# Introduction to Robotics

Amitabha Mukerjee

IIT Kanpur, India

Readings:
R&N 3d ed.

ch.25
25.1 to 25.4, 25.6

25.4 does not include PRM: pls follow notes

# What is a Robot?  Mobile Robots

Robot properties:

- Flexibility in Motion
  - Mobile robots

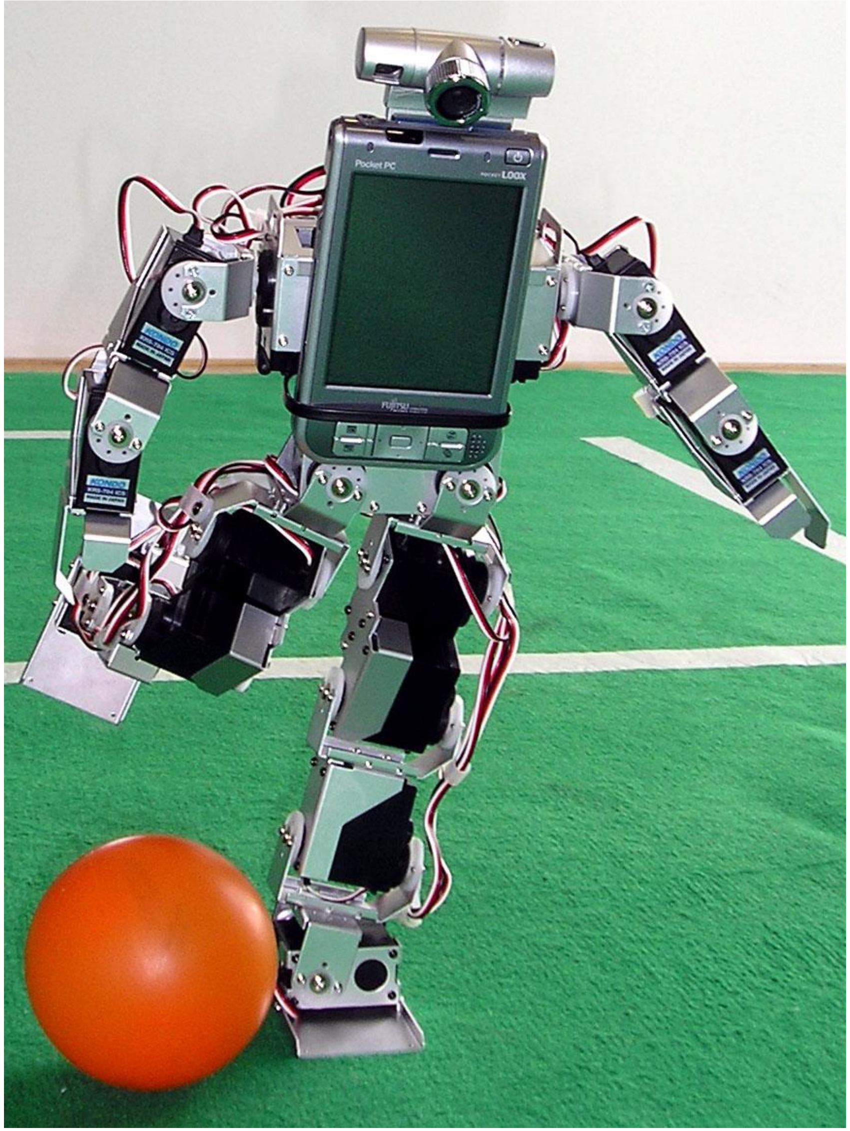

daksh ROV: de-mining robot
20 commissioned in Indian army 2011.
100+ more on order
built by R&D Engineers, Pune

daksh platform derived
gun mounted robot (GMR)

# What is a Robot? Articulated Robots

Robot properties:

- Flexibility in Motion
  - Mobile robots
  - Articulated Robots




Soccer playing humanoid robot
[http://labintsis.com

# Robot you can own


Roomba vacuum
Cleaning robot

By i-robot
Price: ~ rs. 15-30K

# Algorithms for Robot motion

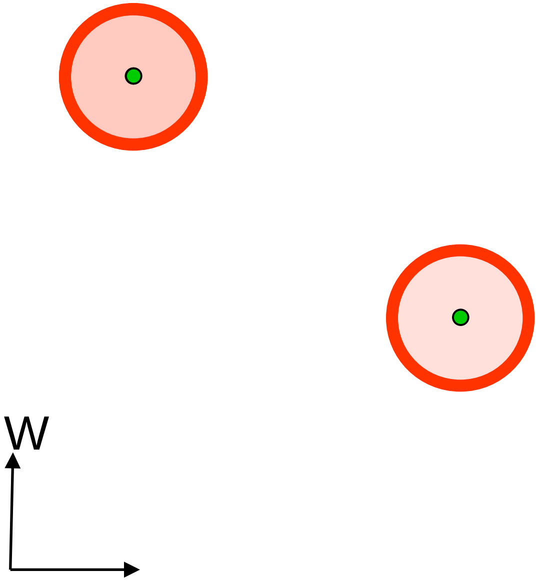
Roomba vacuum
Cleaning robot

By i-robot
Price: ~ rs. 30K

https://www.youtube.com/watch?v=dweVBqei9LA

# Models of Robot Motion

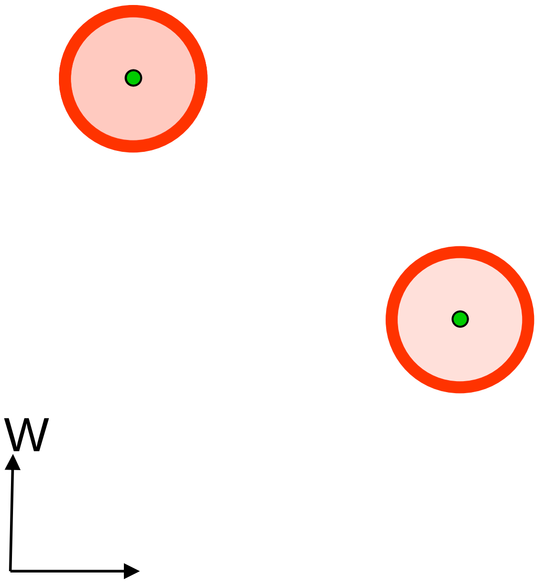
Circular robot




World Frame
(Workspace frame)

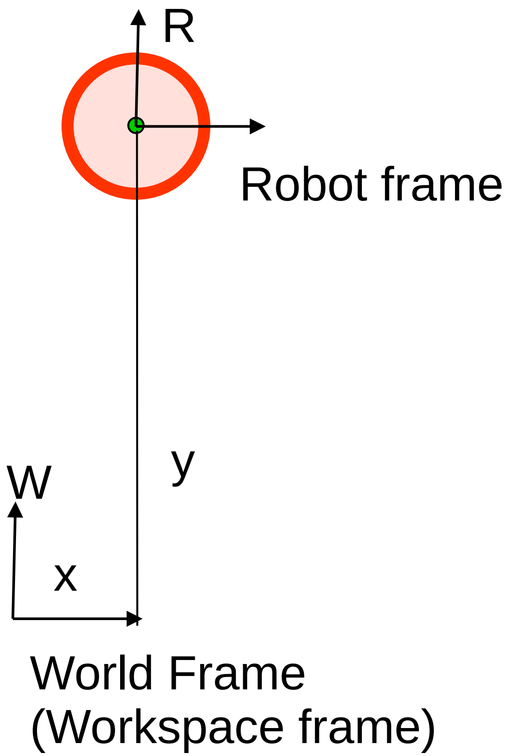
# Models of Robot Motion

Circular robot




World Frame
(Workspace frame)

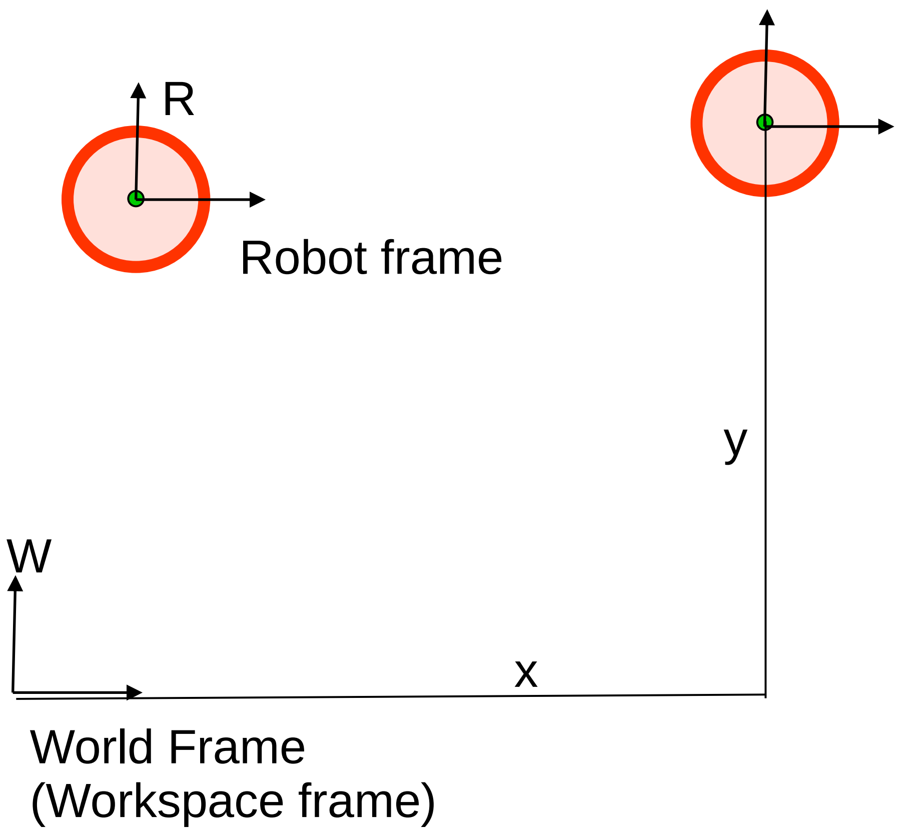
# Models of Robot Motion




DEFINITION:
**degrees of freedom**:
number of parameters needed to fix the robot frame R in the world frame W

(x,y) = **configuration** (vector **q**)

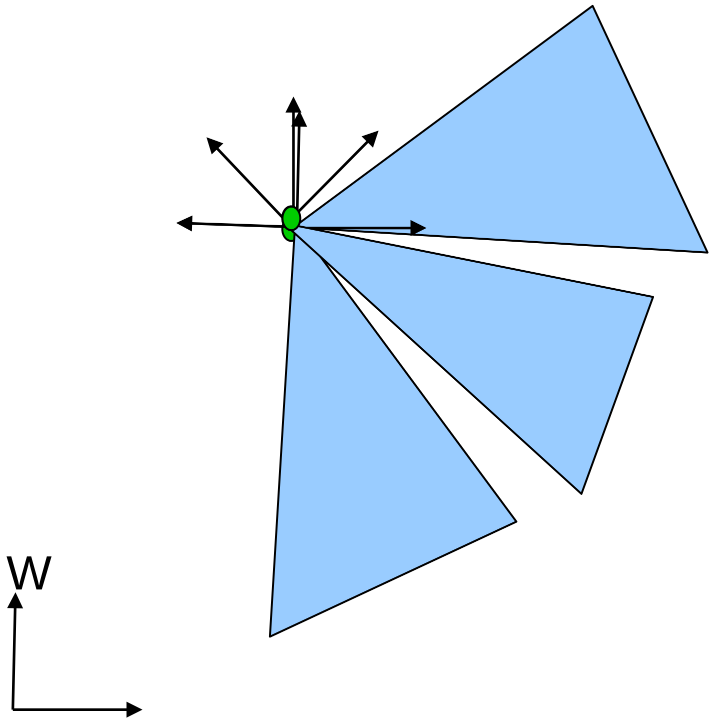
# Models of Robot Motion




NOTE:
Given robot frame R, every point on the robot is known

given configuration **q** for a certain pose of the robot, the set of points on the robot is a function of the configuration: say R(**q**)

# Non-Circular Robot




DEFINITION:
**degrees of freedom**:
number of parameters needed to fix the robot frame R in the world frame W

**Configuration** vector **q** : (x,y, θ)

How many parameters needed to fix the robot frame if it can translate in 3-D?

How many if it can rotate as well?

# Mobile robot

Turtlebot

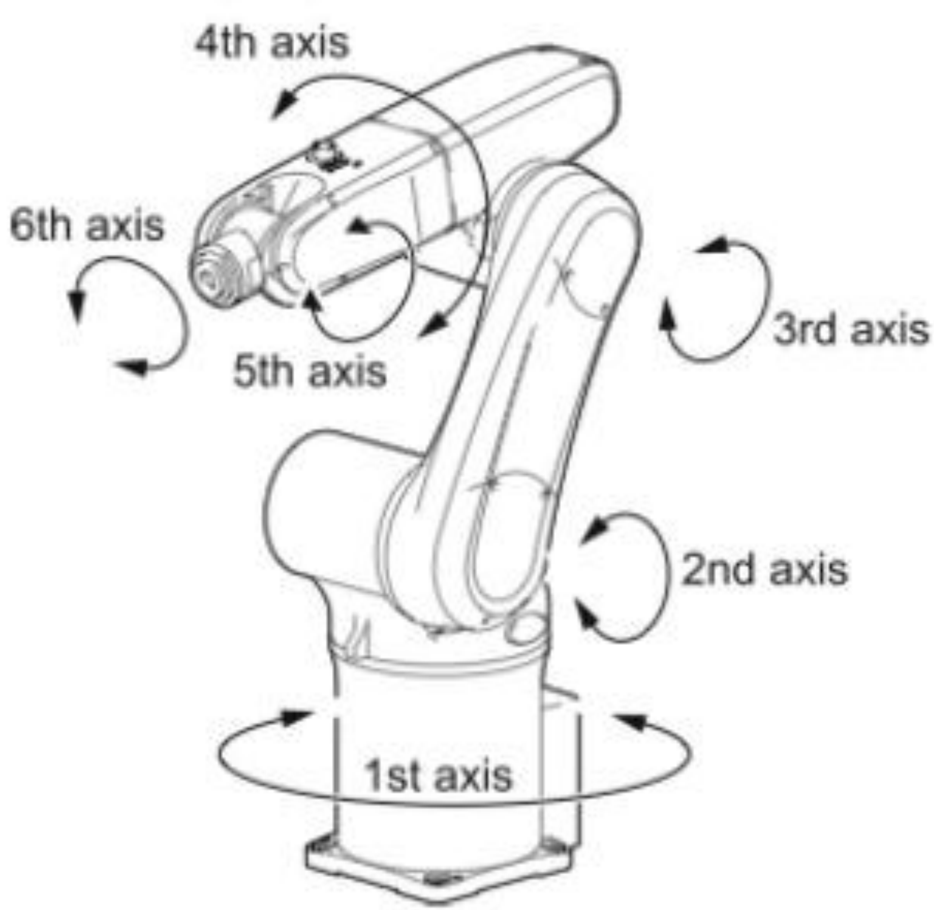
Based on i-robot (roomba) platform
(with kinect RGB-D sensor)

Configuration: **q** : $(x,y,\theta)$

ROS (open-source) software

# Articulated robots

# Articulated Robots




**Kinematic chain:**

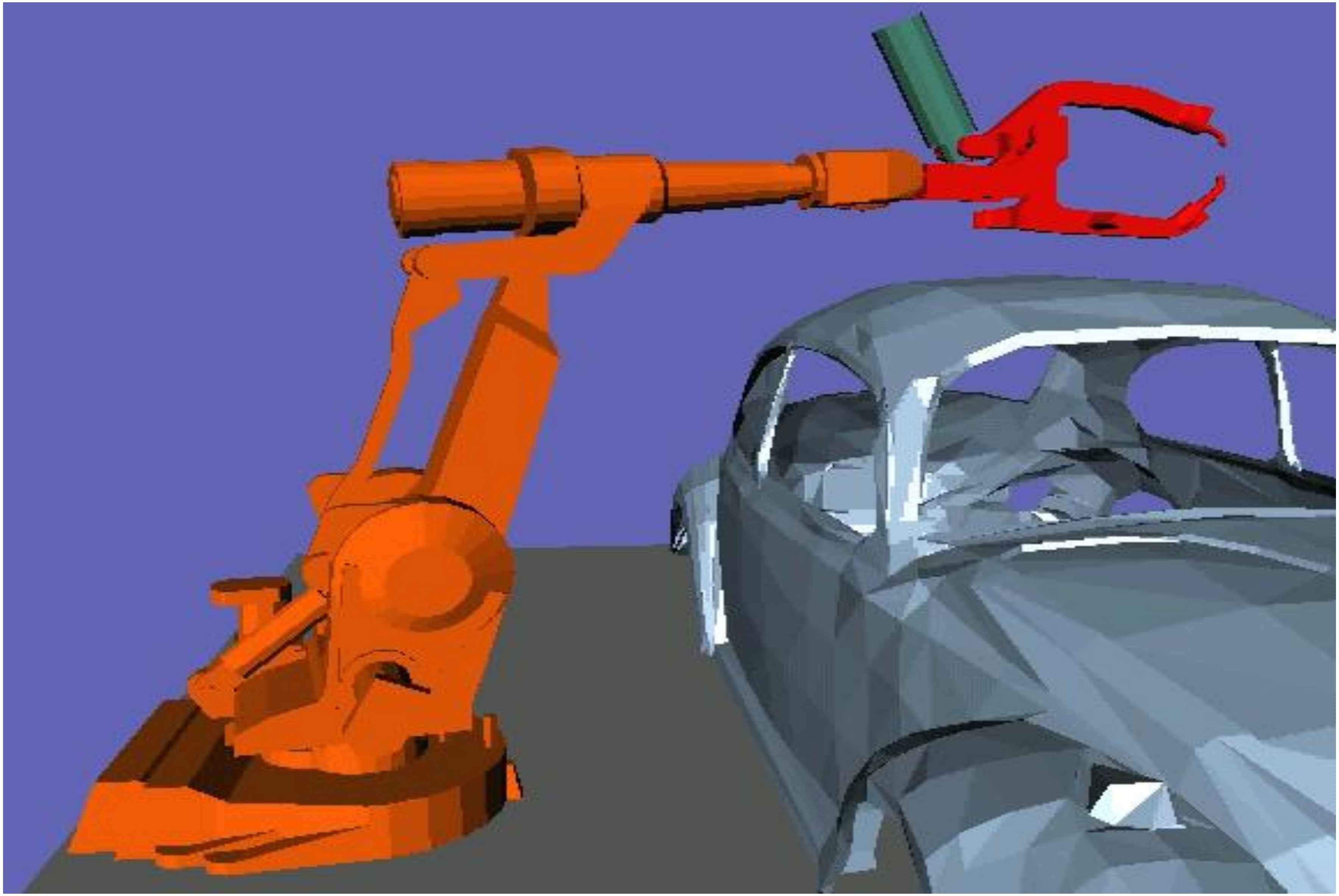
Pose of Link n depends on the poses of Links 1...(n-1)

This industrial robot arm has 6 rotation joints.

Six DOFs =>

$\mathbf{q} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$

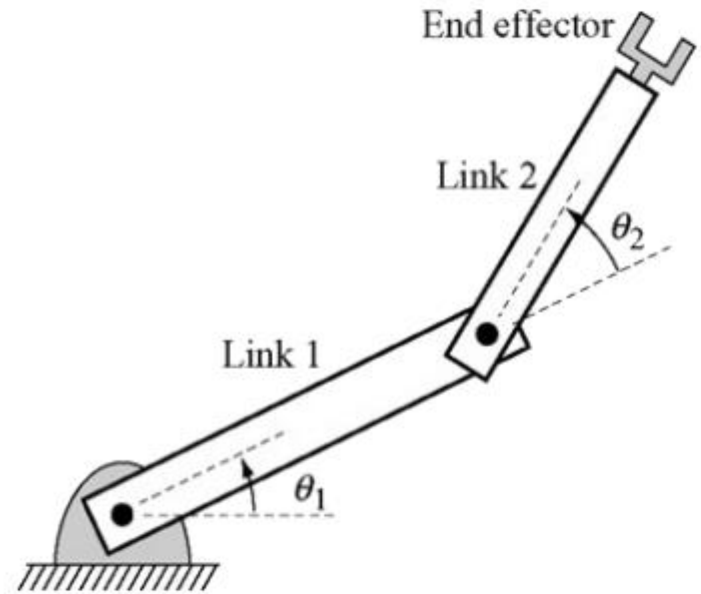
# How to program a welding robot?



# Articulated Robots

This robot has TWO articulated chains

# Modeling Articulated Robots



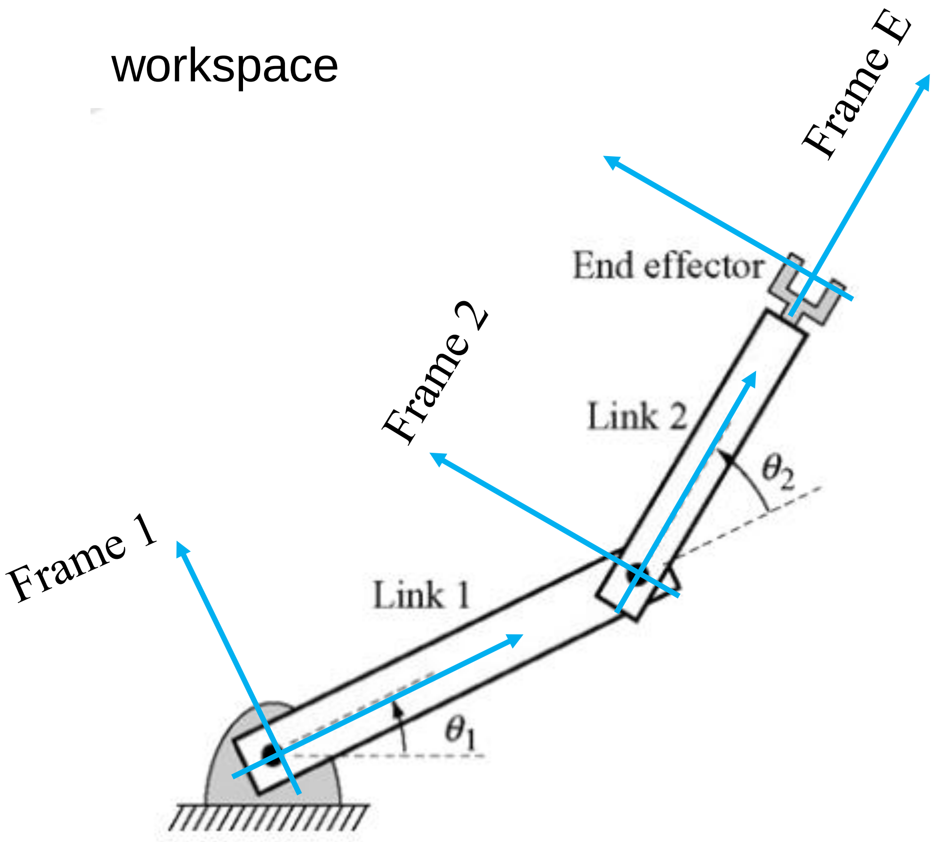

**Kinematic chain:**
Pose of Link n depends on the poses of Links 1...(n-1)

Transformation between frame of link (n-1) and link n, depends on a single motion parameter, say $\theta_n$

Exercise:
What are the coordinates of the end-effector center?

Exercise:
Sketch the robot pose for the configuration [0, -90]

# Fixing frames




**Link Frames:**

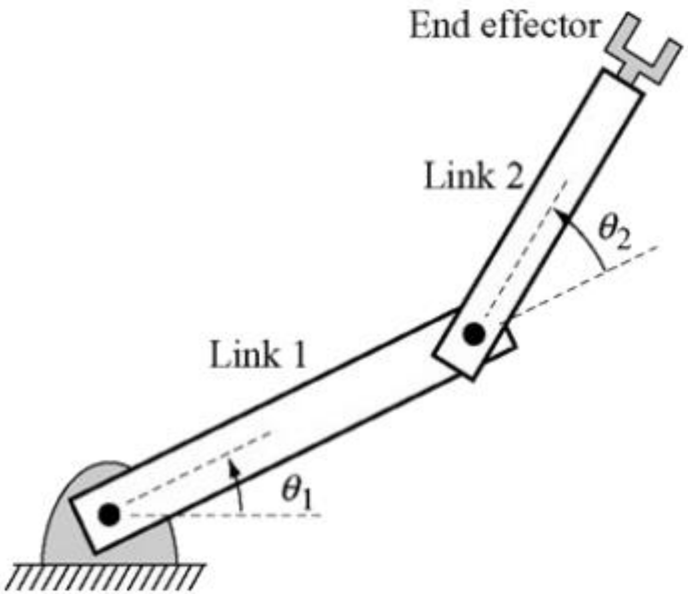
Fix $\text{frame}_n$ on Link n.
Every point on the link is rigidly fixed to $\text{frame}_n$.

$\text{Link}_n$ pose is fully determined given $\theta_1 \ldots \theta_n$

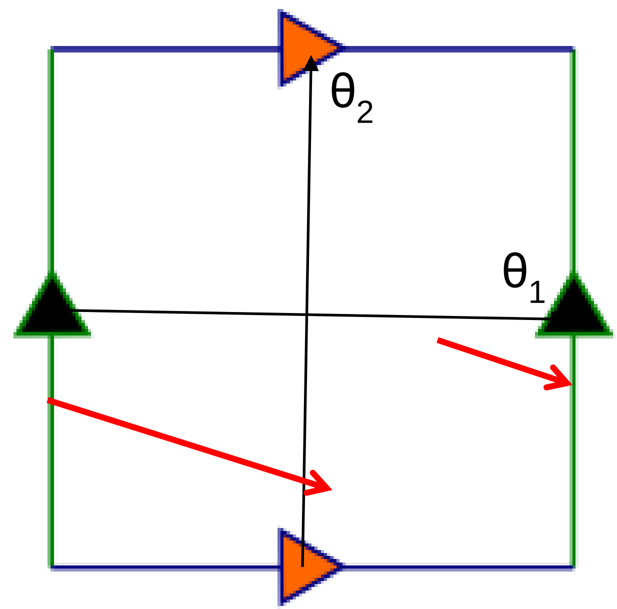
*R(q)* = set of points in robot in configuration *q*.

# Configuration Spaces

workspace



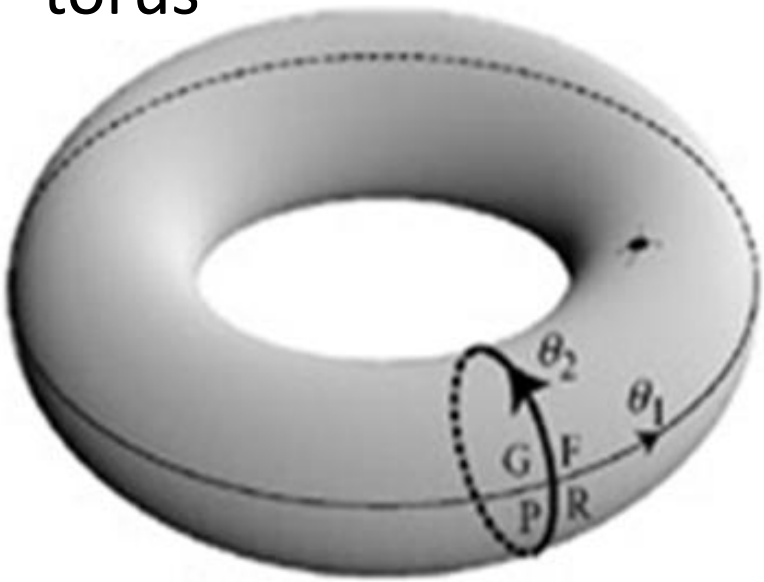

configuration space

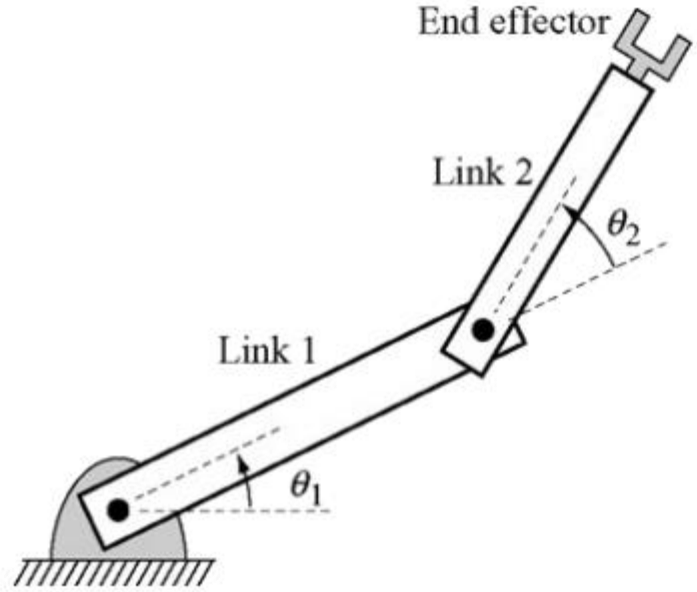



What is the nature of the C-space if $\theta_1$, $\theta_2$ can rotate all around?

# C-space as manifolds

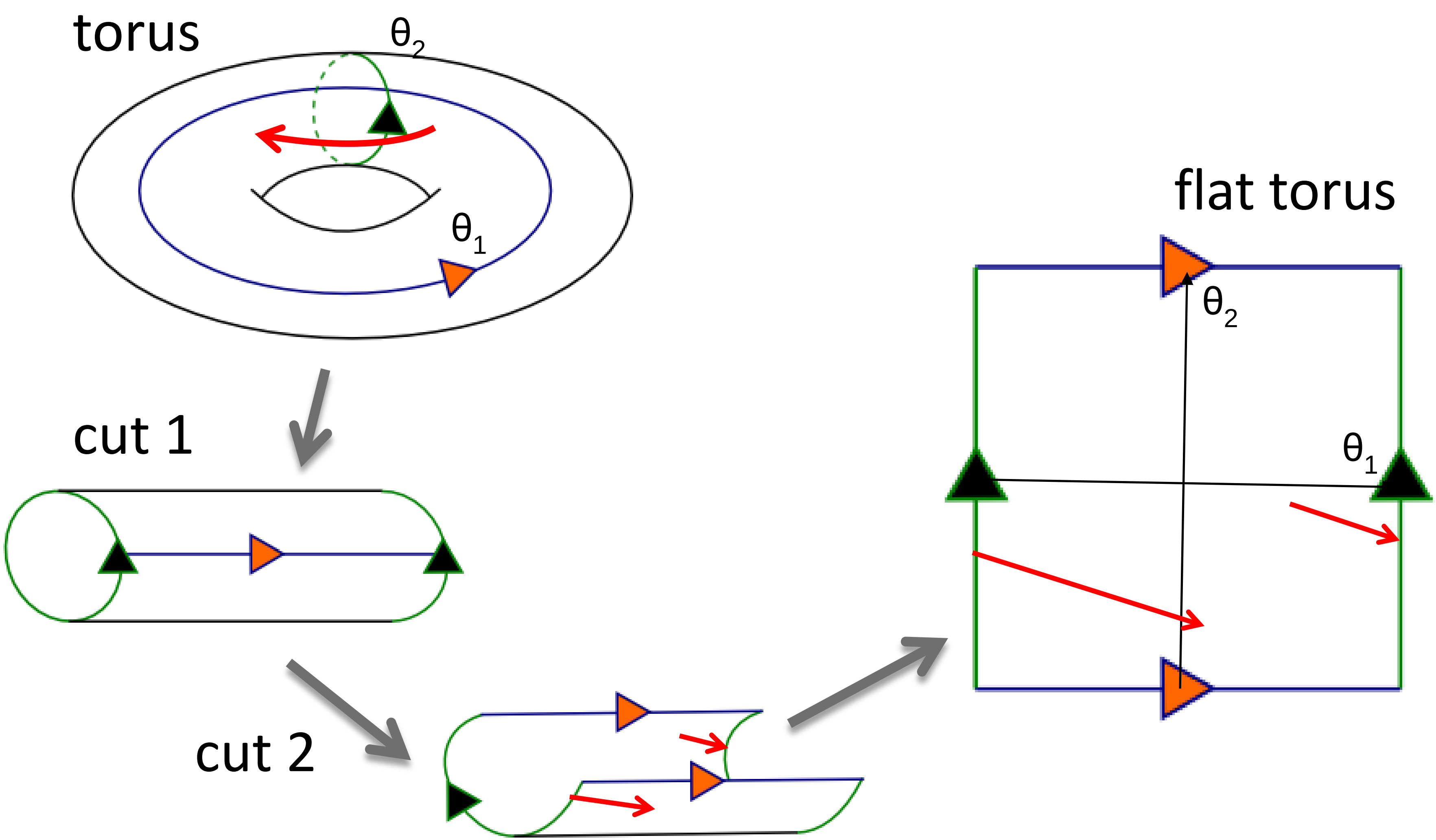
torus



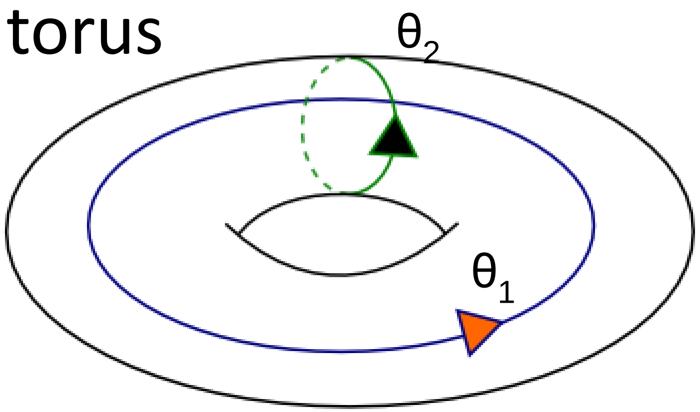



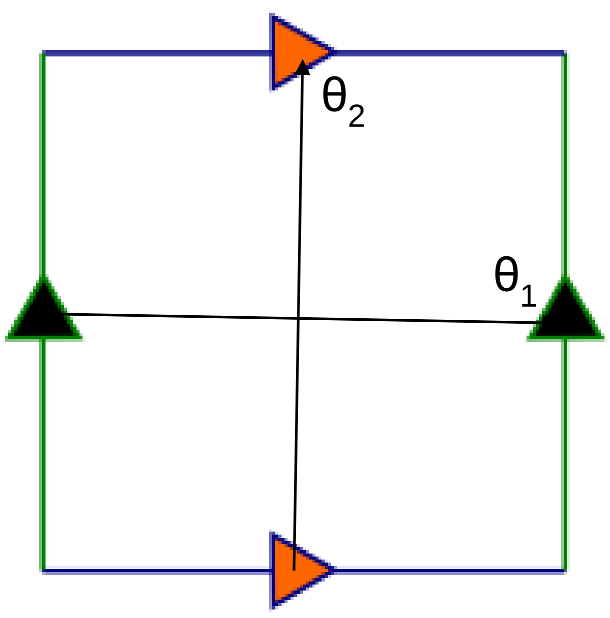

Choset, H etal 2007, Principles of robot motion: Theory, algorithms, and implementations, chapter 3

# Configuration Space Topology



# Configuration Space Topology



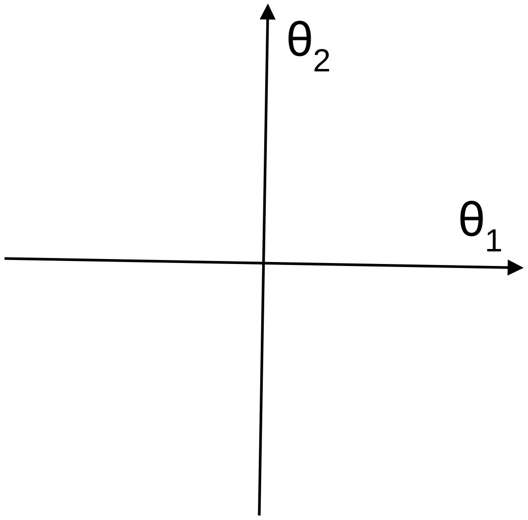

Circle (sphere-1) topology : $S^1$

Torus surface = $(\theta_1, \theta_2)$ Cartesian product of two circles : $S^1 \times S^1$

# Configuration Space Topology

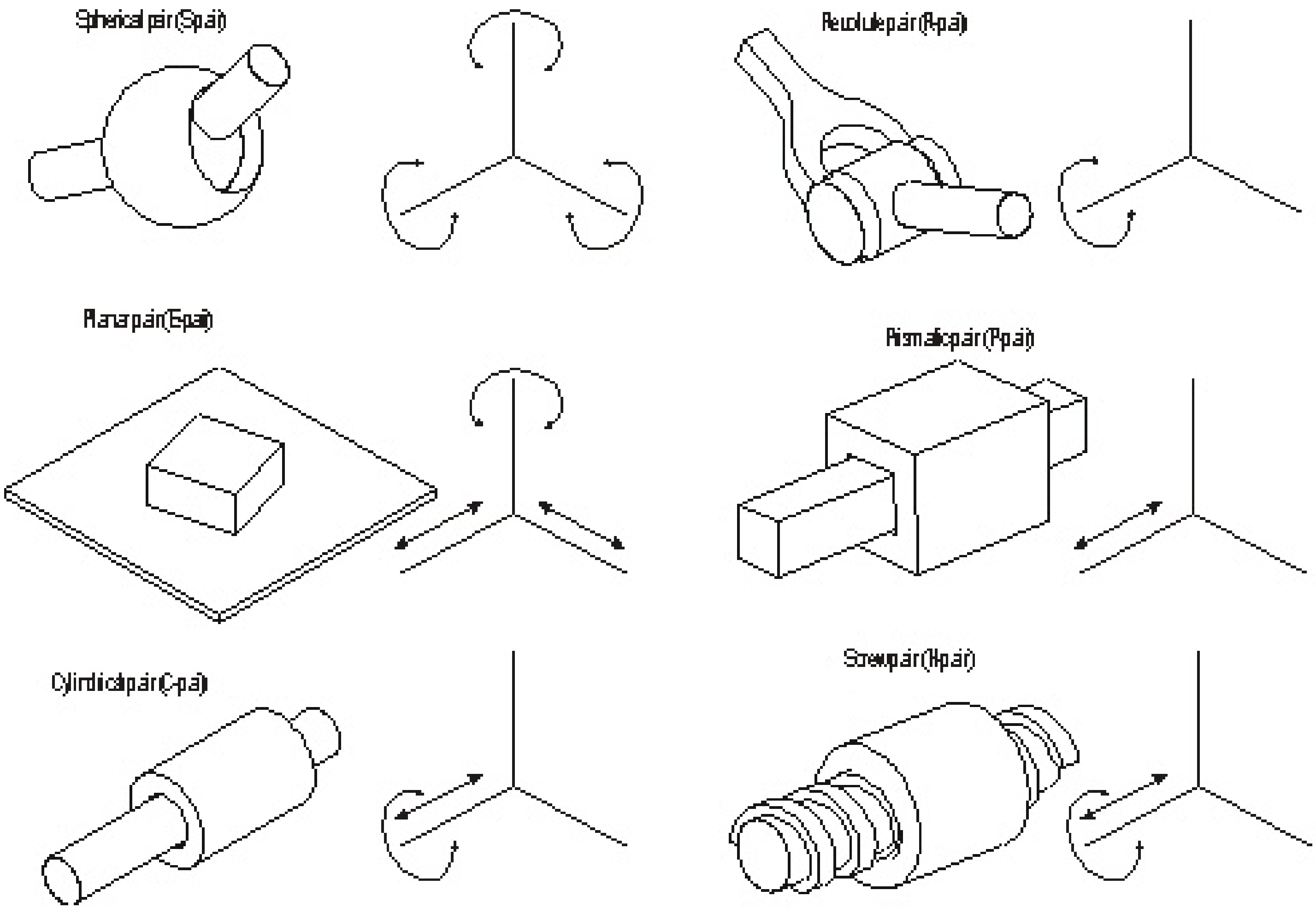
When the rotation is not a full circle?

Can approximate it as bounded region → Euclidean toplogy can also be used.



# Controlled Mobility

# Articulated Mechanisms



# Articulated Mechanisms

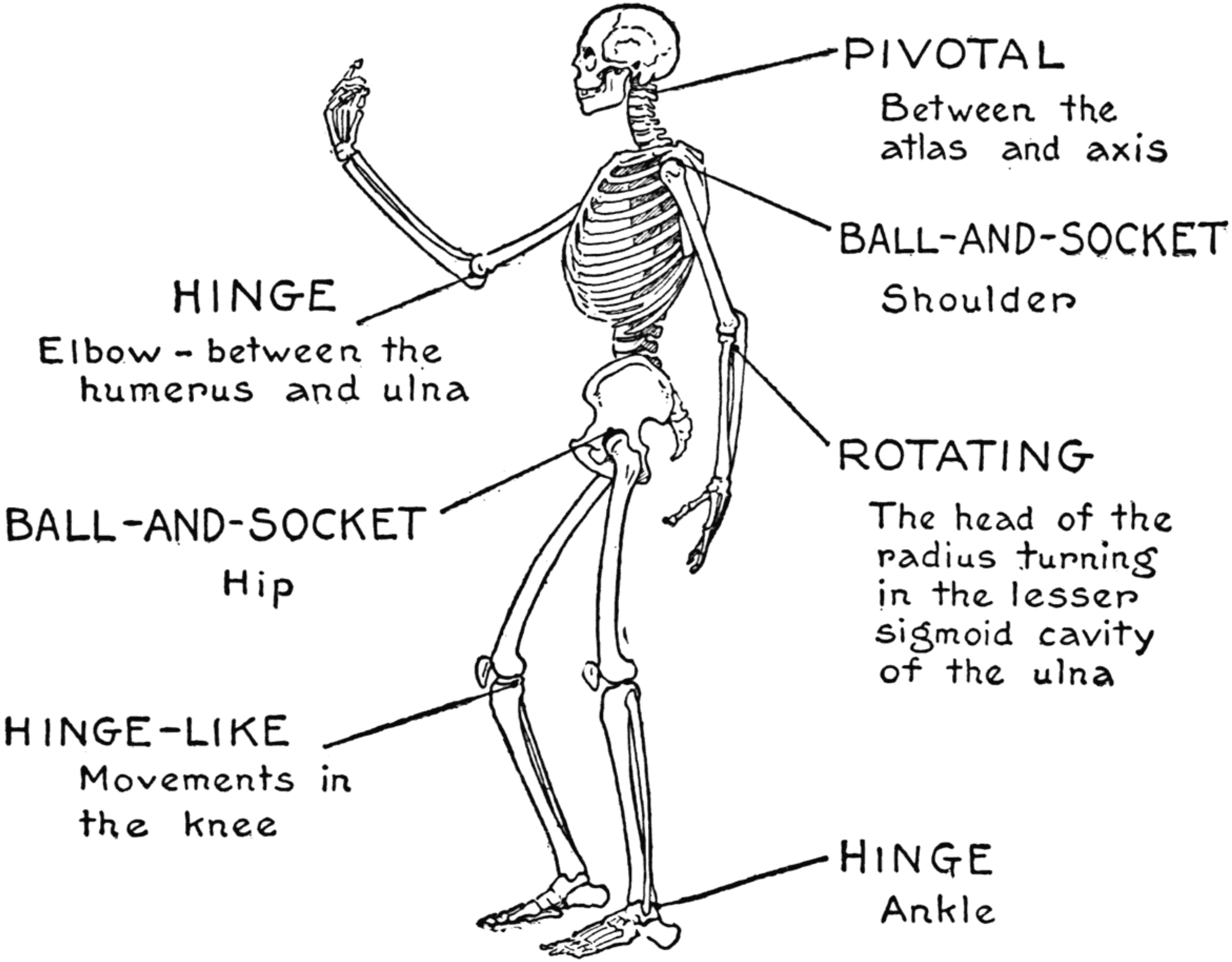
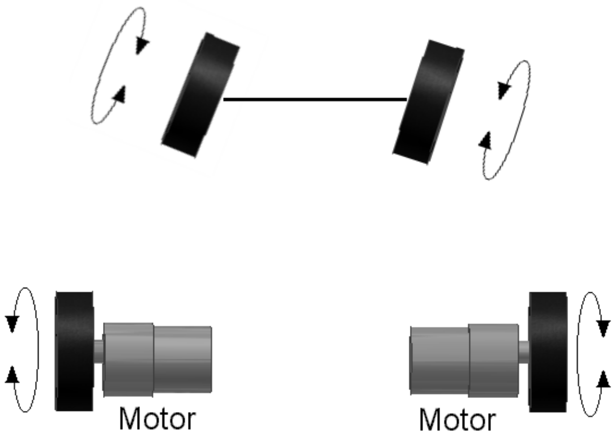
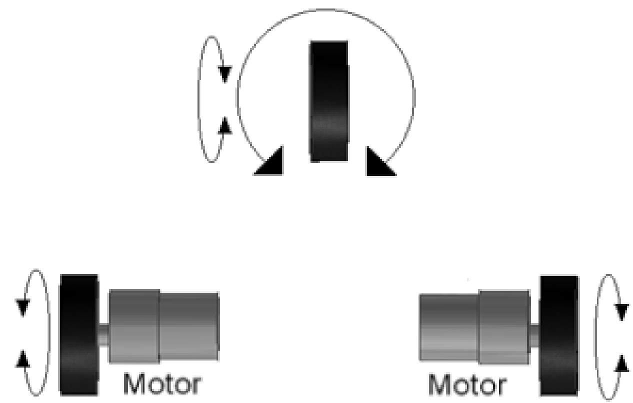



Image: [lutz 1918]

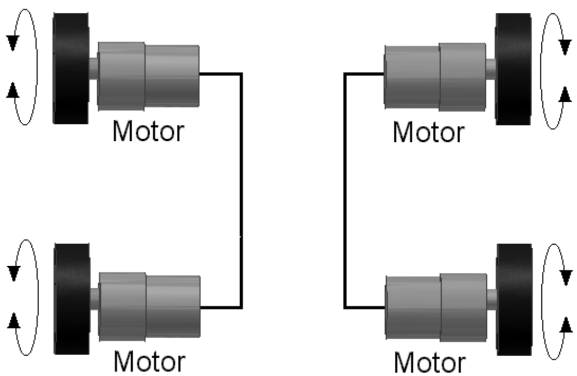
# Mobile Mechanisms




car-like steering

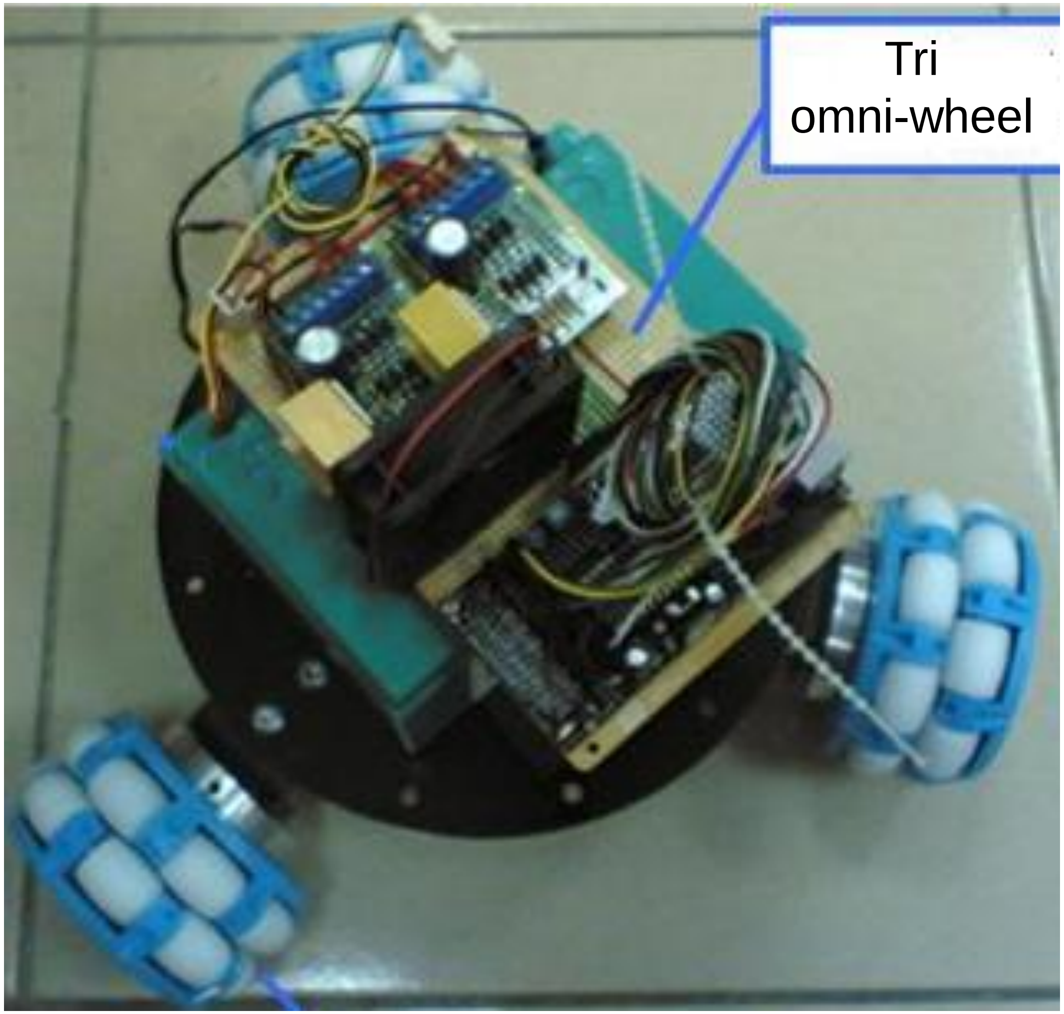



tricycle steering




omni-wheel steering

# Omni-wheel platforms



# Mobile Mechanisms

# Robot Motion Planning

Amitabha Mukerjee
IIT Kanpur, India

# Designing motion algorithms

Assume that environment and robot parameters are known

Objective:

- Model the robot's body (geometry + kinematics), as R(q) a function of its configuration q
- Model the obstacles B

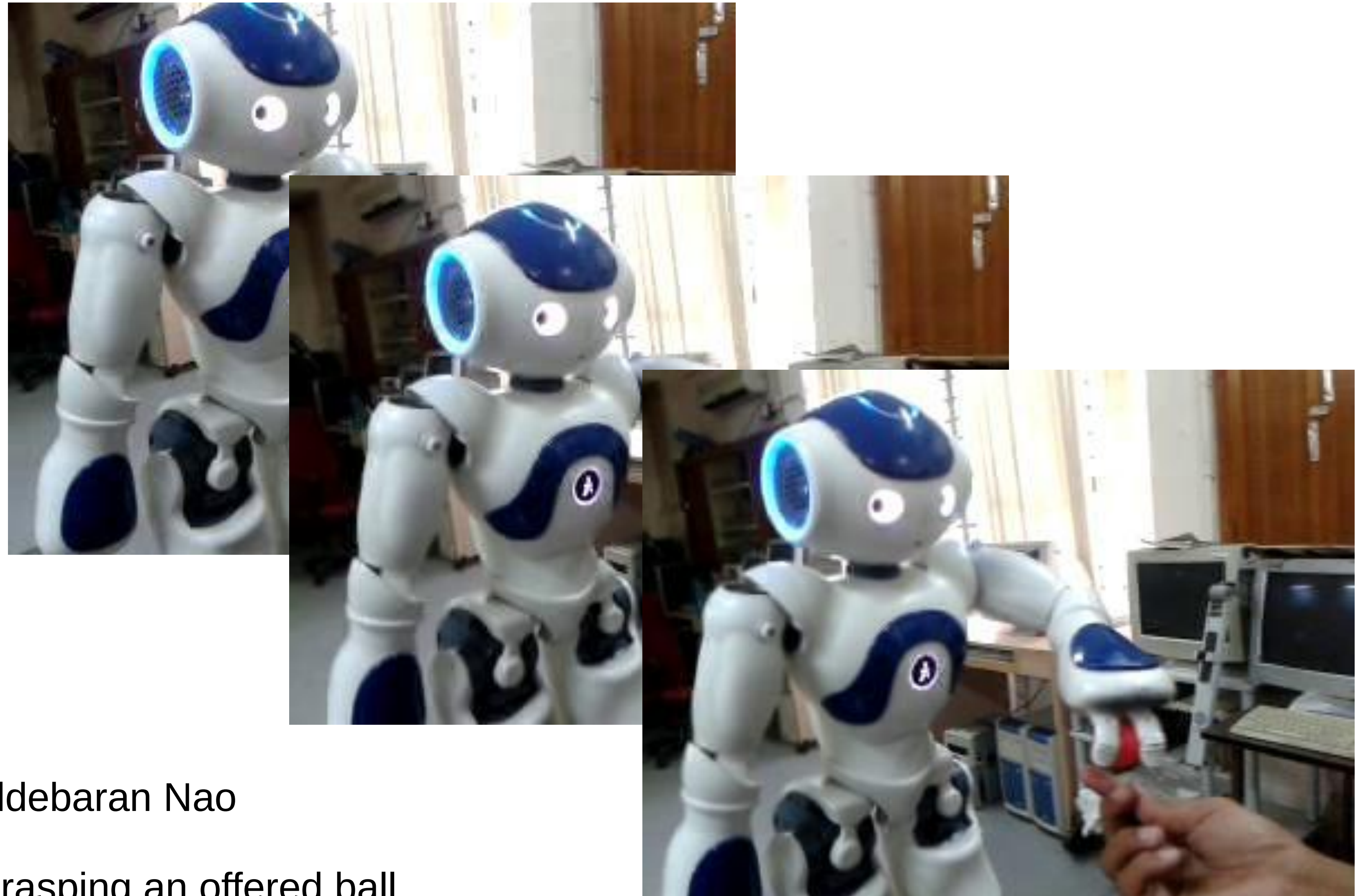
- find path P from $q_S$ to $q_G$ s.t. for all $q \in P$, $R(q) \cap B = \emptyset$

# Sensing and Motion Planning

[bohori venkatesh singh mukerjee 05]

Bohori/Venkatesh/Singh/Mukerjee:2005

# Programming a robot



Aldebaran Nao

Grasping an offered ball

# Programming a robot

1. detect ball using colour:

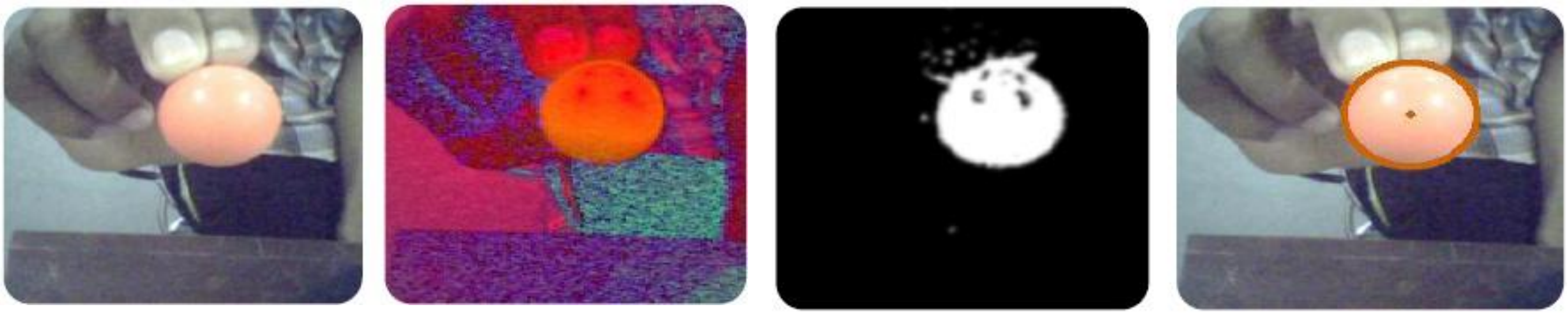


image captured by nao HSV binarized contour detected

2. estimate distance of ball (depth) from image size

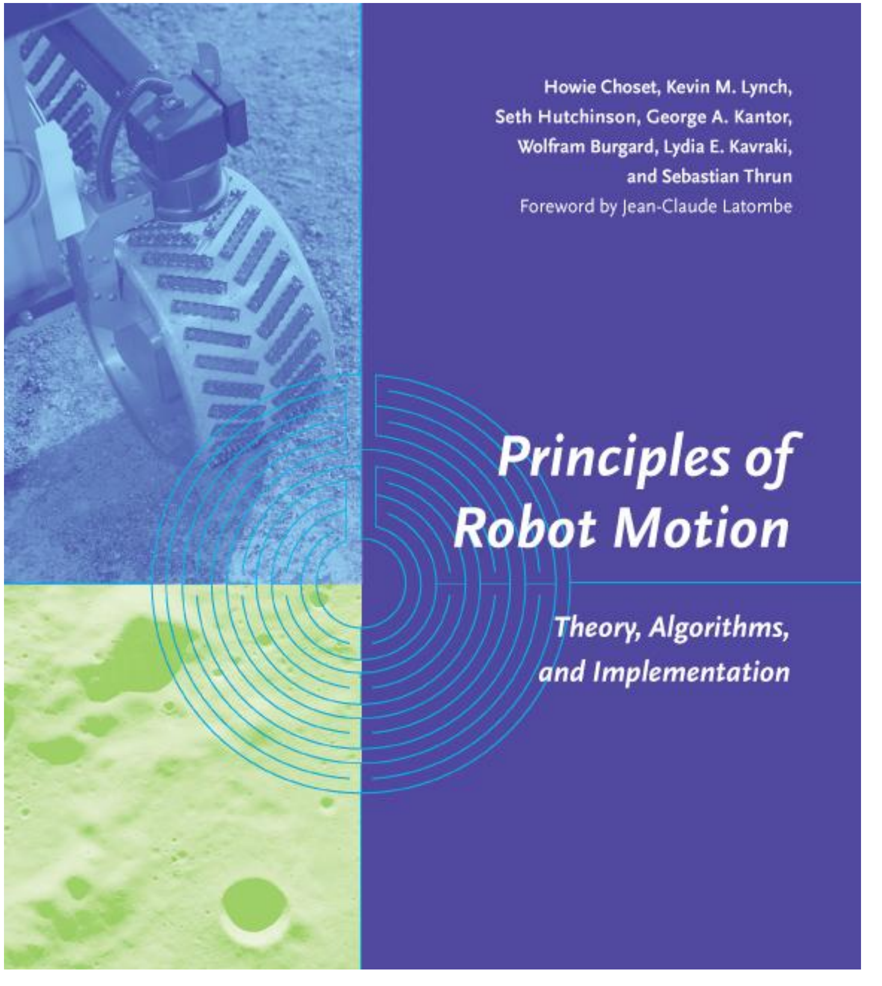
3. Inverse kinematics to grasp ball

Sensing in the workspace

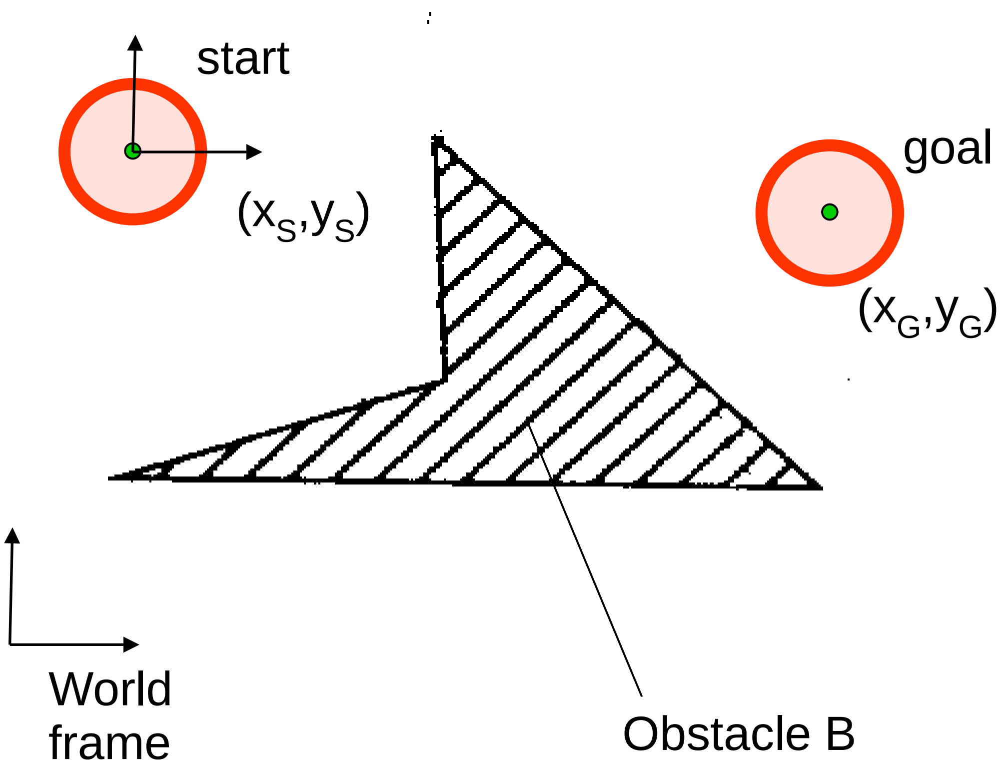
Motion planning in C-space

# Configuration Space

indian edition
rs 425



# Robot Motion Planning



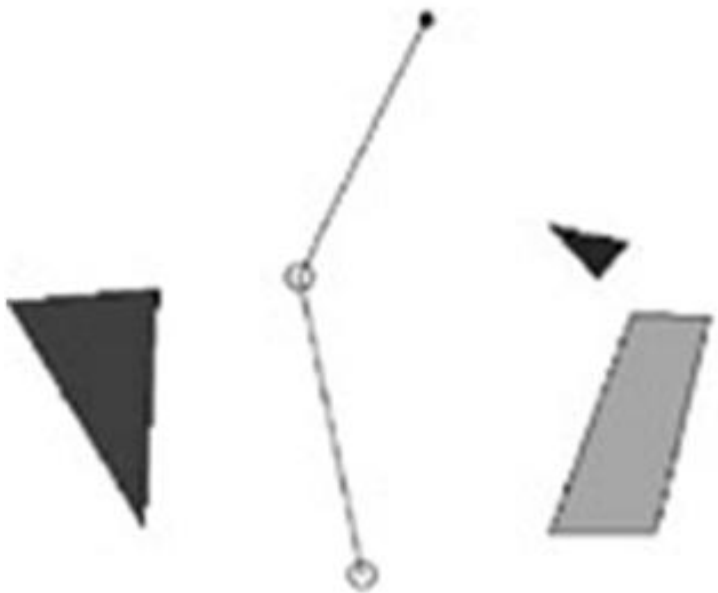

Valid paths will lie among those where the robot does not hit the obstacle

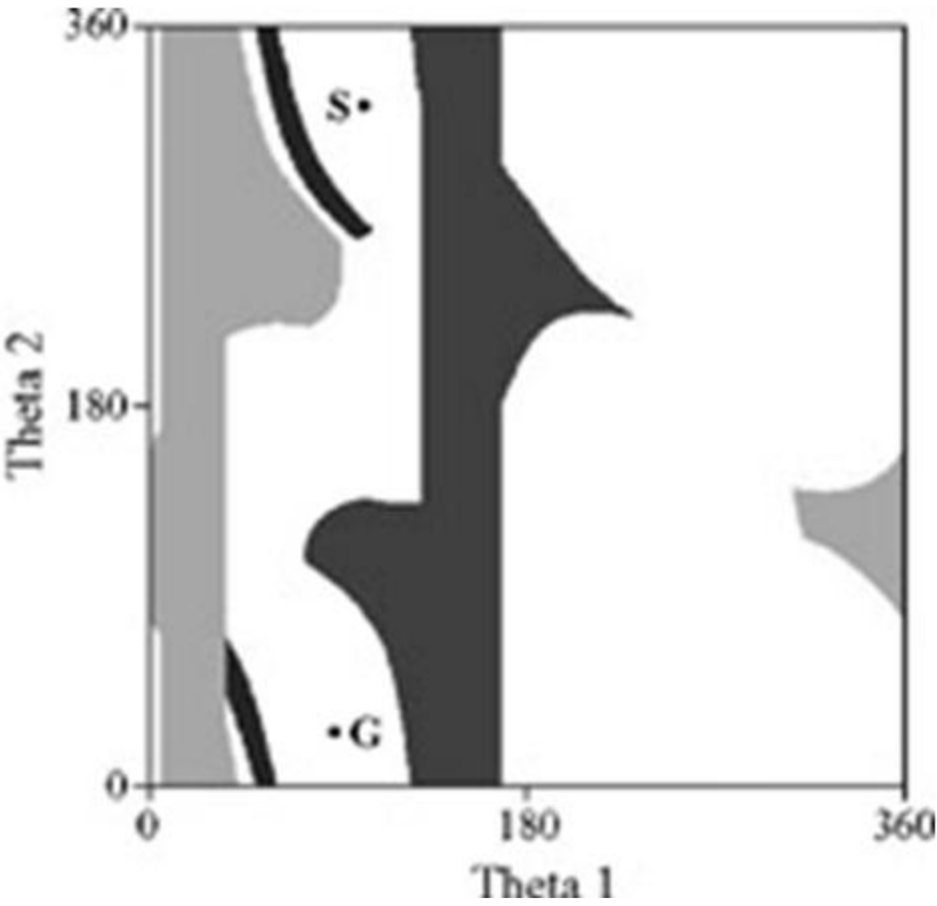
find path *P* from start to goal s.t.

for all $t$, $R(t) \cap B = \emptyset$

How to characterize the set of poses for which the robot does not hit the obstacle B?

# Robot Motion Planning





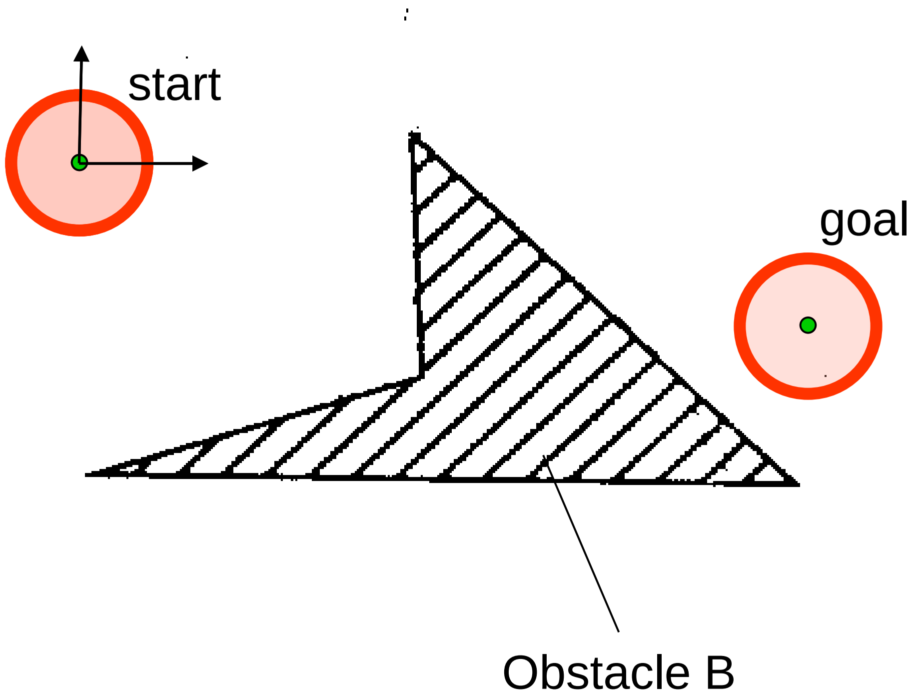
# Continuum approaches vs Discretization

**Two approaches to Robot motion planning**:

- **continuum**:
  treat motion space as single continuum
  → optimization

- **discretization**:
  decompose motion space into regions / segments
  → graph-search

# Potential fields



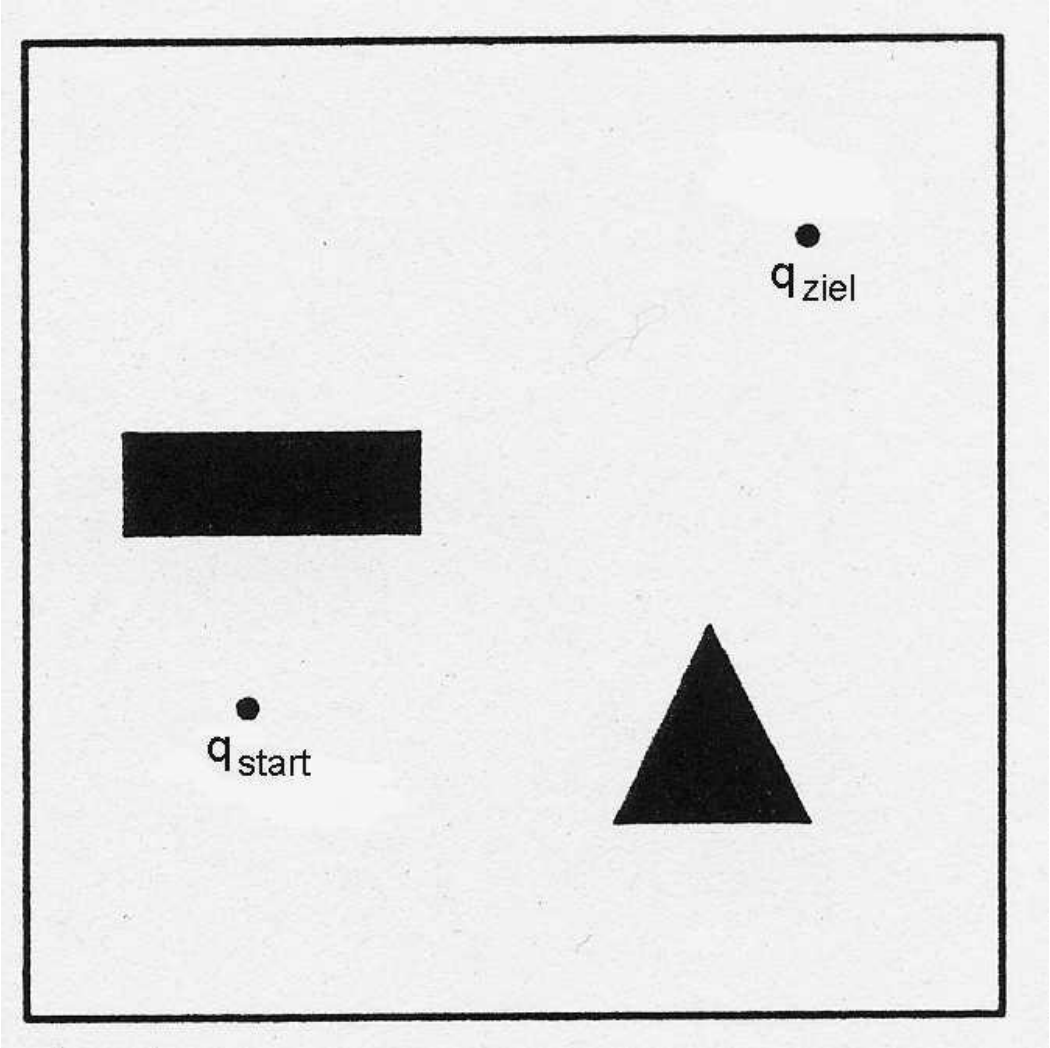

**Potential fields**

1. Goal: negative (attractive) potential
   Obstacles: positive (repulsive) potential
2. Robot moves along gradient
3. Problems:

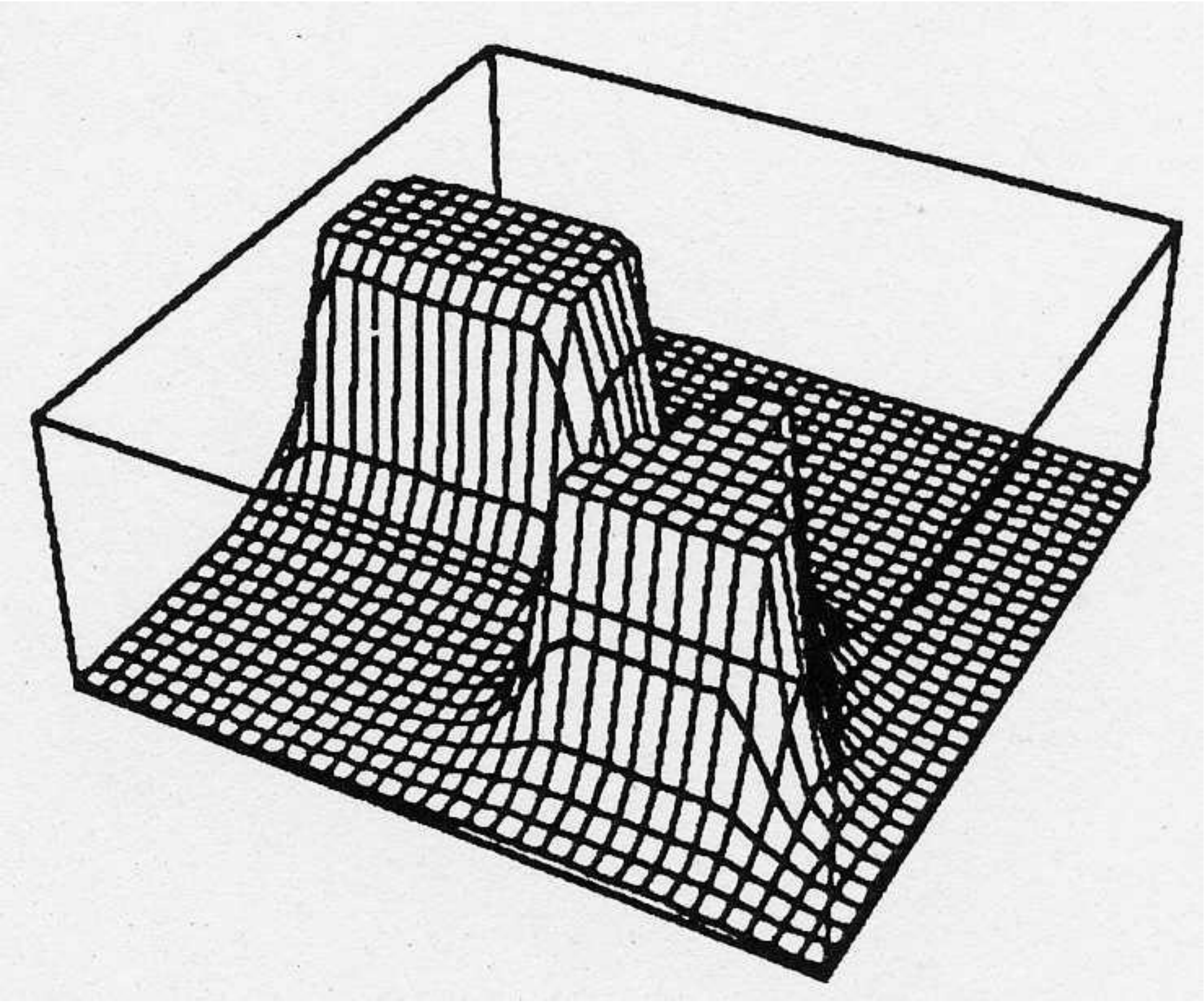
   - need to integrate the potential over the area of robot

   - problem of local minima

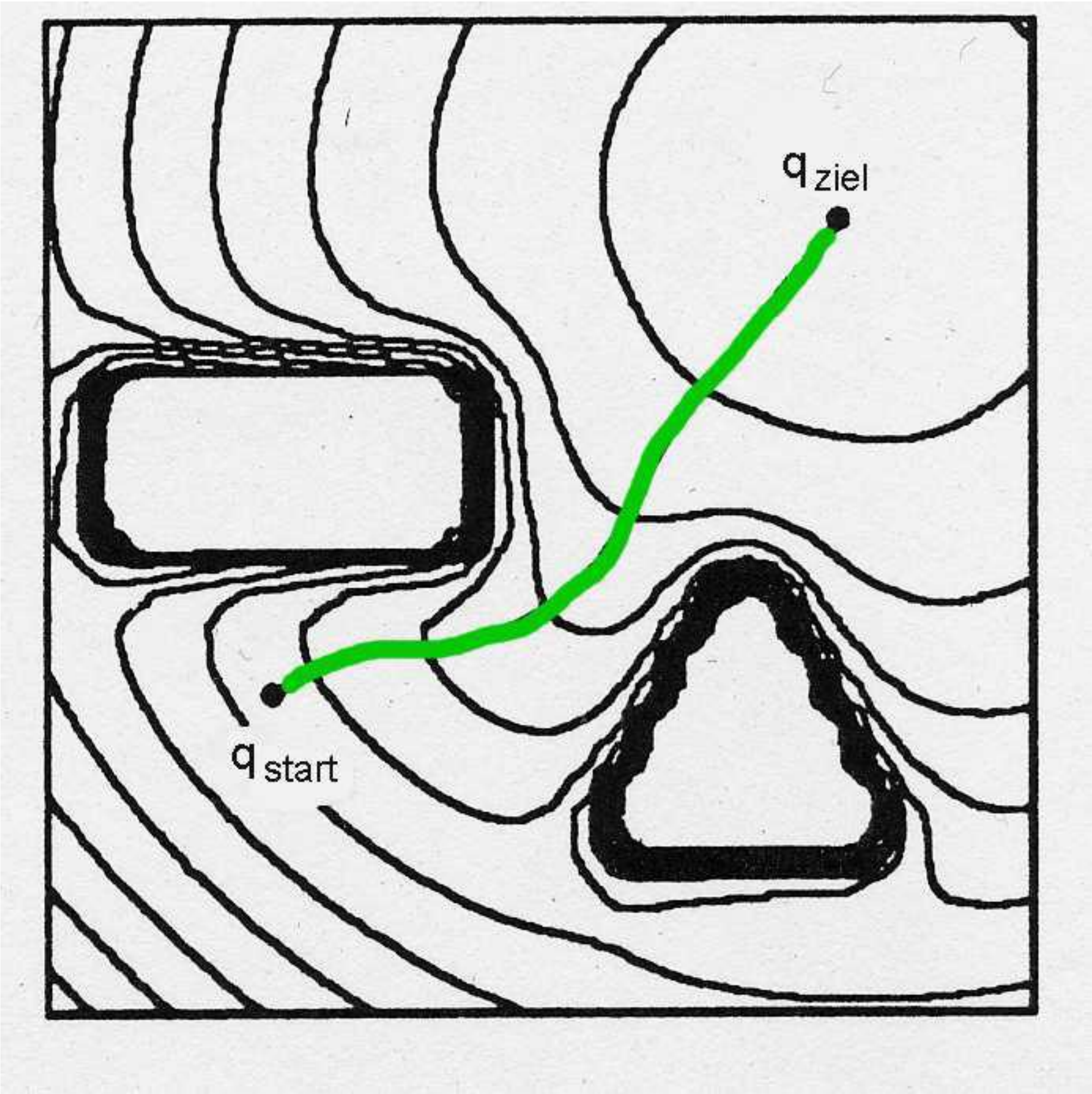
# Potential fields



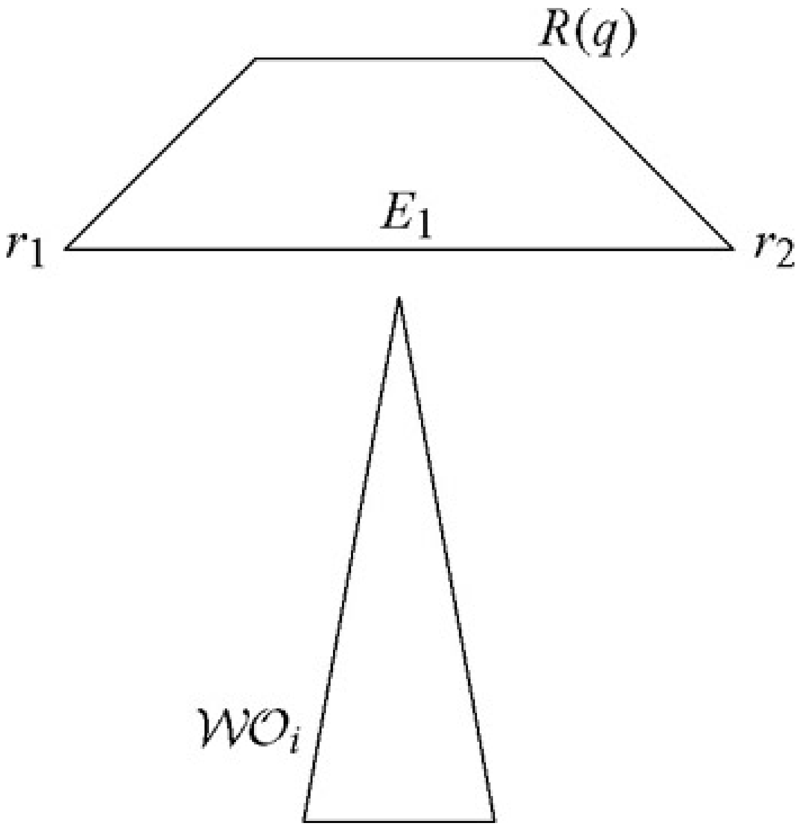
# Potential fields



# Potential fields



# Finite area robots




Instead of integrating over robot area, restrict to a set of *control* points

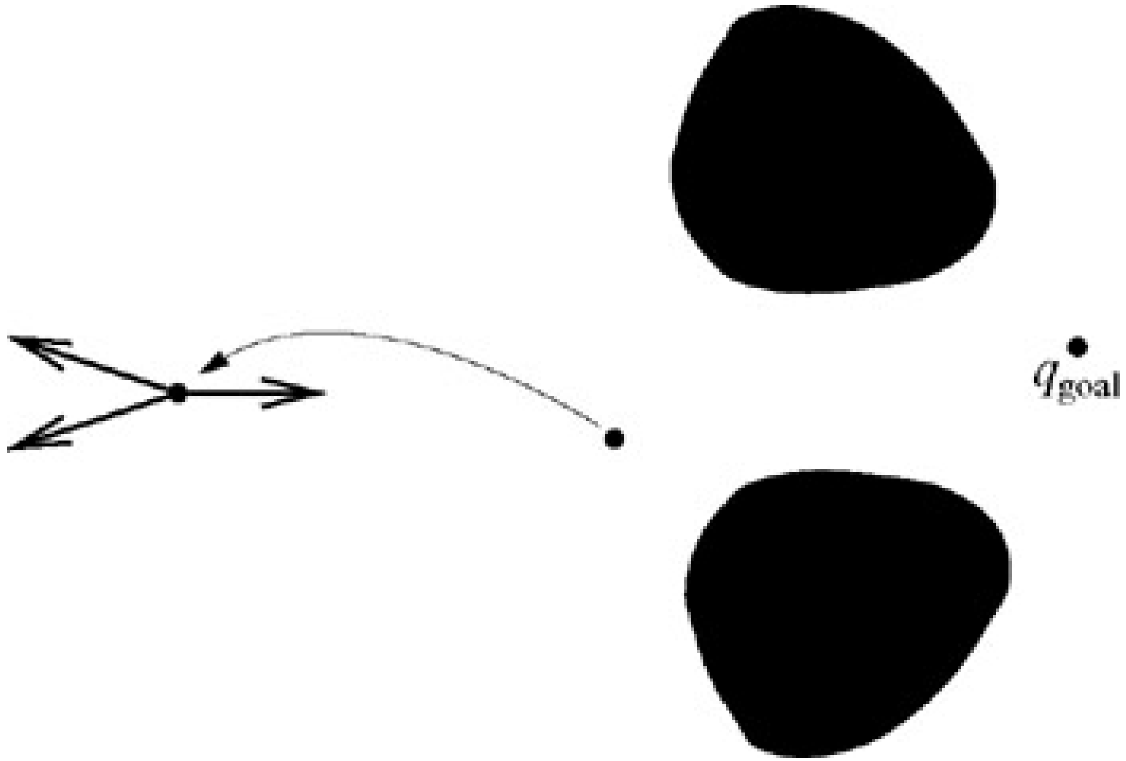
e.g. vertices

Problem:

With control points r1 and r2 on robot R(q), edge E1 may still hit Obstacle.

→ Attempt to reduce computation to points

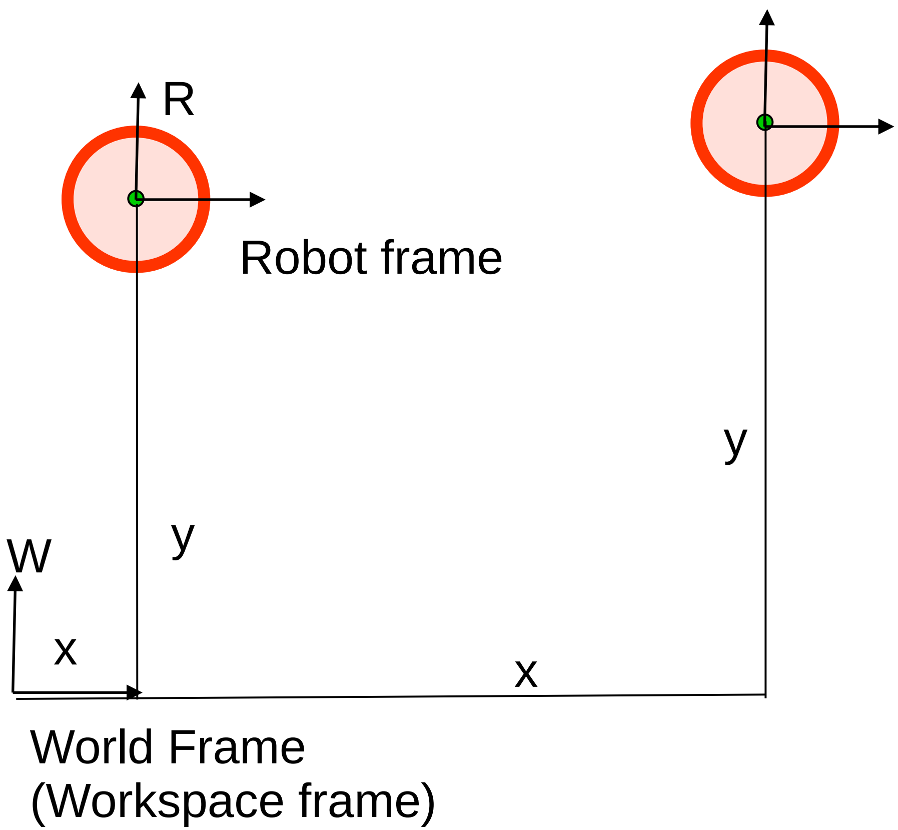
# Local Minima




persists even for point robots

# Configuration spaces

# Models of Robot Motion

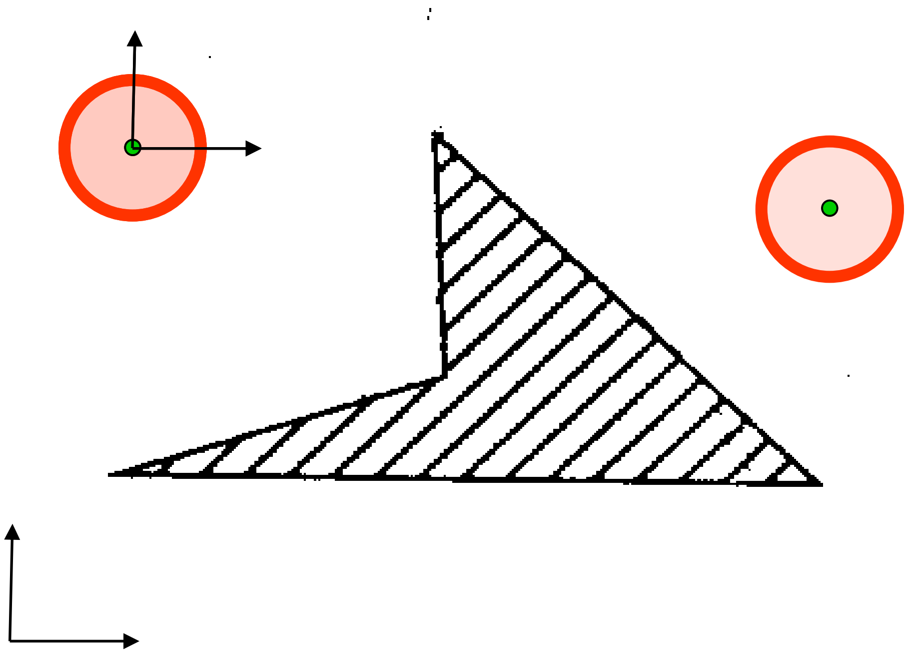



DEFINITION:
**degrees of freedom**:
number of parameters needed to fix the robot frame R in the world frame W

(x,y) = **configuration**
(vector **q**)

given configuration **q** for a certain pose of the robot, the set of points on the robot is a function of the configuration: say R(**q**)

# Robot Motion Planning

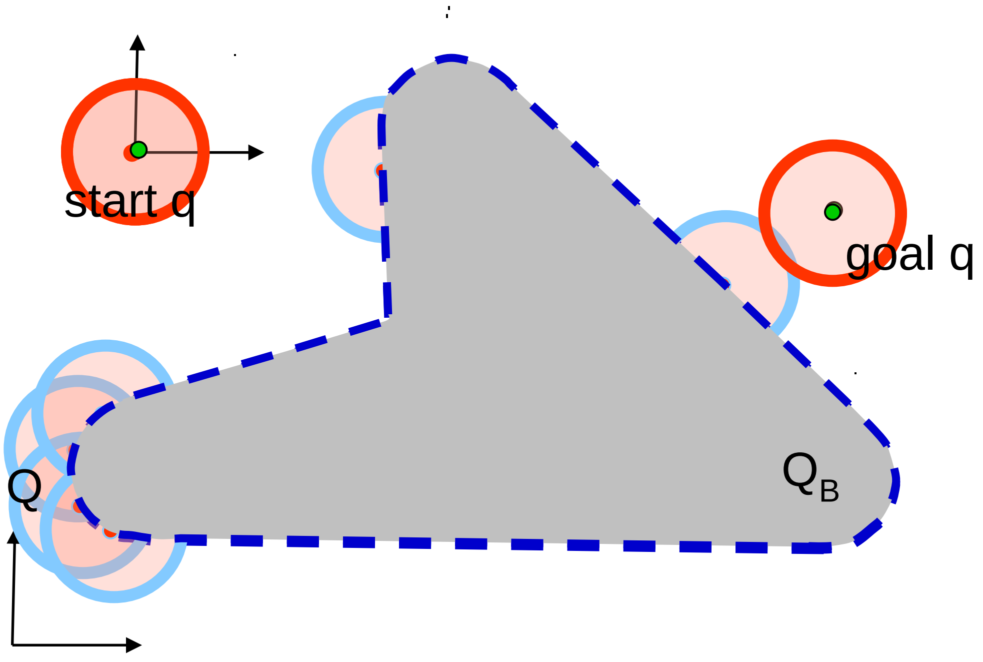


find path P from $\boldsymbol{q}_S$ to $\boldsymbol{q}_G$ s.t. for all $\boldsymbol{q} \in P, R(\boldsymbol{q}) \cap B = \emptyset$

? generate paths and check each point on every path?

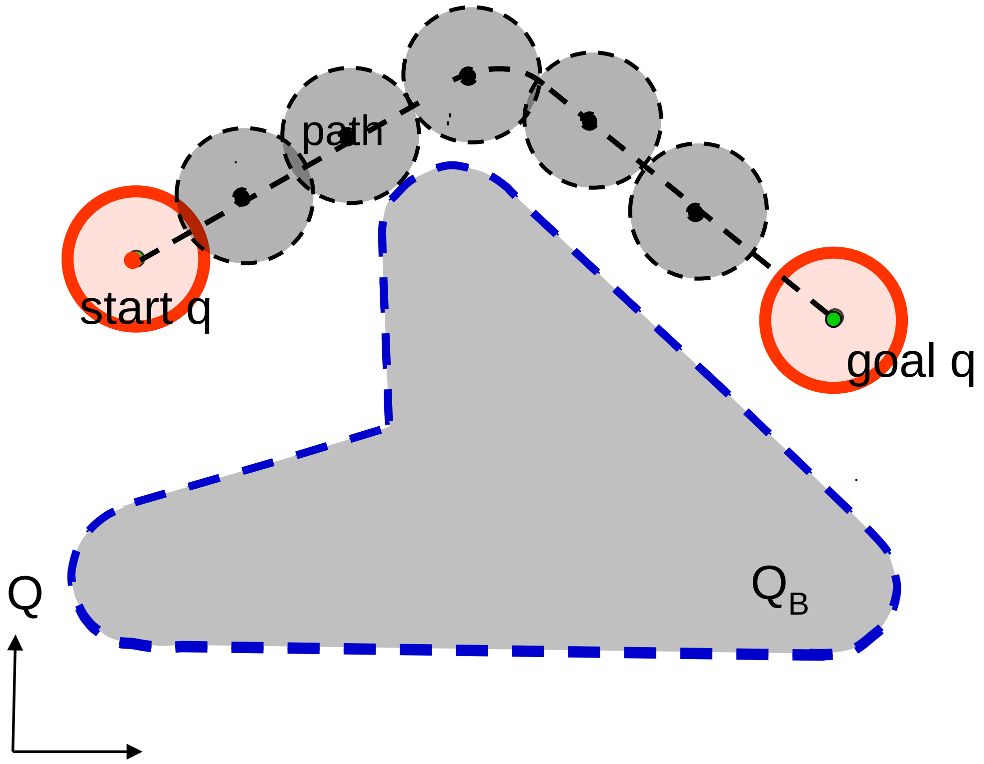
Would it be easier to identify $Q_{free}$ first?

# Robot Motion Planning




$Q_B = [\ \mathbf{q} \mid R(\mathbf{q}) \cap B \neq \emptyset\ \}$

# Motion Planning in C-space

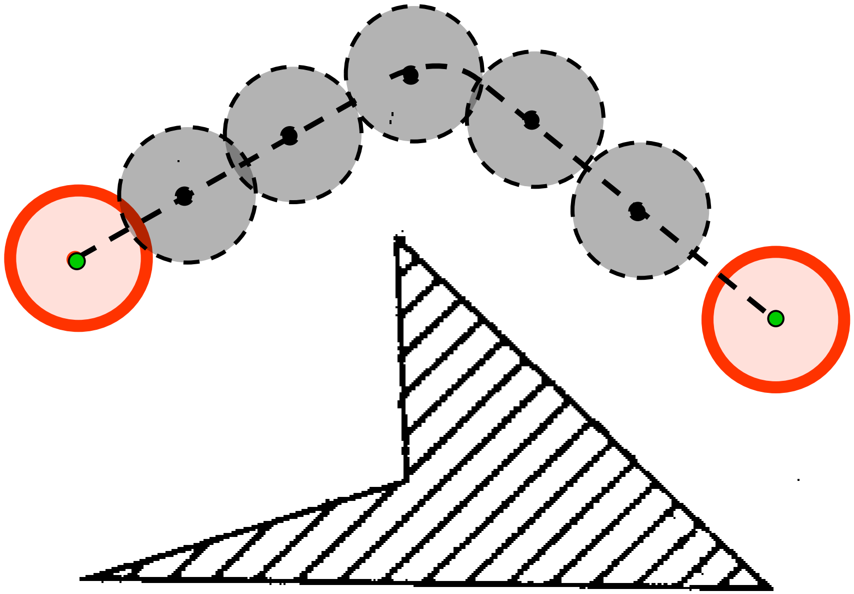



configurations are points in C-space

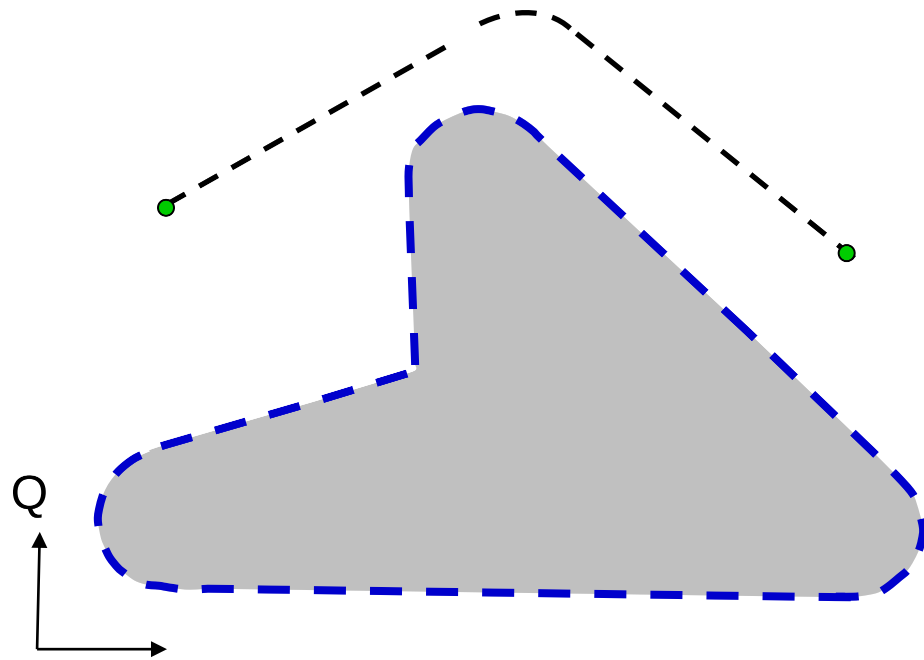
path P is a line

if $P \cap Q_B = \emptyset$, then path is in $Q_{free}$

# Motion Planning in C-space

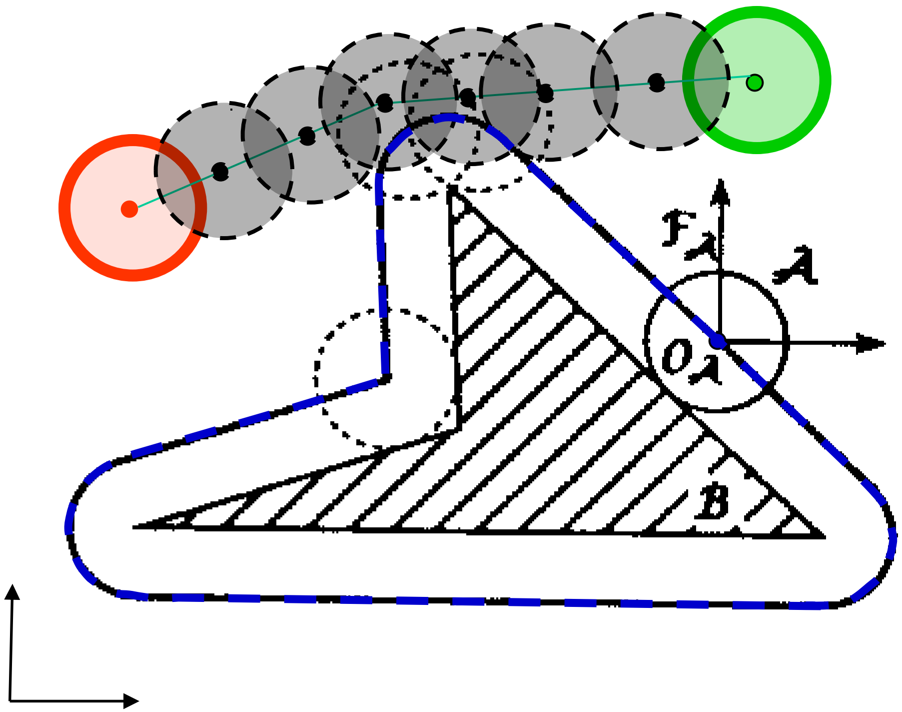


workspace

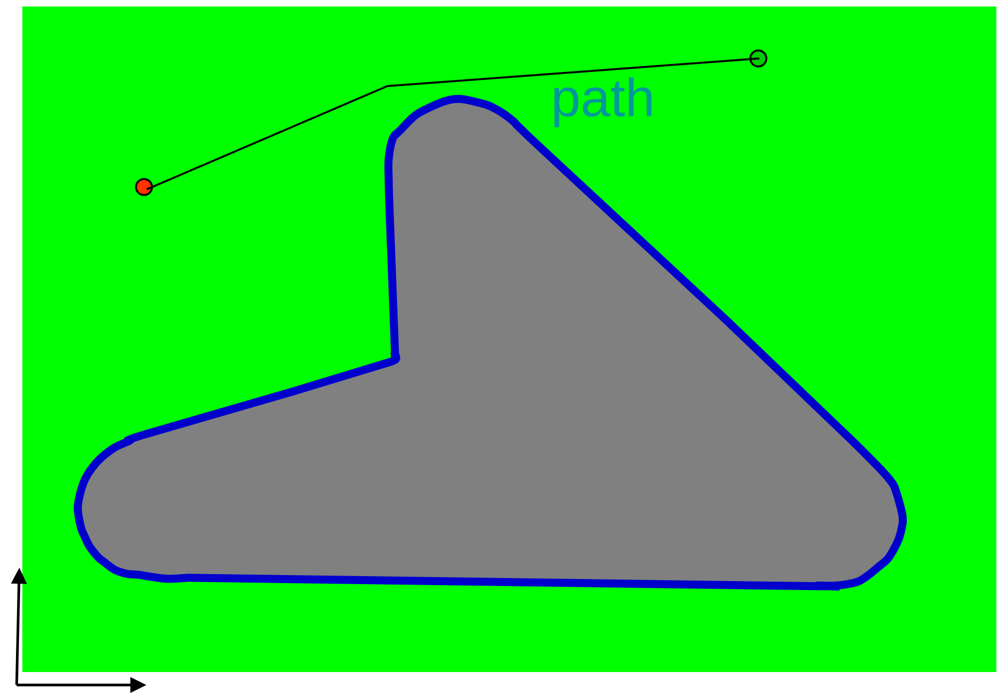



Configuration space

# Robot Motion Planning

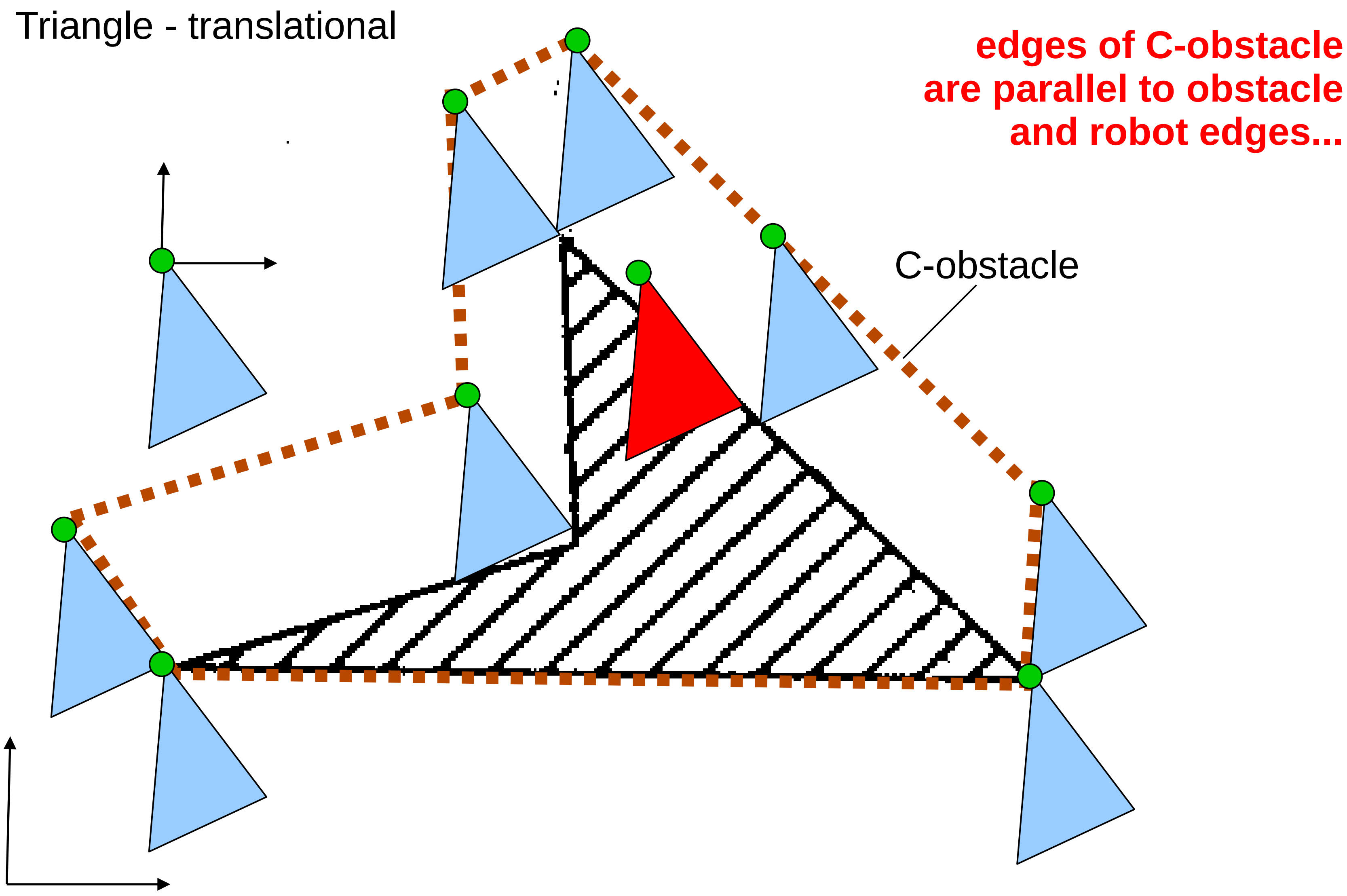



workspace
W

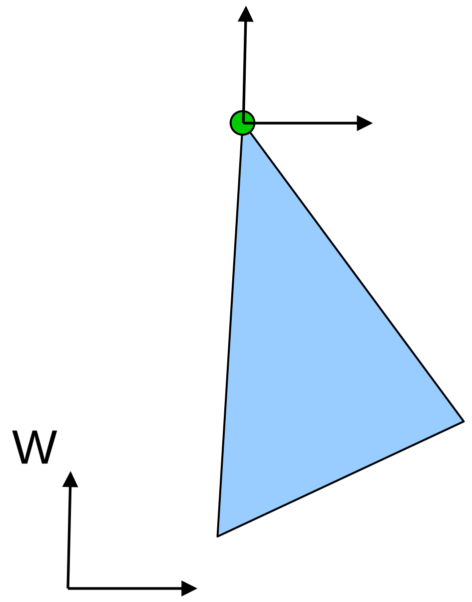



configuration space
C

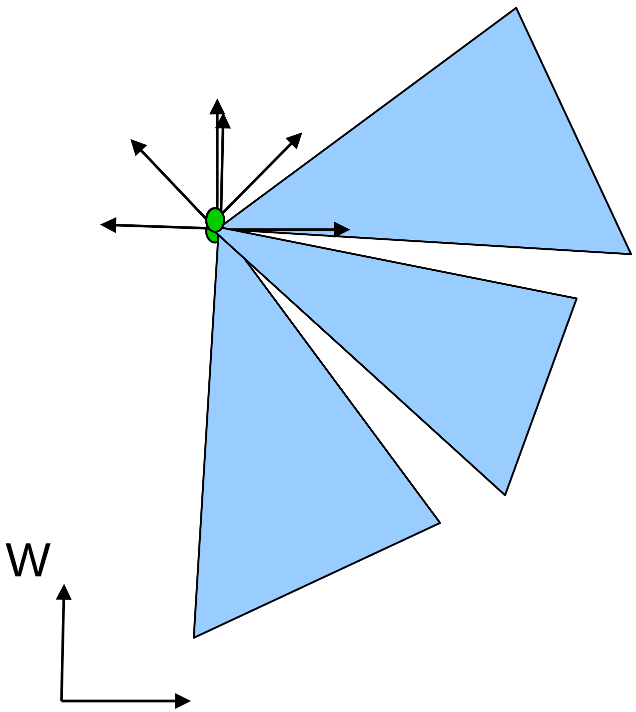
# Non-circular mobile robots



# Mobile robots with Rotation

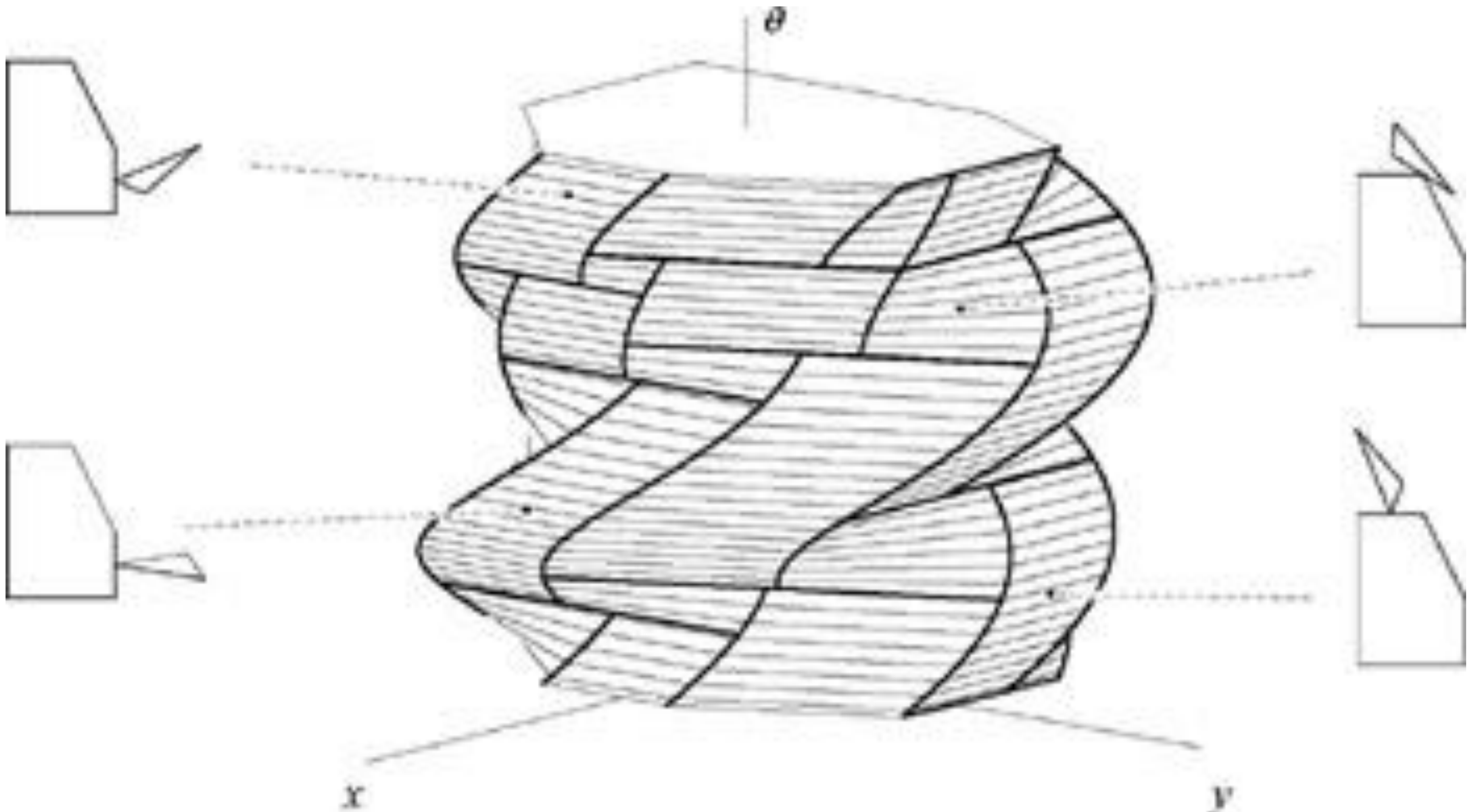


# Mobile robots with Rotation



# Mobile robots with Rotation

C-space with rotation $\theta$ (polygonal obstacle)



# Configuration Space Analysis

Basic steps (for ANY constrained motion system):

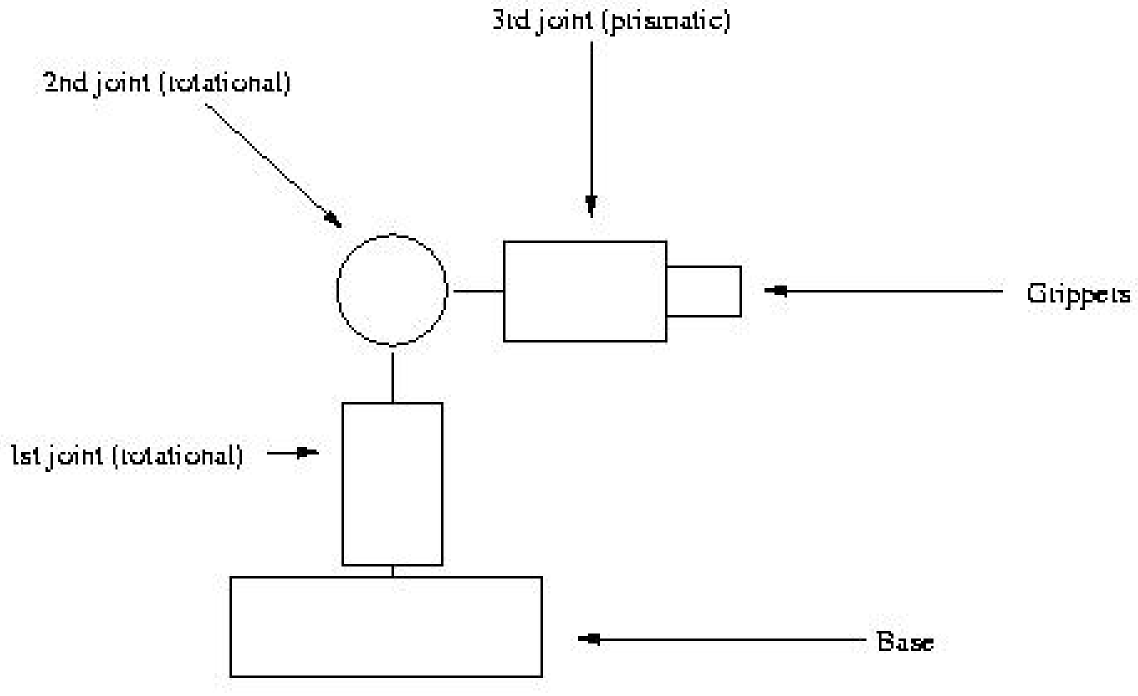
1. determine degrees of freedom (DOF)
2. assign a set of configuration parameters **q**

   e.g. for mobile robots, fix a frame on the robot
3. identify the mapping $R : Q \rightarrow W$, i.e. R(**q**) is the set of points occupied by the robot in configuration **q**
4. For any **q** and given obstacle B, can determine if $R(\mathbf{q}) \cap B = \emptyset$. → can identify $Q_{free}$

   Main benefit: The search can be done for a point
5. However, computation of C-spaces is not needed in practice; primarily a conceptual tool.

# Configuration spaces for Articulated Robots

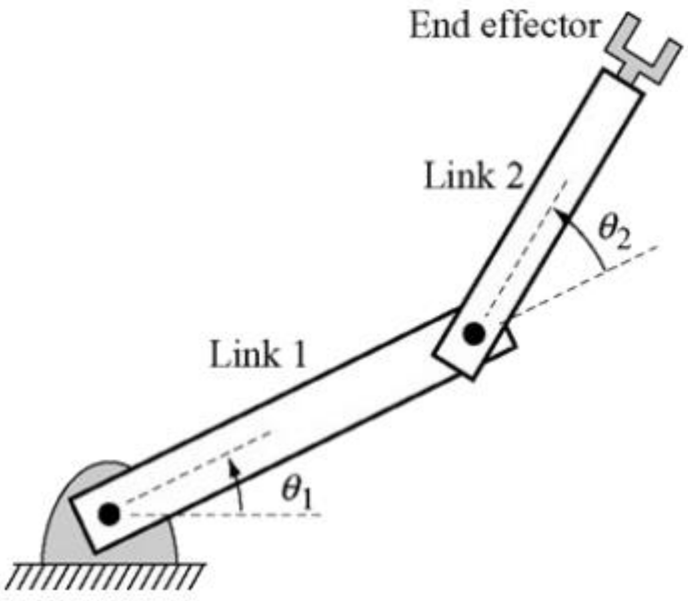
# Articulated Robot




Main idea:

C-Space computation is **same** for ALL kinds of robots

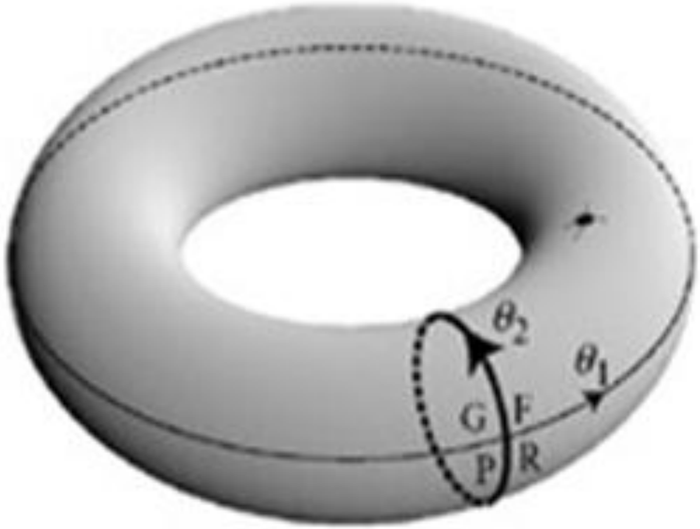
# Articulated Robot C-space




How many parameters needed to fix the robot pose ?

What may be one assignment for the configuration parameters?

# C-space as manifolds




Topology of C-space: Torus  (S1 x S1)

Choset, H etal 2007, Principles of robot motion: Theory, algorithms, and implementations, chapter 3

# C-space as manifolds

- **manifold**: generalization of curves / surfaces

  every point on manifold has a neighbourhood homeomorphic to an open set in $R^n$

- Mapping $\Phi$ : S←→ T is bijective (covers all of T and has unique inverse)

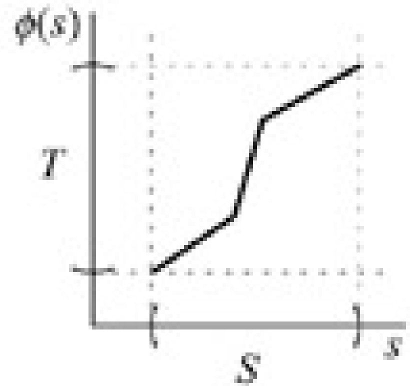
  $\Phi$ is

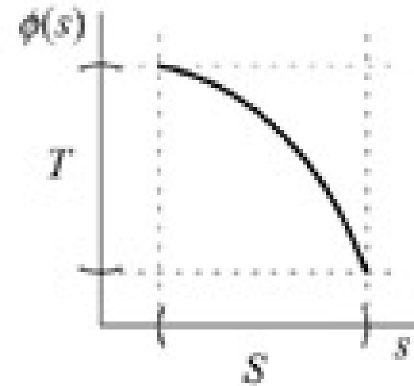
  *homeomorphic*:

  (f / $f^{-1}$ are continuous)

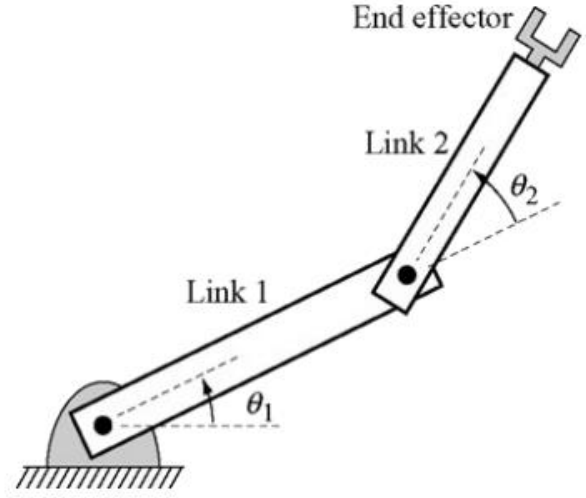
  *diffeomorphic* :

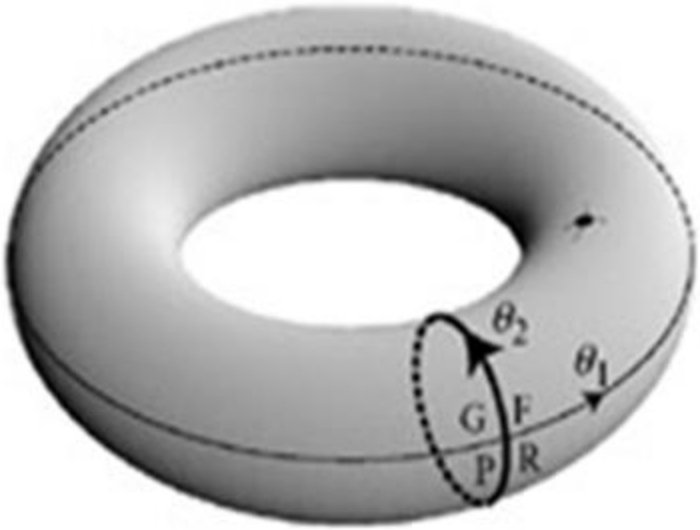
  (f / $f^{-1}$ are $C^\infty$ smooth)



# C-space as manifolds




Neighbourhood of q is mappable to R2

global topology is not R2 but S1 x S1 (torus)

# Map from C-space to W

Given configuration **q**, determine volume occupied by R(**q**) in workspace

For multi-link manipulators, spatial pose of link (n+1) depends on joint configuration **q** for joints 1, 2, ..., n.

→ Forward Kinematics

Map from W to C-space: given pose in workspace, find **q**

→ Inverse Kinematics

# Configuration Space Analysis

Basic steps (for ANY constrained motion system):

1. determine degrees of freedom (DOF)
2. assign a set of configuration parameters **q**

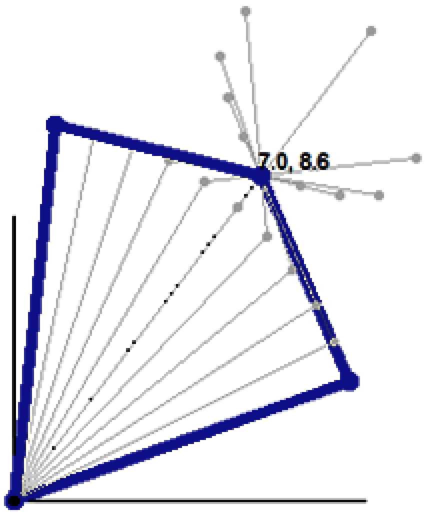
   e.g. for mobile robots, fix a frame on the robot

3. identify the mapping $R : Q \rightarrow W$, i.e. $R(\mathbf{q})$ is the set of points occupied by the robot in configuration **q**
4. For any **q** and given obstacle B, can determine if $R(\mathbf{q}) \cap B = \emptyset$. → can identify $Q_{free}$

   Main benefit: The search can be done for a point

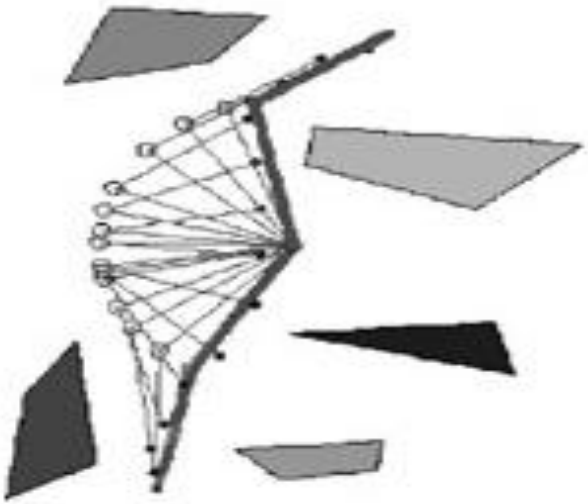
5. However, computation of C-spaces is not needed in practice; primarily a conceptual tool.

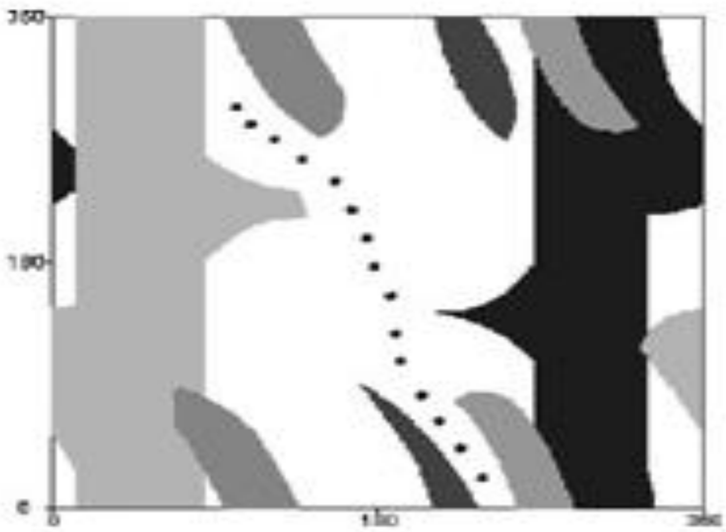
# Mapping obstacles




Point obstacle in workspace

# Articulated Robot C-space



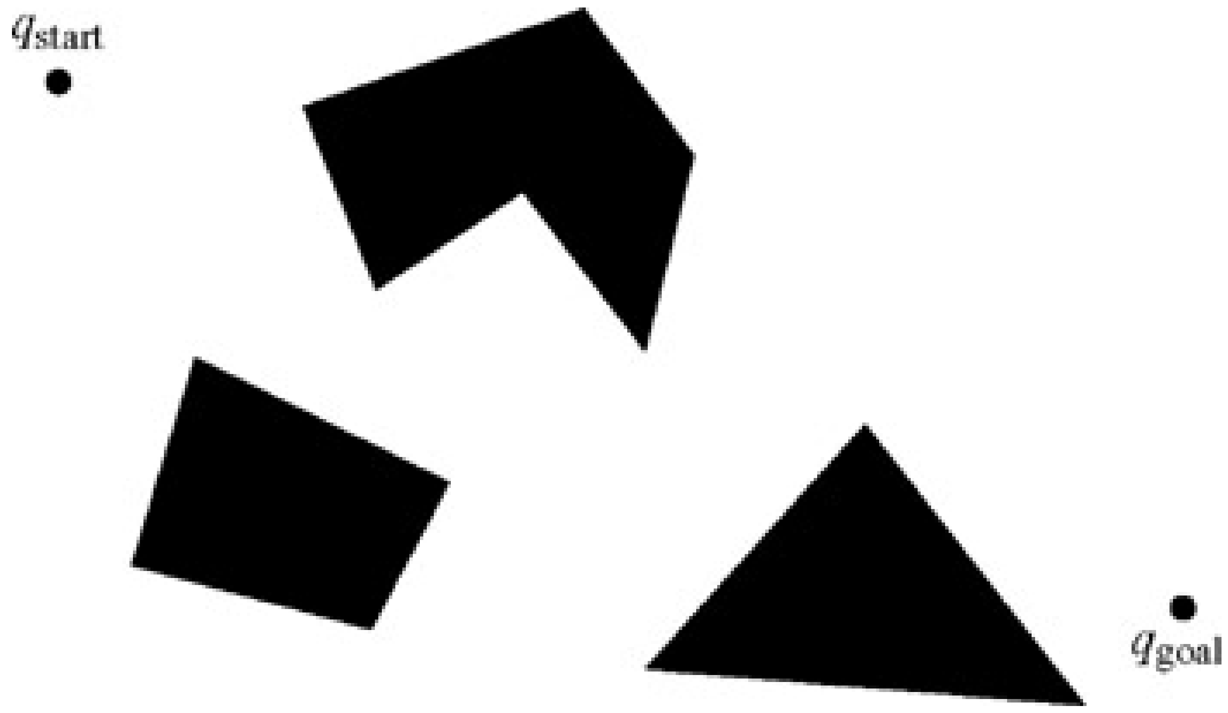


Path in workspace

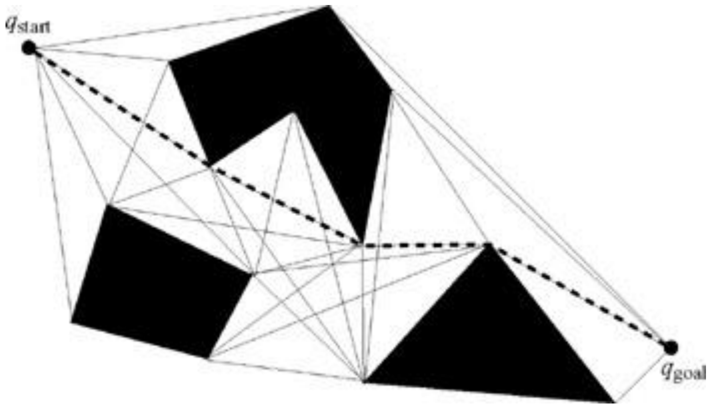
Path in Configuration Space

# Graph-based Motion Planning

# Visibility Graph methods



# Visibility Graph methods




Construct edges between visible vertices

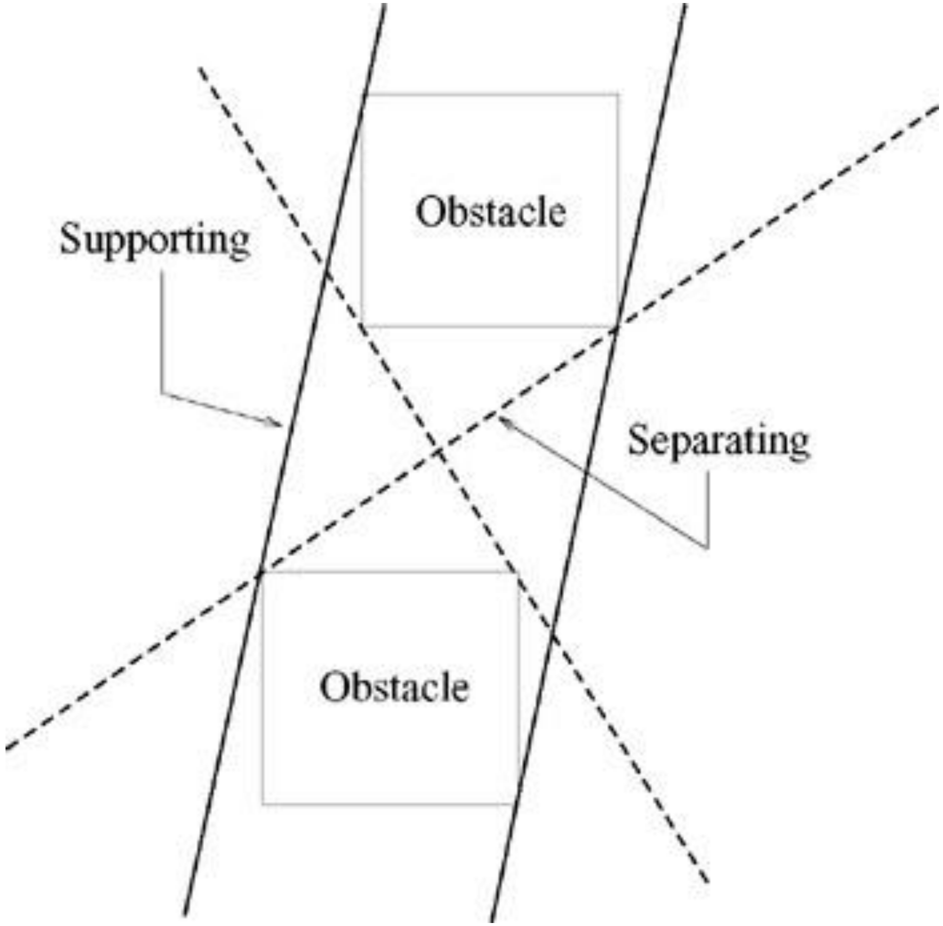
Sufficient to use only supporting and separating tangents

Complexity:

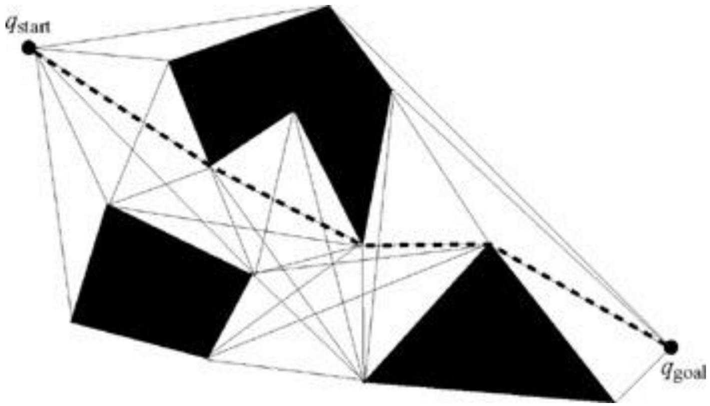
Direct visibility test: $O(n^3)$
(tests for each vtx: O(n) emanations x O(n) obst edges)

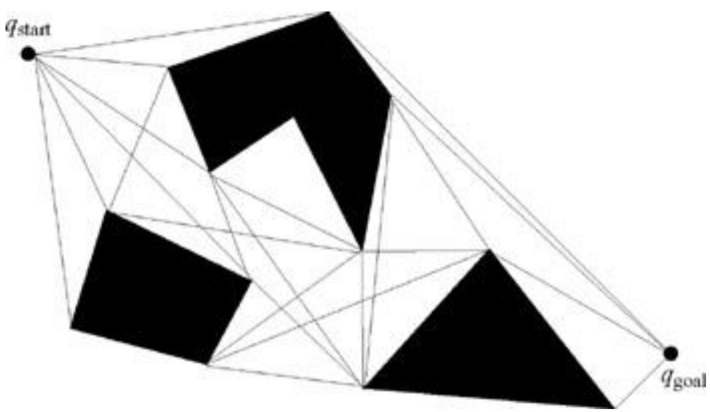
Plane sweep algorithm: $O(n^2 \log n)$

# Visibility Graph methods



# Reduced Visibility Graph




Sufficient to use only supporting and separating tangents

Finds “shortest” path – but too close to obstacles

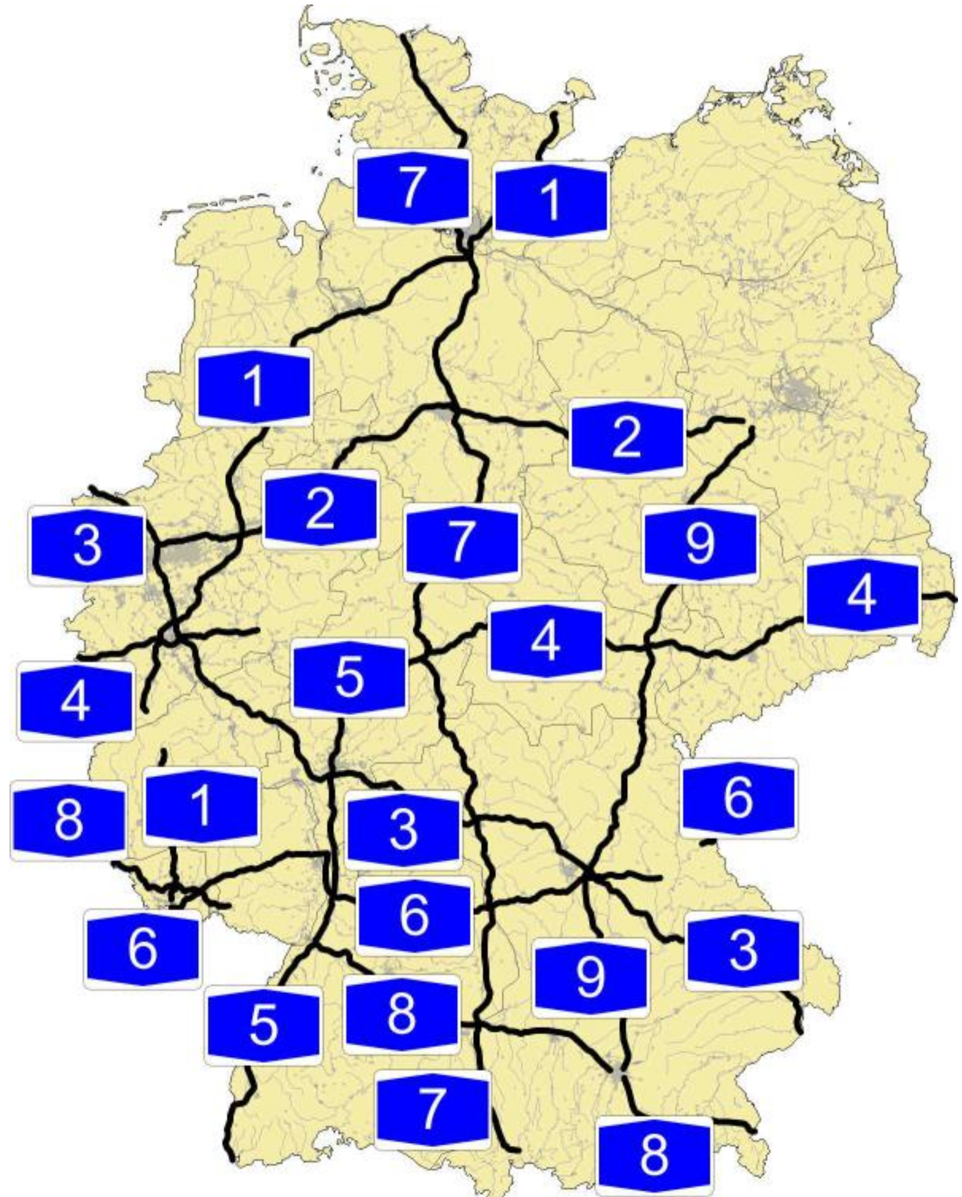
# Roadmap methods

# Roadmaps

To go from A to B, we use a set of known "via points" or landmarks on a map

e.g. To go from Delhi to Varanasi, you can go via Agra, Kanpur, Allahabad.

Roadmap = graph (V,E). Set of edges E connect nodes V.



# Roadmaps

any roadmap RM must have three properties:

*Connectivity*:
path exists between any $q'_{START}$ and $q'_{GOAL}$ in RM

*Accessibility*:
exists a path from any $q_{START} \in Q_{free}$ to some $q'_{START} \in RM$

*Departability*:
exists a path from some $q'_{GOAL} \in RM$ to any $q_{GOAL} \in Q_{free}$

# Staying away from Obstacles: Generalized Voronoi Graphs

Voronoi Region of obstacle i :

$$\mathcal{F}_i = \{q \in Q_{\text{free}} \mid d_i(q) \leq d_h(q) \quad \forall h \neq i\},$$

Voronoi diagram:

set of *q* equidistant from at least two obstacles

# GVG Roadmaps

Accessibility / Deparability:

Gradient descent on distance from dominant obstacle :

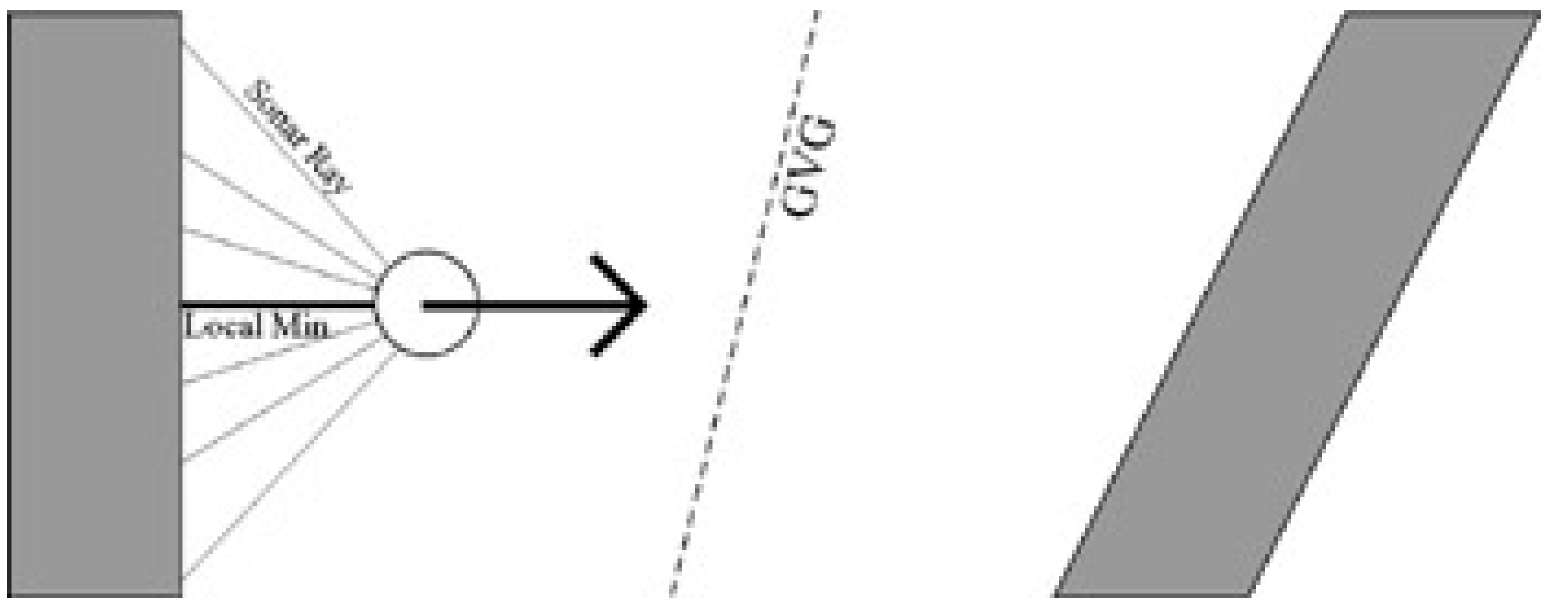
→ guaranteed to reach from any $q_{START} \in Q_{free}$ to some $q'_{START} \in RM$

→ motion is along a "retract" or brushfire trajectory

Connectivity:

GVG is Connected if path exists

# Sensor based Voronoi roadmap construction



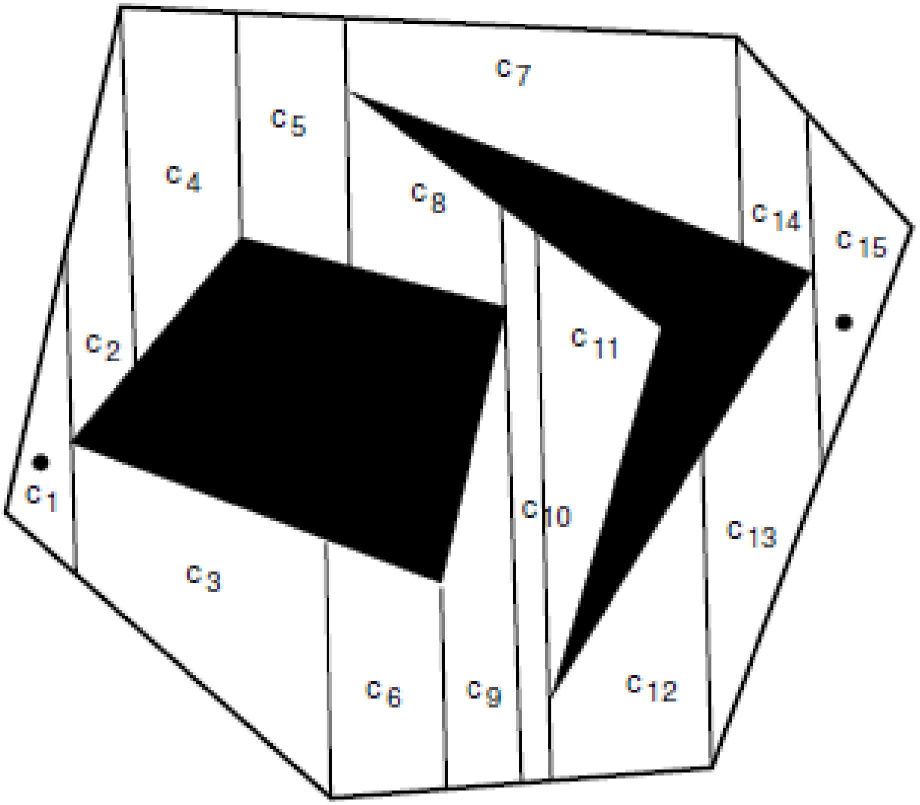
# Cell decomposition methods

Trapezoidal decomposition:
Each cell is convex.

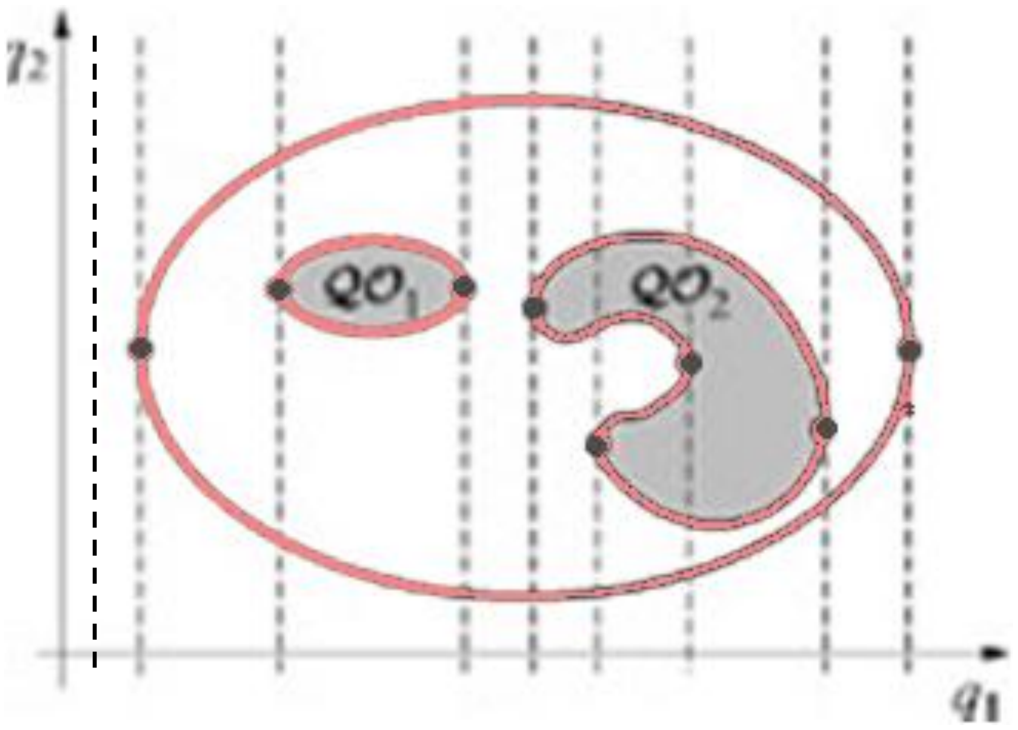
Sweep line construction:
O(nlogn)

Graphsearch: O(nlogn)

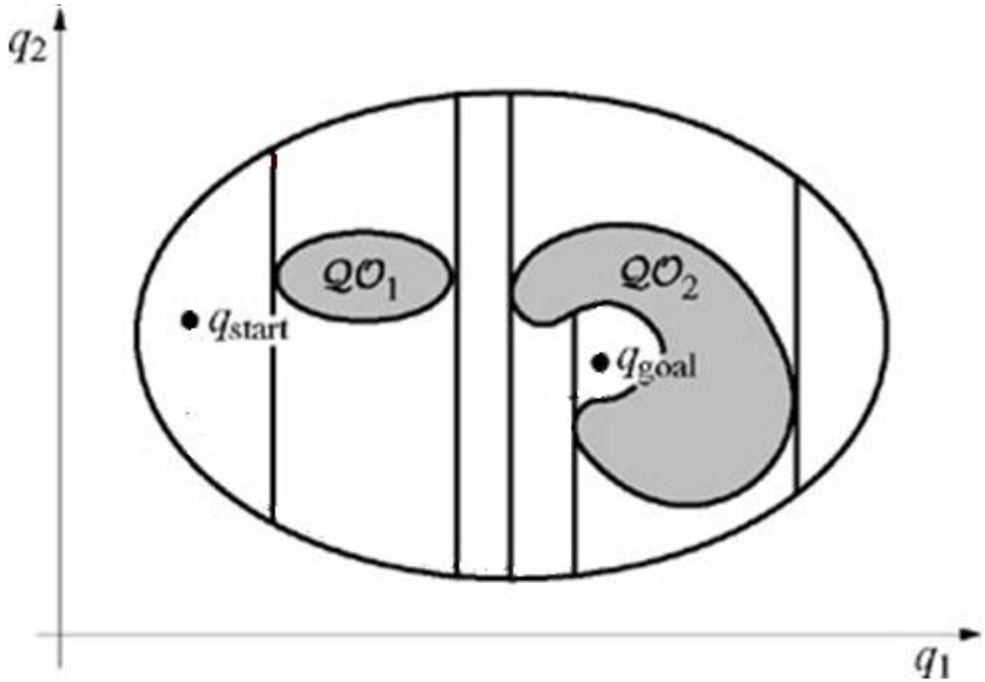
Path: avoids obstacle boundary but has high curvature bends



# Canny's Silhouette roadmap



# Canny's Silhouette roadmap



# Canny's Complexity Analysis

n: = degrees of freedom of robot (dim of C-space)

obstacles C-space boundaries represented as p polynomials of maximum degree w

Complexity:
any navigation path-planning problem can be solved in $p^n(\log p)w^{O(n^4)}$ time

# Probabilistic Roadmap (PRM)

# Probabilistic Roadmap

Nodes V and edges E are obtained via monte carlo sampling of the C-space.

NO NEED to construct actual C-space.

# Probabilistic Roadmap

Sample n poses $q_1 \dots q_n$ in the WORKSPACE

**Free space nodes**: Reject $q_i$ that intersect with an obstacle, remaining nodes $q$ are in $Q_{free}$

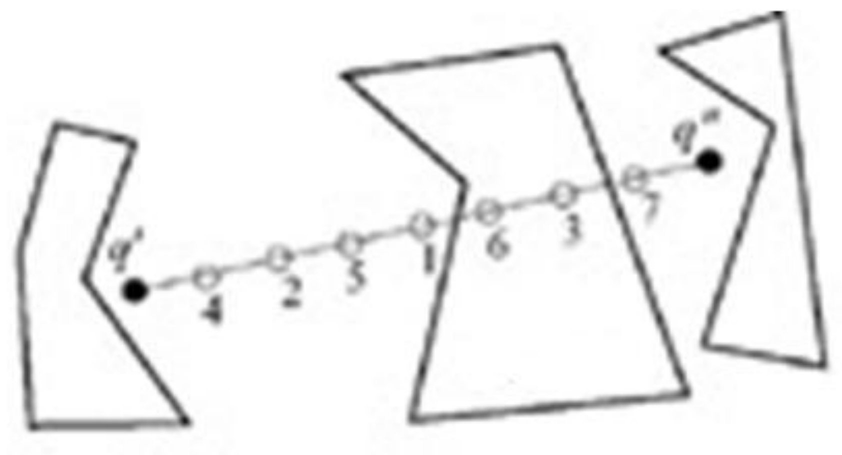
**Local planning**: in k-nearest neighbours, if path $<q_i,q_j>$collision-free, add edge to graph

Resulting graph = *Probabilistic Roadmap*

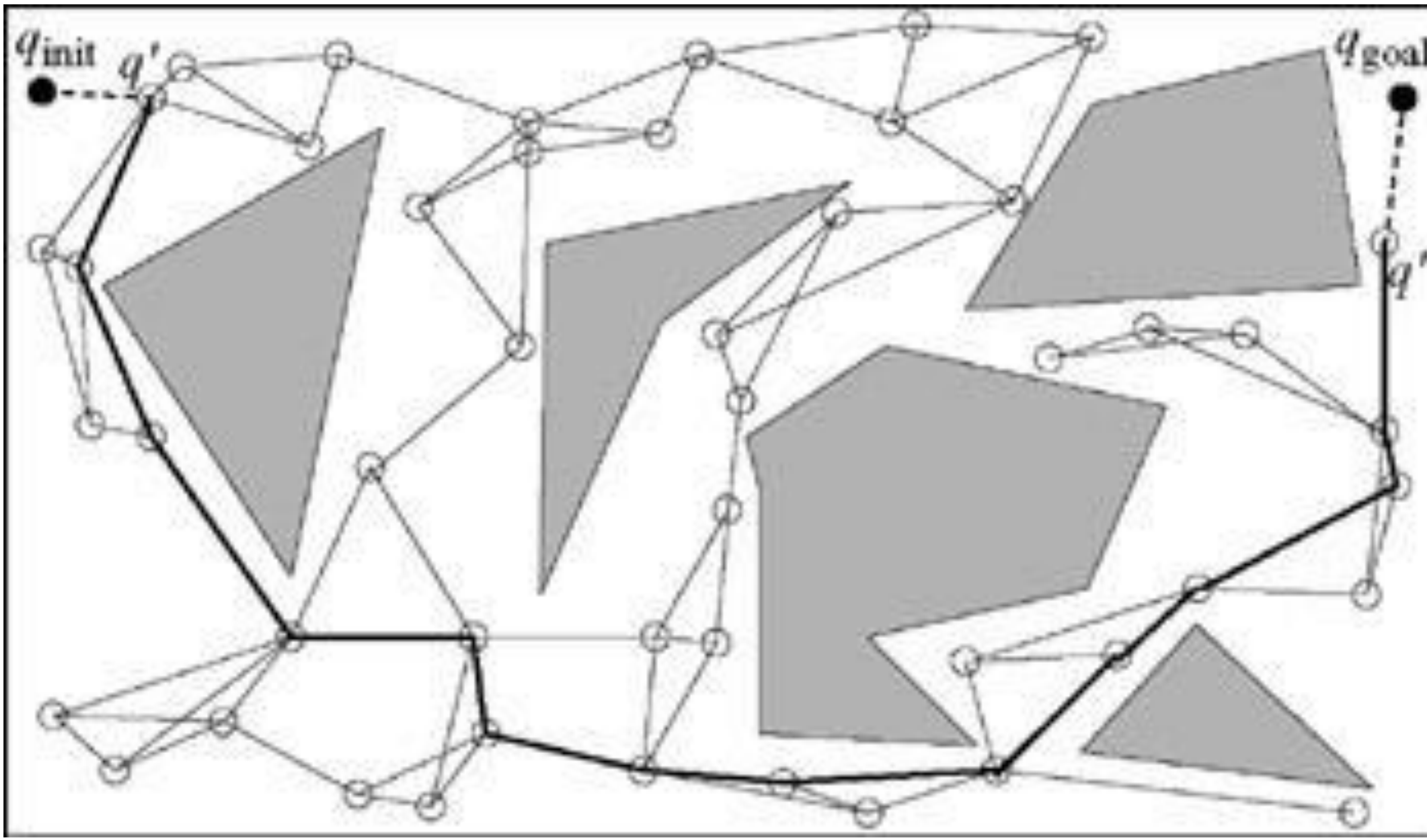
# Local Planner

Objective: Test if path <$q_i$,$q_j$> is collision-free

Linear Subdivision algorithm: start at midpoint($q_i$,$q_j$) ; subdivide recursively until desired precision



# Probabilistic Roadmaps (PRM)



# Sampling-based motion planning

Sample n poses $q_1 \ldots q_n$ in the workspace

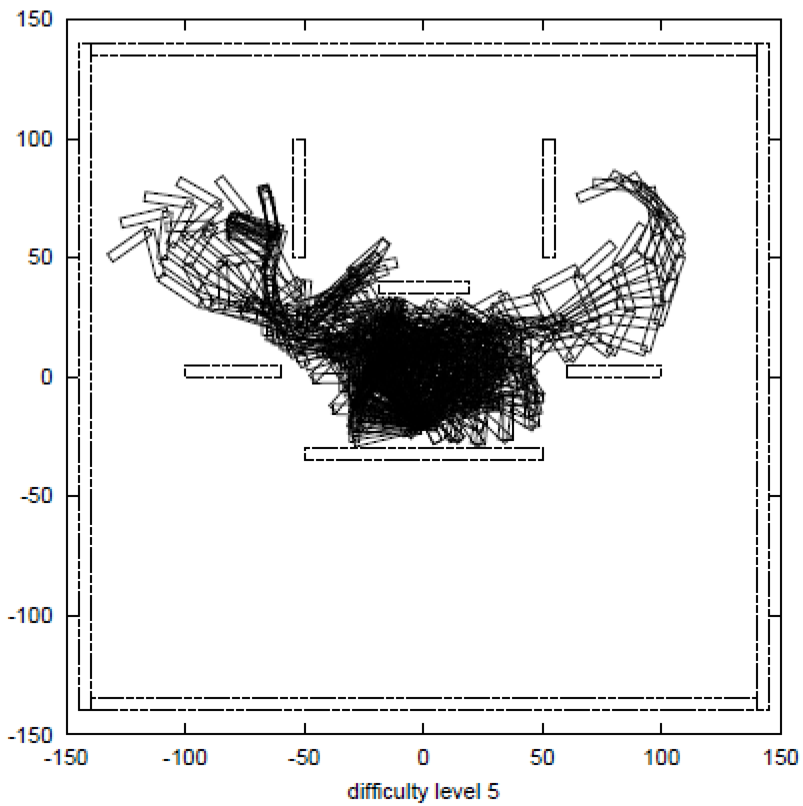
Reject $q$ that overlap with an obstacle, remaining poses are in $Q_{free}$

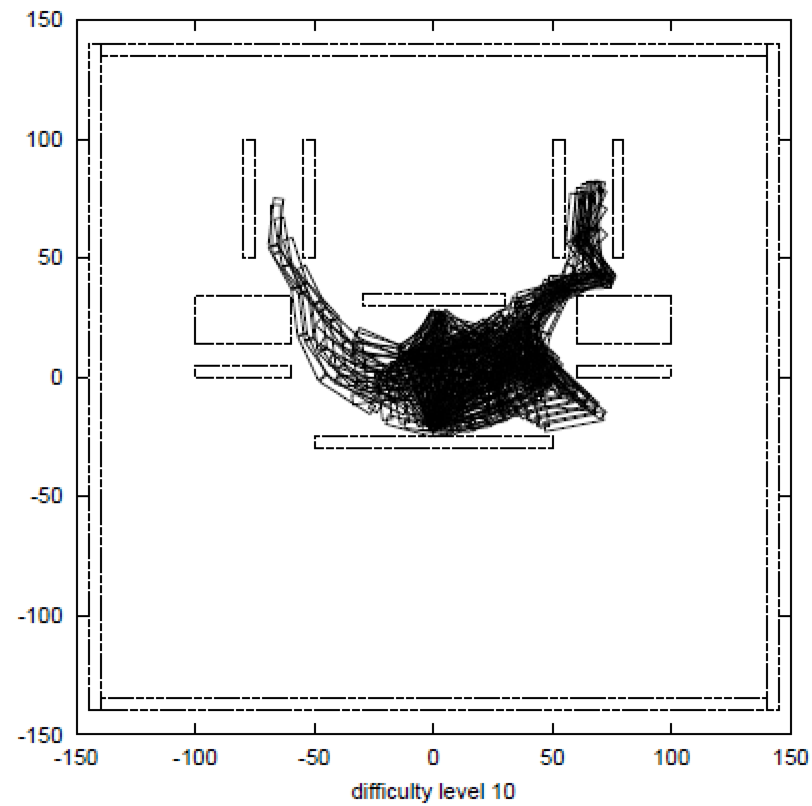
Use local planning to determine if a path exists between neighbours $q_i$ and $q_j$.

Resulting graph = *Probabilistic Roadmap*

Probabilistically complete:
As #samples n → ∞,   Prob (success) → 1

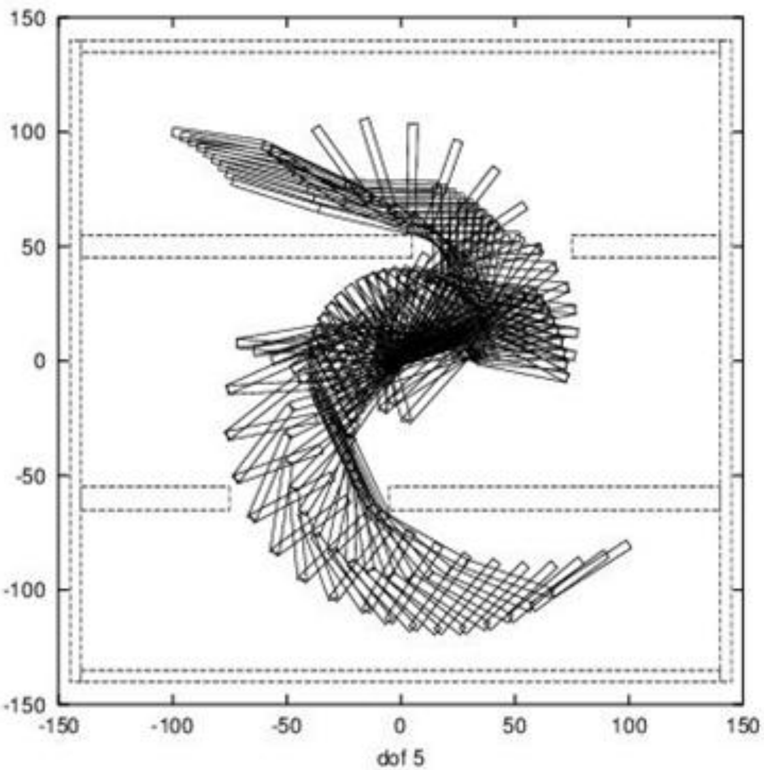
# Hyper-redundant robot motion planning using PRM

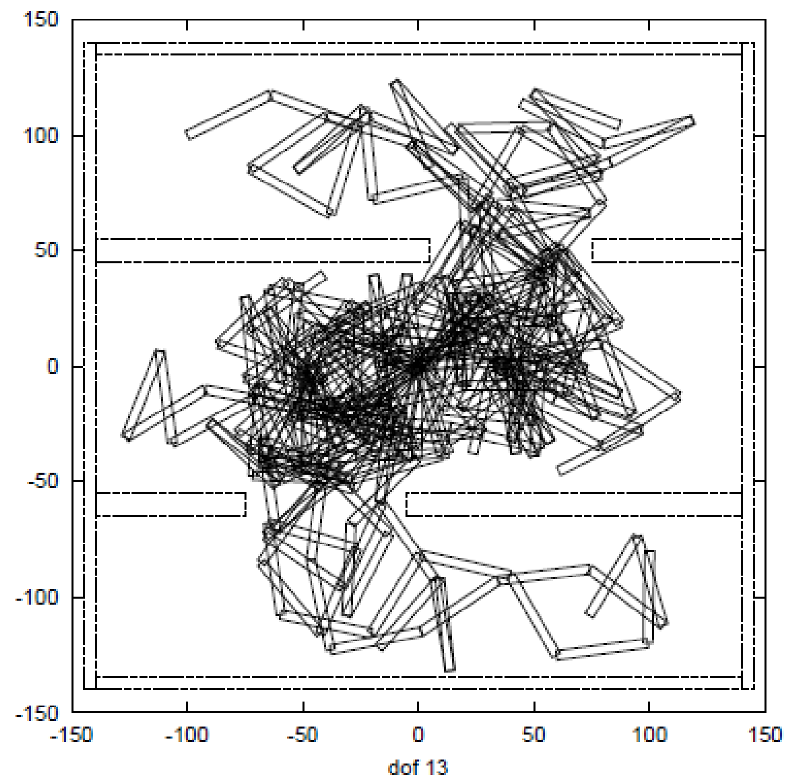



[sinha mukerjee dasgupta 02]

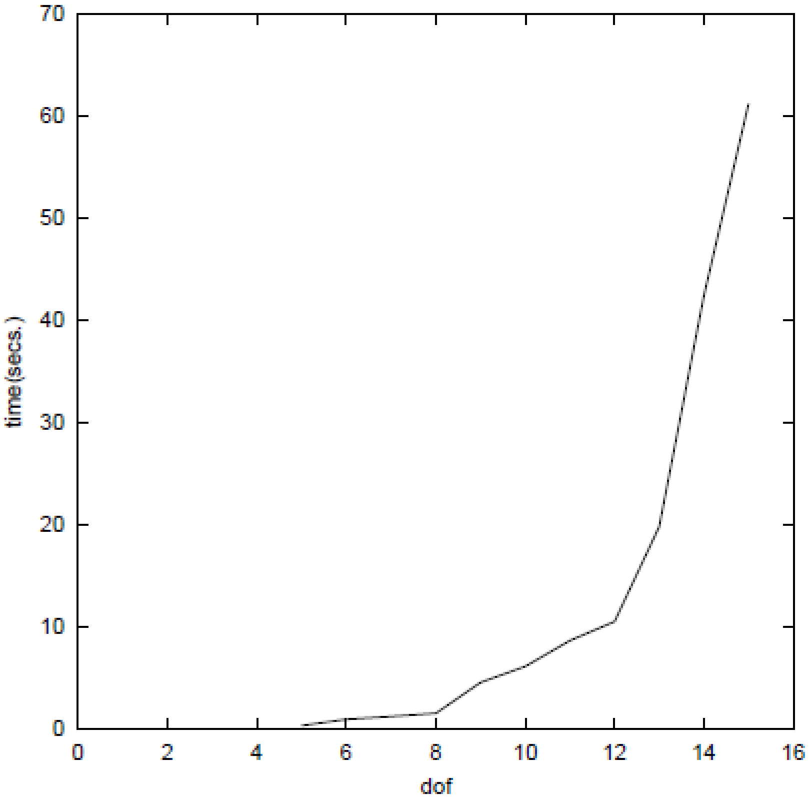
# Hyper-redundant robot motion planning using PRM




[sinha mukerjee dasgupta 02]

# Hyper-redundant motion planning

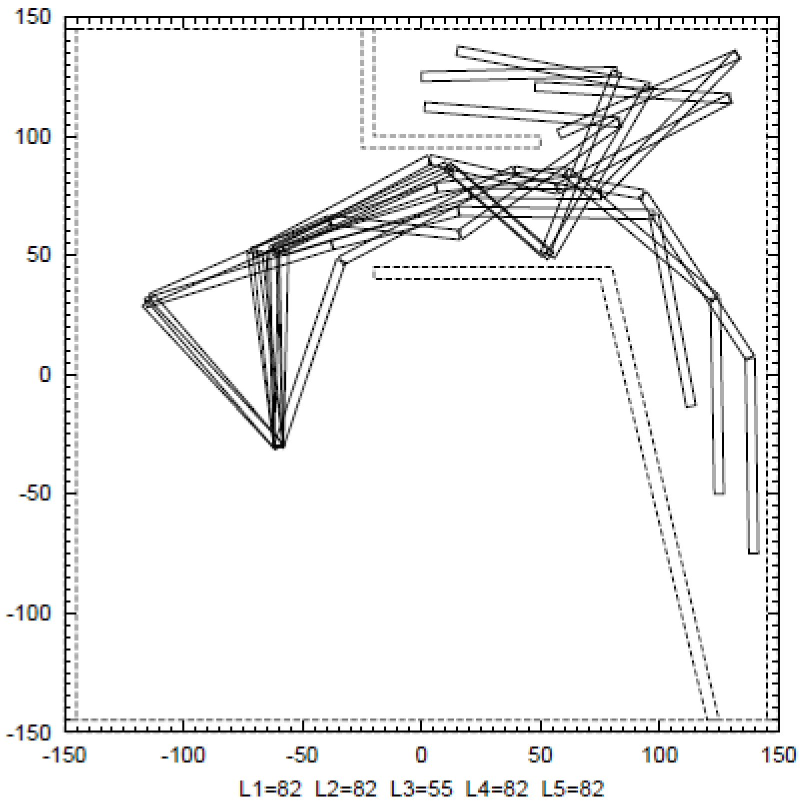


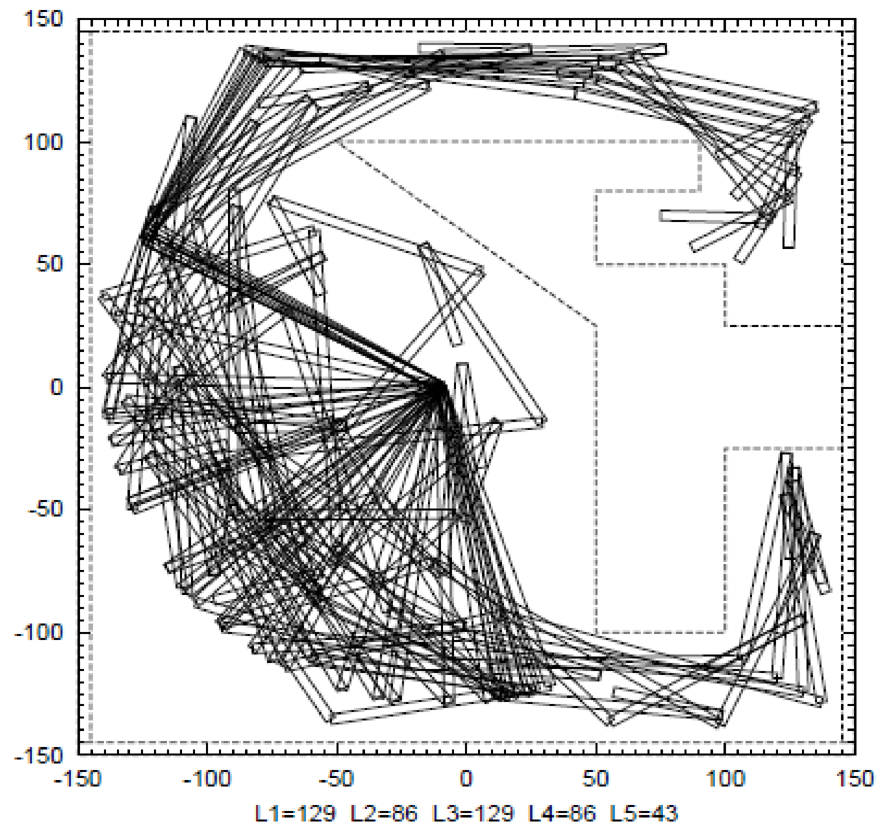

Time:

Exponential in DOFs

[sinha mukerjee dasgupta 02]

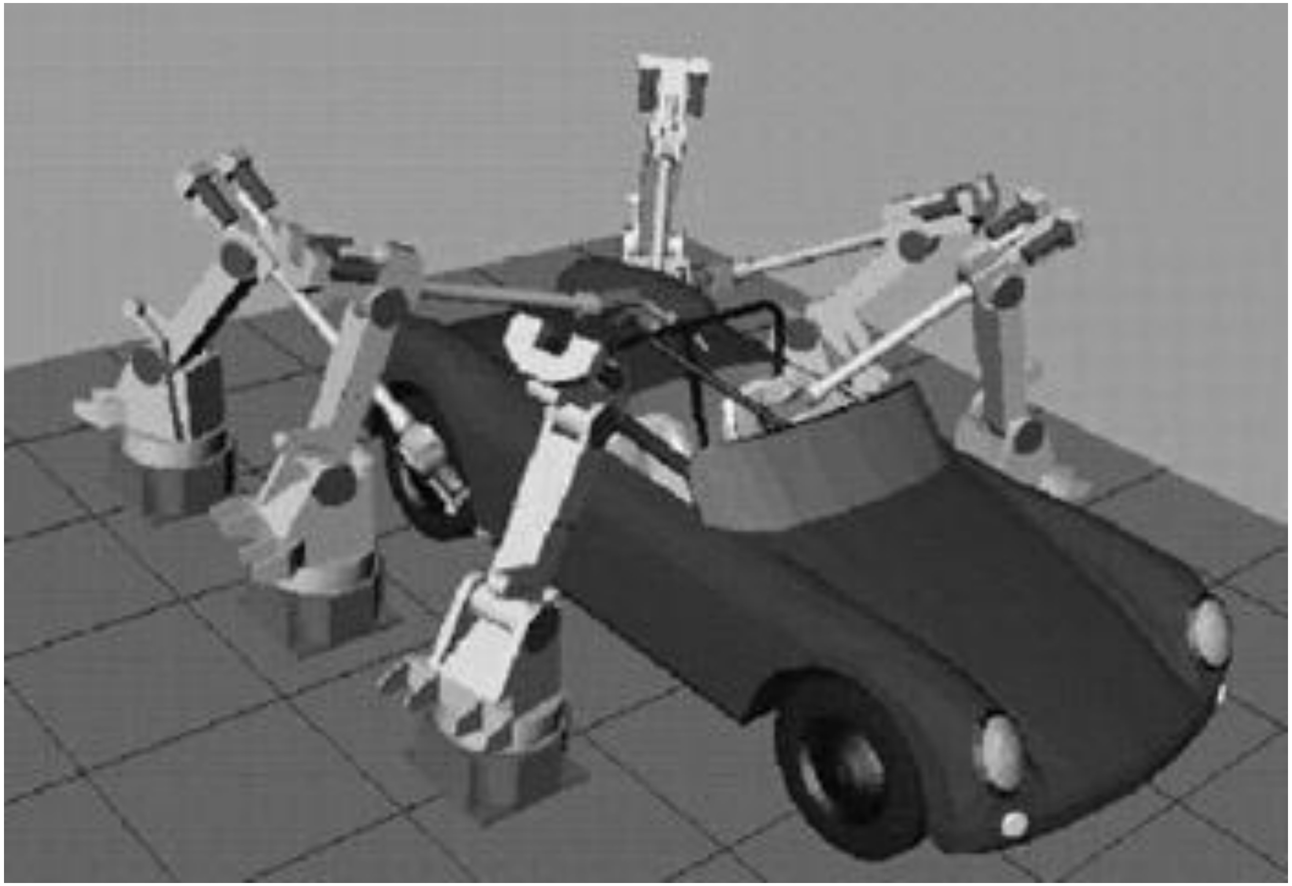
# Design for manipulability



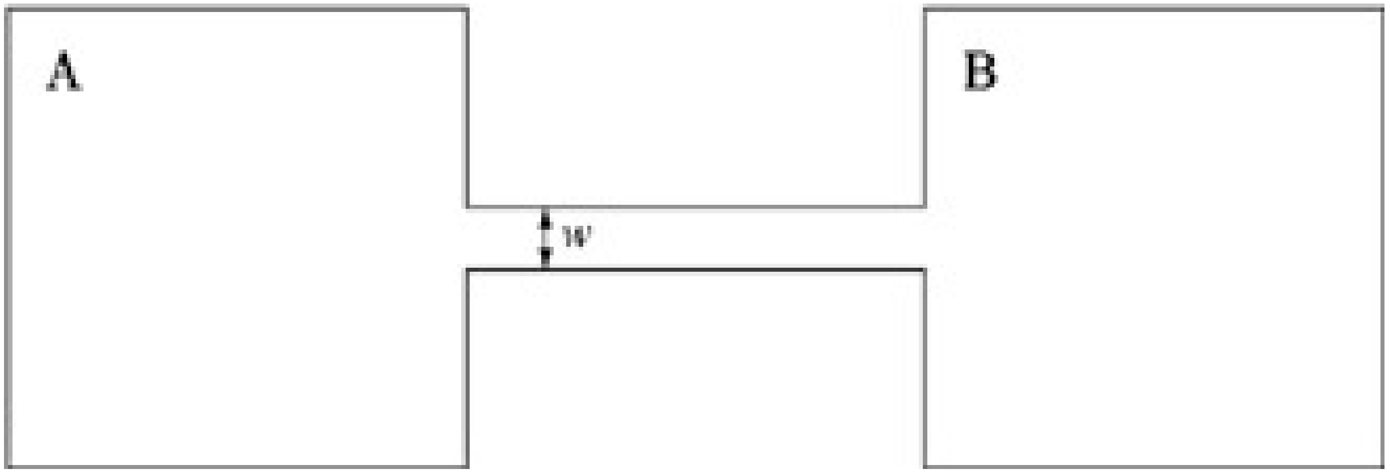

[sinha mukerjee dasgupta 02]

# PRM applications



42 DOFs: [Sánchez and J. C. Latombe 02]

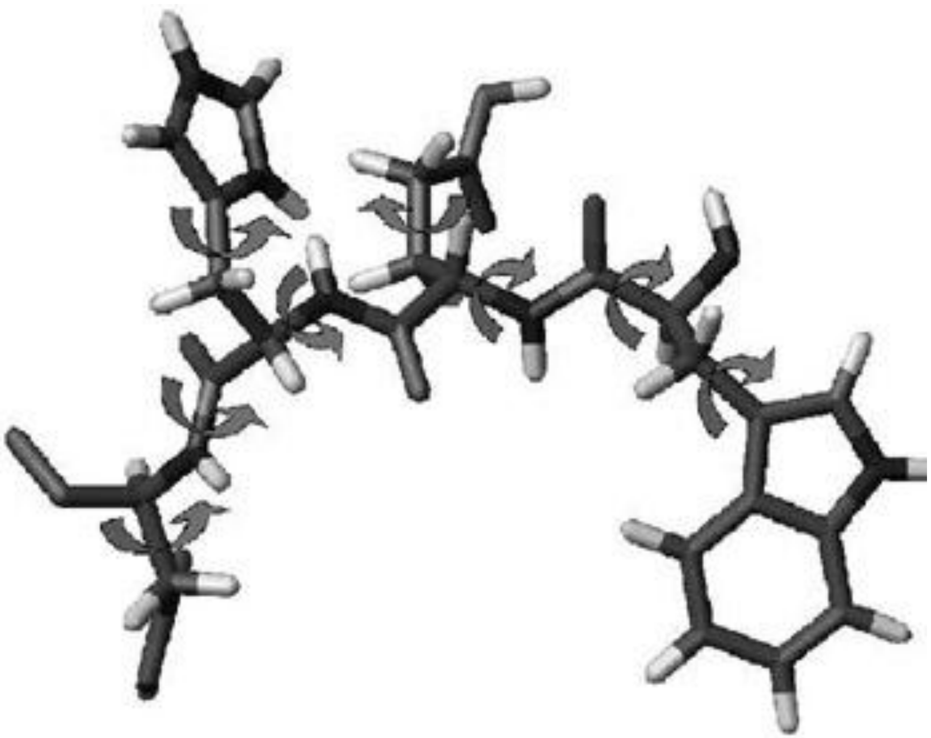
# Narrow corridor problem

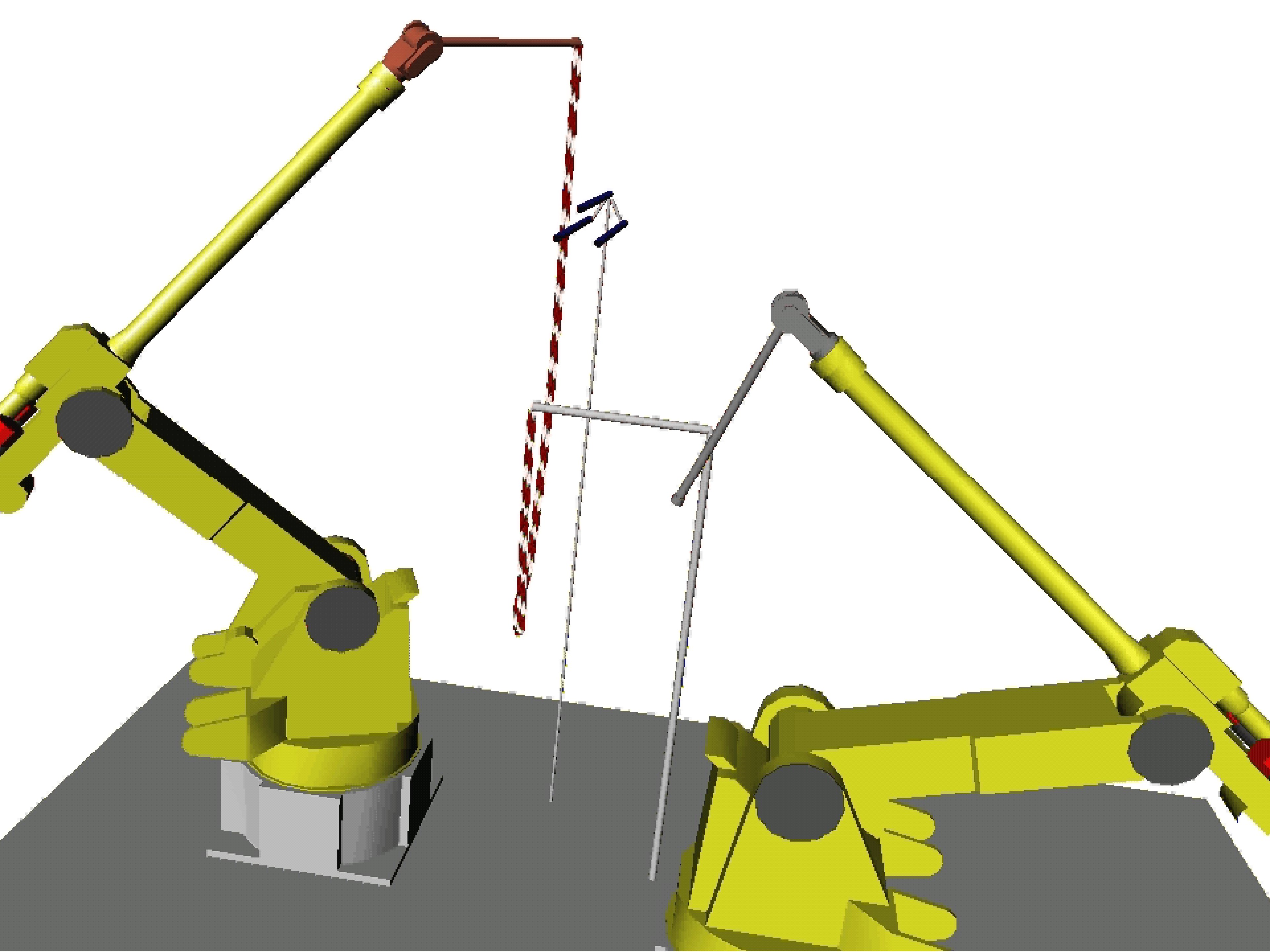



Solution: generate more samples near boundary

- bias the sample towards boundary region
- if midpoint between two obstacle nodes is free, add

# PRM applications : Protein folding



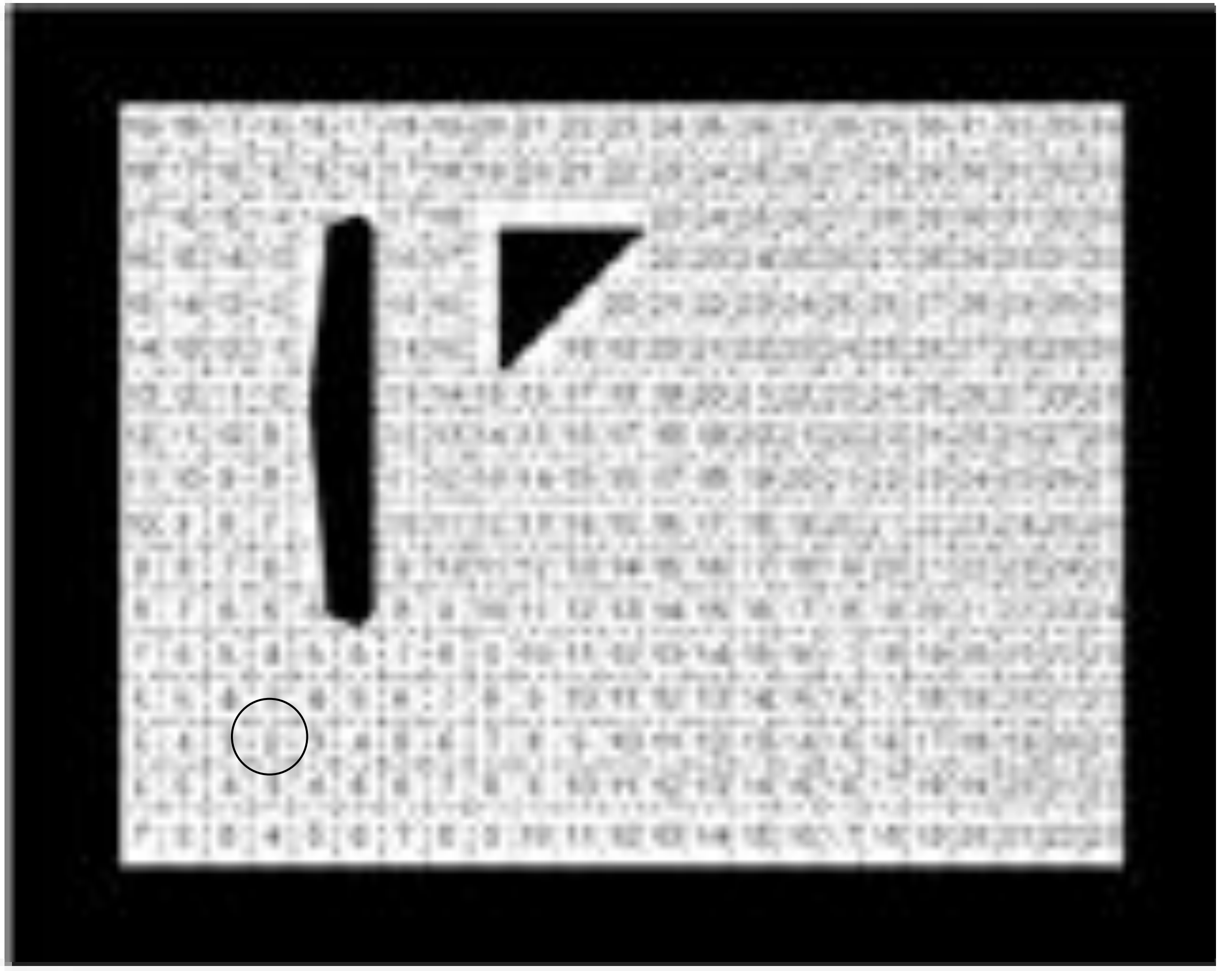


# Continuum methods:
# Overcoming Local minima

# Grid-based: Wave-front

- Grid-based model
- given a start grid cell $\mathbf{q}_S$ assign it the value "2"
  - Every neighbour gridcell gets +1
  - Until grid is filled
- Given a goal cell $\mathbf{q}_G$ use greedy search to find path back to goal

# Grid-based: Wave-front



$O(k^d)$ space / time

# Navigation Function : Sphere space

- Spherical wall ($r_0$), with spherical obstacles inside
- Obstacle distance

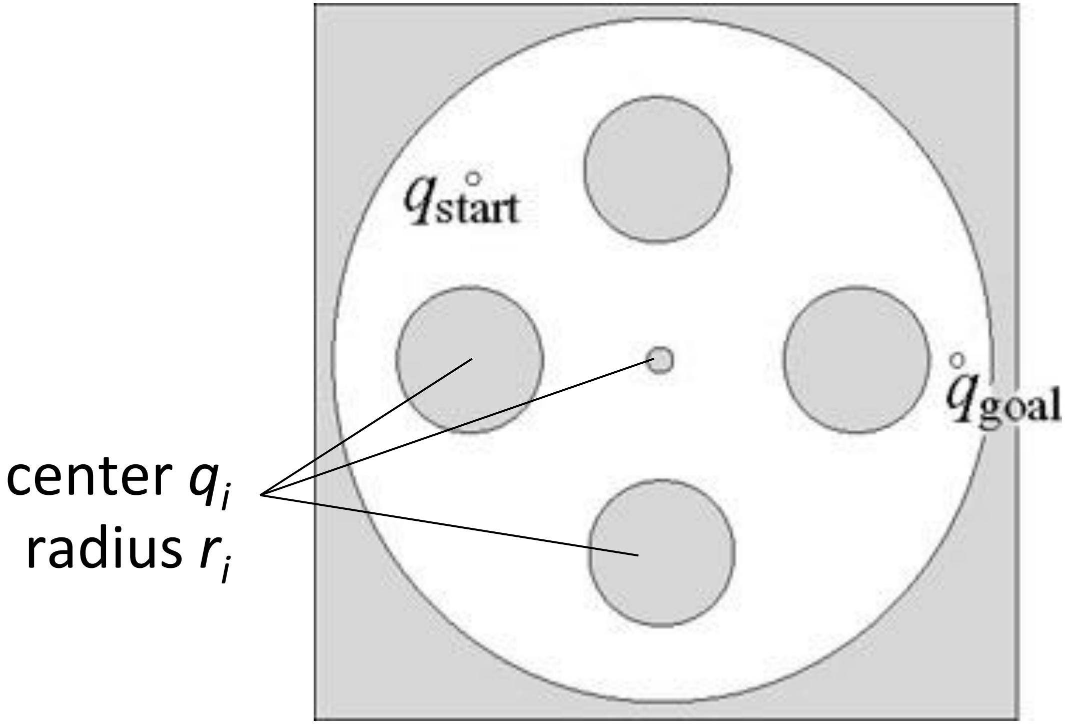
$$QO_i = \{q \mid \beta_i(q) \le 0\}$$

$$\beta_0(q) = -d^2(q, q_0) + r_0^2,$$ — wall

$$\beta_i(q) = d^2(q, q_i) - r_i^2,$$ — obstacles

[Rimon Koditschek 92]

# Sphere space




Rimon Koditschek 92

# Navigation Function : Sphere space

- Spherical wall ($r_0$), with spherical obstacles inside
- Obstacle distance

$$\beta_0(q) = -d^2(q, q_0) + r_0^2, \text{ — wall}$$

$$QO_i = \{q \mid \beta_i(q) \leq 0\}$$

$$\beta_i(q) = d^2(q, q_i) - r_i^2, \text{ — obstacles}$$

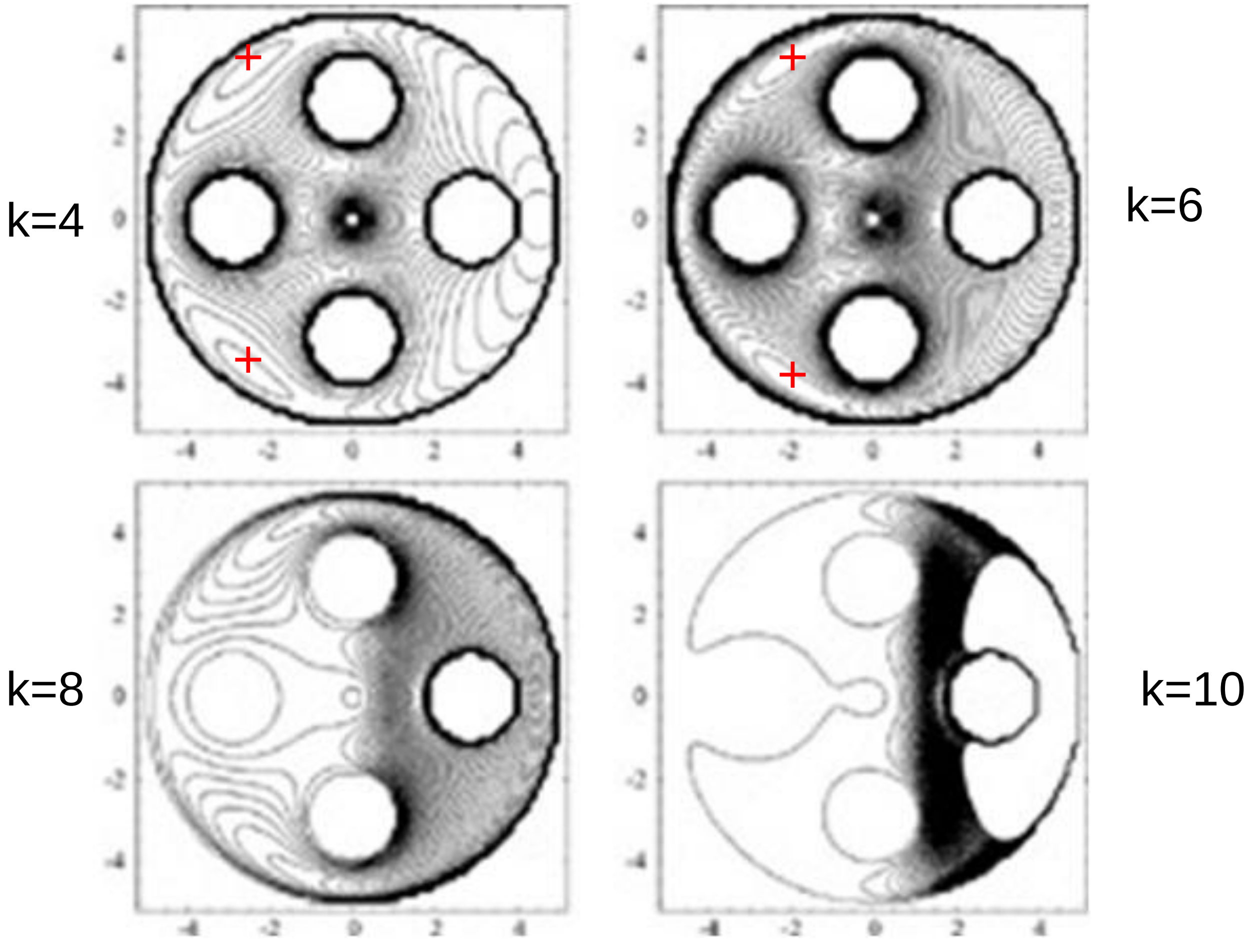
- Goal potential with high exponent $\gamma_\kappa(q) = (d(q, q_{\text{goal}}))^{2\kappa}$
- Instead of sum, use product to combine obstacle potentials

$$\beta(q) = \prod_{i=0}^{n} \beta_i(q).$$

- For high k, $\frac{\gamma_\kappa}{\beta}(q)$ has unique minima at goal

[Rimon Koditschek 92]

# Navigation Function



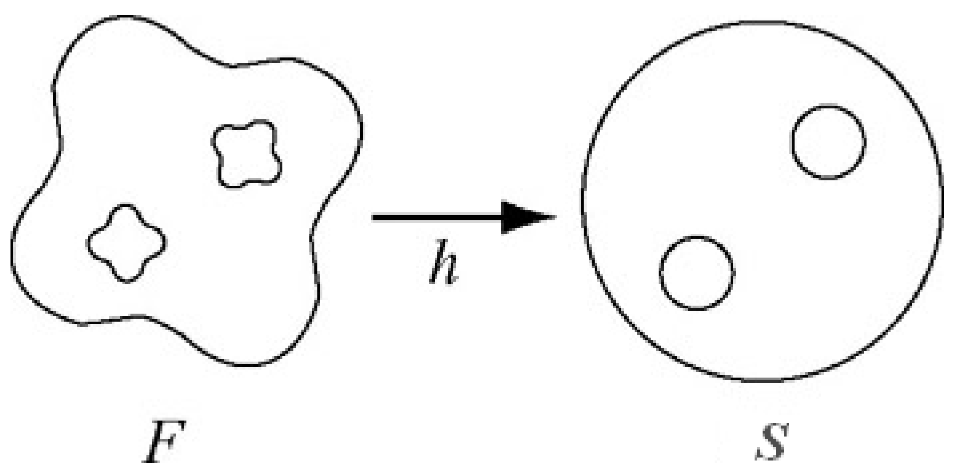

Choset etal 05

# Navigation Function

φ : *S* → [0, 1] :
navigation function on sphere space S.

For any space F if exists diffeomorphic mapping *h* : *F* → *S*
(i.e. h is smooth, bijective, and has a smooth inverse),

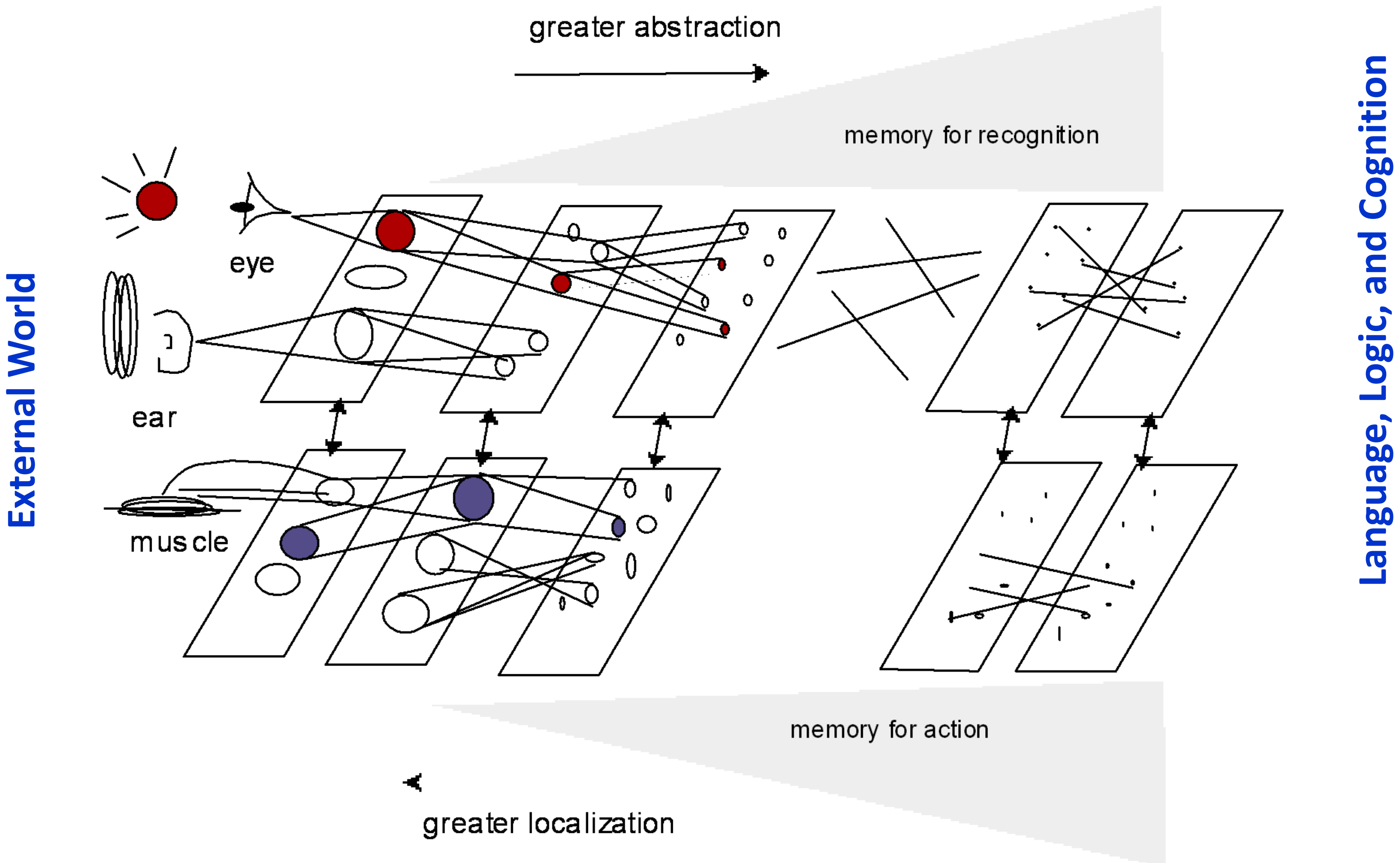
then φ = φ∘ *h* is a navigation function on *F*




Choset etal 05

# Sensori-motor map learning

# Cognitive Architecture: Levels of Abstractions



# Visuo-Motor expertise

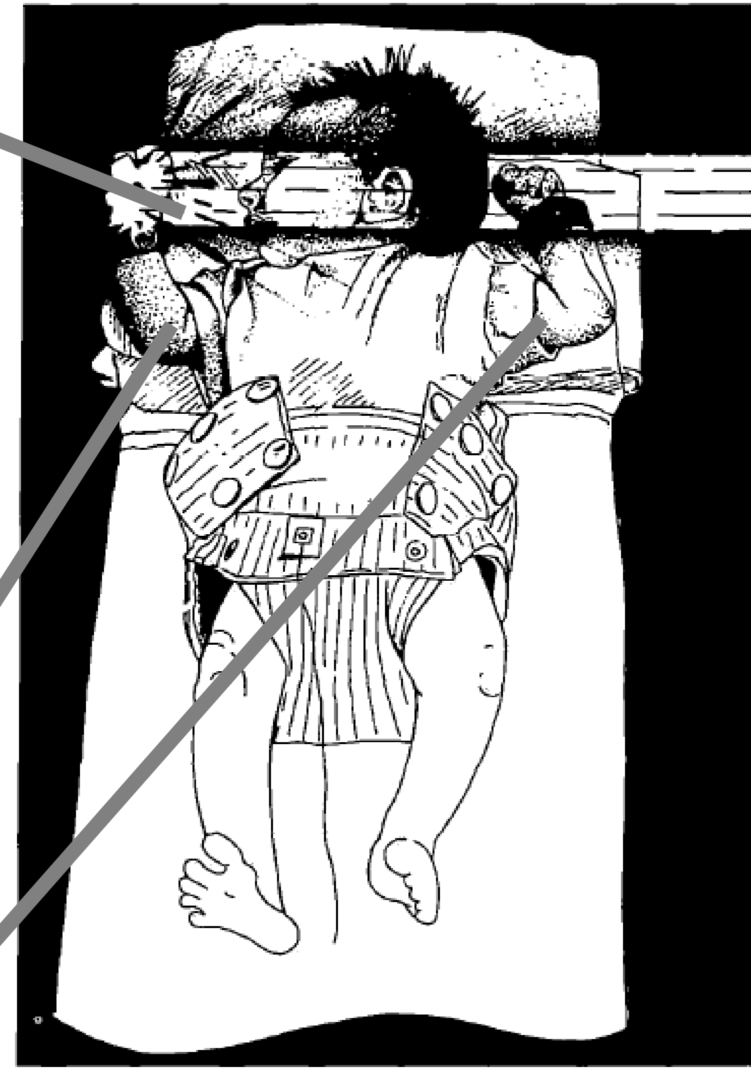
in darkened room,
works hard to position arm
in a narrow beam of light

Newborns
(10-24 days)

Small weights
tied to wrists

Will resist weights to move
the arm they can see

Will let it droop if
they can't see it



[A. van der Meer, 1997: Keeping the arm in the limelight]

# Observing self motions




# Mobility and Intelligence

The capacity to predict the outcome of future events—critical to successful movement— is, most likely, the
ultimate and most common of all global brain functions.

- Rodolfo Llinas

# Motricity → Nervous system

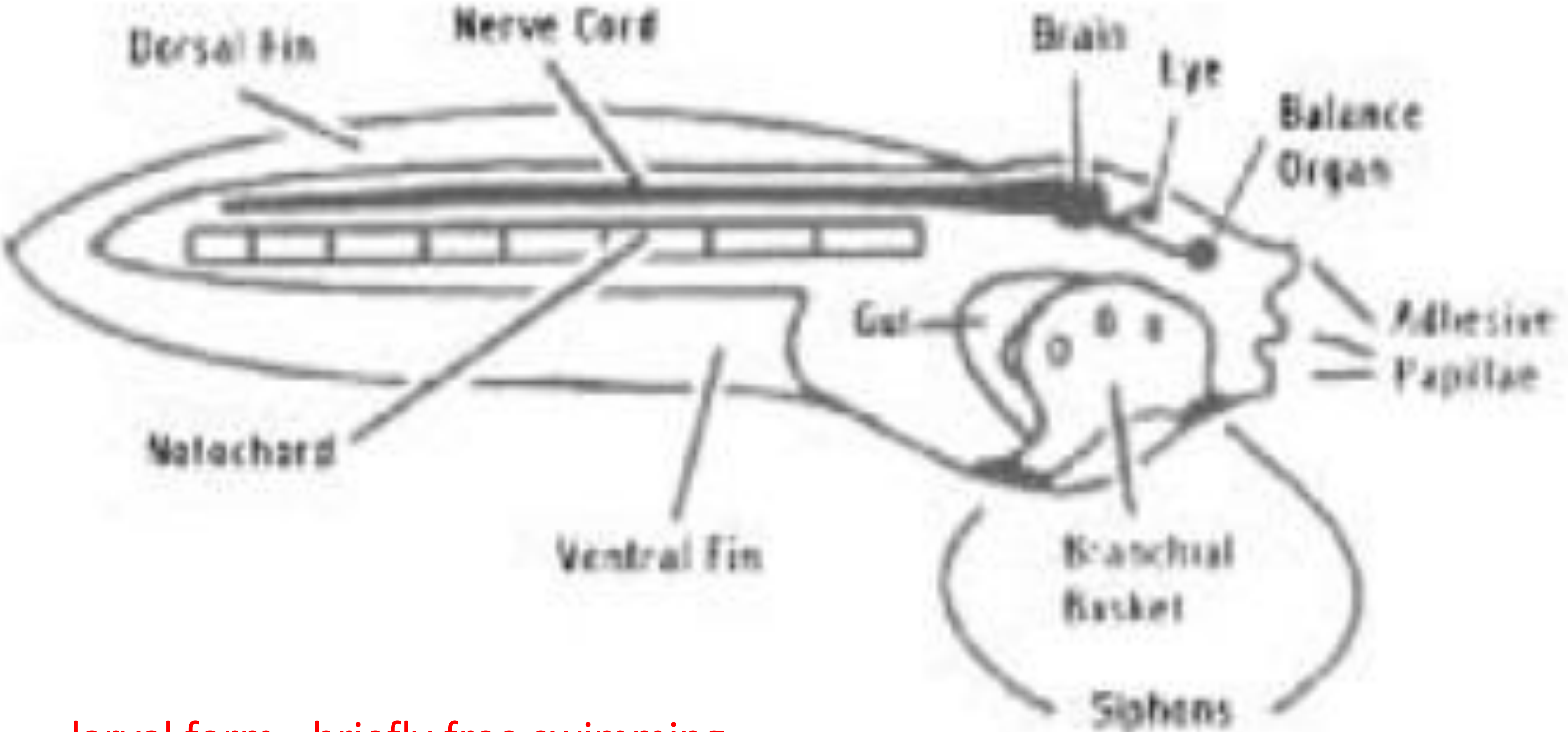
Tunicates (sea squirts) : stage in evolution of chordata

larval form - briefly free swimming

larva has 300 cell ganglion + notochord

# Motricity → Nervous system

Tunicates (sea squirts) : larva – free flying form




larval form - briefly free swimming

larva has 300 cell ganglion + notochord

# Motricity → Nervous system

Tunicates (sea squirts) : stage in evolution of chordata

adult - immobile (sessile)

nervous system – digests it after it finds and attaches to a site

# Predicting → Planning

# Movement and the "mind"

Rodolfo Llinas, *The I of the Vortex:*

- *Itch on the back* : generates a sensorimotor image
- The image *pulls* toward the action to be performed
- Brain has evolved as
  - goal-oriented device
  - inherited, pre-wired mechanism, implements predictive / intentional interactions w environment.
  - requires creating internal image of the world for comparing sensory data
- Mind is "co-dimensional" with the brain
- Generates "self-controlled" electrical storms - Emergent

# Designing motion algorithms

A. Engineering approach:

- Model the robot's body (geometry + kinematics)
- Model the obstacles
- find path P from $q_S$ to $q_G$ s.t. for all $q \in P$, $R(q) \cap B = \emptyset$

B. Cognitive Approach

- Use early experience to learn correlation between motor to sensory spaces
- Configuration coordinate is NOT KNOWN
- Map obstacles and find path in this space