CS 4649/7649
Robot Intelligence: Planning

Differential Kinematics, Probabilistic Roadmaps

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

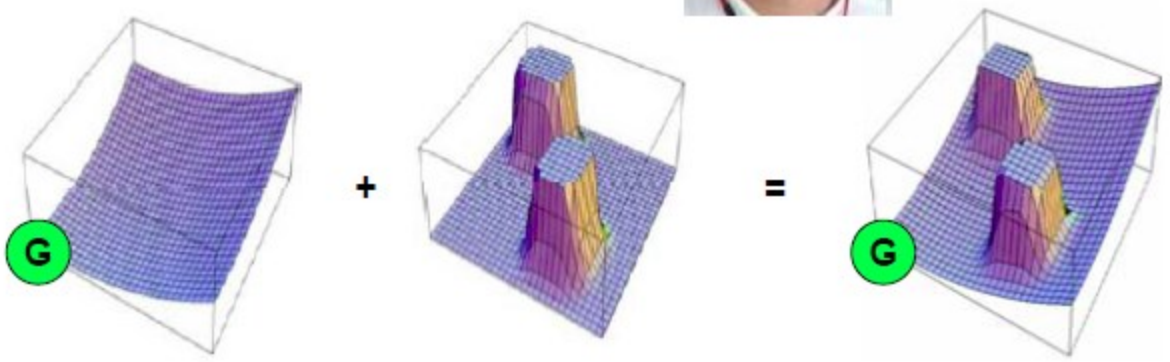
*Slides based in part on Dr. Mike Stilman's lecture slides

Potential Fields

- Potential Function
  - $U_a(q)$ Attracts to goal
  - $U_r(q)$ Repels from obstacles

Oussama Khatib

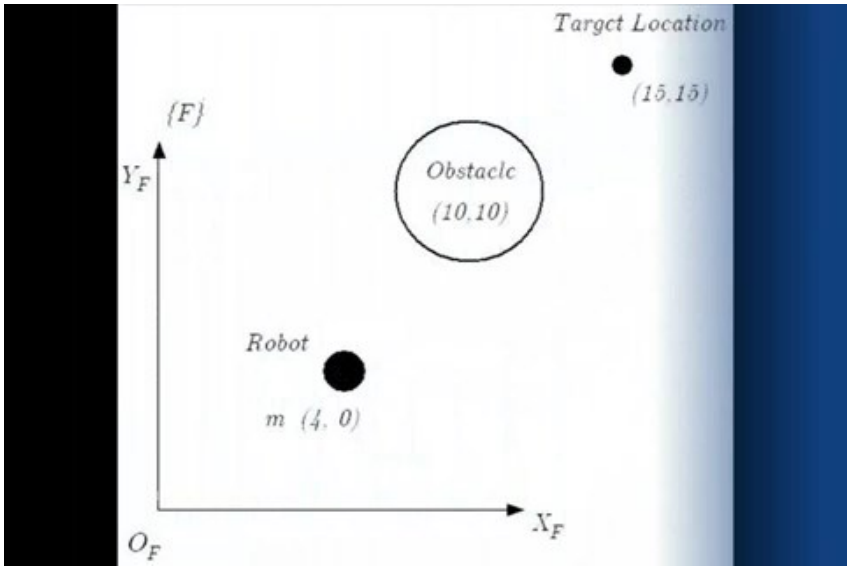
'86

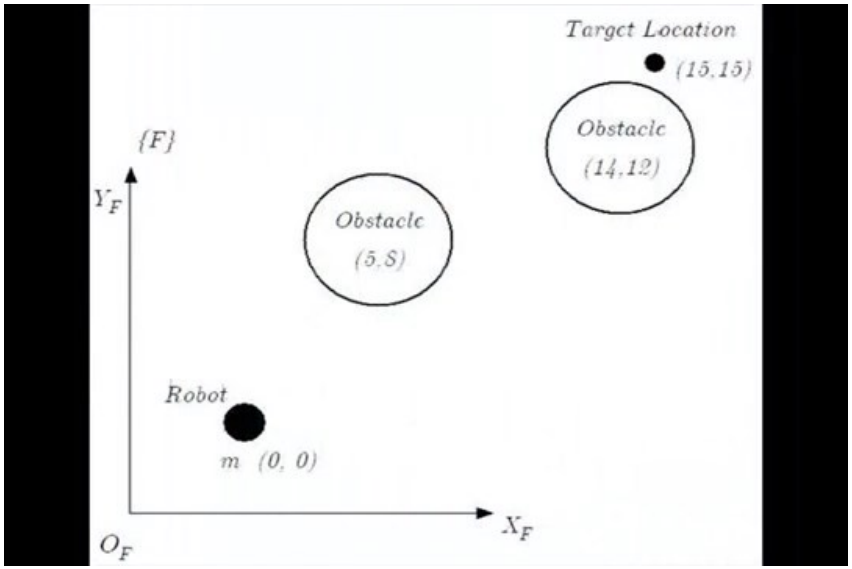


$$U_a(q) + U_r(q) = U(q)$$

- Typically smooth **Because we follow gradient: $\nabla U(q)$**

Potential Fields





*Simulation by Leng-Feng Lee@Buffalo Univ.

Summary: Navigation Algorithms

	Complete	Optimal	Efficiency	Model Required
Bug 1	Yes	No	~	No
Bug 2	Yes	No	Usually > B1	No
Visibility	Yes	Goal Dist	$n^2 \log n + A^*$	Yes
Voronoi	Yes	Obs Dist	$n \log n? + A^*$	Yes
Voronoi Bug	Yes	Obs Dist	~	No
Voronoi Brushfire	Resolution	Obs Dist	~ # cells	Yes
Exact Cell	Yes	No	$n \log n + A^*$	Yes
Approximate Cell	Resolution	Manh. Dist.	~ # cells	Yes
Potential Fields	No	Locally	Linear	Yes

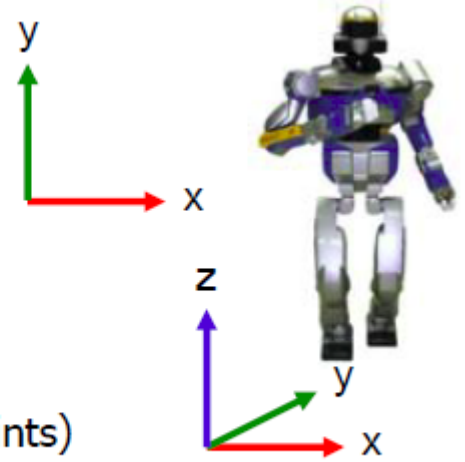
Coordinates

- **Coordinate System:** A set of numbers that specifies configuration
  - Points
  - Rigid Bodies
  - Articulated Manipulators

- **Degrees of Freedom (DOF):** Minimal number of independent coordinates

- How many DOF?

– Point in the plane	2 (x,y)
– Point in 3D	3 (x,y,z)
– Body in 2D	3 (x,y, θ)
– Body in 3D	6 (x,y,z, R,P,Y)
– Robot Arm	n (# of joints)
– Humanoid Robot	6+n (Body + # of joints)



Homogeneous Transform 3D

$$\mathbf{T}_B^A = \begin{bmatrix} & \mathbf{R}_B^A & & \mathbf{t}_B^A \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} \mathrm{p} &= x^A \hat{x}_A + y^A \hat{y}_A + z^A \hat{z}_A \\ &= x^B \hat{x}_B + y^B \hat{y}_B + z^B \hat{z}_B \end{aligned}$$

Interpretations:

- Maps p^B to p^A $\quad p^A = \mathrm{T}_{\mathrm{B}}^{\mathrm{A}} p^B$

(1) Change the representation of a vector in frame B to frame A

- Transform operator: creates p_2^A from p_1^A

(2) Change a vector in a frame to another vector (rotated) in the

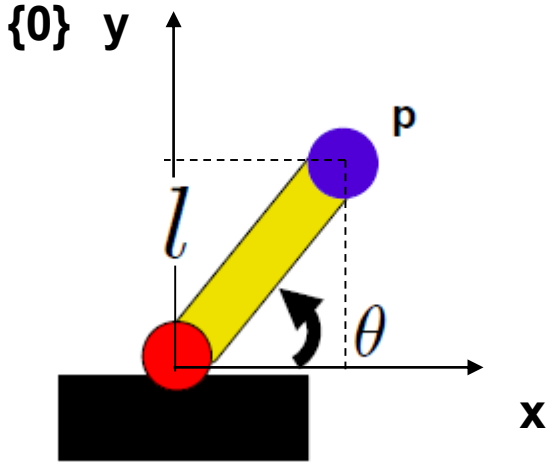
- Describes frame B relative to frame A

t_B^A = position of the frame

$$\mathbf{R}_B^A = \begin{bmatrix} \mathbf{x}_B^A & \mathbf{y}_B^A & \mathbf{z}_B^A \end{bmatrix}$$

(3) Unit vectors of frame B represented(expressed) in frame A

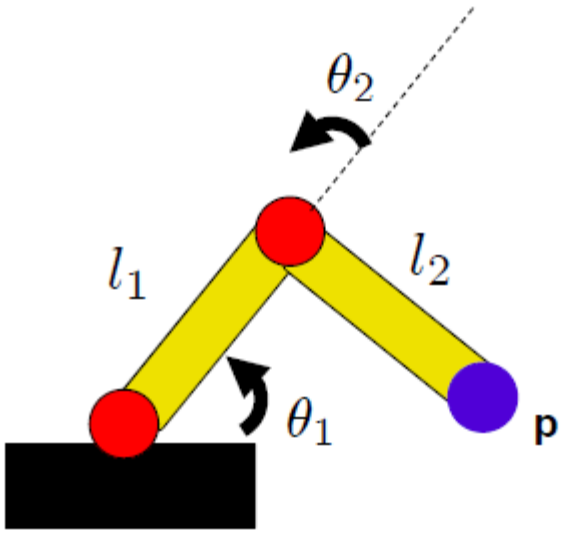
Forward(Direct) Kinematics



1-DOF Robot Arm

$$p_x = l\cos\theta$$
$$p_y = l\sin\theta$$

Forward Kinematics



2-DOF Robot Arm

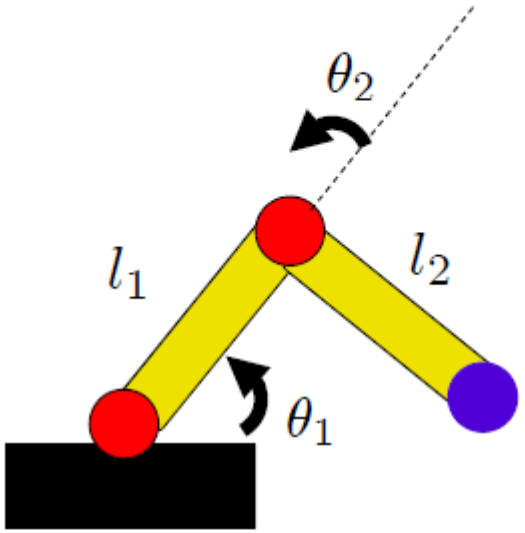
$$p_x = l_1 c_1 + l_2 c_{12}$$
$$p_y = l_1 s_1 + l_2 s_{12}$$

$$c_{12} = \cos(\theta_1 + \theta_2)$$
$$s_{12} = \sin(\theta_1 + \theta_2)$$

$$p_x = l_1 c_1 + l_2 (c_1 c_2 - s_1 s_2)$$

$$p_y = l_1 s_1 + l_2 (s_1 c_2 + c_1 s_2)$$

Forward Kinematics



2-DOF Robot Arm

$$\mathbf{T}_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_2^1 = \begin{bmatrix} c_2 & -s_2 & l_1 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_3^2 = \begin{bmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

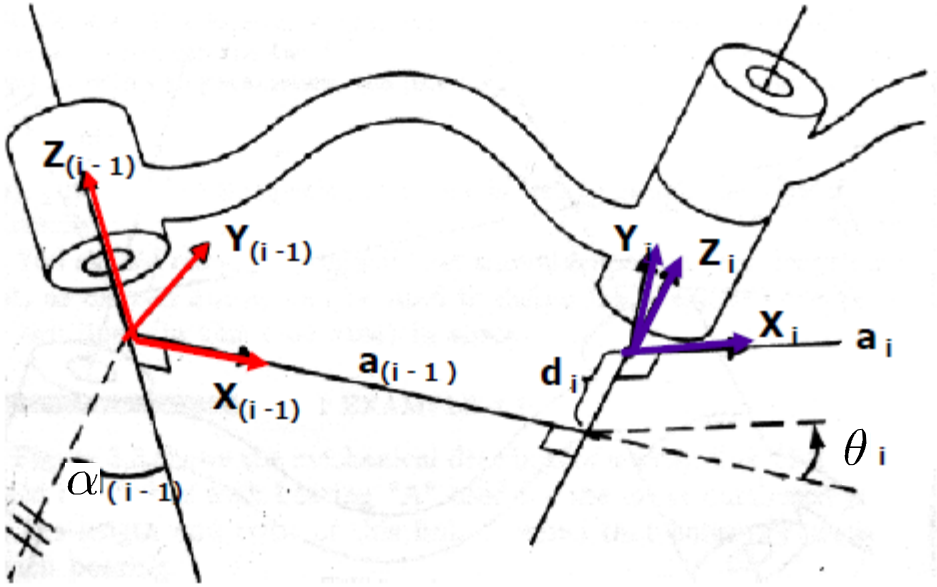
$$\mathbf{T}_2^0 = \mathbf{T}_1^0\mathbf{T}_2^1 = \begin{bmatrix} c_1c_2 - s_1s_2 & -c_1s_2 - s_1c_2 & l_1c_1 \\ s_1c_2 + c_1s_2 & -s_1s_2 + c_1c_2 & l_1s_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{12} & -s_{12} & l_1c_1 \\ s_{12} & c_{12} & l_1s_1 \\ 0 & 0 & 1 \end{bmatrix}$$

Representation of frame2 in frame0

$$\mathbf{T}_3^0 = \mathbf{T}_2^0\mathbf{T}_3^2 = \begin{bmatrix} c_{12} & -s_{12} & l_1c_1 + l_2c_{12} \\ s_{12} & c_{12} & l_1s_1 + l_2s_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

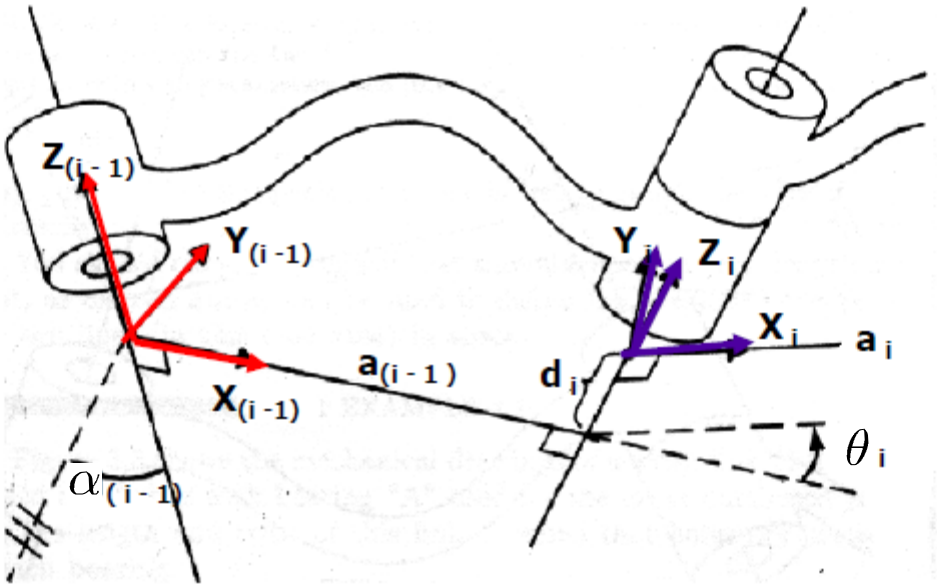
Representation of frame3 in frame0

Denavit-Hartenberg (DH) Parameters



First, label the coordinate frames

DH Parameters



For a revolute joint

Next, calculate the parameters:

link parameters

a_i = the distance from Z_i to Z_{i+1} measured along X_i

α_i = the angle between Z_i and Z_{i+1} measured about X_i

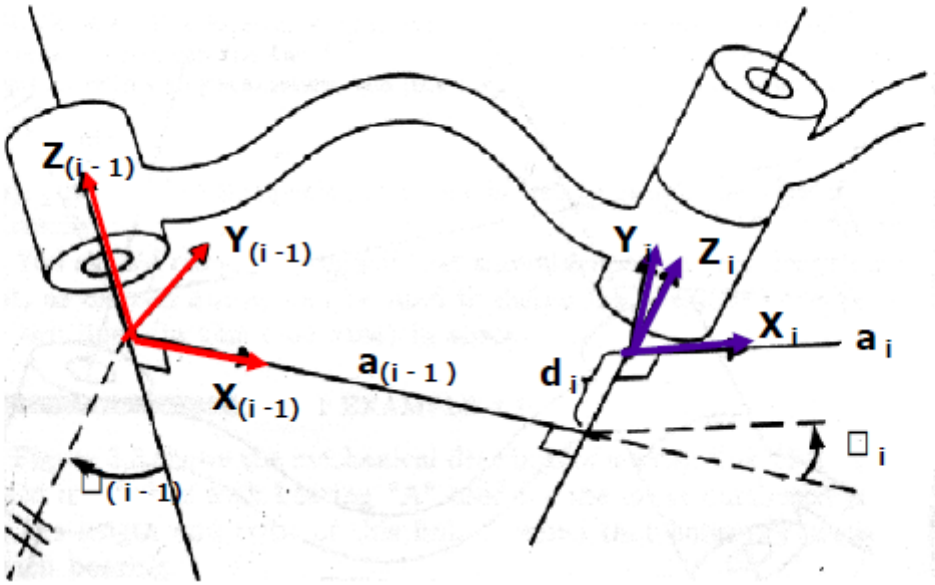
d_i = the distance from X_{i-1} to X_i measured along Z_i

joint variable

θ_i = the angle between X_{i-1} and X_i measured about Z_i

*For a prismatic joint: d_i is a joint variable

DH Parameters



$$\mathbf{T}_i^{i-1} = Rot(X_i, \alpha_{i-1})Trans(X_i, a_{i-1})Rot(Z_i, \theta_i)Trans(Z_i, d_i)$$

$$= \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i & 0 & \alpha_{i-1} \\ \mathrm{s}\theta_i \mathrm{c}\alpha_{i-1} & \mathrm{c}\theta_i \mathrm{c}\alpha_{i-1} & -\mathrm{s}\alpha_{i-1} & -\mathrm{s}\alpha_{i-1} d_i \\ \mathrm{s}\theta_i \mathrm{s}\alpha_{i-1} & \mathrm{c}\theta_i \mathrm{s}\alpha_{i-1} & \mathrm{c}\alpha_{i-1} & \mathrm{c}\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

DH Parameters

- 4 Parameters describe how frame (i) relates to frame (i-1)
- Compact description of manipulator kinematics
- Mechanical method for deriving transformations
- Widely used as a specification for robot manipulators

***special links – first, last**
***special cases – eg. parallel links**

Kinematics

- Forward Kinematics

  Mapping **from** joint angles **to** link positions

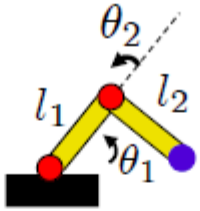
- **Inverse Kinematics**

  Mapping **from** link positions **to** joint angles

  Goals are defined in world coordinates, not joints coordinates.

Inverse Kinematics

Now, given x_3 and y_3 solve for θ_1 and θ_2



- Equations are nonlinear (lots of Trig)

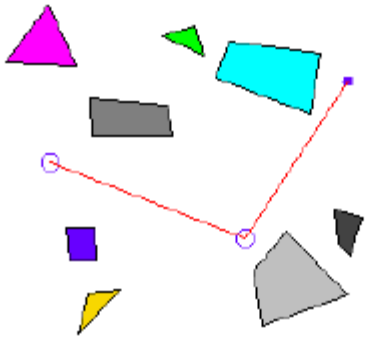
$$\mathbf{T}_3^0 = \begin{bmatrix} c_{12} & -s_{12} & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_3 = l_1 c_1 + l_2 c_{12}$$
$$y_3 = l_1 s_1 + l_2 s_{12}$$

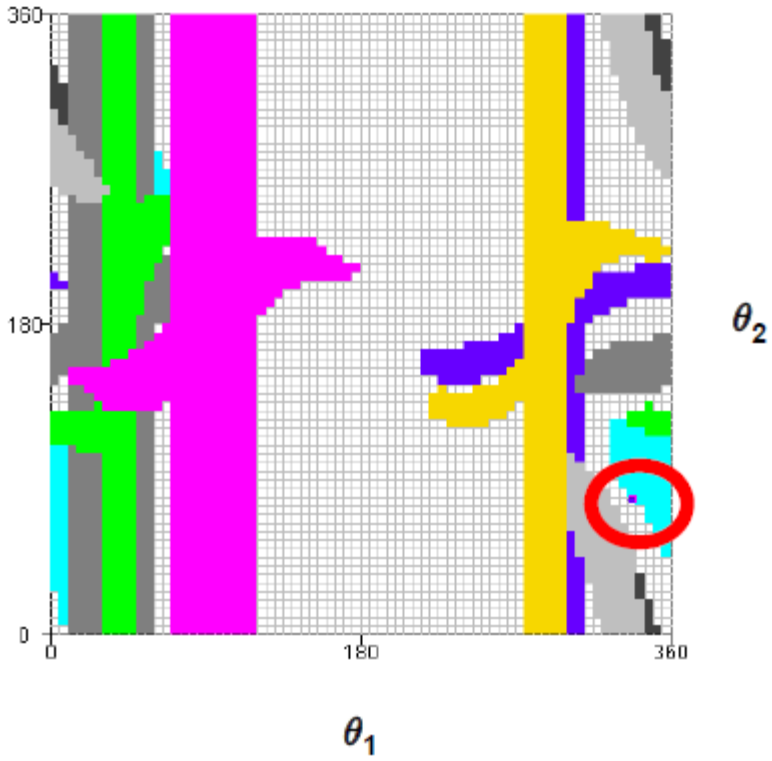
- Multiple Solutions
- Not always possible to find closed-form solution

Configurations Space

Work Space

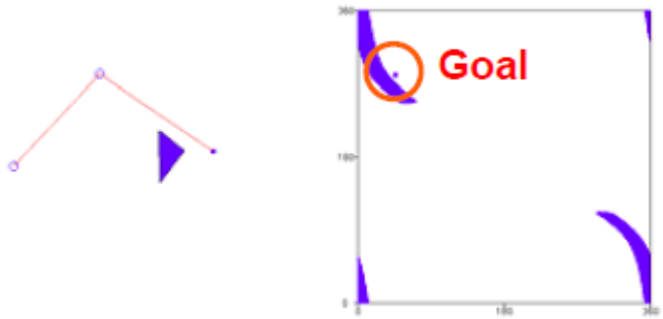


Configuration Space



Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

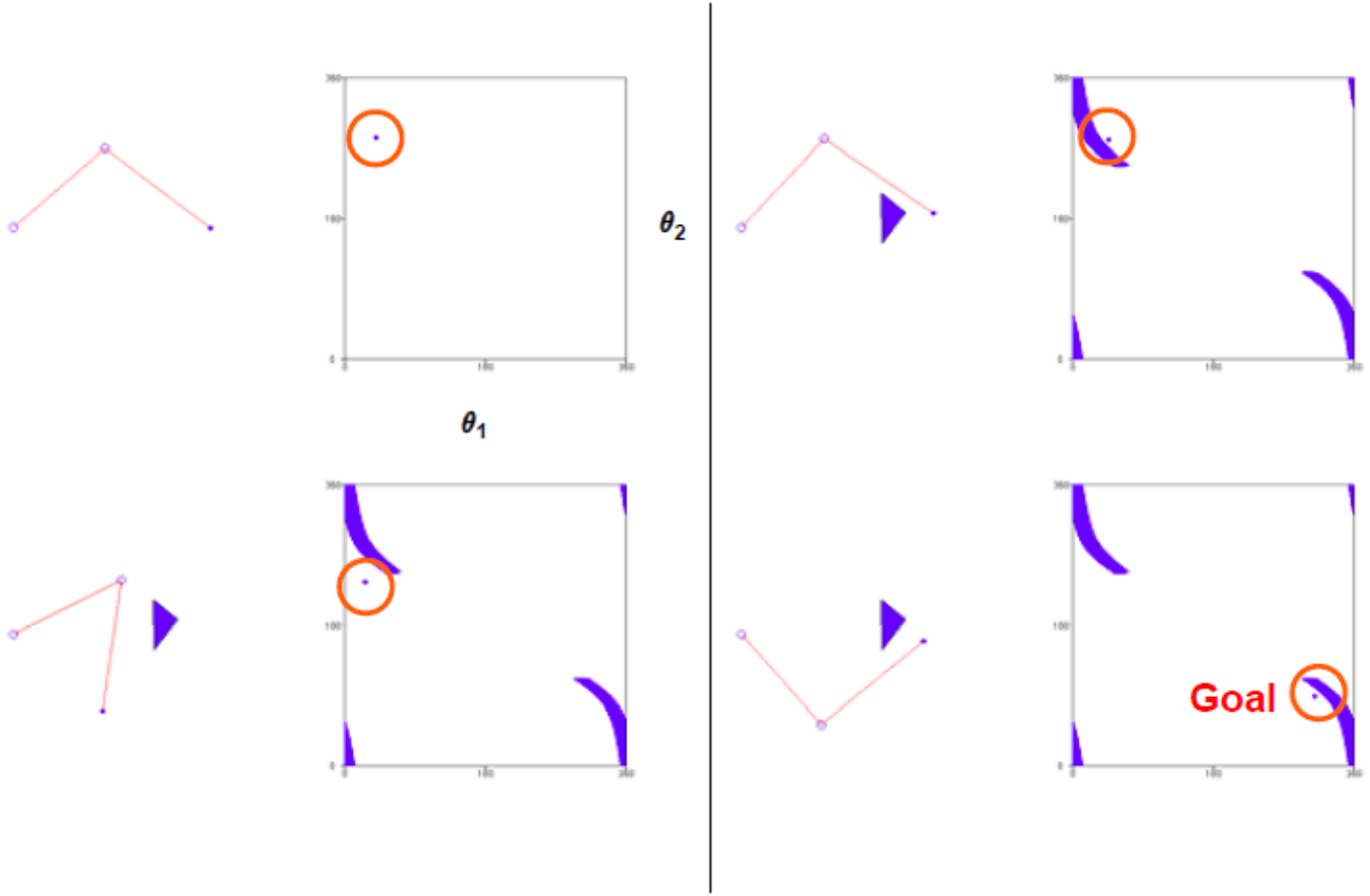
Configuration Space: IK Solution



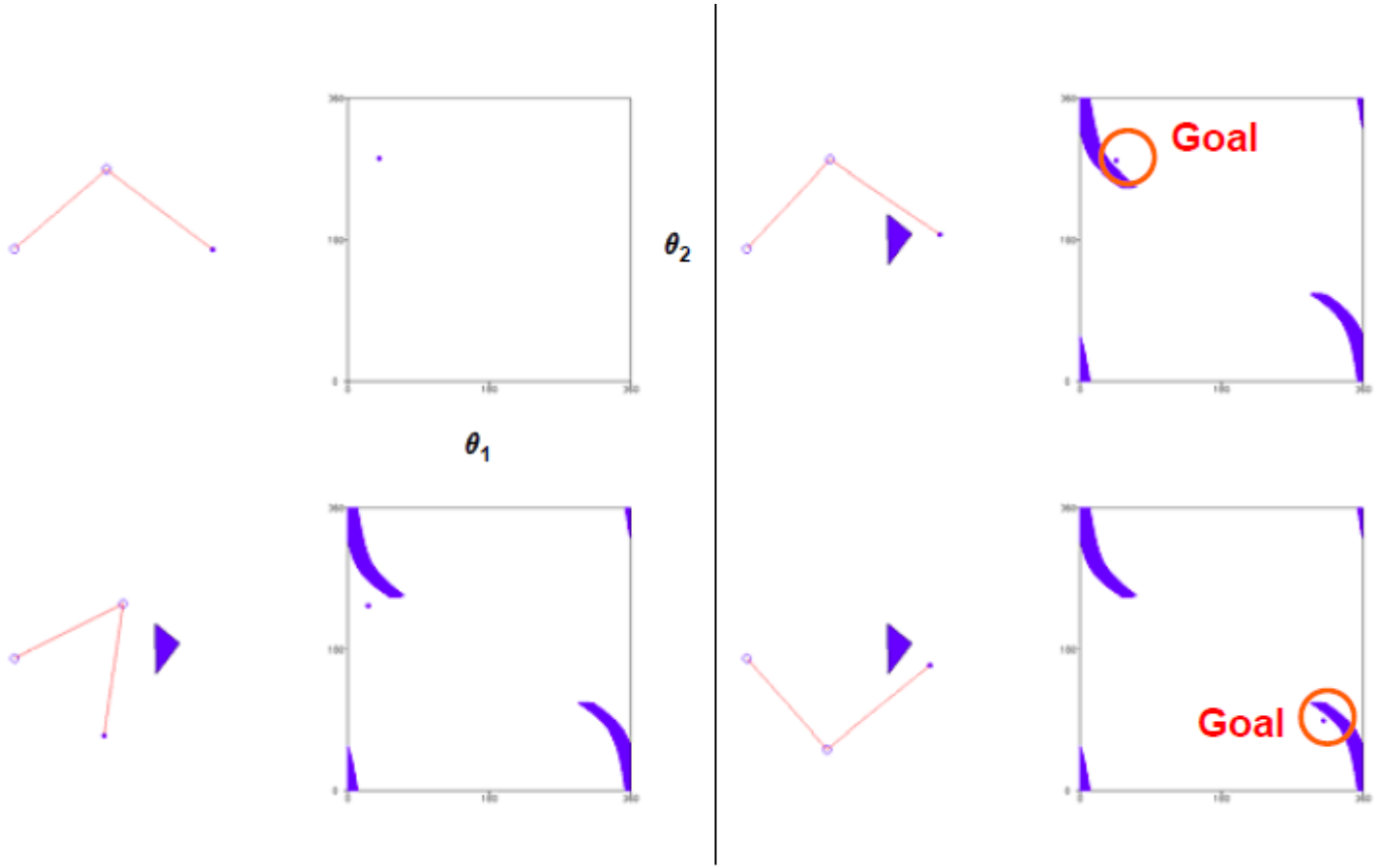
Configuration Space w/ Obstacle

How many IK solutions for goal?

Configuration Space: IK Solution



Configuration Space: IK Solution



IK Solutions are: Hard to get, discontinuous, often infinite

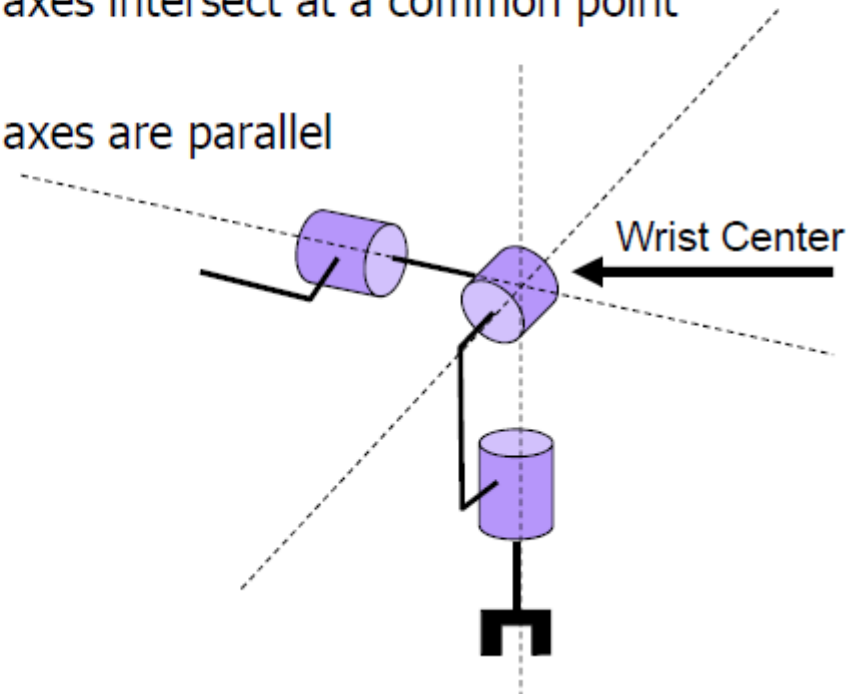
Analytical Methods

- Very fast and numerically stable
- No general solution!
  Each solution applies to a particular robot or class of robots
- Requires algebraic/geometric intuition
- Possible for robots with simple kinematics

Analytical Methods

Robots with simple kinematics: 6 DOF has closed-form IK if either:

- Three consecutive revolute axes intersect at a common point
- Three consecutive revolute axes are parallel



Analytical IK

- Difficult to find solution & it does not always exist
- Places constraints on robot construction
- The solution is very fast!

ALTERNATIVE:

- Differential Kinematics

Infinitesimal Changes

Forward Kinematics $\mathbf{x} = \mathbf{f}(\mathbf{q})$ $\mathbf{q} = [\theta_1, \theta_2, \ldots, \theta_n]^T$

$$\delta\mathbf{x} = \delta\mathbf{f}(\mathbf{q})$$

$$= [\delta f_1, \delta f_2, \ldots, \delta f_m]^T$$

$$\delta\mathbf{x} = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \delta\mathbf{q}$$

Differential (Forward) Kinematics

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}}$$

Differential Kinematics

What is the robot Jacobian?

$$\mathbf{J}(\theta_1, ..., \theta_n) = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} & \cdots & \frac{\delta x}{\delta \theta_n} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} & & \frac{\delta y}{\delta \theta_n} \\ \frac{\delta z}{\delta \theta_1} & \frac{\delta z}{\delta \theta_2} & & \frac{\delta z}{\delta \theta_n} \\ \frac{\delta \omega_x}{\delta \theta_1} & \frac{\delta \omega_x}{\delta \theta_2} & & \frac{\delta \omega_x}{\delta \theta_n} \\ \frac{\delta \omega_y}{\delta \theta_1} & \frac{\delta \omega_y}{\delta \theta_2} & & \frac{\delta \omega_y}{\delta \theta_n} \\ \frac{\delta \omega_z}{\delta \theta_1} & \frac{\delta \omega_z}{\delta \theta_2} & & \frac{\delta \omega_z}{\delta \theta_n} \end{bmatrix}$$

Matrix of Partial Derivatives of Kinematics w.r.t. each joint variable.

Why is it useful?

$$\frac{\delta x}{\delta t} = \frac{\delta x}{\delta \theta_1} \frac{\delta \theta_1}{\delta t}$$

Workspace velocity

Joint Velocity

Differential Kinematics

What is the robot Jacobian?

$$\mathbf{J}(\theta_1, ..., \theta_n) = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} & \cdots & \frac{\delta x}{\delta \theta_n} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} & & \frac{\delta y}{\delta \theta_n} \\ \frac{\delta z}{\delta \theta_1} & \frac{\delta z}{\delta \theta_2} & & \frac{\delta z}{\delta \theta_n} \\ \frac{\delta \omega_x}{\delta \theta_1} & \frac{\delta \omega_x}{\delta \theta_2} & & \frac{\delta \omega_x}{\delta \theta_n} \\ \frac{\delta \omega_y}{\delta \theta_1} & \frac{\delta \omega_y}{\delta \theta_2} & & \frac{\delta \omega_y}{\delta \theta_n} \\ \frac{\delta \omega_z}{\delta \theta_1} & \frac{\delta \omega_z}{\delta \theta_2} & & \frac{\delta \omega_z}{\delta \theta_n} \end{bmatrix}$$

Matrix of Partial Derivatives of Kinematics w.r.t. each joint variable.

Why is it useful?

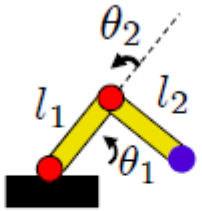
$$\frac{\delta x}{\delta t} = \frac{\delta x}{\delta \theta_1}\frac{\delta \theta_1}{\delta t} \longrightarrow \dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & ... & \dot{\theta}_n \end{bmatrix}^T$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\omega}_x & \dot{\omega}_y & \dot{\omega}_z \end{bmatrix}^T$$

How do we compute J? – Method1

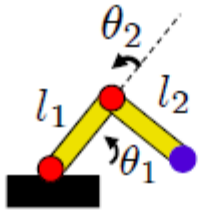
$$\mathbf{T}_3^0 = \begin{bmatrix} c_{12} & -s_{12} & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{J} = \begin{bmatrix} \dfrac{\delta x}{\delta \theta_1} \\ -l s_1 - l_2 s_{12} \\ \end{bmatrix}$$

How do we compute J? – Method1

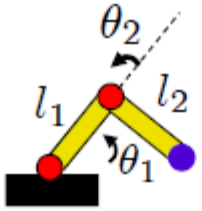
$$\mathbf{T}_3^0 = \begin{bmatrix} c_{12} & -s_{12} & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{J} = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ -l s_1 - l_2 s_{12} & -l_2 s_{12} \\ & \end{bmatrix}$$

How do we compute J? – Method1

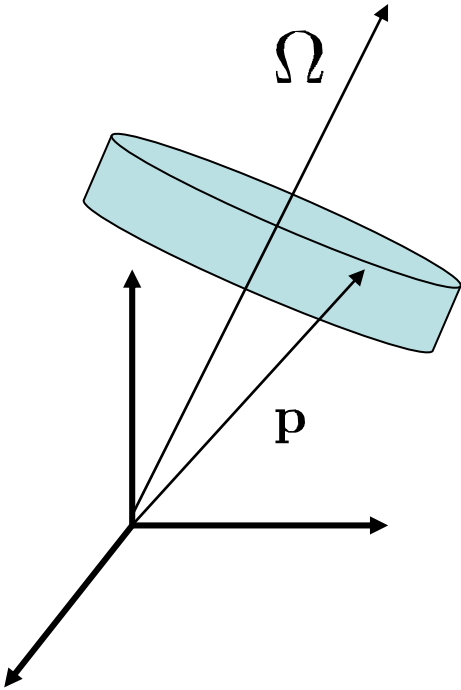
$$\mathbf{T}_3^0 = \begin{bmatrix} c_{12} & -s_{12} & l_1c_1 + l_2c_{12} \\ s_{12} & c_{12} & l_1s_1 + l_2s_{12} \\ 0 & 0 & 1 \end{bmatrix}$$



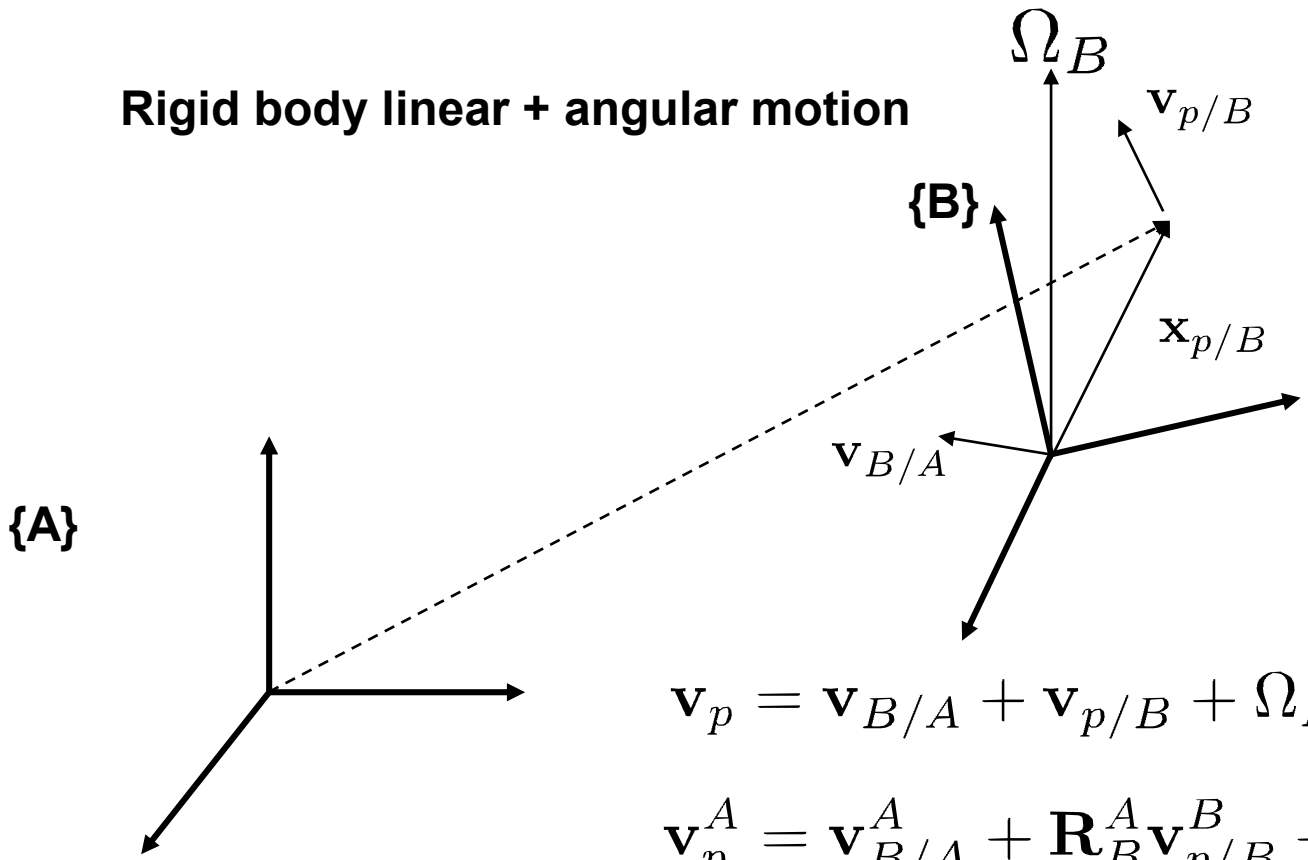
$$\mathbf{J} = \begin{bmatrix} \frac{\delta x}{\delta \theta_1} & \frac{\delta x}{\delta \theta_2} \\ \frac{\delta y}{\delta \theta_1} & \frac{\delta y}{\delta \theta_2} \end{bmatrix} = \begin{bmatrix} -ls_1 - l_2s_{12} & -l_2s_{12} \\ l_1c_1 + l_2c_{12} & l_2c_{12} \end{bmatrix}$$

How do we compute J? – Method2

Rigid body angular motion $\mathbf{v}_p = \Omega \times \mathbf{p}$



Rigid body linear + angular motion



$$\mathbf{v}_p = \mathbf{v}_{B/A} + \mathbf{v}_{p/B} + \Omega_B \times \mathbf{x}_{p/B}$$

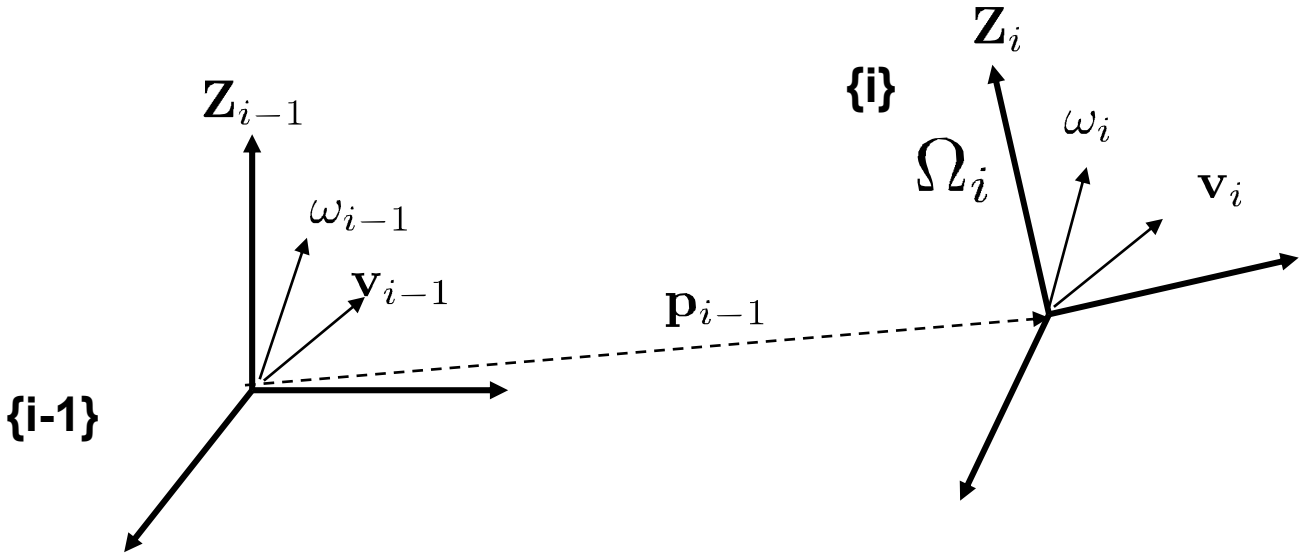
$$\mathbf{v}_p^A = \mathbf{v}_{B/A}^A + \mathbf{R}_B^A \mathbf{v}_{p/B}^B + \Omega_B^A \times \mathbf{R}_B^A \mathbf{x}_{p/B}$$

How do we compute J? – Method2

Consider the propagation of velocities: linear velocity & angular velocity

d_i = the distance from X_{i-1} to X_i measured along Z_i

θ_i = the angle between X_{i-1} and X_i measured about Z_i



Linear velocity propagation $\mathbf{v}_i = \mathbf{v}_{i-1} + \omega_{i-1} \times \mathbf{p}_{i-1} + \dot{d}_i \mathbf{Z}_i$

Angular velocity propagation $\omega_i = \omega_{i-1} + \Omega_i$

$$\Omega_i = \dot{\theta}_i \mathbf{Z}_i$$

How do we compute J? – Method2

Linear velocity propagation $\mathbf{v}_{i+1} = \mathbf{v}_i + \omega_i \times \mathbf{p}_i + \underbrace{\dot{q}_{i+1}\mathbf{Z}_{i+1}}$

Prismatic Joint V_i

$$\mathbf{v} = \sum (\epsilon_i V_i + (1-\epsilon_i)(\Omega_i \times \mathbf{p}_i))$$
$$= \sum (\epsilon_i \dot{q}_i \mathbf{Z}_i + (1-\epsilon_i)(\dot{q}_i \mathbf{Z}_i \times \mathbf{p}_i))$$

$$\epsilon_i = \begin{cases} 1 & \text{for prismatic joint} \\ 0 & \text{for revolute joint} \end{cases}$$

Angular velocity propagation $\omega_{i+1} = \omega_i + \Omega_{i+1}$ $\underbrace{\Omega_i = \dot{q}_i \mathbf{Z}_i}$

Revolute Joint

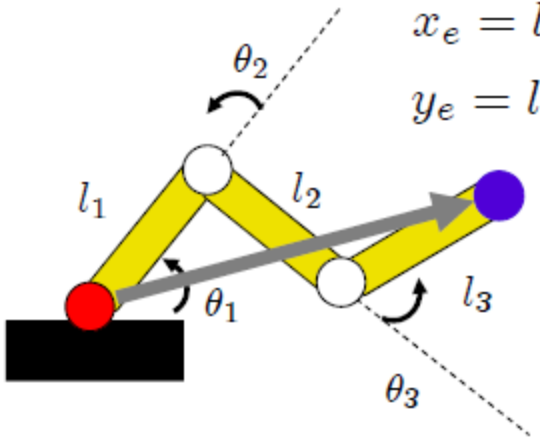
$$\omega = \sum (1-\epsilon_i)\Omega_i$$
$$= \sum (1-\epsilon_i)\dot{q}_i \mathbf{Z}_i$$

$$\epsilon_i = \begin{cases} 1 & \text{for prismatic joint} \\ 0 & \text{for revolute joint} \end{cases}$$

How do we compute J? – Method2

$$\mathbf{T}_i^0 = \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \mathcal{J}_{P1} & \cdots & \mathcal{J}_{Pn} \\ \mathcal{J}_{O1} & \cdots & \mathcal{J}_{On} \end{bmatrix}$$



$$x_e = l_1 c_1 + l_2 c_{12} + l_3 c_{123}$$
$$y_e = l_1 s_1 + l_2 s_{12} + l_3 s_{123}$$

$$\epsilon_i = \begin{cases} 1 & \text{for prismatic joint} \\ 0 & \text{for revolute joint} \end{cases}$$

$$\begin{bmatrix} \mathcal{J}_{Pi} \\ \mathcal{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_i \\ 0 \end{bmatrix} & \text{(prismatic joint)} \\ \begin{bmatrix} \mathbf{z}_i \times \mathbf{p}_i \\ \mathbf{z}_i \end{bmatrix} & \text{(revolute joint)} \end{cases}$$

$$\mathbf{v} = \sum (\epsilon_i \dot{q}_i \mathbf{Z}_i + (1 - \epsilon_i)(\dot{q}_i \mathbf{Z}_i \times \mathbf{p}_i))$$
$$\omega = \sum (1 - \epsilon_i) \dot{q}_i \mathbf{Z}_i$$

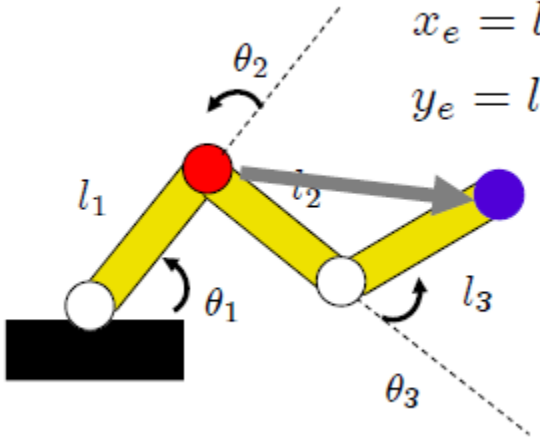
$$\mathbf{J} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdots$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_1 c_1 + l_2 c_{12} + l_3 c_{123} - 0 \\ l_1 s_1 + l_2 s_{12} + l_3 s_{123} - 0 \\ 0 \end{bmatrix}$$

How do we compute J? – Method2

$$\mathbf{T}_i^0 = \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \mathcal{J}_{P1} & \cdots & \mathcal{J}_{Pn} \\ \mathcal{J}_{O1} & \cdots & \mathcal{J}_{On} \end{bmatrix}$$



$$x_e = l_1 c_1 + l_2 c_{12} + l_3 c_{123}$$
$$y_e = l_1 s_1 + l_2 s_{12} + l_3 s_{123}$$

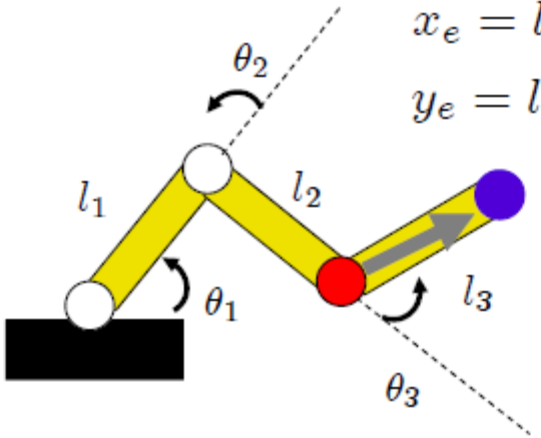
$$\begin{bmatrix} \mathcal{J}_{Pi} \\ \mathcal{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_i \\ 0 \end{bmatrix} & \text{(prismatic joint)} \\ \begin{bmatrix} \mathbf{z}_i \times \mathbf{p}_i \\ \mathbf{z}_i \end{bmatrix} & \text{(revolute joint)} \end{cases}$$

$$\mathbf{J} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} & \cdots \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_2 c_{12} + l_3 c_{123} & \\ 0 & 0 & \\ 0 & 0 & \\ 0 & 0 & \\ 1 & 1 & \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_1 c_1 + l_2 c_{12} + l_3 c_{123} - l_1 c_1 \\ l_1 s_1 + l_2 s_{12} + l_3 s_{123} - l_1 s_1 \\ 0 \end{bmatrix}$$

How do we compute J? – Method2

$$\mathbf{T}_i^0 = \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{z}_i & \mathbf{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \mathcal{J}_{P1} & \cdots & \mathcal{J}_{Pn} \\ \mathcal{J}_{O1} & \cdots & \mathcal{J}_{On} \end{bmatrix}$$



$$x_e = l_1 c_1 + l_2 c_{12} + l_3 c_{123}$$
$$y_e = l_1 s_1 + l_2 s_{12} + l_3 s_{123}$$

$$\begin{bmatrix} \mathcal{J}_{Pi} \\ \mathcal{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_i \\ 0 \end{bmatrix} & \text{(prismatic joint)} \\ \begin{bmatrix} \mathbf{z}_i \times \mathbf{p}_i \\ \mathbf{z}_i \end{bmatrix} & \text{(revolute joint)} \end{cases} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} l_1 c_1 + l_2 c_{12} + l_3 c_{123} - (l_1 c_1 + l_2 c_{12}) \\ l_1 s_1 + l_2 s_{12} + l_3 s_{123} - (l_1 s_1 + l_2 s_{12}) \\ 0 \end{bmatrix}$$

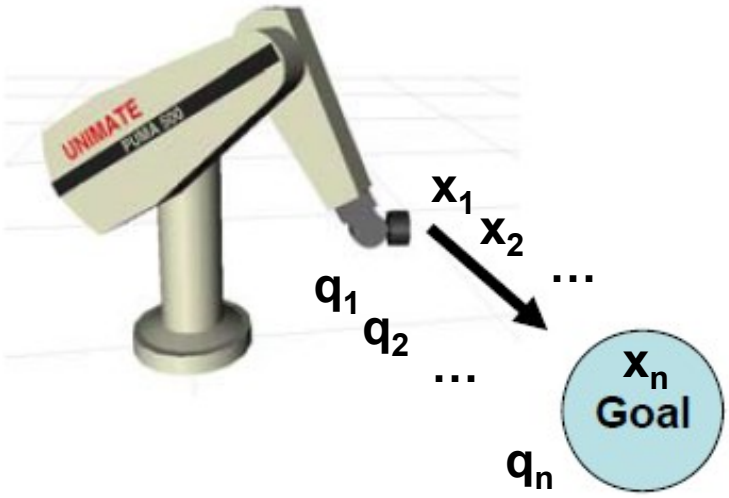
$$\mathbf{J} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} & -l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_2 c_{12} + l_3 c_{123} & l_3 c_{123} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

How do we compute J? – Method3

1. Simulate a small displacement $\Delta\theta_i$ for each joint
2. Observe Δx and numerically compute J
   - Less accurate, more time consuming
   - Sometimes the only option

Gradient IK - How do we use J?

- Workspace goal
- How do we get a joint space goal?



- Assuming 6 D.O.F and **J** is full rank: $\Delta q = J^{-1}\Delta x$
  Iterate until convergence $\mathbf{x}_i = \mathbf{f}(\mathbf{q}_i)$
- Otherwise Still Possible (Pseudo-Inverse & Variants): $J^{+} = JT\ (JJ^{T})^{-1}$

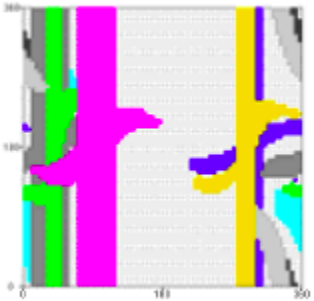
Gradient IK

- Generally applicable to n-DOF kinematics
- Many existing implementations
- Slower and unstable around singularities

Planning

Goals for manipulators are really that complicated!

Add what it takes to represent obstacles:



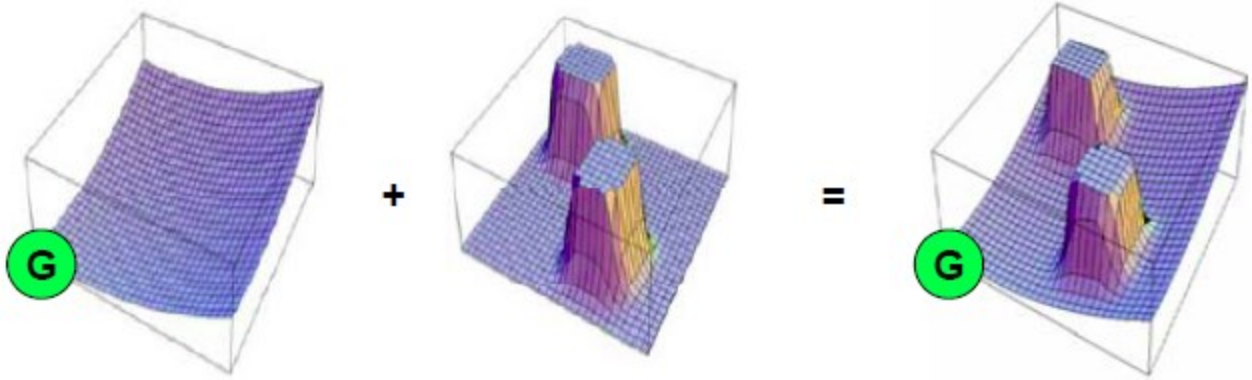
But, does this remind you of anything?

$\Delta q = J^{-1}\Delta x$

Planning – Manipulator + Potential Field

Oussama Khatib

'86



$U_a(q)$ + $U_r(q)$ = $U(q)$

Planning – Manipulator + Potential Field

- First effective planning method for high-DOF robot arms
- Very fast (potential fields)

However:

- Local Minima
- Complex obstacle representation
- For Soundness must define obstacles with respect to each link

Planning – Manipulator + Potential Field

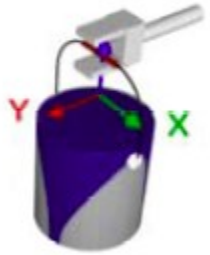
Combine Kinematics & Geometric Planning

- Roadmap Methods:
  - Visibility Graphs
    → No polygons in C-space.
  - Voronoi Diagrams
    → Canny's Thesis: NP-hard Algorithms for Motion Planning

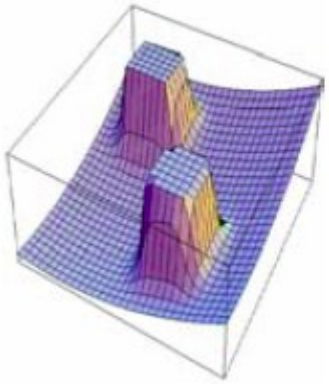
- Cell Decomposition:
  - Exact
    → Same troubles as Visibility Graphs.
  - Approximate
    → Ok for 2-3 dimensions
    → Good luck in 4+ dimensions

Kinematic Planning Algorithms

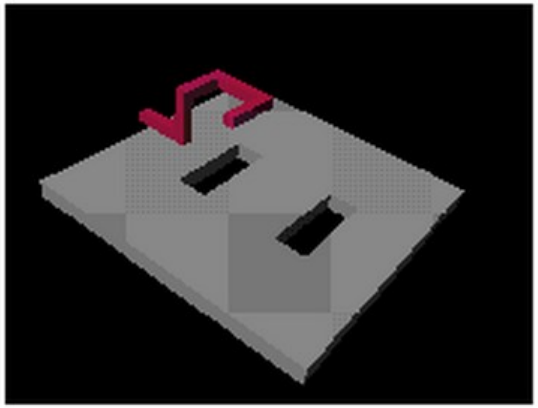
- Pre-1980
  - Mainly 2D, 3D planners
  - In case of kinematics:
    - →Ignore robot geometry
    - →Robot is an ethereal entity that obeys kinematics

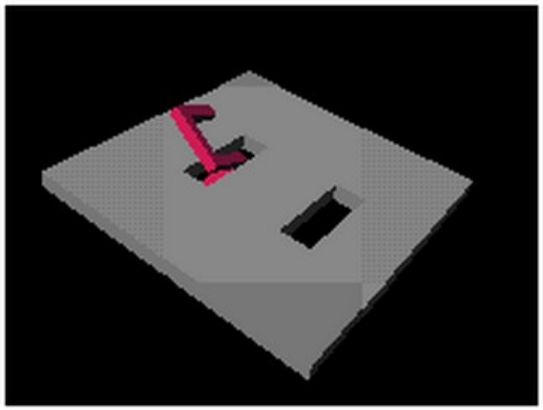


- 1980s
  - Potential Fields!
    - → First feasible SOUND solution
    - → Locally optimal control strategy
    - → Not globally complete or optimal

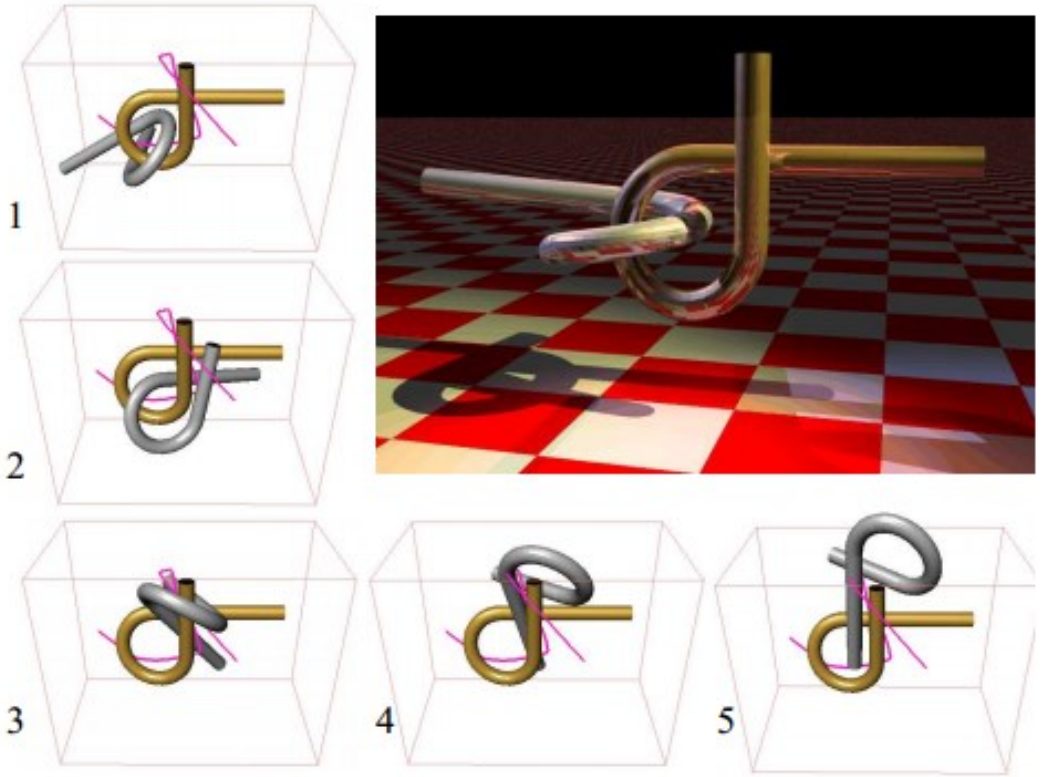


Can we solve these planning problems?





http://www.kavrakilab.org/robotics/prm.html



"Planning Algorithms", S. Lavalle

Key Idea

- What did Visibility, Voronoi, Cells, Fields have in common?
  - Some form of explicit environment representation
  - Attempt at some form of optimality

- New concepts from 1990s:
  - Forget optimality altogether
  - Focus on Completeness
  - Think about Free Space

A New Kind of Roadmap

- Lydia Kavraki '94, '96 – Present
- Mark Overmars '92, '96 - Present

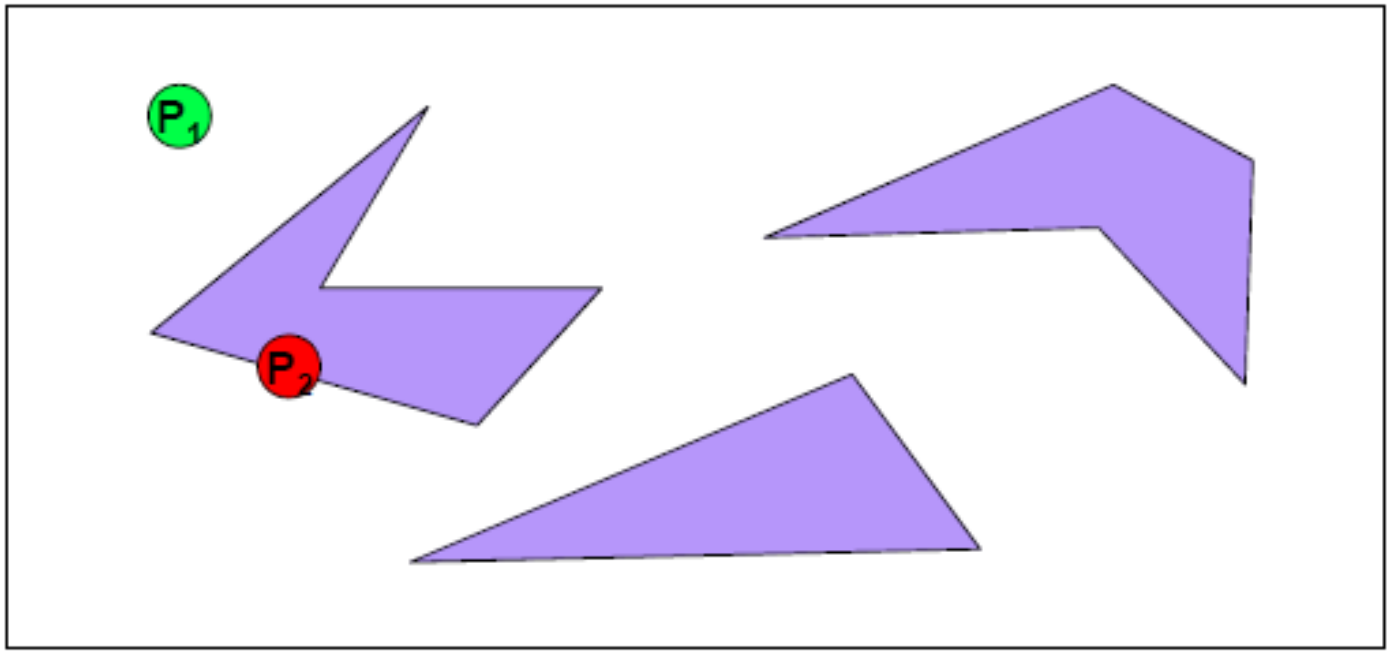
- Previous roadmaps used features related to actual obstacle features.

- Probabilistic Roadmaps (PRM)
  - Features: Sampled free points
  - Edges: Verified connections

"Probabilistic roadmaps for path planning in high-dimensional configuration spaces" By Kavraki, Svestka, Latombe, and Overmars, 1996, IEEE Transactions on Robotics and Automation

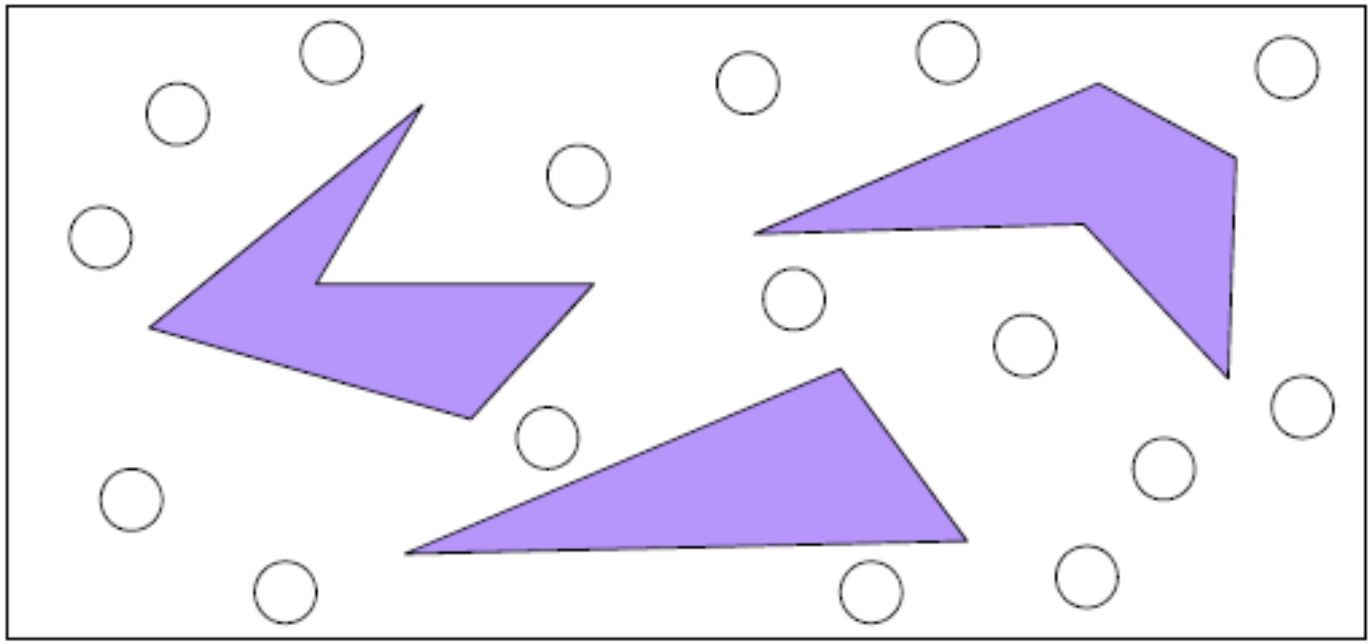
Probabilistic Roadmap: Step 1

Randomly sample a configuration: P Keep only if P is in Free Space



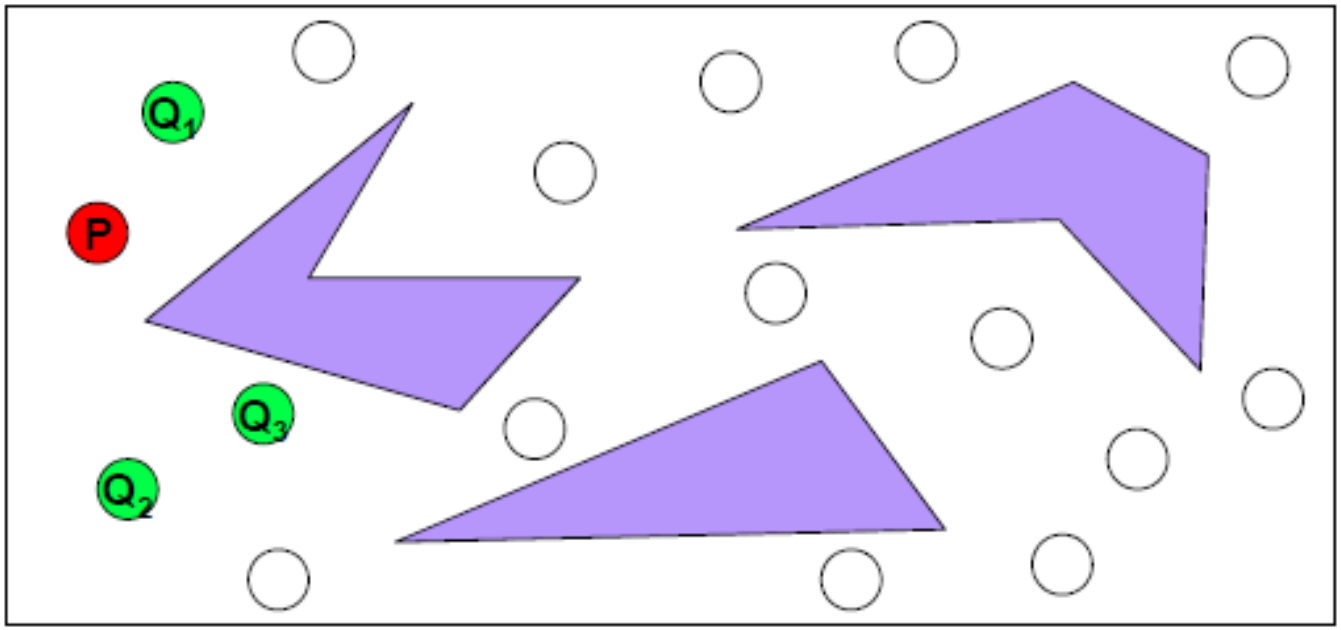
Probabilistic Roadmap: Step 1

Randomly sample a configuration: P Keep only if P is in Free Space



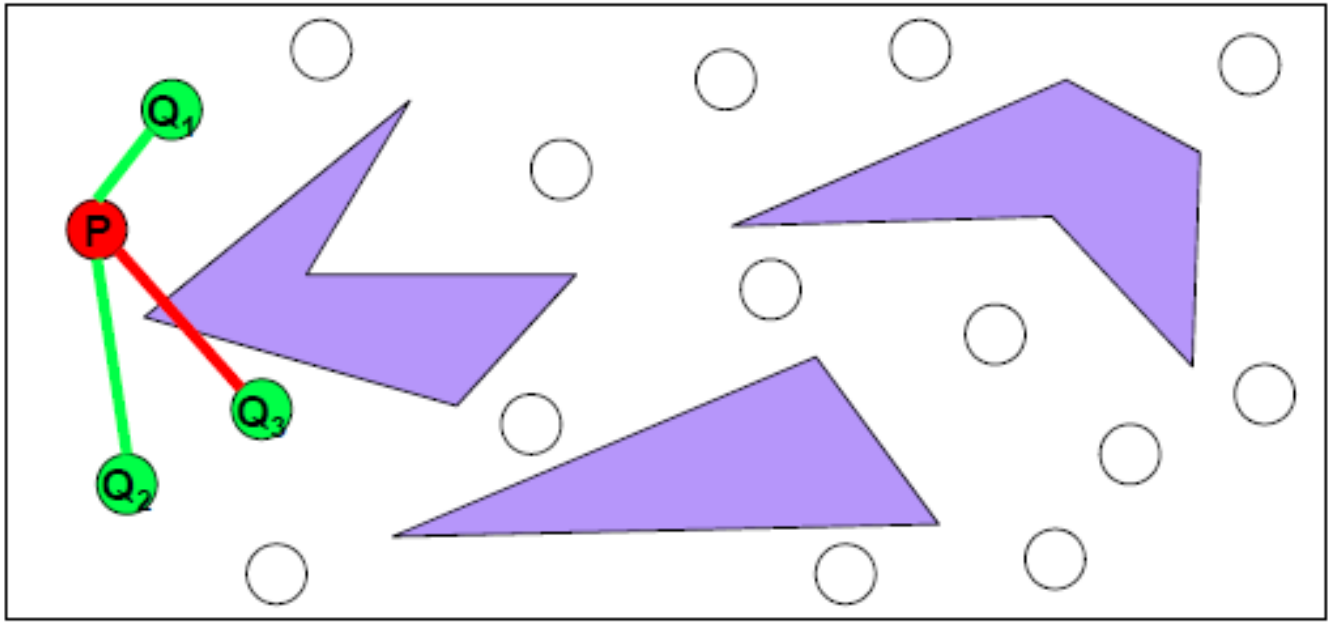
Probabilistic Roadmap: Step 2

For each node P find k nearest neighbors: Q_1 … Q_k



Probabilistic Roadmap: Step 3

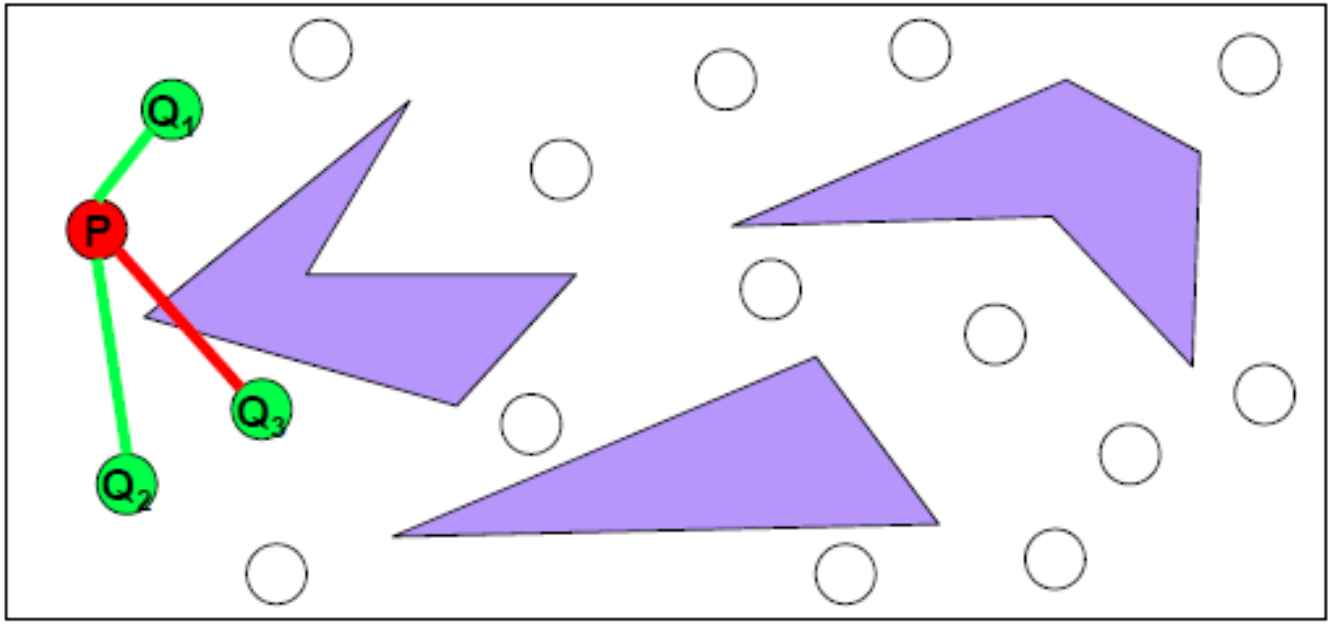
For each node P find k nearest neighbors: Q_1 … Q_k



Use a local planner to test connectivity between P and Q_i

Probabilistic Roadmap: Step 3

For each node P find k nearest neighbors: Q_1 … Q_k

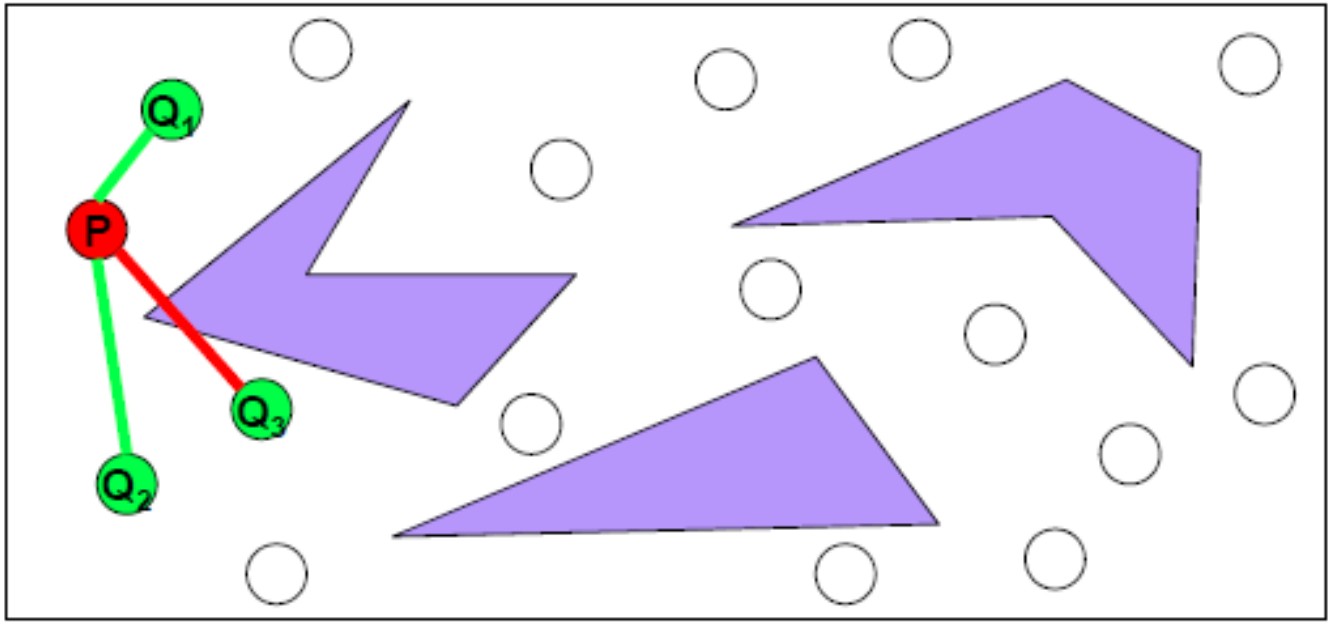


Use a local planner to test connectivity between P and Q_i

What could be a local planner? What happens next?

Probabilistic Roadmap: Step 4

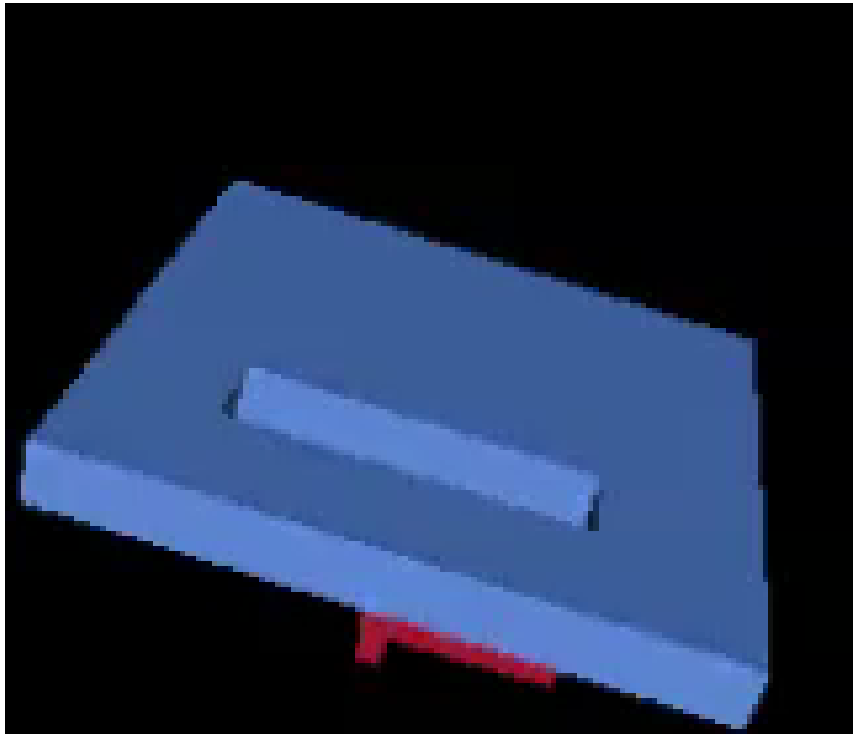
For each node P find k nearest neighbors: Q_1 … Q_k



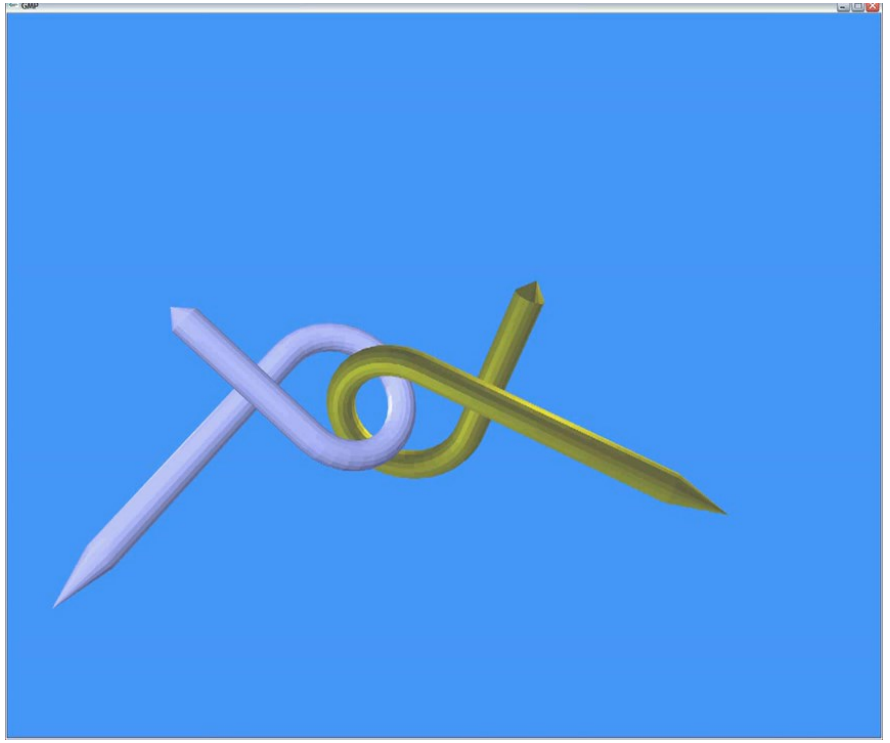
Use a local planner to test connectivity between P and Q_i

Find a path: Uniform Cost, A*, …

We can solve these planning problem



http://www.kavrakilab.org/robotics/prm.html



http://www.youtube.com/watch?v=FRGzsyXHBqQ