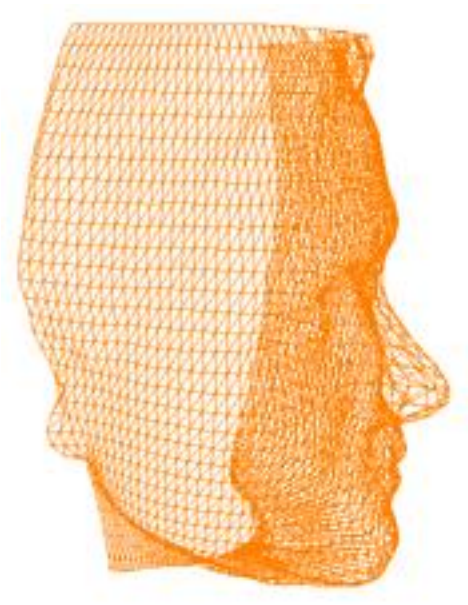
COSC422 Advanced Computer Graphics



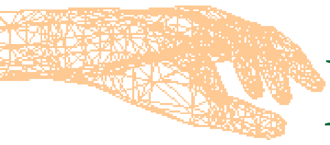
12 Kinematics

Semester 2
2021

R. Mukundan (mukundan@canterbury.ac.nz)
Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

Lecture Outline

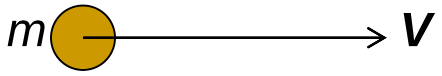
- Forward Kinematics
  - Joint angle transformations
  - Global positions of joints and end-site.
- Inverse Kinematics
  - 2-Link inverse kinematics
  - Link reduction
  - Cyclic coordinate descent (CCD) algorithm
  - FABRIK algortihm

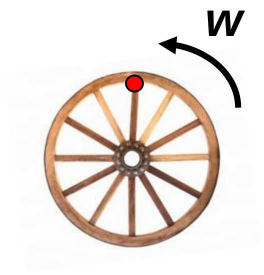


Definitions

- Kinematics:
  - Motion specification (position and velocity) without direct reference to mass, force or torque.
    - Position P
    - Velocity $V = \frac{dP}{dt}$
    - Acceleration $A = \frac{dV}{dt}$

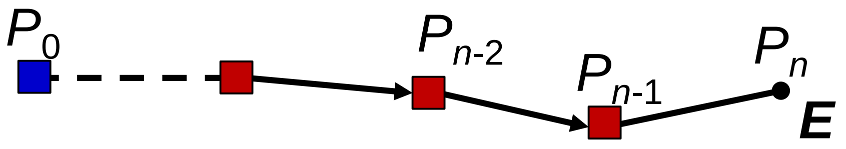
    Angular velocity $w = \frac{d\theta}{dt}$
- Dynamics:
  - Study of motion resulting from applied forces and torques.
    - Acceleration $A = F/m$ (m = mass, F = Applied force)
    - Linear Momentum $L = mV$
    - Angular Momentum $H = Iw$ (I = Moment of inertia)
    - Torque = $I\frac{dw}{dt}$

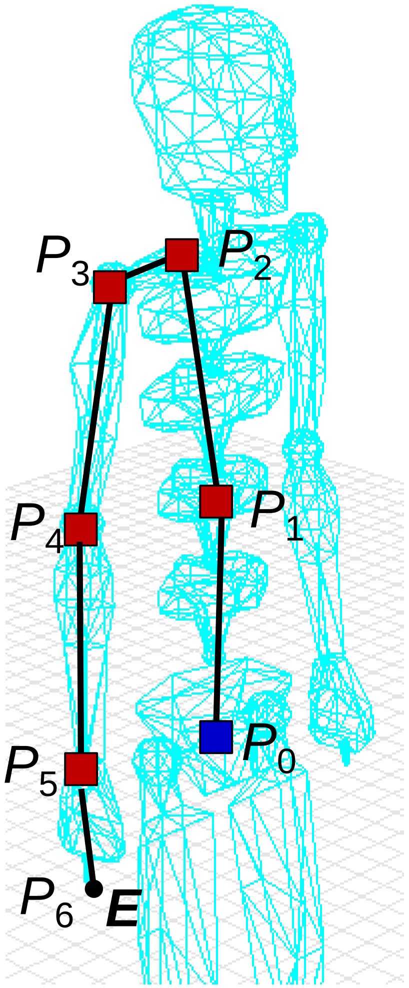




Definitions

- An end-effector *E* is the end point of the last link of a joint chain. It is specified as the "End site" in a BVH mocap file.

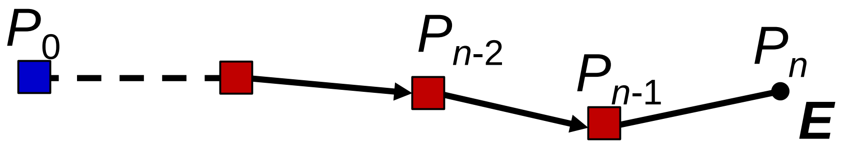




- The above is an *n*-link kinematic chain. P_i is the relative position of a joint with respect to its parent
- The global position of the end effector *E* is $E = P_0 + P_1 + \ldots + P_n$

Definitions

- Forward Kinematics:
  - Initial configuration is given using joint positions (eg. skeleton parameters) with translation matrices: T_i, $i = 0 \dots n$
  - Joint angles required to attain another configuration are also given (eg. Motion capture data): θ_i, $i = 0 \dots n-1$

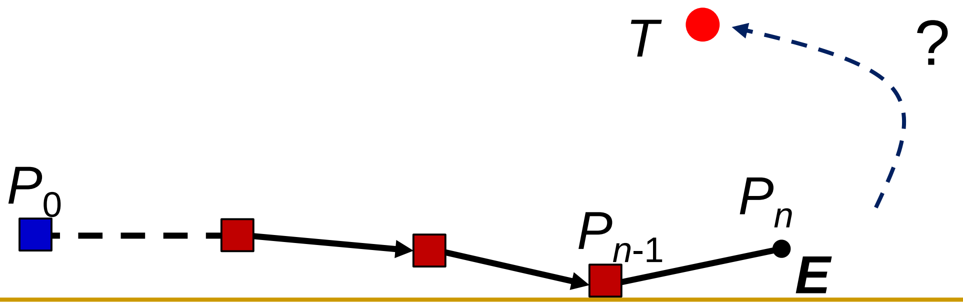


Forward kinematics problems:

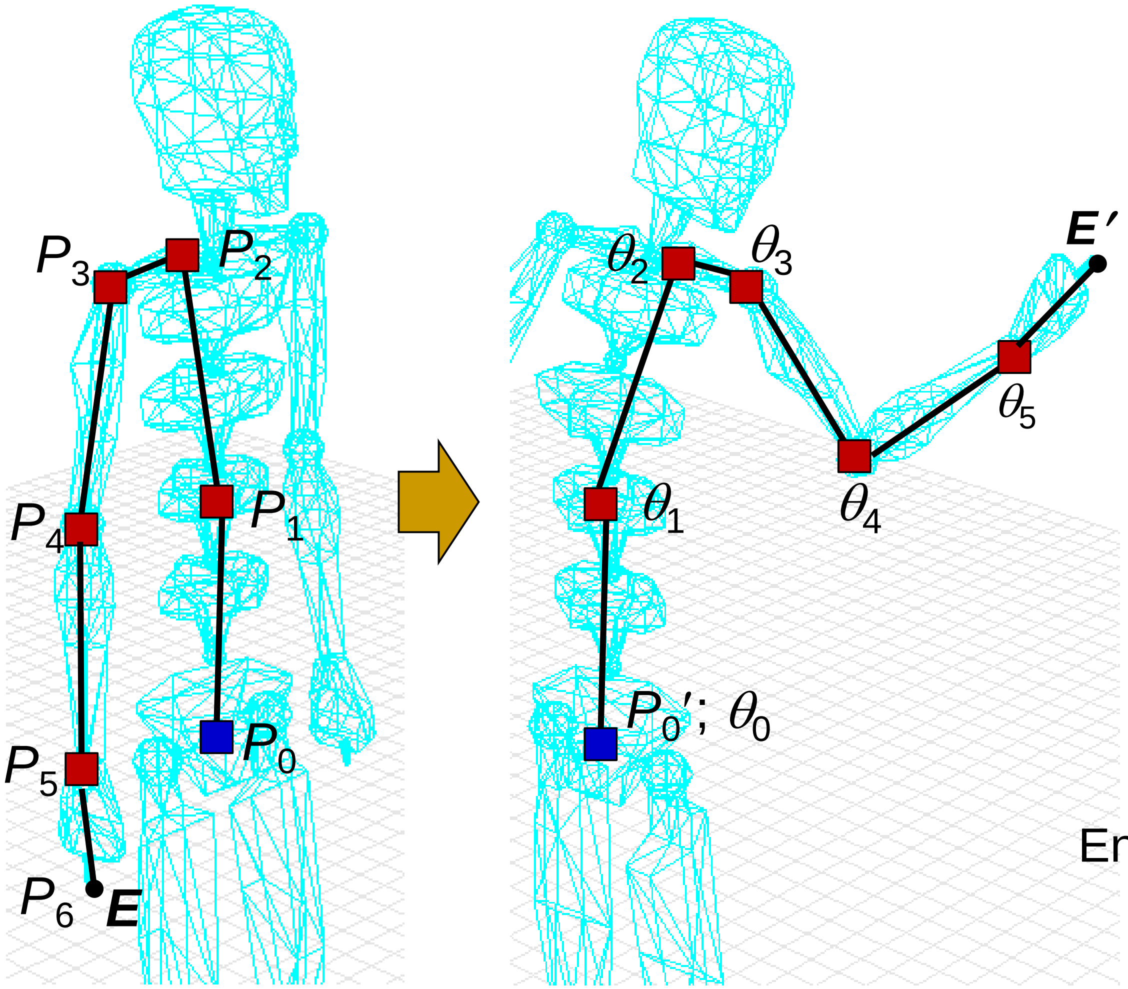
- How does the position of *E* vary with joint angles?
- What is the velocity of *E* if the rate of change of each joint angle is known?

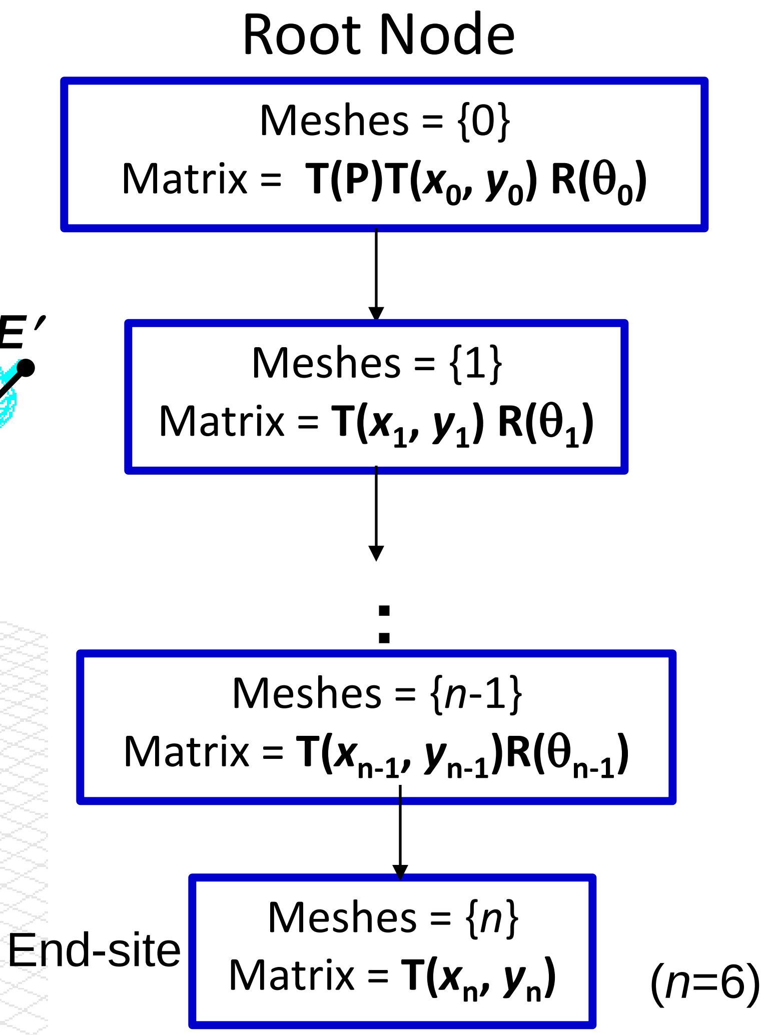
Definitions

- Inverse Kinematics (IK):
  - Initial configuration and the required final position of the end-effector(s) are given.
  - We have to compute the joint angles of the chain for which the required end-effector position (and optionally orientation) is given.
    - There could be multiple solutions for joint angles. We will have to use **joint angle constraints** and the shortest path from the current configuration
    - IK solutions are required for generating goal-directed motion.



Forward Kinematics





Root Node
Meshes = {0}
Matrix = $\mathbf{T(P)T(}x_0, y_0\mathbf{)\ R(}\theta_0\mathbf{)}$
Meshes = {1}
Matrix = $\mathbf{T(}x_1, y_1\mathbf{)\ R(}\theta_1\mathbf{)}$
:
Meshes = {n-1}
Matrix = $\mathbf{T(}x_{n-1}, y_{n-1}\mathbf{)R(}\theta_{n-1}\mathbf{)}$
End-site
Meshes = {n}
Matrix = $\mathbf{T(}x_n, y_n\mathbf{)}$
(n=6)

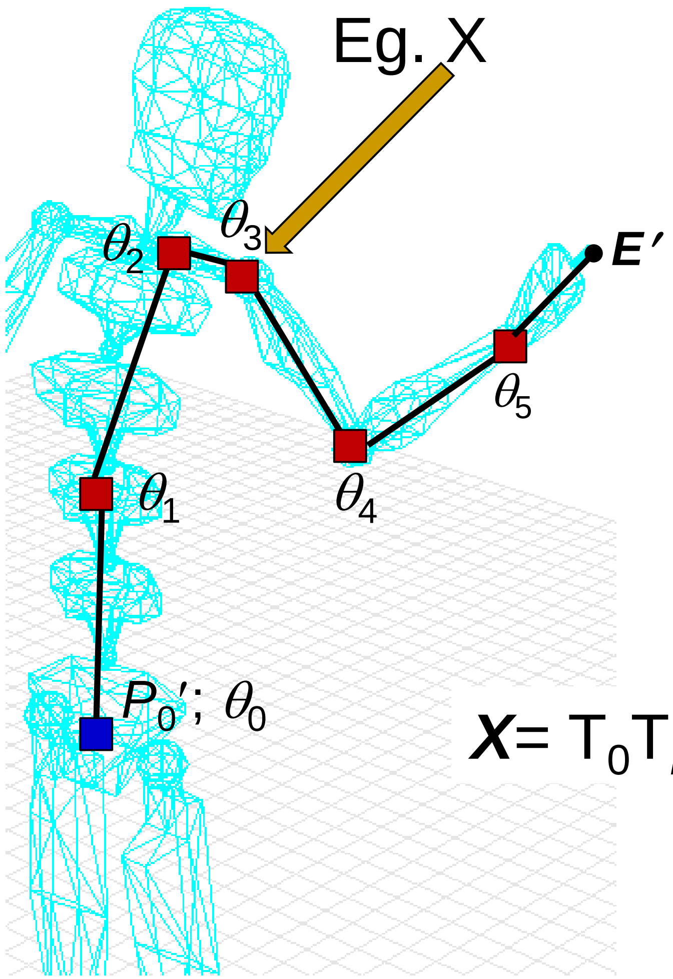
$$\boldsymbol{E'} = \mathrm{T}_0\mathrm{T}_P\mathrm{R}(\theta_0)\mathrm{T}_1\ \mathrm{R}(\theta_1)\ \ldots \mathrm{T}_{n-1}\ \mathrm{R}(\theta_{n-1})\mathrm{T}_n\ \mathbf{0}$$

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Forward Kinematics

We can use the previous method to find the global position of any joint.

Eg. X

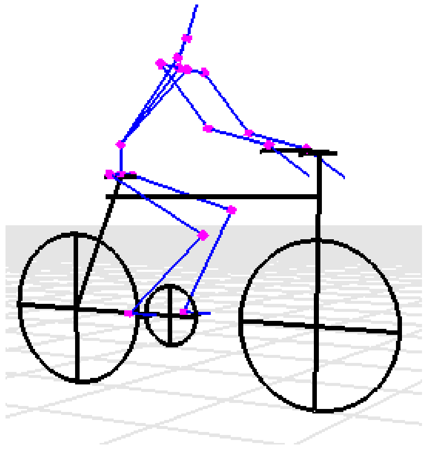
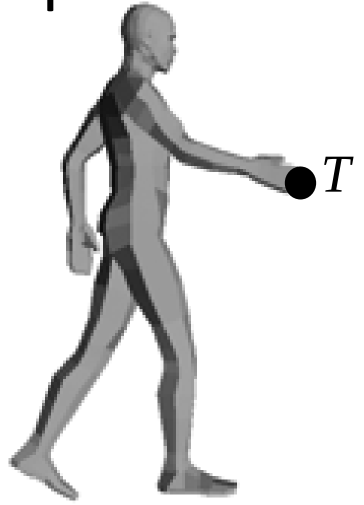


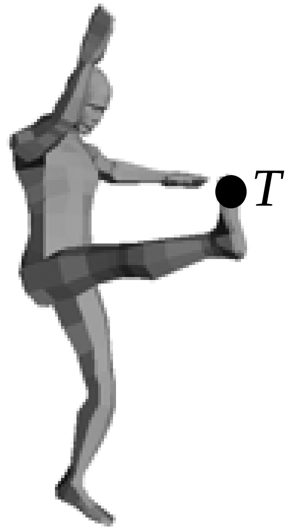
$$\boldsymbol{X} = T_0 T_P R(\theta_0) T_1 R(\theta_1)\, T_2 R(\theta_2)\, T_3\, \cancel{R(\theta_3)}\, \mathbf{0}$$

$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

Inverse Kinematics (IK)

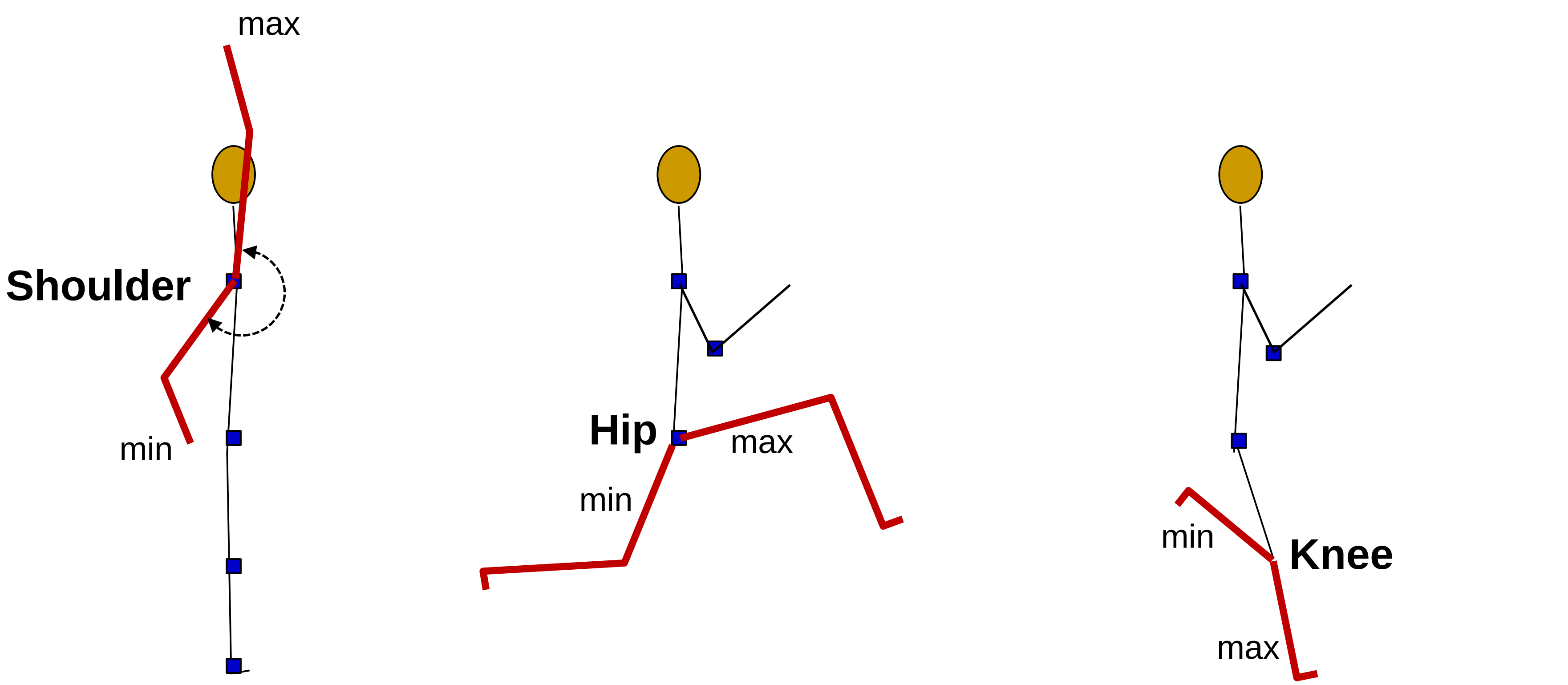
- Inverse kinematics solutions are needed for animating an articulated figure using only the desired positions (and optionally orientations) of end points as inputs.
- The IK solution on one link is often combined with forward kinematics on other links to generate a character animation sequence.



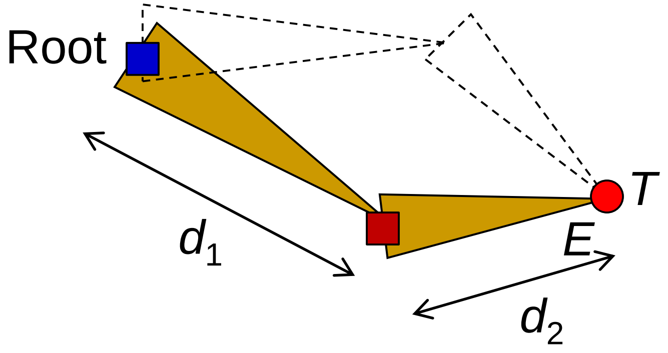
Joint Angle Constraints

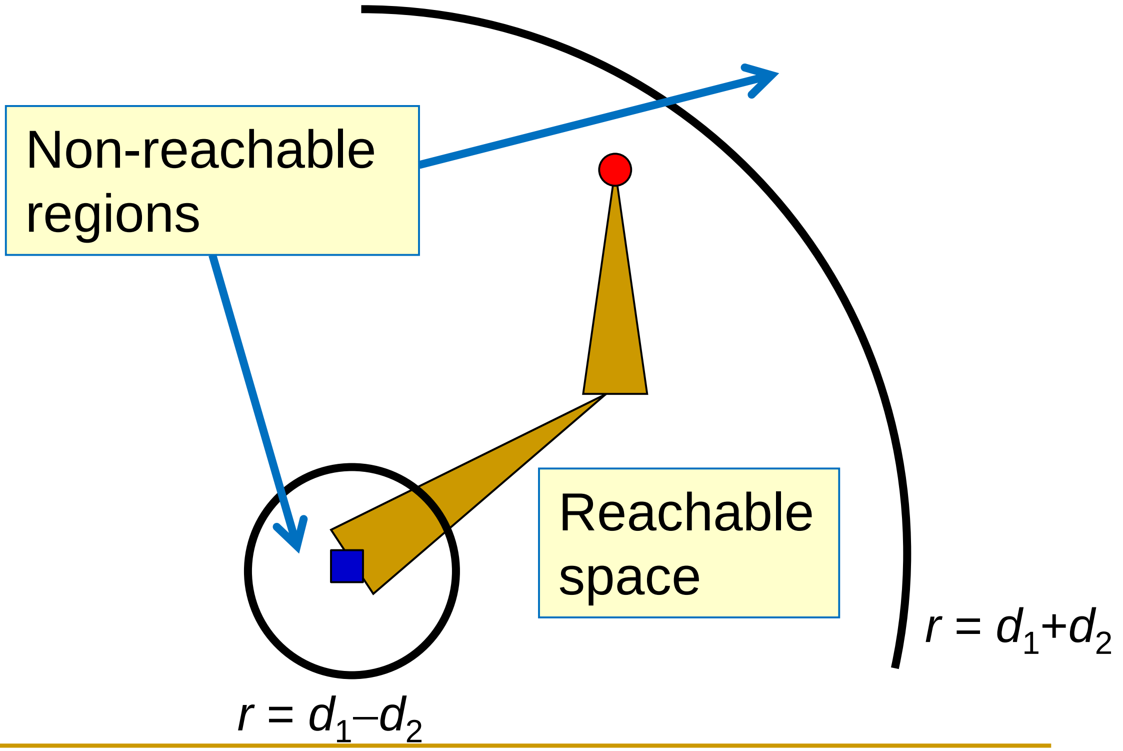
For realistic motion of a character model, we need to include angle constraints $(\theta_{\min}, \theta_{\max})$ at each joint.



Inverse Kinematics: 2-Link Chain

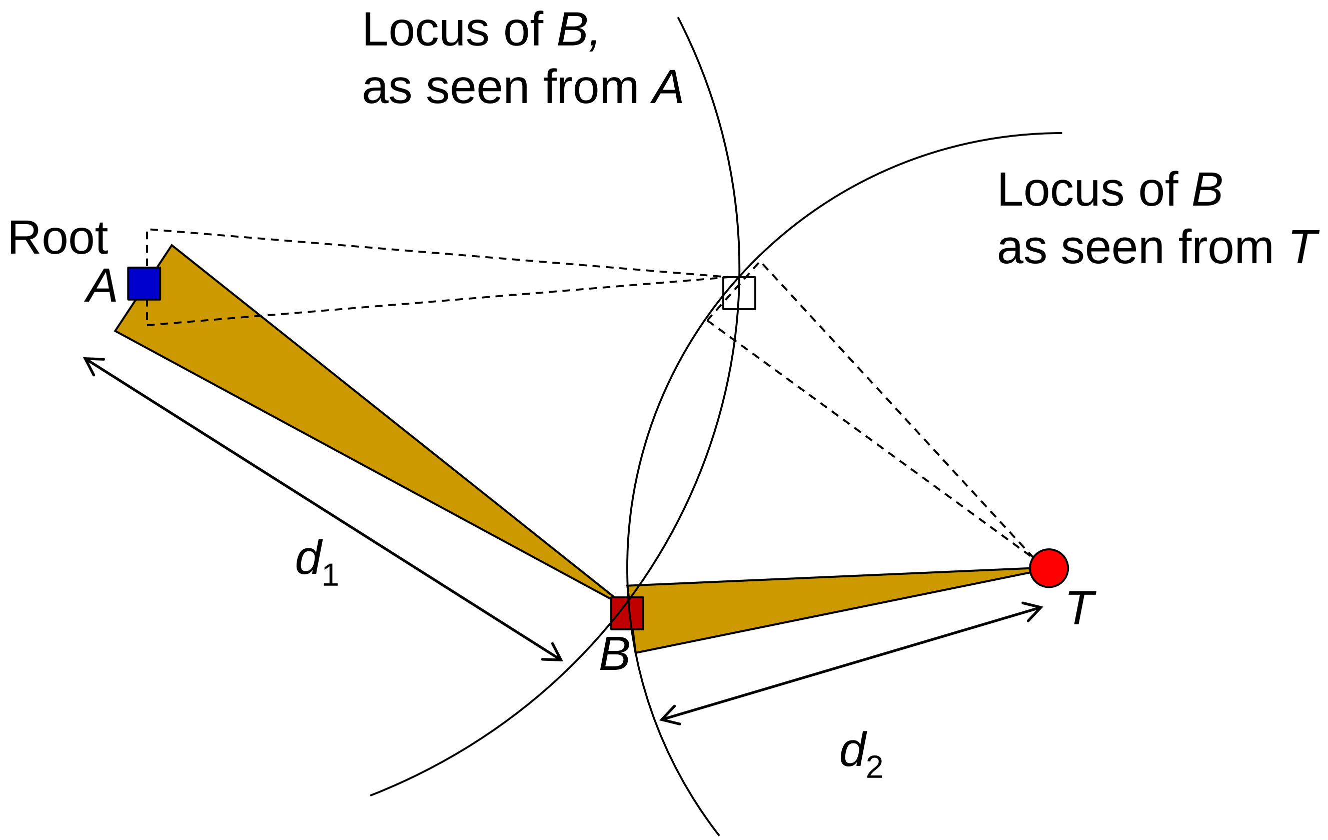
- Two-link chains have simple solutions.
- There may be several possible solutions for the same target position. We will have to use joint angle constraints to select the most appropriate solution.





Inverse Kinematics

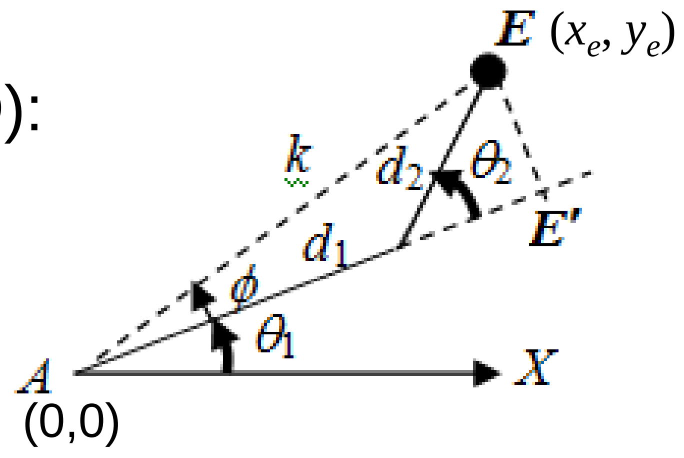
2-Link Solutions (in 2D):



In 3D, the two spheres meet along a circle, giving infinite solutions.

Inverse Kinematics

Analytical solution for a 2-link chain (2D):



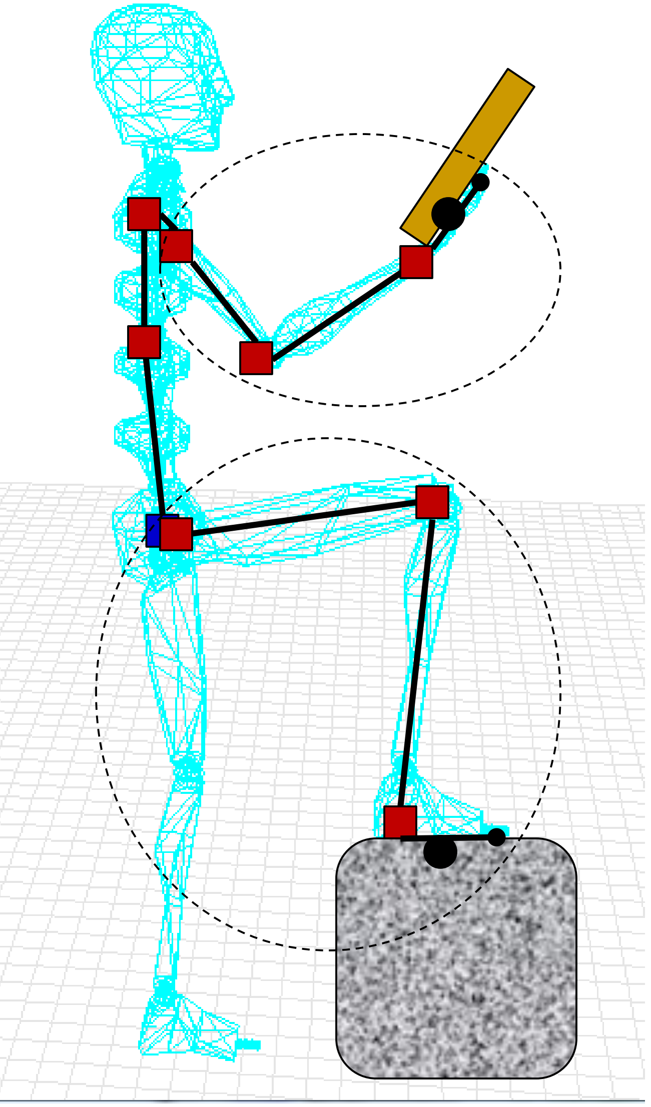
$$\theta_2 = \cos^{-1}\left(\frac{x_e^2 + y_e^2 - d_1^2 - d_2^2}{2d_1d_2}\right)$$

$$\theta_1 = \tan^{-1}\left(\frac{y_e}{x_e}\right) - \tan^{-1}\left(\frac{d_2 \sin\theta_2}{d_1 + d_2\cos\theta_2}\right)$$

Valid only if

$$(d_1 - d_2)^2 \leq x_e^2 + y_e^2 \leq (d_1 + d_2)^2$$

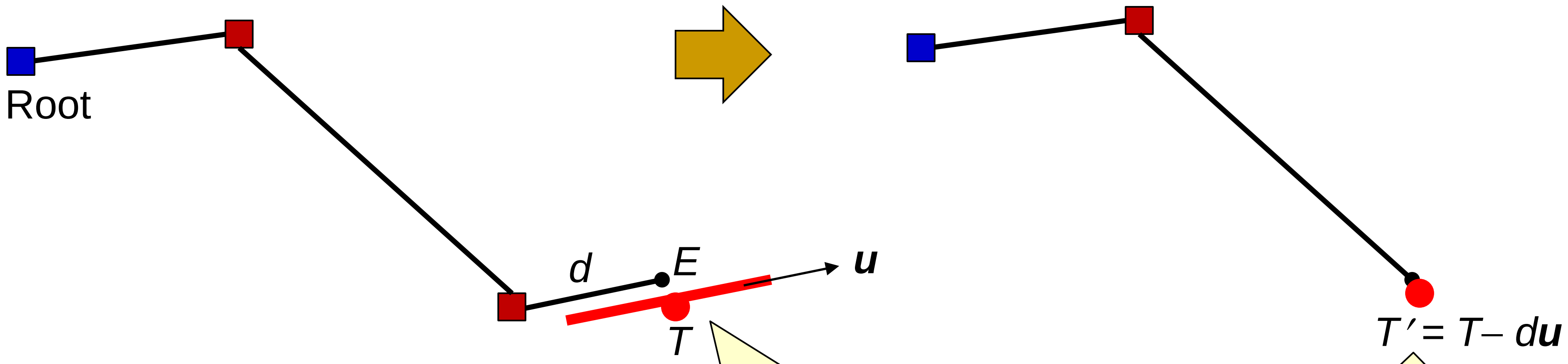
Link Reduction



- Link reduction is the process of selecting only a small sub-chain of the whole joint chain for finding the inverse kinematic solution. The remaining joint angles of the chain are determined using other constraints.
- In several situations, a target orientation will also be specified. This additional requirement is useful for link reduction.

Link Reduction

A 3-link chain with the required position of the end-effector and also the orientation of the last link specified, can be reduced to a 2-link chain for the purpose of deriving IK solution.



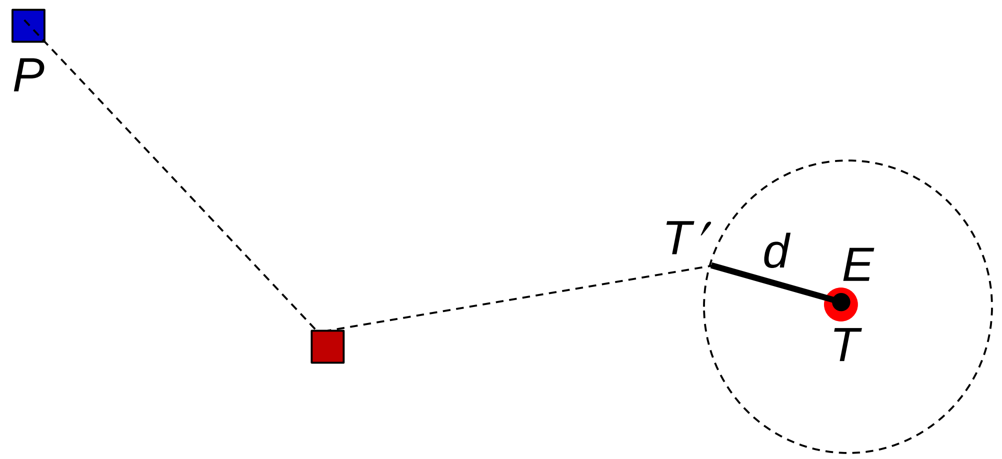
Both target position and direction (***u***) are specified.

Updated position of the target for the 2-link configuration

Link Reduction

In many cases, we can reduce a 3-link IK problem to a 2-link problem. Suppose you are given a 3-link chain with *P* as the root position, and *T* the target.

Using the last link's distance, find the point in the neighbourhood of *T* that is closest to the root. Use this point as the target *T'* for the 2-link chain.



Iterative IK Algorithms for n-links

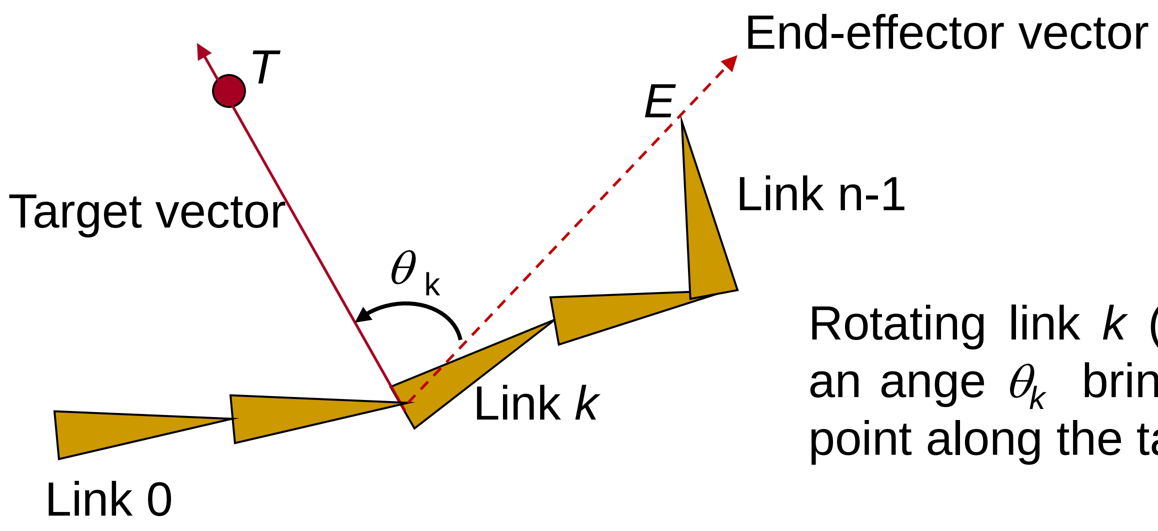
- For a general n-link chain, we have to use an iterative algorithm for finding the joint angles, given the target position.
- In each iteration, the algorithm tries to move the end point closer to the target.

  - Cyclic Coordinate Descent (CCD)
  - Forward and Backward Reaching Inverse Kinematics (FABRIK)

Cyclic Coordinate Descent (CCD)

- A series of joint angle rotations are applied in sequence from the last link (end effector) towards the root node of the chain, using two vectors at each link: A target vector, and an end-effector vector.

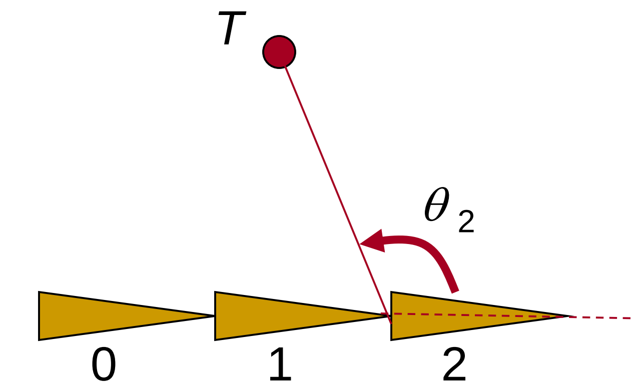
Idea:



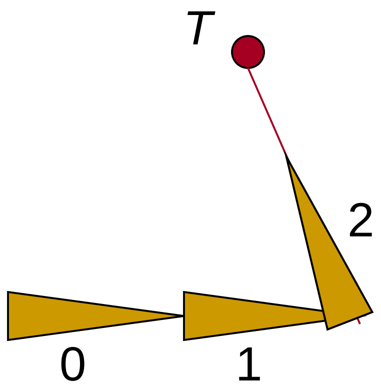
Rotating link *k* (and its child nodes) by an ange θ_k brings the end effector to a point along the target vector.

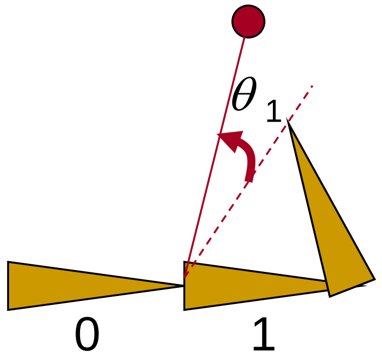
CCD Example $(n = 3)$

Target vector
End-effector vector

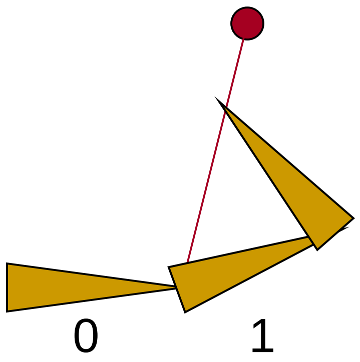


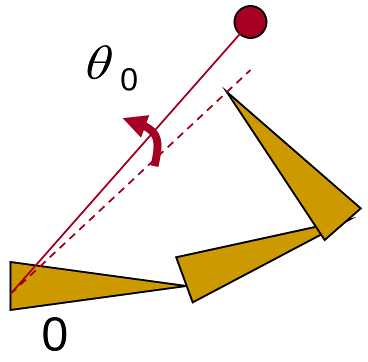
Step 1
Rotate link-2 by angle θ_2





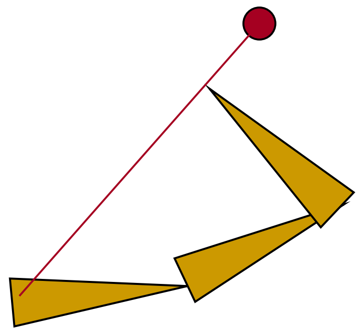
Step 2
Rotate link-1 by angle θ_1





Step 3

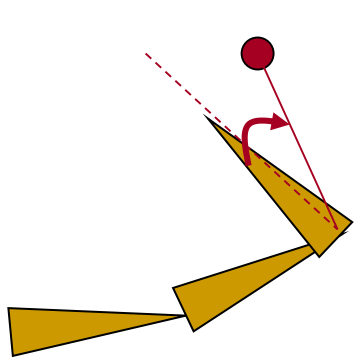
Rotate link-0 by angle θ_0



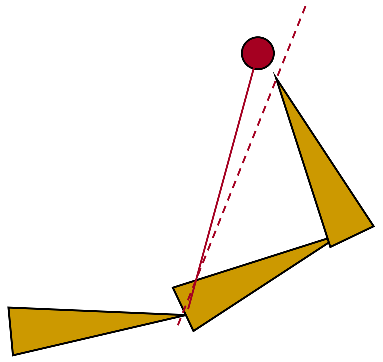
First iteration Completed

Cyclic Coordinate Descent (CCD)

After completing the first iteration, the algorithm continues again performing the rotation at the end effector, followed by rotations of each parent link towards the root, using the angle between the end-effector vector and the target vector at each link.



Step 4

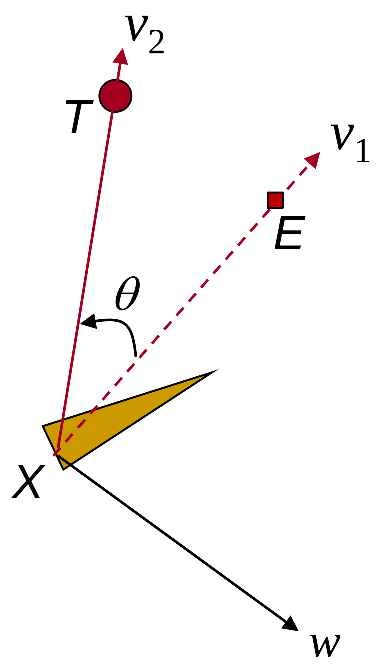


Step 5

... and so on.

3D Rotations

- The global position of the target *T* is known.
- We can find the global positions of *E* and *X* (slides 7,8), using which we can compute unit vectors v_1, v_2.



Angle of rotation $\theta = \cos^{-1}(v_1.v_2)$

Axis of rotation = $w = v_1 \times v_2 = (l, m, n)$

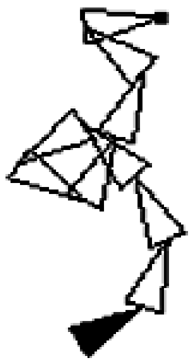
Quaternion of rotation:

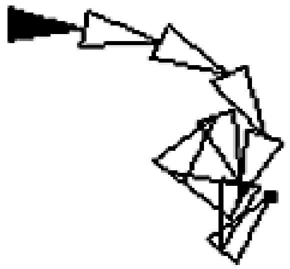
$$\left(\cos\frac{\theta}{2}, l\sin\frac{\theta}{2}, m\sin\frac{\theta}{2}, n\sin\frac{\theta}{2}\right)$$

Cyclic Coordinate Descent: Summary

- Start iteration from the last link of the chain. Perform a rotation of the link so that the end effector vector coincides with the target vector.
- Rotate the parent link so that the end-effector vector at its joint coincides with the target vector.
- Repeat the process for the parent of each rotated link.
- After rotating the root node, start the next iteration all over again from the last link.
- Use multiple iterations to refine the solution.

Cyclic Coordinate Descent





- In some cases, particularly when the target is located close to the base, the CCD algorithm causes a chain to form a loop, intersecting itself
- For certain target positions, the algorithm can take a large number of iterations
- CCD can generate large angle rotations. This can be avoided by using joint angle constraints

FABRIK

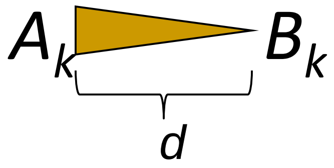
FABRIK: Forward and Backward Reaching Inverse Kinematics Algorithm

- Does not require angle computation or matrix transformations.
- Fast convergence.
- Uses two passes per iteration.
  - Forward pass: Moves the end effector to the target, and adjusts the remaining link positions.
  - Backward pass: Moves the first link back to its original position and adjusts the remaining link positions.

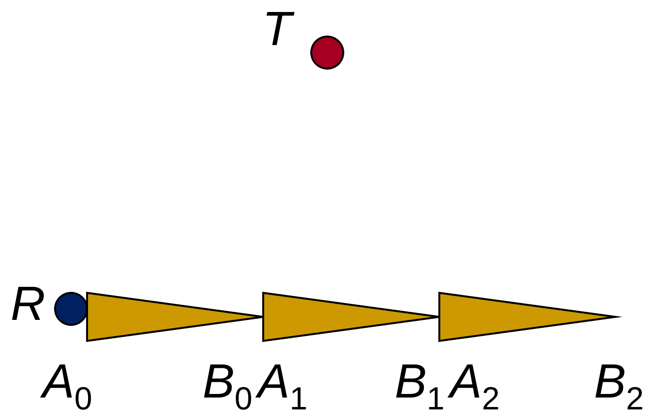
Ref: Aristidou, A., Lasenby, J. (2011) FABRIK: A fast, iterative solver for the inverse kinematics problem. Graphical Models (Elsevier), 73, pp. 243-260.

FABRIK

Notation: The end points of link-k are denoted by A_k and B_k. The length of each link is d.



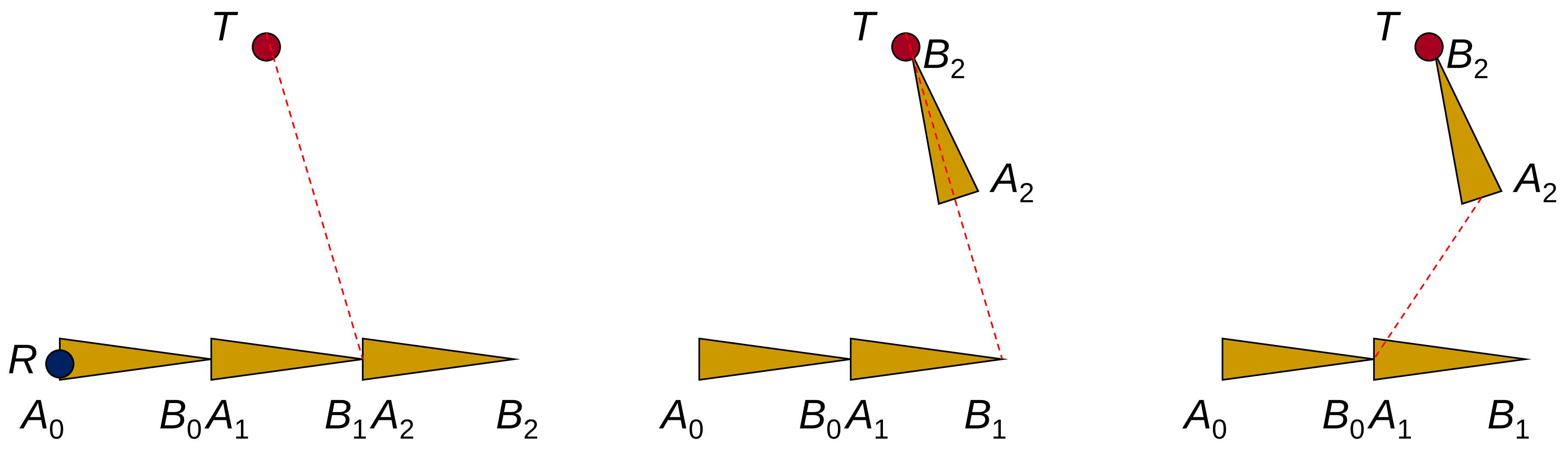
For a joint chain, $B_k = A_{k+1}$, $k = 0...n\text{-}2$.



T: Target
R: Root

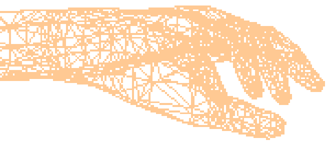
FABRIK

Forward pass:

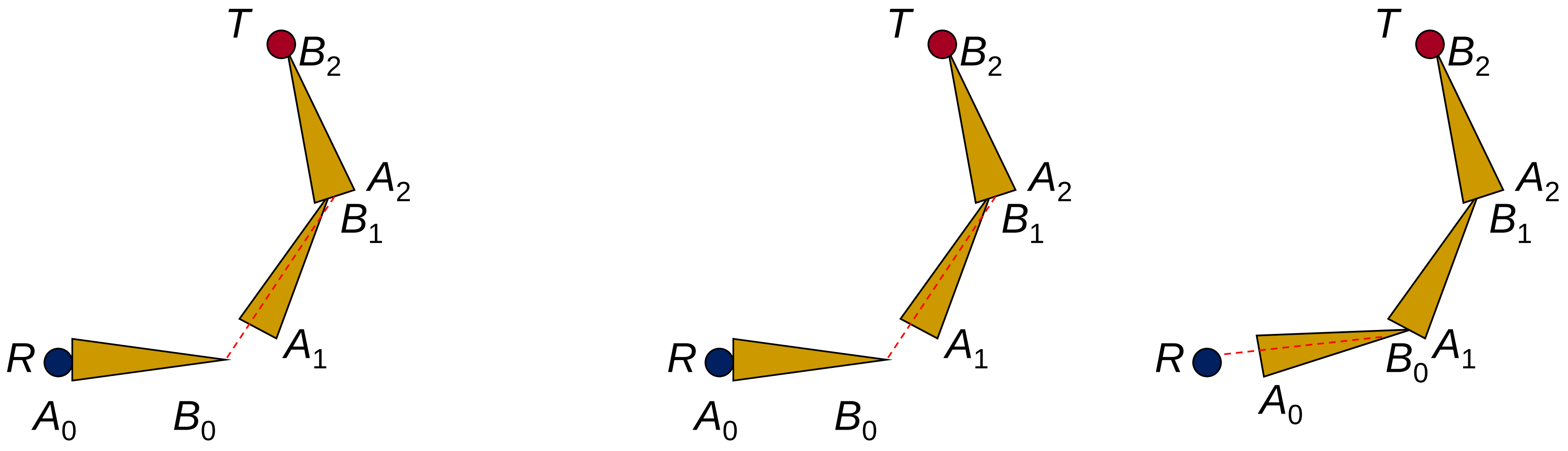


Move B_2 to the target. A_2 is a point at a distance d from B_2 along the line B_1T

Move B_1 to the new position of A_2. A_1 is a point at a distance d from B_1 along the line B_0A_2



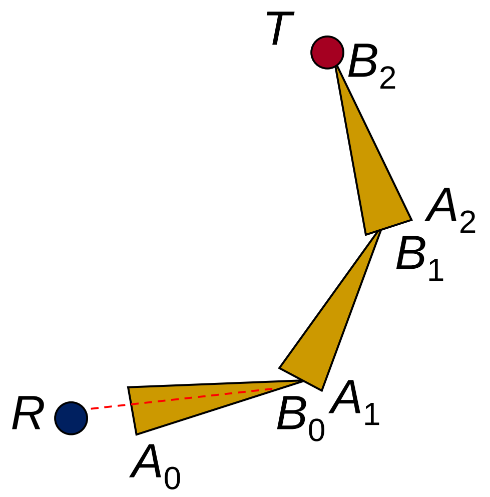
FABRIK: Forward Pass

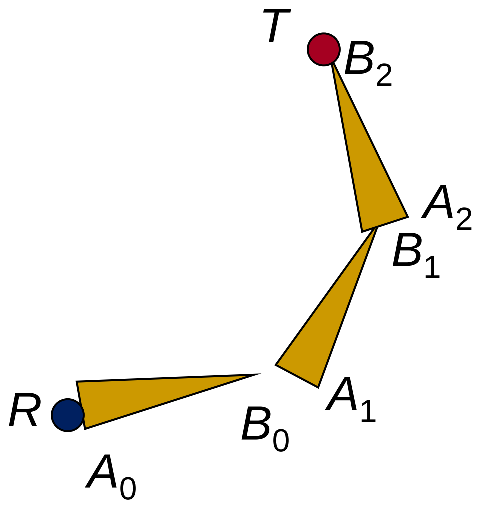


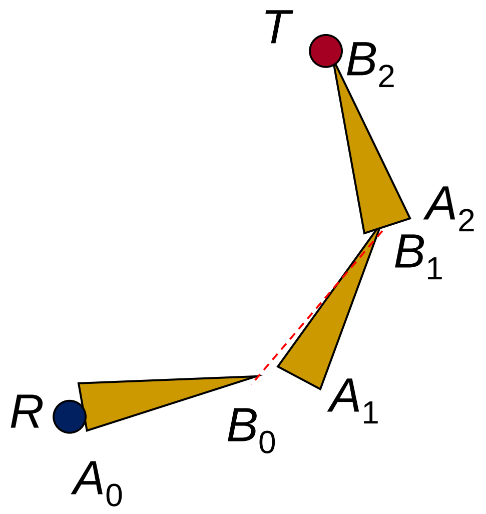
Move B_0 to the new position of A_1. A_0 is a point at a distance d from B_0 along the line RA_1

This completes the forward pass

FABRIK: Backward Pass



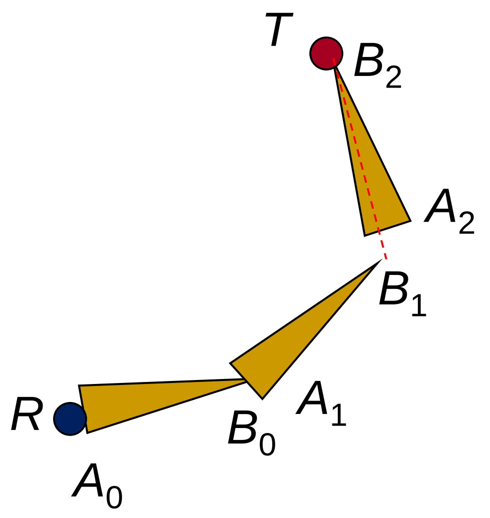


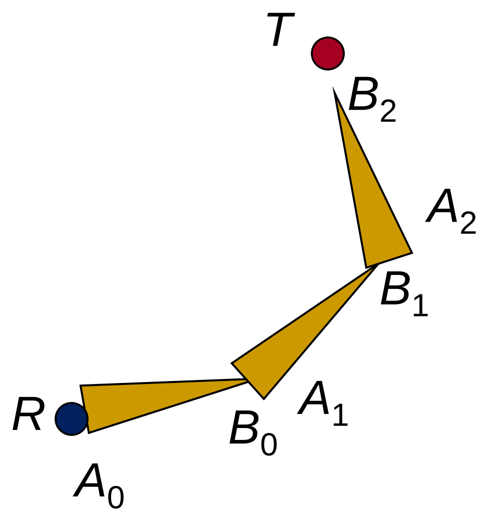


Move A_0 to R. B_0 is a point at a distance d from A_0 along the line RA_1

Move A_1 to the new position of B_0. B_1 is a point at a distance d from A_1 along the line B_0A_2

FABRIK: Backward Pass





Move A_2 to the new position of B_1. B_2 is a point at a distance d from A_2 along the line B_1T

This completes the backward pass

The forward and backward passes are repeated till the required configuration is achieved.

NOTE: Before performing the iterations, we much check if the target is reachable.
$RT \leq n\,d$,   where n is the number of links, and d is the link length.