ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

centre adscrit a: UNIVERSITAT DE BARCELONA

Animation Foundations

12. Inverse Kinematics. FABRIK

Dr Joan Llobera – joanllobera@enti.cat

Reminder

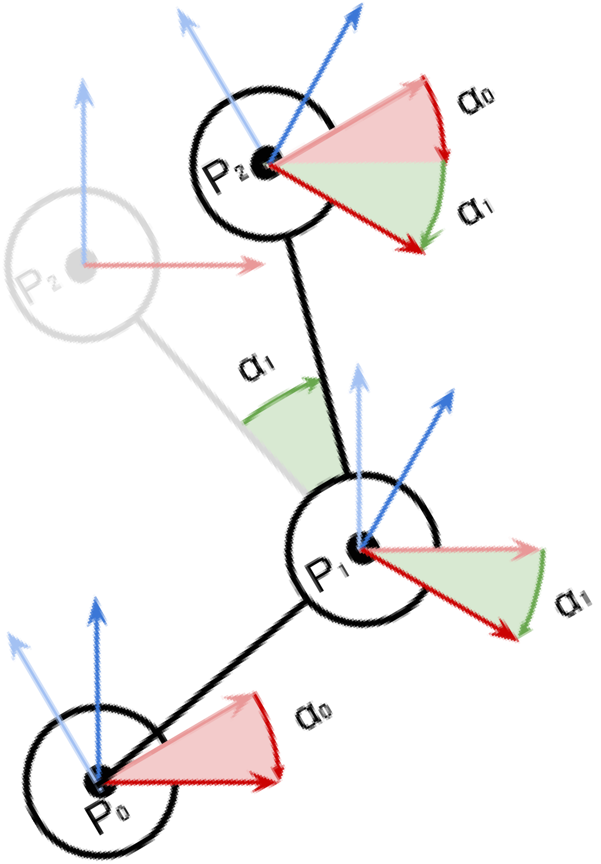
Inverse kinematics. The intro

Idea!

We can define an optimization function to minimize a distance depending on a certain number of angles

Min function(distance(angles) ,angles) = ?

But how?

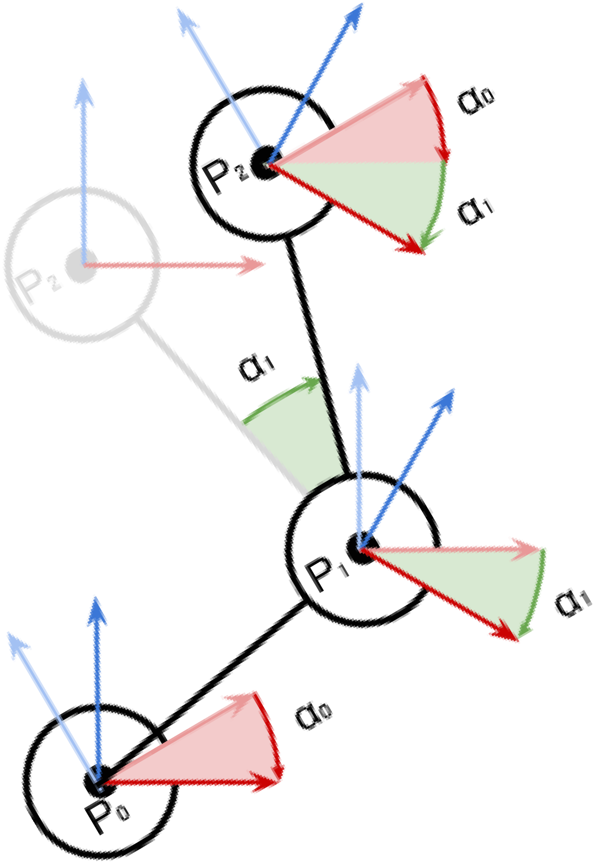


Reminder

Inverse kinematics. The methods

3 methods:

- Gradient Descent (GD)
- Cyclic Coordinate Descent (CCD)
- Forward and Backward Recursive Inverse Kinematic (FABRIK)



ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

UNIVERSITAT DE BARCELONA

CCD

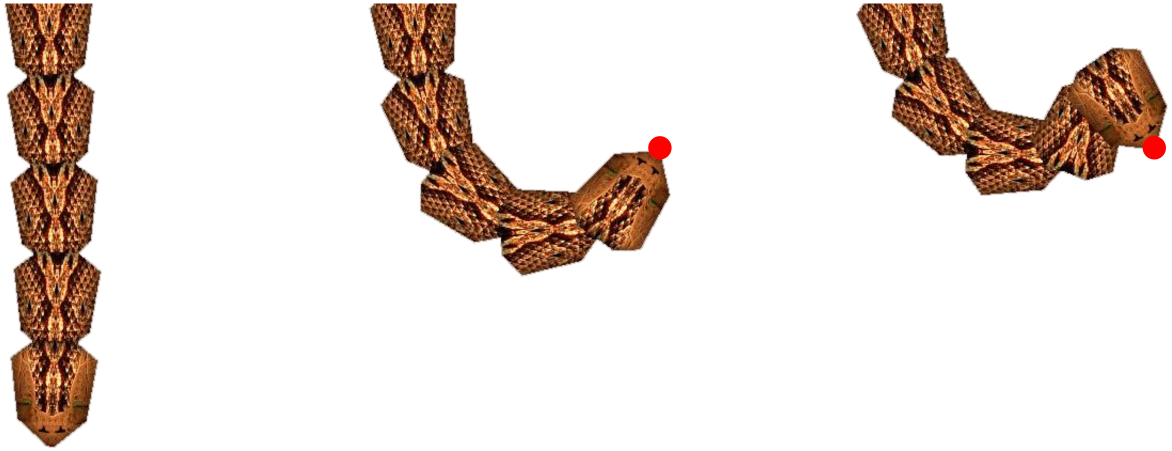
Exercise:

Implement IK based on CCD

CCD

- Advantages
  - Fast computation of one iteration
  - Easy to implement
  - Handling of joint limits

- Drawbacks
  - Slow convergence
  - Bad distribution of the adaptation
  - Unnatural posture



⇒ First joints are more modified than the following ones!

CCD

- First solution: use damping
  - Threshold on the variation of the joint parameters
  - Minimizes the adaptation of each joint
  - But increases the number of iterations

- More homogeneous adaptation
  ⇒ Bigger computation cost

- CCD is not suitable for postural adaptation of humanoids
- Our goal
  - Find natural postures
  - Computation time compatible with interactive animation of hundreds of characters

Comparison of analytic and iterative IK

- Analytic is best suited for simple case like isolated arm, leg, etc…
- Iterative is more general but requires multiple steps to converge towards the solution
  - Due to the non-linearity of the problem
  - If big steps are used, it becomes unstable

  - Or due to solving only for one DOF at a time (CCD)

Decomposition aligned with the z axis

Simpler algorithm:

Given rotation q_r

$$q_r = q_{twist} q_{swing}$$

Algorithm:

q_{twist} = normalize(Quaternion(0, 0, qr.z, qr.w);

q_{swing} = qr * conjugate(qt);

Reminder

Constraints. Twist

In the previous project, add the script constraintsTwist (found in intranet).

1. Use and complete that script to cancel the twist in the mirror joint
2. Extend the script so it has a minimum and a maximum angle
3. Apply the rotation limit script to the humanoid wrist (.fbx found in intranet)

Rotate a vector. Reminder

Intro.

Rotating a vector **p** by a quaternion **q** is:

$$p' = qpq^*$$

However,in Unity, given

```
Vector3 p1;
Quaternion q;
Vector3 p2;
```

We can write:

```
p2= q*p1;
```

(vector3) = (Quaternion) * (Vector3)

This does the following:

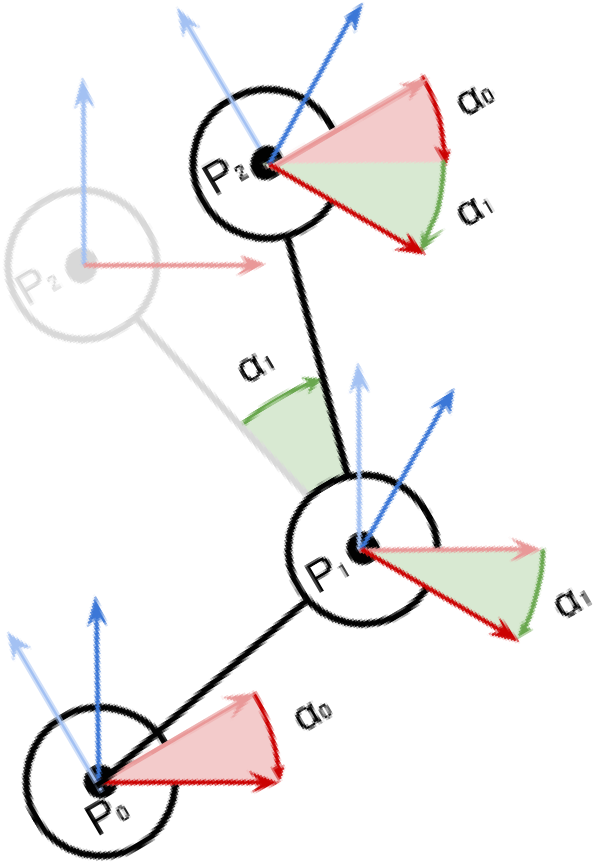
```
public static Vector3 operator *(Quaternion quat, Vector3 vec){
    float num = quat.x * 2f;     float num2 = quat.y * 2f;     float num3 = quat.z * 2f;
 float num4 = quat.x * num;     float num5 = quat.y * num2;     float num6 = quat.z * num3;
float num7 = quat.x * num2;     float num8 = quat.x * num3;     float num9 = quat.y * num3;
float num10 = quat.w * num;     float num11 = quat.w * num2;     float num12 = quat.w *num3;
    Vector3 result;
    result.x = (1f - (num5 + num6)) * vec.x + (num7 - num12) * vec.y + (num8 + num11) * vec.z;
    result.y = (num7 + num12) * vec.x + (1f - (num4 + num6)) * vec.y + (num9 - num10) * vec.z;
    result.z = (num8 - num11) * vec.x + (num9 + num10) * vec.y + (1f - (num4 + num5)) * vec.z;
    return result;
 }
```

https://answers.unity.com/questions/372371/multiply-quaternion-by-vector3-how-is-done.html

Inverse kinematics. The methods

3 methods:

- Gradient Descent (GD)
- Cyclic Coordinate Descent (CCD)
- **Forward and Backward Recursive Inverse Kinematic (FABRIK)**

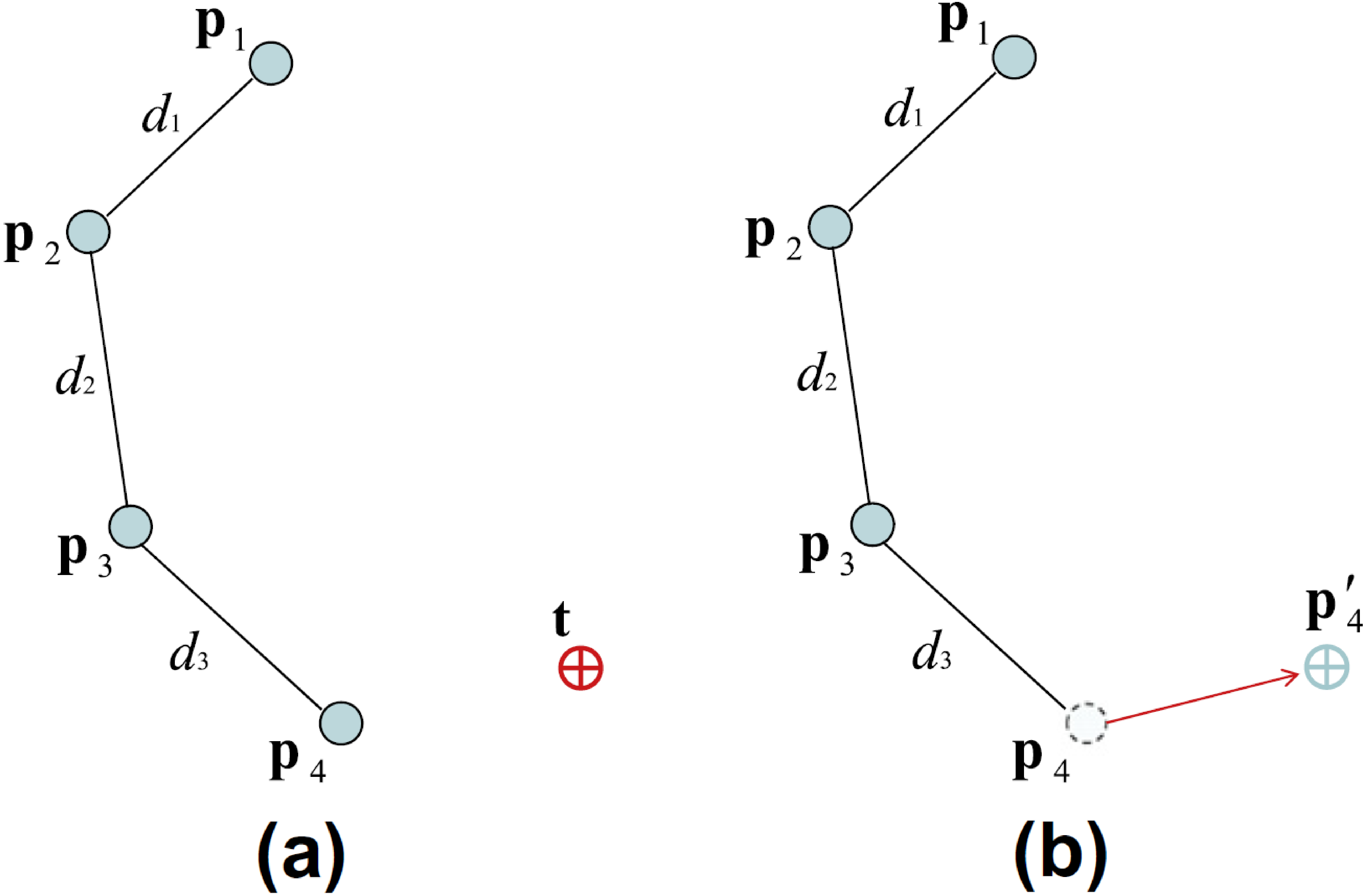


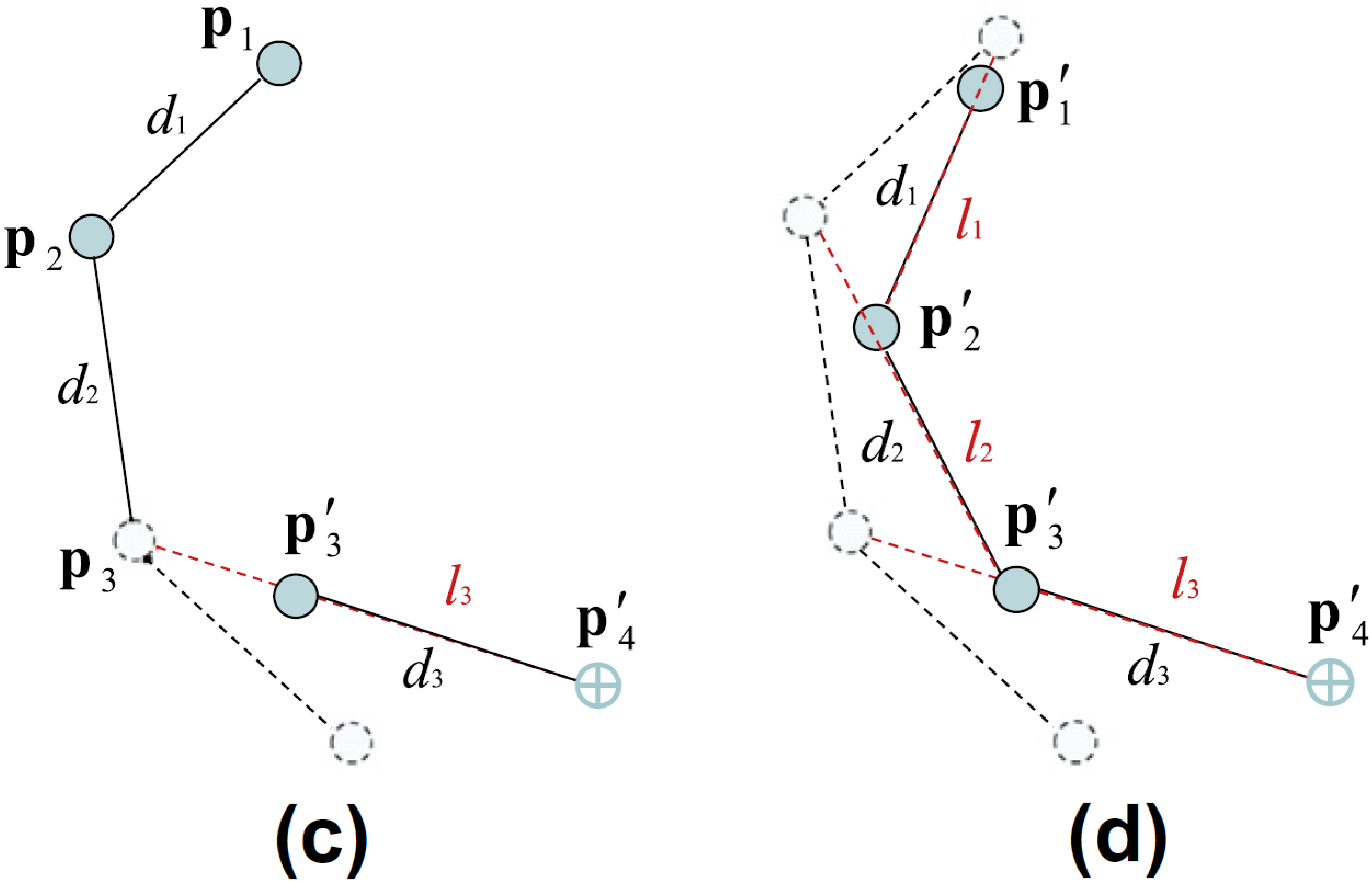
ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

UNIVERSITAT DE BARCELONA

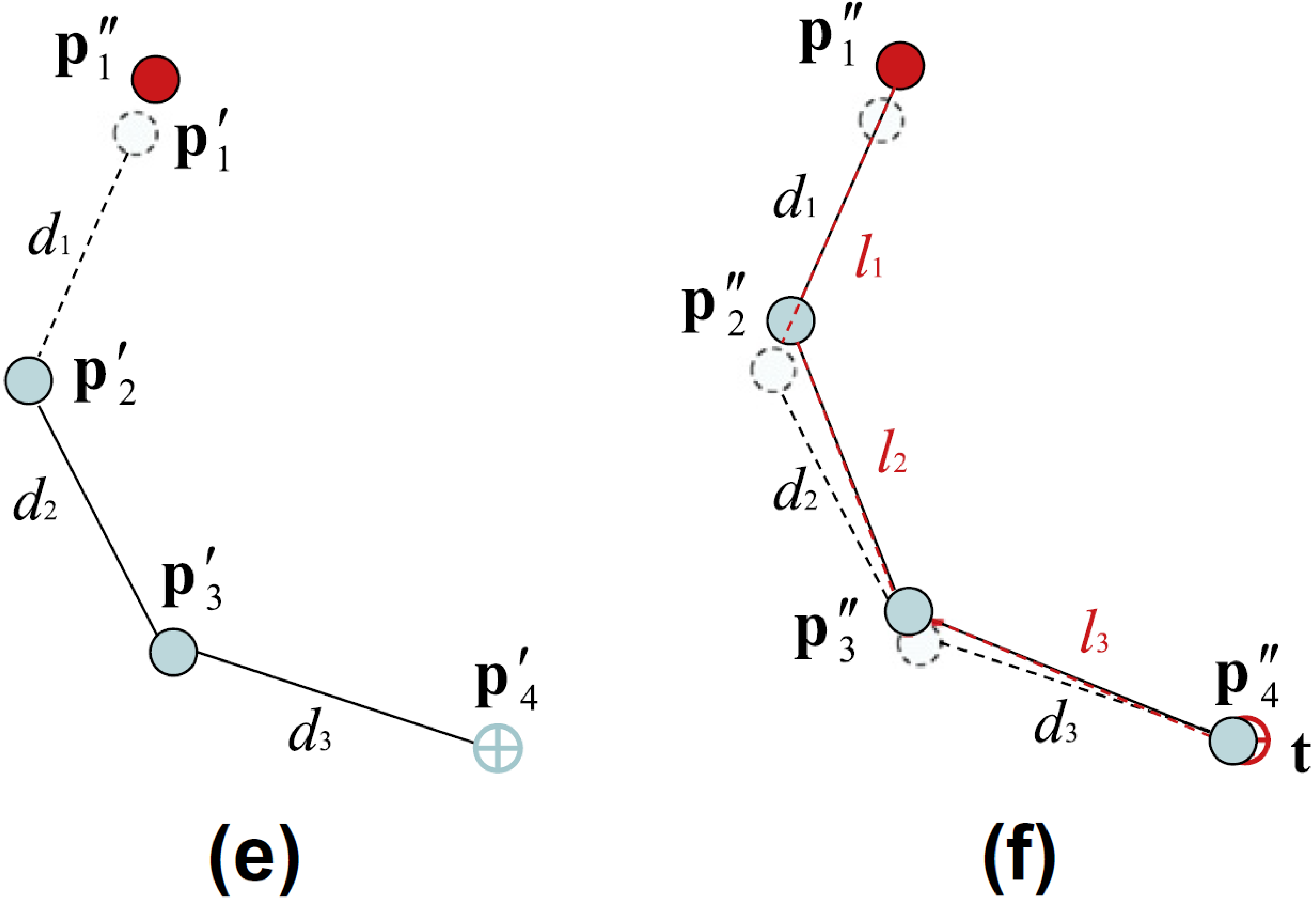
Fabrik

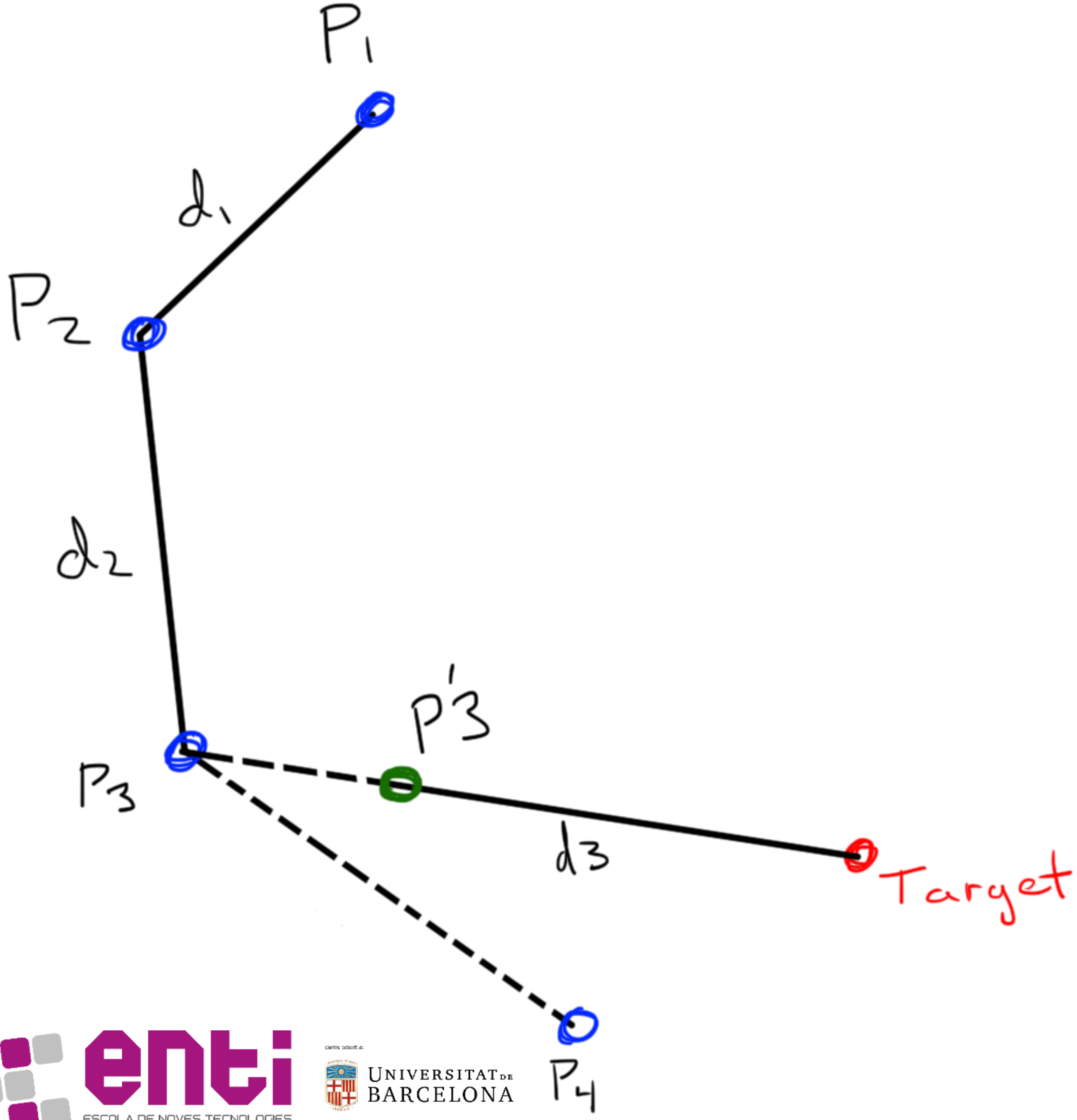
The basic algorithm


$\mathbf{p}_1$
d_1
$\mathbf{p}_2$
d_2
$\mathbf{p}_3$
d_3
$\mathbf{p}_4$
$\mathbf{t}$
(a)
$\mathbf{p}_1$
d_1
$\mathbf{p}_2$
d_2
$\mathbf{p}_3$
d_3
$\mathbf{p}_4$
$\mathbf{p}'_4$
(b)

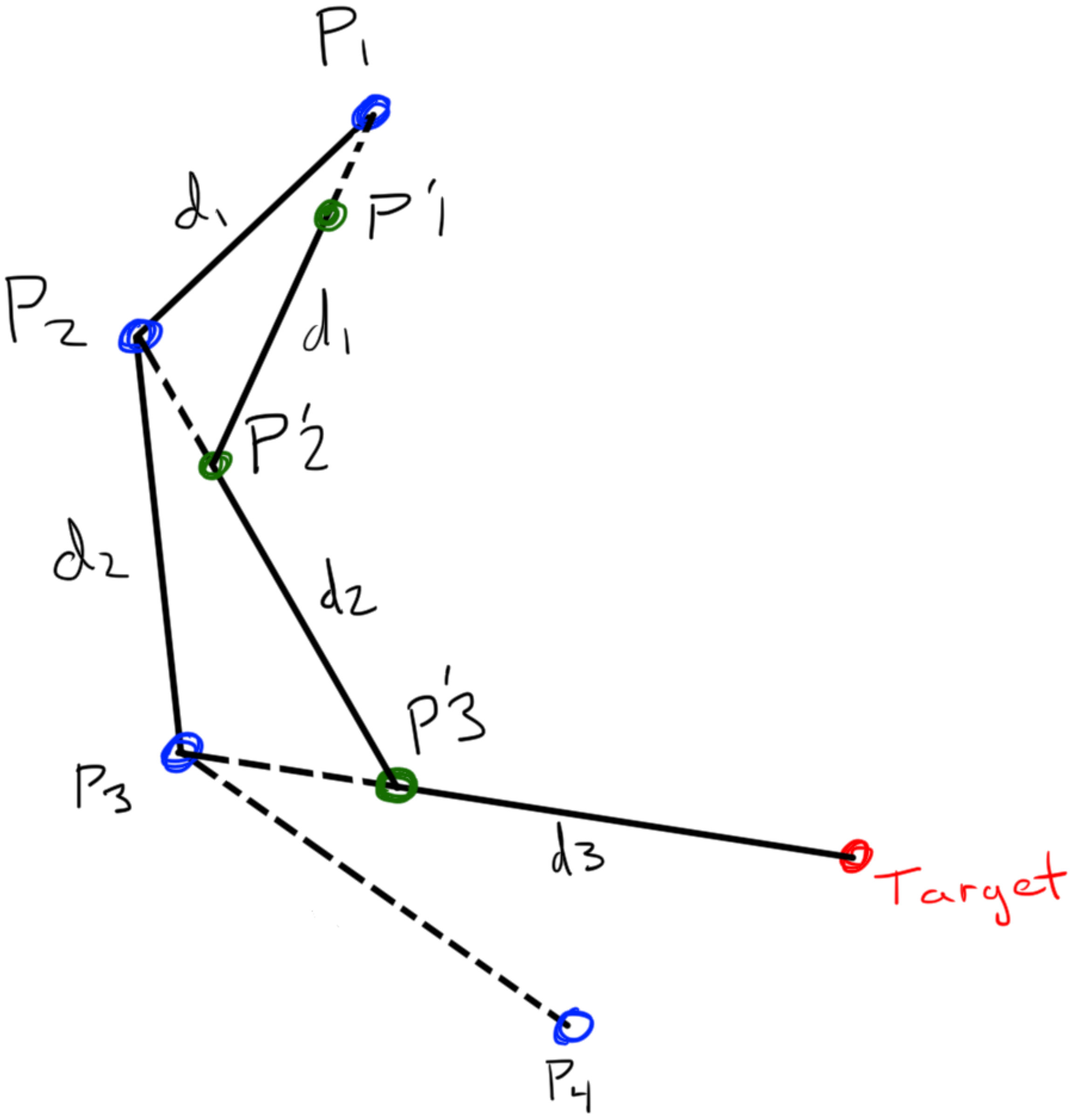


$\mathbf{p}_1$
d_1
$\mathbf{p}_2$
d_2
$\mathbf{p}_3$
$\mathbf{p}'_3$
l_3
d_3
$\mathbf{p}'_4$
(c)
$\mathbf{p}'_1$
d_1
l_1
$\mathbf{p}'_2$
d_2
l_2
$\mathbf{p}'_3$
l_3
d_3
$\mathbf{p}'_4$
(d)


$\mathbf{p}_1''$
$\mathbf{p}_1'$
d_1
$\mathbf{p}_2'$
d_2
$\mathbf{p}_3'$
d_3
$\mathbf{p}_4'$
(e)
$\mathbf{p}_1''$
d_1
l_1
$\mathbf{p}_2''$
l_2
d_2
$\mathbf{p}_3''$
l_3
d_3
$\mathbf{p}_4''$
$\mathbf{t}$
(f)

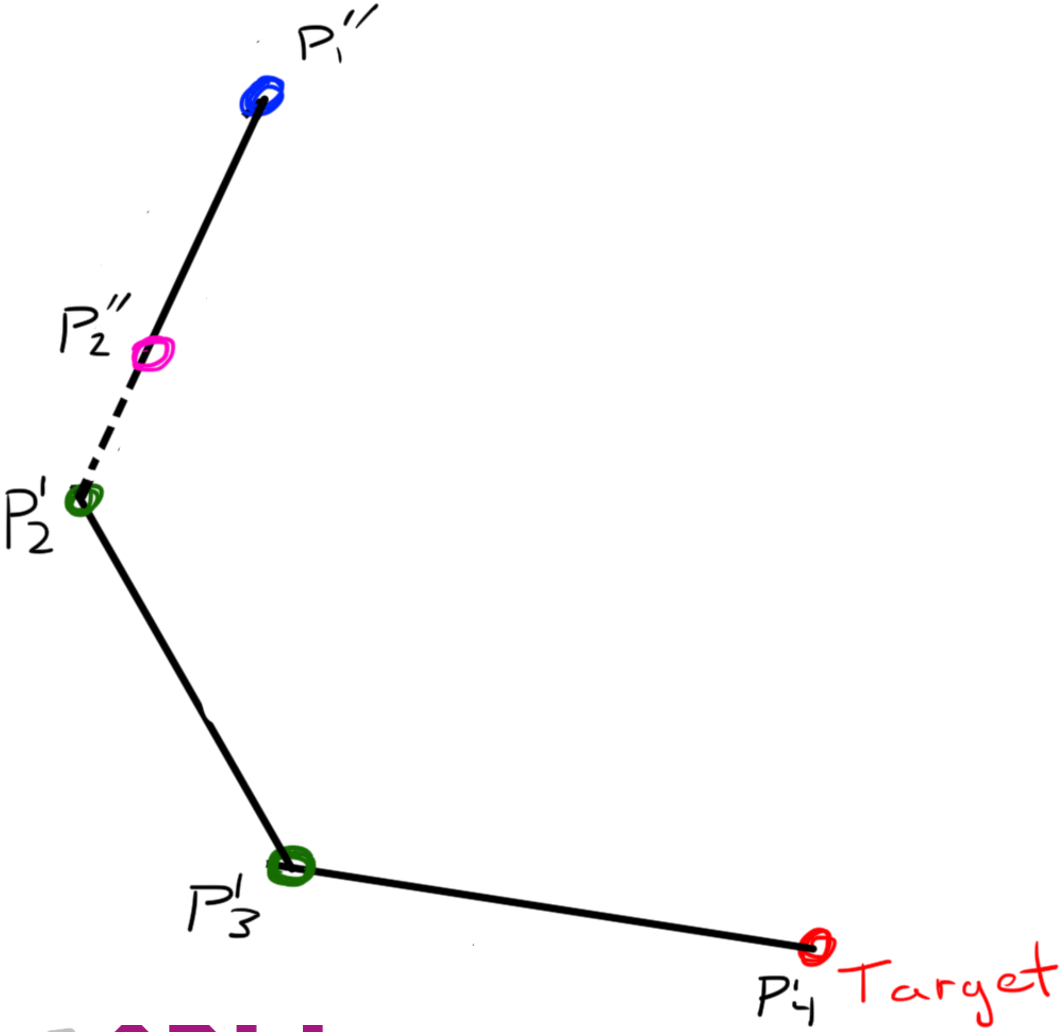


Moving one point

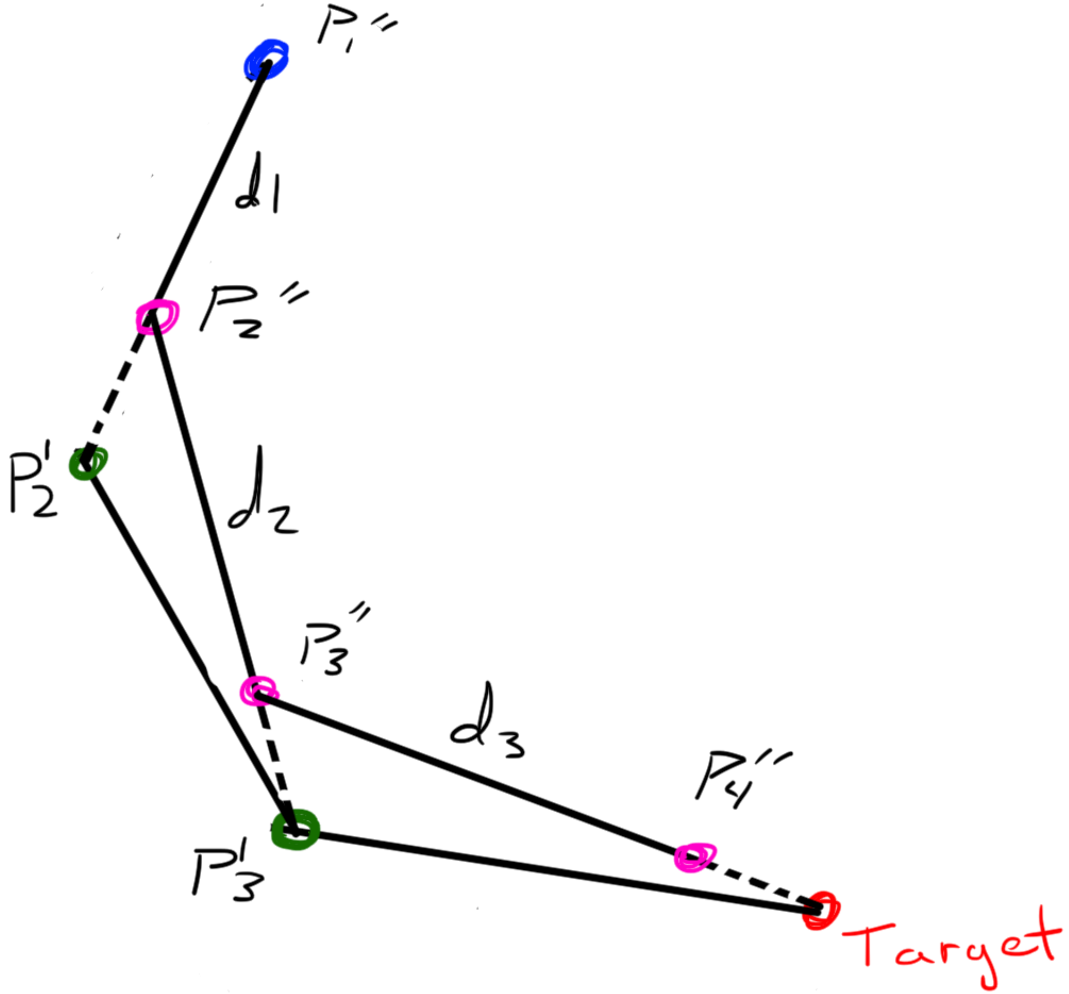


Moving the whole chain

Moving one point



Moving the whole chain



Benefits

A. Aristidou, J. Lasenby / Graphical Models 73 (2011) 243–260

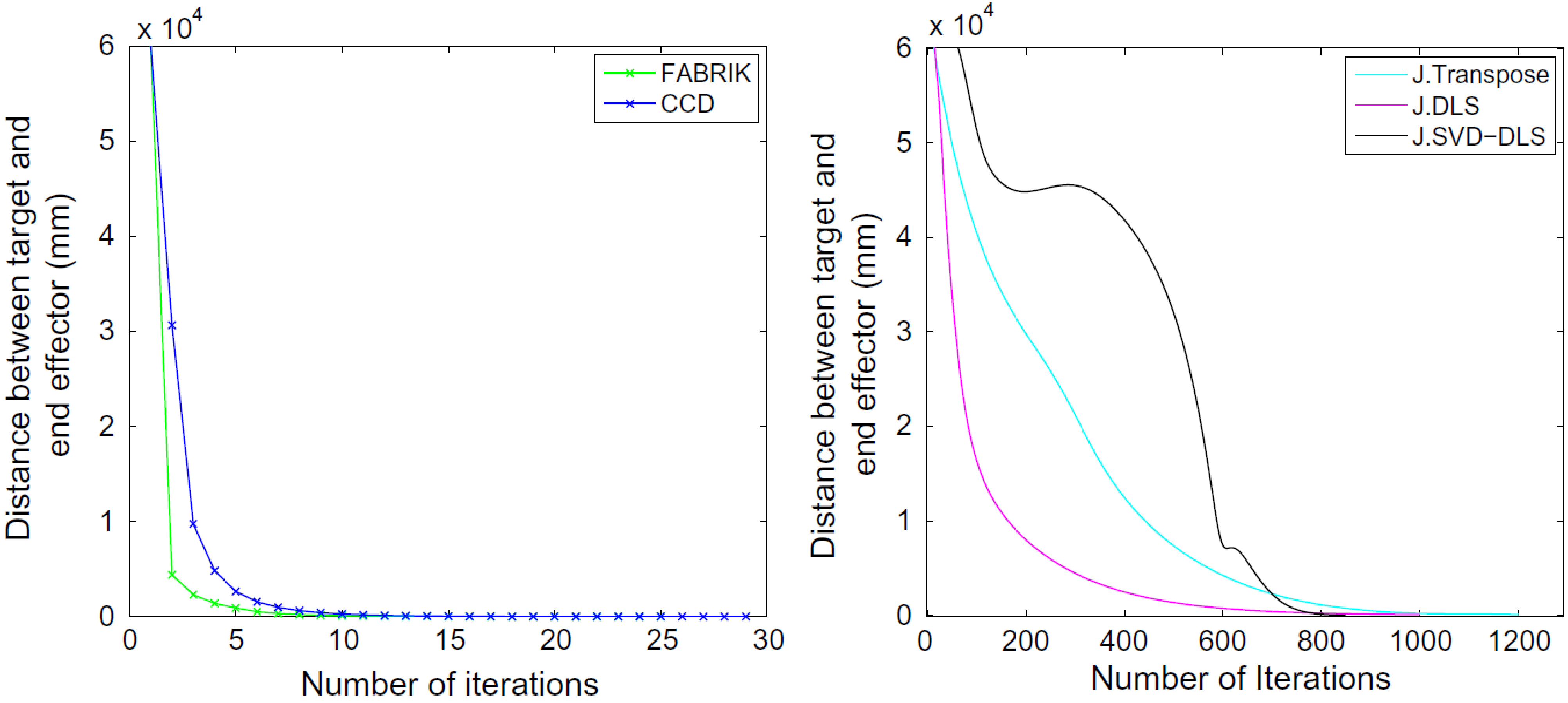
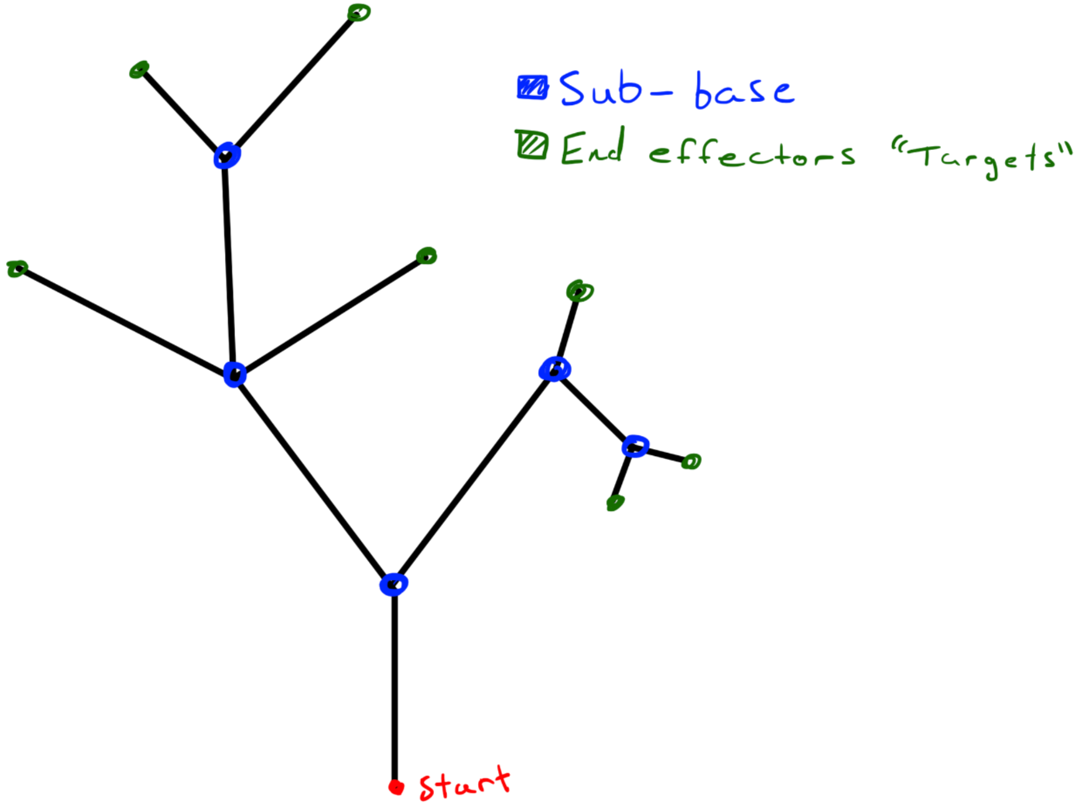


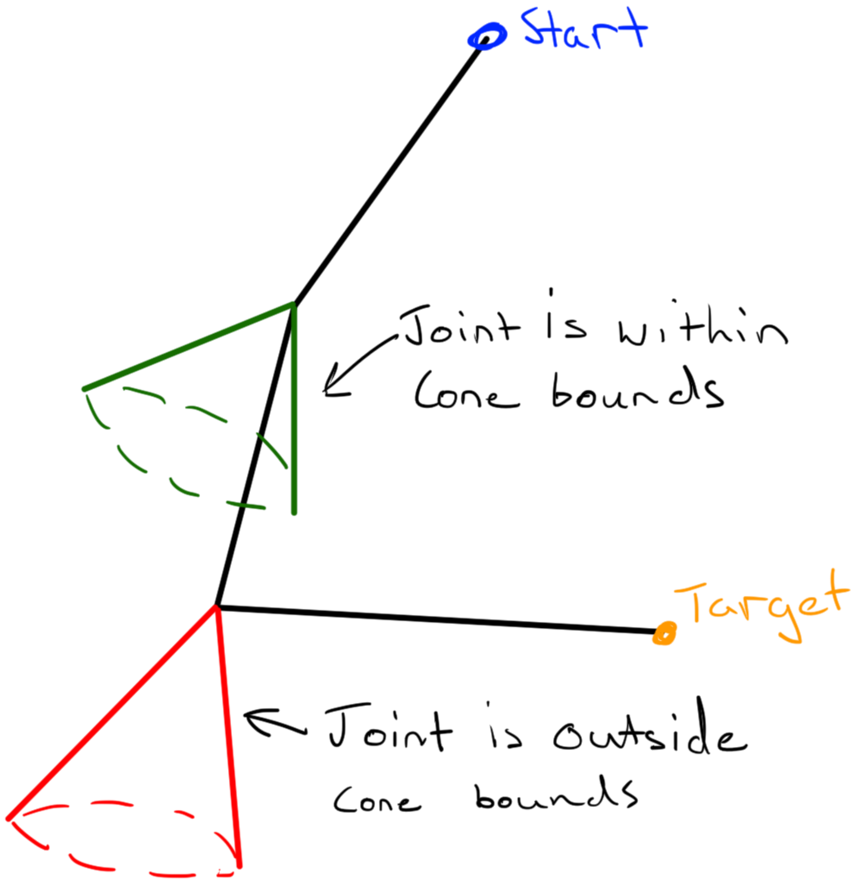
Fig. 11. The number of iterations needed to reach the target against the distance between target and end effector as this changes over time.

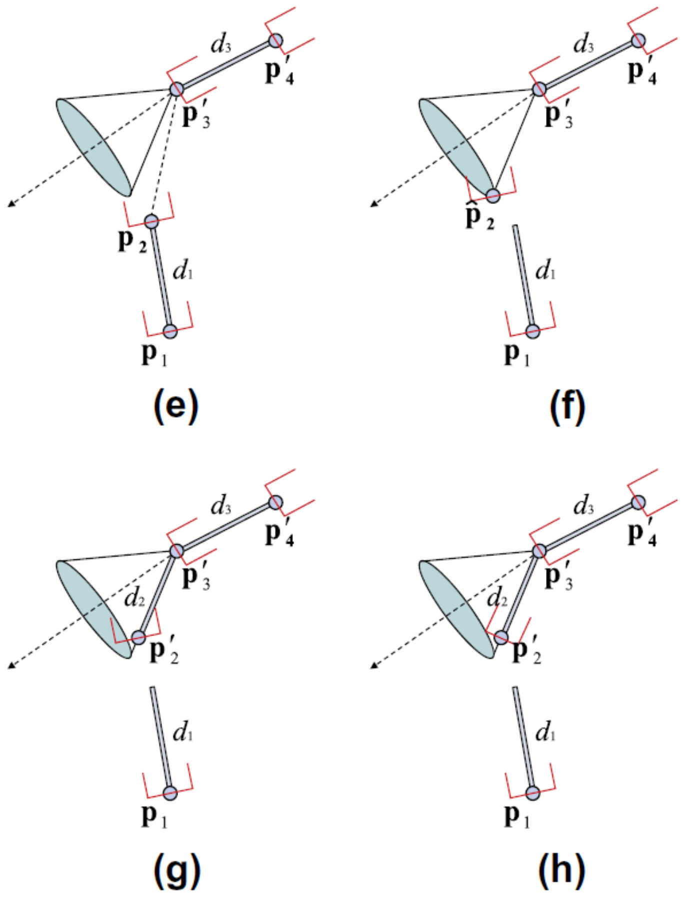
Benefits (2)

- It converges fast (see previous slide)
- It works well with multiple end-effectors (think of the algorithm, a sub-chain can be analysed independently)
- Many good resources online (see final slide)



Drawbacks





- FABRIK does NOT preserve the chain integrity
- Therefore, constraints in FABRIK are tricky (quite more tricky than in CCD!)

FABRIK. See also

- Blog description:

https://developer.roblox.com/articles/Inverse-Kinematics-for-Animation#FABRIK

- Video (25 minutes)

https://www.youtube.com/watch?time_continue=2&v=UNoX65PRehA

- Web from the author (very detailed!), with links to implementations

http://www.andreasaristidou.com/FABRIK.html