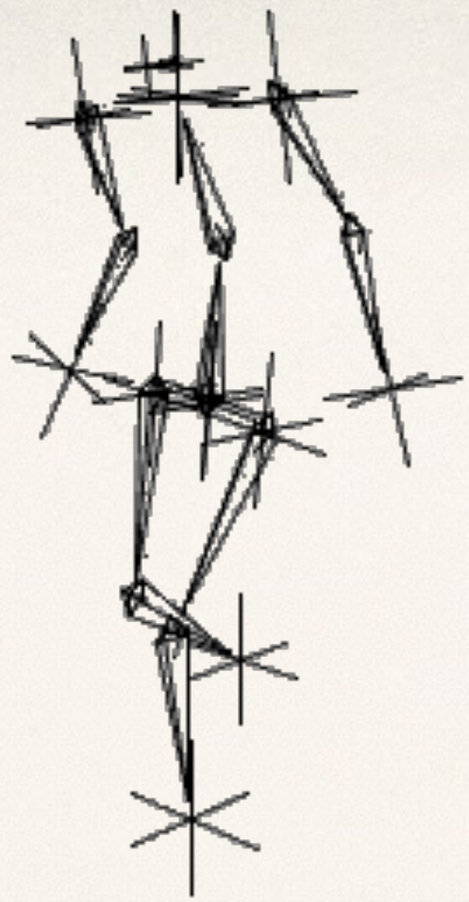


Softimage tutorials: Actor module, articulated / kinematic chains

# Kinematics of articulated characters

Advanced Computer Animation Techniques
Aug-Dec 2014

cesteves@cimat.mx

# Hierarchical kinematic modeling

- Hierarchical modeling is the enforcement of connectivity (or relative placement) constraints among objects organized in a treelike structure.
  - Individual objects are defined in their own local coordinate systems.
  - These objects are assembled into a figure in world coordinate system using nested transformations.
- The global transformations applied to any particular object within the skeleton can be computed by traversing the hierarchy from the root to the segment and concatenating the local transformations at each joint visited by the traversal.

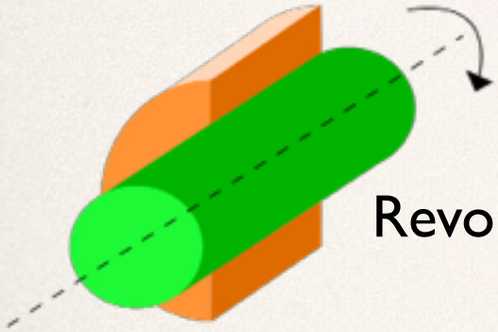
# Hierarchical kinematic modeling

- A skeleton can be animated by varing the local rotations applied at each joint over time as well as the global translation applied at the root joint.
- The control problem is managing the way in which these transformations change over time.
  - Kinematics
    - Forward Kinematics
    - Inverse Kinematics
  - Dynamics
    - Forward Dynamics
    - Inverse Dynamics

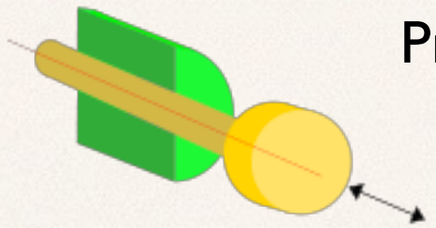
# Hierarchical kinematic modeling

- **Links** :
  - rigid objects forming the connections between the joints.
  - components of the hierarchy that represent objects that are physically connected.
- **End effector**: free end of the chain of alternating joints and links.
- **Frame** : local coordinate system associated with each joint.
- Motion is often restricted.
- Animate linkages by specifying or determine position of parameters over time.

# Degrees of freedom (DOFs)

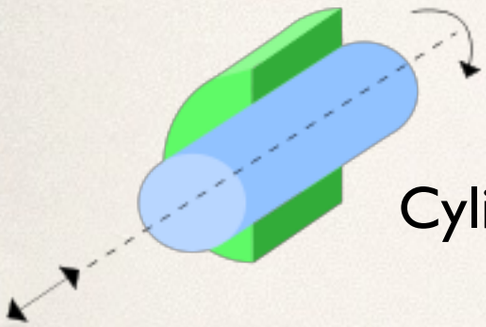


Revolute joint

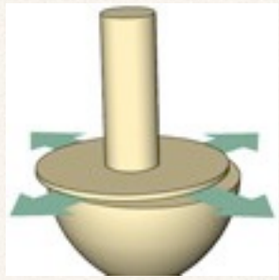



Prismatic joint

Ball-and-socket joint



Cylindrical joint

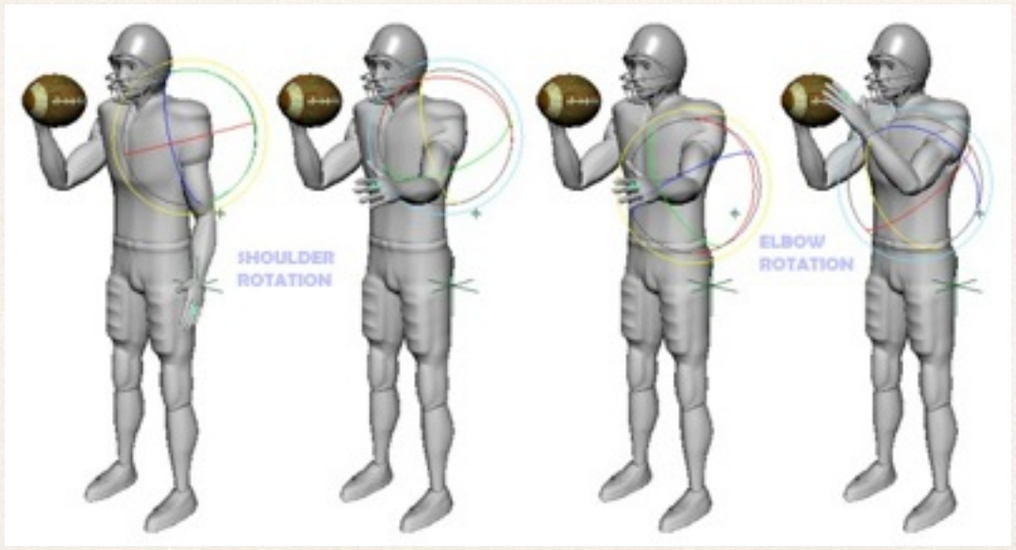


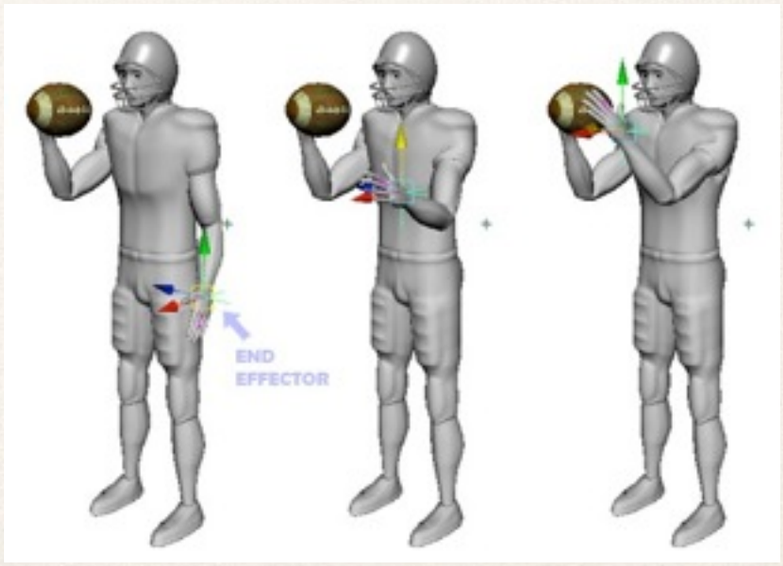
Planar joint

- Minimum number of coordinates required to specify completely the motion of an object.
- Complex joints of DOF *n* can be modeled as *n* one-DOF joints connected by *n-1* links of zero length.

# Kinematics

- Branch of mechanics concerned with the motions of objects without regard to the forces causing them.
- **Forward kinematics** : animator specifies values of articulation variables, global transform for each linkage is computed.
- **Inverse kinematics** : animator specifies final desired global transform of an end-effector and the system solves for the joint angles that satisfy those constraints.



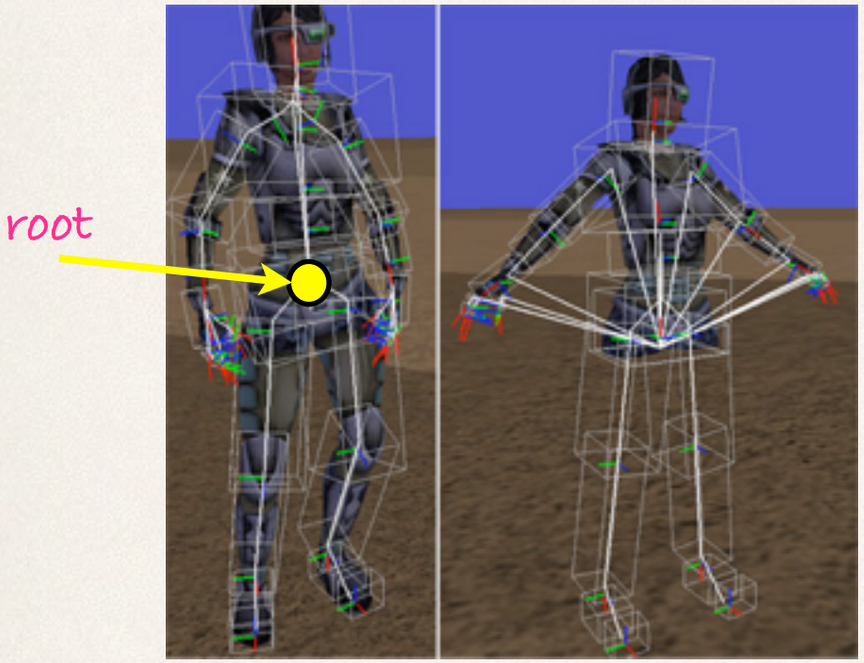



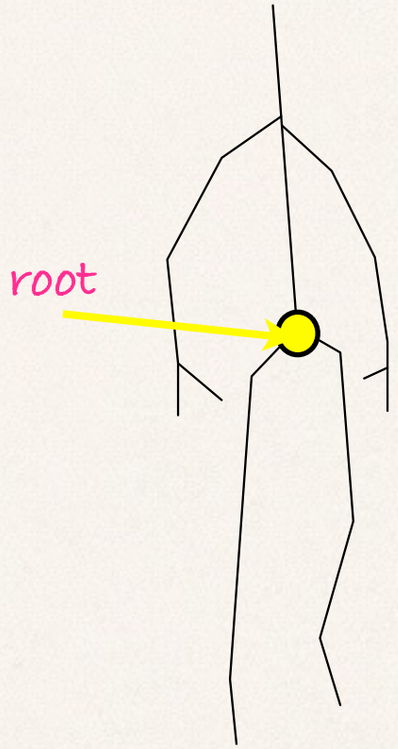

Forward and Inverse Kinematics. R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

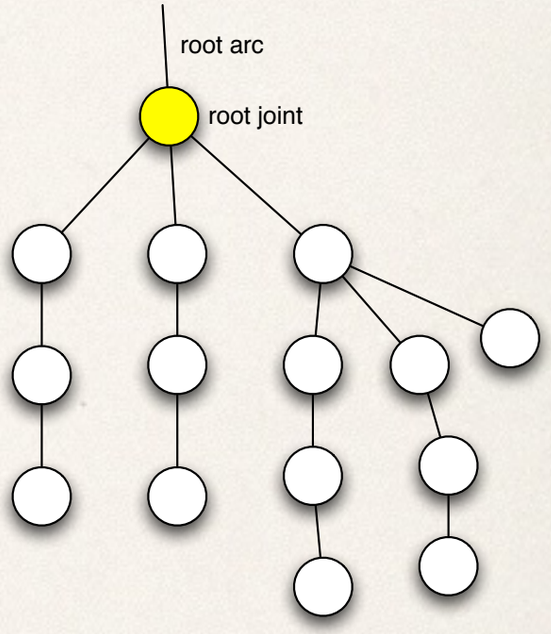
# Hierarchical kinematic modeling

- Links and joints are represented as a tree structure of nodes connected by arcs.
- The highest node in the tree is the root node, which corresponds to the root object of the hierarchy.
- Root node : corresponds to the root object whose position is known in the global coordinate sytem.
- All the other nodes are specified relative to the root node or previously defined nodes that can be tracked up to the root.
- Leaf nodes: end-effectors.

# Hierarchical representation

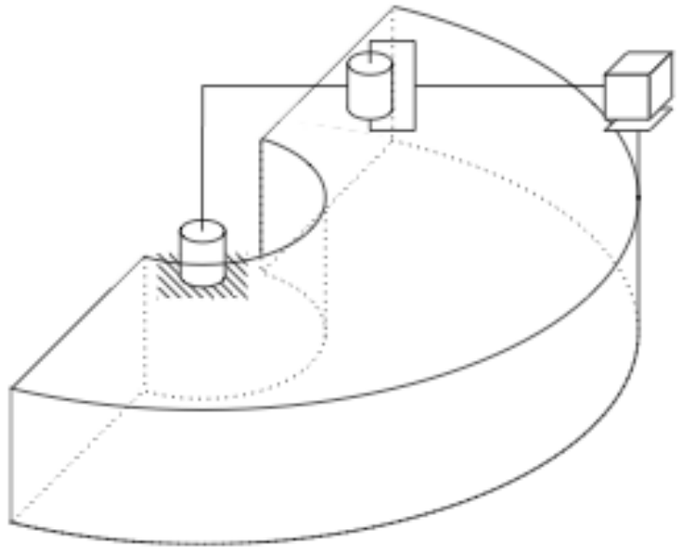


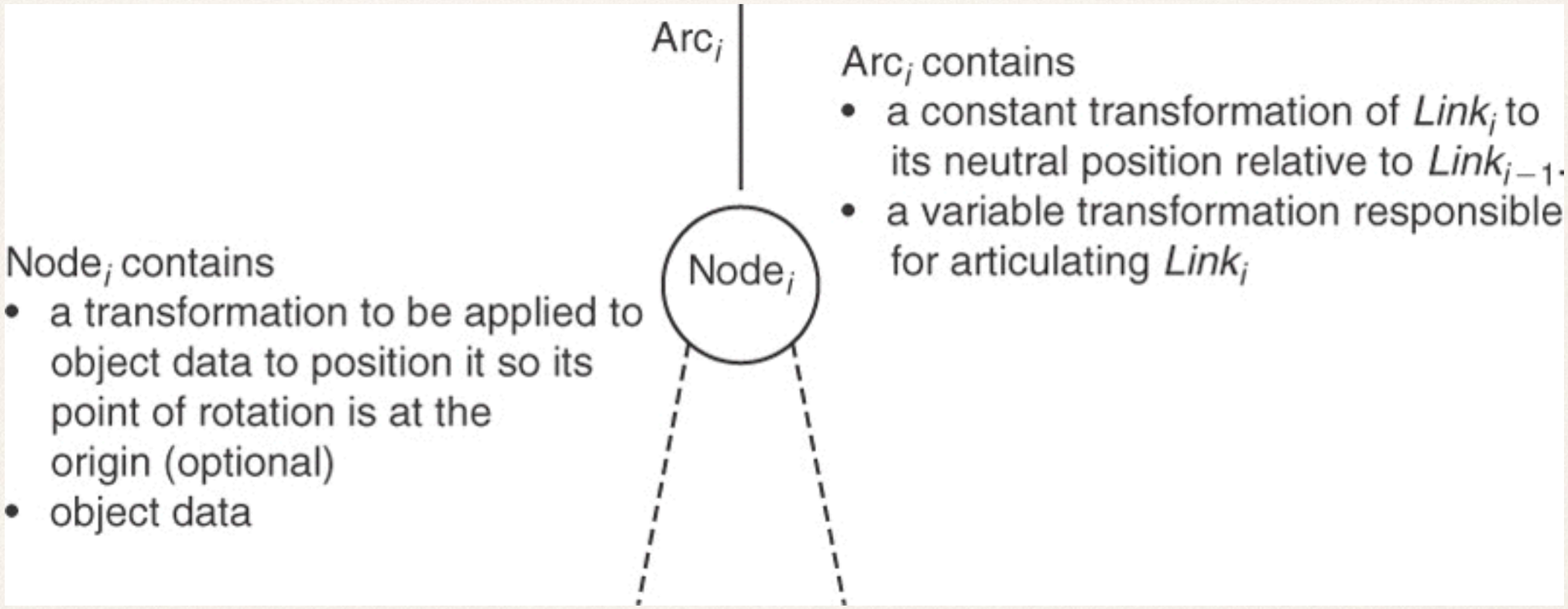
# Configuration space vs. Workspace

- The **configuration** of an articulated mechanism is the complete specification of the position and orientation of its bodies.
- The **configuration space** is the set of all possible configurations that the mechanism can reach.
- The number of **degrees of freedom** (dofs) is the number of variables in which the configuration is specified.
- The **workspace** of a manipulator is the total volume that the end-effector can sweep while the mechanism moves through all its posible configurations.



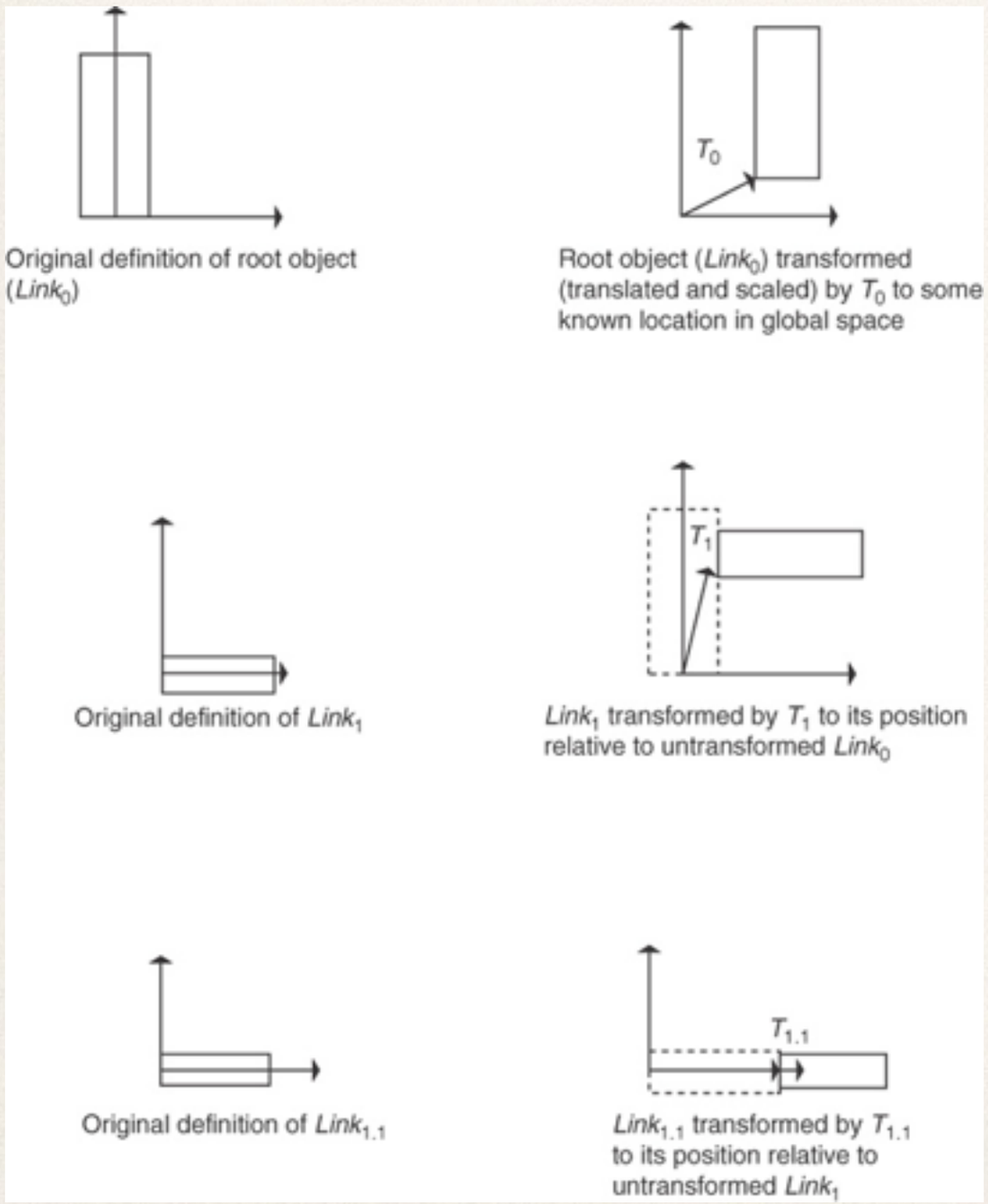
Y. Nakamura, Advanced Robotics. Addison-Wesley. 1991

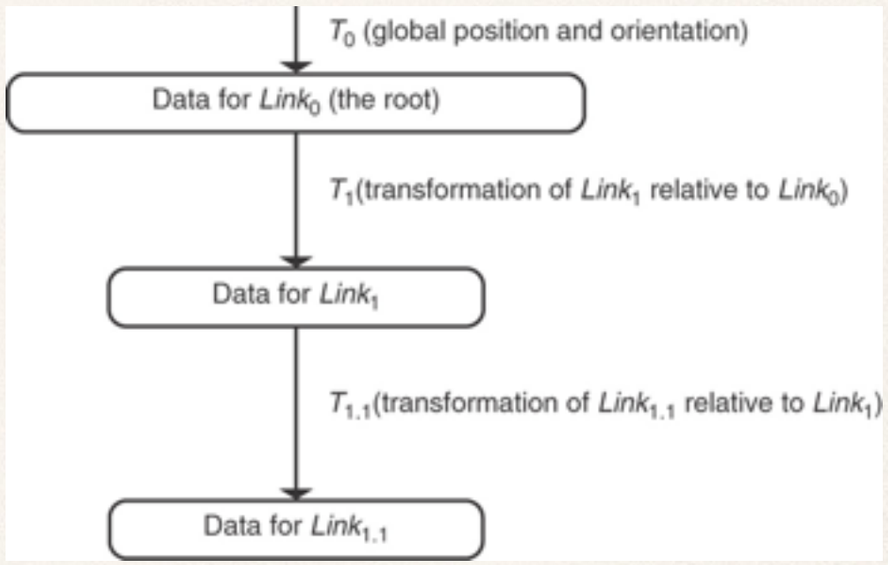
# Tree structure




R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

# Tree structure



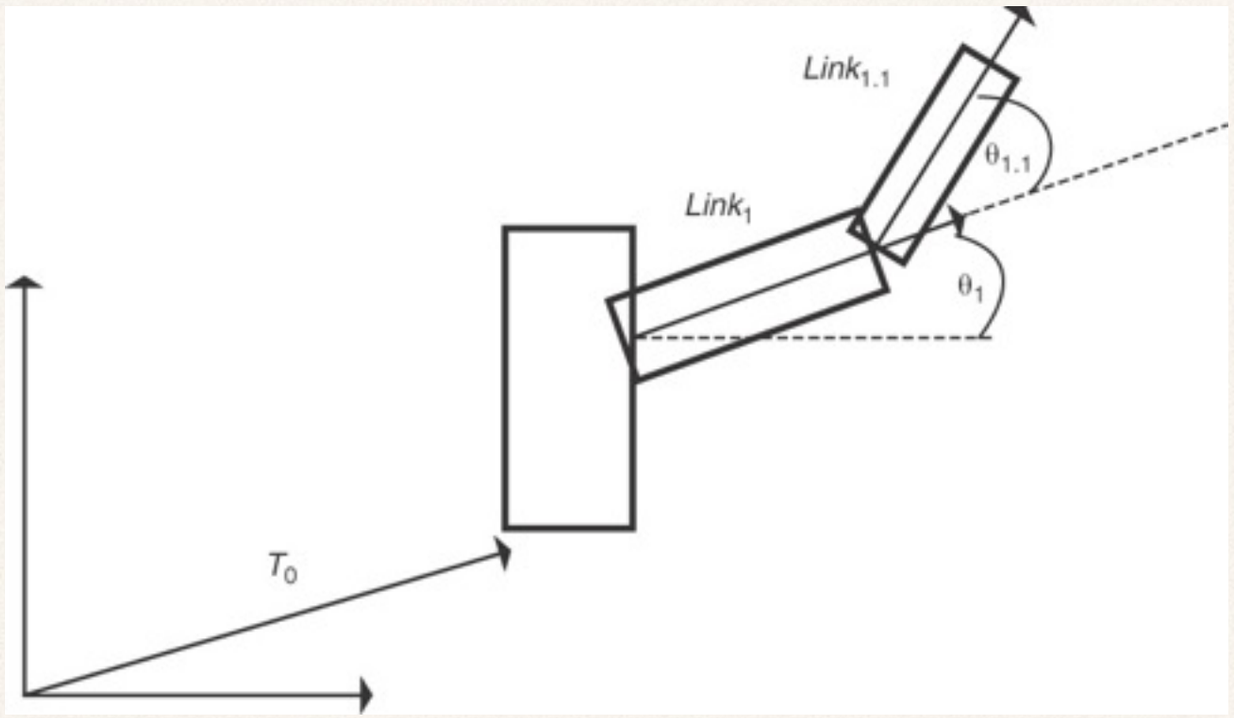

R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

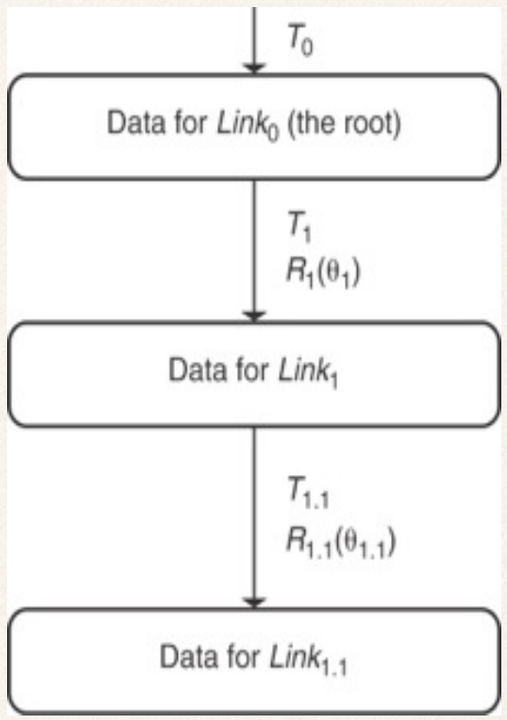
$V_0$ : a vertex of $Link_0$
world coordinates: $V_0' = T_0 V_0$

$V_1$ : a vertex of $Link_1$
world coordinates: $V_1' = T_0 T_1 V_1$

$V_{1,1}$ : a vertex of $Link_{1,1}$
world coordinates: $V_{1,1}' = T_0 T_1 T_{1,1} V_{1,1}$

# Tree structure



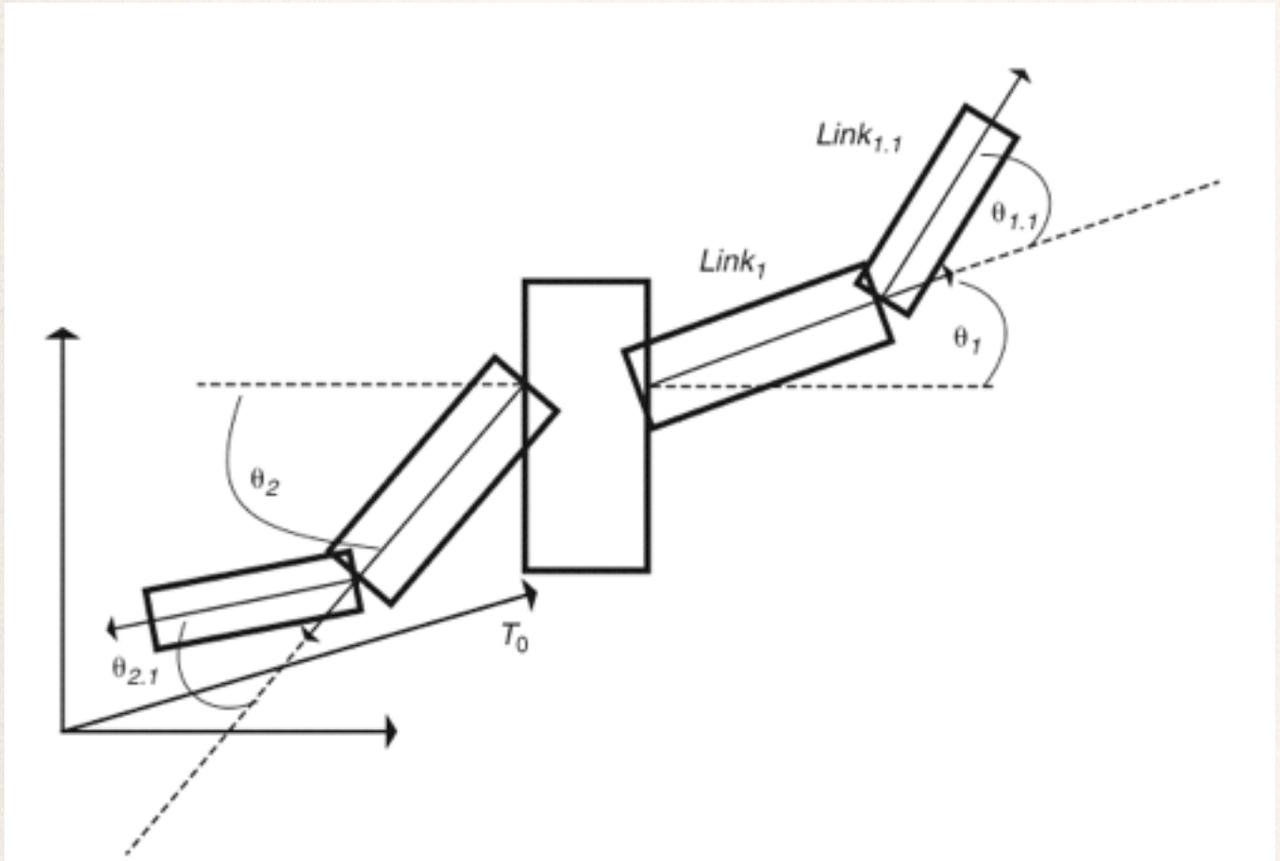

R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

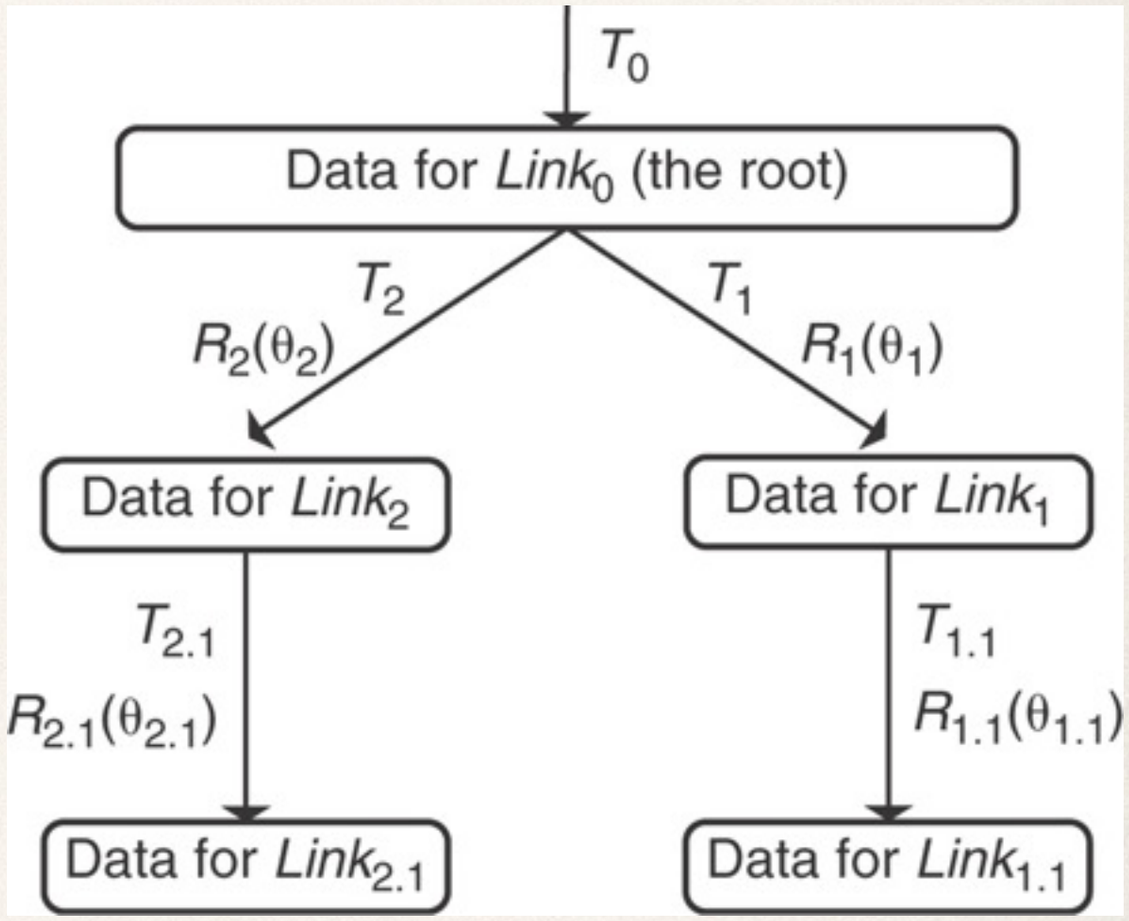
$V_0$ : a vertex of $Link_0$
world coordinates: $V_0' = T_0 V_0$

$V_1$ : a vertex of $Link_1$
world coordinates: $V_1' = T_0 T_1 R_1(\theta_1) V_1$

$V_{1,1}$ : a vertex of $Link_{1,1}$
world coordinates: $V_{1,1}' = T_0 T_1 R_1(\theta_1) T_{1,1} R_{1,1}(\theta_{1,1}) V_{1,1}$

# Tree structure with two branches



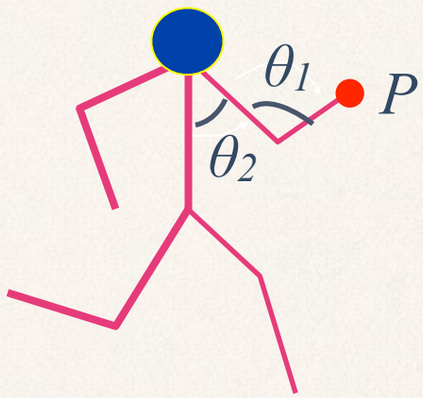

R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

# Forward kinematics (FK)

- Set the position and orientation of objects at specific frame times.
- For skeletons: set rotations at selected joints and global translation applied to root joint to create a pose.
- Do this for keyframes and interpolate joint parameters using normally piecewise splines to get continuous velocity and acceleration.

# Forward kinematics (FK)

- **Joint space** - space formed by joint angles
- **Cartesian space** - 3D (or 2D) space.
- **Forward Kinematics** - mapping from joint space to cartesian space.




**Forward Kinematics**

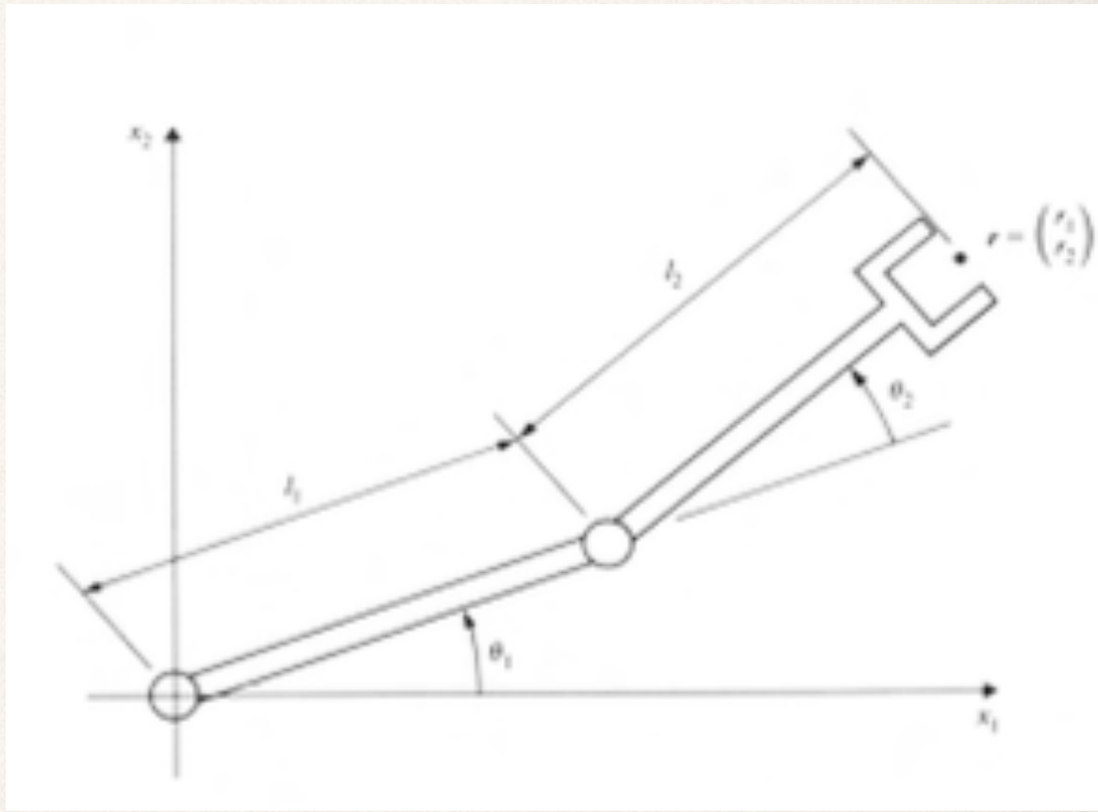
$$P = f(\theta_1, \theta_2)$$

# Forward kinematics (FK)

The position of the end-effector $r \in R^2$ is represented as a function of the vector of articular values $\theta = (\theta_1\ \theta_2)^T \in S^2$:

$$r = \begin{pmatrix} r_1(\theta) \\ r_2(\theta) \end{pmatrix} =$$

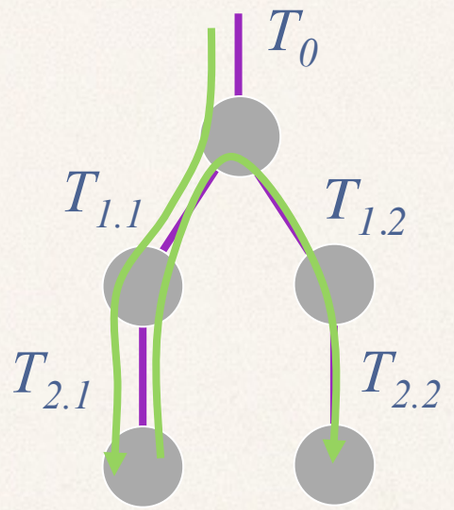
$$= \begin{pmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{pmatrix}$$



Y. Nakamura, Advanced Robotics. Addison-Wesley. 1991

# Forward kinematics (FK)

- Evaluate the tree
  - Depth-first traversal from the root to leafs.
- Repeat the following until all nodes and arcs have been visited:
  - The traversal then backtracks up the tree until an unexplored downward arc is encountered.
  - The downward arc is then traversed.
- In implementation requires a stack of transformations.




$M = I$

$M = T_0$

$M = T_0 * T_{1.1}$

$M = T_0 * T_{1.1} * T_{2.1}$

$M = T_0 * T_{1.1}$

$M = T_0$

$M = T_0 * T_{1.2}$

$M = T_0 * T_{1.2} * T_{2.2}$

# Forward Kinematics (FK)

- The only joint which an animator can explicitly position is the root joint in the hierarchy.
- The positions of all other objects in the skeleton depend on the rotations at ancestor joints.

- Example:
  - maintain a foot placed on the floor to specify keyframes.
  - maintain the foot placed at the floor when interpolating between two keyframes.

# Inverse Kinematics (IK)

- Inverse kinematics provide direct control over the placement of an end-effector object at the end of a kinematic chain of joints.
- Provide higher-level control over joint hierarchies than simple forward kinematics.
- Motion exhibits a weightless quality.

# Dynamics

- Object descriptions must include physical attributes:
  - Center of Mass (CoM)
  - Total Mass
  - Moments
  - Products of Inertia.
- F = ma.
- The motion generated by physical simulation is controlled by the application of forces and torques, which may vary over time.

# Forward Dynamics

- Apply time-varying forces and torques to objects.
- Some forces, such as those due to gravity and collisions between objects, can be handled automatically.
- Other forces are applied by the animator.
- Motion is approximated by taking a series of discrete steps in time.
  - Solve the equations of motion for the acceleration given the forces.
  - Given the position and velocity from the previous step.
  - The acceleration a can be twice integrated to determine a new velocity and position respectively from the current time step.

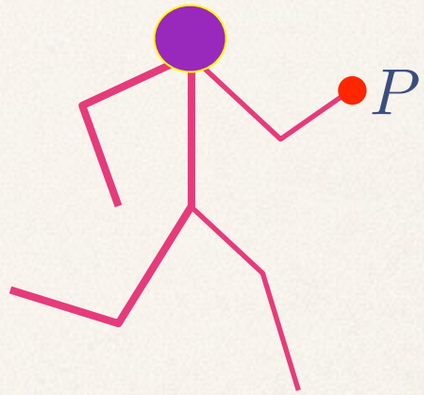
http://graphics.ucsd.edu/courses/cse169_w04/welman.pdf

# Inverse Dynamics

- Determine force and torque functions needed to accomplish a stated goal.

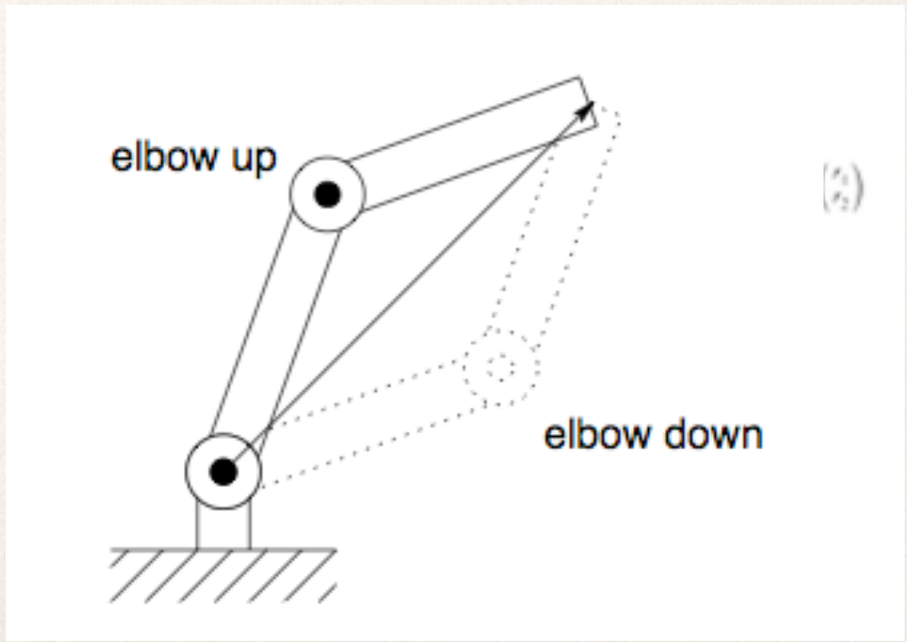
# Inverse kinematics (IK)

- Given goal position (and orientation) for end effector
  - compute the internal joint angles.
- If the mechanism is simple enough - analytic solution
- Else - numeric iterative solution
- Because the equations for forward kinematics are non-linear, IK is not always easy to compute.
- In general there is no a single solution, even for 6 DOFs.

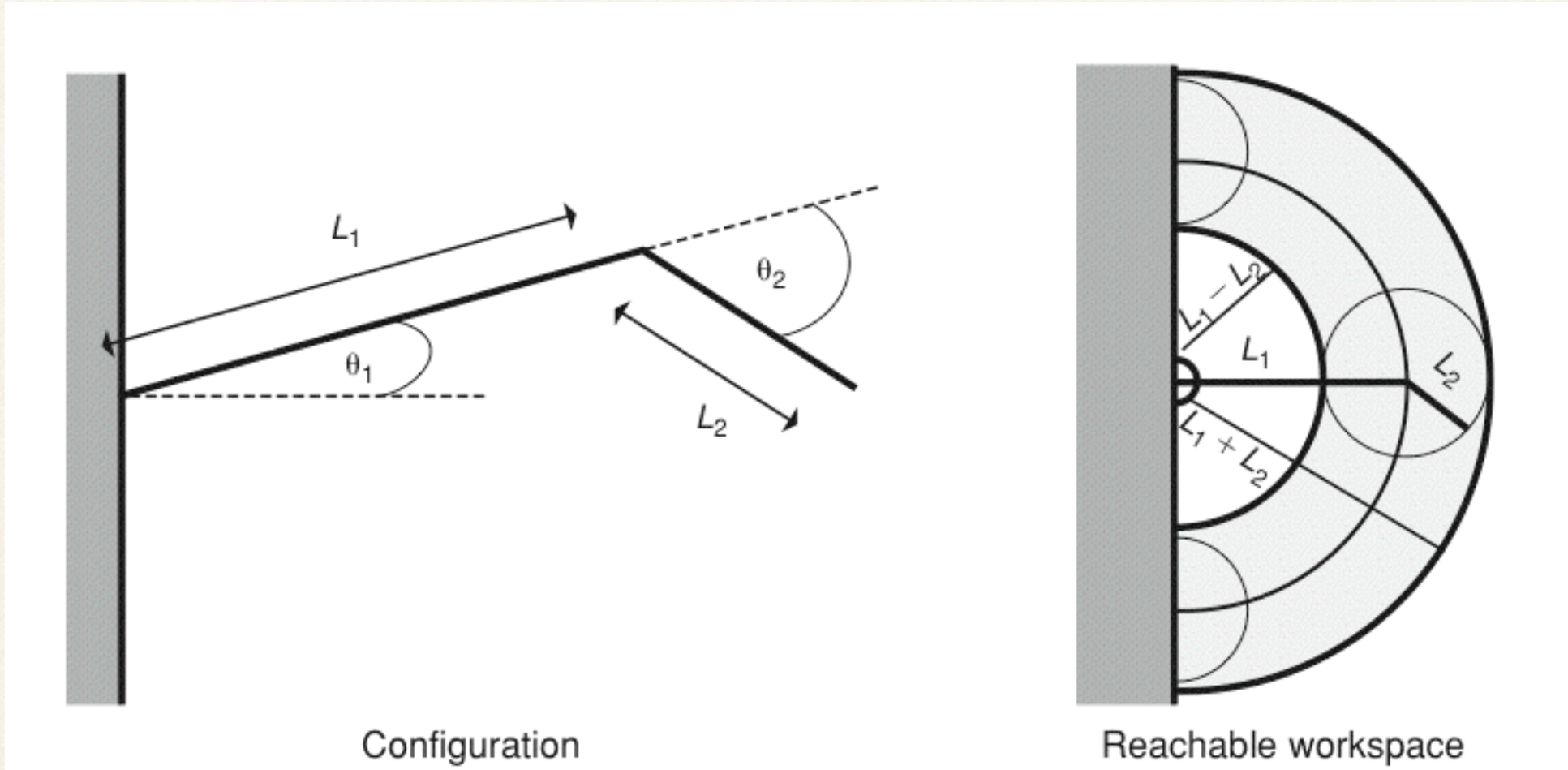



Inverse Kinematics

$$\theta_1, \theta_2 = f^{-1}(P)$$



# Inverse kinematics (IK) - spaces

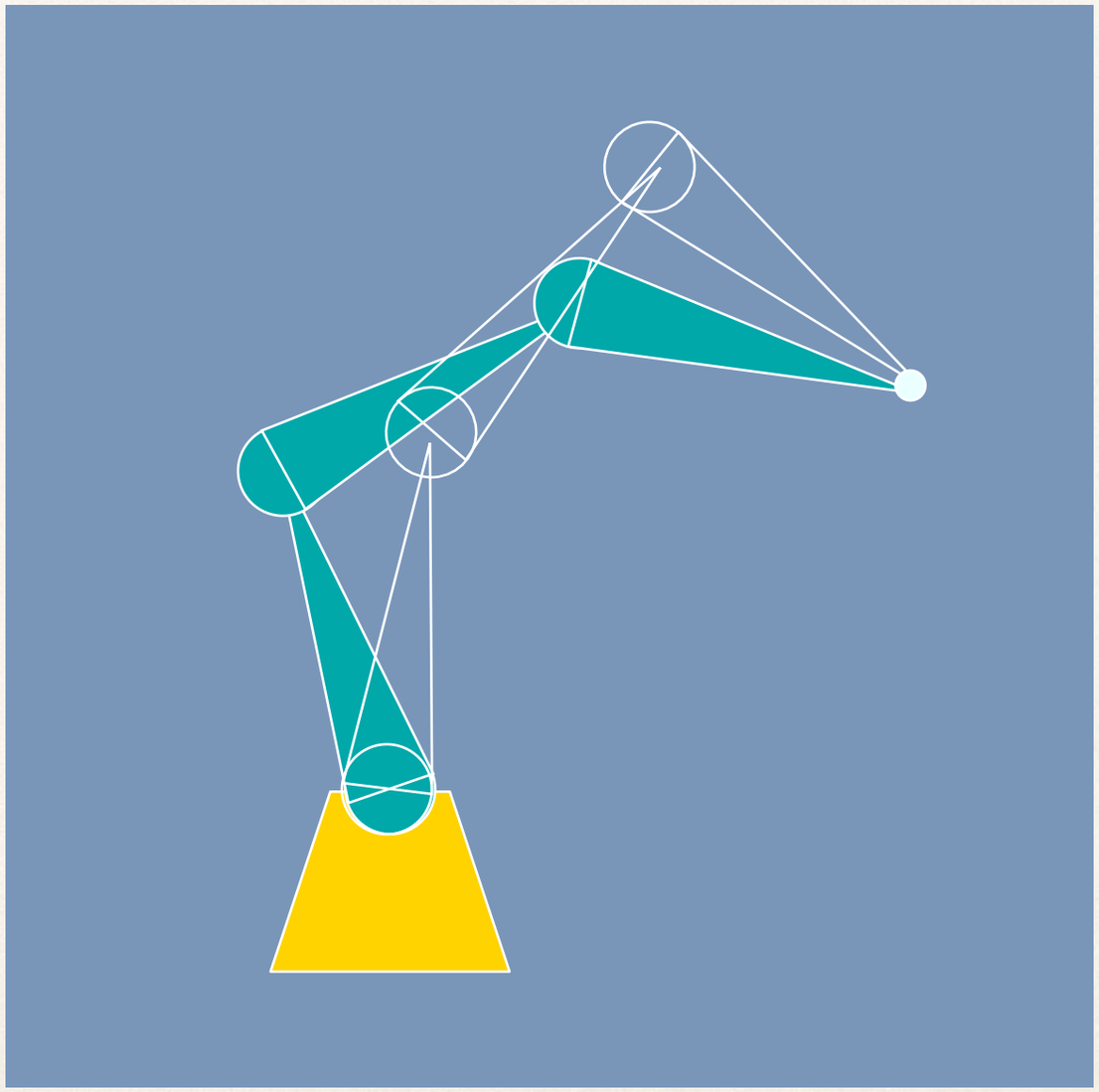



Configuration

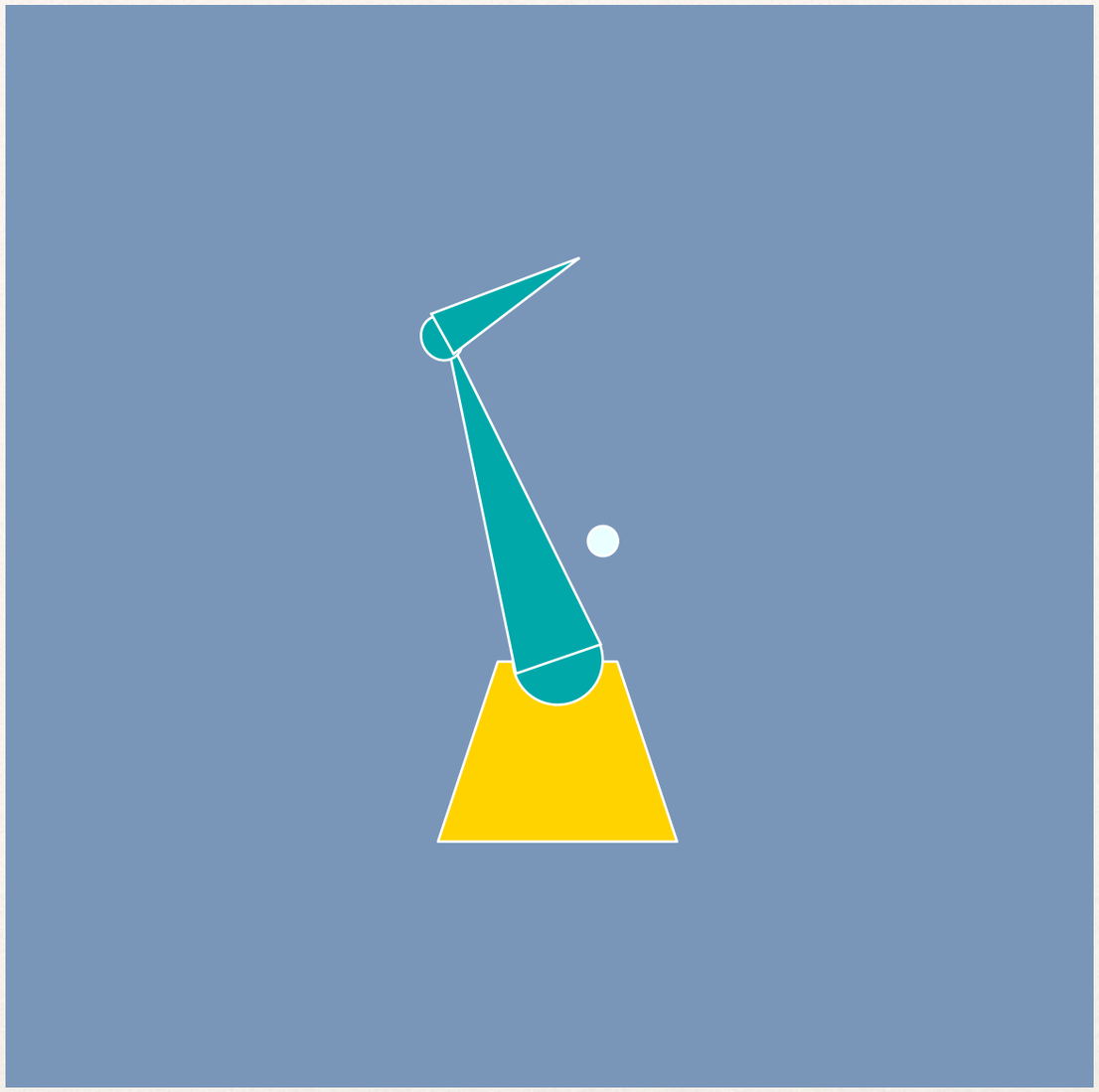
Reachable workspace

R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

# IK - infinite number of solutions



# IK - no solution



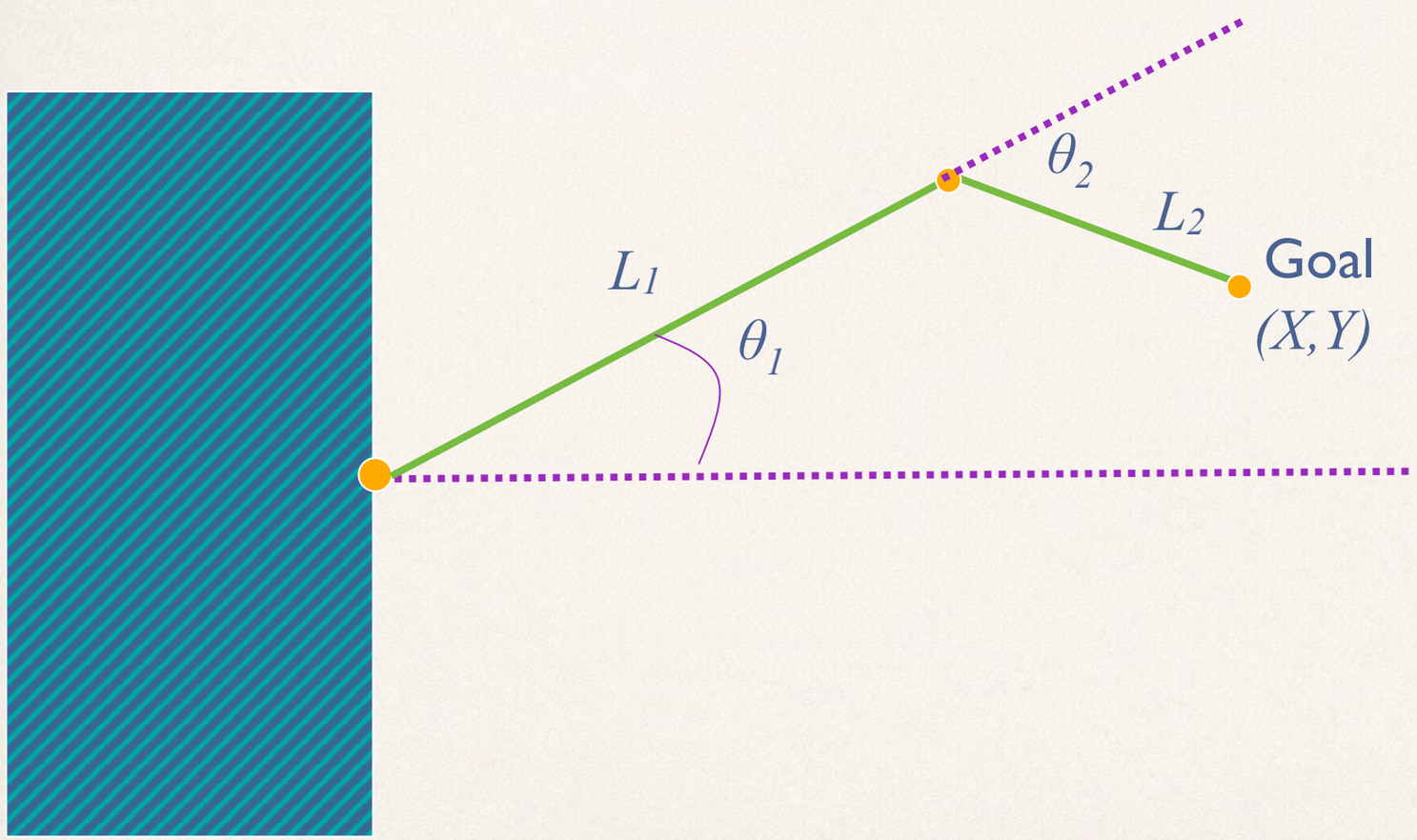
# Inverse Kinematics (IK)

- Once the joint values are calculated, the figure can be animated by interpolating the initial pose vector values to the final pose vector.
  - Does not provide a precise or an appropriate control when there are large differences between initial and final pose vectors.
  - Alternatively...
    - interpolate pose vectors.
    - do inverse kinematics for each interpolated pose vector.

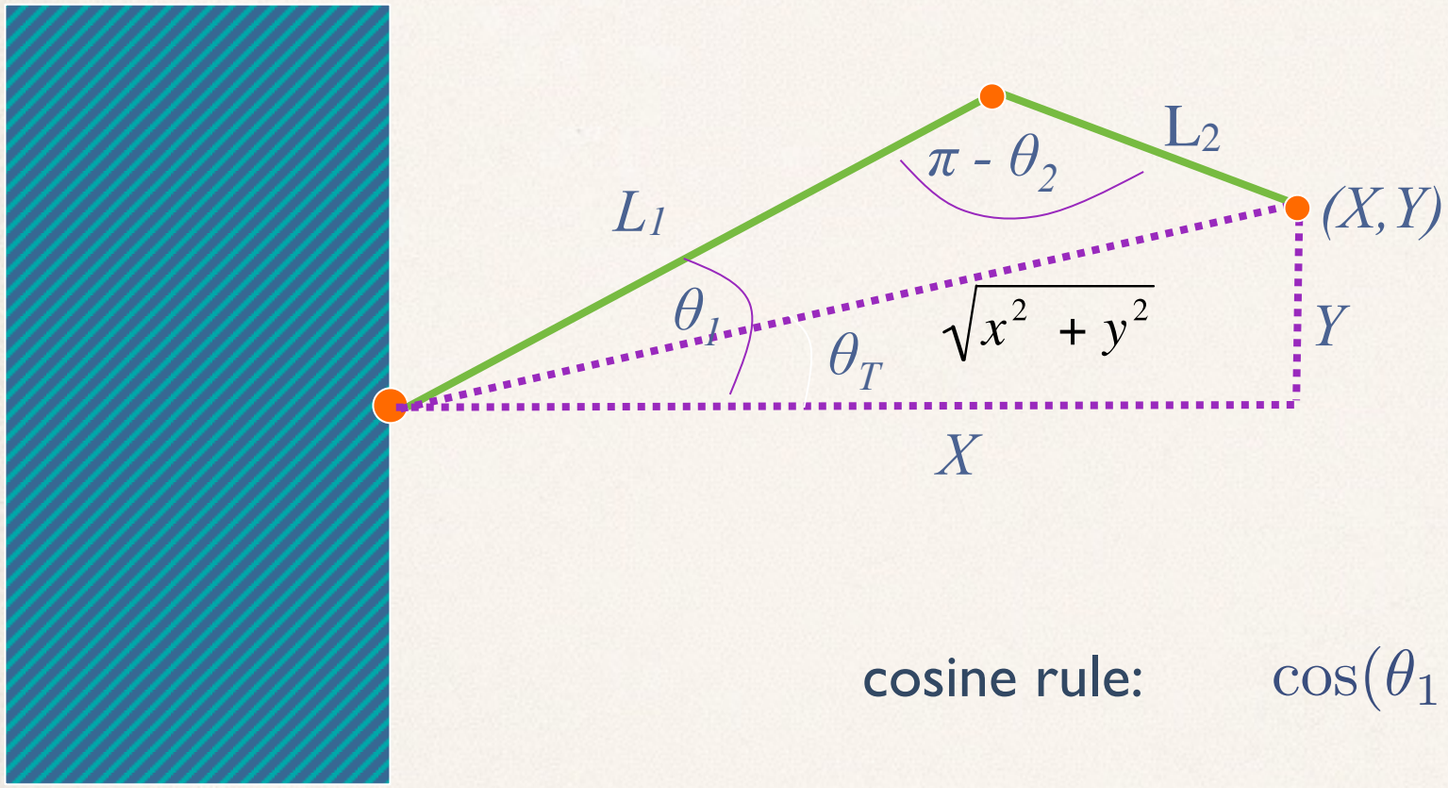
# Analytic vs. numerical methods for solving IK

- Analytic solutions exist only for relatively simple linkages.
  - Inspecting the geometry of linkages.
  - Algebraic manipulation of equations that describe the relationship of the end effector to the base frame.
- Numerical incremental approach:
  - Inverse Jacobian method.
  - Other methods.

# Analytic methods for solving IK



# Analytic methods for solving IK




$$\cos(\theta_T) = \frac{X}{\sqrt{X^2 + Y^2}}$$

$$\theta_T = \cos^{-1}\left(\frac{X}{\sqrt{X^2 + Y^2}}\right)$$

cosine rule:

$$\cos(\theta_1 - \theta_T) = \frac{{L_1}^2 + X^2 + Y^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}}$$

cosine rule:

$$\cos(\pi - \theta_2) = \frac{{L_1}^2 + {L_2}^2 - (X^2 + Y^2)}{2L_1L_2}$$

$$\theta_1 = \cos^{-1}\left(\frac{{L_1}^2 + X^2 + Y^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}}\right) + \theta_T$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{{L_1}^2 + {L_2}^2 - (X^2 + Y^2)}{2L_1L_2}\right)$$

# Why is IK hard?

- Redundancy
- Natural motion control
  - joint limits
  - minimum jerk
  - style?
- Singularities
  - ill-conditioned
  - singular

# Numerical methods for solving IK

- Resolved motion-rate control.
- Optimization-based methods.
- Example-based methods.

# Resolved motion rate control

- In 1969, Whitney proposed to study the differential kinematic relation of robot manipulators.
  - use differential relations to solve the motion of the articulations given a Cartesian motion of the end-effector.
- The relation between the pose of the end-effector and the joint angles can be represented with the non-linear equation:

$$f(\theta) \in \mathbb{R}^6, \quad \theta \in \mathbb{R}^n$$

- where n is the number of articular variables.
- Differentiating with respect to time, the relation between r and θ is given by:

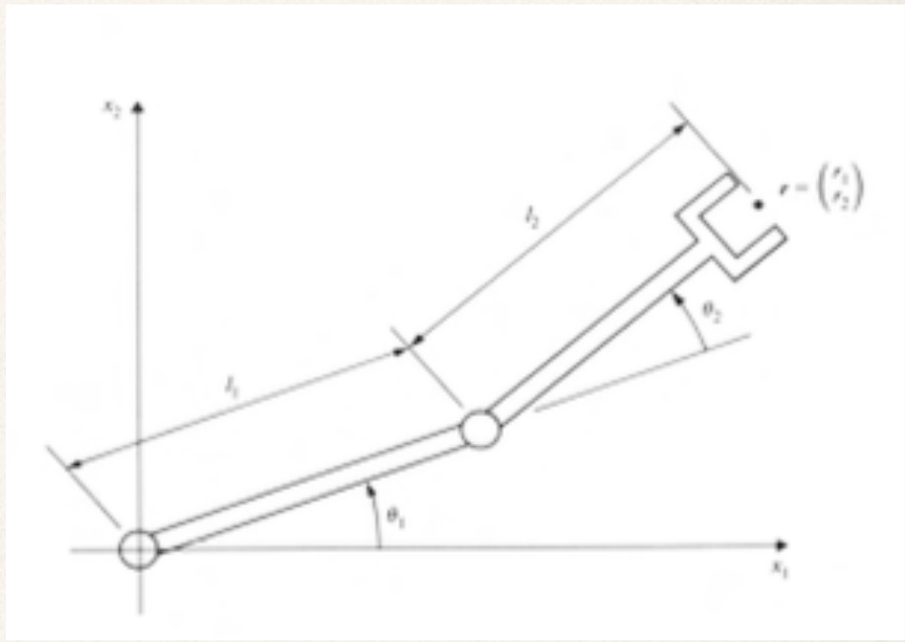
$$\dot{r} = J(\theta)\dot{\theta}$$

$$J(\theta) \triangleq \frac{\partial f}{\partial \theta} \in \mathbb{R}^{6\times n}$$

# Resolved motion rate control

$$r = \begin{pmatrix} r_1(\theta) \\ r_2(\theta) \end{pmatrix} = \begin{pmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{pmatrix}$$

- When $\theta_1$ and $\theta_2$ change as functions of time, the velocity of the end-effector is computed:



Y. Nakamura, Advanced Robotics. Addison-Wesley. 1991

$$\frac{dr}{dt} = \frac{\partial r}{\partial \theta}\frac{d\theta}{dt} = \begin{pmatrix} \partial r_1/\partial\theta_1 & \partial r_1/\partial\theta_2 \\ \partial r_2/\partial\theta_1 & \partial r_2/\partial\theta_2 \end{pmatrix}$$

$$= \begin{pmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{pmatrix} \begin{pmatrix} d\theta_1/dt \\ d\theta_2/dt \end{pmatrix}$$

- where the matrix $\partial \mathrm{r}/\partial\theta$ is the Jacobian matrix.

# Resolved motion rate control

- Whitney's method: Resolved Motion Rate Control takes advantage that finding articular velocities from the end-effector's velocity is a linear relation.
- The relation is found inverting the Jacobian matrix:

$$\dot{\theta} = J^{-1}\dot{r}$$

- and iterating over the desired pose over a series of incremental steps.