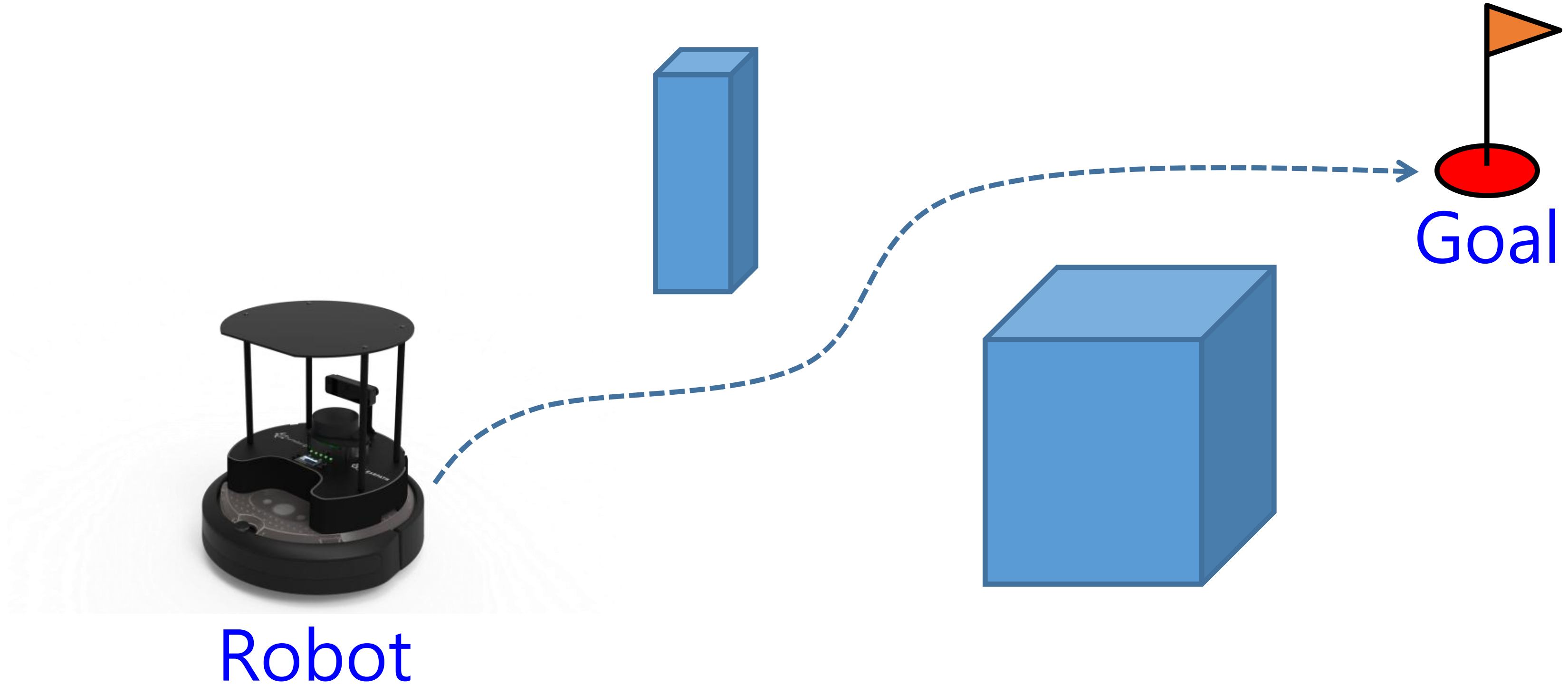
ROS2: Path Planning

운영체제의 실제

안인규 (Inkyu An)

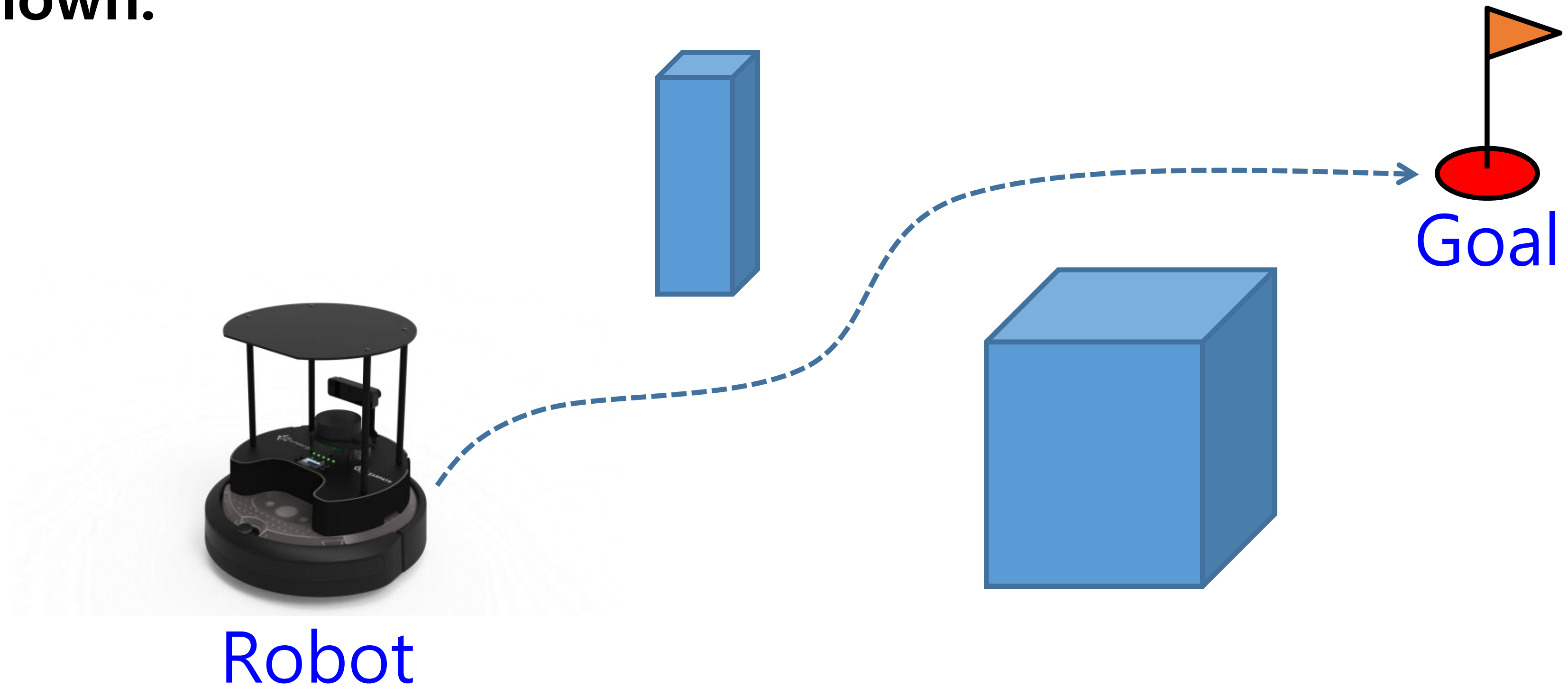
Path Planning

- Path planning is the process of finding the most efficient route from a start point to a goal **while avoiding obstacles**



Path Planning

- Path planning is the process of finding the most efficient route from a start point to a goal **while avoiding obstacles**
  - **In this lecture, we assume that the locations of all obstacles are known.**



Path Planning

- Configuration space
  - The 'world' has two entities: **robots and obstacles**. Both considered as closed subsets of the world (or workspace)
  - The 'space' for motion planning is the set of possible transformations that could be applied to the robot (considered as a rigid body)
  - We refer to this as the **configuration space**
  - Important **abstraction** that allows to use the same motion planning algorithms to problems that differ in geometry and kinematics

referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

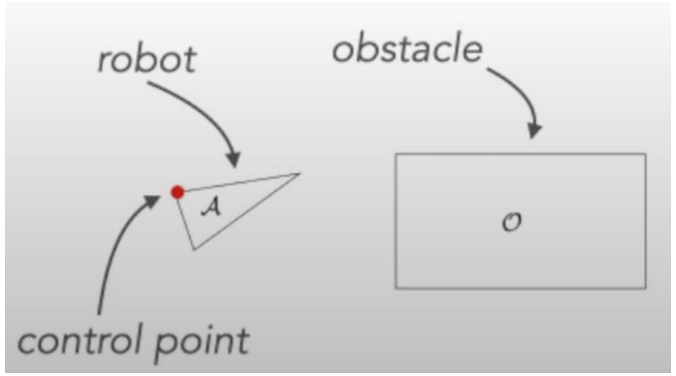
Path Planning

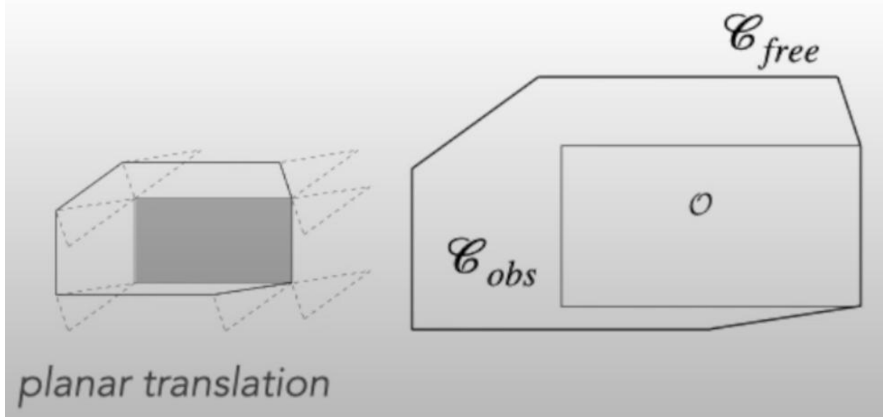
- Configuration space
  - The 'world' has two entities: **robots and obstacles**. Both considered as closed subsets of the world (or workspace)
  - The 'space' for motion planning is the set of possible transformations that could be applied to the robot (considered as a rigid body)
  - We refer to this as the **configuration space**
  - Important **abstraction** that allows to use the same motion planning algorithms to problems that differ in geometry and kinematics

referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

Path Planning

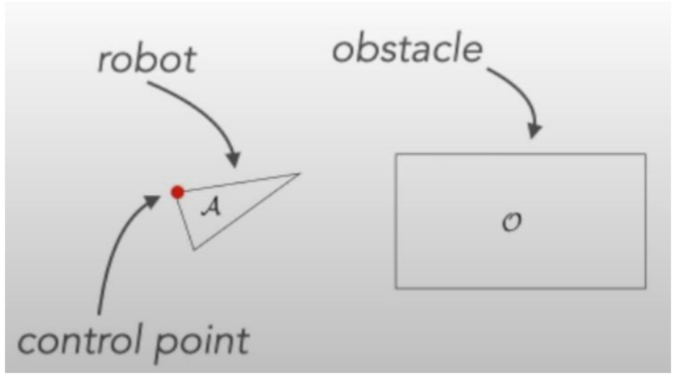
- Configuration space
  - The robot is mapped to **a single point** in C-space
  - Complete specification of robot configuration can be described by a **vector** of generalized joint coordinates
  - Each coordinate can be:
    - an angle (for a rotational joint)
    - a length (for a sliding joint)



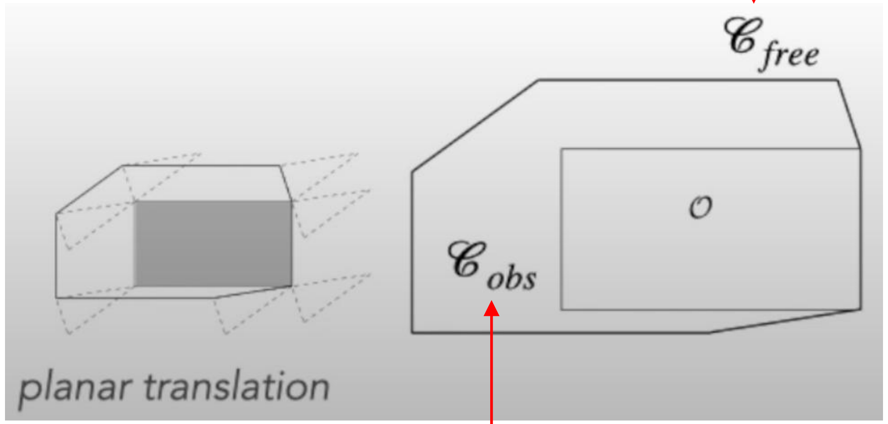


Path Planning

- Configuration space
  - The robot is mapped to **a single point** in C-space
  - Complete specification of robot configuration can be described by a **vector** of generalized joint coordinates
  - Each coordinate can be:
    - an angle (for a rotational joint)
    - a length (for a sliding joint)



Free space

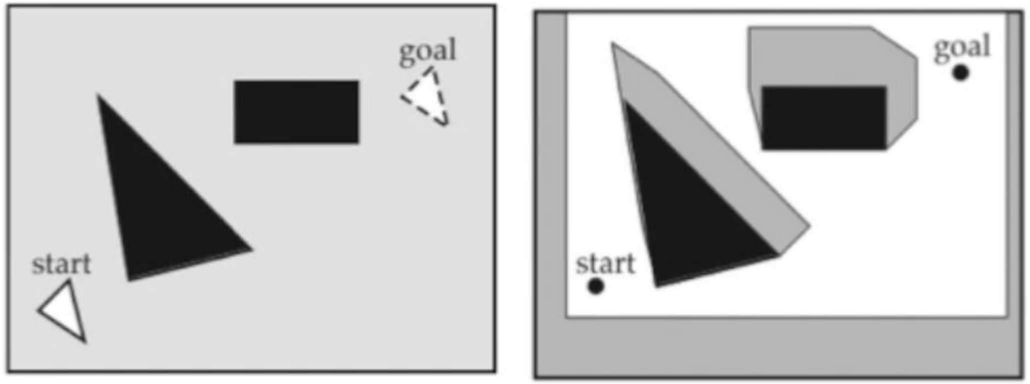


Obstacle space

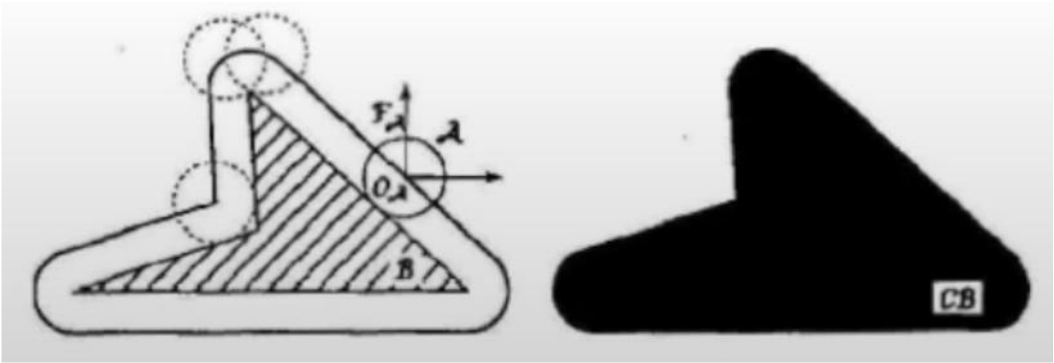
Path Planning

- Configuration space
  - How to compute $\mathcal{C}_{obs}$ (obstacle C-space) and $\mathcal{C}_{free}$ (free C-space)?

Example 1:



Example 2:



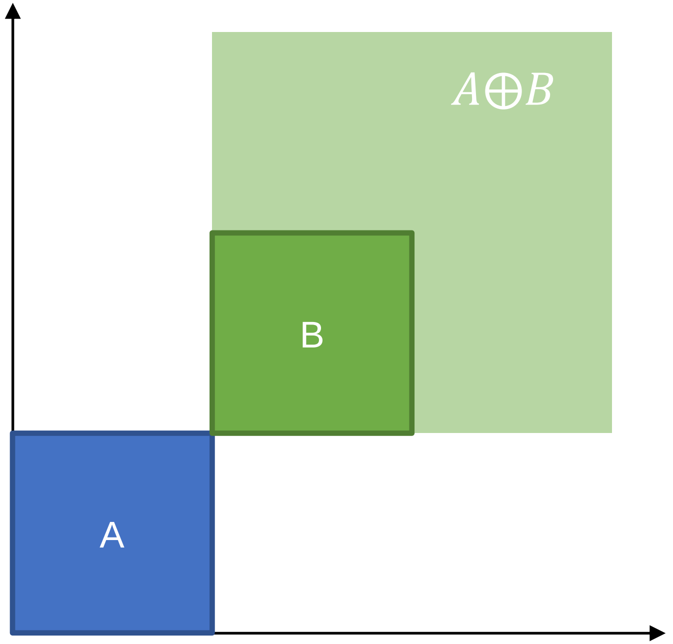
Various methods, e.g., reflection points, <u>**Minkowski sum**</u>, convex Hull.

referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning

- Minkowski Sum
  - In geometry, the Minkowski sum of two sets of position vectors A and B in Euclidean space is formed by adding each vector in A to each vector in B, i.e., the set:

$$A \oplus B = \{a + b | a \in A, b \in B\}$$

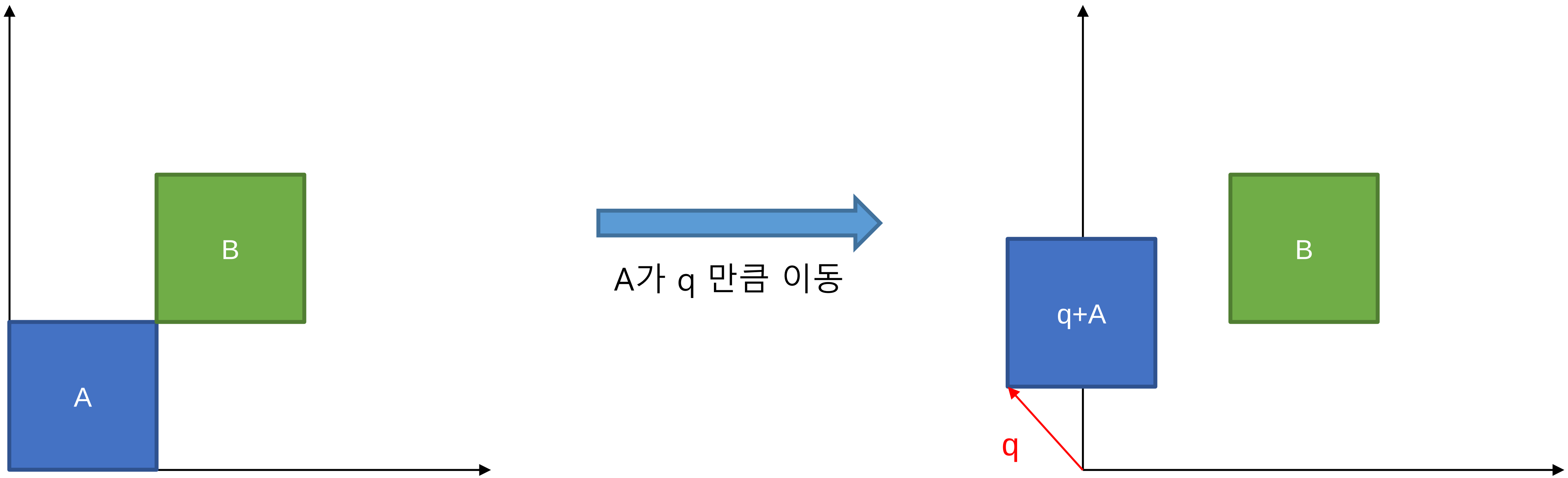


referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning

- Minkowski Sum
  - 로봇 A가 q 위치로 이동:

$$A + q = \{a + q | a \in A\}$$



referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

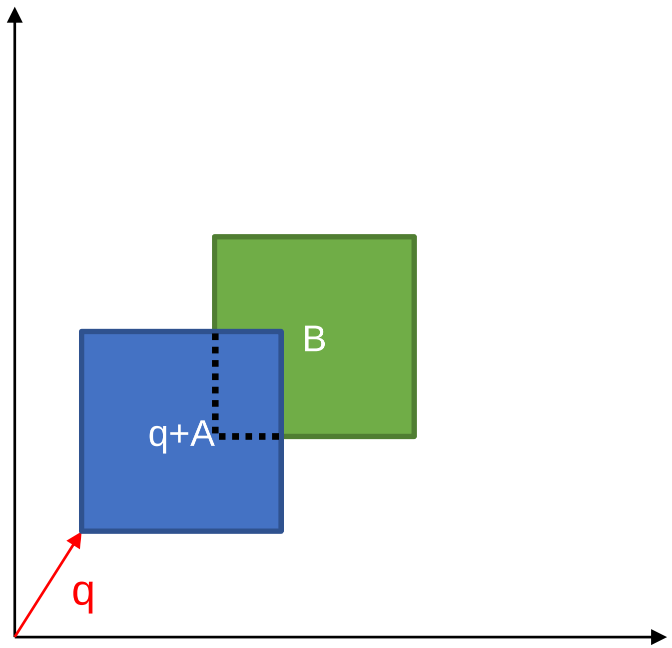
Path Planning

- Minkowski Sum
  - 로봇 A가 q 위치로 이동:

$$A + q = \{a + q | a \in A\}$$

  - 충돌이 발생!

$$A + q \cap B \neq \emptyset$$



referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning

- Minkowski Sum
  - 로봇 A가 q 위치로 이동:

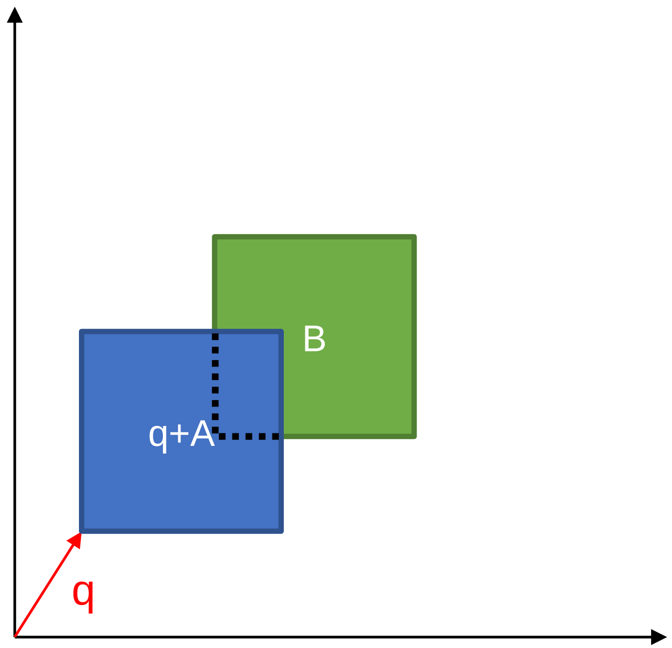
$$A + q = \{a + q | a \in A\}$$

  - 충돌이 발생!

$$A + q \cap B \neq \emptyset$$

  - 즉, A+q와 B에 모두 속하는 q 위치 (vector)가 존재:

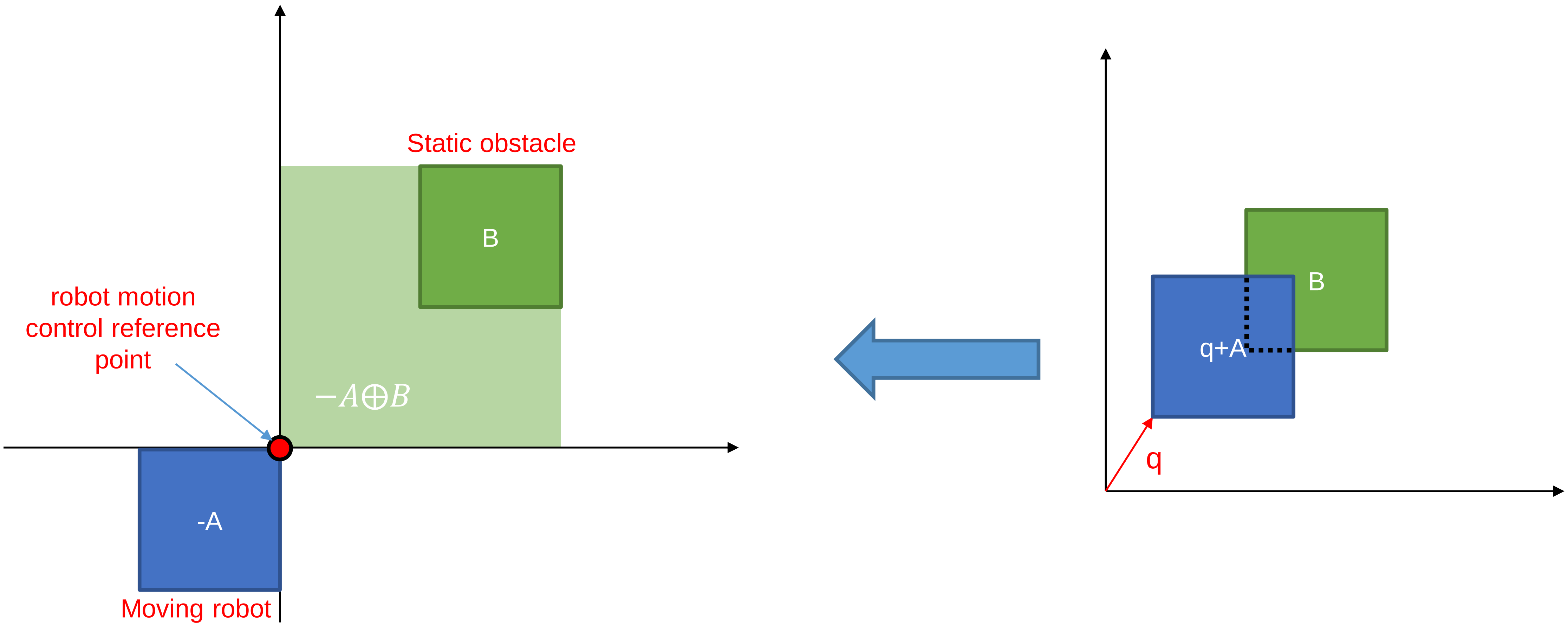
$$a + q = b$$
$$q = b - a$$
$$\{(-a) + b | a \in A, b \in B\} = -A \oplus B$$



referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

Path Planning

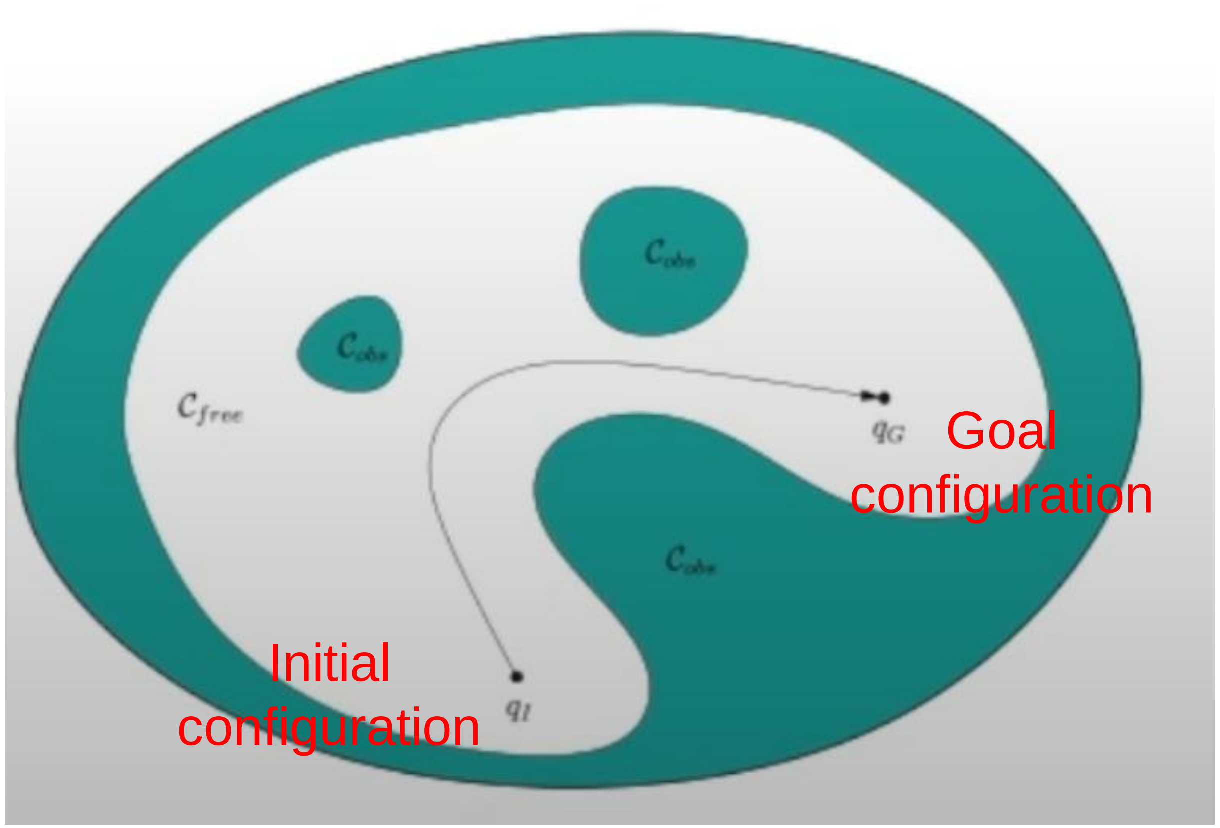
- Occupied C-space



referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

Path Planning

- Assume a workspace, obstacle region, and configuration, with definitions of free and occupied C-spaces



referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning

- Path planning approaches
  - <u>Combinatorial (조합론) planning (exact)</u>
  - Sampling-based planning (Probabilistic)
  - Potential-field methods

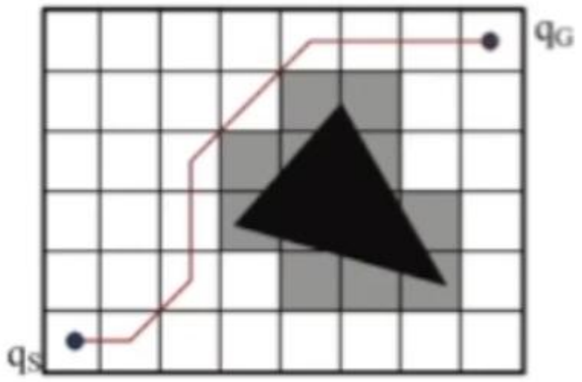
referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning

- Combinatorial methods:
  1. Compute C-space

  2. Generate a roadmap (i.e., a graph) in C-space
     - Cell decomposition methods: visibility graphs / Voronoi cells / <u>occupancy grid maps</u>
     - A valid roadmap guarantees accessibility and is connectivity preserving w.r.t. C-space

  3. Compute the minimum-cost path from initial to goal configuration (cast as a graph search algorithm)



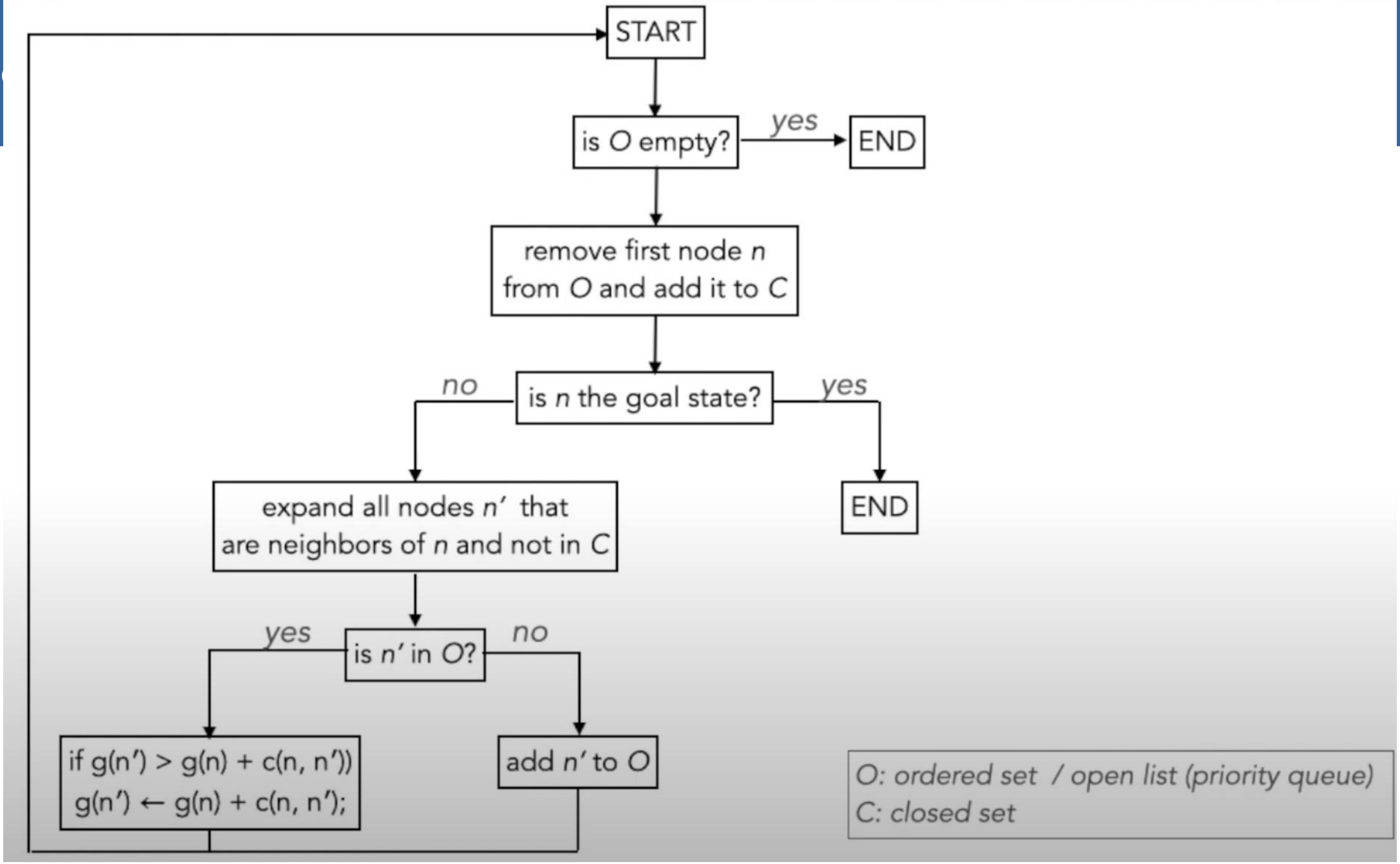
referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning – A* algorithm

- How to search roadmap for minimum-cost path?
- One well-known example: A* algorithm
- Extension of Dijkstra' search algorithm, to reduce number of states explored (exploiting an informed search using a heuristic)
- A* plans path from start state to end state
- Forward search, applied to path planning:
  - Evaluation function: f(n)=g(n)+h(n)
  - Operating cost function g(n): cost of path already traversed
  - Heuristic function h(n): information used to find promising nodes to traverse; heuristic must be **admissible (허용되는)** (i.e., <u>must underestimate true cost: $h(n) \leq h^*(n)$</u>)

referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

P



referred from "Lecture 6 of Dr. Amanda Prorok, University of Cambridge"

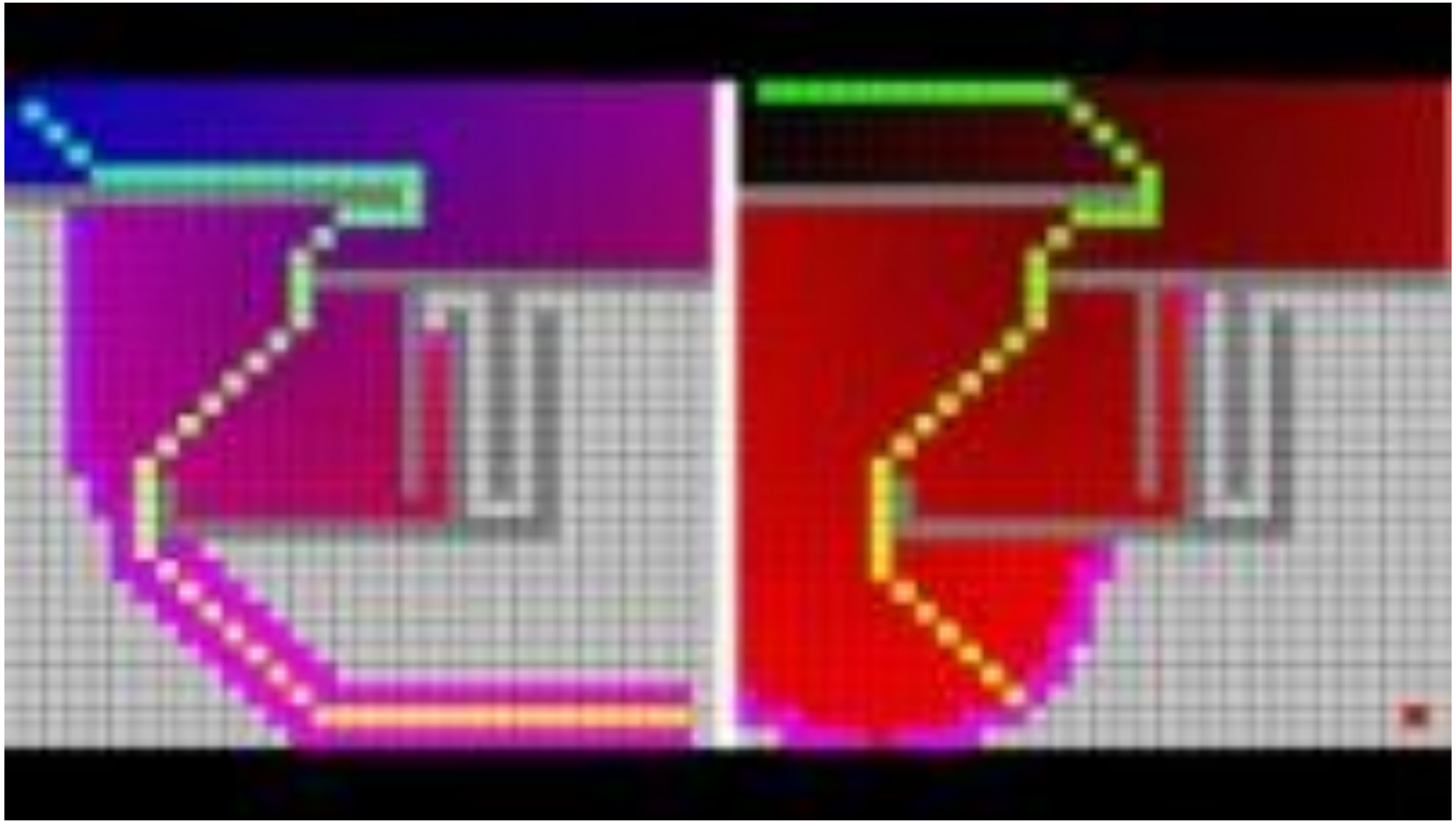
Path Planning – A* algorithm

- Requirements
  - Preprocessing to generate roadmap (connected graph) that represents free C-space
- Pros
  - Optimal path cost and complete
- Cons
  - Memory inefficient
  - Curse of dimensionality

referred from “Lecture 6 of Dr. Amanda Prorok, University of Cambridge”

Path Planning – A* algorithm

- A* algorithm vs. Dijkstra's algorithm



https://www.youtube.com/watch?v=9REexHx0hDY