CS 4649/7649
Robot Intelligence: Planning

Potential Field, Kinematics

Sungmoon Joo

School of Interactive Computing
College of Computing
Georgia Institute of Technology

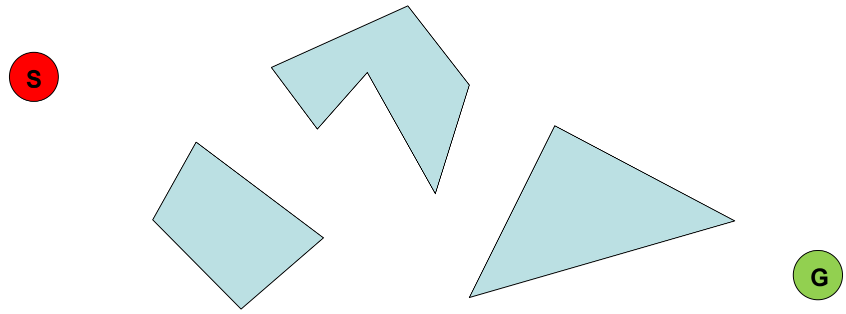
*Slides based in part on Dr. Mike Stilman's lecture slides

Course Info.

- HW#1 due Oct 6th
  - Wiki - Add your group info.
  - Need a repo.?
  - Late policy – No late HWs

Navigation Planning

Assuming full knowledge, how does the robot 'plan' its path from S to G?

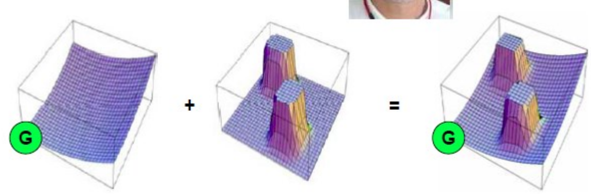


Final Navigation Planner: Potential Fields

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

Oussama Khatib

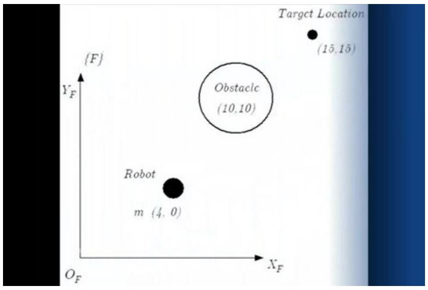
'86

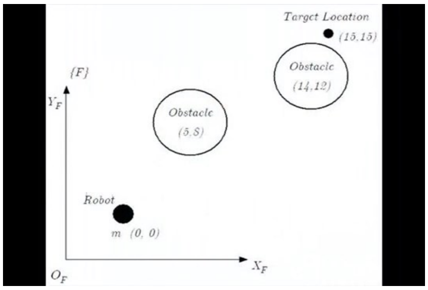


$U_a(q) + U_r(q) = U(q)$

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

Potential Fields

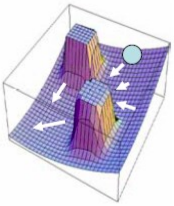


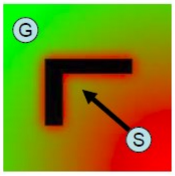


*Simulation by Leng-Feng Lee@Buffalo Univ.

Potential Fields

- Gradient Descent: $\dot{q} = -\nabla U(q)$
- Complete? No (local minima)



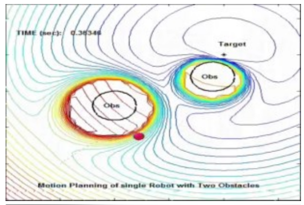


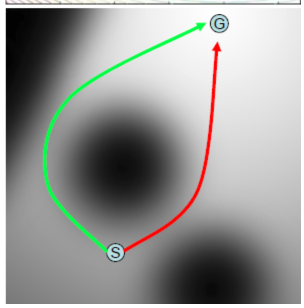
Potential Fields

- Gradient Descent: $\dot{q} = -\nabla U(q)$

- Complete? No (local minima)

- Optimal?
  - Metric = Summed Potential Function
  - Trades off travel & obstacle distance
  - Which way would the robot travel?





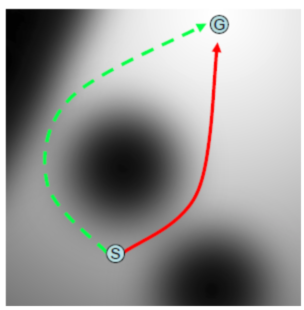
Potential Fields

- Gradient Descent: $\dot{q} = -\nabla U(q)$

- Complete? No (local minima)

- Optimal? Locally Yes, **Globally No**
  - Metric = Summed Potential Function
  - Trades off travel & obstacle distance
  - Takes the long way
  - **No concept of future obstacles**

Advantage: Efficiency!



Alternate Use of Potential Fields

Use as a heuristic in some other best-first or A* search

- Pros:
  - Gains Completeness
  - Efficient Heuristic to compute

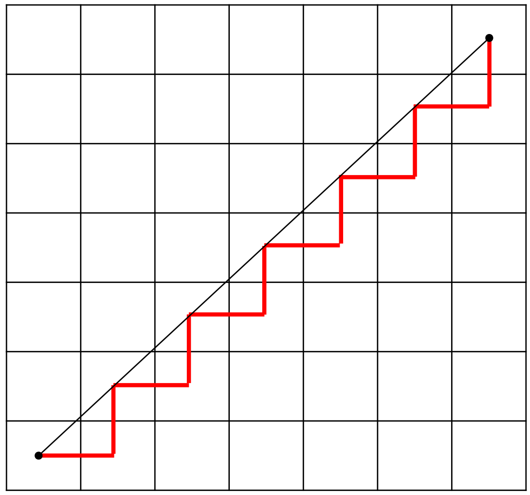
- Cons:
  - Requires additional data structure (any previous algorithm)
  - Overall not nearly as efficient

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

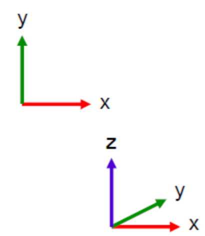
Manhathan Distance

Manhathan Distance – Distance measure in grid world



Coordinates

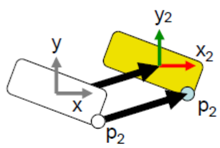
- **Coordinate System:** A set of numbers that specifies configuration
  - Points
  - Rigid Bodies
  - Articulated Manipulators
- **Degrees of Freedom (DOF):** Minimal number of independent coordinates
- How many DOF?
  - Point in the plane 2 (x,y)
  - Point in 3D 3 (x,y,z)
  - Body in 2D 3 (x,y, θ)



Rigid Body Displacements

Must preserve **rigid** property, reflections are not allowed.

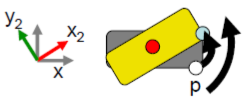
- **Translation:** Every point moves a fixed distance in a specified direction.



$$x_2 = x_1 + d_x$$
$$y_2 = y_1 + d_y$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- **Rotation**: One point is fixed. Others move a specified **angle** relative to fixed point.



$$x_2 = x_1 \cos\theta - y_1 \sin\theta$$
$$y_2 = x_1 \sin\theta + y_1 \cos\theta$$

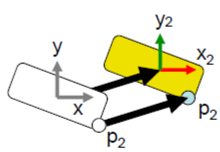
$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathrm{c}\theta & -\mathrm{s}\theta \\ \mathrm{s}\theta & \mathrm{c}\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\mathrm{c}\theta = \cos(\theta)$$
$$\mathrm{s}\theta = \sin(\theta)$$

Rigid Body Displacements

Must preserve **rigid** property, reflections are not allowed.

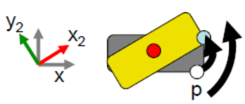
- **Translation:** Every point moves a fixed distance in a specified direction.



$$x_2 = x_1 + d_x$$
$$y_2 = y_1 + d_y$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- **Rotation**: One point is fixed. Others move a specified **angle** relative to fixed point.



$$x_2 = x_1 \cos\theta - y_1 \sin\theta$$
$$y_2 = x_1 \sin\theta + y_1 \cos\theta$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathrm{c}\theta & -\mathrm{s}\theta \\ \mathrm{s}\theta & \mathrm{c}\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

- Every displacement can be represented as **1 Translation** and/or **1 Rotation**

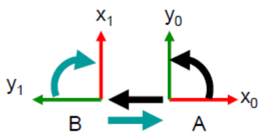
Homogeneous Transformations: 2D

Method for representing displacements and relative coordinates

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

Compact Representation. Allows for simple concatenation:

$$\mathbf{T}_C^A = \mathbf{T}_B^A \mathbf{T}_C^B$$

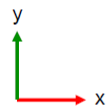


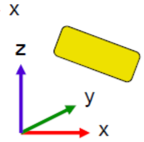
$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A to B   B to A   =   Identity

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)





Representations of Rotation (Coordinates)

- Fixed Axis (x,y,z) or (roll, pitch, yaw)
  - Convenient and intuitive, 12 variations
- Euler Angles (z,y,x), (z,y,z) - (moving axes)
  - Equivalent to reverse order fixed-axis
- Unit Quaternions (4 numbers)
  - Easy to compose
  - Meaningful Interpolation
  - Useful for numerical stability, sampling, optimization
- Angle-Axis (4 numbers = 3 axis + 1 angle)
- **Rotation Matrices (9 numbers – Orthonormal Matrix)**

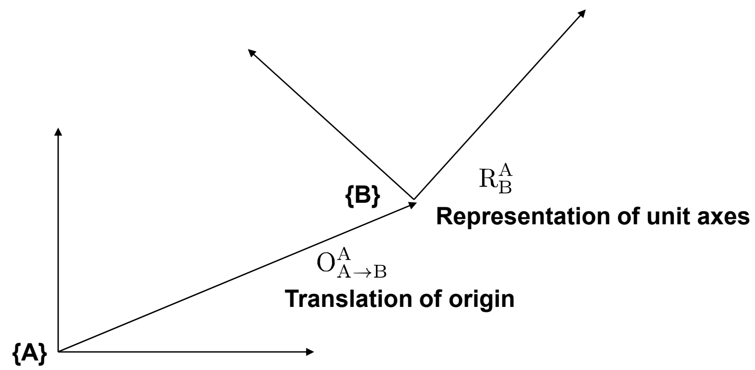
Fixed Axis to Rotation Matrices

$$\mathbf{R}^A_{B\ XYZ}(\gamma,\beta,\alpha) = \mathbf{R}_Z(\alpha)\mathbf{R}_Y(\beta)\mathbf{R}_X(\gamma) =$$

$$\begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix} =$$

$$\begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}$$

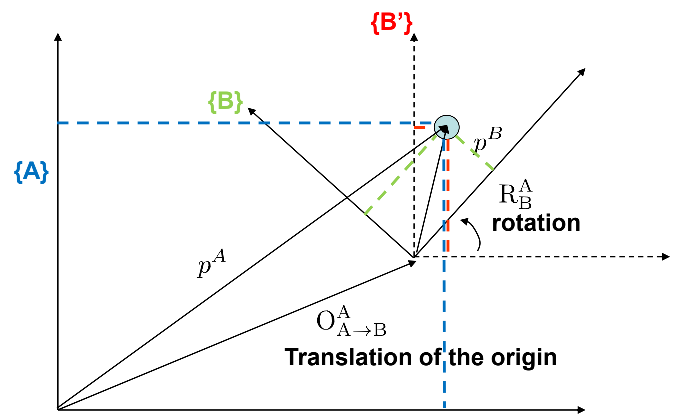
Homogeneous Transform 3D



Description of frame B relative to frame A

$$\{B\} = \{R_B^A, O_{A\to B}^A\}$$

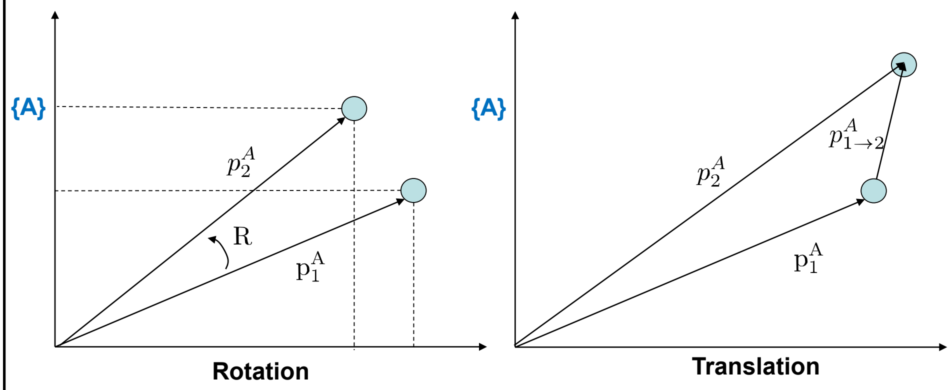
Homogeneous Transform 3D



Mapping: Change descriptions of (the same)point p from frame B to frame A

$$p^{B'} = R_B^A p^B \qquad p^A = p^{B'} + O_{A\to B}^A \qquad \begin{bmatrix} p^A \\ 1 \end{bmatrix} = \begin{bmatrix} R_B^A & O_{A\to B}^A \\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^B \\ 1 \end{bmatrix}$$

Homogeneous Transform 3D



Operator: Moving Points – Rotational operator & Translational operator

$$p_2^A = \mathrm{R} p_1^A \qquad p_2^A = p_1^A + p_{1\to 2}^A \qquad \begin{bmatrix} p_2^A \\ 1 \end{bmatrix} = \begin{bmatrix} R & p_{1\to 2}^A \\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_1^A \\ 1 \end{bmatrix}$$

Homogeneous Transform 3D

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

Interpretations:

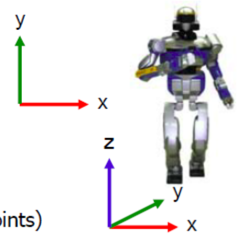
- Maps p^B to p^A $\quad p^A = \mathrm{T}_{\mathrm{B}}^{\mathrm{A}} p^B$
- Transform operator: creates p_2^A from p_1^A
- Describes frame B relative to frame A
  t_B^A = position of the frame

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

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)

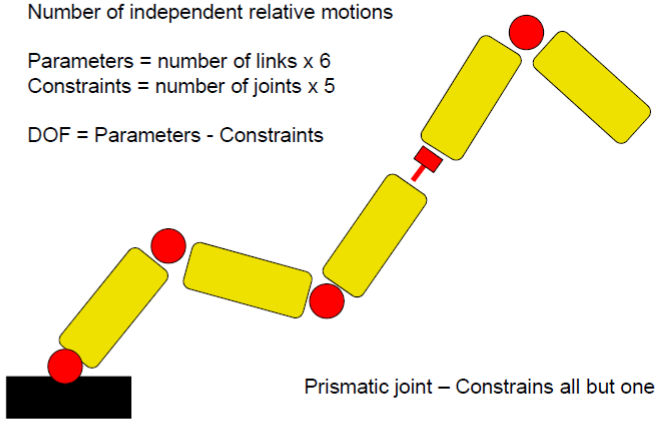


Robot Degrees of Freedom

Number of independent relative motions

Parameters = number of links x 6
Constraints = number of joints x 5

DOF = Parameters - Constraints



Prismatic joint – Constrains all but one translation

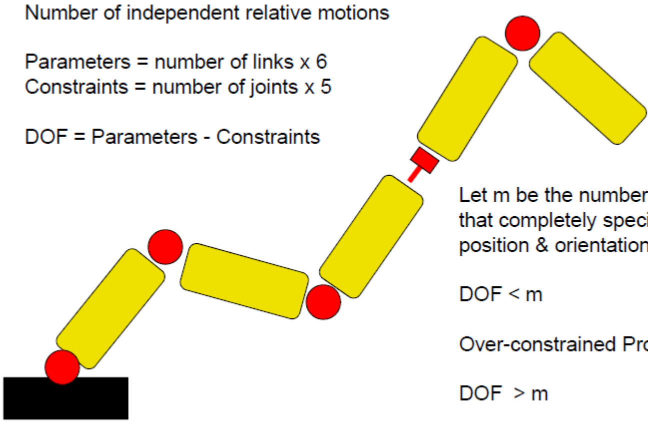
Revolute joint – Constrains all but one rotation

Robot Degrees of Freedom

Number of independent relative motions

Parameters = number of links x 6
Constraints = number of joints x 5

DOF = Parameters - Constraints



Let m be the number of parameters that completely specify end-effector position & orientation (typically 6)

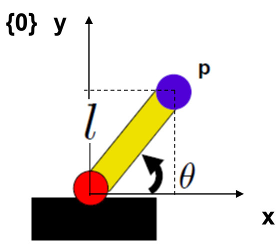
DOF < m

Over-constrained Problem

DOF > m

Redundant Degrees of Freedom

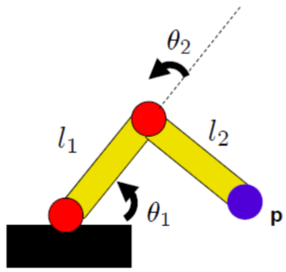
Forward Kinematics



1-DOF Robot Arm

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

Forward Kinematics



2-DOF Robot Arm

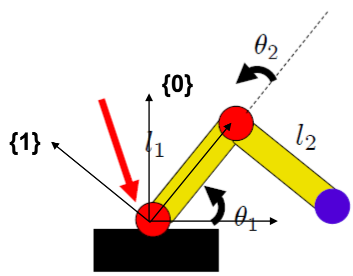
$$p_x = l_1 c_1 + l_2 c_{12}$$
$$p_y = l_1 s_1 + l_2 s_{12}$$

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

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

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

Forward Kinematics

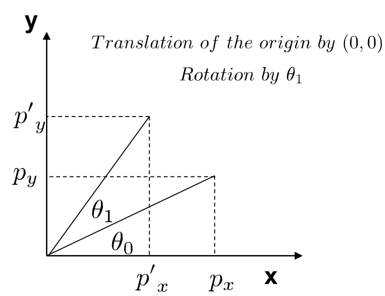


$$p_x = l_1 c_1 + l_2 c_{12}$$
$$p_y = l_1 s_1 + l_2 s_{12}$$

2-DOF Robot Arm

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

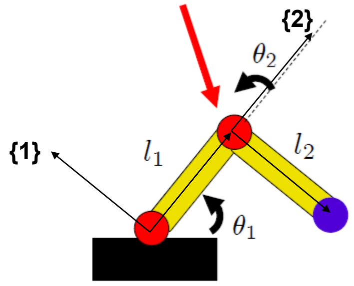
Representation of frame1 in frame0*



$$p'_x = c_1 p_x - s_1 p_y$$
$$p'_y = s_1 p_x + c_1 p_y$$

***Alternative interpretation – Moving operation**

Forward Kinematics



2-DOF Robot Arm

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

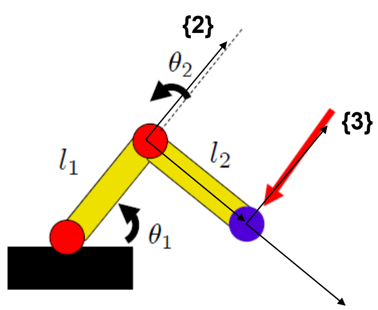
$$\mathbf{T}_2^1 = \begin{bmatrix} c_2 & -s_2 & l_1 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Representation of frame2 in frame1

Translation of the origin by $(l_1, 0)$

Rotation by θ_2

Forward Kinematics



2-DOF Robot Arm

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

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

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

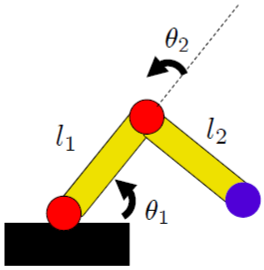
Representation of frame3 in frame2

Translation of the origin by $(l_2, 0)$

Rotation by 0

Forward Kinematics

2-DOF Robot Arm



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

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

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

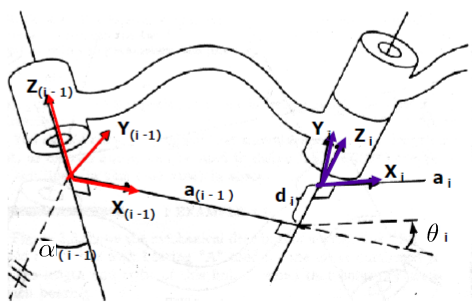
$$\mathbf{T}_2^0 = \mathbf{T}_1^0\mathbf{T}_2^1 = \begin{bmatrix} c_1c_2 - s_1s_2 & -c_1s_2 - s_1c_2 & l_1c_1 \\ s_1c_2 + c_1s_2 & -s_1s_2 + c_1c_2 & l_1s_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{12} & -s_{12} & l_1c_1 \\ s_{12} & c_{12} & l_1s_1 \\ 0 & 0 & 1 \end{bmatrix}$$

Representation of frame2 in frame0

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

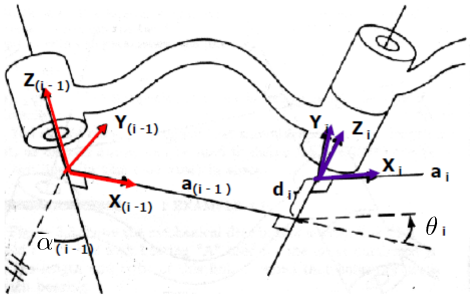
Representation of frame3 in frame0

Denavit-Hartenberg (DH) Parameters



First, label the coordinate frames

DH Parameters



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

DH Parameters

- Parameter Table:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	0

- DH Matrix:

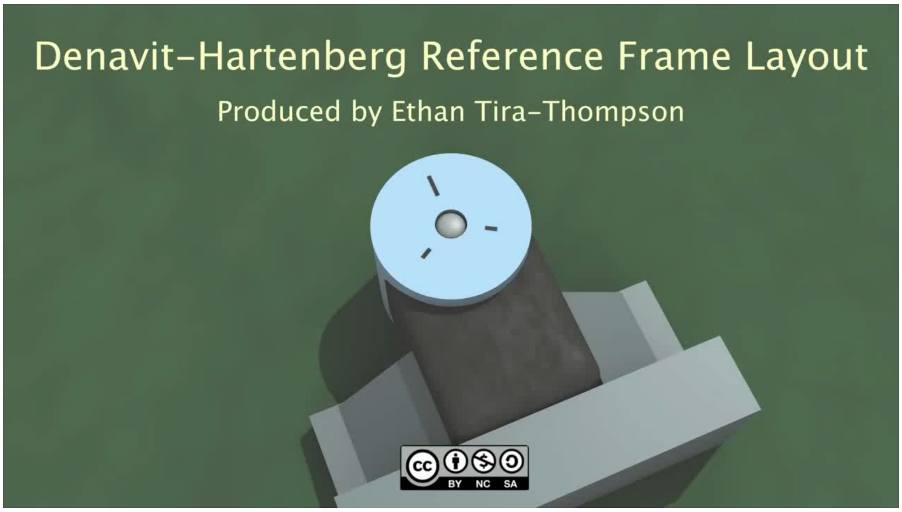
$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & \alpha_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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**

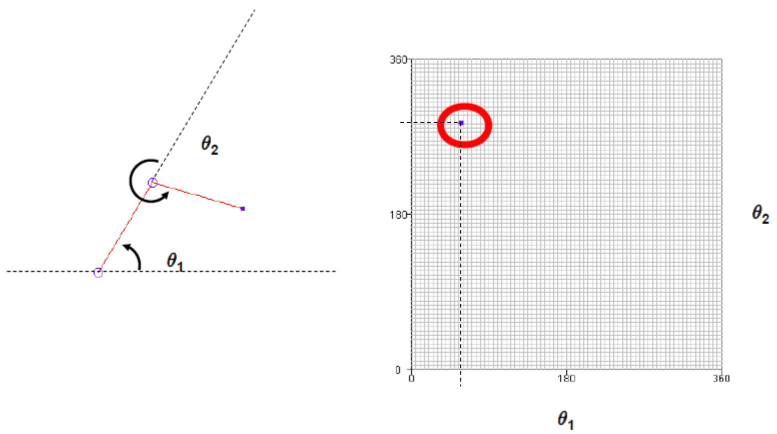
DH Parameters



http://www.youtube.com/watch?v=rA9tm0gTln8

***Beware of the fact that the notation is slightly different!**

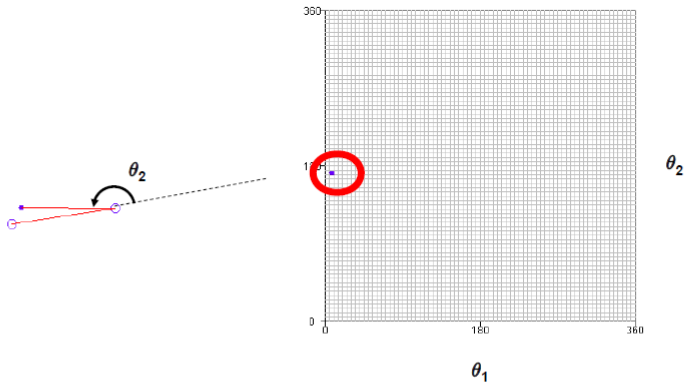
Configurations Space



Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

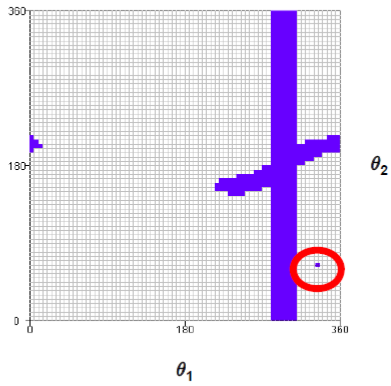
https://ford.ieor.Berkeley/cspace

Configurations Space



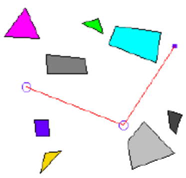
Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

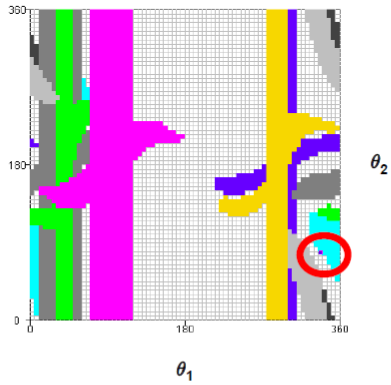
Configurations Space



Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

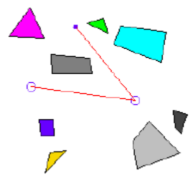
Configurations Space

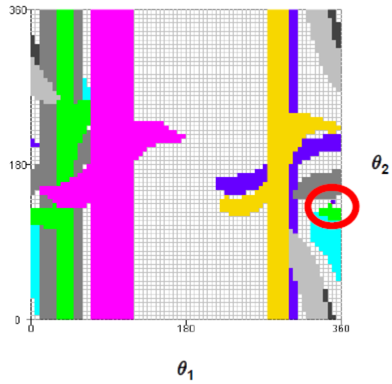




Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

Configurations Space





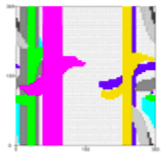
Java Applet: Jeff Wiegley, Eric Lee, Ken Goldberg

Interim Summary

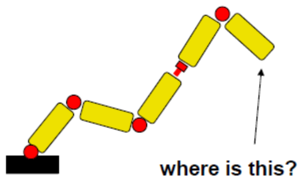
- 6 DOF Necessary for 3D motion
- N-DOF for Robot Motion
- Joint Space is a Configuration Space
- Workspace does not map nicely to Configuration Space!

Kinematics

- Robot Configuration Space = Robot Joint Space

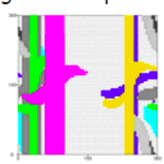


- Robot Kinematics = Mapping **from** joint angles **to** link positions

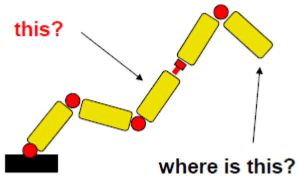


Kinematics: Standard Representation

- Robot Configuration Space = Robot Joint Space



- Robot Kinematics = Mapping **from** joint angles **to** link positions



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

$$\mathbf{T}_n^0 = \mathbf{T}_1^0 \mathbf{T}_2^1 \mathbf{T}_3^2 \mathbf{T}_4^3 \ldots$$

Why Kinematics?

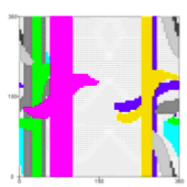
(Robot Kinematics = Mapping **from** joint angles **to** link positions)

- AKA: Why do we need to know where the links are?
- Because that's all that really matters:
  - Where is my hand?
  - Where is my elbow?
  - Where is my center of mass?
  - Where are my eyes?
  - etc…
- Because obstacles and constraints are in world coordinates

S. Joo (sungmoon.joo@cc.gatech.edu) 10/2/2014 45

Let's Consider Planning

- Configuration (State) Space
  - Defined by robot joint values
- Start State
  - Current robot joint values
- Actions?
  - Displacements to robot joints
- Still to do:
  - GOAL STATE?
  - Valid actions?



Goals are rarely specified in joint coordinates!
How do we compute joint space obstacles?

S. Joo (sungmoon.joo@cc.gatech.edu) 10/2/2014 46

Kinematics

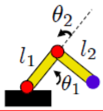
- Forward Kinematics
  Mapping **from** joint angles **to** link positions

- **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)

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

$$x_3 = l_1c_1 + l_2c_{12}$$
$$y_3 = l_1s_1 + l_2s_{12}$$

- Multiple Solutions

- Not always possible to find closed-form solution

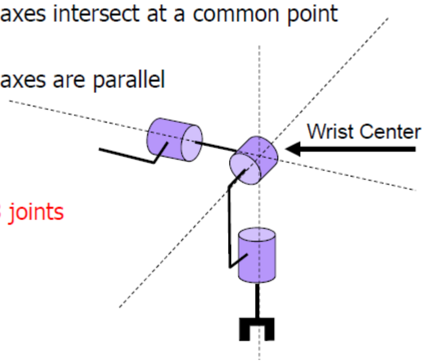
Analytical Methods

- 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

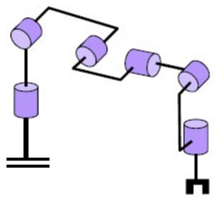


- Wrist position not affected by 3 joints

Analytical IK Example

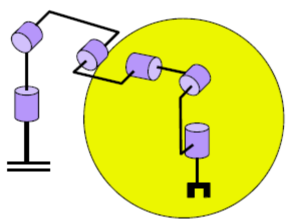
PUMA Style 6 DOF Robot

Goal: $\mathbf{p}_g^0$ and $\mathbf{R}_g^0 = [\mathrm{x}_g \quad \mathrm{y}_g \quad \mathrm{z}_g\ ]^\mathsf{T}$



Analytical IK Example

Goal: $\mathbf{p}_g^0$ and $\mathbf{R}_g^0 = [\mathrm{x}_g \quad \mathrm{y}_g \quad \mathrm{z}_g\ ]^\mathsf{T}$

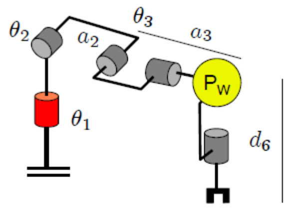


Analytical IK Example

Goal: $\mathbf{p}_G$ and $\mathbf{R}_G^0 = [\mathbf{x}_g \quad \mathbf{y}_g \quad \mathbf{z}_g\ ]^\mathsf{T}$

$$\mathbf{p}_W = \mathbf{p}_G - d_6 \mathbf{z}_g$$

$$\mathbf{T}_3^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_2c_2+a_3c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_2c_2+a_3c_{23}) \\ s_{23} & c_{23} & 0 & a_2s_2+a_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



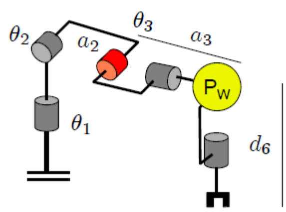
$$\theta_1 = \text{atan2}(p_{Wy}, p_{Wx}) \quad \text{or} \quad \theta_1 = \text{atan2}(p_{Wy}, p_{Wx}) + \pi$$

Analytical IK Example

Goal: $\mathbf{p}_G$ and $\mathbf{R}_G^0 = [\mathbf{x}_g \quad \mathbf{y}_g \quad \mathbf{z}_g\ ]^\mathsf{T}$

$$\mathbf{p}_W = \mathbf{p}_G - d_6 \mathbf{z}_g$$

$$\mathbf{T}_3^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_2c_2+a_3c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_2c_2+a_3c_{23}) \\ s_{23} & c_{23} & 0 & a_2s_2+a_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

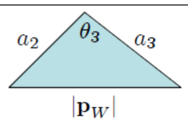


$\theta_1 = \text{atan2}(p_{Wy}, p_{Wx})$ or $\theta_1 = \text{atan2}(p_{Wy}, p_{Wx}) + \pi$

$$\mathbf{p}^T\mathbf{p} = a_2^2 + a_3^2 - 2a_2a_3\cos(\theta_3)$$

$$c_3 = \frac{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 - a_2^2 - a_3^2}{2a_2a_3}$$

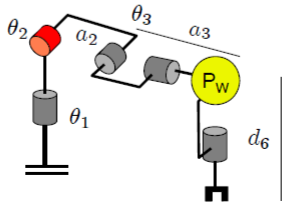
$$s_3 = \pm\sqrt{1 - c_3^2}$$



$$\theta_3 = \text{atan2}(s_3, c_3)$$

Analytical IK Example

Goal: $\mathbf{p}_G$ and $\mathbf{R}_G^0 = [\mathbf{x}_g \quad \mathbf{y}_g \quad \mathbf{z}_g]^\mathsf{T}$



$$\mathbf{p}_W = \mathbf{p}_G - d_6\mathbf{z}_g$$

$$\mathbf{T}_3^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_2c_2 + a_3c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_2c_2 + a_3c_{23}) \\ s_{23} & c_{23} & 0 & a_2s_2 + a_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\theta_1 = \text{atan2}(p_{Wy}, p_{Wx})$ or $\theta_1 = \text{atan2}(p_{Wy}, p_{Wx}) + \pi$

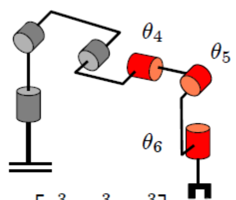
$\theta_3 = \text{atan2}(s_3, c_3)$ $\quad c_3 = \dfrac{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 - a_2^2 - a_3^2}{2a_2a_3}$ $\quad s_3 = \pm\sqrt{1 - c_3^2}$

$$s_2 = \frac{(a_2 + a_3c_3)p_{Wz} - a_3s_3\sqrt{p_{Wx}^2 + p_{Wy}^2}}{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2} \qquad c_2 = \frac{(a_2 + a_3c_3)\sqrt{p_{Wx}^2 + p_{Wy}^2} + a_3s_3p_{Wz}}{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2}$$

$$\theta_2 = \text{atan2}(s_2, c_2)$$

Analytical IK Example: Wrist

Goal: $\mathbf{p}_G$ and $\mathbf{R}_G^0 = [\mathbf{x}_g \quad \mathbf{y}_g \quad \mathbf{z}_g]^\mathsf{T}$



$$\mathbf{R}_6^3 = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 \\ -s_5c_6 & s_5s_6 & c_5 \end{bmatrix} = \begin{bmatrix} n_x^3 & s_x^3 & a_x^3 \\ n_y^3 & s_y^3 & a_y^3 \\ n_z^3 & s_z^3 & a_z^3 \end{bmatrix}$$

$\theta_5 \in (0, \pi)$

$$\begin{aligned} \theta_4 &= \text{atan2}(a_y^3, a_x^3) \\ \theta_5 &= \text{atan2}(\sqrt{(a_x^3)^2 + (a_y^3)^2}, a_z^3) \\ \theta_6 &= \text{atan2}(s_z^3, -n_z^3) \end{aligned}$$

$\theta_5 \in (-\pi, 0)$

$$\begin{aligned} \theta_4 &= \text{atan2}(-a_y^3, -a_x^3) \\ \theta_5 &= \text{atan2}(-\sqrt{(a_x^3)^2 + (a_y^3)^2}, a_z^3) \\ \theta_6 &= \text{atan2}(-s_z^3, n_z^3) \end{aligned}$$

Analytical IK : Summary

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

ALTERNATIVE:

- Differential Kinematics