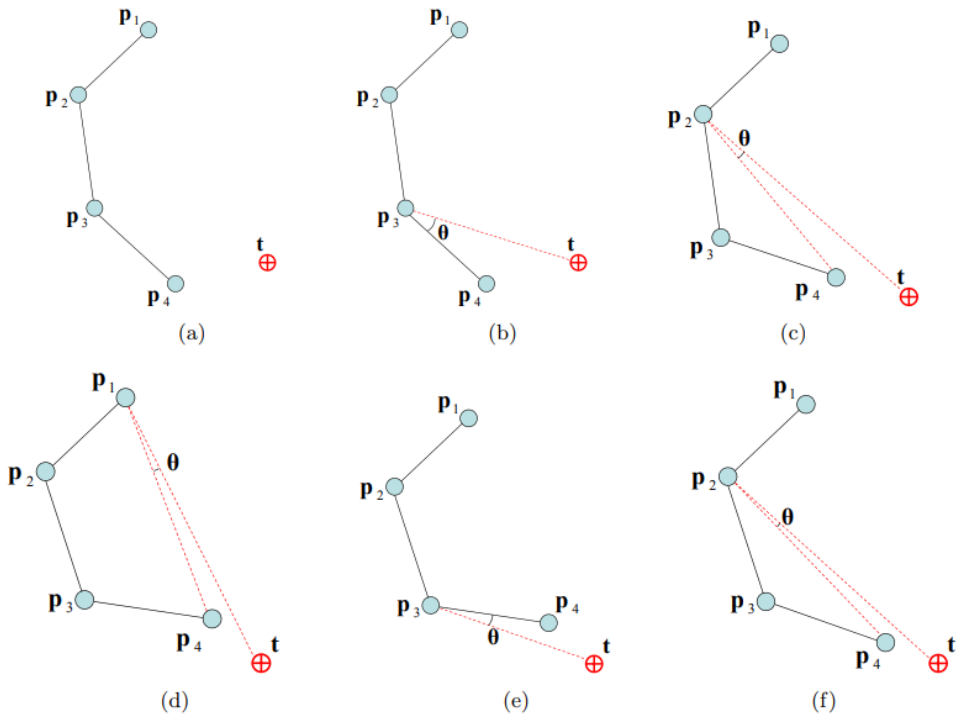
6. Heuristic Inverse Kinematics Algorithms

Game Engineering
Prof. HyeongYeop Kang
siamiz@khu.ac.kr
IIIXR LAB

Cyclic Coordinate Descent

Iterative Heuristic Search

- Cyclic Coordinate Descent (CCD)[1] is an iterative heuristic search technique that is suitable for interactive control of an articulated body.
  - The CCD method attempts to minimize position and orientation errors by transforming one joint variable at a time.
  - The algorithm states that, starting from the end effector inward towards the manipulator base, each joint must be transformed in order to move the end effector as close as possible to the target.
  - This procedure is repeated until a satisfactory solution is obtained.
  - The computational cost for each joint is low and therefore a solution can be formulated very quickly.

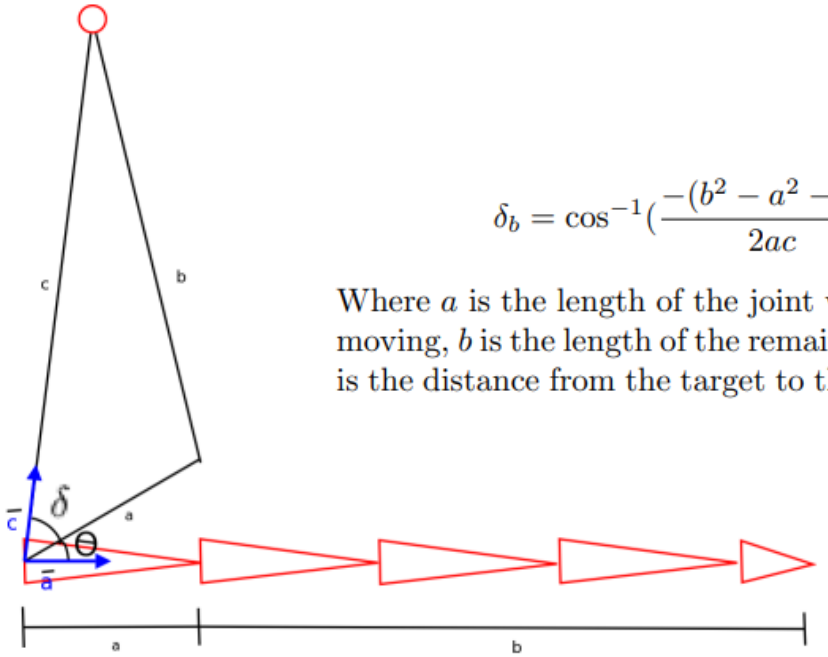


→ *An example of visual solution of the IK problem using the CCD algorithm. (a) The initial position of the manipulator and the target, (b) find the angle θ between the end effector, joint $\mathbf{p}_3$ and the target and rotate the joint $\mathbf{p}_4$ by this angle, (c) find the angle θ between the end effector, joint $\mathbf{p}_2$ and the target and rotate joints $\mathbf{p}_4$ and $\mathbf{p}_3$ by this angle, (d), (e) and (f) repeat the whole process for as many iterations as needed. Stop when the end effector reaches the target or gets sufficiently close.*

Triangulation IK

Law of Cosines for IK

- Triangulation Inverse Kinematics approach[2] uses the cosine rule to calculate each joint angle starting at the root of the kinematic chain moving outward towards the end effector.
  - It is guaranteed to find a solution when used with unconstrained joints and when the target is in range.
  - The Triangulation algorithm incurs a lower computational cost than the CCD algorithm, since it needs only 1 iteration to reach the target.



$$\delta_b = \cos^{-1}\left(\frac{-(b^2 - a^2 - c^2)}{2ac}\right)$$

Where a is the length of the joint we are currently moving, b is the length of the remaining chain and c is the distance from the target to the current joint.

Triangulation IK

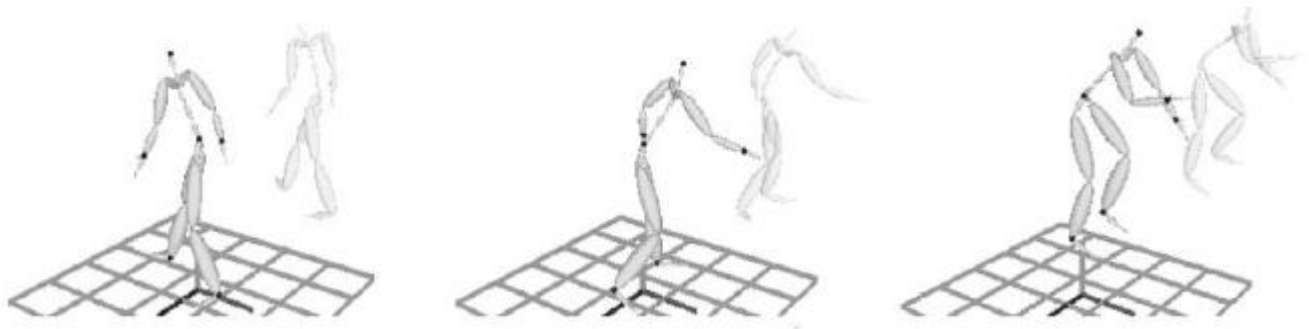
Law of Cosines for IK

- However, the results are not realistic.
  - The joints close to the end effector are usually in a straight line, with the emphasis on rotation of the joints neighboring the root.
  - The Triangulation IK method can only be applied to problems with a single end effector; kinematic chains with multiple end effectors cannot be solved and it cannot be used for complex character models.
  - Another drawback of this algorithm is that, when constraints are applied, the end effector often cannot reach the target, even if there is a solution.
  - This happens because each joint position is calculated independently without considering the restrictions of the next joint.

Sequential Inverse Kinematics

Sequential Analytic-iterative IK

- Sequential Inverse Kinematics (SIK)[3] is an analytic-iterative IK method that reconstructs 3d human full-body movements in real-time.
  - The inputs to this method are end effector positions, such as wrists, ankles, head and pelvis (the least possible input in order to be usable within a low-cost motion capture system in real-time), which are used to find the human pose.
- The procedures are as follows:
  1. The orientation of the root joint is estimated from the known positions.
  2. The configuration of the spine is found using a hybrid IK method that combines this estimated orientation with the positions of the root and head markers.
  3. Then the orientations of clavicles are determined with the positions of their known end-effector positions and the already positioned spine.
  4. Finally each of the limbs is situated according to their known end-effector positions.

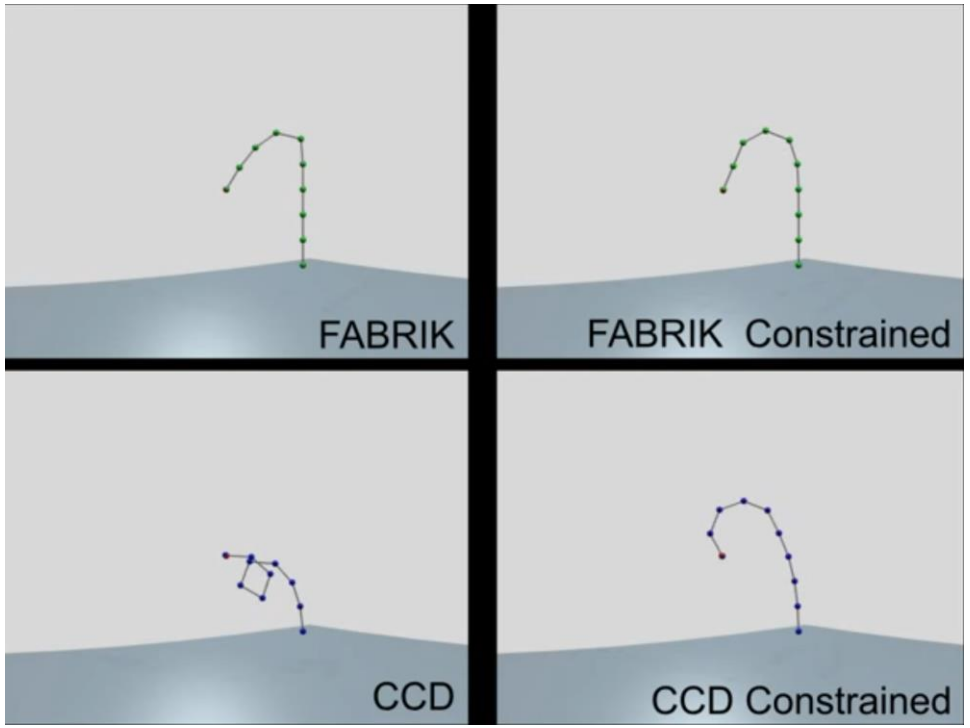


samples of the playground animation: the inputs are marked with black verticies

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

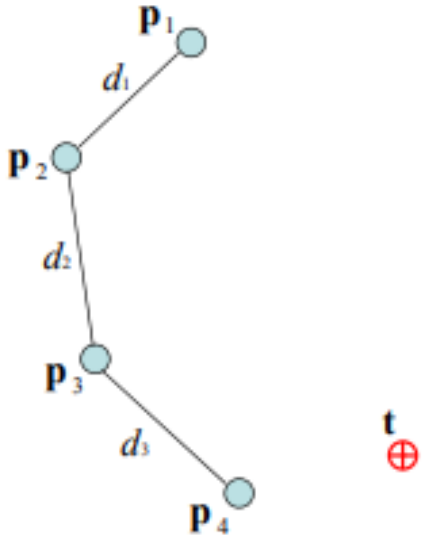
- Forward and Backward Reaching Inverse Kinematics (FABRIK) uses the previously calculated positions of the joints to find the updates in a forward and backward iterative mode.
- FABRIK involves minimizing the system error by adjusting each joint angle one at a time.
- The proposed method starts from the last joint of the chain and works forwards, adjusting each joint along the way. Thereafter, it works backward in the same way, in order to complete a full iteration.
- This method, instead of using angle rotations, treats finding the joint locations as a problem of finding a point on a line; hence, it converges in fewer iterations, has low computational cost, and produces visually realistic poses.
- FABRIK has been integrated in several game engines, including the Unreal Engine, Unity3D, and Panda.



FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- Before explaining the procedure, let's define several parameters.
  - Assume p_1, ..., p_n are the joint positions of a manipulator.
  - Also, assume that p_1 is the root join and p_n is the end effector, for the simple case where only a single end effector exists.
  - The target is symbolized as **t** and the initial base position by **b**.
  - Assume that we have a single target and 4 joints.

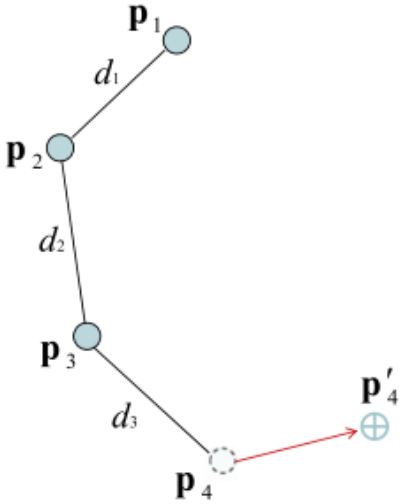


- The procedures are as follows:
  1. Calculate the distances between each joint $d_i = |p_{i+1} - pi|$, for i = 1, ..., n-1.
  2. Check whether the target is reachable or not; find the distance between the root and the target, and if this distance is smaller than the total sum of all the inter-joint distances, the target is within reach, otherwise, it is unreachable.
  3. If the target is within reach, a full iteration is constituted by two stages.

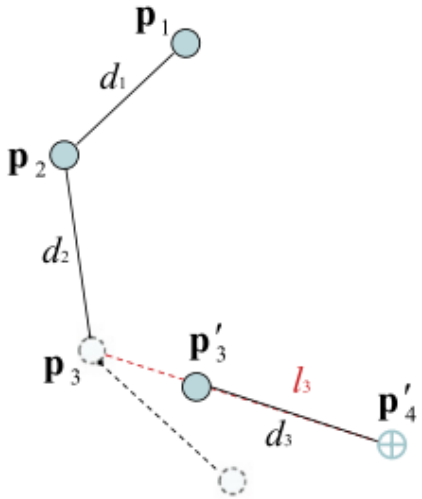
FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

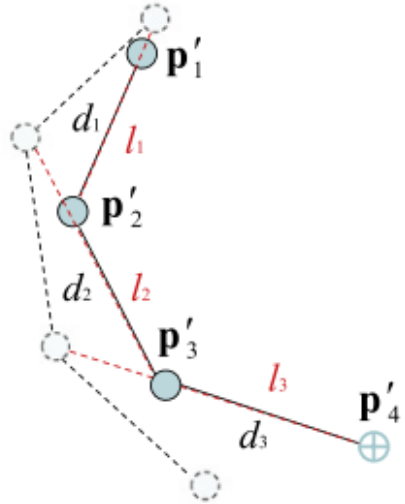
- The procedures are as follows (continued):
  4. In the first stage:
     a. The algorithm estimates each joint position starting from the end effector, p_n, moving inwards to the manipulator base, p_1.
     b. So, let the new position of the end effector be the target position, p'_n = **t**.
     c. Find the line, $l_{n\text{-}1}$, which passes through the joint positions p_n and p'_n .
     d. The new position of the $(n\text{-}1)^{th}$ joint, p'_{n-1}, lies on that line with distance $d_{n\text{-}1}$ from p'_n.
     e. Similarly, the new position of the $(n\text{-}2)^{th}$ joint, p'_{n-2}, can be calculated using the line $l_{n\text{-}2}$, which passes through the p_{n-2} and p'_{n-1}, and has distance $d_{n\text{-}2}$ from p'_{n-1}.
     f. The algorithm continues until all new joint positions are calculated, including the root, p'_1.



step 4.b.



step 4.c. ~ d.

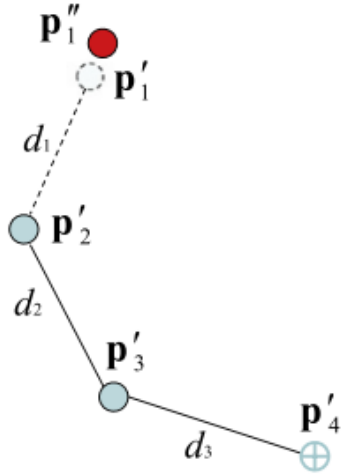


step 4.e. ~ f.

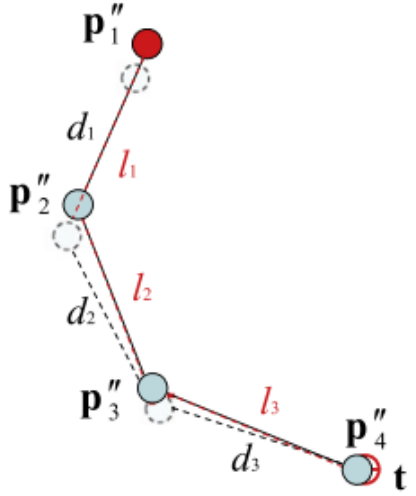
FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- The procedures are as follows (continued):
  5. In the second stage:
     a. The same procedure is repeated but this time starting from the root joint and moving outwards to the end effector.
     b. Let the new position for the 1st joint, p_1'', be its initial position.
     c. Then, using the line l_1, that passes through the points p_1'' and p_2'', we define the new position of the point p_2'' as the point on that line with distance d_1 from p_1''.
     d. This procedure is repeated for all the remaining joints, including the end effector.
  6. After one complete iteration, it is almost always the case that the end effector is closer to the target.
  7. The procedure is then repeated until the end effector is identical or close enough to the desired target.



step 5.b.



step 5.c. ~ d.

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK Algorithm

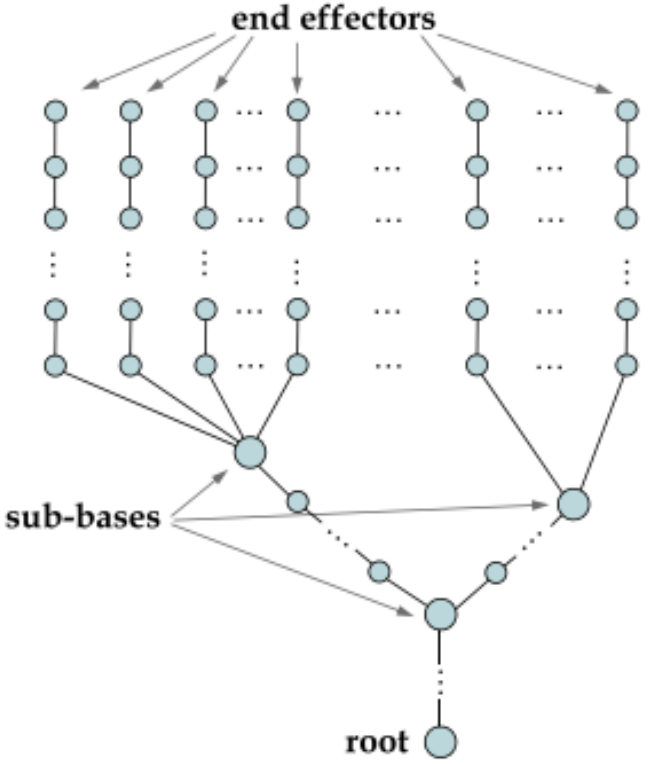
Input: The joint positions $\mathbf{p}_i$ for $i = 1,\ldots,n$, the target
position $\mathbf{t}$ and the distances between each joint
$d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1,\ldots,n-1$.
Output: The new joint positions $\mathbf{p}_i$ for $i = 1,\ldots,n$.

1.1 *% The distance between root and target*
1.2 $dist = |\mathbf{p}_1 - \mathbf{t}|$
1.3 *% Check whether the target is within reach*
1.4 **if** $dist > d_1 + d_2 + \ldots + d_{n-1}$ **then**
1.5 *% The target is unreachable*
1.6 **for** $i = 1,\ldots,n-1$ **do**
1.7 *% Find the distance r_i between the target $\mathbf{t}$ and the joint position $\mathbf{p}_i$*
1.8 $r_i = |\mathbf{t} - \mathbf{p}_i|$
1.9 $\lambda_i = d_i / r_i$
1.10 *% Find the new joint positions $\mathbf{p}_i$.*
1.11 $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{t}$
1.12 **end**
1.13 **else**
1.14 *% The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$*
1.15 $\mathbf{b} = \mathbf{p}_1$
1.16 *% Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater than a tolerance.*
1.17 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.18 **while** $dif_A > tol$ **do**
1.19 *% STAGE 1: FORWARD REACHING*
1.20 *% Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$*
1.21 $\mathbf{p}_n = \mathbf{t}$
1.22 **for** $i = n-1,\ldots,1$ **do**
1.23 *% Find the distance r_i between the new joint position $\mathbf{p}_{i+1}$ and the joint $\mathbf{p}_i$*
1.24 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.25 $\lambda_i = d_i / r_i$
1.26 *% Find the new joint positions $\mathbf{p}_i$.*
1.27 $\mathbf{p}_i = (1 - \lambda_i)\,\mathbf{p}_{i+1} + \lambda_i \mathbf{p}_i$
1.28 **end**
1.29 *% STAGE 2: BACKWARD REACHING*
1.30 *% Set the root $\mathbf{p}_1$ its initial position.*
1.31 $\mathbf{p}_1 = \mathbf{b}$
1.32 **for** $i = 1,\ldots,n-1$ **do**
1.33 *% Find the distance r_i between the new joint position $\mathbf{p}_i$ and the joint $\mathbf{p}_{i+1}$*
1.34 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.35 $\lambda_i = d_i / r_i$
1.36 *% Find the new joint positions $\mathbf{p}_i$.*
1.37 $\mathbf{p}_{i+1} = (1 - \lambda_i)\mathbf{p}_i + \lambda_i \mathbf{p}_{i+1}$
1.38 **end**
1.39 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.40 **end**
1.41 **end**

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with Multiple End Effectors
  - In reality, most of the multibody models, such as hands, human or legge bodies etc., are comprised of several kinematic chains, and each chain generally has more than 1 end effector.
  - Therefore, it is essential for an IK solver to be able to solve problems with multiple end effectors and targets.
  - FABRIK can be easily extended to process models with multiple end effectors.
    - ✓ However, prior knowledge of the model, such as the subbase joint and the number and structure of chains is needed.
    - ✓ A sub-base joint is a joint which connects 2 or more chains.



FABRIK

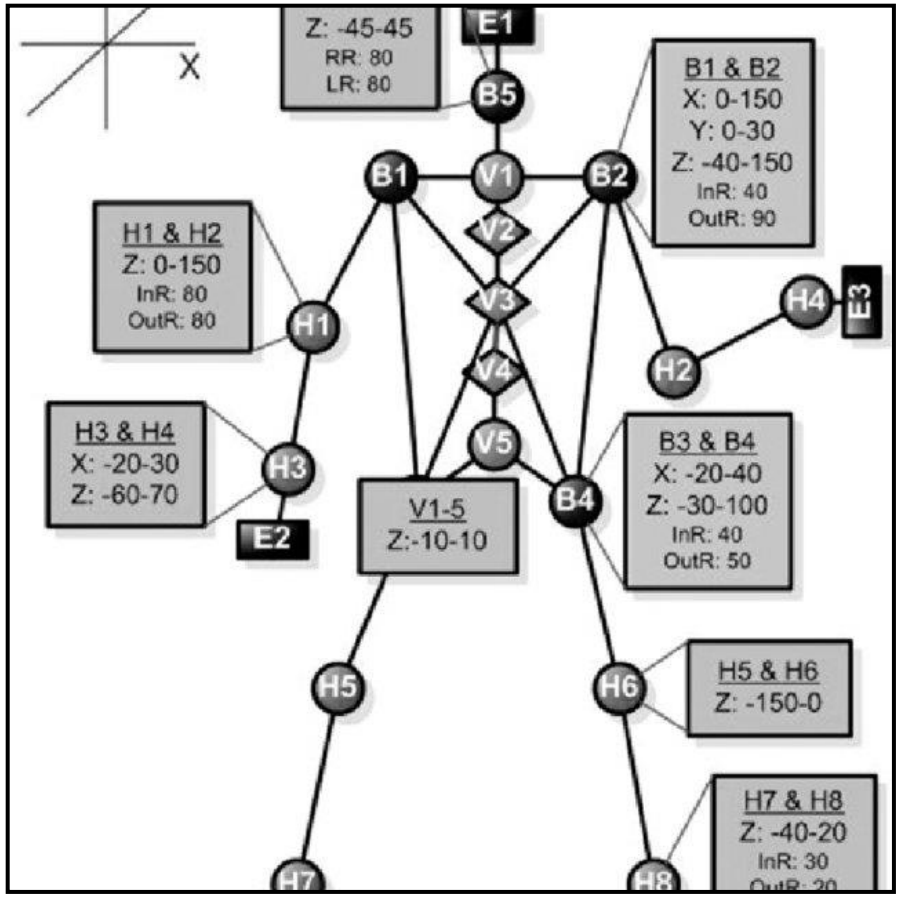
A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with Multiple End Effectors
  - The algorithm is divided into two stages, as in the single end effector case.
  - First stage:
    1. The normal algorithm is applied but this time starting from each end effector and moving inwards to the parent sub-base.
    2. This will produce as many different positions of the sub-base as the number of end effectors connected with that specific sub-base.
    3. The new position of the sub-base will then be the centroid of all these positions.
    4. Thereafter, the normal algorithm should be applied inwards starting from the sub-base to the manipulator root. If there are more intermediate sub-bases, the same technique should be used.
  - Second stage:
    1. The normal algorithm is applied starting now from the root and moving outwards to the sub-base.
    2. Then, the algorithm should be applied separately for each chain until the end effector is reached; if more sub-bases exist, the same process is applied.
    3. The method is repeated until all end effectors reach the targets or there is no significant change between their previous and their new positions.

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

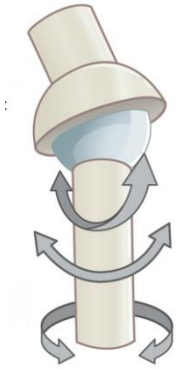
- FABRIK with joint limits (constraints)
  - Most human body models are consisted of joints and the joints have biomechanical constraints, which restricts the rotations for achieving realistic body motion.
  - Since FABRIK is iterative, the joint restrictions can be enforced at each step just by taking the resultant orientation and forcing it to stay within the valid range.
    - ✓ The main idea is the re-positioning and re-orientation of the target to be within the allowed range bounds ensuring that the joint restrictions are always satisfied.
    - ✓ In contrast to most existing techniques for joint constraints, the proposed methodology simplifies the 3D problem into a 2D problem, meaning that the complexity and the required processing time is reduced.



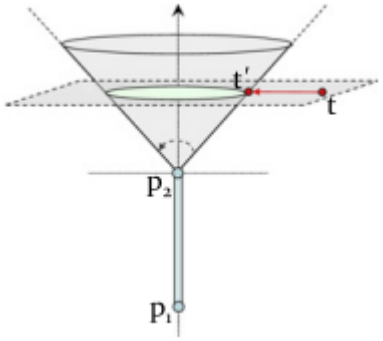
FABRIK

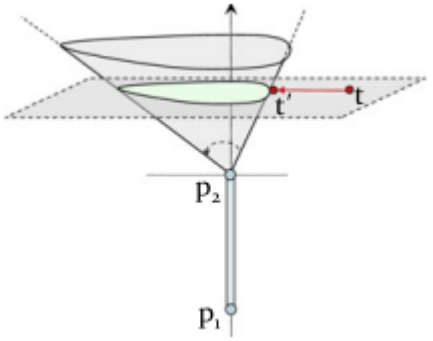
A fast, iterative solver for the Inverse Kinematics Problem

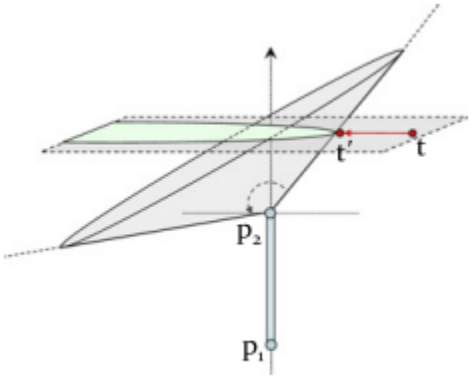
- FABRIK with joint limits (constraints)
  - Assume we have a ball-and-socket joint with orientational limits described by the rotor R and rotational limits described by the angles θ_1, ..., θ_4.
  - A graphical representation of a joint limit boundary could be an irregular cone which is defined by these angles.
  - There are 3 possible conic sections, according to the angles defining the irregular cone:
    1. If all θs are equal, the conic section is a circle.
    2. If there are θs are greater or smaller than 90 and are not equal, then the conic section has an ellipsoidal shape.
    3. If there are θs both greater and smaller than 90, then the joint boundary limits are defined by a parabolic shape.



ball & socket joint





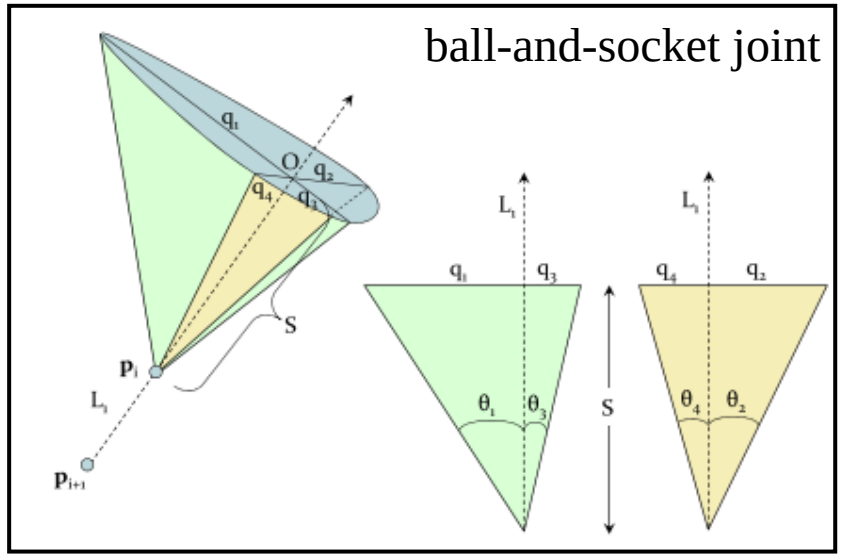


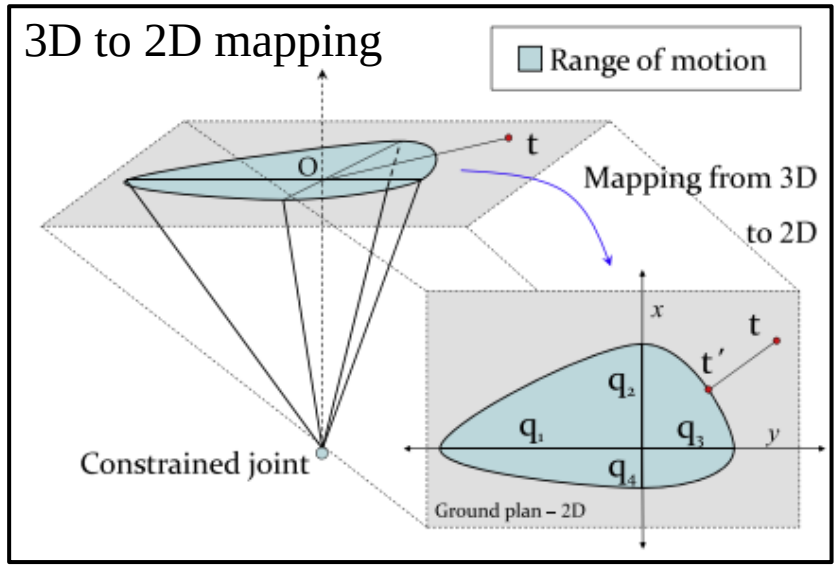
3 types of joint restriction

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with joint limits (constraints)
  - In this lecture, the joint limits are assumed to be defined by an ellipsoidal shape, since this is the most common case, but similar procedures apply for different conic sections.
  - The graphical illustration of the ball-and-socket joint is shown right:
    - ✓ The description of symbols are given on subsequent pages.
  - Then the composite ellipsoidal shape created by the distances q_j mapped from 3D to 2D can be illustrated as shown in the right:





FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with joint limits (constraints)
  - The orientation of the joint can be assigned as follows:
    1. Assume we are in the first stage of the algorithm, i.e. we have just calculated the new position of joint p'_n, and we want to find the new position of the $(i\text{-}1)^{th}$ joint, p'_{i-1}.
    2. Find the rotor expressing the rotation between the orientation frames at joints p'_i and $p_{i-1}$and if this rotor represents a rotation greater than a limit, reorient the joint p_{i-1} in such a way that the rotation will be within the limits.

Algorithm 2. The orientational constraints

Input: The rotor **R** expressing the rotation between the
orientation frames at joints $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$.
Output: The new re-oriented joint $\mathbf{p}'_{i-1}$.

2.1 Check whether the rotor **R** is within the motion range
bounds
2.2 **if** within the bounds **then**
2.3 do nothing and exit
2.4 **else**
2.5 reorient the joint $\mathbf{p}_{i-1}$ in such a way that the rotor will
be within the limits
2.6 **end**

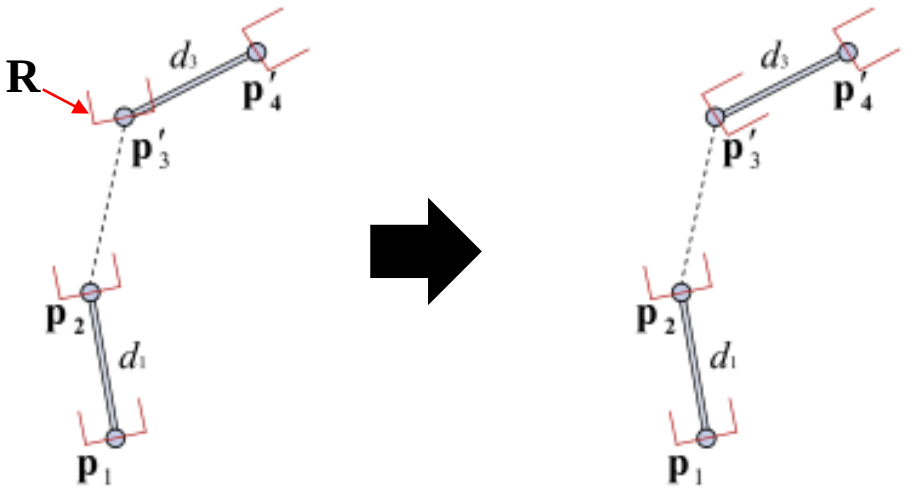
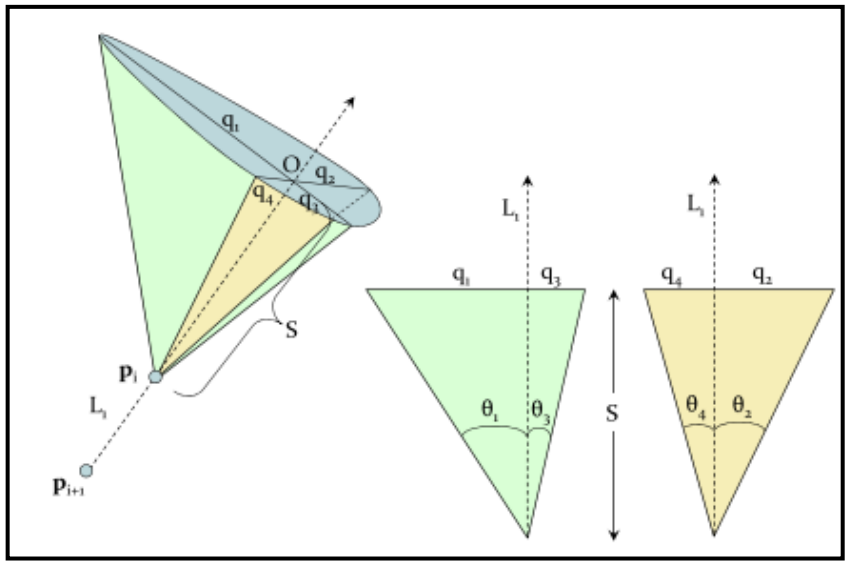


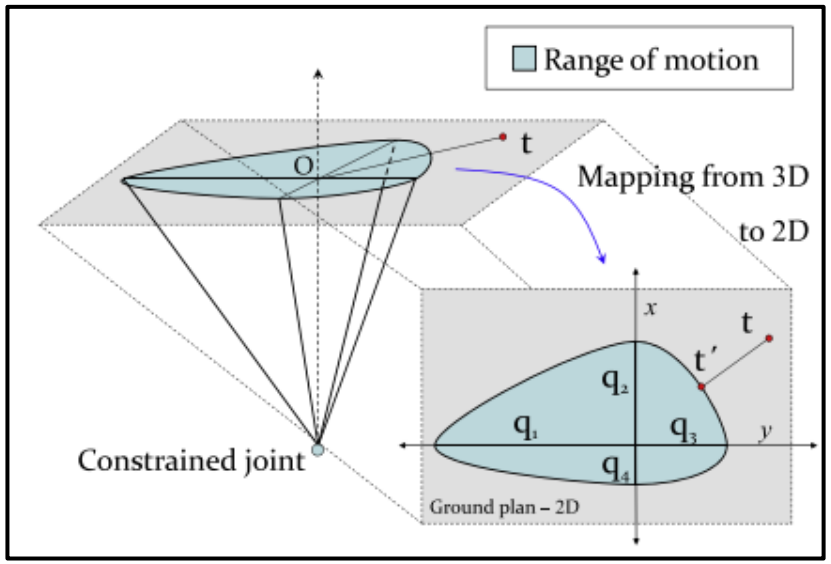
Illustration of input and output

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with joint limits (constraints)
  - Once the joint orientation is established, the rotational limits, described by angles θ_1, ..., θ_4, can be applied as follows:
    1. Find the projection O of the target **t** on line L_1, where L_1 is the line passing through the joint under consideration, p_i, and the previous joint of the chain, p_{i+1}.
    2. Then determine the distance S from the point O to the joint position p_i and calculate the distances $q_j = S * \tan\theta_j$, for j = 1,..., 4, as shown on the figure left below.
    3. Then apply a rotation and translation which takes O to the origin and the axes defining the constraints to the x and y axes, as shown on the figure right below. Working in this 2D plane, locate the target in a particular quadrant and find the ellipse defined on that quadrant using the associated distances q_j.





FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK with joint limits (constraints)
  - Once the joint orientation is established, the rotational limits, described by angles θ_1, ..., θ_4, can be applied as follows:
    4. Find the nearest point on that ellipse from the target, if the target is not in the allowed motion range.
    5. It is not necessary to calculate all the ellipses which define the composite ellipsoidal shape, but only the ellipse related to the quadrant in which the target is located.
    6. Lastly, undo the initial transformation which mapped O to the origin.

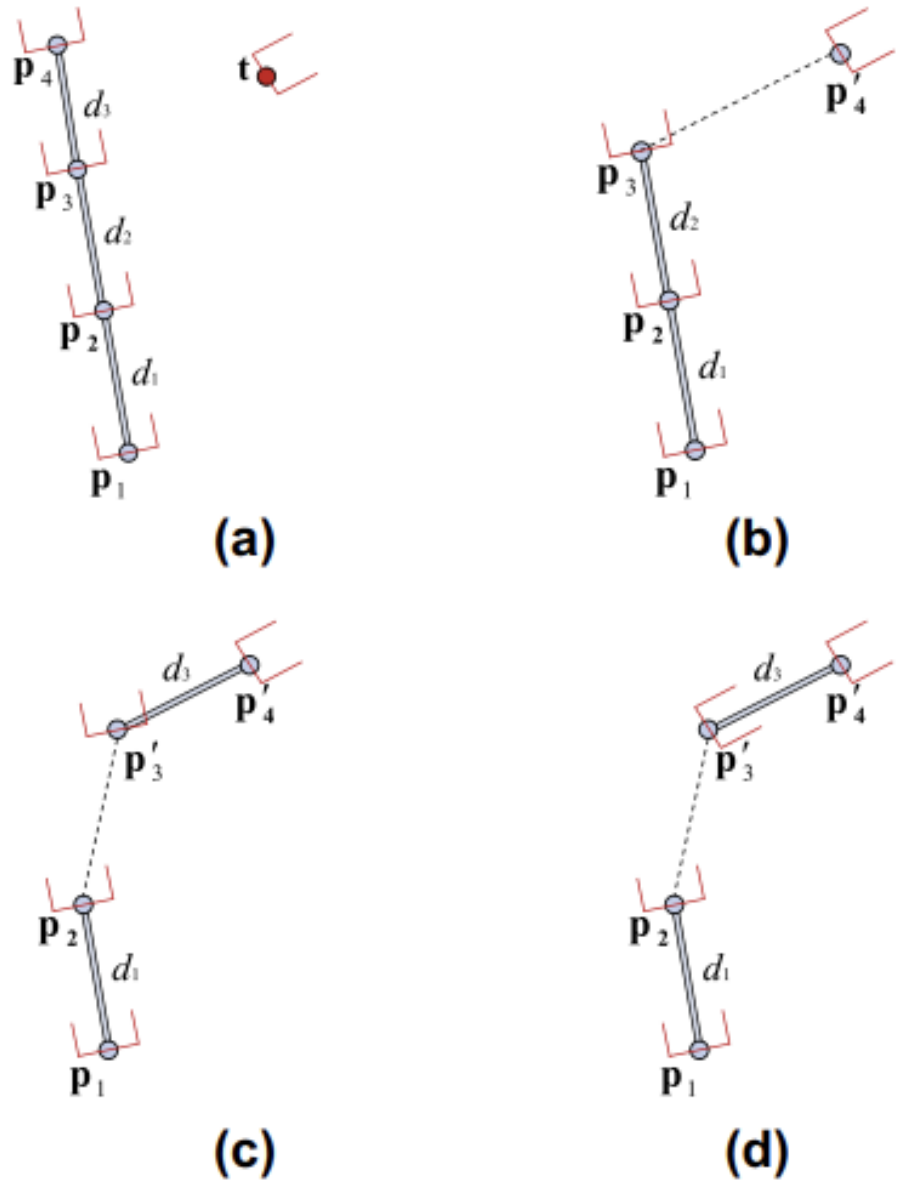
Input The target position **t** and the angles defining the rotation constraints θ_j for $j = 1,\ldots,4$.
Output: The new target position **t**′.

3.1 Find the line equation L_1
3.2 Find the projection O of the target **t** on line L_1
3.3 Find the distance between the point O and the joint position
3.4 Map the target (rotate and translate) in such a way that O is now located at the axis origin and oriented according to the x and y-axis ⇒ Now it is a 2D simplified problem
3.5 Find in which quadrant the target belongs
3.6 Find what conic section describes the allowed range of motion
3.7 Find the conic section which is associated with that quadrant using the distances $q_j = \mathrm{Stan}\theta_j$, where $j = 1,..,4$
3.8 *% Check whether the target is within the conic section or not*
3.9 **if** within the conic section **then**
3.10 use the true target position **t**
3.11 **else**
3.12 Find the nearest point on that conic section from the target
3.13 Map (rotate and translate) that point on the conic section via reverse of **3.4** and use that point as the new target position
3.14 **end**

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

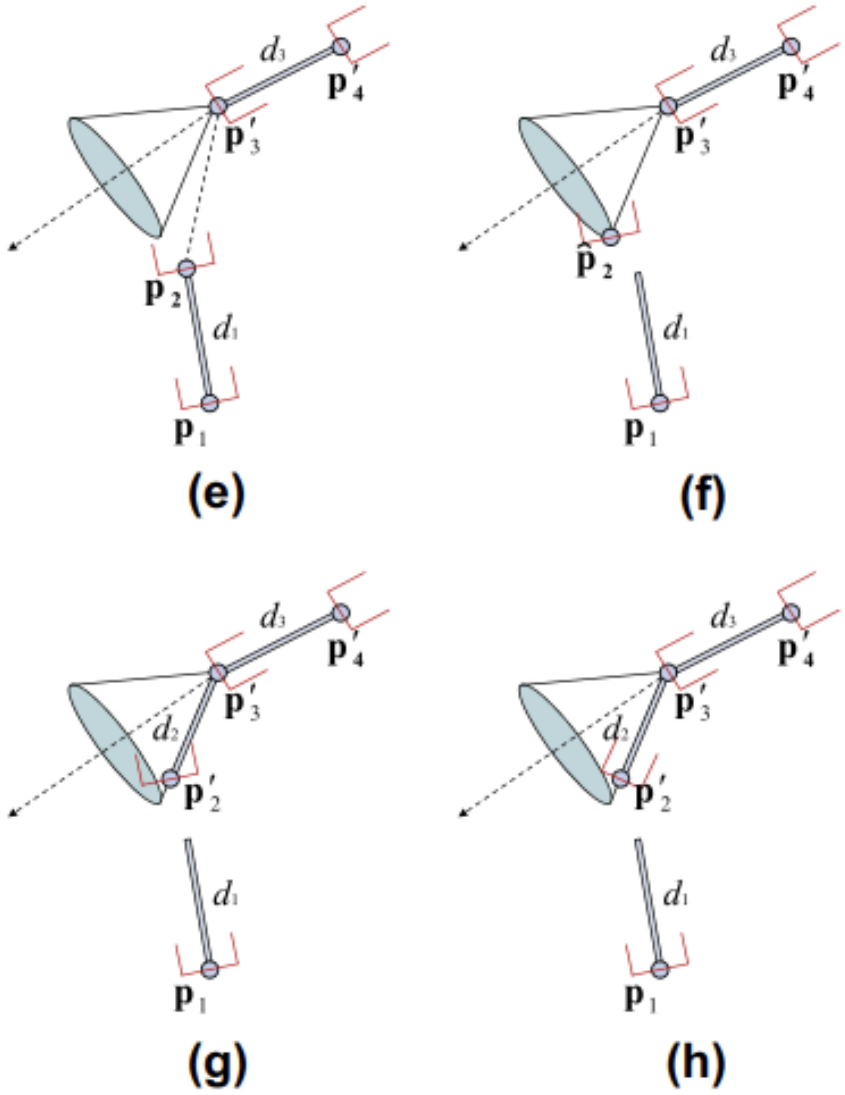
- FABRIK with joint limits (constraints)
  - Incorporating rotational and orientational constraints within FABRIK.
  - (a) The initial configuration of the manipulator and the target.
  - (b) Relocate and reorient joint p_4 to target **t.**
  - (c) Move joint p_3 to p_3', which lies on the line that passes through the points p_4' and p_3 and has distance d_3 from p_4'.
  - (d) Reorient joint p_3' in such a way that the rotor expressing the rotation between the orientation frames at joints p_3' and p_4' is within the motion range bounds.



FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

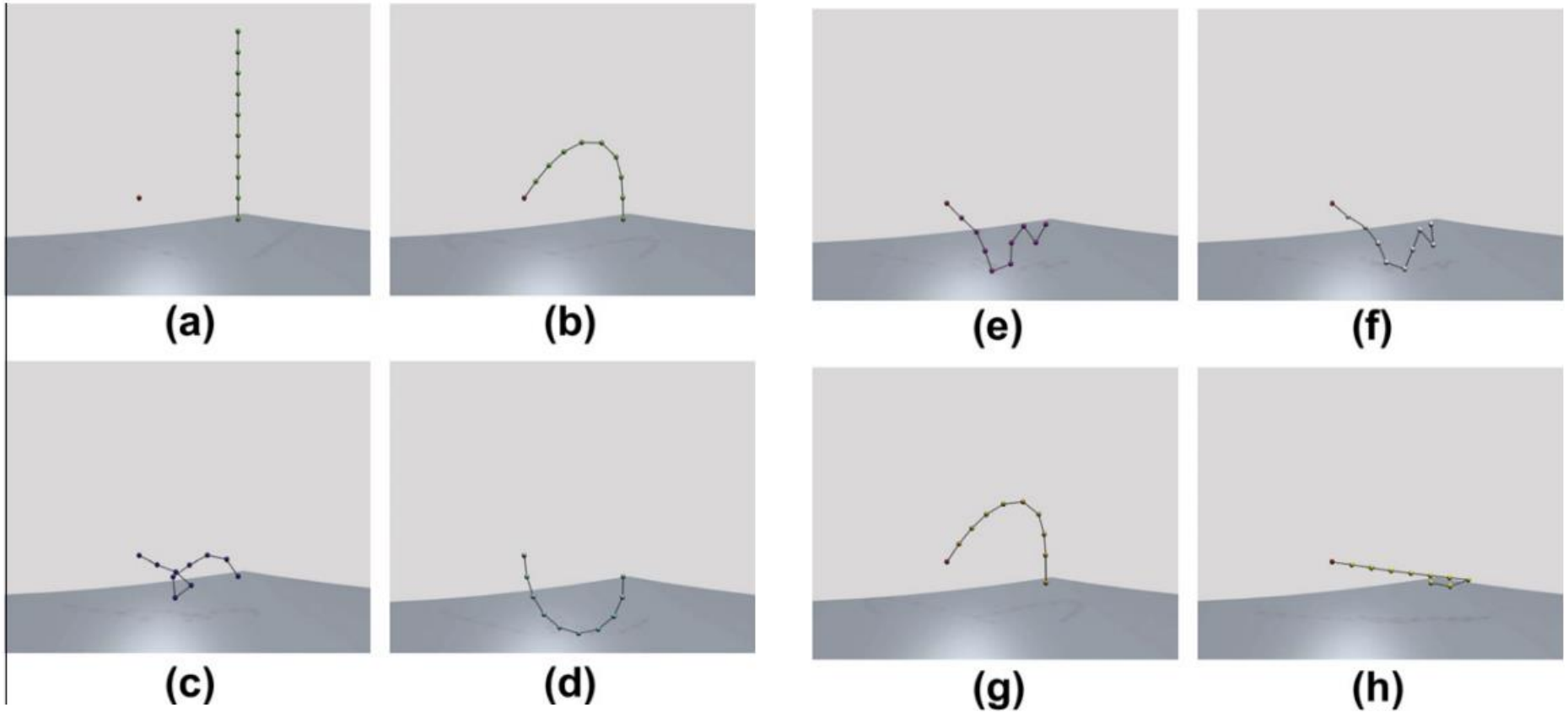
- FABRIK with joint limits (constraints)
  - (e) The rotational constraints: the allowed regions shown as a shaded composite ellipsoidal shape.
  - (f) The joint position p_2 is relocated to a new position, $\hat{p}_2$, which is the nearest point on that composite ellipsoidal shape from p_2, ensuring that the new joint position p_2' will be within the allowed rotational range.
  - (g) Move $\hat{p}_2$ to p_2', to conserve bone length.
  - (h) Reorient the joint p_2' in order to satisfy the orientation limits. This procedure is repeated for all the remaining joints in a forward and backward fashion



FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK produces visually the smoothest and most natural movements than previous IK methods.
  - In the figure below, comparison results with popular IK methods are presented.

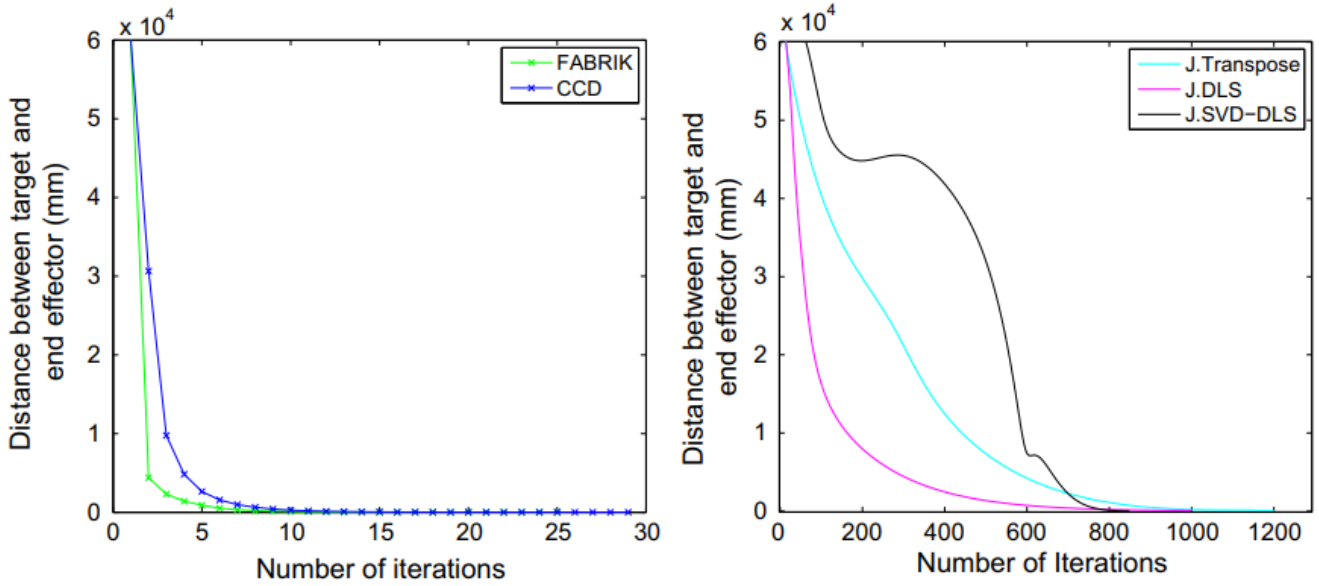


(a) Initial position, (b) FABRIK, (c) CCD, (d) J. Transpose, (e) J. DLS, (f) J. SVD-DLS, (g) Follow-the-leader, (h) Triangulation.

FABRIK

A fast, iterative solver for the Inverse Kinematics Problem

- FABRIK produces results significantly faster than all IK methods tested.
  - FABRIK is approximately 10 times faster than the CCD method and a thousand times faster than the Jacobian-based methods.
  - FABRIK has the lowest computational cost and, at the same time, produces visually the smoothest and most natural movements.
  - FABRIK needs the fewest iterations to reach the target, it converges faster to the desired position and, when the target is unreachable, it keeps the end effector pointing to the target.

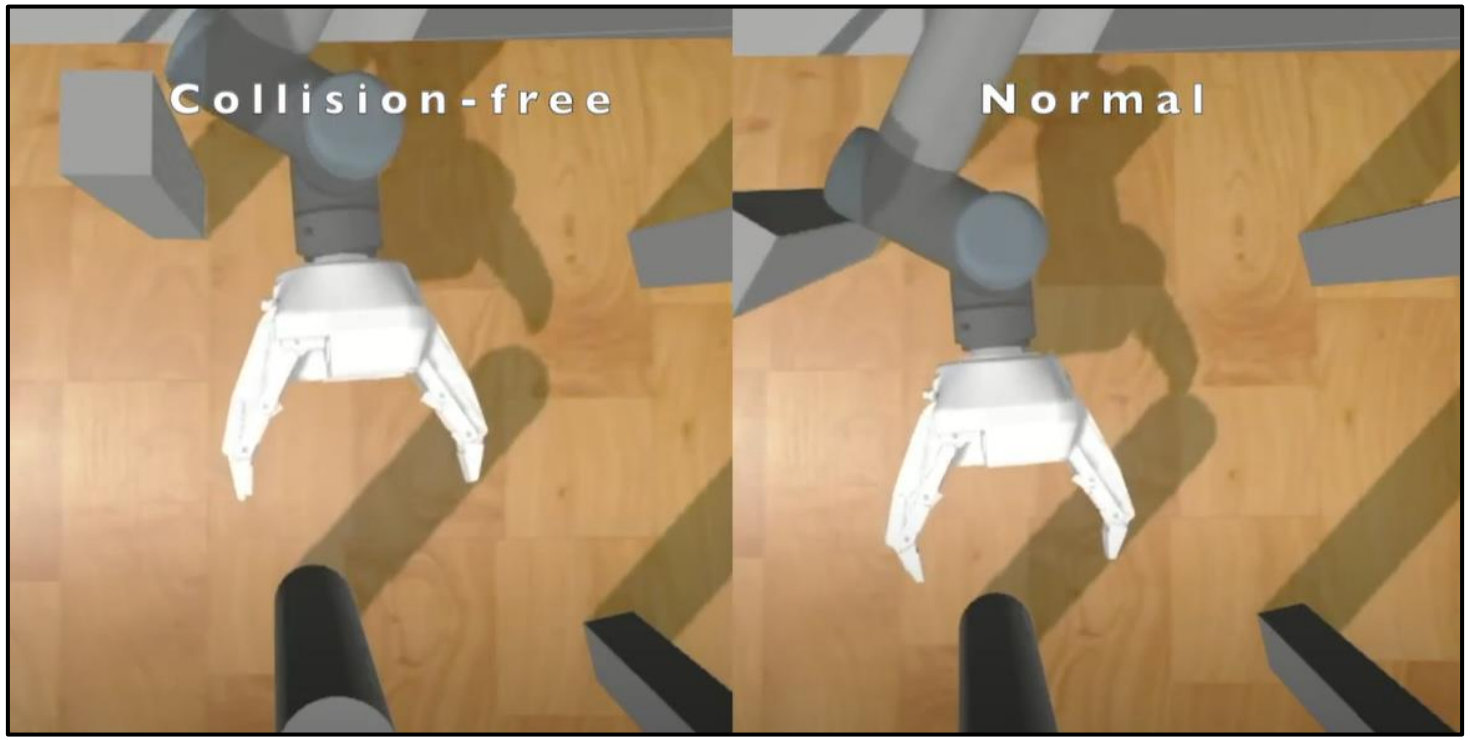


The number of iterations needed to reach the target against the distance between target and end effector as this changes over time.

More Advanced Methods

Directions for improving

- Warm start for FABRIK
- Collision-free FABRIK
- Data-driven IK
- Deep learning for IK



Reference

[1] https://en.wikipedia.org/wiki/Definite_matrix
[2] R. M¨uller-Cajar and R. Mukundan. Triangulation: A new algorithm for inverse kinematics. In Proceedings of the Image and Vision Computing New Zealand 2007, pages 181–186, New Zealand, December 2007.
[3] Luis Unzueta, Manuel Peinado, Ronan Boulic, and Angel Suescun. Full-body performance animation with sequential inverse kinematics. Graph. Models, 70(5):87–104, 2008.
[4] Andreas Aristidou and Joan Lasenby. FABRIK: a fast, iterative solver for the inverse kinematics problem. Submitted to Graphical Models, 2010.