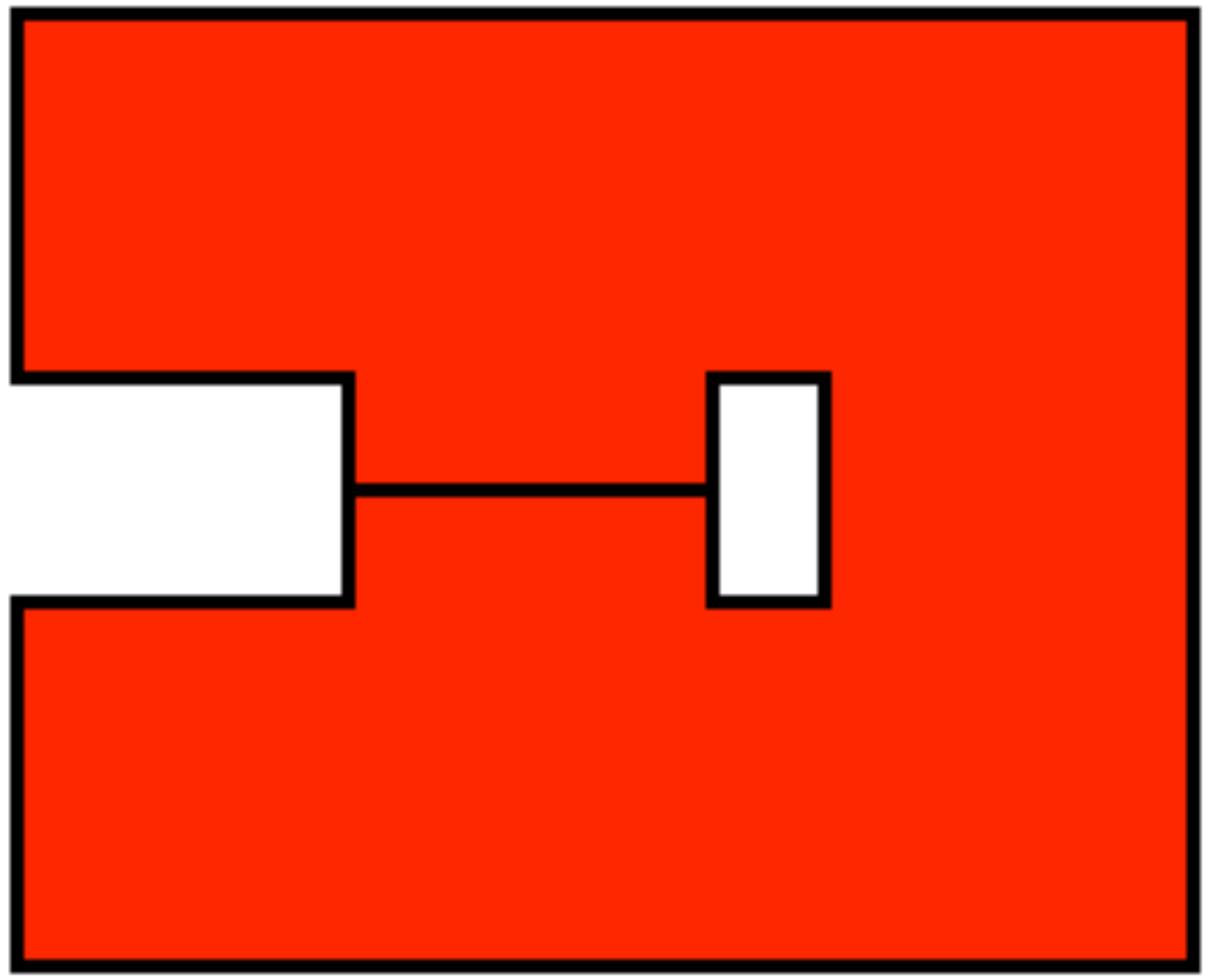
Minkowski sum: identify non intersecting robot locations

Robot geometry

$\oplus$

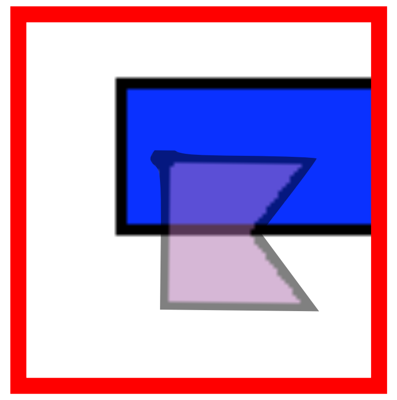


=

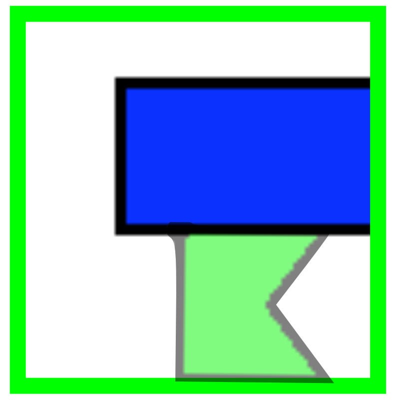


Space of valid paths defined by Minkowski sum

Minkowski Planning



Occlude location if robot intersects obstacle

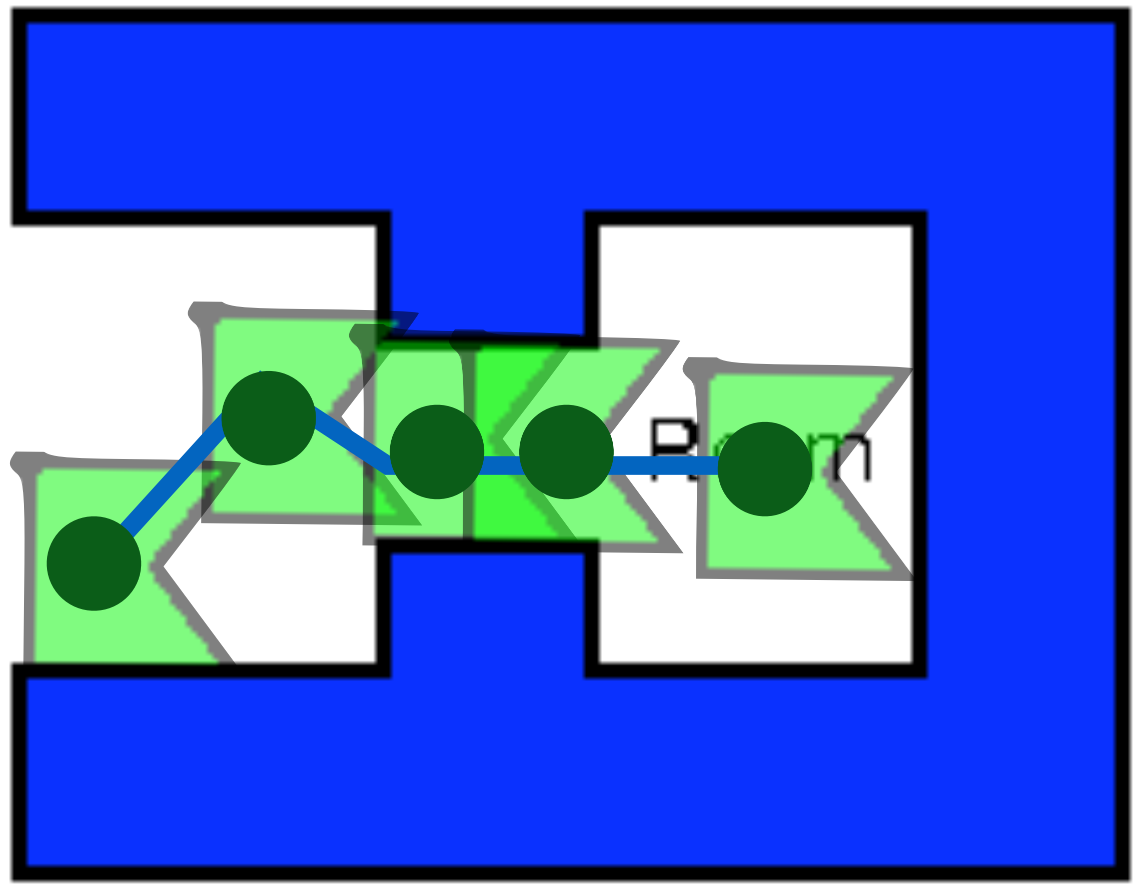


Leave location free if robot non-intersecting with object

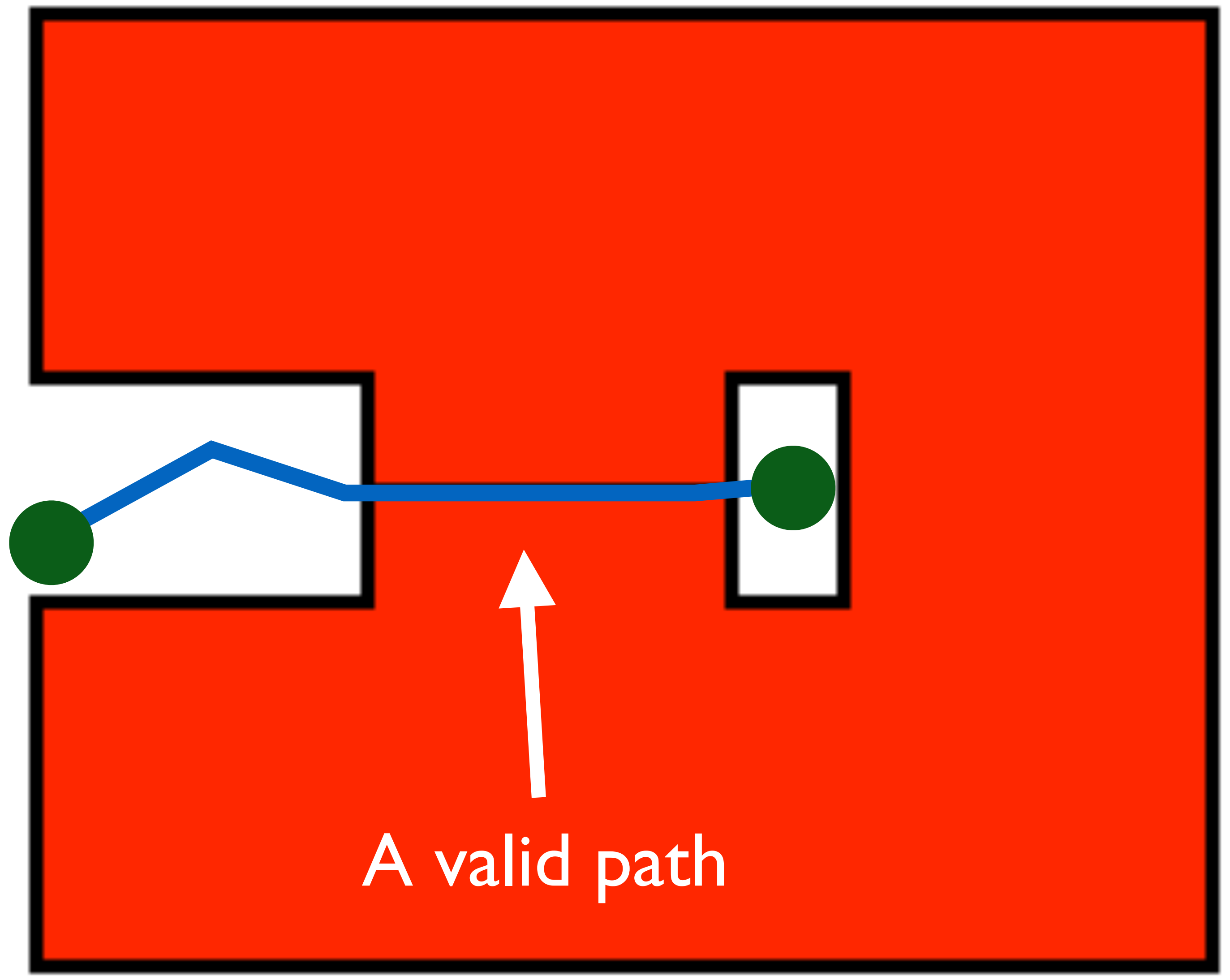
Minkowski sum: identify non intersecting robot locations

Robot geometry

$\oplus$



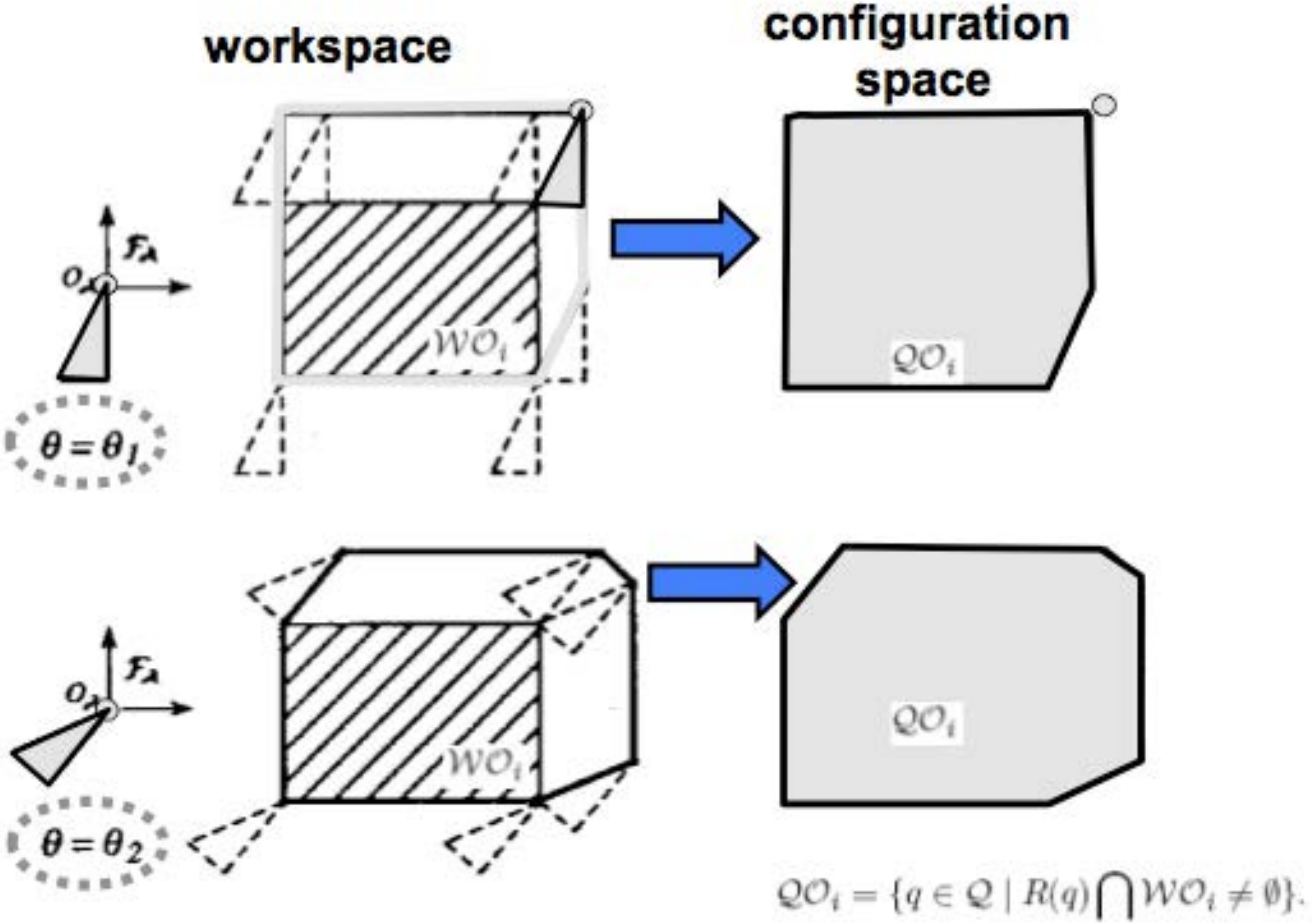
=



Space of valid paths defined by Minkowski sum

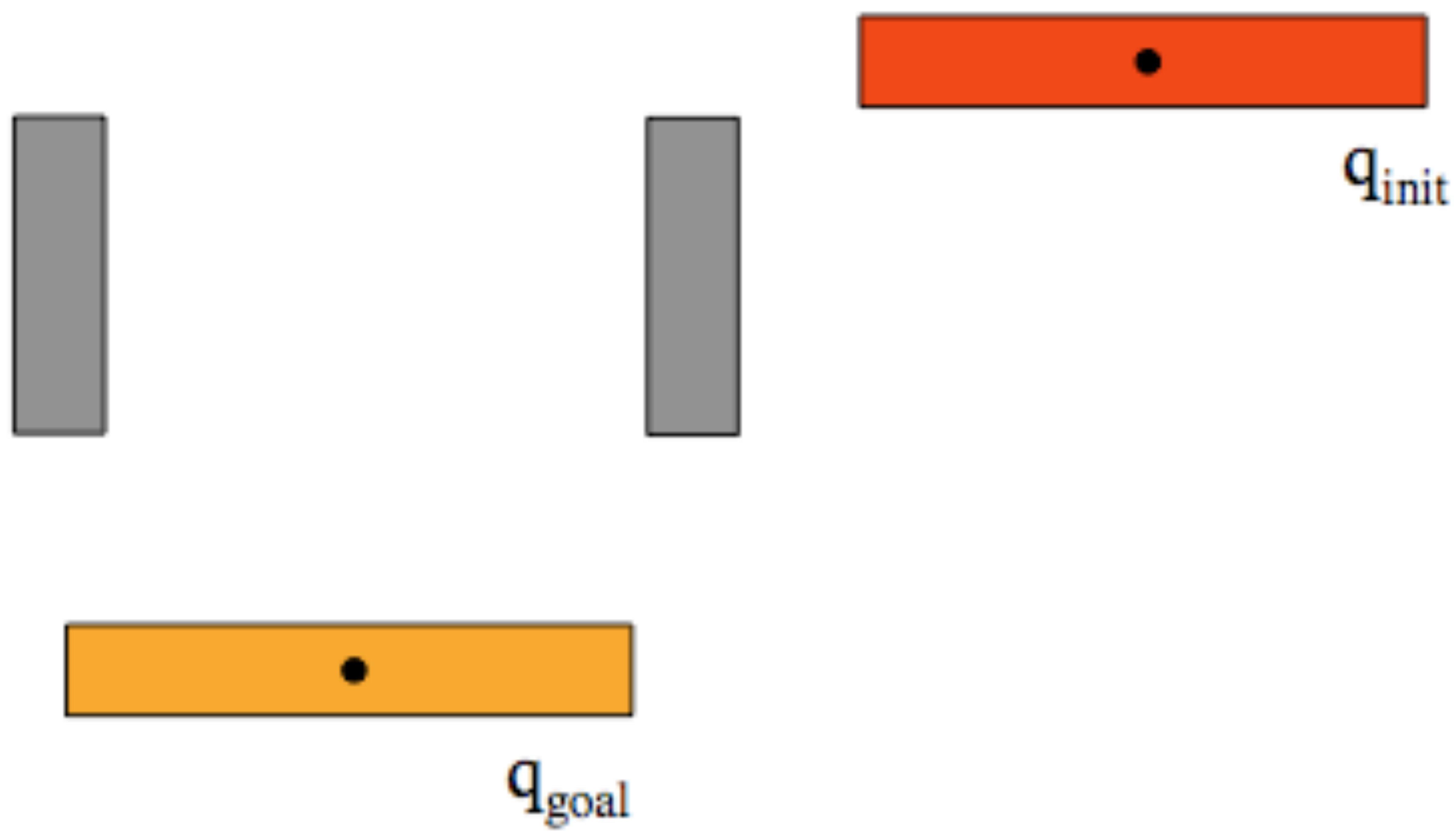
What does an obstacle look like in configuration space?

C-space depends on rotation

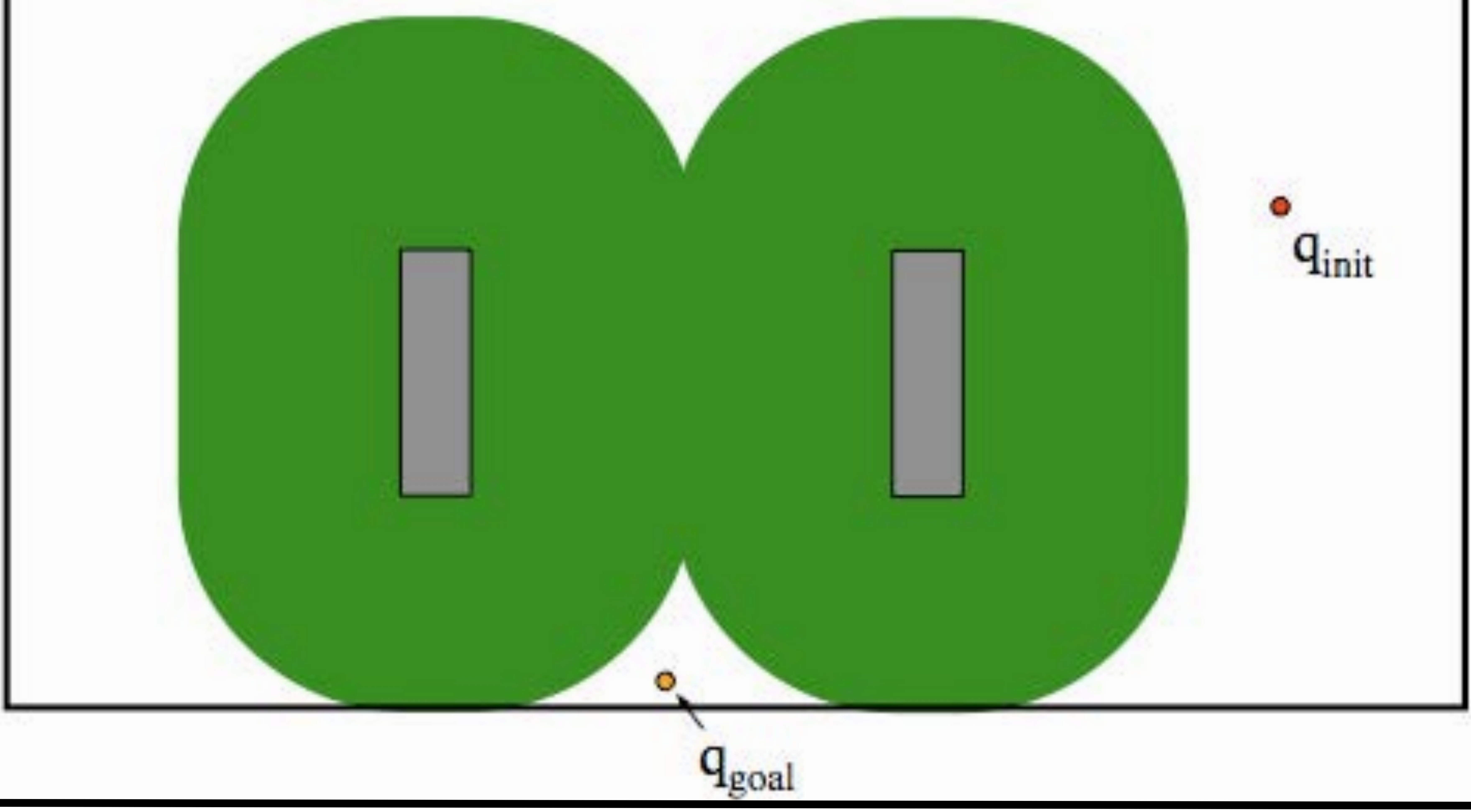


Slide borrowed from Michigan Robotics autorob.org

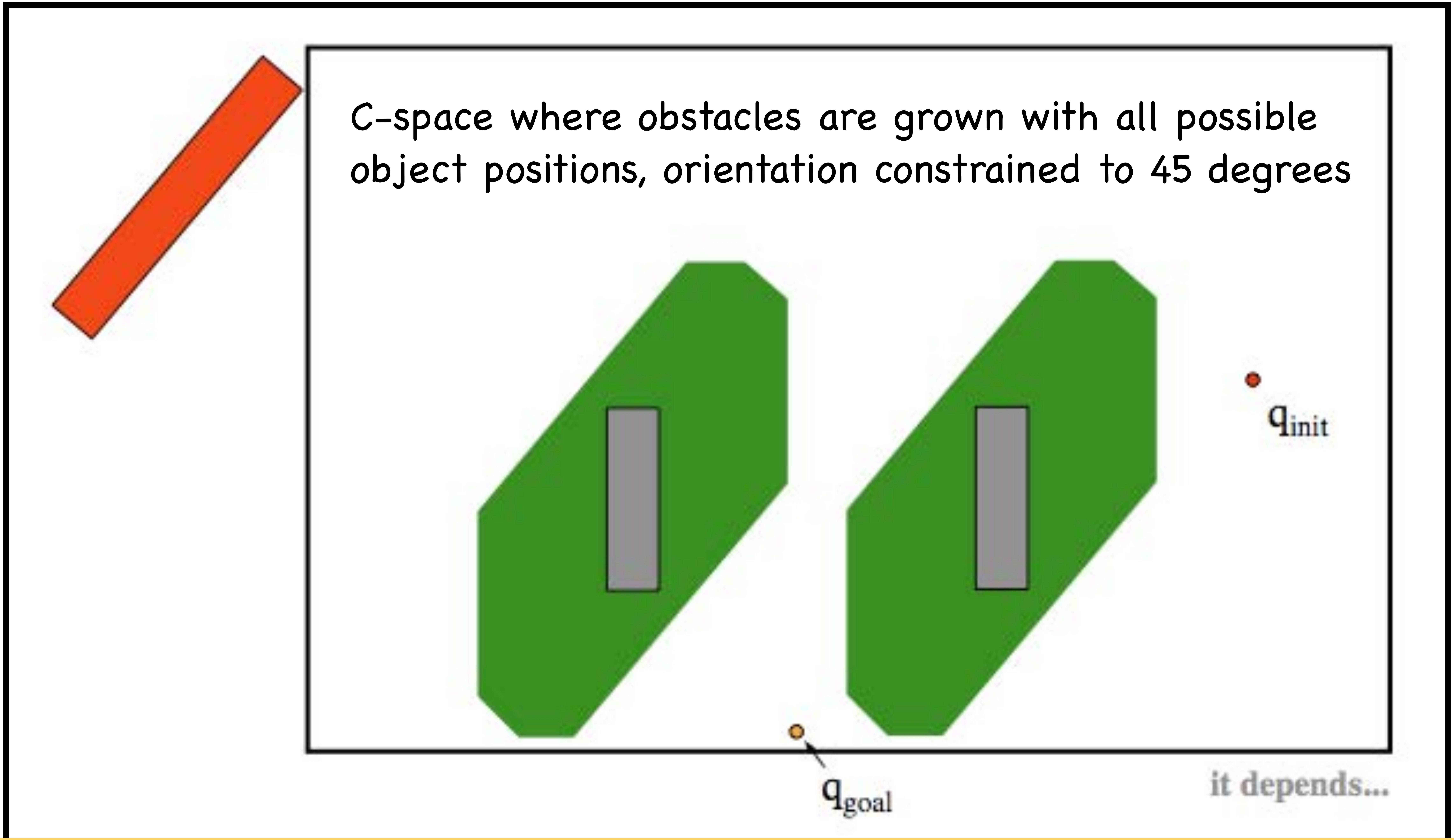
Consider this workspace…



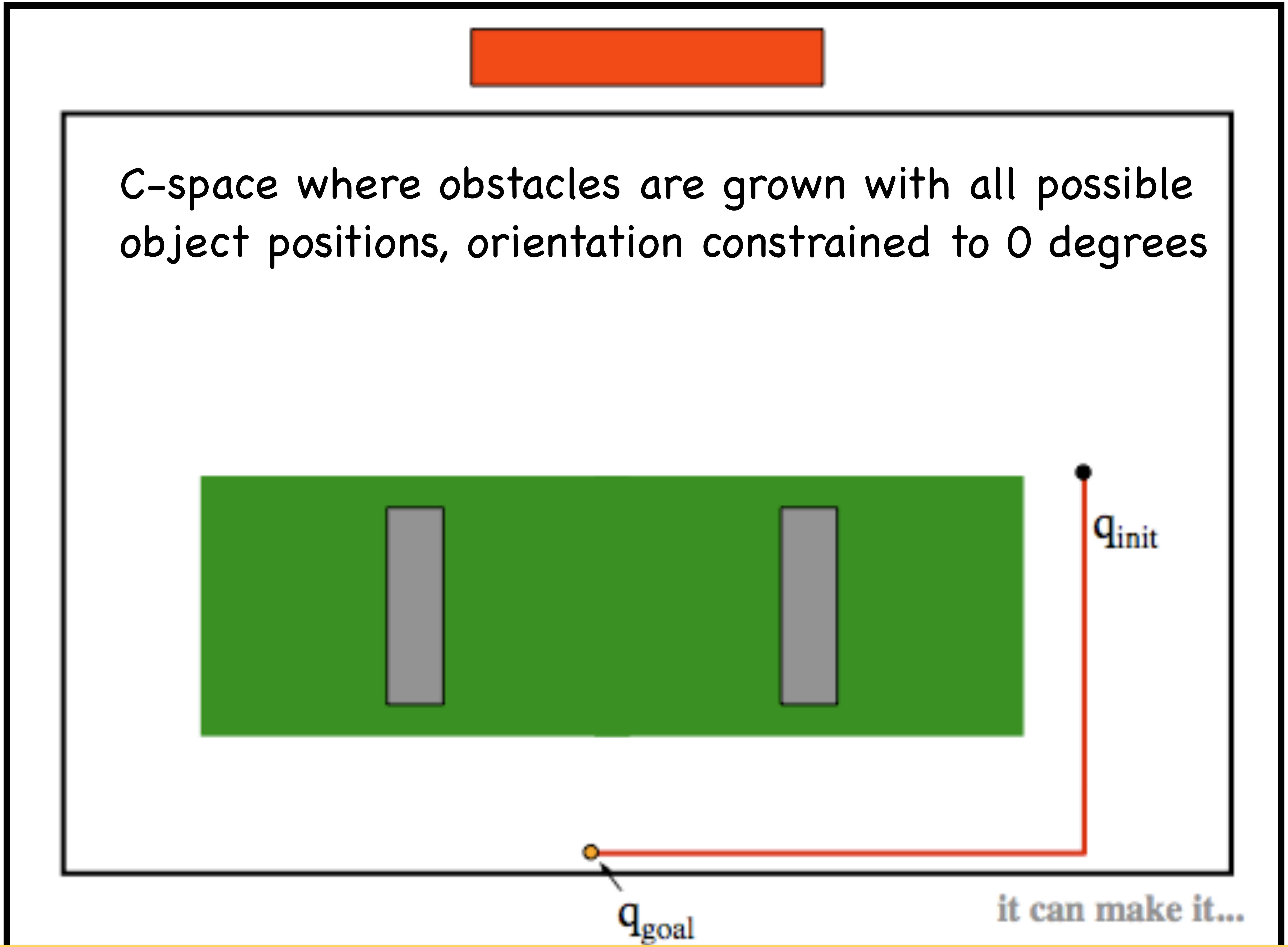
C-space where obstacles are grown with all possible object positions and orientations



C-space where obstacles are grown with all possible object positions, orientation constrained to 45 degrees

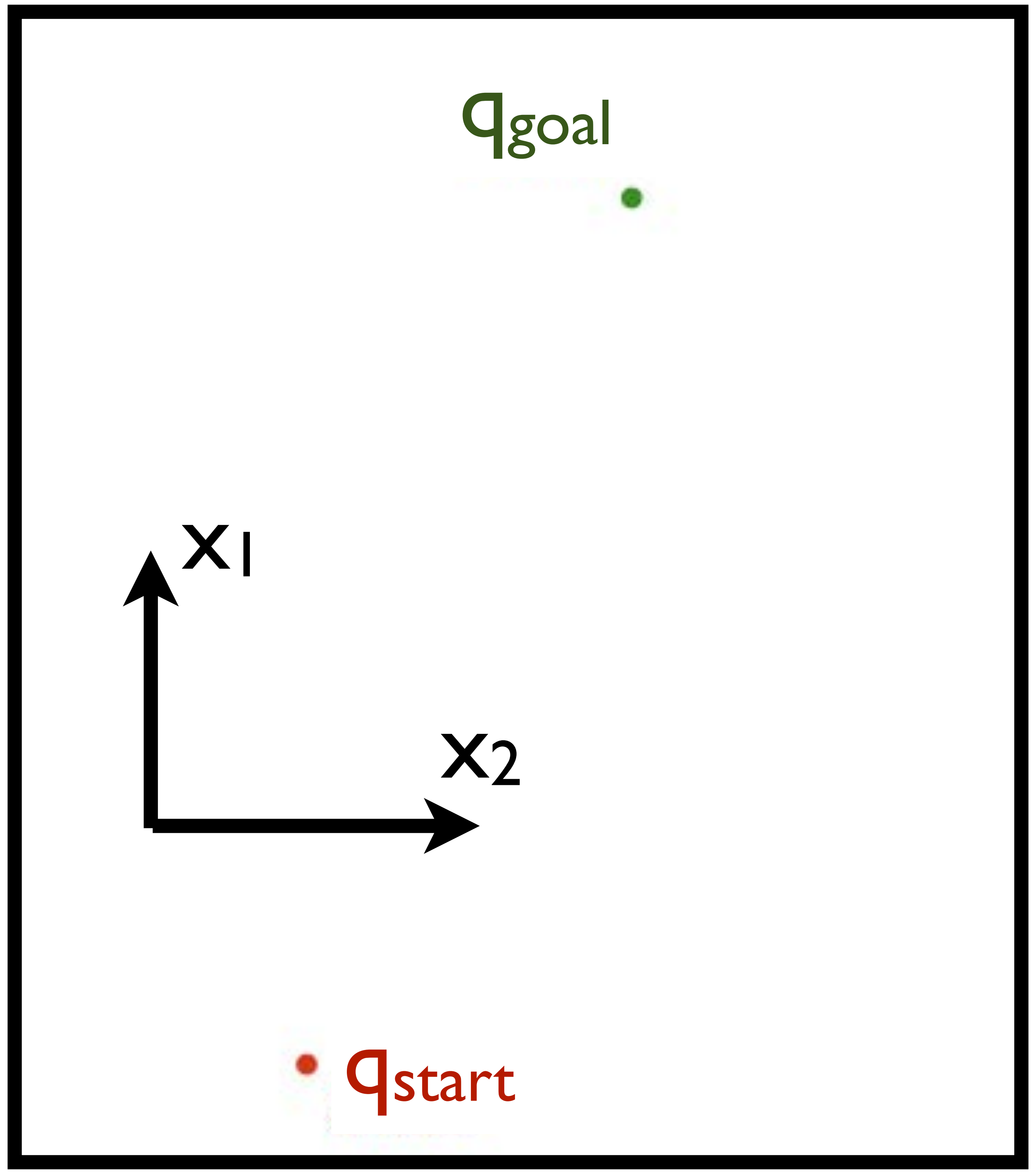


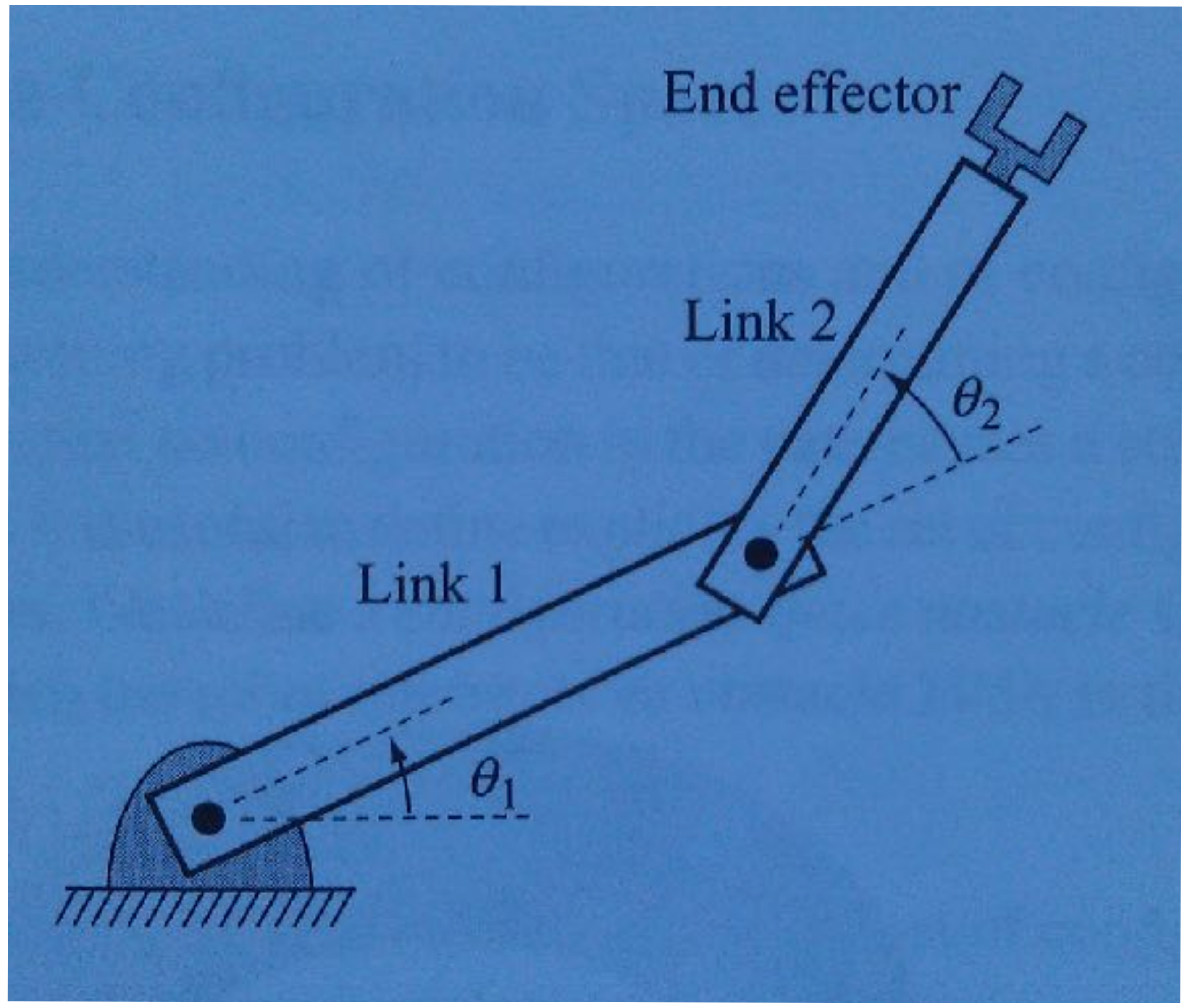
C-space where obstacles are grown with all possible object positions, orientation constrained to 0 degrees

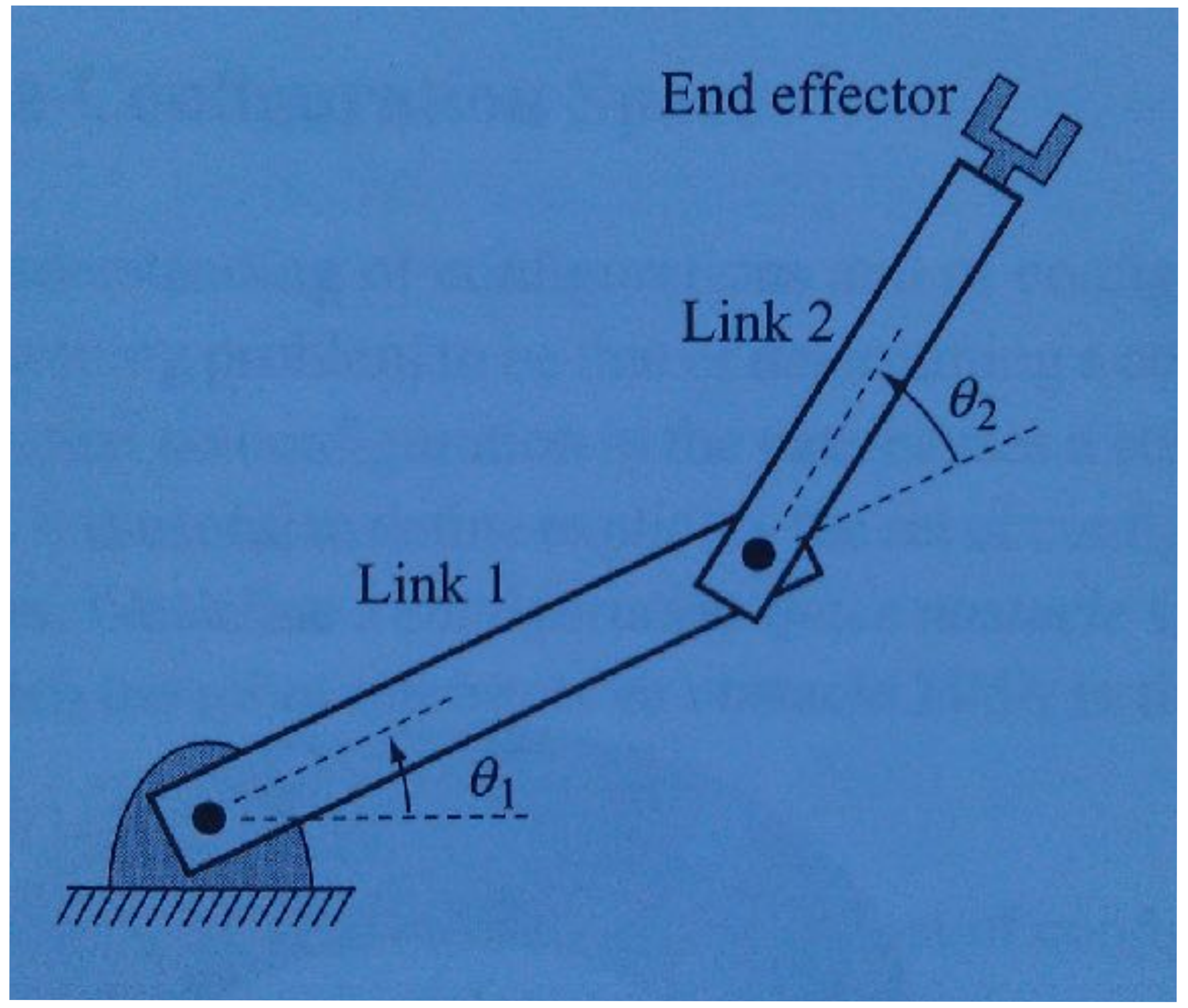


Configuration v. Workspaces

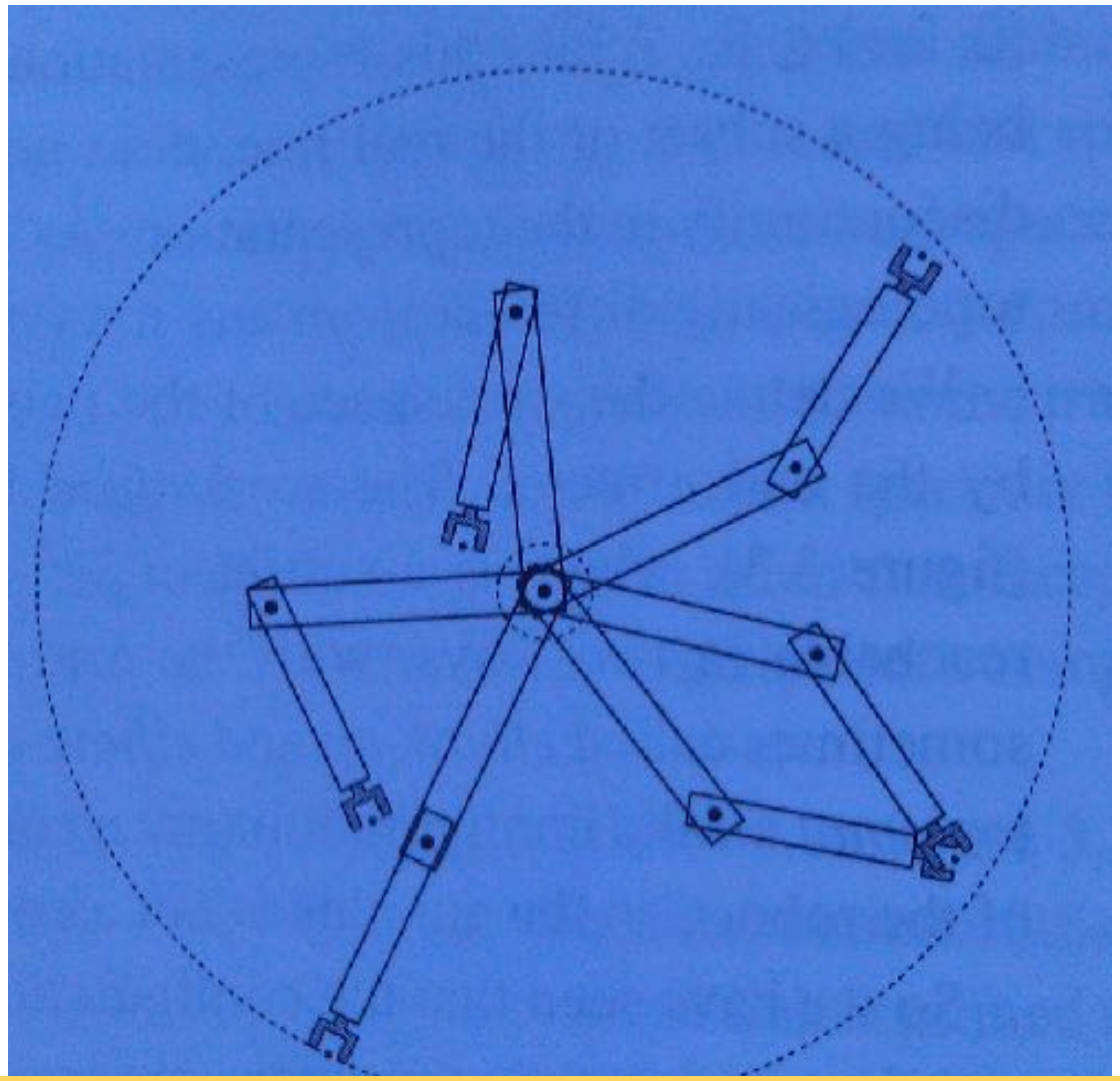
- Other than rotation and geometry, how is the 2-link arm different than the point robot?



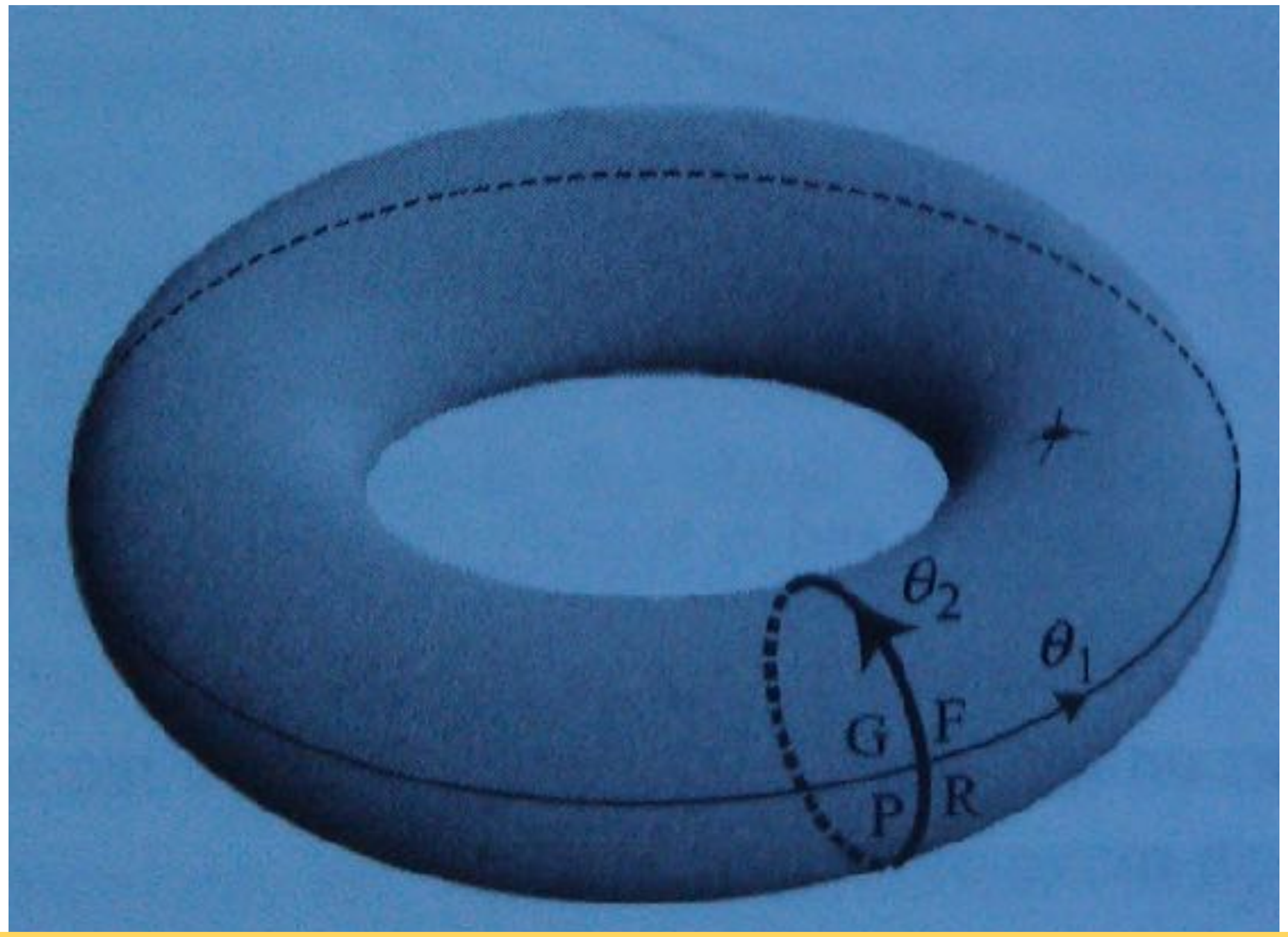




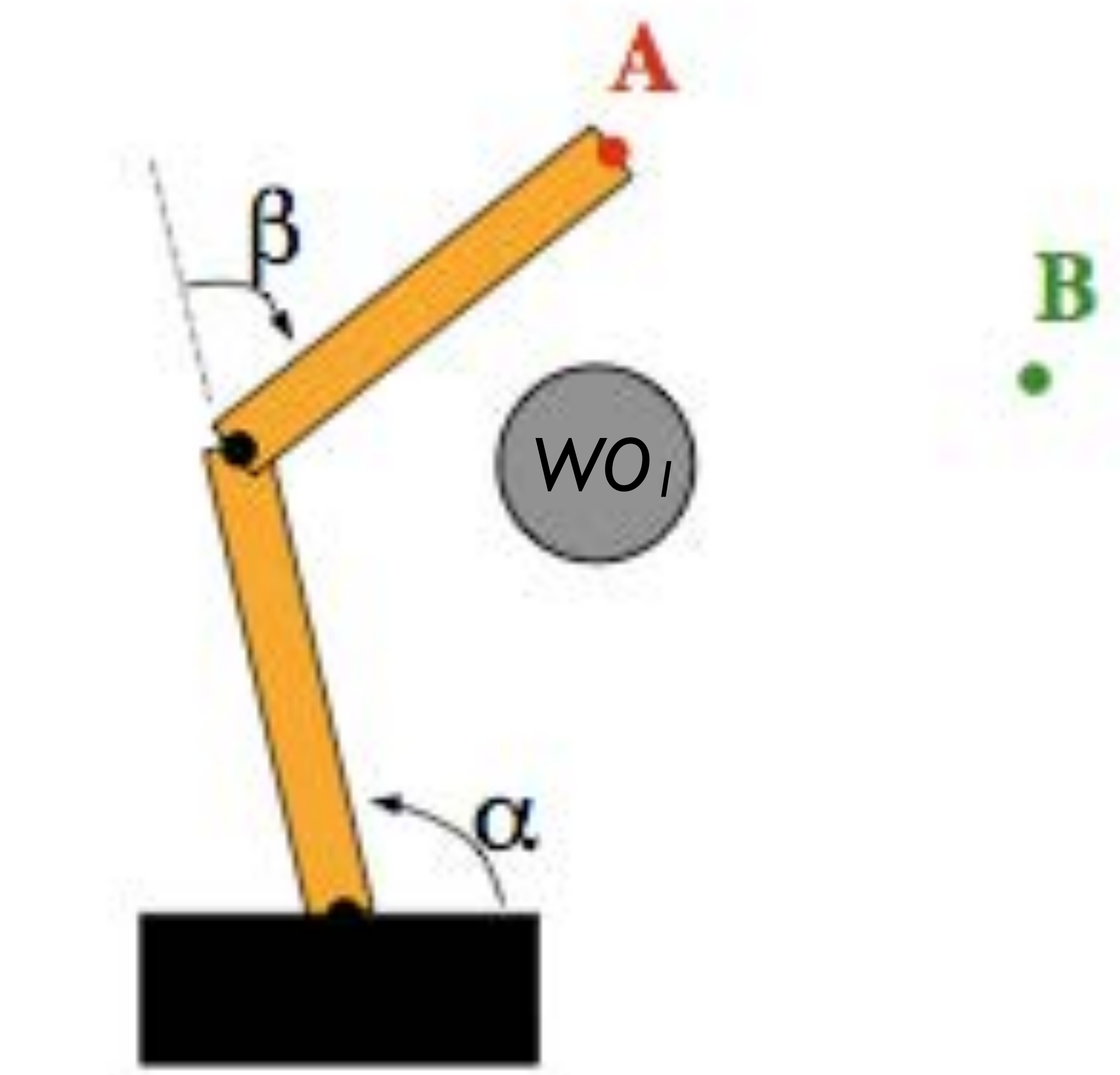
Workspace is w.r.t. end-effector position *(x,y)*



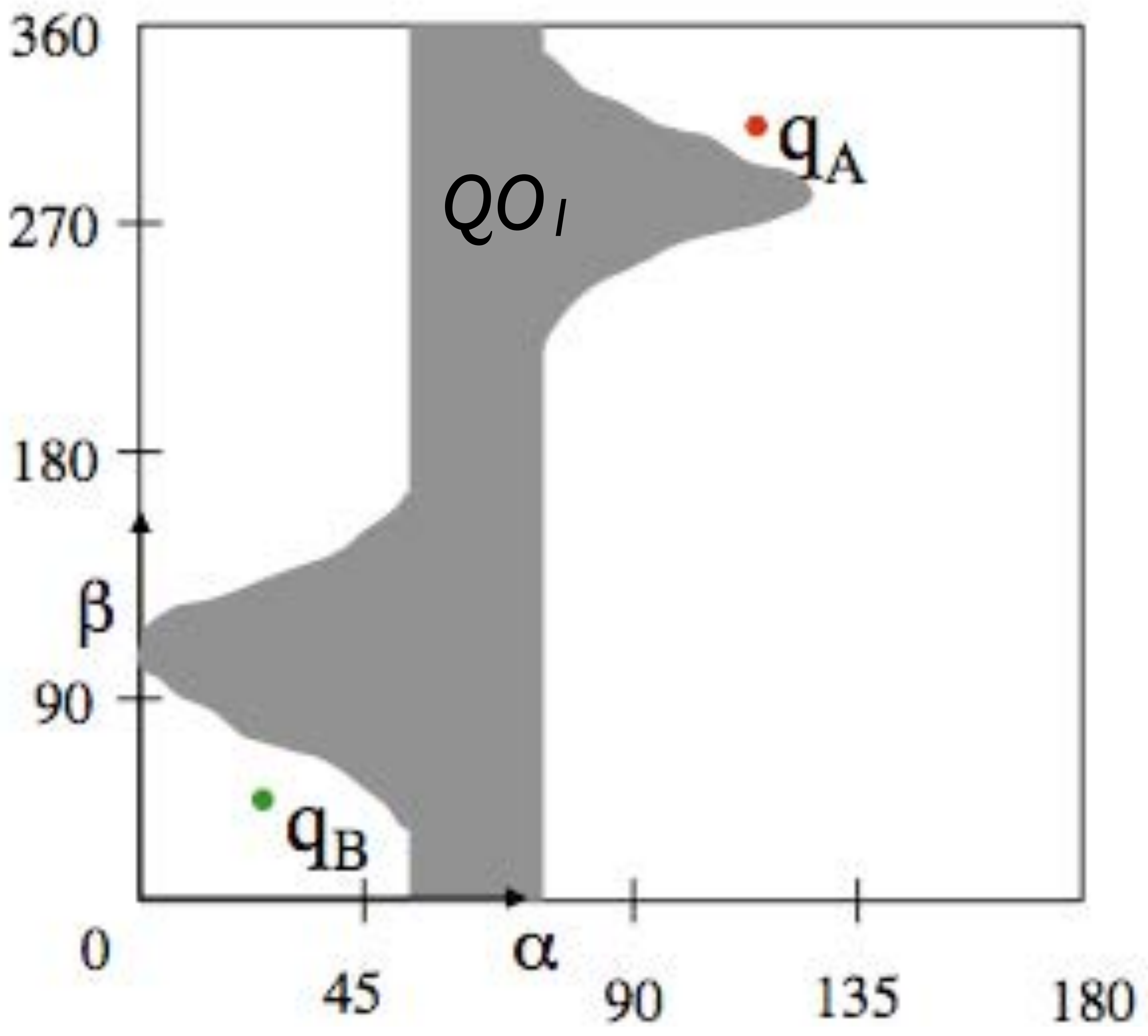
C-space is w.r.t. joint angles *(Θ_1, Θ_2)*



Obstacles in T^2

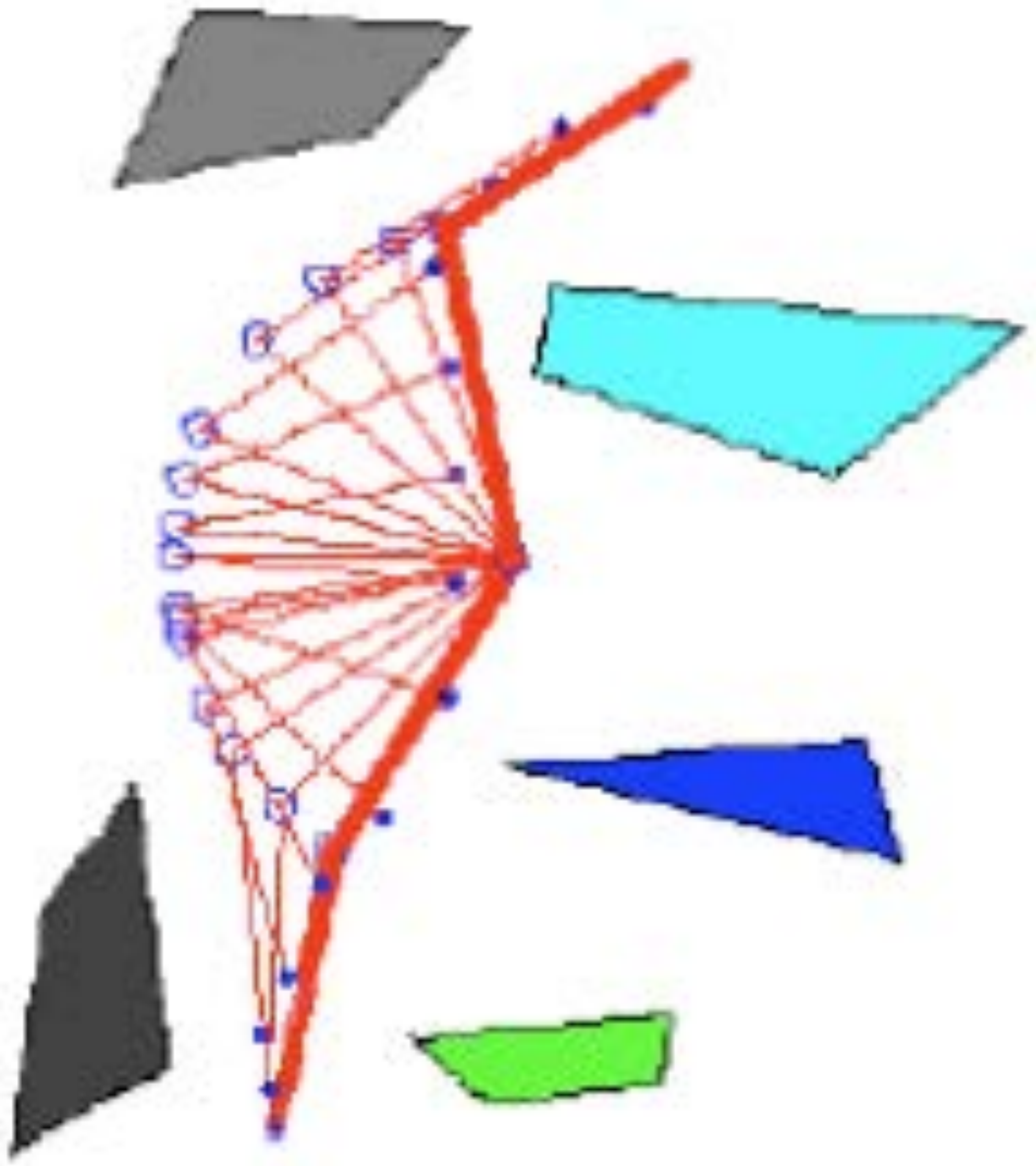


Circular obstacle in workspace

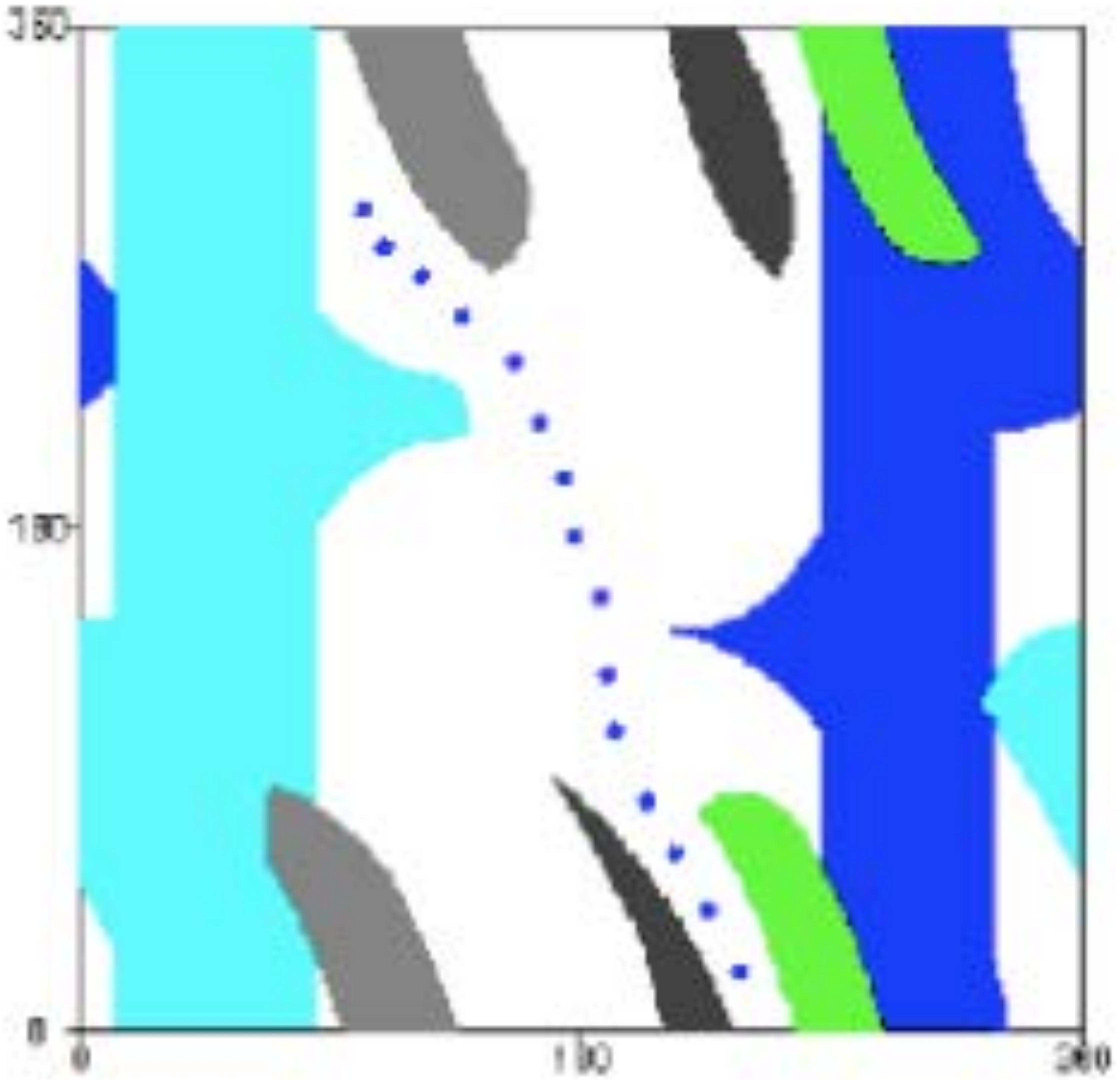


C-space representation

Path in T^2 with several obstacles

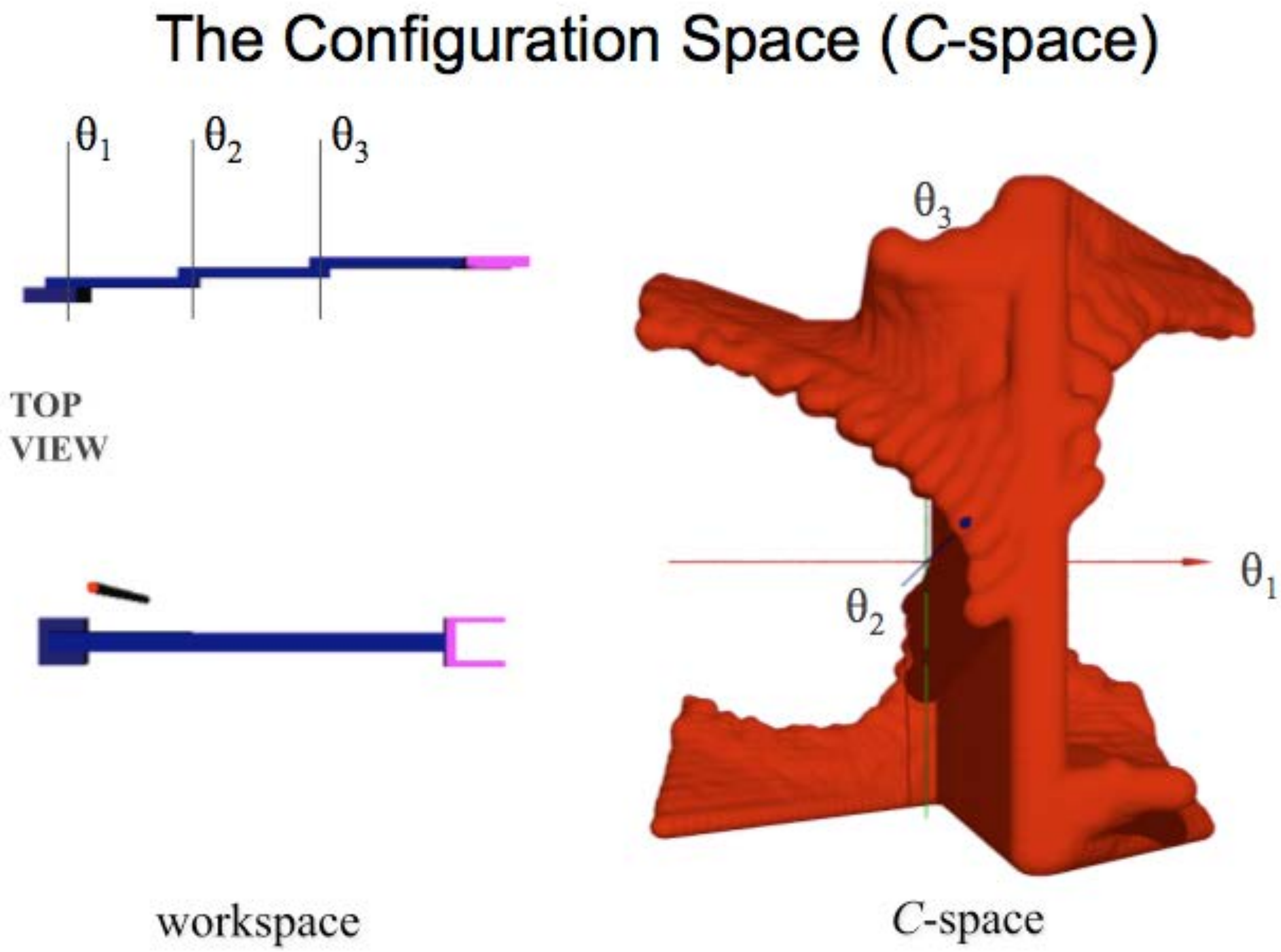


Arm navigation in workspace



C-space representation

C-space for 3-link arm



Slide borrowed from Michigan Robotics autorob.org

Generalizing graph search for robot configurations

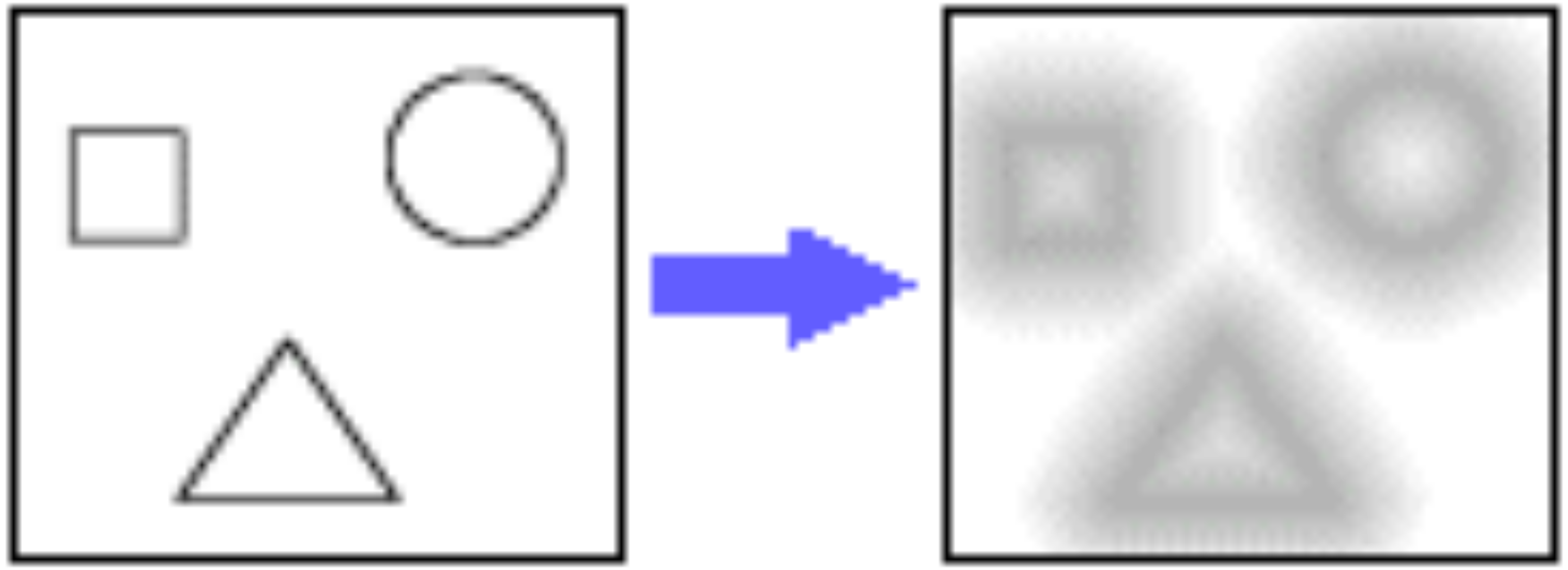
Costmaps: Graph Search Revisited

- Optimality: Path length vs. Path cost?

- **Costmap** provides weights on graph nodes based on cost factors:
  - Robot motion: joint limits, holonomicity, smoothness
  - Collisions and safety: distance from objects, trajectory predictions
  - Environmental conditions: traversability, slip

Distance Transform

Compute distance of each grid cell to nearest obstacle boundary;
Weight grid cell cost higher if closer to a boundary



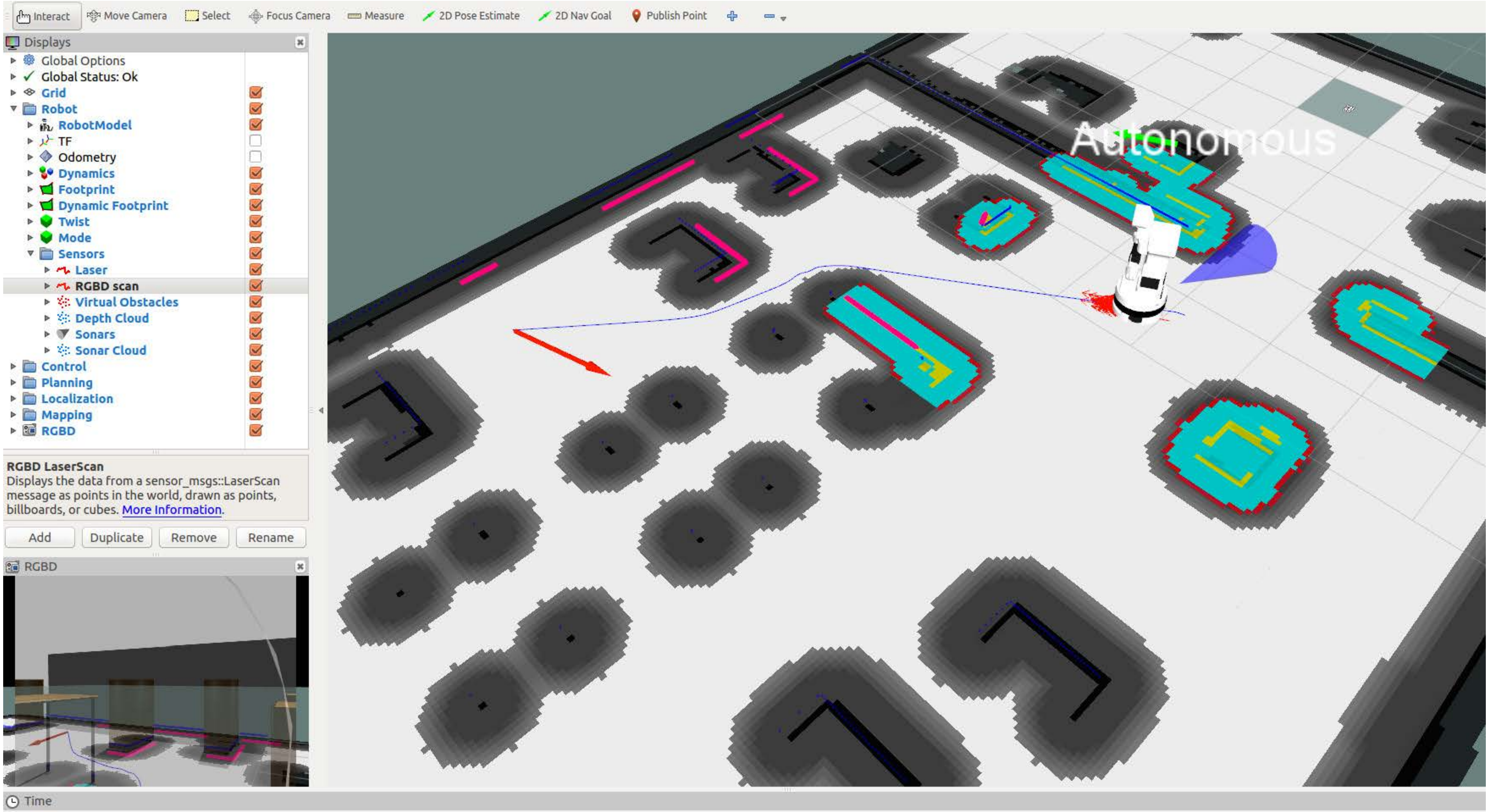
http://www.gavrila.net/Research/Chamfer_System/chamfer_basics2.gif

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	2	4	2	1	0	0
0	0	1	4	8	4	1	0	0
0	0	1	2	4	2	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0

Nasonov and Krylov 2010
(zero indicates obstacle)

https://blog.pal-robotics.com/blog/tiago-ros-simulation-tutorial-2-autonomous-robot-navigation/



Search algorithm template

all nodes ← {$dist_{start}$ ← infinity, $parent_{start}$ ← none, $visited_{start}$ ← false}

start_node ← {$dist_{start}$ ← 0, $parent_{start}$ ← none, $visited_{start}$ ← true}

visit_list ← start_node

while visit_list != empty && current_node != goal

cur_node ← highestPriority(visit_list)

$visited_{cur\_node}$ ← true

for each nbr in not_visited(adjacent(cur_node))

add(nbr to visit_list)

if $dist_{nbr}$ > $dist_{cur\_node}$ + distance(nbr,cur_node)

$parent_{nbr}$ ← current_node

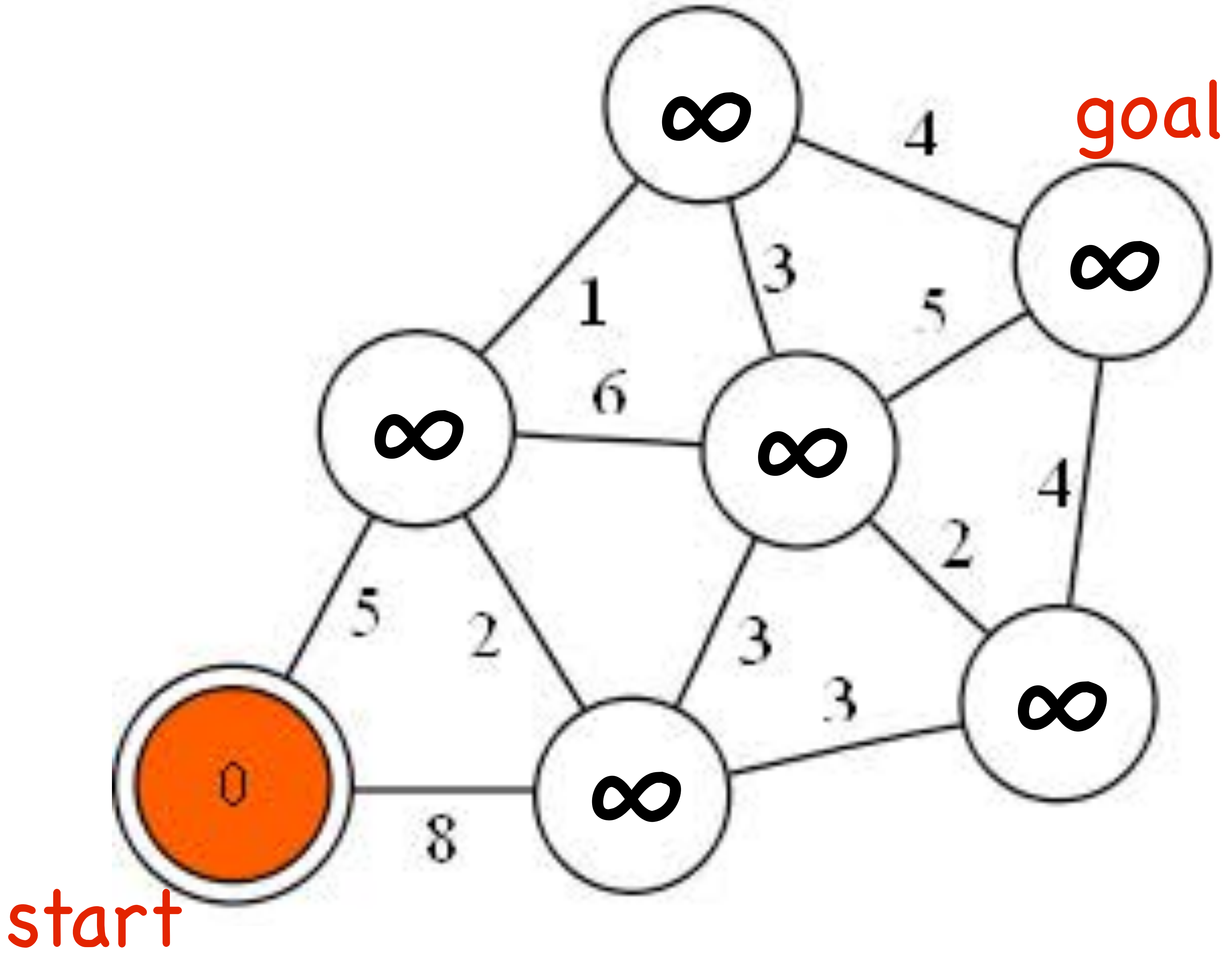
$dist_{nbr}$ ← $dist_{cur\_node}$ + distance(nbr,cur_node)

end if

end for loop

end while loop

output ← parent, distance



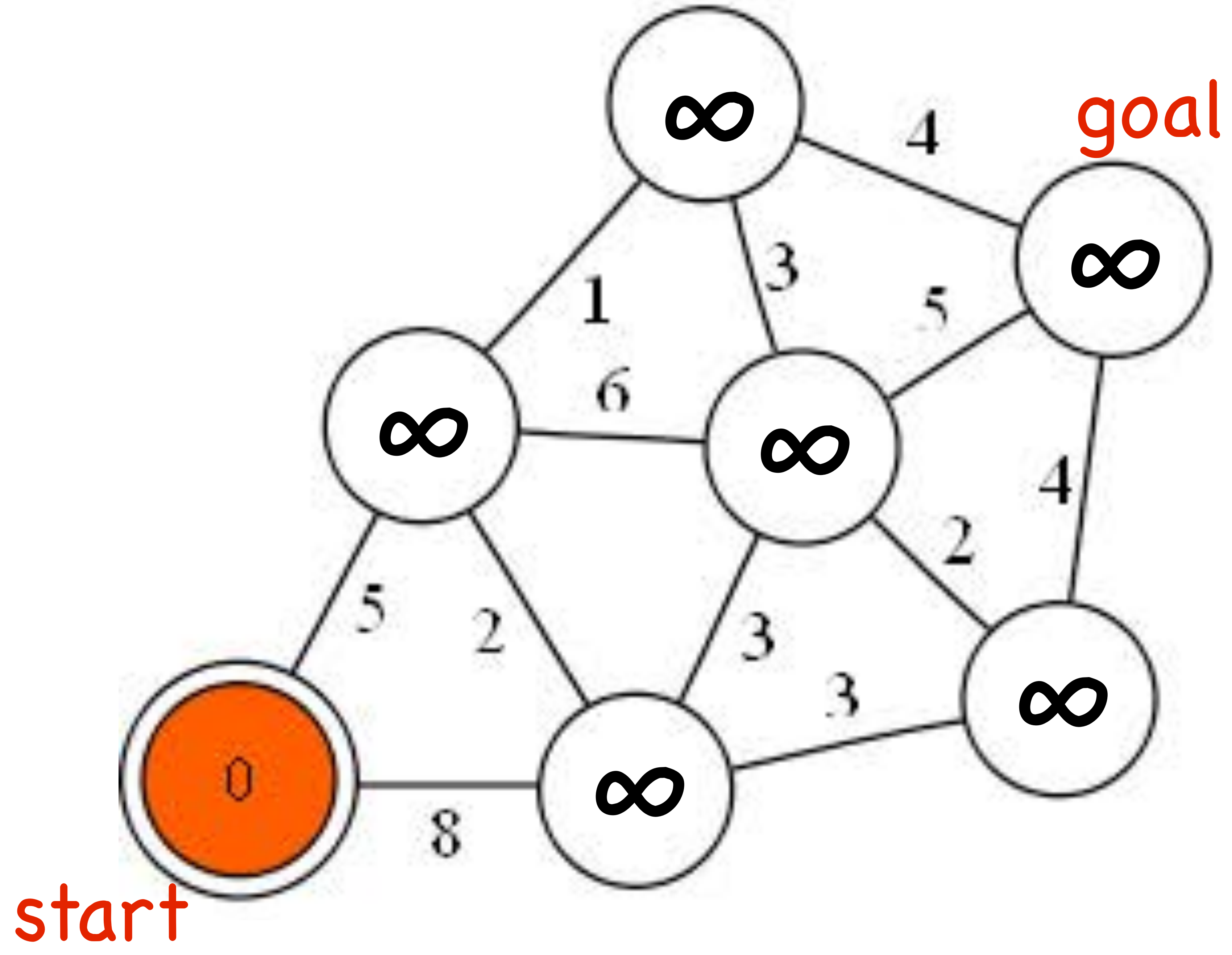
Search algorithm template

all nodes ← {$cost_{start}$ ← infinity, $parent_{start}$ ← none, $visited_{start}$ ← false}

start_node ← {$cost_{start}$ ← 0, $parent_{start}$ ← none, $visited_{start}$ ← true}

visit_list ← start_node

```
    while visit_list != empty && current_node != goal
      cur_node ← highestPriority(visit_list)
      visited_cur_node ← true
      for each nbr in not_visited(adjacent(cur_node))
        add(nbr to visit_list)
        if cost_nbr > cost_cur_node + cost(nbr)
          parent_nbr ← current_node
          cost_nbr ← cost_cur_node + cost(nbr)
        end if
      end for loop
    end while loop
output ← parent, distance
```



Slide borrowed from Michigan Robotics autorob.org

A-star shortest path algorithm

all nodes ← {$\text{cost}_{\text{start}}$ ← infinity, $\text{parent}_{\text{start}}$ ← none, $\text{visited}_{\text{start}}$ ← false}

start_node ← {$\text{cost}_{\text{start}}$ ← 0, $\text{parent}_{\text{start}}$ ← none, $\text{visited}_{\text{start}}$ ← true}

visit_queue ← start_node

    while (visit_queue != empty) && current_node != goal

        dequeue: cur_node ← f_score(visit_queue)

        $\text{visited}_{\text{cur\_node}}$ ← true

        for each nor in not_visited(adjacent(cur_node))

            enqueue: nbr to visit_queue

            if $\text{cost}_{\text{nbr}} > \text{cost}_{\text{cur\_node}} + \text{cost}(\text{nbr})$

                $\text{parent}_{\text{nbr}}$ ← current_node

                $\text{cost}_{\text{nbr}} \leftarrow \text{cost}_{\text{cur\_node}} + \text{cost}(\text{nbr})$

                $\text{f\_score} \leftarrow \text{cost}_{\text{nbr}} + \text{line\_distance}_{\text{nbr,goal}}$

            end if

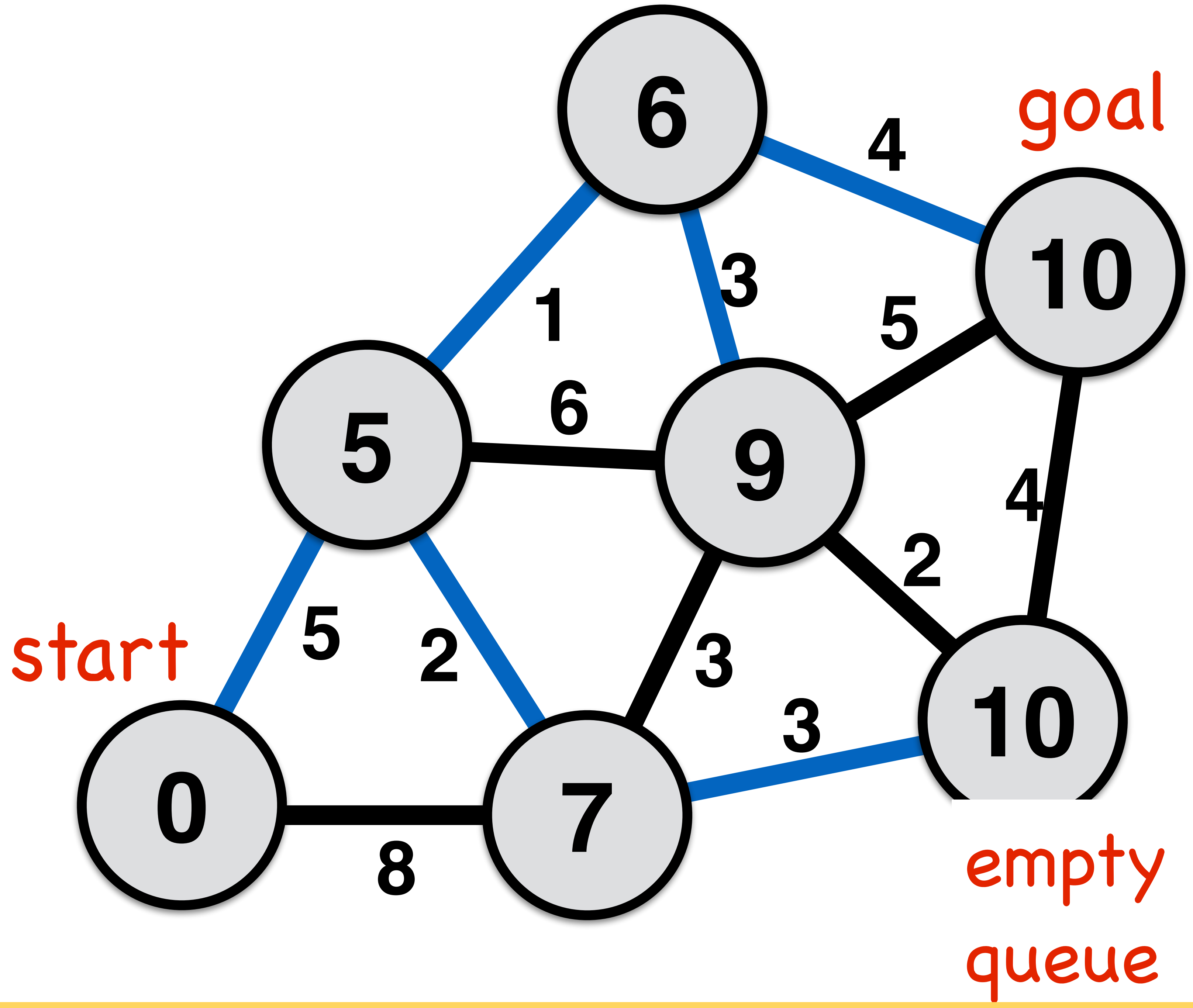
        end for loop

    end while loop

output ← parent, distance

g_score:
cost from start

h_score:
optimistic cost to goal



Slide borrowed from Michigan Robotics autorob.org

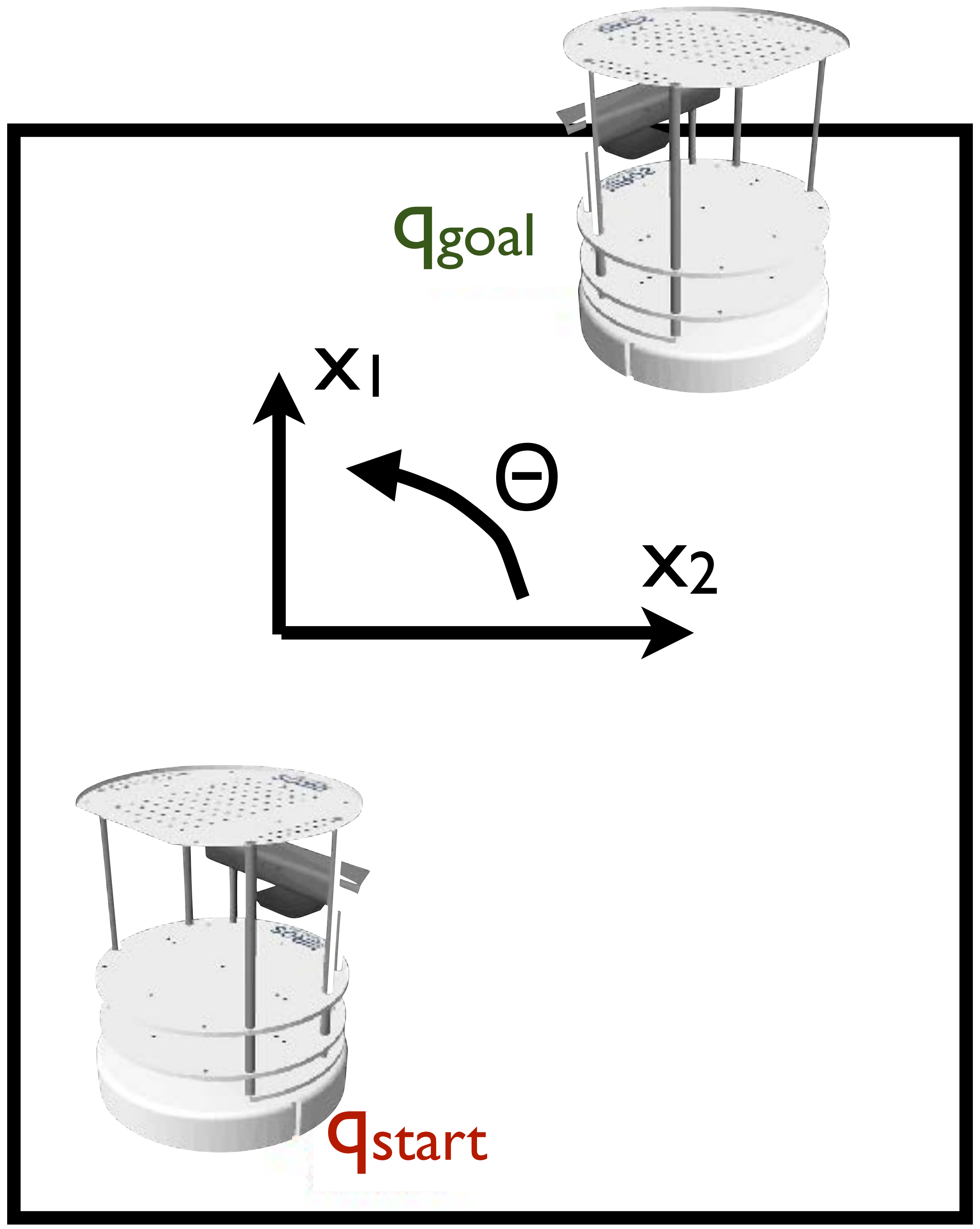
Can a robot move in any direction instantaneously?

Holonomicity

- Does the Turtlebot have 2 DOFs, instead of 3?

- The Turtlebot can only move along 2 axes
  - linear: forward/backward
  - angular: turning

Turtlebot is nonholonomic



Holonomicity

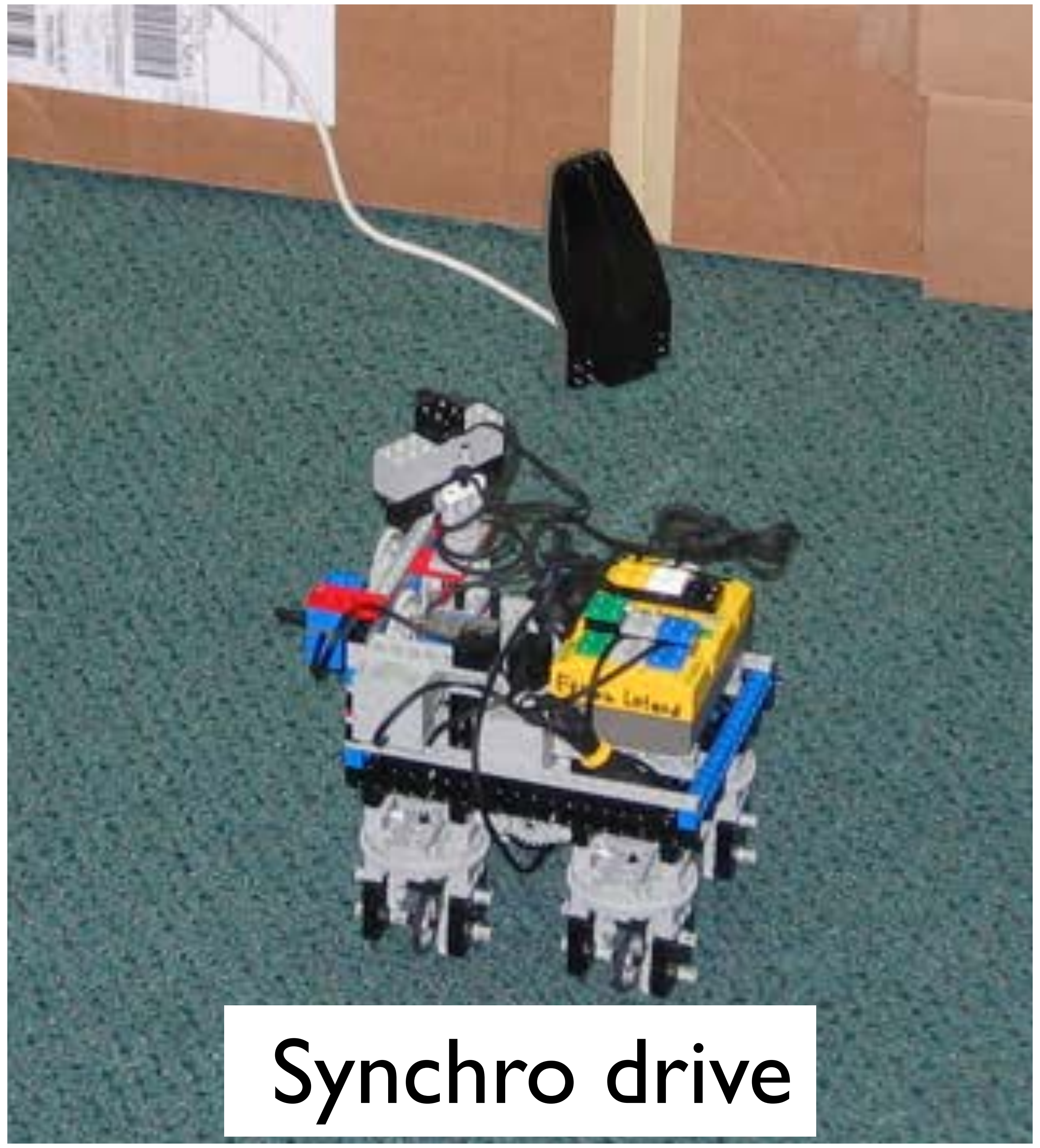
https://www.youtube.com/watch?v=c-lEjVsoiGo

https://www.youtube.com/watch?v=1ak17mdRg5I&t=75s

- A robot is holonomic if it can change its pose instantaneously to move in all directions
- Otherwise, the robot is nonholonomic

Holonomic mobile robot systems

Omni-wheel drive



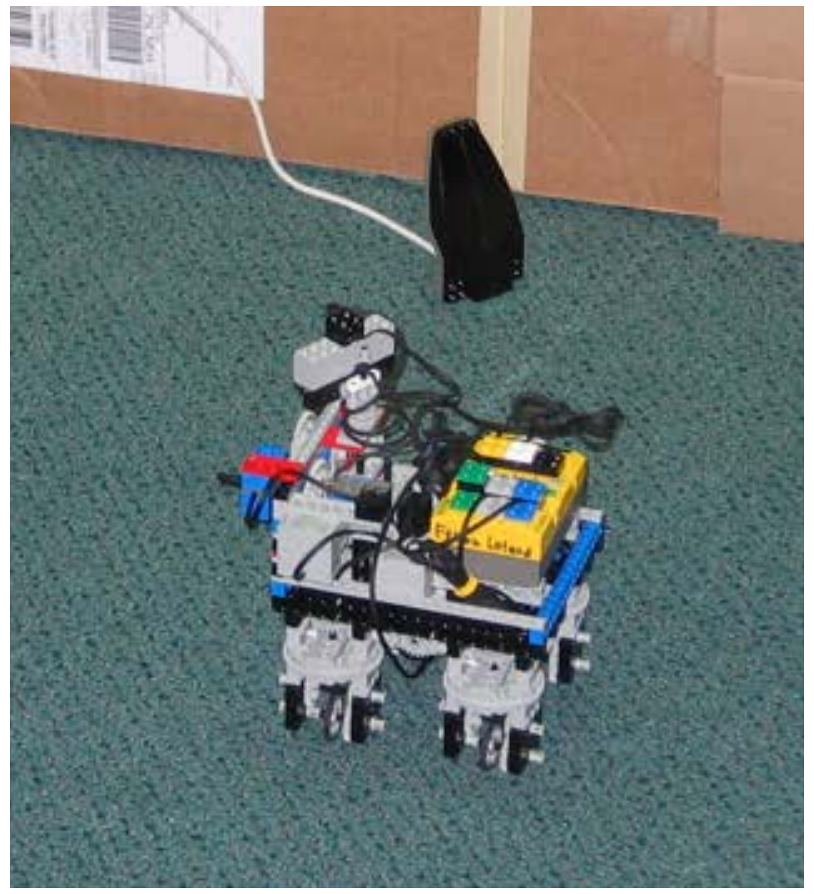
Synchro drive

E. Leland, Segway, robotthoughts.com

Mecanum wheels

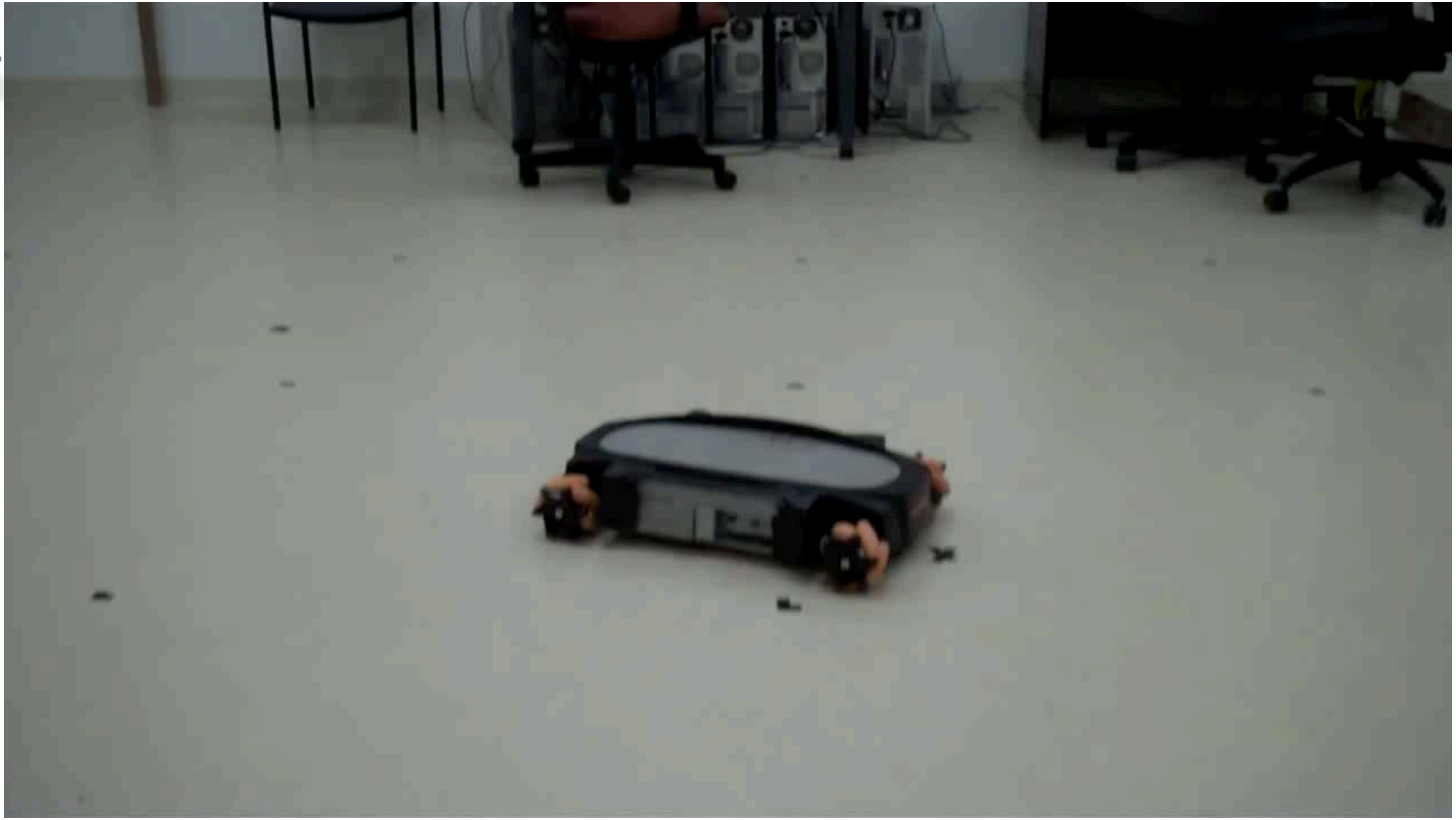
Killough platform

Synchro Drive



markclego, https://www.youtube.com/watch?v=THdu6QD8Roc

KUKA YouBot with Mecanum wheels

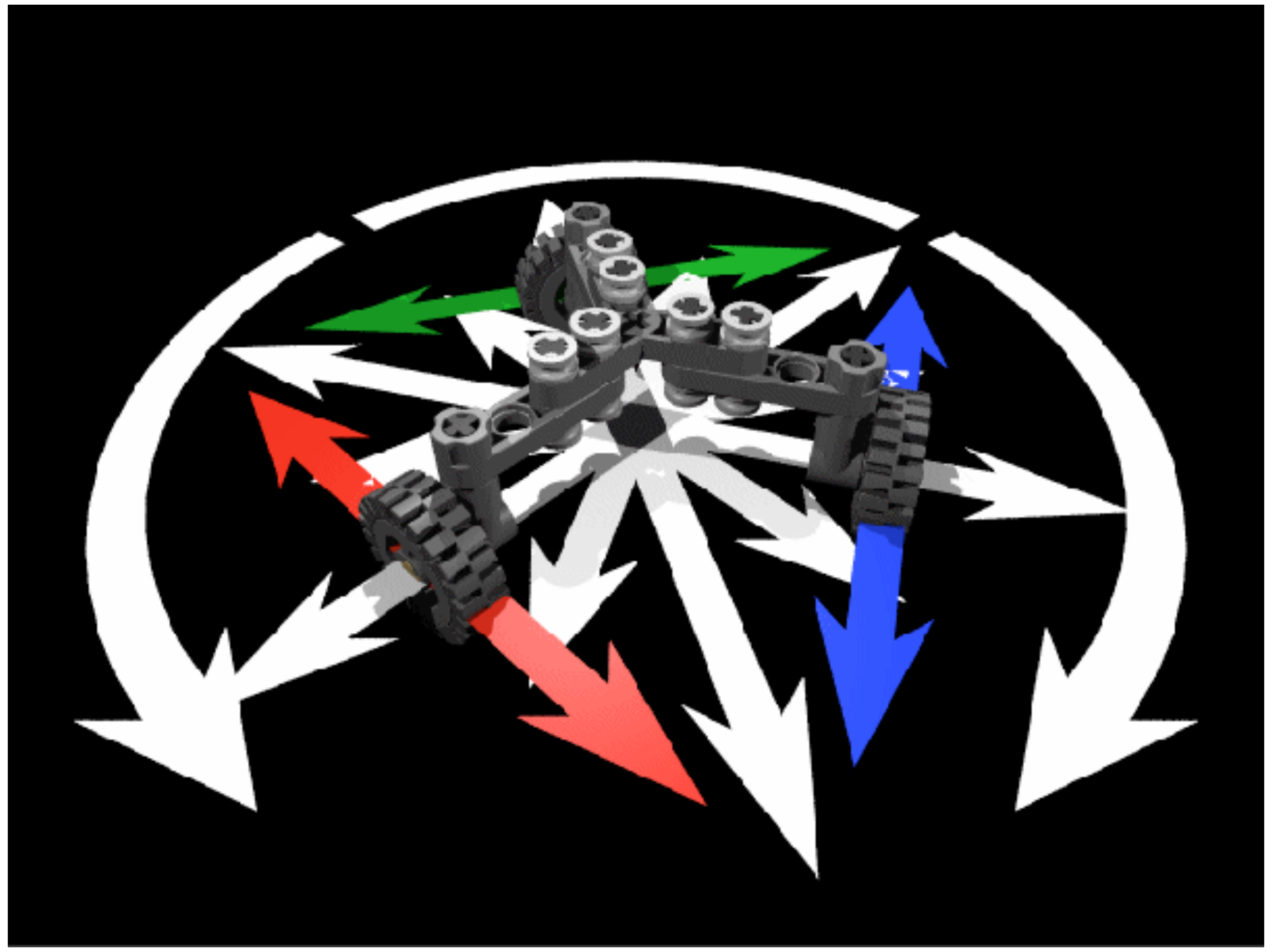


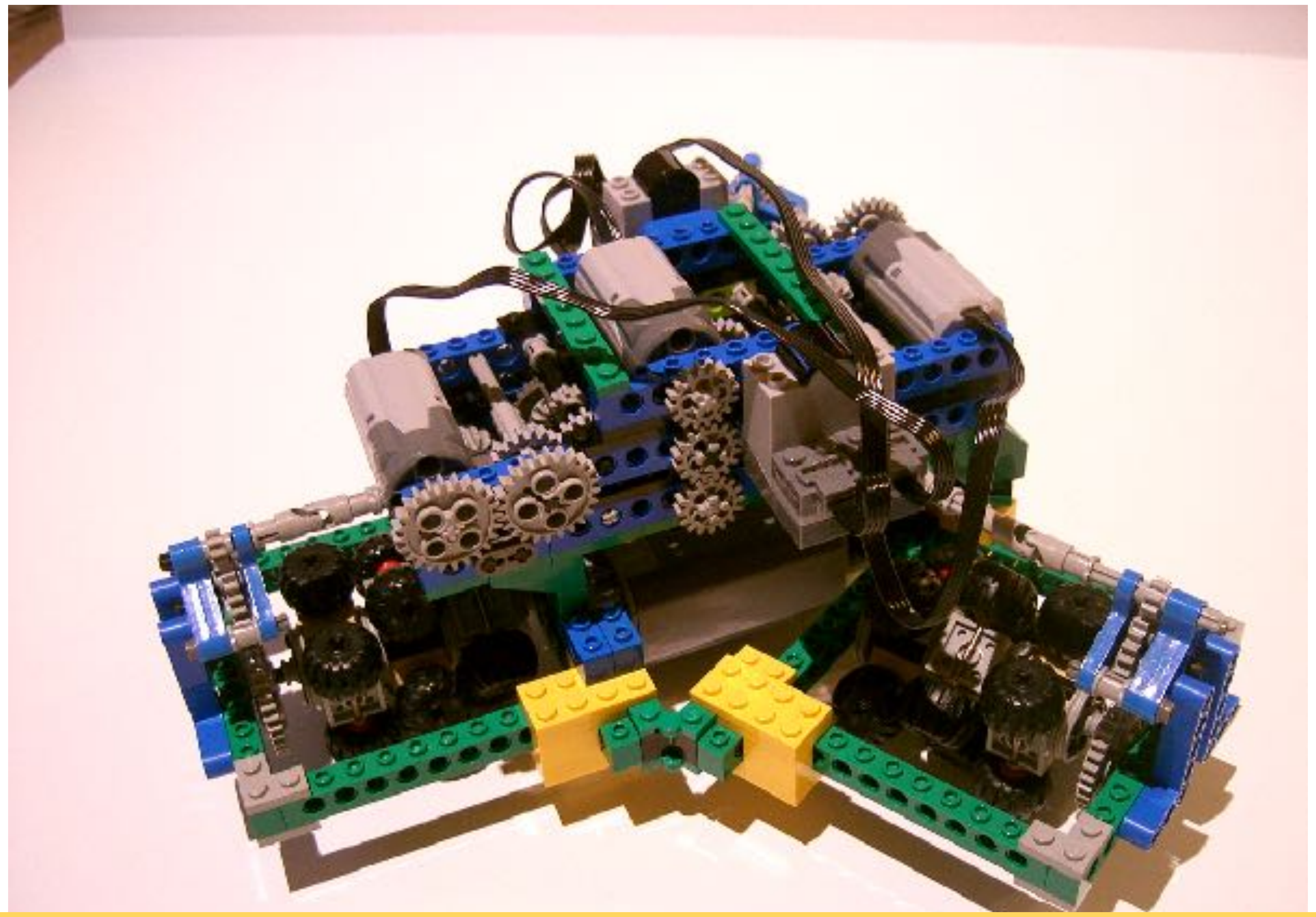
Me teleoperating KUKA YouBot from my house via a web browser, http://youtu.be/sWrRiy0AM_w

DJI Robomaster Racing

Japan Times, https://www.youtube.com/watch?v=52skH4NpnvI

Killough platform

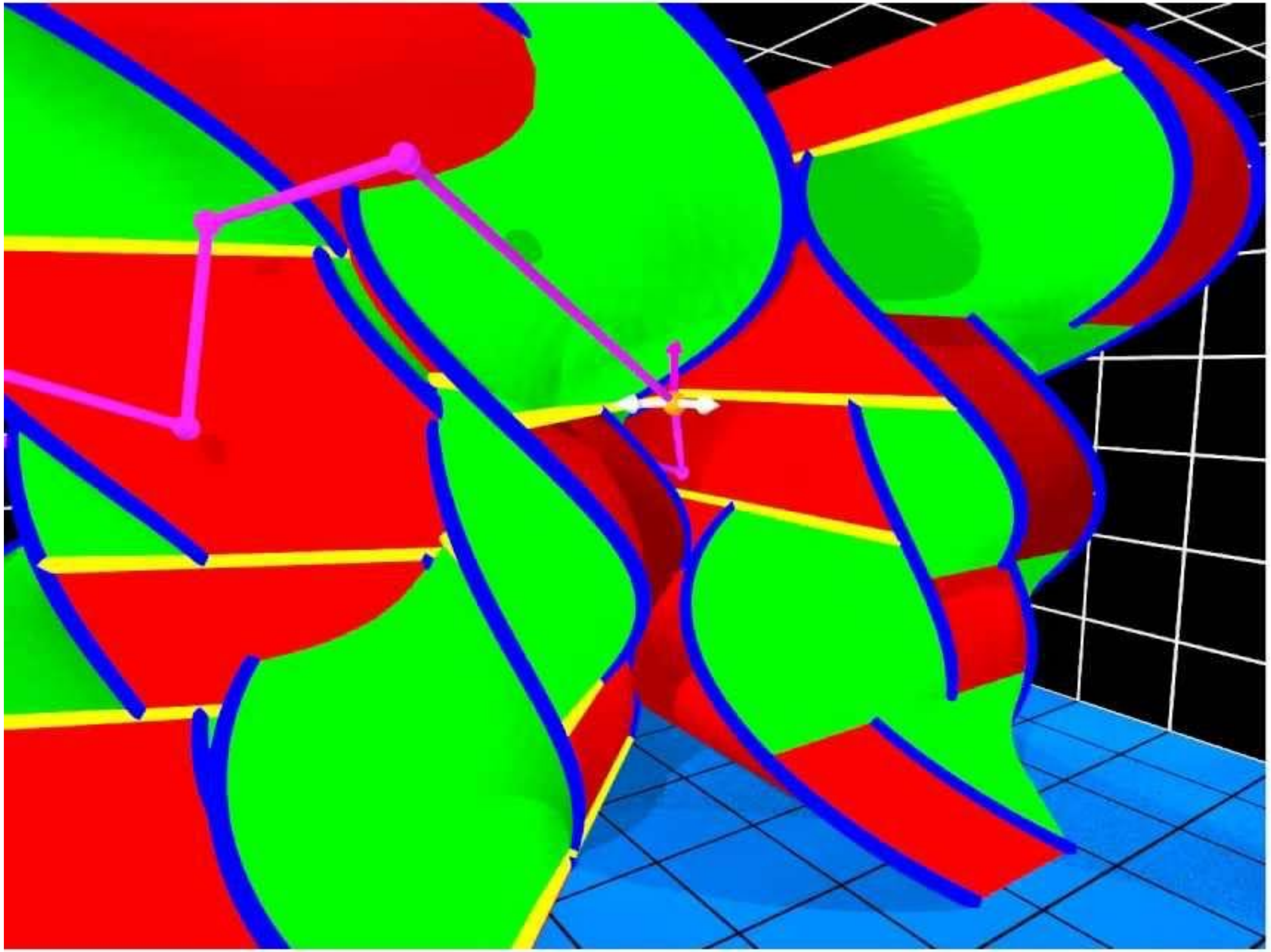




robotthoughts.com; http://technicbricks.blogspot.com/2008/08/going-to-all-places-in-all-directions_29.html

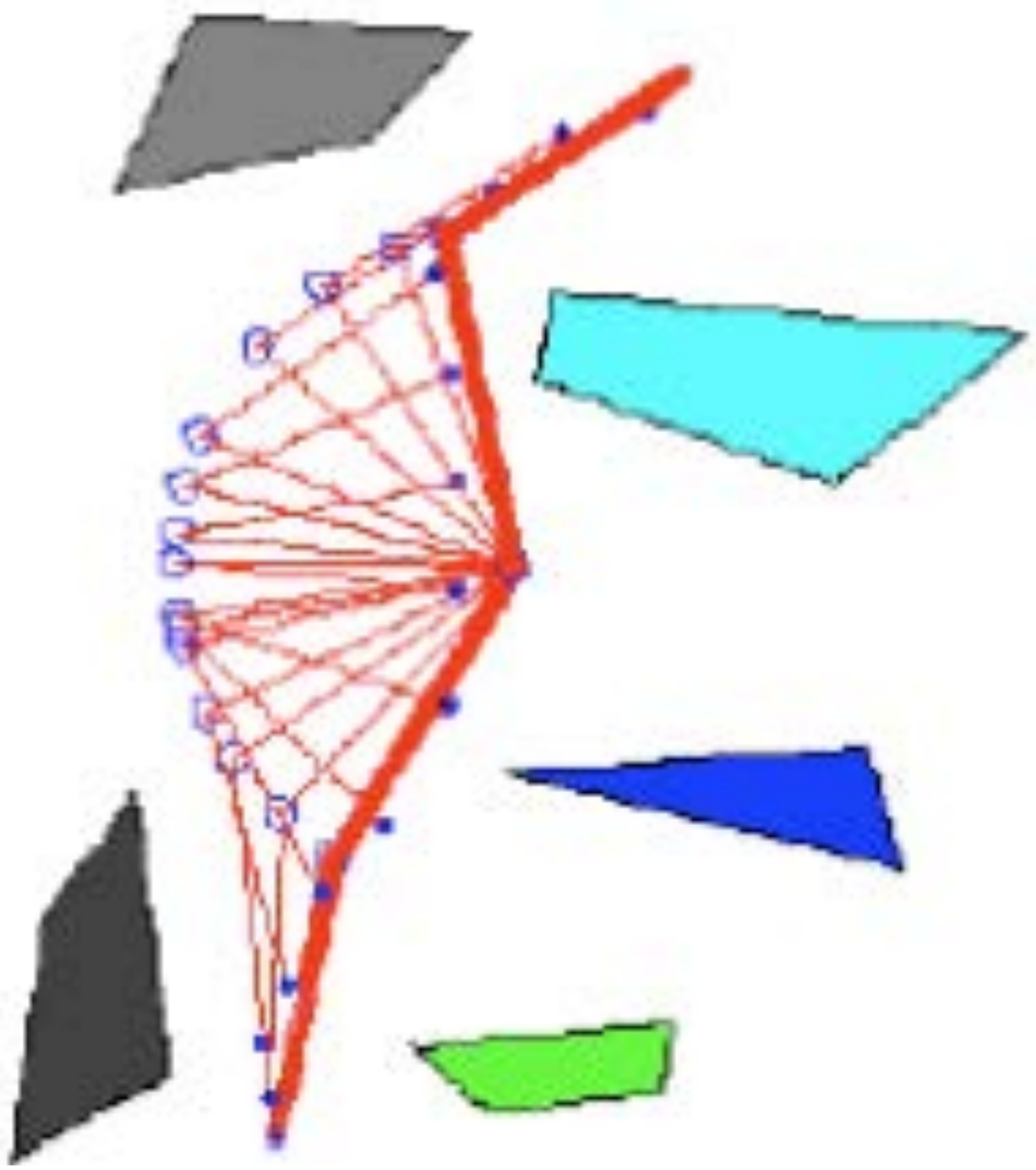
Recommended: D'Andrea on Omni-drive

https://www.youtube.com/watch?v=p_WI-C-ORso

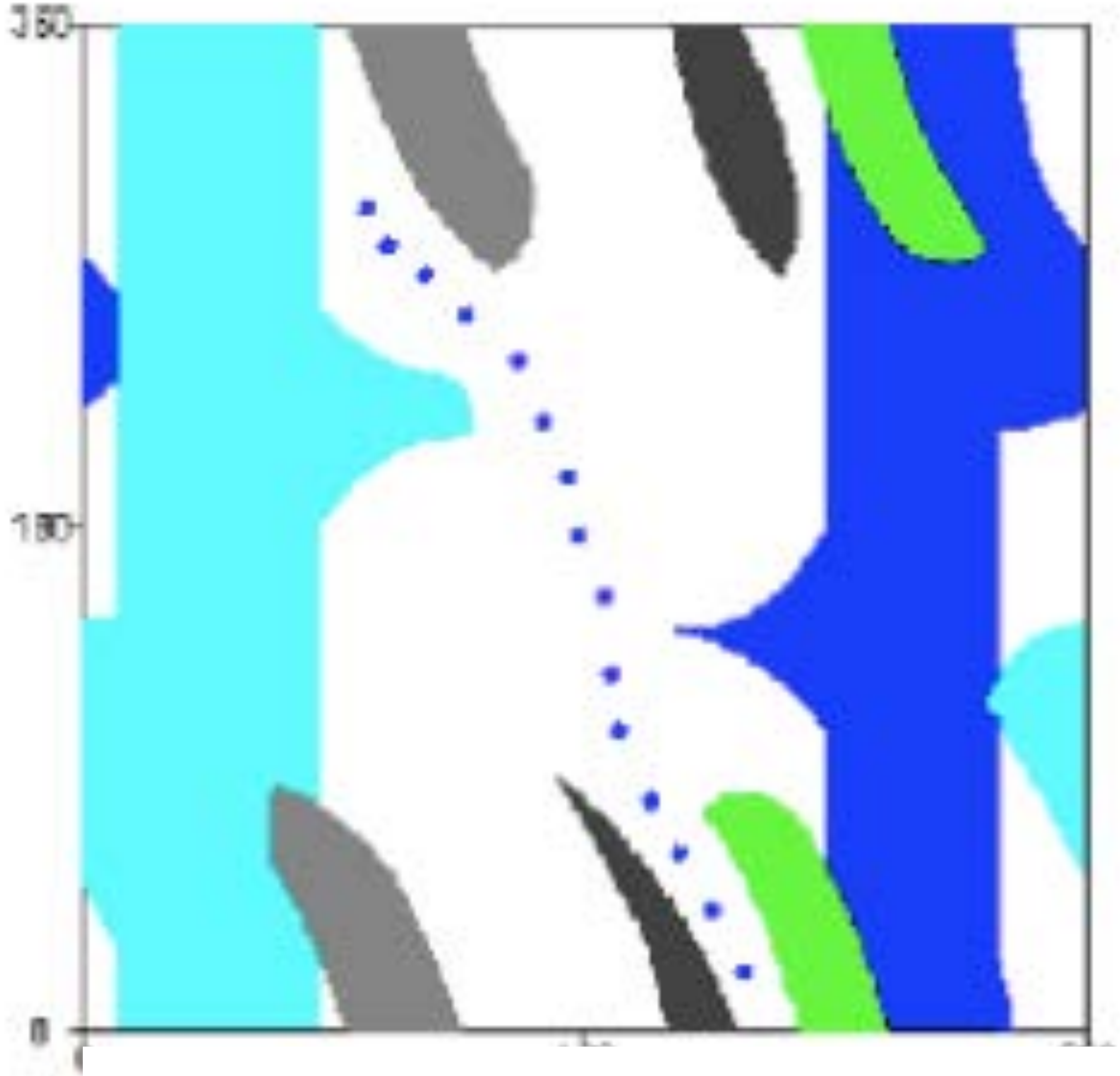


Visualization developed by Dror Atariah and Günter Rote - https://www.youtube.com/watch?v=SBFwgR4K1Gk

How do we search arbitrary C-spaces?



Arm navigation in workspace



C-space representation

How build graphs in arbitrary C-spaces?

Next Lecture

Planning - IV - Sampling-based Planning