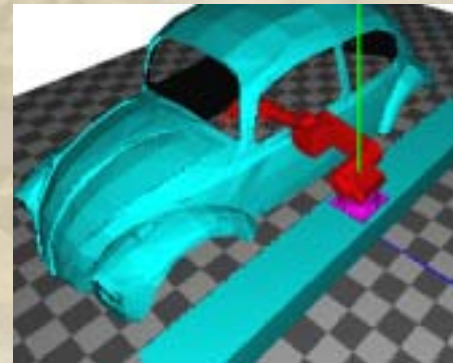
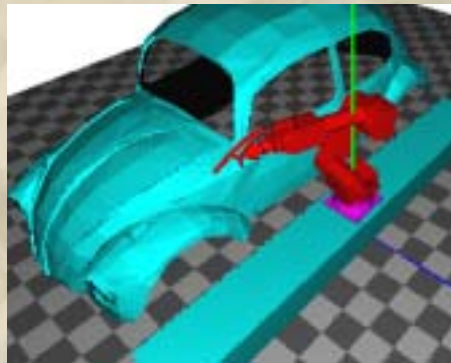


# A Bi-directional Local Search for Robot Motion Planning Problem with Many degrees of Freedom



**\*Shunji Umetani**

The University of Electro-Communications

**Takashi Kurakake, Yasuyuki Suzuki and Masatake Higashi,**

Toyota Technological Institute

# Outline

- Background
- Robot Motion Planning Problem
  - Configuration Space Formulation
- Robot Motion Planning Problem with Many Degrees of Freedom
  - Collision Detection
  - Sampling-based (heuristic) Approaches
  - Bi-directional Local Search Approach
- Computational Results for an articulated robot
- Conclusion and Future Work

# Background

- Autonomous robots have been introduced in a wide variety of applications, ex., industry, construction, space exploration, undersea work, etc.
- Planning the motions of autonomous robots has become too difficult for human operators because of their complex geometry and many degrees of their freedom.



Welding robot system



Assembling robot system



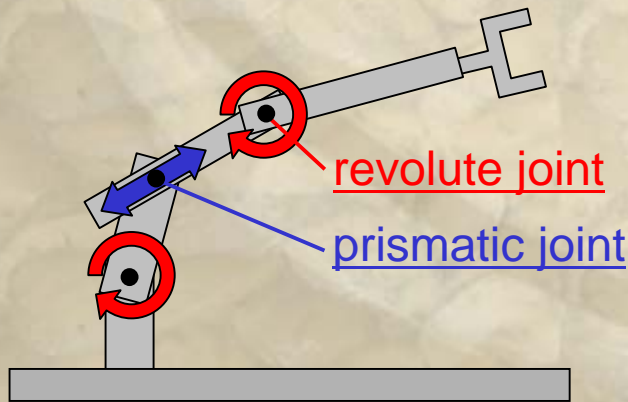
NASA Mars Rover Mission

How to plan desired robot motion automatically?

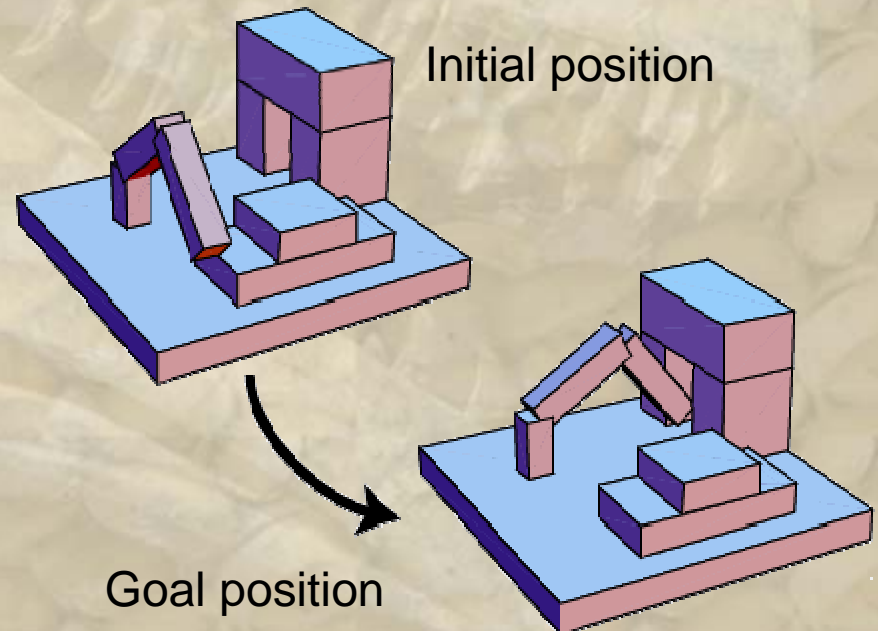
# Robot Motion Planning Problem

**Instance:** Let  $\mathcal{A}$  be a robot moving in a 2D or 3D Euclidean space called *work space*  $\mathcal{W}$ , consisting of a set of obstacles  $\mathcal{S} = \{\mathcal{B}_1, \dots, \mathcal{B}_n\}$ .

**Question:** Given an *initial position*  $q_{\text{init}}$  and a *goal position* of the  $q_{\text{goal}}$  robot, find a path  $\mathcal{P}$  specifying a **continuous sequence of positions** of the robot avoiding collision with the obstacles. If no such path exists, report failure.

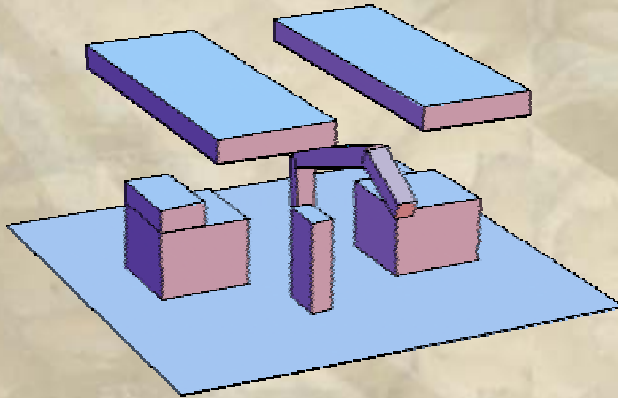


The robot has polyhedral rigid body and several degrees of freedom such as *translational* and *rotational* movement.

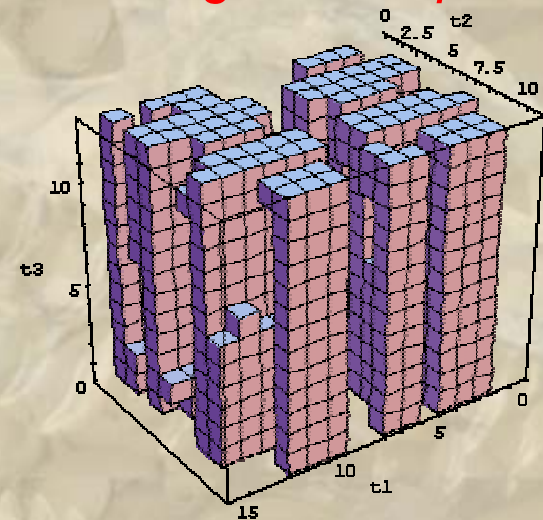
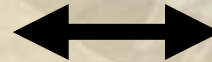


# Configuration Space Formulation

- A configuration of a robot is specified by a number of parameters corresponding to the number of freedom.
- The parameter space of a robot is called its *configuration space*  $\mathcal{C}$  (*C-space*)



3-dof articulated robot and obstacles



Corresponding C-space

*C-obstacle* ( $\mathcal{C}_{\text{obs}}$ ): the part of C-space that the robot collides with obstacles  
*Free space* ( $\mathcal{C}_{\text{free}}$ ): the complement of C-obstacle, i.e.,  $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$

The C-obstacle forms much more complicated geometry than the original work space.

# Robot Motion Planning Problem with Many Degrees of Freedom

- Robot motion planning problem for an articulated robot with multiple joint is a *PSPACE-hard* problem [Reif,1979]
- Constructing an explicit description of C-obstacle (ex., No-fit-polygon, Voronoi regions, etc.) requires unrealistic computational time for most instances.

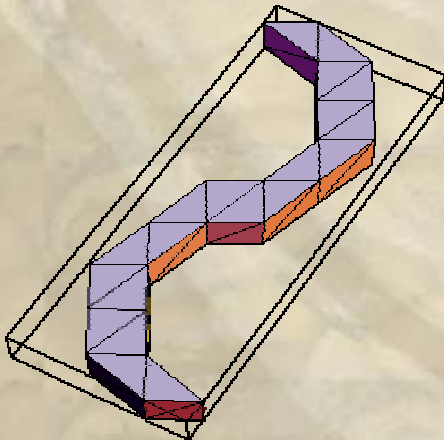


- Sampling-based (heuristic) approach
  - Find a collision-free path based on information of limited number of configurations in the C-space, ex., Randomized path planner (RPP), Probabilistic roadmap planner (PRM), Rapidly-exploring random tree (RTT), etc.
  - Computational time of most motion planners depend on the total number of evaluated configurations, i.e., the number of collision detections.

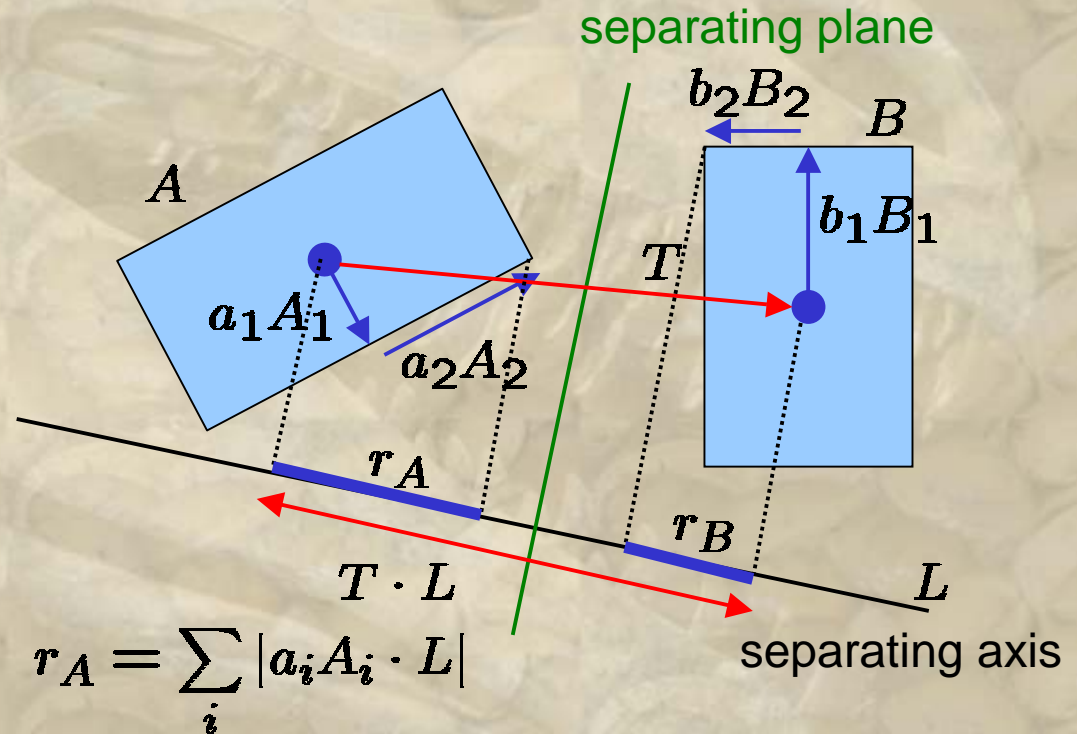
How to reduce the number of collision detections?

# Collision Detection (1)

- Enclose a part of the object by a simple volume, for example, *oriented bounding box (OBB)*.
- We can quickly check collision between a pair of OBBs using separating axis.



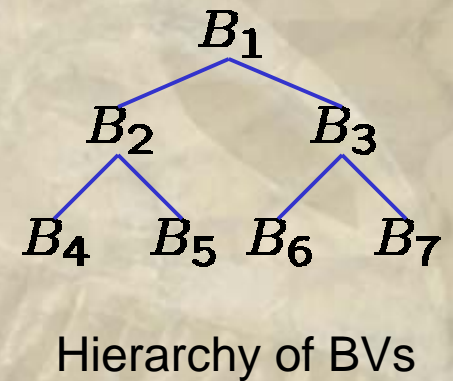
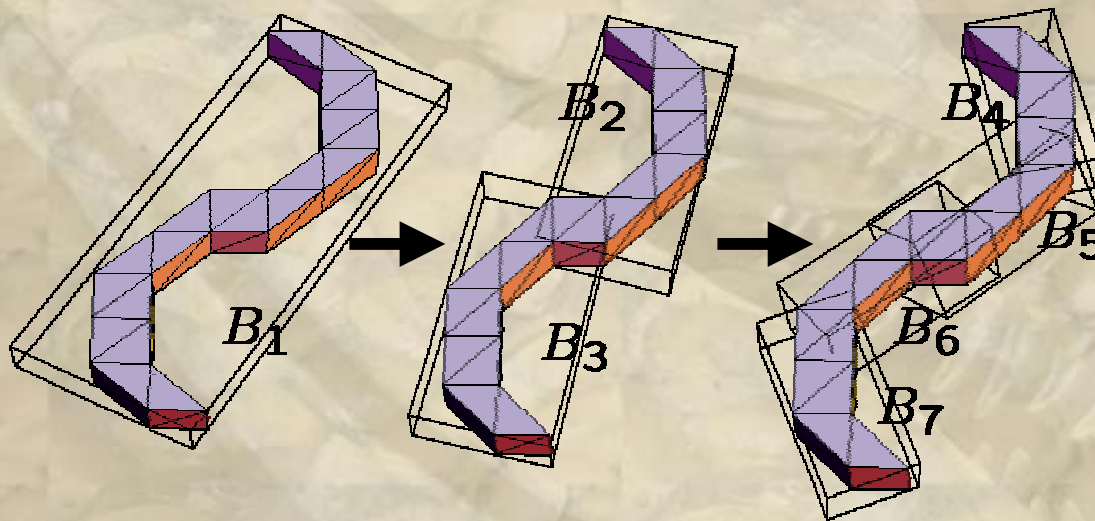
oriented bounding box (OBB)



If  $|T \cdot L| > r_A + r_B$  holds, OBBs A and B are not overlapped.

## Collision Detection (2)

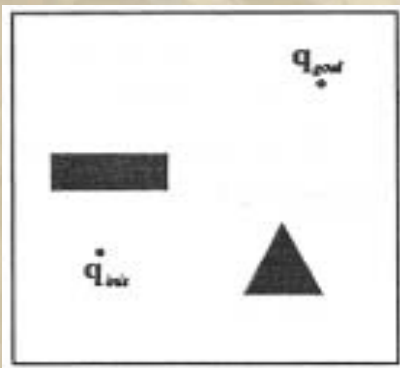
- Construct a hierarchical structure of bounding volume, called *bounding volume tree (BVT)*.



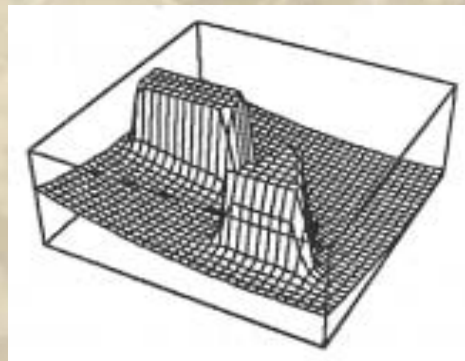
BVT approach performs well when a pair of objects is away from each other.

# Randomized Path Planner (RPP)

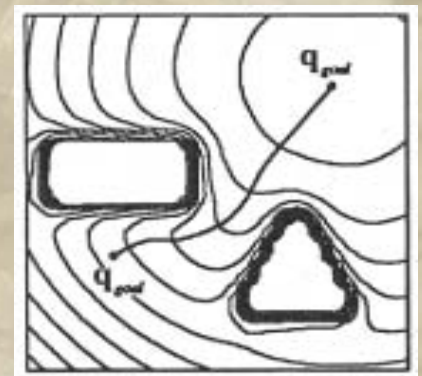
- Define a potential field over free space  $C_{\text{free}}$ 
  - Attracting potential of the goal configuration
  - Repulsing potentials of the obstacles
- Repeat descending moves in the potential field until it reaches a locally minimal point
  - If the local minimal point is not the goal configuration, execute a series of random walk to escape the point.



Original C-space



Define potential field

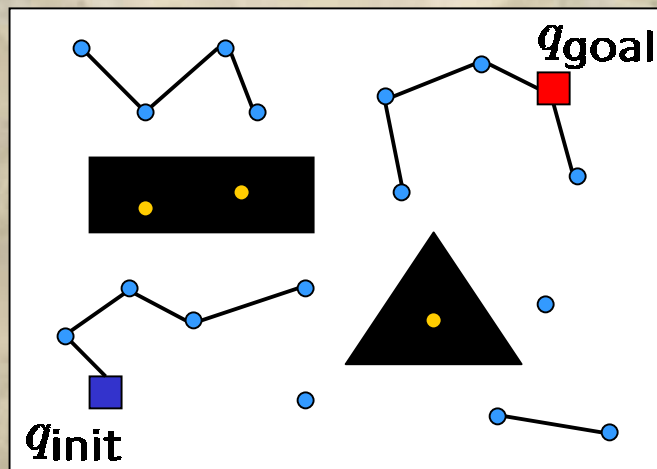


Apply gradient method

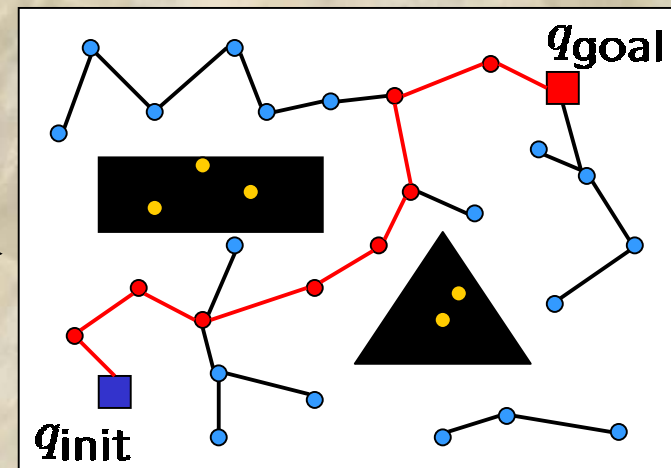
Since potential field has many local minimal points in general, RPP often performs poorly for practical motion planning problem.

# Probabilistic Roadmap Planner (PRM)

- Construct a roadmap by generating a number of random configurations as subgoals, and connecting pairs of subgoals in a neighborhood by simple paths called local paths.
- Finding a collision-free path by solving the shortest path problem on the roadmap.



Repeat generating subgoals and local paths

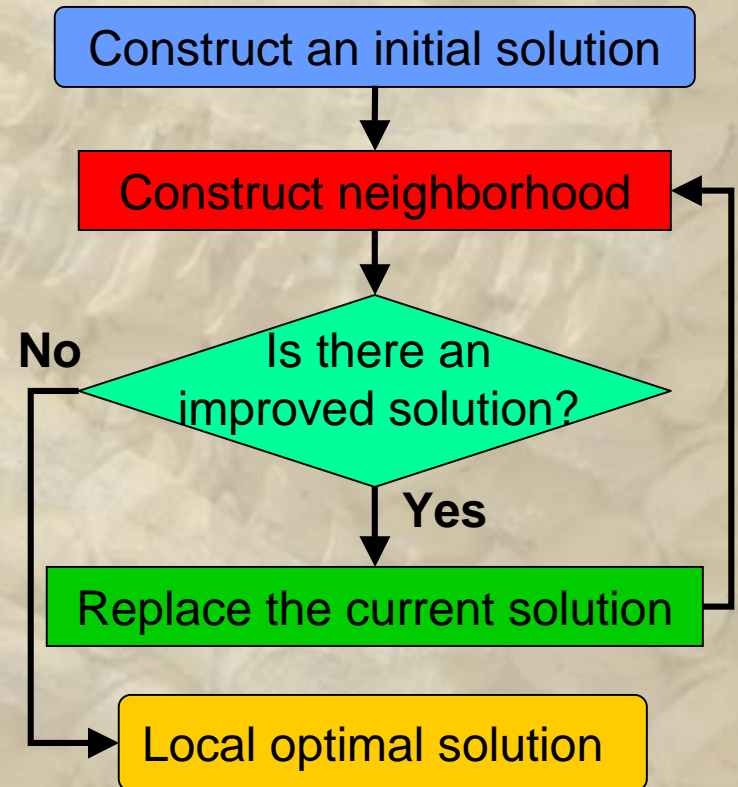
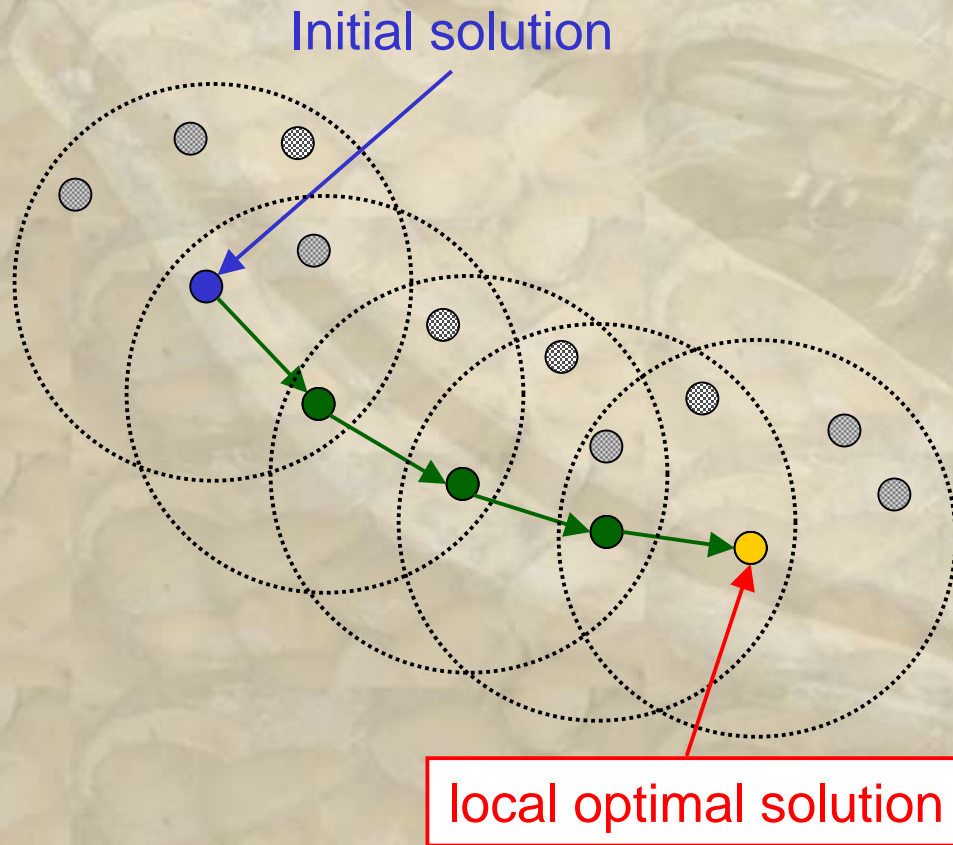


Find a collision-free path on the roadmap

PRM needs to sample a large number of configurations to find a path through narrow passages

# Local Search

- LS starts an initial solution  $x$  and repeats replacing it with a better solution  $x'$  in its neighborhood  $N(x)$  until no better solution is found in the neighborhood.



# Local Search for Robot Motion Planning (1)

- We are given an articulated robot  $\mathcal{A}$  with  $m$  revolute joints and  $n$  obstacles in 3D workspace.
- We consider a discretized C-space specified by angles of revolute joint  $(\theta_1, \theta_2, \dots, \theta_m)$ , and a configuration  $q \in \mathcal{C}$  of the robot  $\mathcal{A}$  is represented by  $q = (\theta_1(q), \theta_2(q), \dots, \theta_m(q))$

- Evaluation function for a configuration

$$d(q, q_{\text{goal}}) = \sum_{i=1}^m |\theta_i(q) - \theta_i(q_{\text{goal}})|$$

- Neighborhood of a configuration

$$N(q) = \{q' \in \mathcal{C} \mid d(q, q') \leq 1\}$$

}  $d(q, q')$  is Manhattan distance

- Collision detection for a configuration

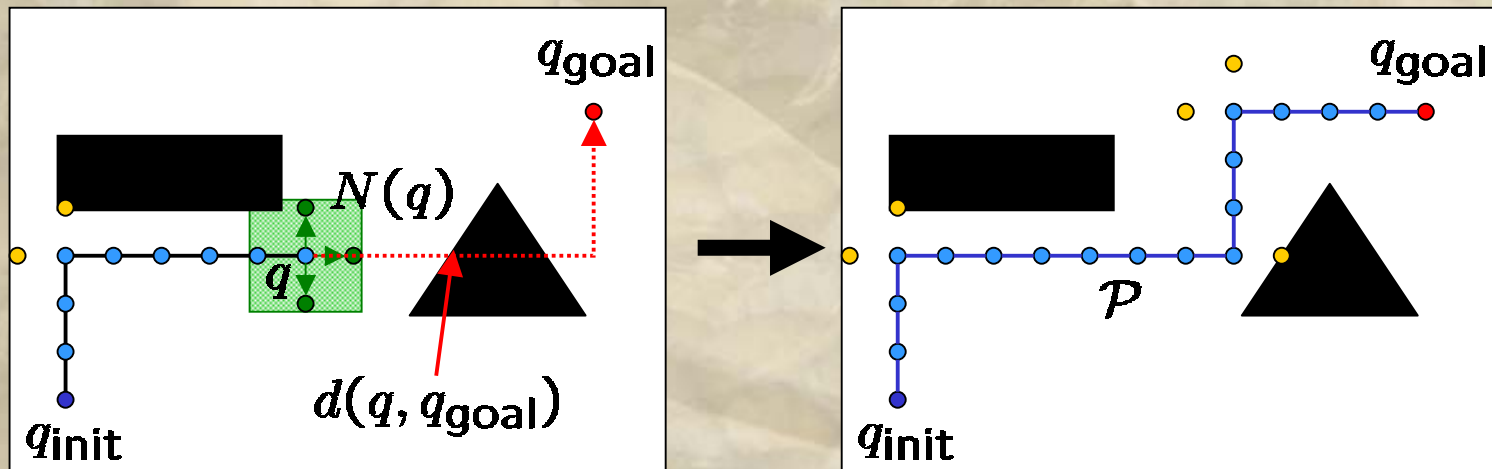
Use the proximity query checker package (PQP) based on OBB Trees

# Local Search for Robot Motion Planning (2)

**Step1:** Set  $q := q_{init}$  and  $\mathcal{P} := (q_{init})$

**Step2:** If  $d(q, q_{goal}) = 0$  holds, output the collision-free path  $\mathcal{P}$  and halt

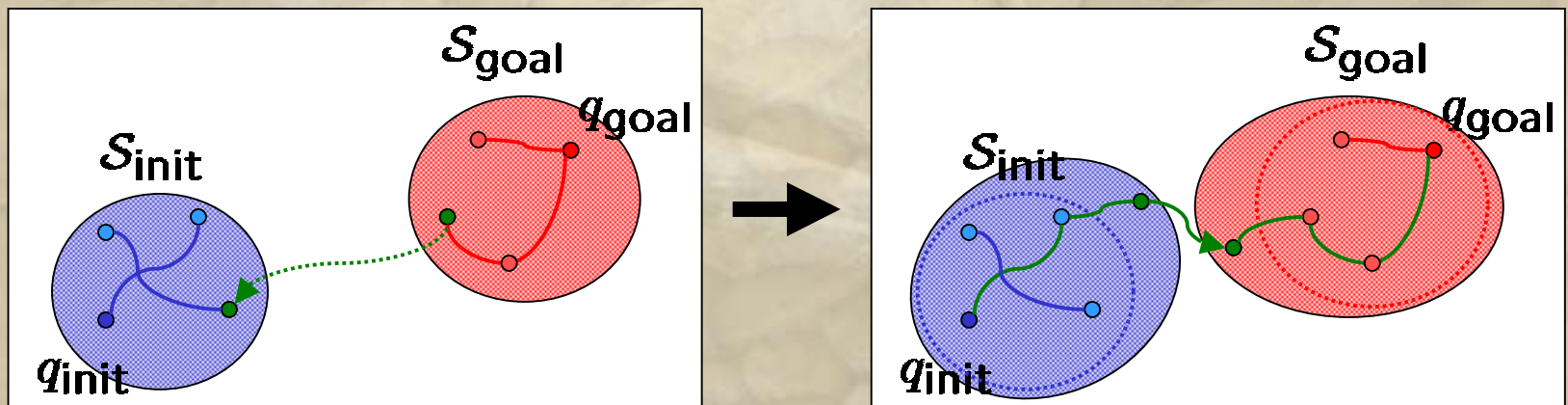
**Step3:** If there is a collision-free configuration  $q' \in N(q)$  such that  $d(q', q_{goal}) < d(q, q_{goal})$  holds, set  $q := q'$  and add  $q$  to  $\mathcal{P}$ .  
Otherwise output failure and halt.



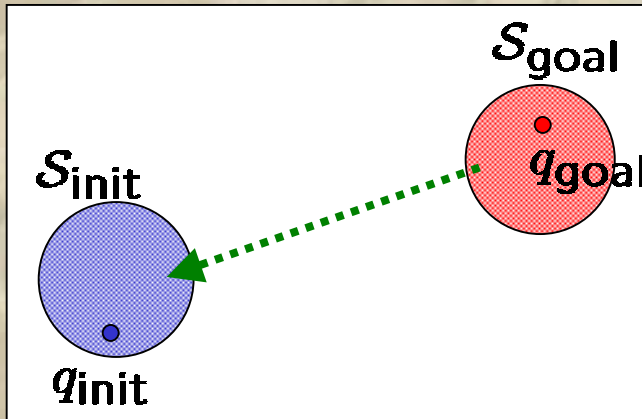
LS applies the collision check only to configurations  $q' \in N(q)$  satisfying  $d(q', q_{goal}) < d(q, q_{goal})$

# Bi-Directional Local Search (1)

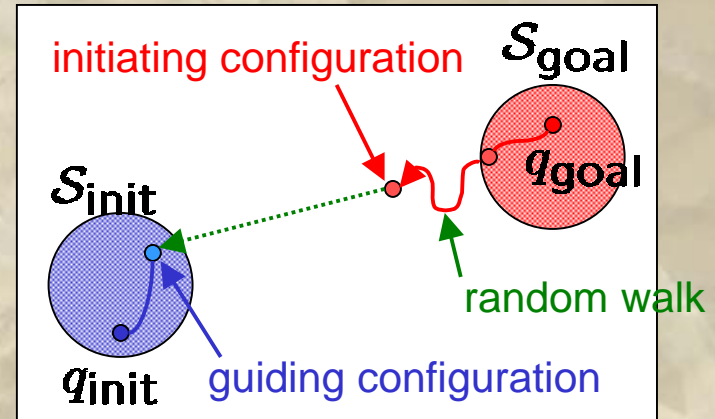
- Simple local search often falls into local minima.
  - Local search often finds a collision-free path when we exchange the start configuration and the goal configuration.
- ↓
- Bi-directional Local Search (BLS)
    - A variant of path re-linking approach
    - Any configuration  $q \in \mathcal{S}_{\text{init}}$  (resp.,  $q \in \mathcal{S}_{\text{goal}}$ ) has at least a collision-free path from  $q_{\text{init}}$  (resp.,  $q_{\text{goal}}$ )
    - Repeat simple local search for different pairs of an initiating configuration  $q'_{\text{init}} \in \mathcal{S}_{\text{init}}$  and a guiding configuration  $q'_{\text{goal}} \in \mathcal{S}_{\text{goal}}$



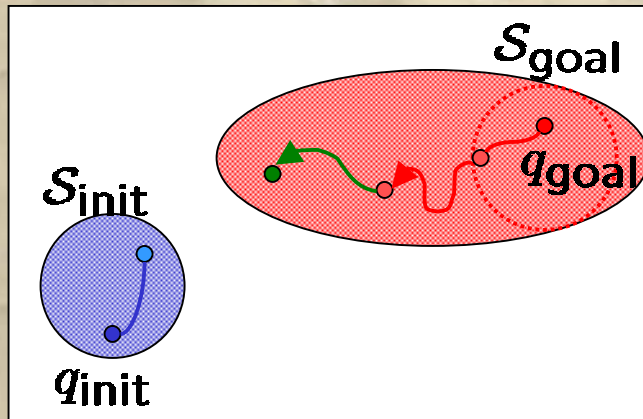
# Bi-directional Local Search (2)



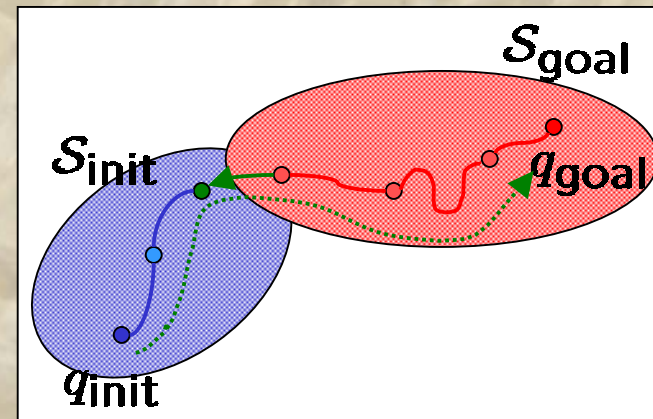
Select an initiating set with probability  $\frac{1}{2}$



Select a guiding configuration randomly, and set an initiating configuration



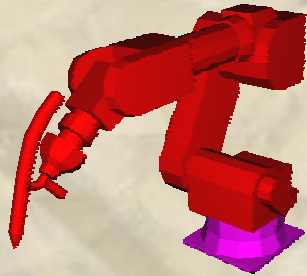
Apply local search and add a new path to the initiating set



If  $S_{init}$  and  $S_{goal}$  are connected, find a collision-free path

# Computational Experiments

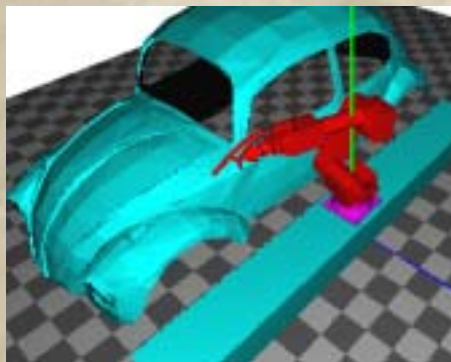
- We tested simulated annealing (SA), iterated local search (ILS), and bi-directional local search (BLS)
- We tested them on several instances that represent a 6-dof welding articulated robot handling a body shell of a car



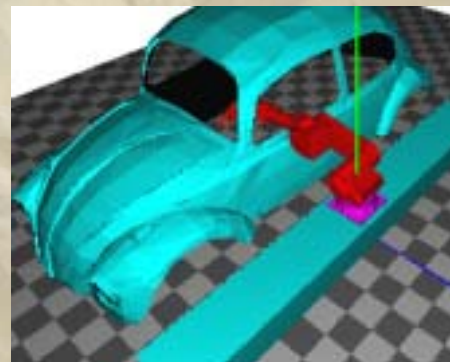
6-dof articulated robot (987 triangles)



body shell of a car (2069 triangles)



initial position



goal position

# Computational Results

- The C-space is discretized by 2.5 degrees for each joint
- We tested all algorithms 20 times for the instance, where CPU time is limited within 300 seconds.

	Num of conf in the path	Total num of evaluated conf	CPU time (seconds)
SA	623	11966	0.73
ILS	289	18556	1.15
<b>BLS</b>	<b>295</b>	<b>9128</b>	<b>0.79</b>

## Observations

1. ILS evaluated larger number of configurations than those of SA and BLS
2. BLS and ILS generated shorter collision-free path than that of SA
3. In other instances, only BLS found collision-free paths for all 20 runs

Diversification oriented metaheuristics are more effective than intensification oriented ones

# Conclusion and Future Work

- Robot Motion Planning Problem
  - Configuration Space Formulation
- Collision Detection
  - Bounding Volume Tree
- Sampling-based Approach
  - Randomized Path Planner (RPP)
  - Probabilistic Roadmap Planner (PRM)
- Metaheuristics Approach
  - Local Search
  - Bi-directional Local Search
- Computational Experiments

1. More detailed computational experiments to compare the sampling-based approaches
2. How to explore the continuous C-space?