Robot Arm Transformations, Path Planning, and Trajectories

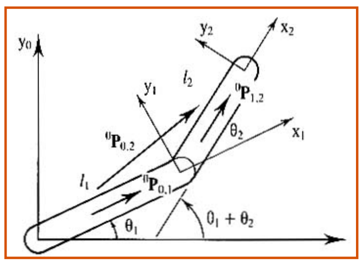
Robert Stengel
Robotics and Intelligent Systems
MAE 345, Princeton University, 2015

- Joint-link-joint transformations
  - Denavit-Hartenberg representation
- Path planning
  - Voronoi diagrams and Delaunay triangulation
  - Probabilistic Road Map
  - Rapidly Exploring Random Tree
- Closed-form trajectories; connecting the dots
  - Polynomials and splines
  - Aceleration profiles

Series of Homogeneous Transformations

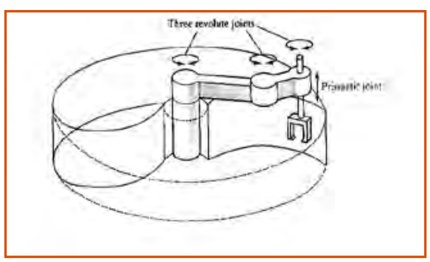
Two serial transformations can be combined in a single transformation

$$\boxed{\mathbf{s}_2 = \mathbf{A}_1^2\mathbf{A}_0^1\ \mathbf{s}_0 = \mathbf{A}_0^2\ \mathbf{s}_0}$$

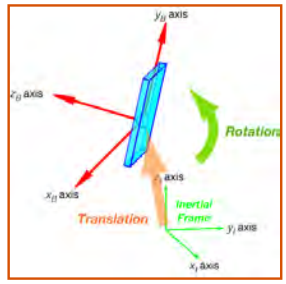


Four transformations for SCARA robot

$$\boxed{\mathbf{s}_4 = \mathbf{A}_3^4\mathbf{A}_2^3\mathbf{A}_1^2\mathbf{A}_0^1\ \mathbf{s}_0 = \mathbf{A}_0^4\ \mathbf{s}_0}$$



Recall: The homogeneous transformation matrix expresses rotation and translation in a single transformation



$$\mathbf{s}_{new} = \left[\begin{array}{c|c} \begin{pmatrix} \text{Rotation} \\ \text{Matrix} \end{pmatrix}_{old}^{new} & \begin{pmatrix} \text{Location} \\ \text{of Old} \\ \text{Origin} \end{pmatrix}_{new} \\ \hline \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{array} \right] \mathbf{s}_{old} = \mathbf{A}_{old}^{new}\ \mathbf{s}_{old}$$

$$(4\times 1)_{new} = \left[\begin{array}{c|c} (3\times 3) & (3\times 1) \\ \hline (1\times 3) & (1\times 1) \end{array} \right] (4\times 1)_{old} = \left[(4\times 4) \right] (4\times 1)_{old}$$

Recall: Homogeneous Transformation

- **Rotation and translation can be expressed in terms of homogeneous coordinates**
  - **Single matrix-vector product produces rotation and transformation**

$$\mathbf{s}_{new} = \begin{bmatrix} H_{old}^{new} & \mathbf{r}_{old_{new}} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{bmatrix} \mathbf{s}_{old} = \mathbf{A}\ \mathbf{s}_{old}$$

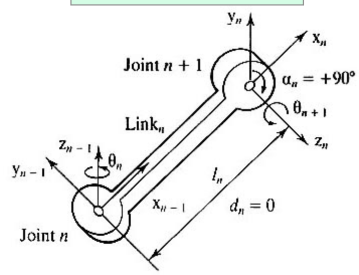
- **or**

$$\begin{bmatrix} x \\ y \\ x \\ 1 \end{bmatrix}_{new} = \left[\begin{array}{ccc|c} h_{11} & h_{12} & h_{13} & x_o \\ h_{21} & h_{22} & h_{23} & y_o \\ h_{31} & h_{32} & h_{33} & z_o \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{old}$$

Transformation for a Single Robotic Joint

- **Each joint requires four sequential transformations:**
  - Rotation about α
  - Translation along d
  - Translation along l
  - Rotation about θ

Type 2 Link

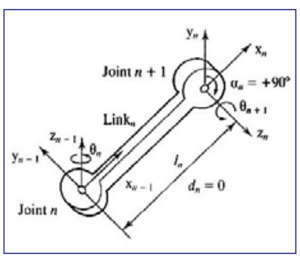


$$\begin{aligned}\mathbf{s}_{n+1} &= \mathbf{A}_3^{n+1}\mathbf{A}_2^{3}\mathbf{A}_1^{2}\mathbf{A}_n^{1}\ \mathbf{s}_n = \mathbf{A}_n^{n+1}\ \mathbf{s}_n \\ &= \mathbf{A}_\theta\mathbf{A}_d\mathbf{A}_l\mathbf{A}_\alpha\ \mathbf{s}_n = \mathbf{A}_n^{n+1}\ \mathbf{s}_n\end{aligned}$$

- **... axes for each transformation (along or around) must be specified**

4th 3rd 2nd 1st

$$\mathbf{s}_{n+1} = \mathbf{A}\left(z_{n-1},\theta_n\right)\mathbf{A}\left(z_{n-1},d_n\right)\mathbf{A}\left(x_{n-1},l_n\right)\mathbf{A}\left(x_{n-1},\alpha_n\right)\mathbf{s}_n = \mathbf{A}_n^{n+1}\ \mathbf{s}_n$$



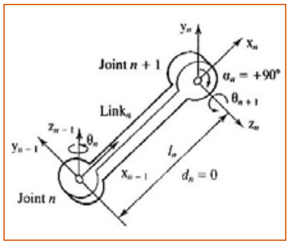
Denavit-Hartenberg Representation of Joint-Link-Joint Transformation

- **Like Euler angle rotation, transformational effects of the 4 link parameters are defined in a specific application sequence (right to left): $\{\theta, d, l, \alpha\}$**

- **4 link parameters**
  - **Angle between 2 links, θ (revolute)**
  - **Distance (offset) between links, d (prismatic)**
  - **Length of the link between rotational axes, l, along the common normal (prismatic)**
  - **Twist angle between axes, α (revolute)**

$$\begin{aligned}\mathbf{A}_n &= \mathbf{A}\left(z_{n-1},\theta_n\right)\mathbf{A}\left(z_{n-1},d_n\right)\mathbf{A}\left(x_{n-1},l_n\right)\mathbf{A}\left(x_{n-1},\alpha_n\right) \\ &= \mathrm{Rot}(z_{n-1},\theta_n)\ \mathrm{Trans}(z_{n-1},d_n)\ \mathrm{Trans}(x_{n-1},l_n)\ \mathrm{Rot}(x_{n-1},\alpha_n) \\ &\triangleq {}^{n}\mathbf{T}_{n+1} \quad \text{in some references (e.g., McKerrow, 1991)}\end{aligned}$$

Denavit-Hartenberg Demo
http://www.youtube.com/watch?v=10mUtjfGmzw



Four Transformations from One Joint to the Next

(Single Link)

Rotation of θ_n about the z_{n-1} axis

$$\mathrm{Rot}(z_{n-1},\theta_n)=\begin{bmatrix}\cos\theta_n & -\sin\theta_n & 0 & 0\\ \sin\theta_n & \cos\theta_n & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Translation of d_n along the z_{n-1} axis

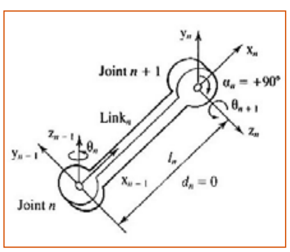
$$\mathrm{Trans}(z_{n-1},d_n)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_n\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Translation of l_n along the x_{n-1} axis

$$\mathrm{Trans}(x_{n-1},l_n)=\begin{bmatrix}1 & 0 & 0 & l_n\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotation of α_n about the x_{n-1} axis

$$\mathrm{Rot}(x_{n-1},\alpha_n)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha_n & -\sin\alpha_n & 0\\ 0 & \sin\alpha_n & \cos\alpha_n & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

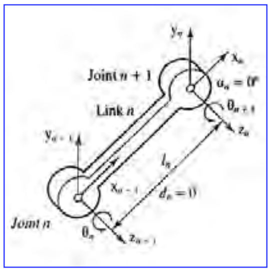


Denavit-Hartenberg Representation of Joint-Link-Joint Transformation

Then **Then** **Then** **First**

$$\mathbf{A}_n=\begin{bmatrix}\cos\theta_n & -\sin\theta_n & 0 & 0\\ \sin\theta_n & \cos\theta_n & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_n\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & l_n\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha_n & -\sin\alpha_n & 0\\ 0 & \sin\alpha_n & \cos\alpha_n & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$\mathbf{A}_n=\begin{bmatrix}\cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & l_n\cos\theta_n\\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & l_n\sin\theta_n\\ 0 & \sin\alpha_n & \cos\alpha_n & d_n\\ 0 & 0 & 0 & 1\end{bmatrix}$$



Example: Denavit-Hartenberg Representation of Joint-Link-Joint Transformation for Type 1 Link

Joint Variable = θ_n

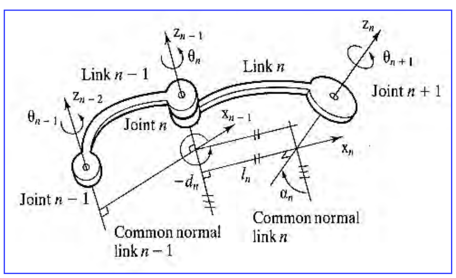
$$\mathbf{A}_n = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & l_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & l_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\theta = \text{variable}$
$d = 0$ m
$l = 0.25$ m
$\alpha = 90$ deg

$$\mathbf{A}_n = \begin{bmatrix} \cos\theta_n & 0 & \sin\theta_n & 0.25\cos\theta_n \\ \sin\theta_n & 0 & -\cos\theta_n & 0.25\sin\theta_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\theta \triangleq 30$ deg
$d = 0$ m
$l = 0.25$ m
$\alpha = 90$ deg

$$\mathbf{A}_n = \begin{bmatrix} 0.866 & 0 & 0.5 & 0.217 \\ 0.5 & 0 & -0.866 & 0.125 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Forward and Inverse Transformations

Forward transformation through links requires pre-multiplication of matrices

$$\mathbf{s}_1 = \mathbf{A}_0^1\mathbf{s}_0 \quad ; \quad s_2 = \mathbf{A}_1^2\mathbf{s}_1 = \mathbf{A}_1^2\mathbf{A}_0^1\mathbf{s}_0 = \mathbf{A}_0^2\mathbf{s}_0$$

Reverse transformation uses the matrix inverse

$$\mathbf{s}_0 = \left(\mathbf{A}_0^2\right)^{-1}\mathbf{s}_2 = \mathbf{A}_2^0\mathbf{s}_2 = \mathbf{A}_1^0\mathbf{A}_2^1\mathbf{s}_2$$

Homogeneous Transformation Matrix is not Orthonormal

$$\mathbf{A}_2^0 = \left(\mathbf{A}_0^2\right)^{-1} \neq \left(\mathbf{A}_0^2\right)^T$$

…but a useful identity makes inversion simple

Matrix Inverse Identity

Given: a square matrix, A, and its inverse, B

$$\mathbf{A} = \left[\begin{array}{c|c} \underset{m\times m}{\mathbf{A}_1} & \underset{m\times n}{\mathbf{A}_2} \\ \hline \underset{n\times m}{\mathbf{A}_3} & \underset{n\times n}{\mathbf{A}_4} \end{array}\right] \quad ; \quad \mathbf{B} \triangleq \mathbf{A}^{-1} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{B}_3 & \mathbf{B}_4 \end{bmatrix}$$

Then

$$\mathbf{A}\,\mathbf{B} = \mathbf{A}\,\mathbf{A}^{-1} = \mathbf{I}_{m+n}$$

$$= \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_n \end{bmatrix} = \begin{bmatrix} \left(\mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_3\right) & \left(\mathbf{A}_1\mathbf{B}_2 + \mathbf{A}_2\mathbf{B}_4\right) \\ \left(\mathbf{A}_3\mathbf{B}_1 + \mathbf{A}_4\mathbf{B}_3\right) & \left(\mathbf{A}_3\mathbf{B}_2 + \mathbf{A}_4\mathbf{B}_4\right) \end{bmatrix}$$

Equating like parts, and solving for $\mathbf{B}_i$

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{B}_3 & \mathbf{B}_4 \end{bmatrix} = \left[\begin{array}{c|c} \left(\mathbf{A}_1 - \mathbf{A}_2\mathbf{A}_4^{-1}\mathbf{A}_3\right)^{-1} & -\mathbf{A}_1^{-1}\mathbf{A}_2\left(\mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2\right)^{-1} \\ \hline -\mathbf{A}_4^{-1}\mathbf{A}_3\left(\mathbf{A}_1 - \mathbf{A}_2\mathbf{A}_4^{-1}\mathbf{A}_3\right)^{-1} & \left(\mathbf{A}_4 - \mathbf{A}_3\mathbf{A}_1^{-1}\mathbf{A}_2\right)^{-1} \end{array}\right]$$

Apply to Homogeneous Transformation

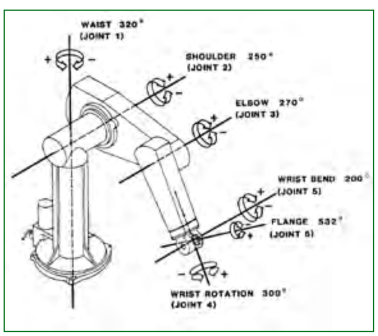
Forward transformation

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{old}^{new} & \mathbf{r}_o \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{bmatrix}$$

Inverse

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_3 & \mathbf{A}_4 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{B}_3 & \mathbf{B}_4 \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{H}_{new}^{old} & -\mathbf{H}_{new}^{old}\mathbf{r}_o \\ \hline \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} h_{11} & h_{21} & h_{31} & -\left(h_{11}x_o + h_{21}y_o + h_{31}z_o\right) \\ h_{12} & h_{22} & h_{32} & -\left(h_{12}x_o + h_{22}y_o + h_{32}z_o\right) \\ h_{13} & h_{23} & h_{33} & -\left(h_{13}x_o + h_{23}y_o + h_{33}z_o\right) \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$



Manipulator Maneuvering Spaces

- **Joint space**: Vector of joint variables, e.g.,

$$\mathbf{r}_J = \begin{bmatrix} \theta_{waist} & \theta_{shoulder} & \theta_{elbow} & \theta_{wrist-bend} & \theta_{flange} & \theta_{wrist-twist} \end{bmatrix}^T$$

- **End-effecter space**: Vector of end-effecter positions, e.g.,

$$\mathbf{r}_E = \begin{bmatrix} x_{tool} & y_{tool} & z_{tool} & \psi_{tool} & \theta_{tool} & \phi_{tool} \end{bmatrix}^T$$

- **Task space**: Vector of task-dependent positions, e.g., locating a symmetric grinding tool above a horizontal surface:

$$\mathbf{r}_T = \begin{bmatrix} x_{tool} & y_{tool} & z_{tool} & \psi_{tool} & \theta_{tool} \end{bmatrix}^T$$

Forward and Inverse Transformations of a Robotic Assembly

Forward Transformation

Transforms homogeneous coordinates from tool frame to reference frame coordinates

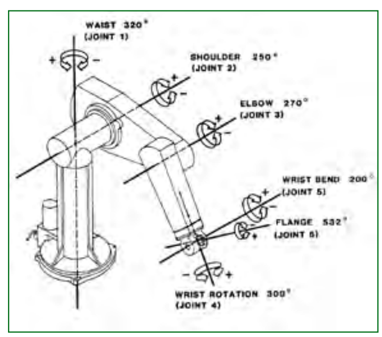
$$
\begin{aligned}
s_{base} &= \mathbf{A}_{tool}^{base}\mathbf{s}_{tool} \\
&= \mathbf{A}_{waist}\mathbf{A}_{shoulder}\mathbf{A}_{elbow}\mathbf{A}_{wrist-bend}\mathbf{A}_{flange}\mathbf{A}_{wrist-twist}\mathbf{s}_{tool}
\end{aligned}
$$

Inverse Transformation

Transform homogeneous coordinate from reference frame to tool frame coordinates

$$
\begin{aligned}
s_{tool} &= \mathbf{A}_{base}^{tool}\mathbf{s}_{base} \\
&= \mathbf{A}^{-1}{}_{wrist-twist}\mathbf{A}^{-1}{}_{flange}\mathbf{A}^{-1}{}_{wrist-bend}\mathbf{A}^{-1}{}_{elbow}\mathbf{A}^{-1}{}_{shoulder}\mathbf{A}^{-1}{}_{waist}\mathbf{s}_{base}
\end{aligned}
$$

Forward and Inverse Kinematics Between Joints, Tool Position, and Tool Orientation



Forward Kinematic Problem: Compute the position of the tool in the reference frame that corresponds to a given joint vector (i.e., vector of link variables)

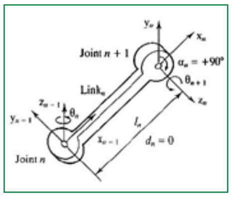
$$
s_{base} = \mathbf{A}_{waist}\mathbf{A}_{shoulder}\mathbf{A}_{elbow}\mathbf{A}_{wrist-bend}\mathbf{A}_{flange}\mathbf{A}_{wrist-twist}\mathbf{s}_{tool} = \mathbf{A}_{tool}^{base}\mathbf{s}_{tool}
$$

$$
\text{To Be Determined} \Leftarrow \text{Given}
$$

Inverse Kinematic Problem: Find the vector of link variables that corresponds to a desired task-dependent position

$$
\mathbf{A}_{waist}\mathbf{A}_{shoulder}\mathbf{A}_{elbow}\mathbf{A}_{wrist-bend}\mathbf{A}_{flange}\mathbf{A}_{wrist-twist}\mathbf{s}_{tool} = \mathbf{A}_{tool}^{base}\mathbf{s}_{0} = s_{base}
$$

$$
\text{To Be Determined} \quad \Leftarrow \quad \text{Given}
$$



Forward and Inverse Kinematics
Single-Link Example

Forward Kinematic Problem: Specify task-dependent position that corresponds to a given joint variable (= θ_n)

$$\mathbf{s}_{n-1} = \mathbf{A}(z_{n-1},\theta_n)\mathbf{A}(z_{n-1},d_n)\mathbf{A}(x_{n-1},l_n)\mathbf{A}(x_{n-1},\alpha_n)\,\mathbf{s}_n$$
$$= \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & l_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & l_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{s}_n$$

Red: Known
Blue: Unknown

Forward and Inverse Kinematics
Single-Link Example

Inverse Problem: Find the joint variable, θ, that corresponds to a desired task-dependent position

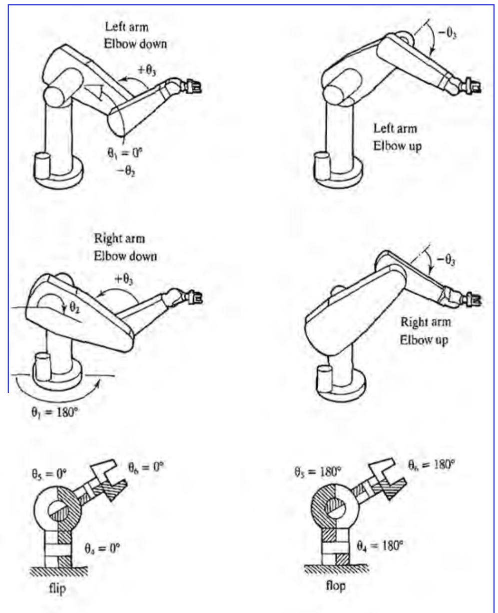
$$\mathbf{s}_{n-1} = \mathbf{A}(z_{n-1},\theta_n)\mathbf{A}(z_{n-1},0)\mathbf{A}(x_{n-1},l_n)\mathbf{A}(x_{n-1},90°)\,\mathbf{s}_n$$
$$= \begin{bmatrix} \cos\theta_n & 0 & \sin\theta_n & l_n\cos\theta_n \\ \sin\theta_n & 0 & -\cos\theta_n & l_n\sin\theta_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{s}_n$$

Red: Known
Blue: Unknown

$$x_{n-1} = x_n\cos\theta_n + z_n\sin\theta_n + l_n\cos\theta_n$$
$$y_{n-1} = x_n\sin\theta_n - z_n\cos\theta_n + l_n\sin\theta_n$$

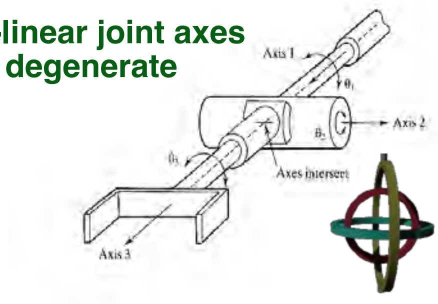
Solve by elimination and inverse trig functions

Manipulator Redundancy and Degeneracy



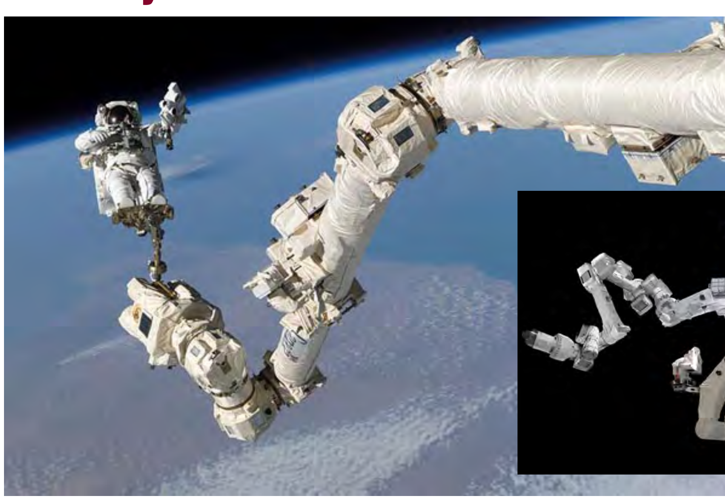
- More than one link configuration may provide a given end point if $\dim(x_J) \geq \dim(x_E) \geq \dim(x_T)$
- **Redundancy**: Finite number of joint vectors provide the same task-dependent vector
- **Degeneracy**: Infinite number of joint vectors provide the same task-dependent vector

- Co-linear joint axes are degenerate



Space Robot Arms are Highly Redundant

- **Why?**



Link	variable	θ	α	l	d
1	θ_1	θ_1	0	l_1	0
2	θ_2	θ_2	0	l_2	0

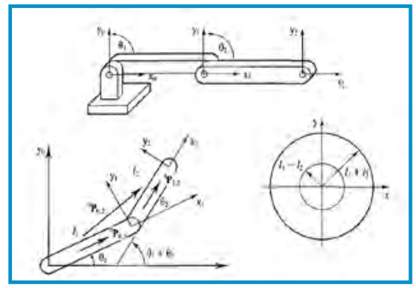
Transformations for a Two-Link Manipulator

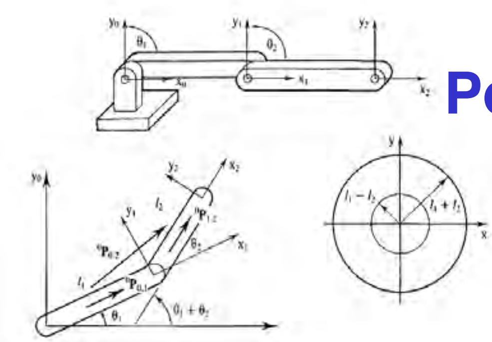
- **Example**: **Type 1 Two-Link Manipulator, neglecting offset** **(e.g., Puma geometry without wave and wrist)**

$$\mathbf{H}_0^1 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ; \quad \mathbf{r}_0 = \begin{bmatrix} -l_1 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{H}_0^1 & \mathbf{r}_0 \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & -l_1 \\ -\sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} \mathbf{H}_1^2 & \mathbf{r}_1 \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 & -l_2 \\ -\sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





Position of Distal Joint Relative to the Base

(2-link manipulator)

$$\boxed{\theta_B = \theta_1 + \theta_2}$$

$$\mathbf{s}_{base} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{base} = \mathbf{A}_1\mathbf{A}_2\mathbf{s}_{distal} = \begin{bmatrix} \cos\theta_B & -\sin\theta_B & 0 & l_1\cos\theta_1 + l_2\cos\theta_B \\ \sin\theta_B & \cos\theta_B & 0 & l_1\sin\theta_1 + l_2\sin\theta_B \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}_{distal}$$

$$= \begin{bmatrix} l_1\cos\theta_1 + l_2\cos\theta_B \\ l_1\sin\theta_1 + l_2\sin\theta_B \\ 0 \\ 1 \end{bmatrix}$$

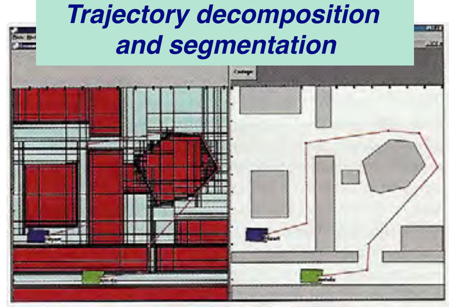
Path Planning

Baxter Path Planning (UNC, 2014)
https://www.youtube.com/watch?v=oY1FfytaD-c

Path Planning

Environment idealization and nominal path

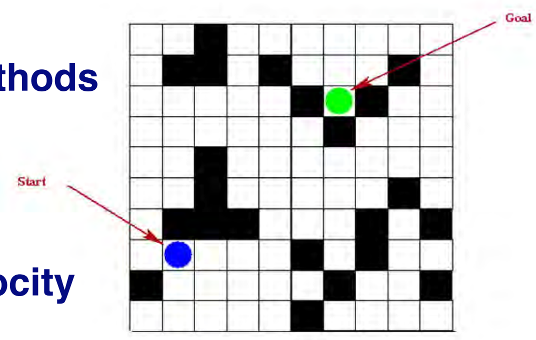
Trajectory decomposition and segmentation



- **Well-defined Start and Goal**
- **Waypoints**
- **Path primitives (line, circle, etc.)**
- **Timing and coordination**
- **Obstacle detection and avoidance**
- **Feasibility and regulation**
- **Optimization and constraint**

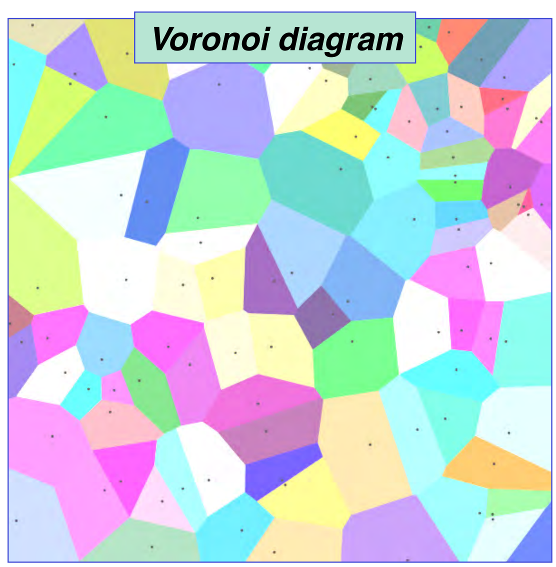
Path Planning with Waypoints

- Define Start, Goal, and Waypoints by position and time
- Connect the dots
- Various interpolation methods
  - Straight lines
  - Polynomials
  - Splines
- Generate associated velocity and acceleration
- Satisfy trajectory constraints

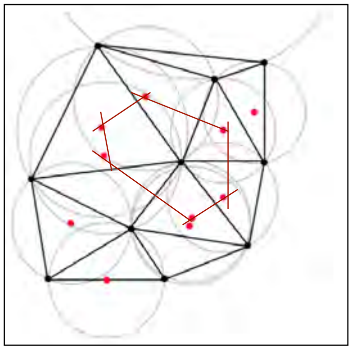


Path Planning with Obstacles and Destinations

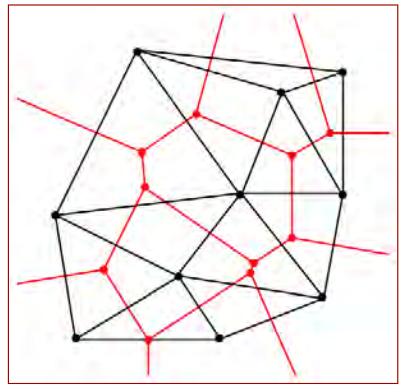
- Given set of points, e.g., obstacles, destinations, or centroids of multiple points
- Chart best path from start to goal
- Tessellation (tiling) of decision space
- 2-D Voronoi diagram
  - Polygons with sides equidistant to two nearest points (black dots)



Delaunay Triangulation Constructs the Voronoi Diagram



- Threats/obstacles are **black** points
- **Edges** (**black**) connect all triplets of **black** points lying on circumferences of empty circles, i.e., circles containing no other **black** points
- "Circumcircle" **centers** are **red** points



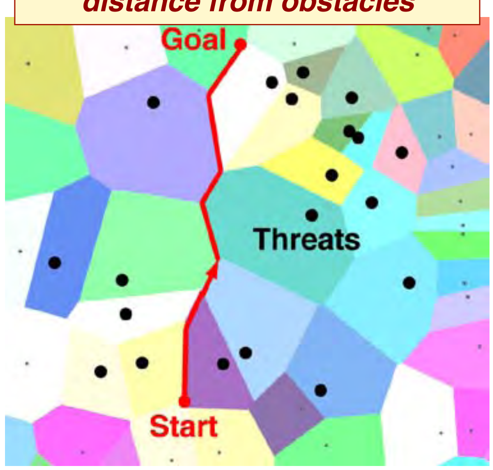
- Voronoi segment boundaries (**red**) connect **centers** and are **perpendicular** to each edge

https://en.wikipedia.org/wiki/Delaunay_triangulation

Voronoi Diagrams in Path Planning

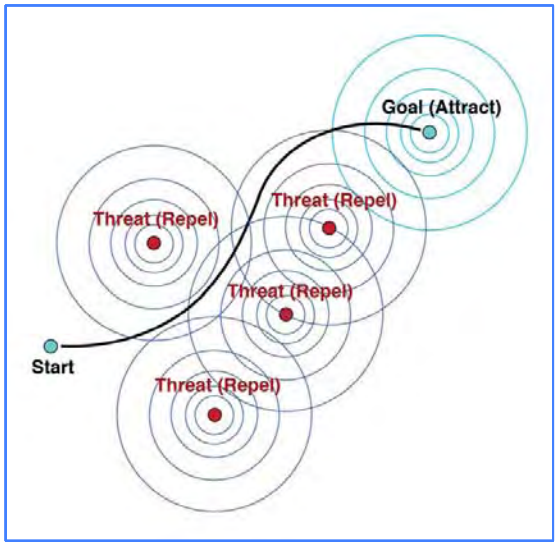
Threat/obstacle avoidance

Choose path with farthest distance from obstacles



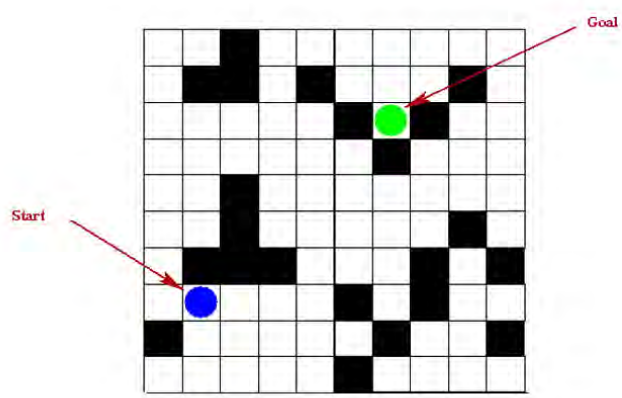
Path Planning with Potential Fields

Map features attract or repel path from Start to Goal, e.g., +/– gravity fields



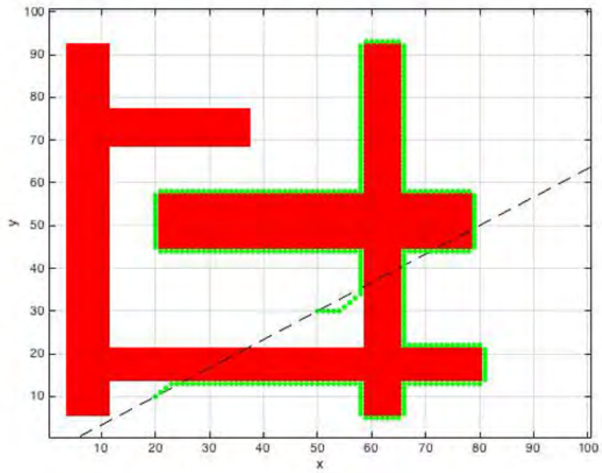
Path Planning on Occupancy Grid

Admissible and Inadmissible Blocks



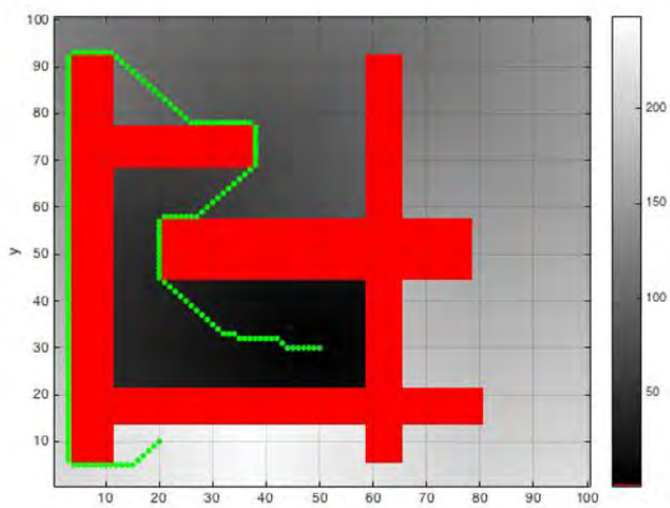
- Identify feasible paths from Start to Goal
- Chose path that best satisfies criteria, e.g.,
  - Simplicity of calculation
  - Lowest cost
  - Highest performance

Bug Path Planning



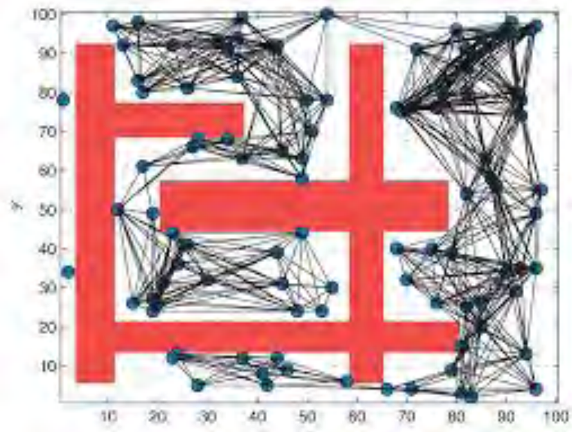
1) Identify shortest unconstrained path from Start to Goal
2) Chose path that navigates the boundary
   1) Stays as close as to possible to unconstrained path
   2) Satisfies constraint
   3) Follows simple rule, e.g., "stay to the left"

D* or A* Path Planning *(TBD)*



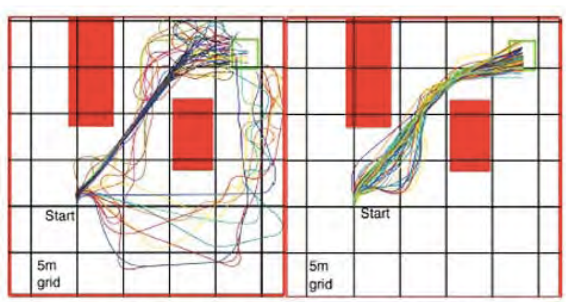
- Determine ***occupancy cost*** of each block
- Chose path from Start to Goal that
  - Reduce occupancy cost with each step

Probabilistic Road Map (PRM)



- Construct random configuration of admissible points
- Connect admissible points to nearest neighbors
- ***Assess incremental cost*** of traveling along each "edge" between points
- Query to find all feasible paths from Start to Goal
- ***Select lowest cost path***

Rapidly Exploring Random Tree (RRT*)



(a) RRT (b) RRT*

Space-filling tree evolves from Start

Open-loop trajectories with state constraints

Initially feasible solution converges to optimal solution through searching

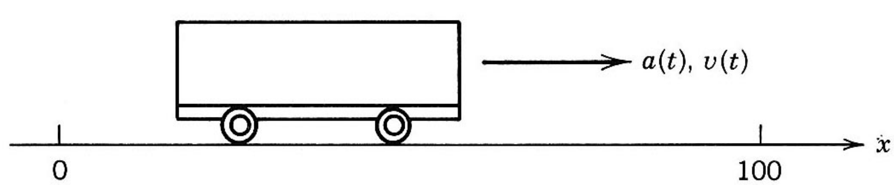
Committed trajectories

Branch-and-bound tree adaptation

Trajectories

One-Dimensional Trajectory

Constant Velocity, *v*



Velocity, *v(t)* vs. *t*, is constant

$$\boxed{v(t) = \dot{x}(t) = v(0)}$$

Position, *x(t)* vs. *t*, is a straight line

$$\boxed{x(t) = x(0) + v(0)}$$

One-Dimensional Trajectory

Constant Velocity, *v*

Position specified at 0 and *t*

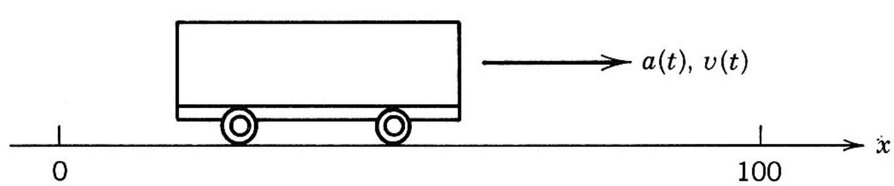
$$\begin{bmatrix} x(0) \\ x(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & t \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \end{bmatrix}$$

Velocity at 0 to be determined

$$\begin{bmatrix} x(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & t \end{bmatrix}^{-1} \begin{bmatrix} x(0) \\ x(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1/t & 1/t \end{bmatrix} \begin{bmatrix} x(0) \\ x(t) \end{bmatrix}$$

One-Dimensional Trajectory

Constant Acceleration, *a*



Velocity, *v(t)* vs. *t*, is a straight line

$$v(t) = \dot{x}(t) = v(0) + at$$

Position, *x(t)* vs. *t*, is a parabola

$$x(t) = x(0) + v(t) + at^2/2$$

One-Dimensional Trajectory

Constant Acceleration, *a*

Position specified at 0 and *t*; velocity specified at 0

$$\boxed{\begin{bmatrix} x(0) \\ x(t) \\ v(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & t & t^2/2 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \\ a(0) \end{bmatrix}}$$

Acceleration at 0 to be determined

$$\boxed{\begin{bmatrix} x(0) \\ v(0) \\ a(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & t & t^2/2 \\ 0 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -2/t^2 & 2/t^2 & -2/t \end{bmatrix} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \end{bmatrix}}$$

One-Dimensional Trajectory

Constant Jerk, *j*, = Derivative of Acceleration, *a*

Acceleration, *a(t)* vs. *t*, is a straight line

$$\boxed{a(t) = \dot{v}(t) = \ddot{x}(t) = a(0) + jt}$$

Velocity, *v(t)* vs. *t*, is a parabola

$$\boxed{v(t) = \dot{x}(t) = v(0) + a(0)t + jt^2/2}$$

Position, *x(t)* vs. *t*, is cubic

$$\boxed{x(t) = x(0) + v(0)t + a(0)t^2/2 + jt^3/6}$$

One-Dimensional Trajectory

Constant Jerk, *j*

Position and **velocity specified at 0 and *t***;
acceleration and jerk at **0** to be determined

$$\begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t & t^2/2 & t^3/6 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & t & t^2/2 \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j \end{bmatrix}$$

One-Dimensional Trajectory

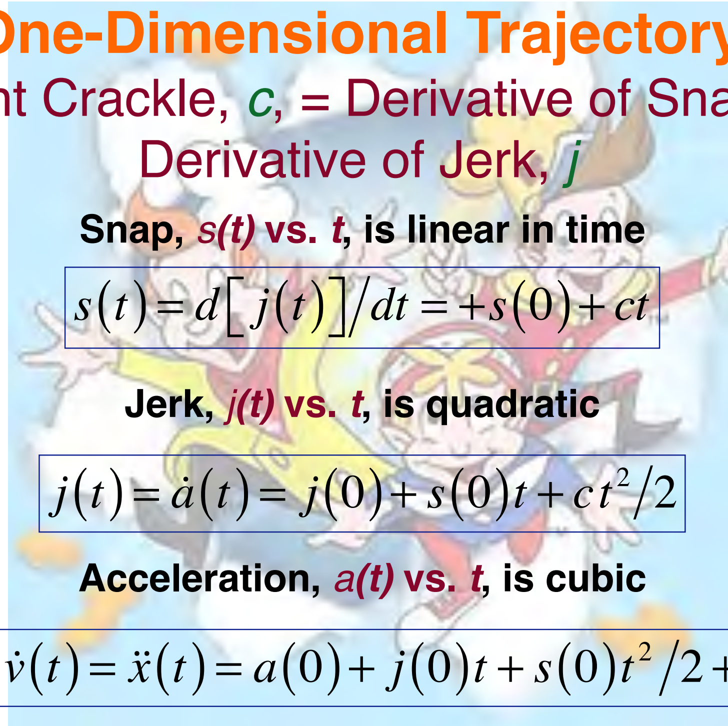
Constant Jerk, *j*

Find *a*(0) and *j* to produce desired position and velocity

$$\begin{array}{c} \text{Start} \\ \text{Finish} \\ \text{Start} \\ \text{Finish} \end{array} \quad \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t & t^2/2 & t^3/6 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & t & t^2/2 \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j \end{bmatrix} \quad \begin{array}{c} \text{Start} \\ \text{Start} \\ \textit{TBD} \\ \textit{TBD} \end{array}$$

Inverse of (4 x 4) relationship defines required *a*(0) and *j*

$$\begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t & t^2/2 & t^3/6 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & t & t^2/2 \end{bmatrix}^{-1} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -6/t^2 & 6/t^2 & -4/t & -2/t \\ 12/t^3 & -12/t^3 & 6/t^2 & 6/t^2 \end{bmatrix} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \end{bmatrix}$$



One-Dimensional Trajectory

Constant Crackle, *c*, = Derivative of Snap, *s*, = Derivative of Jerk, *j*

Snap, *s(t)* vs. *t*, is linear in time

$$\boxed{s(t) = d\left[j(t)\right]/dt = +s(0) + ct}$$

Jerk, *j(t)* vs. *t*, is quadratic

$$\boxed{j(t) = \dot{a}(t) = j(0) + s(0)t + ct^2/2}$$

Acceleration, *a(t)* vs. *t*, is cubic

$$\boxed{a(t) = \dot{v}(t) = \ddot{x}(t) = a(0) + j(0)t + s(0)t^2/2 + ct^3/6}$$

One-Dimensional Trajectory

with Constant Crackle, *c*

Velocity, *v(t)* vs. *t*, is quartic

$$\boxed{v(t) = \dot{x}(t) = v(0) + a(0)t + jt^2/2 + s(0)t^3/6 + ct^4/24}$$

Position, *x(t)* vs. *t*, is quintic

$$\boxed{x(t) = x(0) + v(0)t + a(0)t^2/2 + j(0)t^3/6 + s(0)t^4/24 + ct^5/120}$$

One-Dimensional Trajectory

with Constant Crackle, *c*

Position, velocity, and acceleration specified at 0 and *t*

$$\begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \\ a(0) \\ a(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & t & t^2/2 & t^3/6 & t^4/24 & t^5/120 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & t & t^2/2 & t^3/6 & t^4/24 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & t & t^2/2 & t^3/6 \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j(0) \\ s(0) \\ c \end{bmatrix}$$

One-Dimensional Trajectory

Inverse of (6 x 6) relationship defines controls

$$\begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j(0) \\ s(0) \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & t & t^2/2 & t^3/6 & t^4/24 & t^5/120 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & t & t^2/2 & t^3/6 & t^4/24 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & t & t^2/2 & t^3/6 \end{bmatrix}^{-1} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \\ a(0) \\ a(t) \end{bmatrix}$$

$$\begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j(0) \\ s(0) \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -60/t^3 & 60/t^3 & -36/t^2 & -24/t^2 & -9/t & 3/t \\ 360/t^4 & -360/t^4 & 192/t^3 & 168/t^3 & 36/t^2 & -24/t^2 \\ -720/t^5 & 720/t^5 & -360/t^4 & -360/t^4 & -60/t^3 & 60/t^3 \end{bmatrix} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \\ a(0) \\ a(t) \end{bmatrix}$$

One-Dimensional Trajectory

Eliminate unnecessary equations and define acceleration constants

$$\begin{bmatrix} j(0) \\ s(0) \\ c \end{bmatrix} = \begin{bmatrix} -60/t^3 & 60/t^3 & -36/t^2 & -24/t^2 & -9/t & 3/t \\ 360/t^4 & -360/t^4 & 192/t^3 & 168/t^3 & 36/t^2 & -24/t^2 \\ -720/t^5 & 720/t^5 & -360/t^4 & -360/t^4 & -60/t^3 & 60/t^3 \end{bmatrix} \begin{bmatrix} x(0) \\ x(t) \\ v(0) \\ v(t) \\ a(0) \\ a(t) \end{bmatrix}$$

Corresponding acceleration and force are specified by

$$\boxed{a(t) = a(0) + j(0)t + s(0)t^2/2 + ct^3/6}$$

$$= a_{control}(t) + a_{gravity}(t) + a_{disturbance}(t)$$
$$= \left[f_{control}(t) + f_{gravity}(t) + f_{disturbance}(t) \right] / m(t)$$

One-Dimensional Trajectory

Calculate trajectory components, given acceleration constants

$$\begin{bmatrix} x(t) \\ v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2/2 & t^3/6 & t^4/24 & t^5/120 \\ 0 & 1 & t & t^2/2 & t^3/6 & t^4/24 \\ 0 & 0 & 1 & t & t^2/2 & t^3/6 \end{bmatrix} \begin{bmatrix} x(0) \\ v(0) \\ a(0) \\ j(0) \\ s(0) \\ c \end{bmatrix}$$

Example

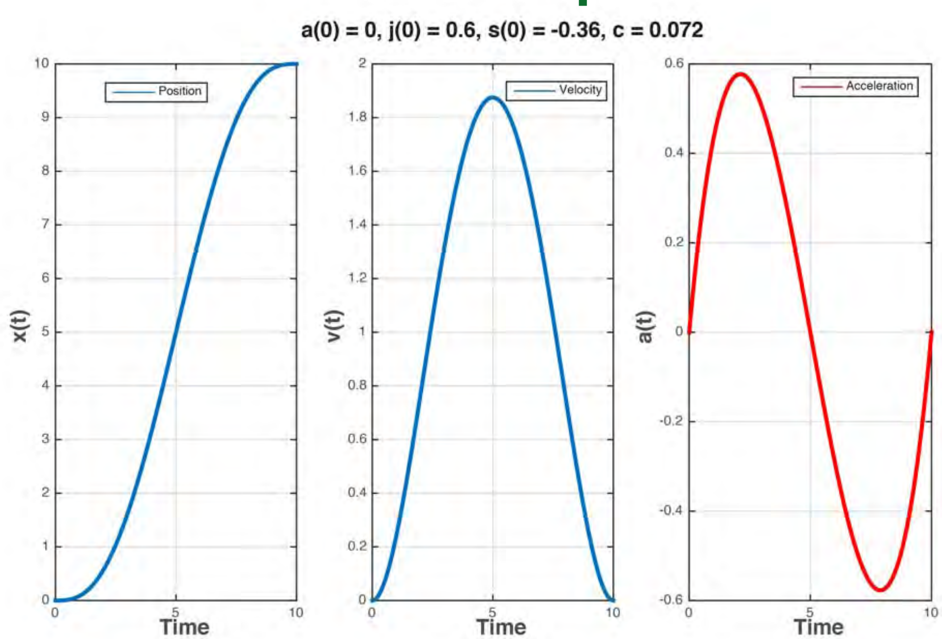
Calculate constants for x(0) = 0, x(10) = 10

$$\begin{bmatrix} 0.6 \\ -0.36 \\ 0.072 \end{bmatrix} = \begin{bmatrix} -60/10^3 & 60/10^3 & -36/10^2 & -24/10^2 & -9/10 & 3/10 \\ 360/10^4 & -360/10^4 & 192/10^3 & 168/10^3 & 36/10^2 & -24/10^2 \\ -720/10^5 & 720/10^5 & -360/10^4 & -360/10^4 & -60/10^3 & 60/10^3 \end{bmatrix} \begin{bmatrix} 0 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Calculate trajectory, given constants for $t_f = 10$

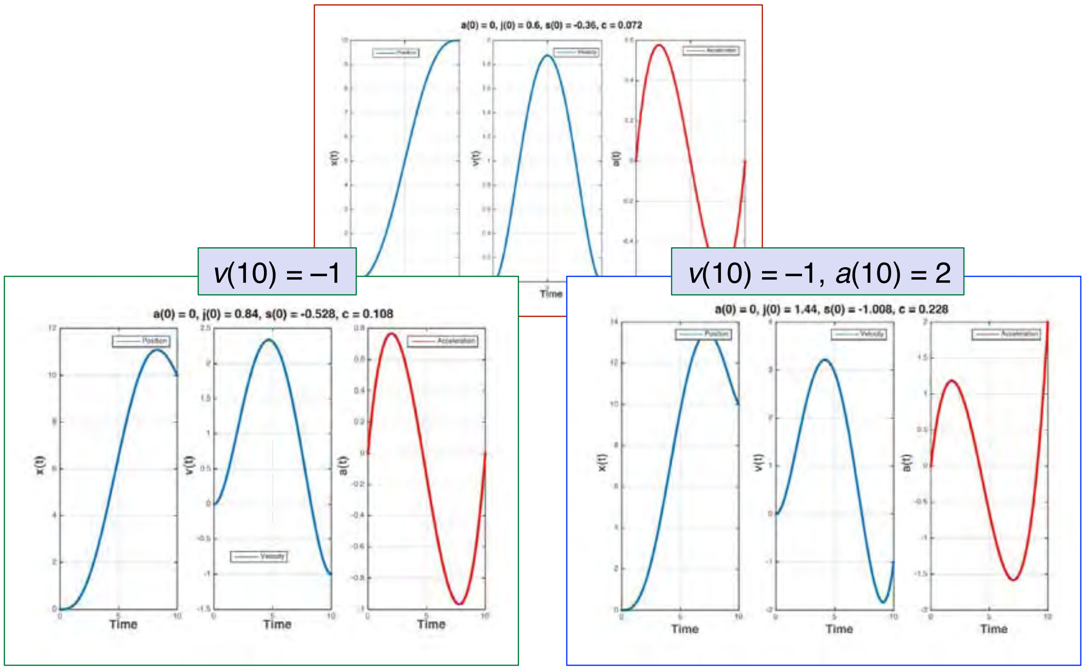
$$\begin{bmatrix} x(t) \\ v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2/2 & t^3/6 & t^4/24 & t^5/120 \\ 0 & 1 & t & t^2/2 & t^3/6 & t^4/24 \\ 0 & 0 & 1 & t & t^2/2 & t^3/6 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.6 \\ -0.36 \\ 0.072 \end{bmatrix}$$

1-D Example



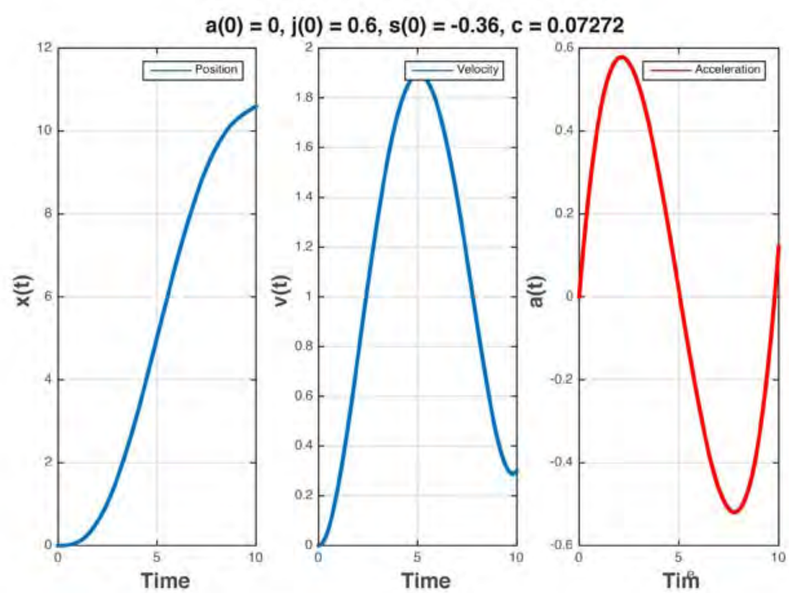
$$a_{net}(t) = (0) + 0.6t - 0.36t^2/2 + 0.072t^3/6$$

Examples with Different End Conditions

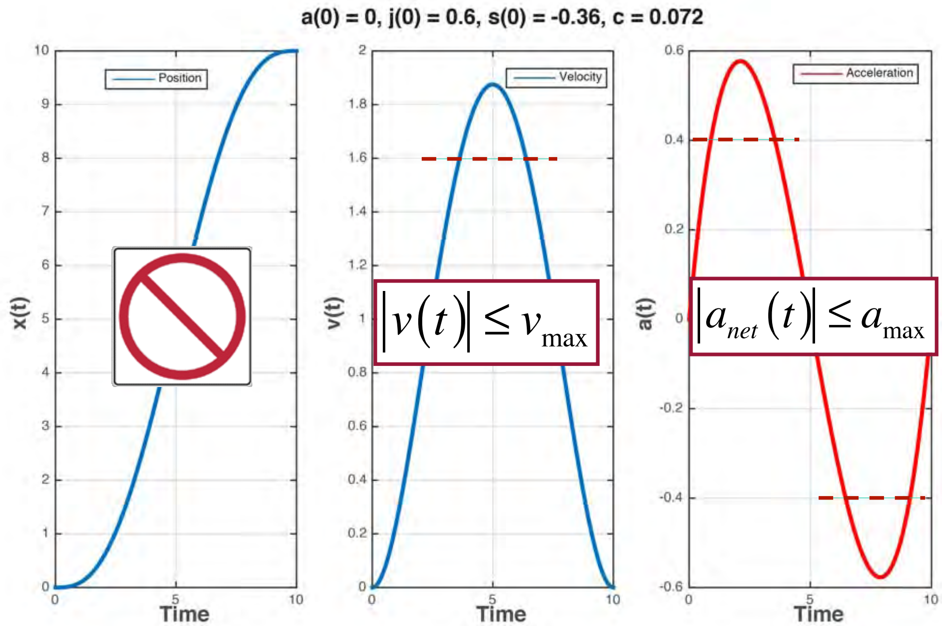


Sensitivity to Errors

1% error in Crackle



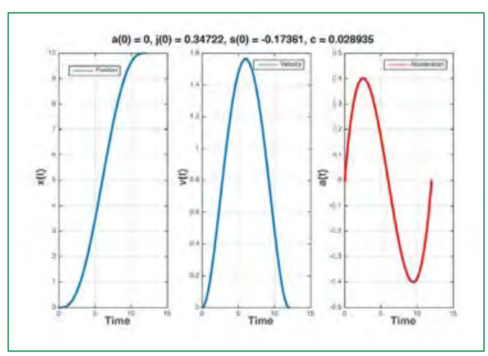
Constrained 1-D Trajectories

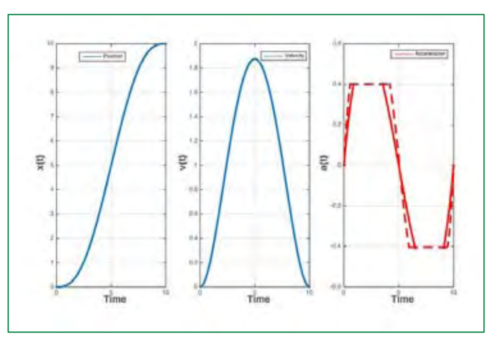


What are the alternatives for achieving desired end conditions?

Alternatives for Reaching End Position

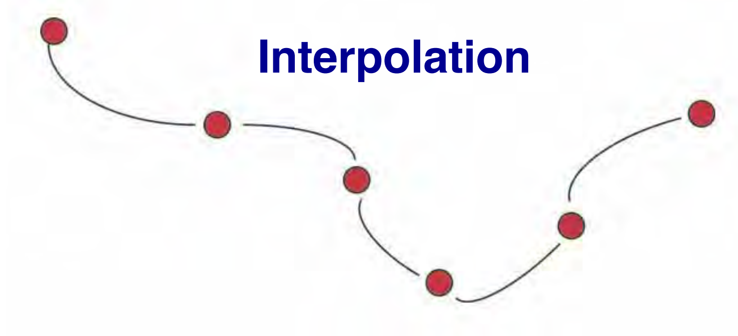
- **Increase end time**
  - **Lower max/min values of velocity and acceleration**
- **"Fatten" velocity and acceleration profiles**
  - **Multi-segment trajectory**
  - **Unconstrained arcs**
  - **Constrained arcs (velocity and/or acceleration held constant**





Connect the Dots

Interpolation

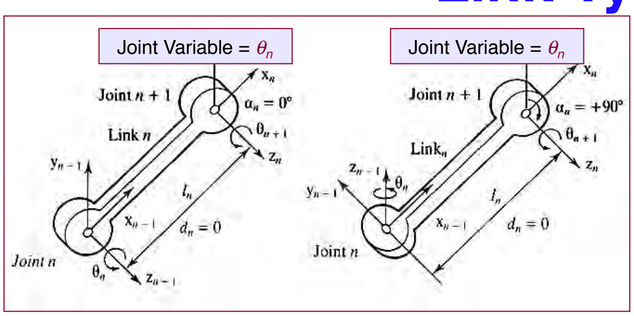


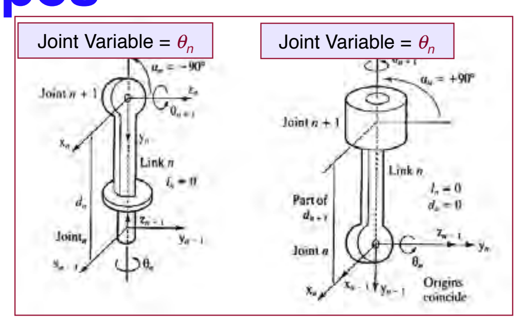
- **Piecewise polynomials (linear -> quintic)**
  - **End-point discontinuities**
  - **End-point constraints**
- **Single polynomial through all points**
  - **Polynomial degree = # of points**
  - **Sensitivity to high-degree terms (e.g., ct^6)**
  - **Possibility of large excursions between points**
- **Polynomials through adjacent points**
  - **e.g., cubic B splines**

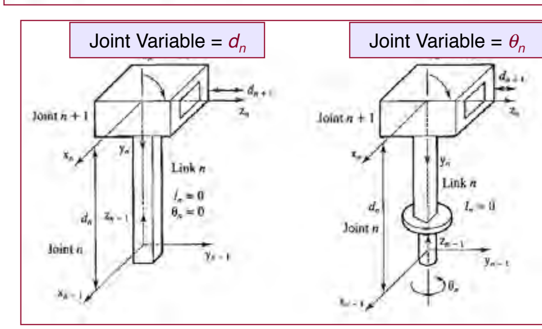
Next Time: Time Response of Dynamic Systems

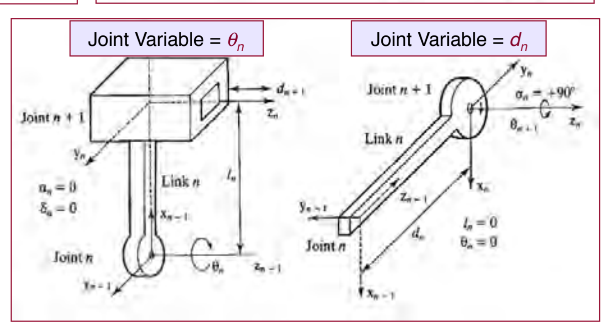
Supplemental Material

Joint Variables for Different Link Types



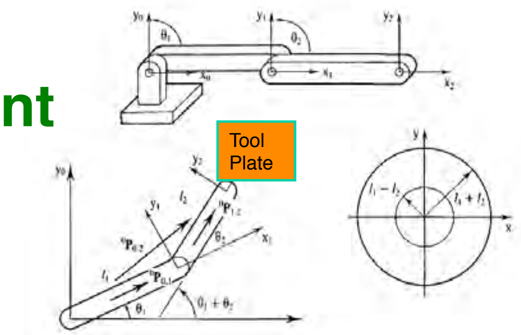






Position of Distal Joint Relative to the Base

(2-link manipulator)



- **Suppose a tool plate is fixed to the distal joint at $(x\ y\ z)_{distal}^T$; then**

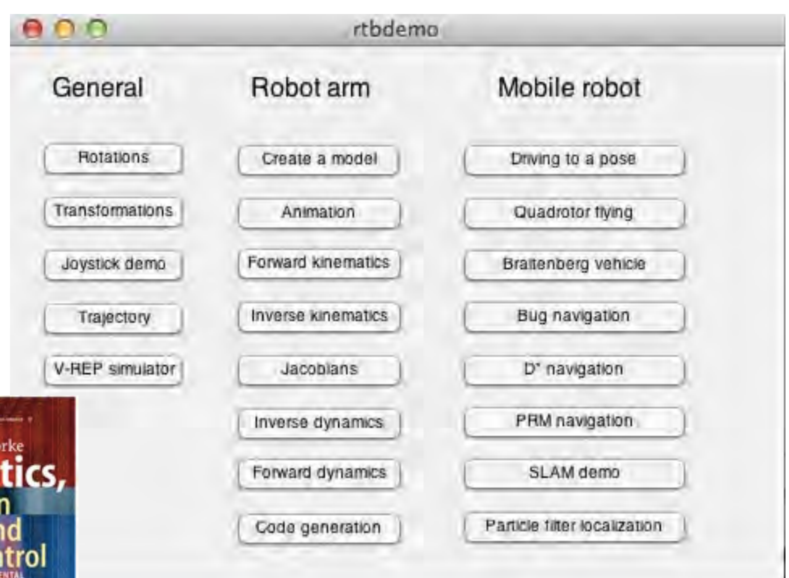
$$\mathbf{s}_{base} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{base} = \mathbf{A}_1\mathbf{A}_2\mathbf{s}_{distal} = \begin{bmatrix} \cos\theta_B & -\sin\theta_B & 0 & l_1\cos\theta_1 + l_2\cos\theta_B \\ \sin\theta_B & \cos\theta_B & 0 & l_1\sin\theta_1 + l_2\sin\theta_B \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{distal}$$

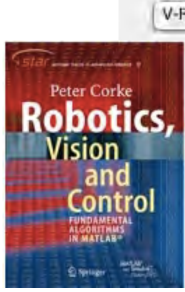
$$= \begin{bmatrix} x\cos\theta_B - y\sin\theta_B + l_1\cos\theta_1 + l_2\cos\theta_B \\ x\sin\theta_B + y\cos\theta_B + l_1\sin\theta_1 + l_2\sin\theta_B \\ z \\ 1 \end{bmatrix}$$

Alternatively, straightforward trigonometry could be used in this example

rtbdemo (rvctools.m)

http://petercorke.com/Robotics_Toolbox.html



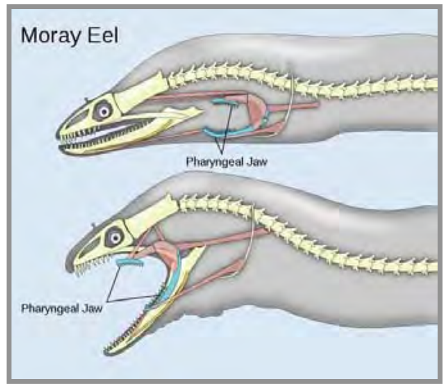


End Effecters, Tool Plates, and Jaws

Multi-Bar Linkages
http://www.youtube.com/watch?v=YDd6VBx9oqU&feature=related

Tool Changer
http://www.youtube.com/watch?v=G8ZqoOIEDHY&feature=related

Another Tool Changer
http://www.youtube.com/watch?v=LkPnt_nudLc&feature=related



Robot Arms for Space


Canadarm2, ISS


Dextre Manipulator, ISS


Kibo Arm, ISS


Curiosity Robot Arm

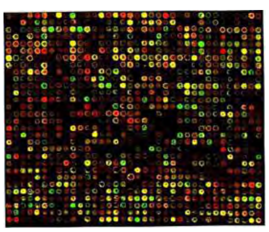
Multi-Jointed Arms

Snake-Like Manipulator

Octopus Arms

OctArm
http://www.youtube.com/watch?v=Qzvqni7O_XQs

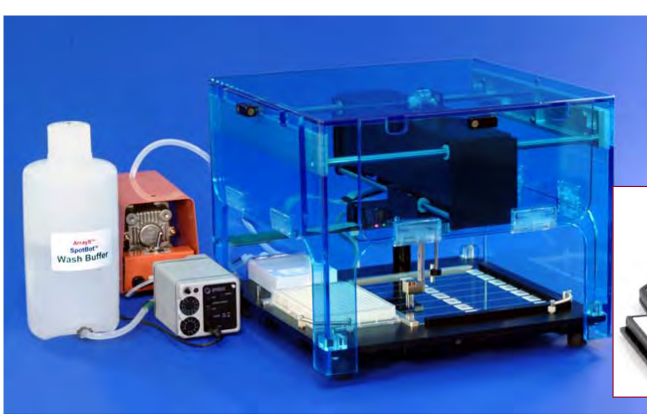
Tentacle Arm
http://www.youtube.com/watch?v=Yk7Muaigd4k



DNA Microarray-Spotting Robot

- **DNA strands representing different genes are spotted on a microscope slide**
- **Finished slide is used to analyze DNA from tissue samples**

http://www.youtube.com/watch?v=Z_KNhD1jz-k



American Android Multi-Arm UGV

(David Handelman, *89)

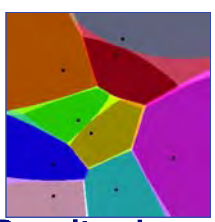
http://www.youtube.com/watch?v=pOi6OdcPKfk

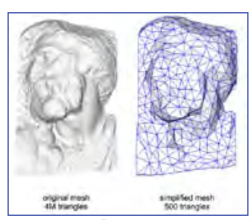
http://www.youtube.com/watch?v=tVZFJ7yivxI

http://www.youtube.com/watch?v=qdM48cAg0U4

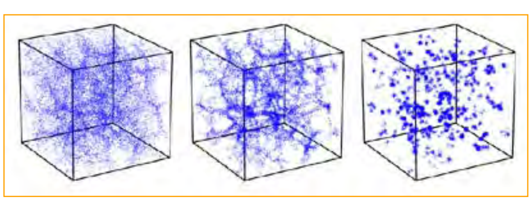
Voronoi Diagrams in Data Processing

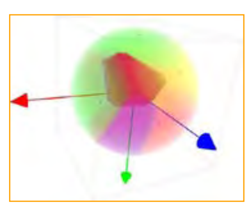
Computer graphics textures (2-D and 3-D meshes)





Density characterization (3-D mesh)





Vector quantization in data compression

http://www.data-compression.com/vqanim.shtml