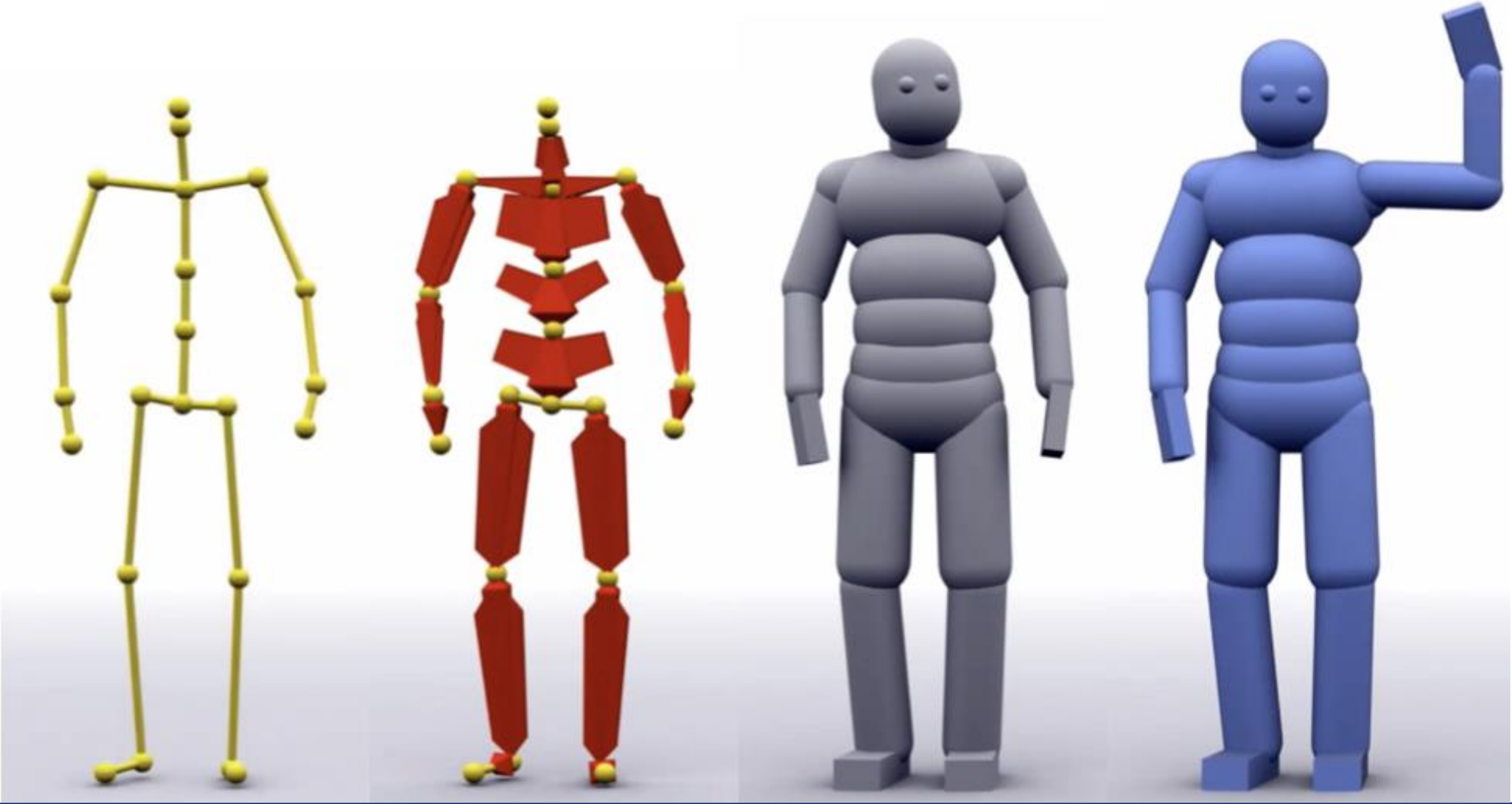


Kinematic Models of Motion

Forward Kinematics

Learning objectives

- Learn how to model virtual characters and robots as articulated structures
- Learn how to parameterize poses and get introduced to some of the most common kinematic motion models
- Understand how to solve forward kinematics problems

Human and animal motions – it’s just life exploiting the laws of physics!

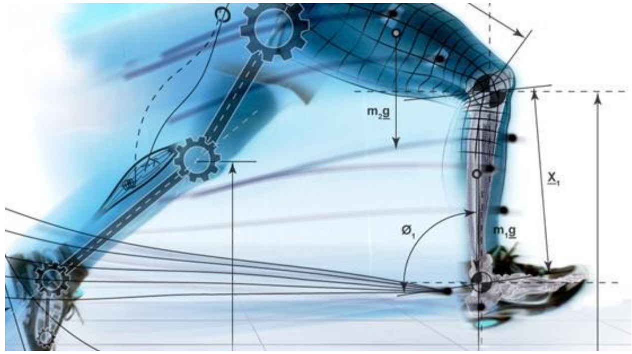
Modeling human and animal motions

- What are the fundamental principles that lead to the motions we see in nature?
  - How can we create virtual agents and robots that can move as skillfully as their biological counterparts?


intent + perception


neural excitation

muscle contractions


physics


motions

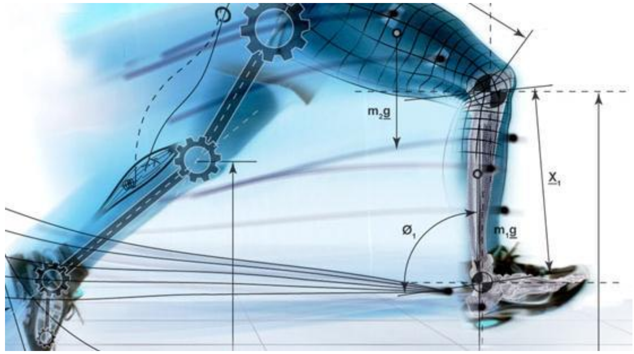
Modeling human and animal motions

- What are the fundamental principles that give rise to human and animal motions?
  - How can we create virtual agents and robots that can move as skillfully as their biological counterparts?

intent + perception

neural excitation

muscle contractions



physics

motions

Controller

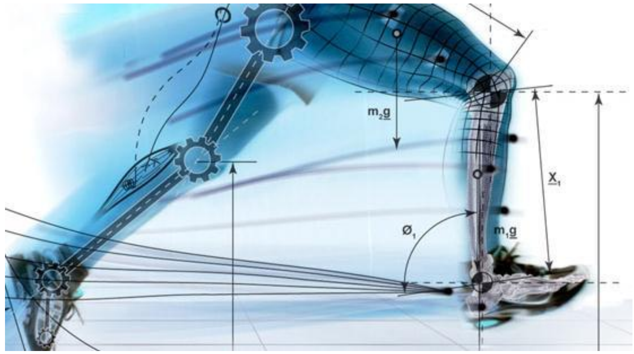
Modeling human and animal motions

- What are the fundamental principles that give rise to human and animal motions?
  - How can we create virtual agents and robots that can move as skillfully as their biological counterparts?

intent + perception

neural excitation

muscle contractions



physics

motions

Dynamics

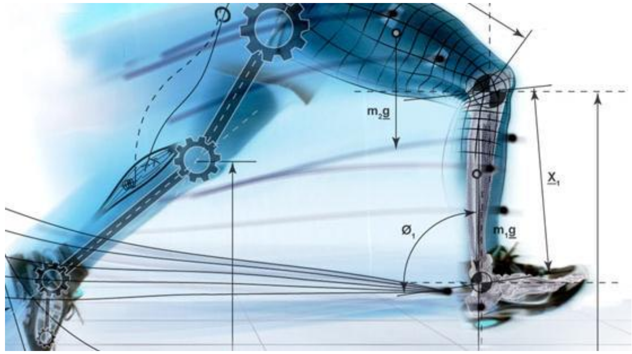
Modeling human and animal motions

- What are the fundamental principles that give rise to human and animal motions?
  - How can we create virtual agents and robots that can move as skillfully as their biological counterparts?

intent + perception

neural excitation

muscle contractions

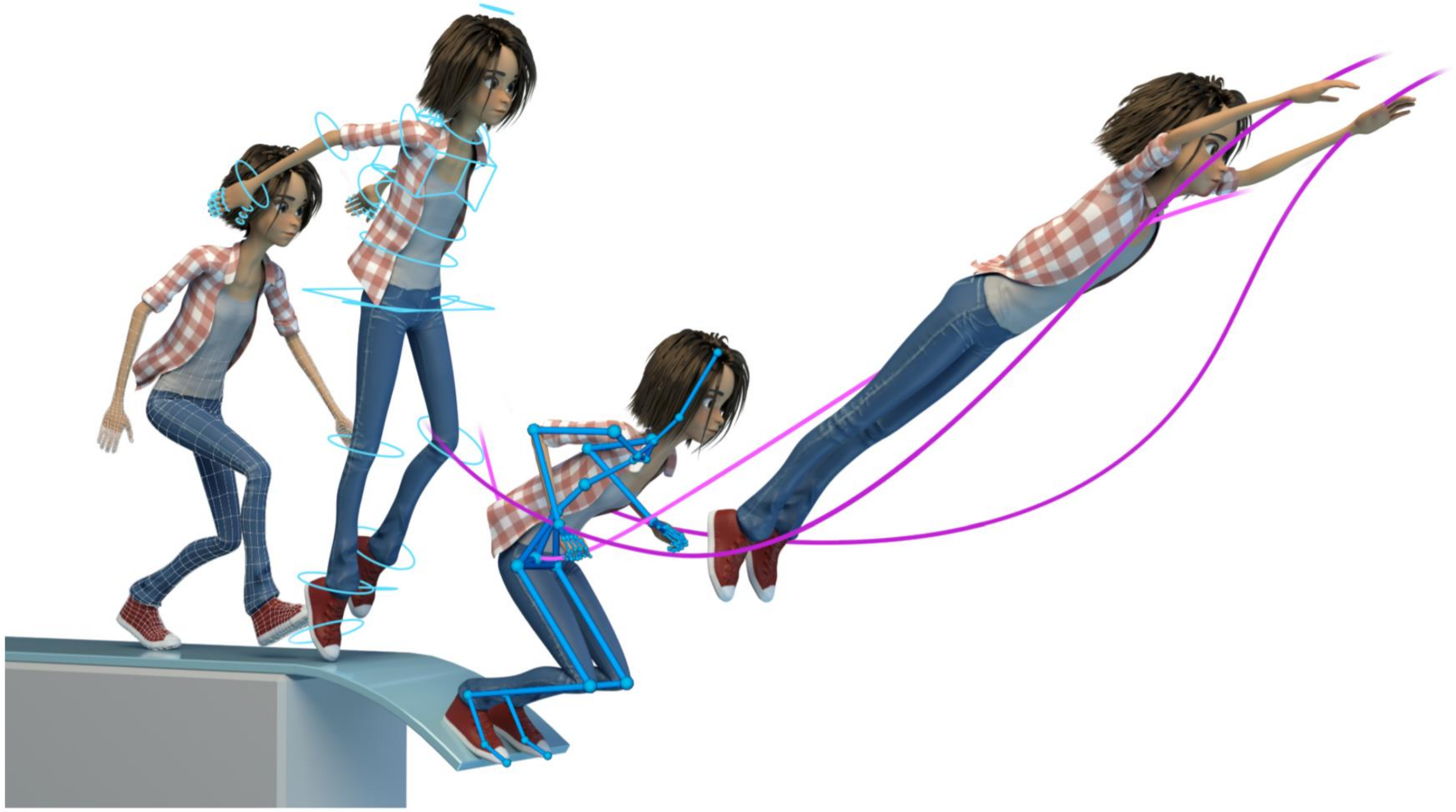


physics

motions

Kinematics

Kinematic models of motion



kinematics

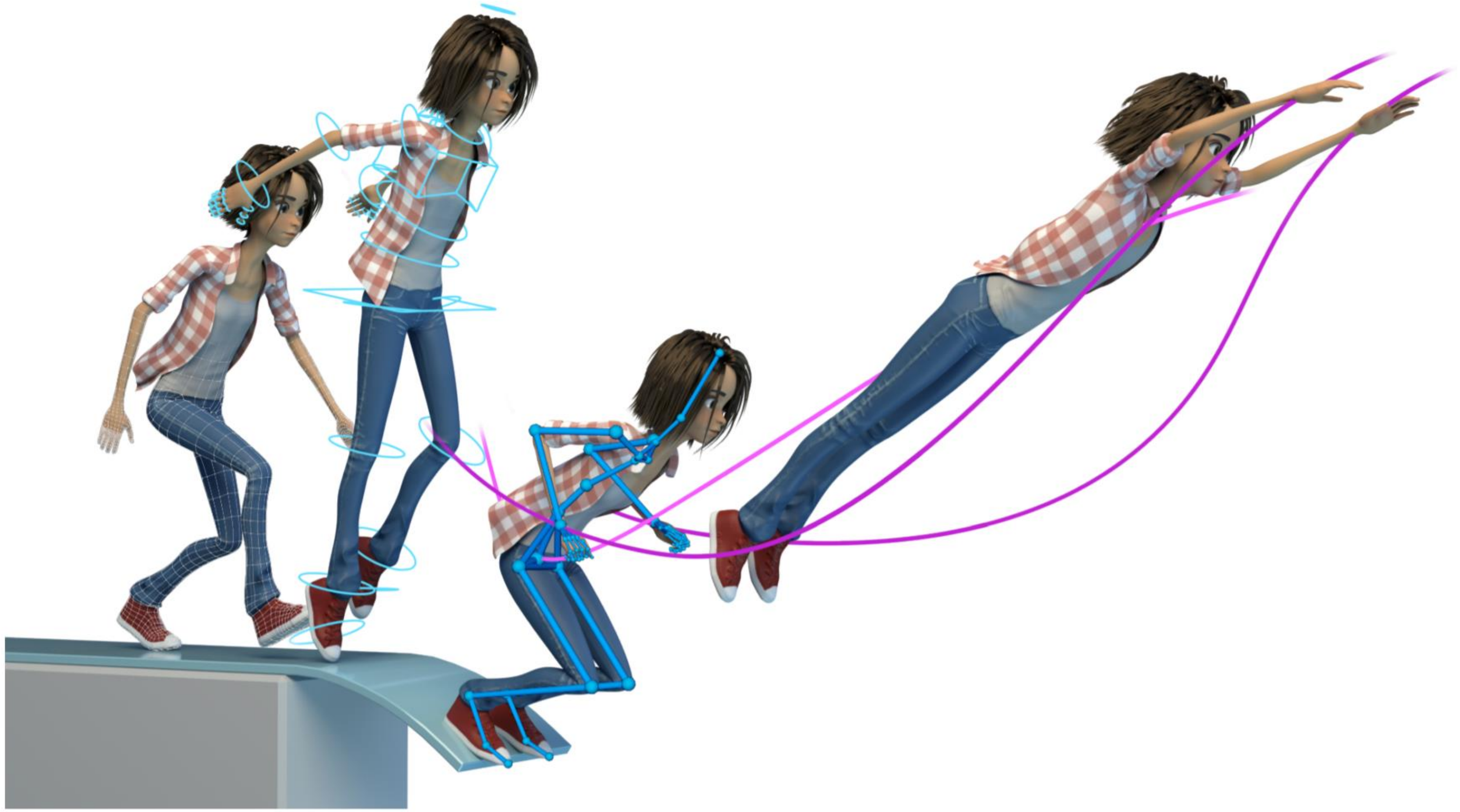
/ˌkɪnɪˈmatɪks, ˌkʌɪnɪˈmatɪks/

noun

the branch of mechanics concerned with the motion of objects without reference to the forces which cause the motion.

- the features or properties of motion in an object.
  plural noun: **kinematics**

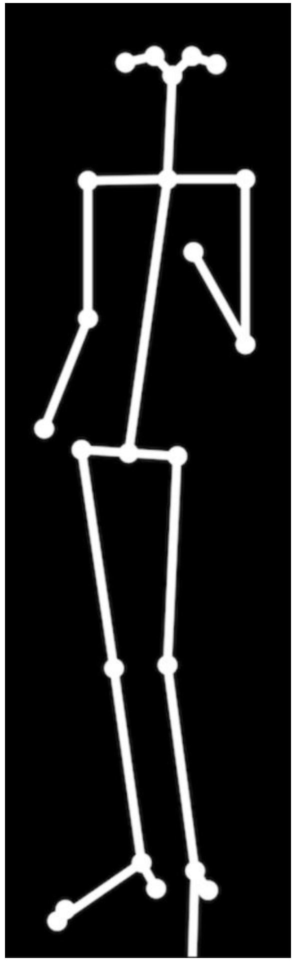
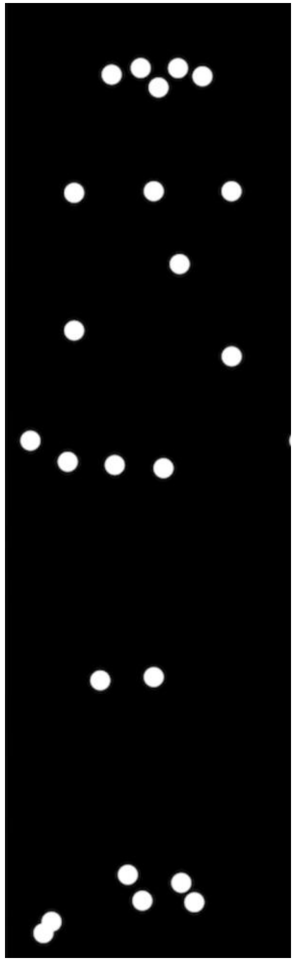
Kinematic models of motion



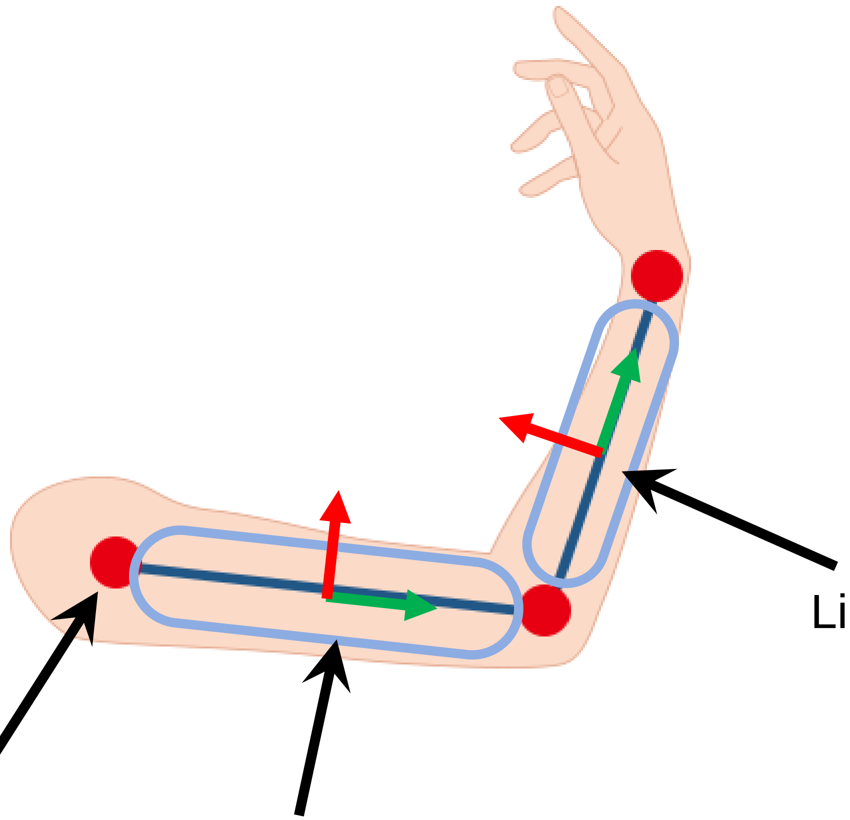
- Model motions as a sequence of **poses** over time.
- Although “causes” of motion are not considered, kinematic models are nevertheless very important in analysis, synthesis and control.

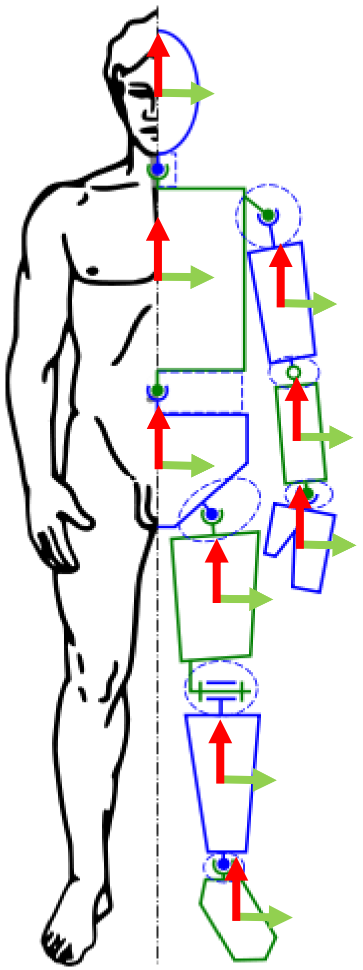
What is a pose?



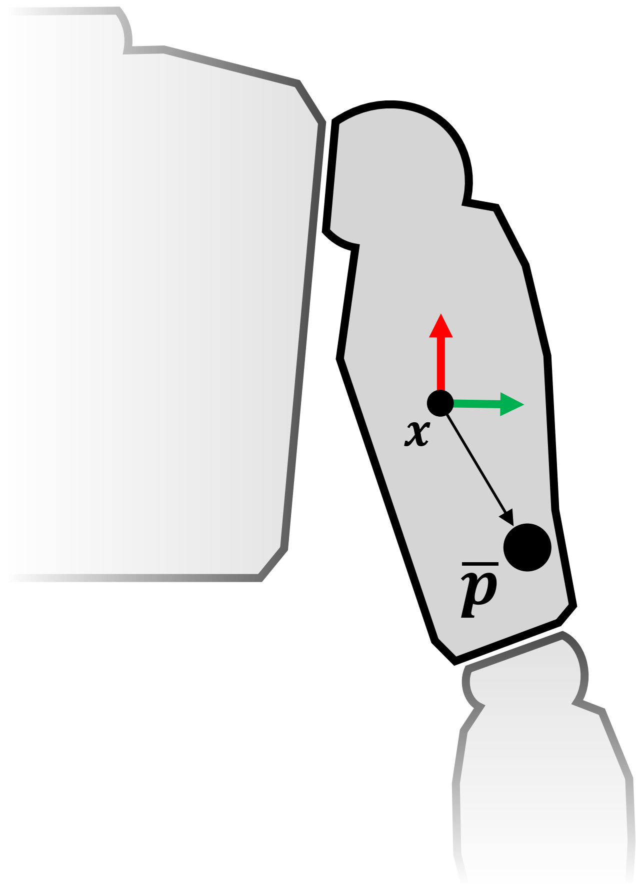
How do we model a pose?





Virtual characters and robots are very often modeled as ***articulated rigid body systems***

Modeling articulated rigid body systems



- Local coordinate frame of each body part described by:
  - Position of center of mass (COM) $\boldsymbol{x}$, and rotation ${}_{W}\boldsymbol{R}_{b}$
- Can now easily talk about points and vectors specified in different coordinate frames. For example:
  - $\bar{\boldsymbol{p}}$: coordinates of a point in the local frame of reference. Center of mass (aka origin) has coordinates (0,0,0) in this local frame of reference.
    - What do the coordinates of point $\bar{\boldsymbol{p}}$ *mean*?
  - Same point, now expressed in world coordinates:

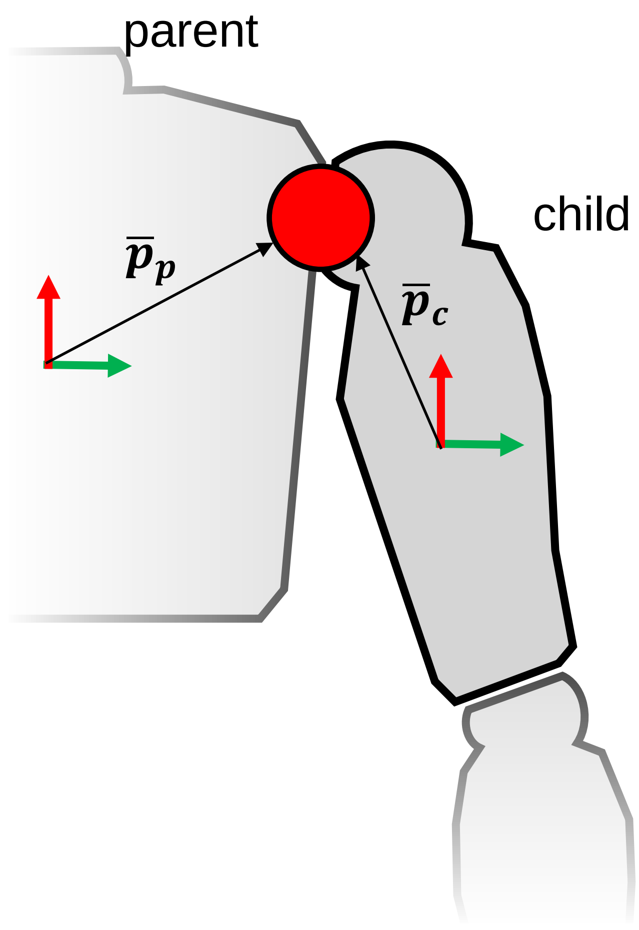
$$\boldsymbol{p} = \boldsymbol{x} + {}_{W}\boldsymbol{R}_{b}\bar{\boldsymbol{p}}$$

Add resulting world-coords vector to position of COM

rotate vector from origin to $\bar{\boldsymbol{p}}$ to bring it to world coordinates

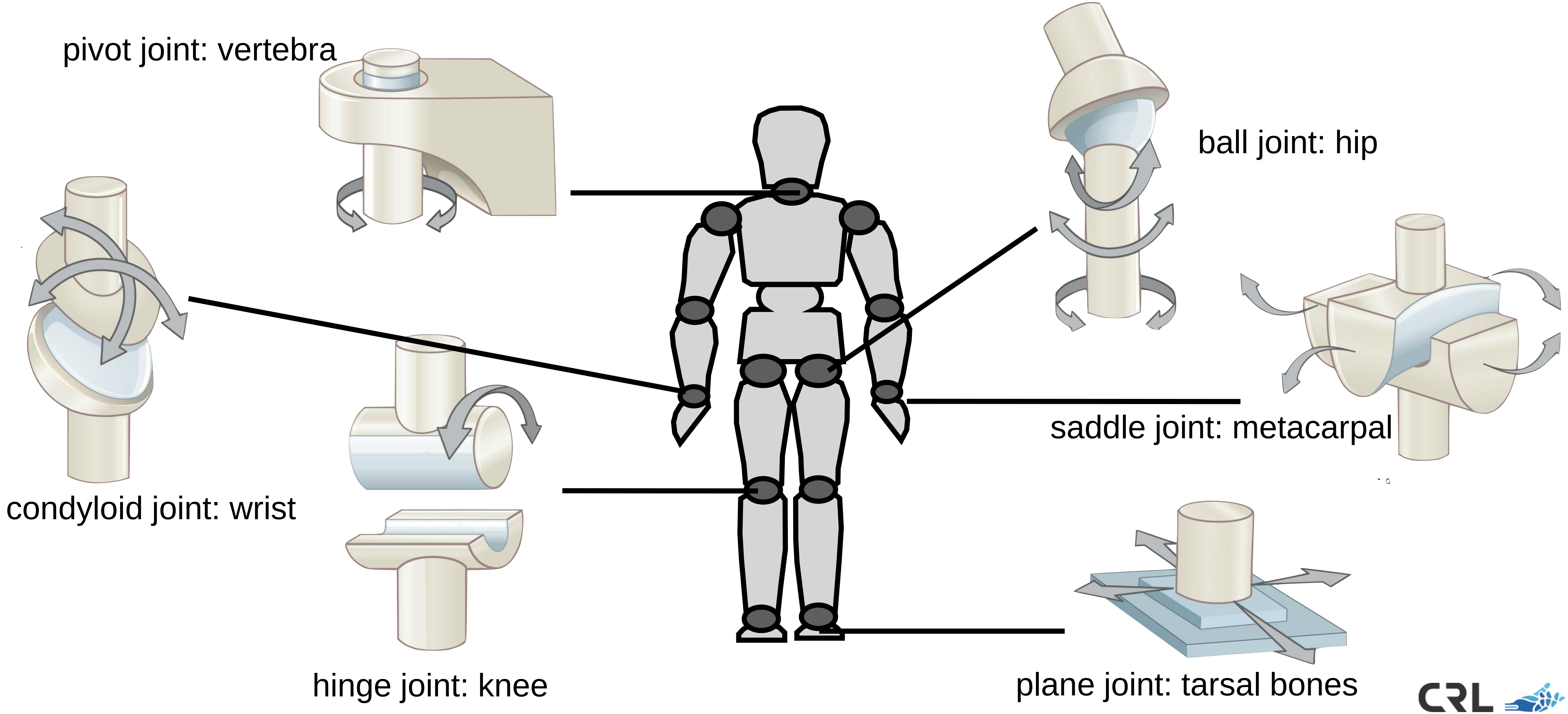
Alternatively, you can think of this as a sequence of transforms: first a rotation, then a translation. Please check lectures from Visual Computing if you need to review transforms, rotation matrices, etc.

Modeling articulated rigid body systems



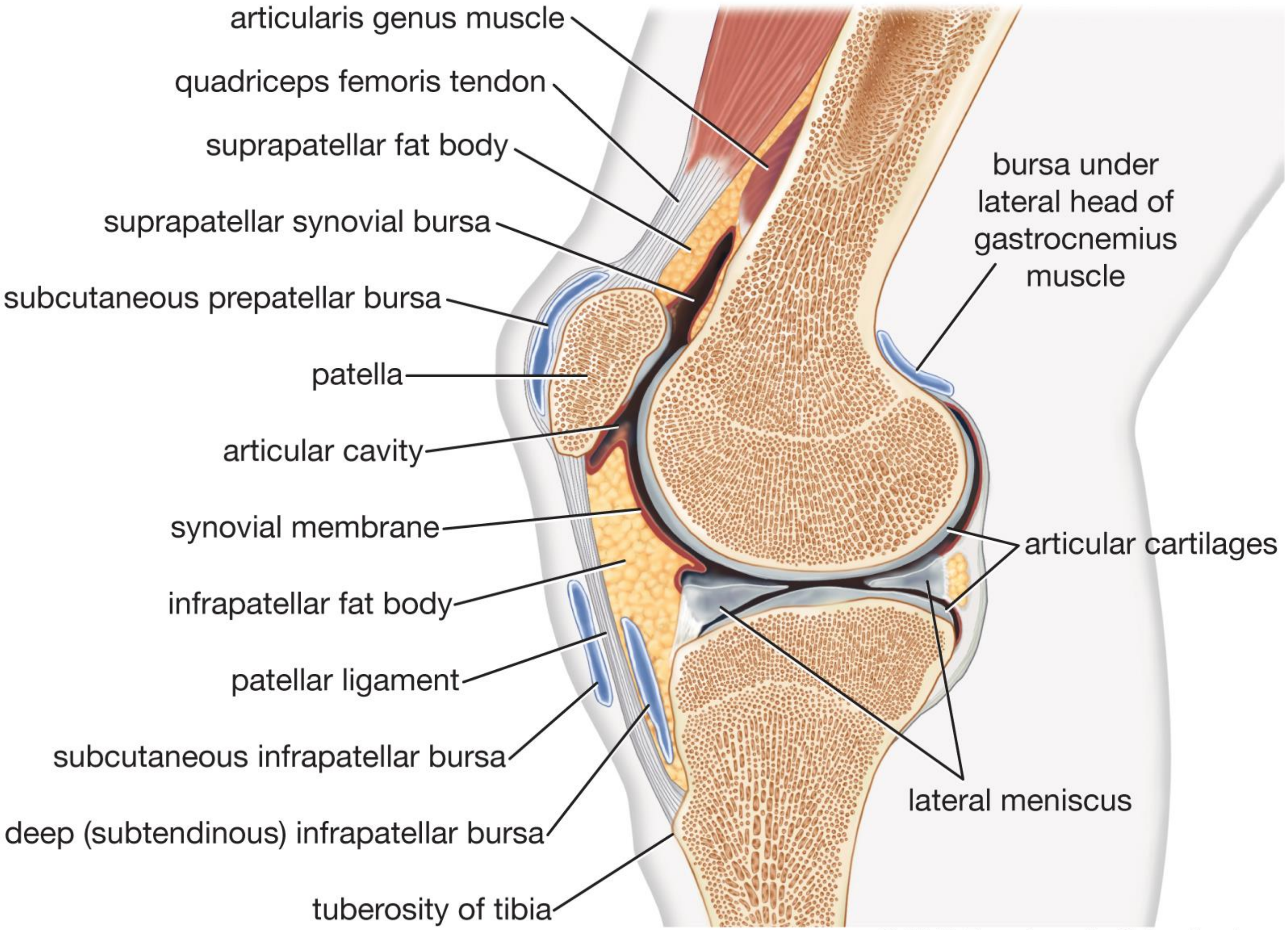
- Joints:
  - used to connect pairs of rigid body parts – a parent and a child
  - Defined by:
    - $\bar{\boldsymbol{p}}_c$: position of joint in local coordinate frame of the child body
    - $\bar{\boldsymbol{p}}_p$: position of joint in local coordinate frame of the parent body
    - Type and properties of joint

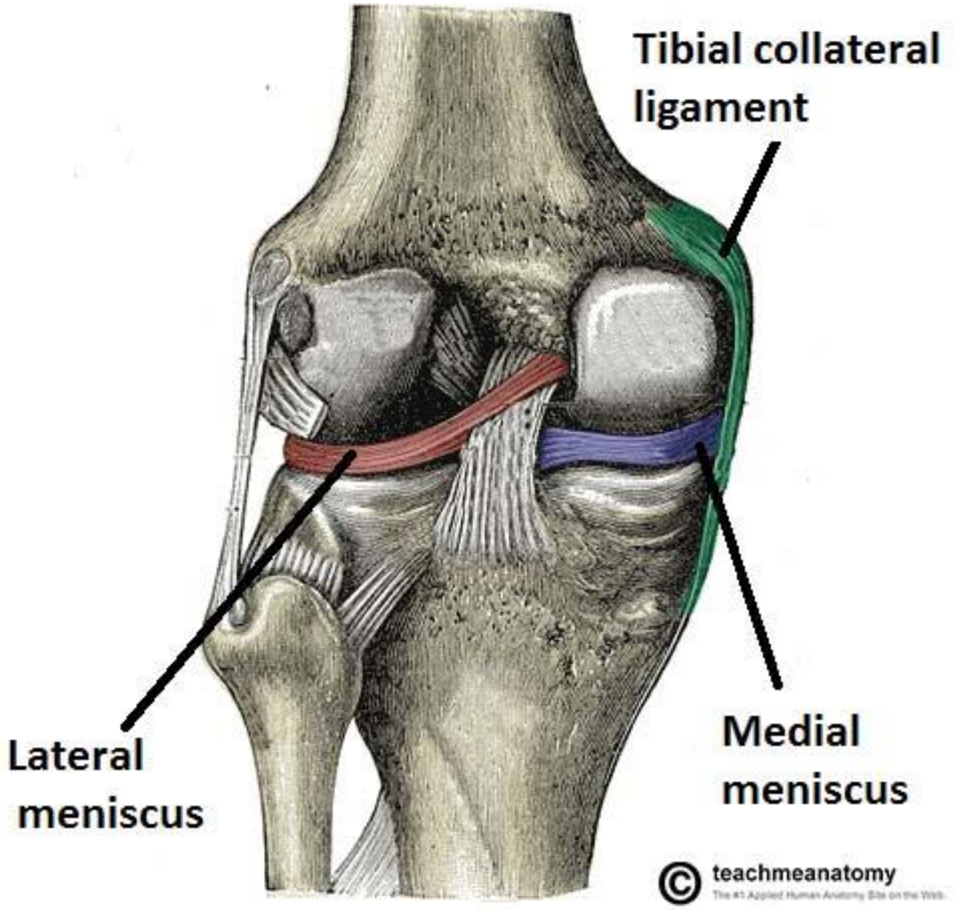
Modeling the human body



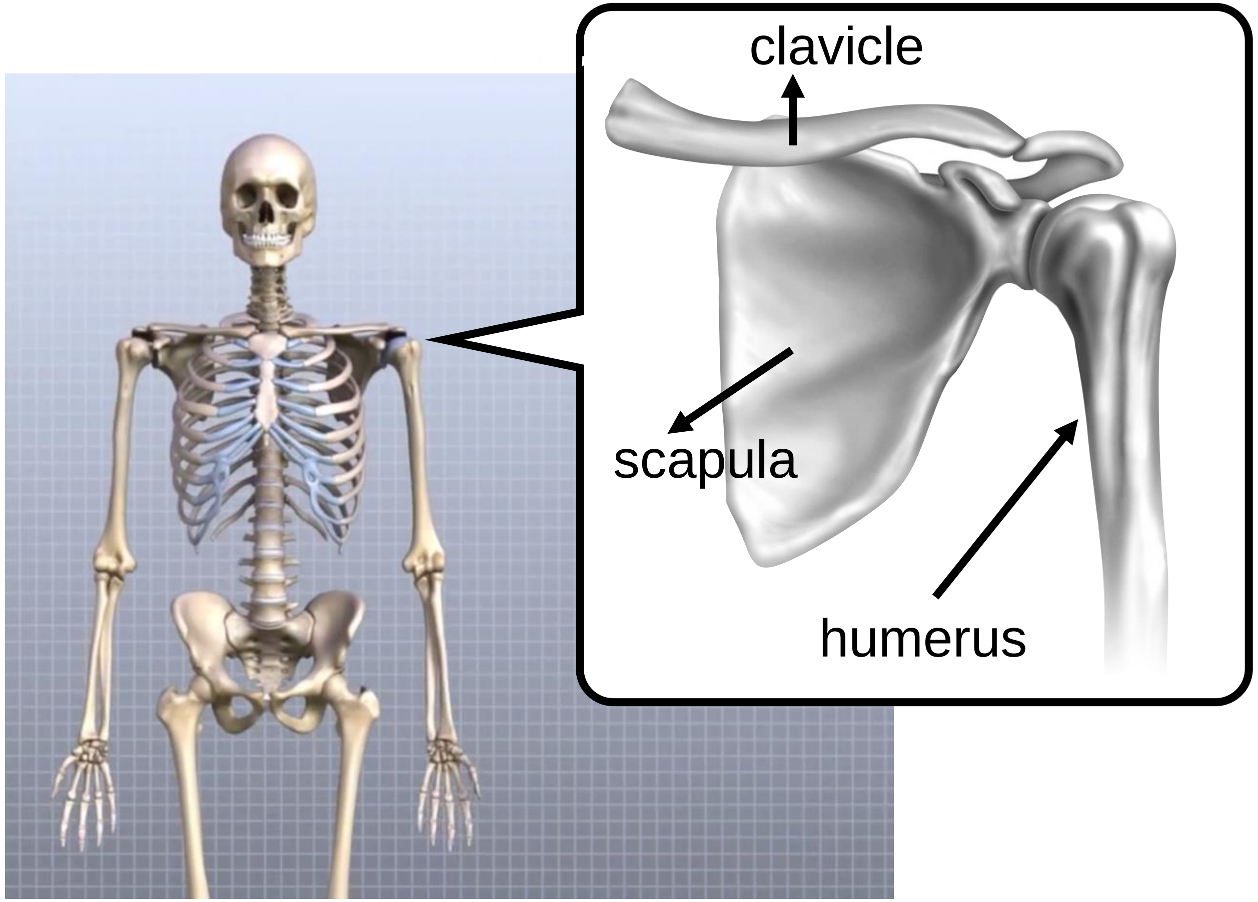
Anatomy of joints

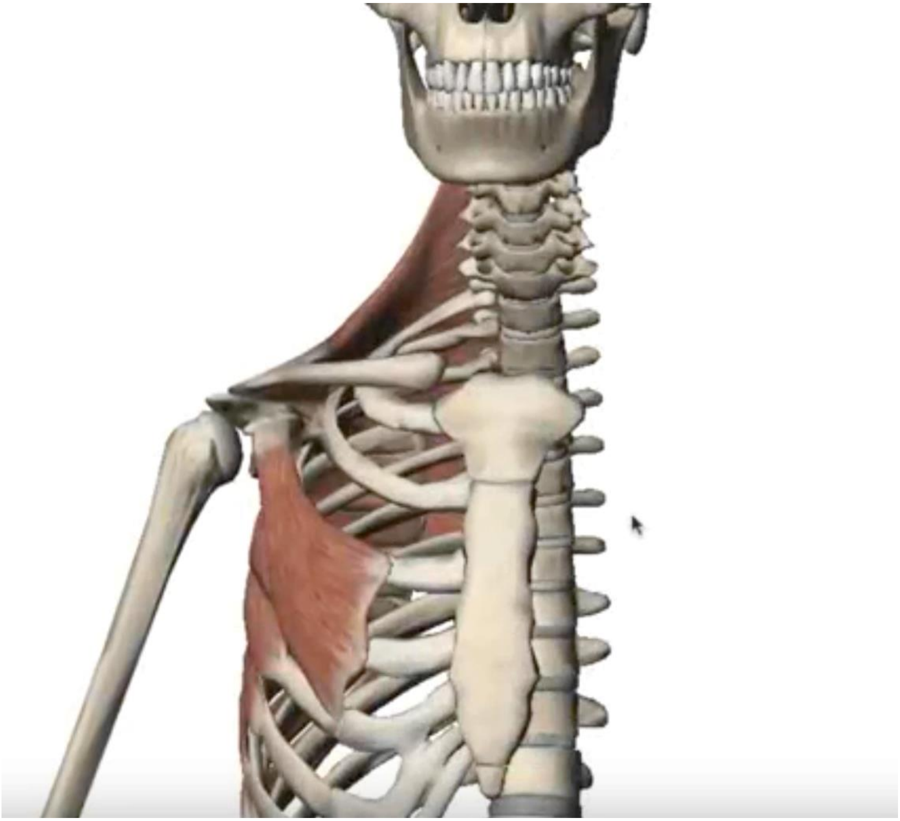
Knee
parasagittal section-lateral to midline of knee



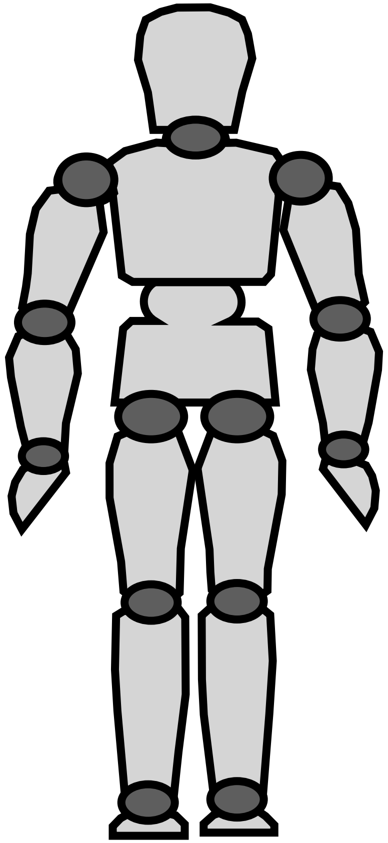


Anatomy of joints

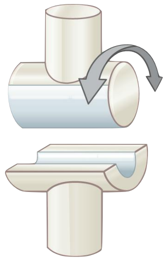




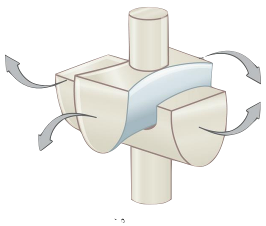
Parameterizing human model



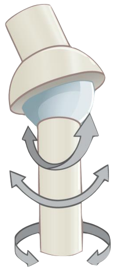
Most commonly used types of joints:



1-DOF joint

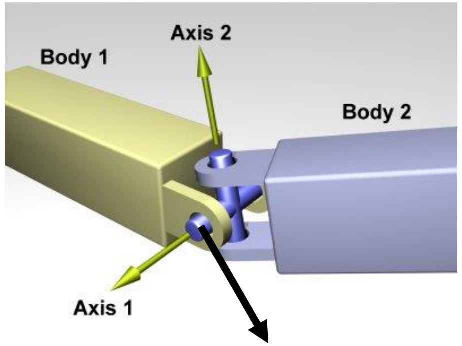


2-DOF joint



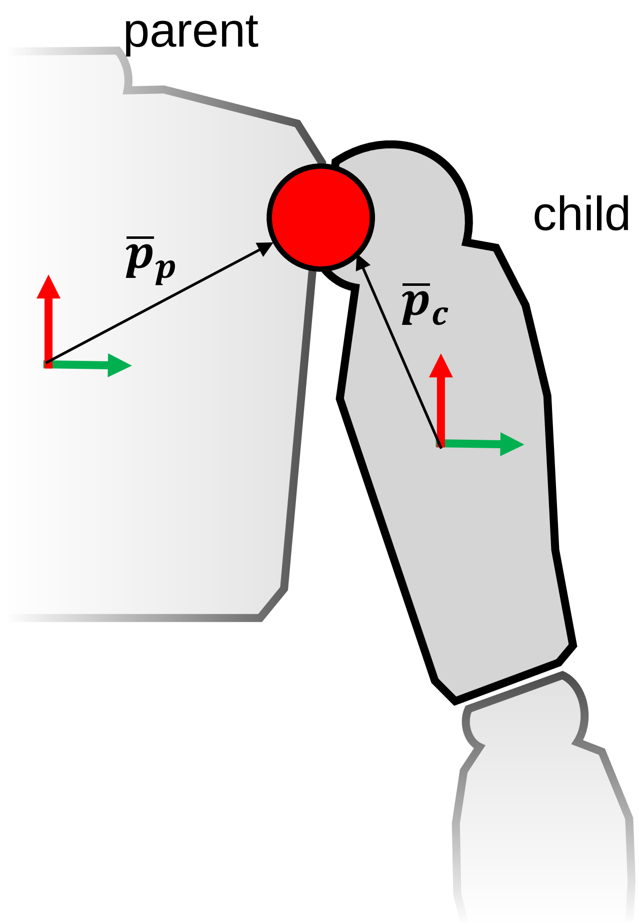
3-DOF joint

- But 1-DOF joints can already take us a long way, as they can be composed in various ways.
- For example, a 2-DOF joint can be modeled like this:



Auxiliary rigid body connected via
1-DOF joints to body 1 and body 2.

Modeling articulated rigid body systems



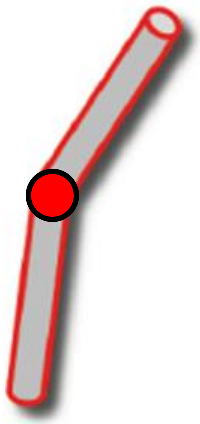
- Joints:
  - used to connect pairs of rigid body parts – a parent and a child
  - defined by:
    - $\overline{\boldsymbol{p}}_c$: position of joint in local coordinate frame of the *child* body
    - $\overline{\boldsymbol{p}}_p$: position of joint in local coordinate frame of the *parent* body
    - Joint rotation axis: $\overline{\boldsymbol{v}}$ (we will work with only 1-DOF joints in this course)
      - And we will assume coordinate frames of child and parents always align when relative rotation about joint axis is 0 (at rest).
  - Joint leads to a relative change in orientation between parent and child bodies:

$$ {}_p\boldsymbol{R}_c = {}_w\boldsymbol{R}_p{}^{-1} * {}_w\boldsymbol{R}_c = {}_p\boldsymbol{R}_w * {}_w\boldsymbol{R}_c $$

- Body parts have:
  - one parent joint, which is NULL only for the *root*
  - zero or more child joints (*end effector* == no child joints)

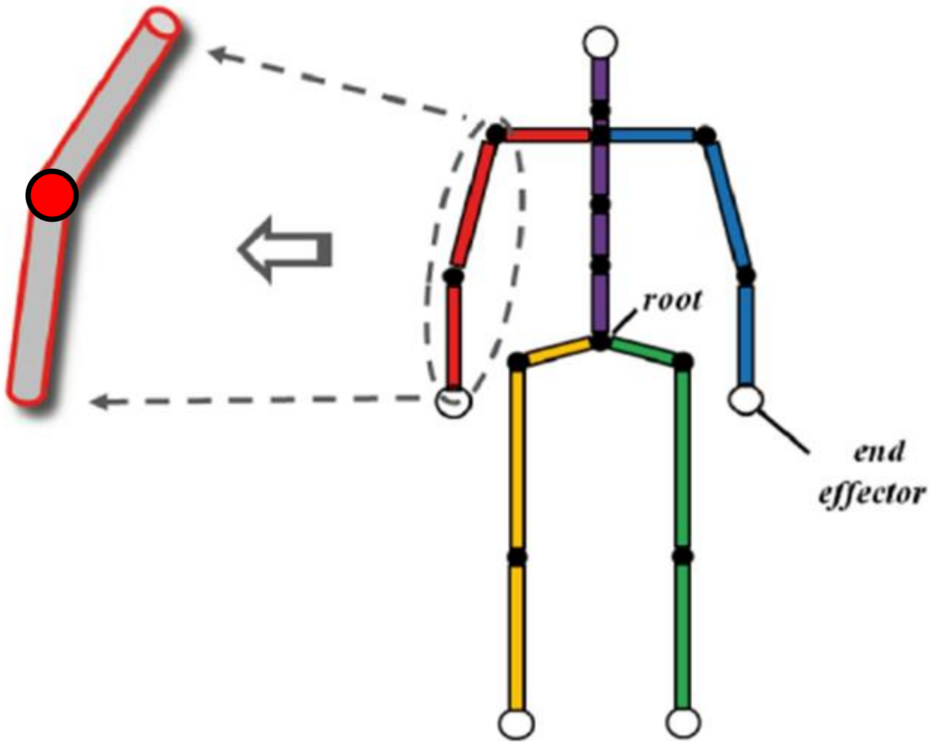
Modeling articulated rigid body systems

Rigid Bodies and Joints



Modeling articulated rigid body systems

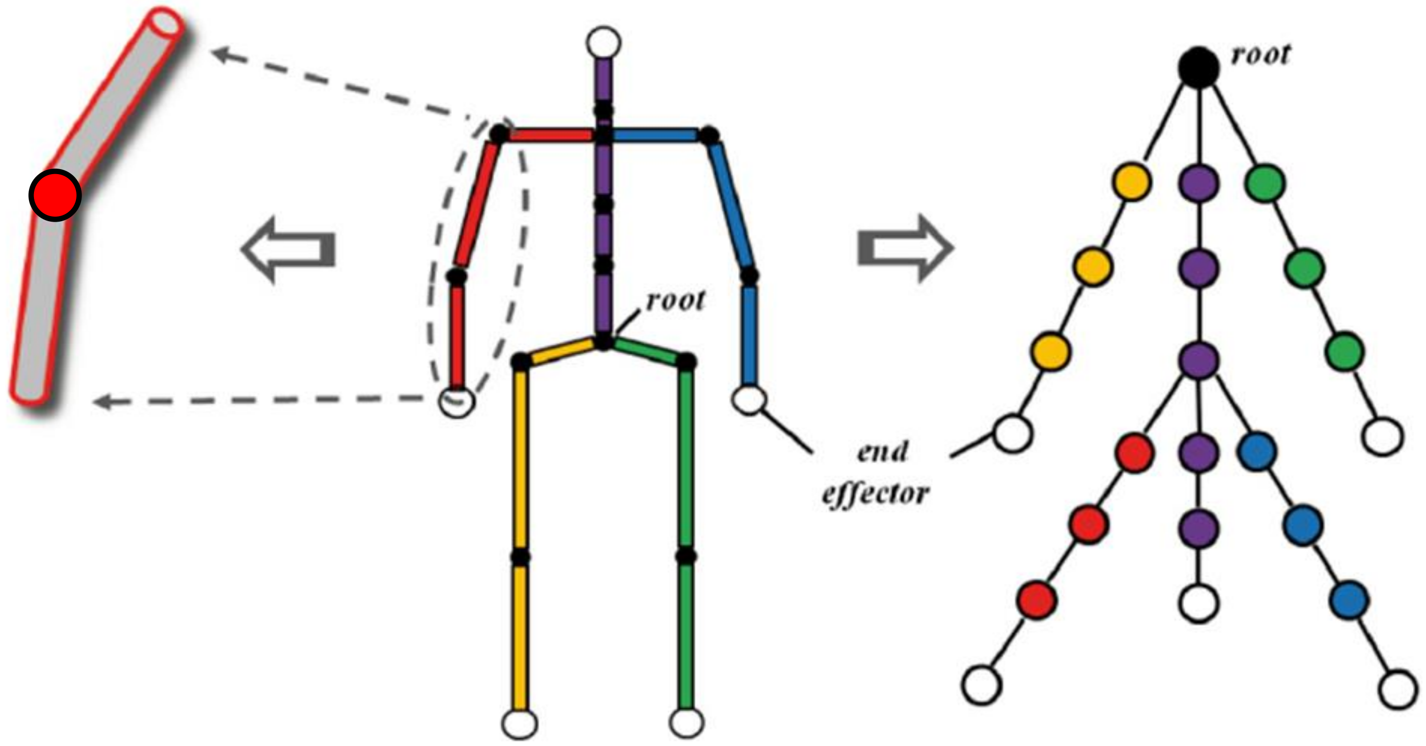
Rigid Bodies and Joints



Character/robot model

Modeling articulated rigid body systems

Rigid Bodies and Joints

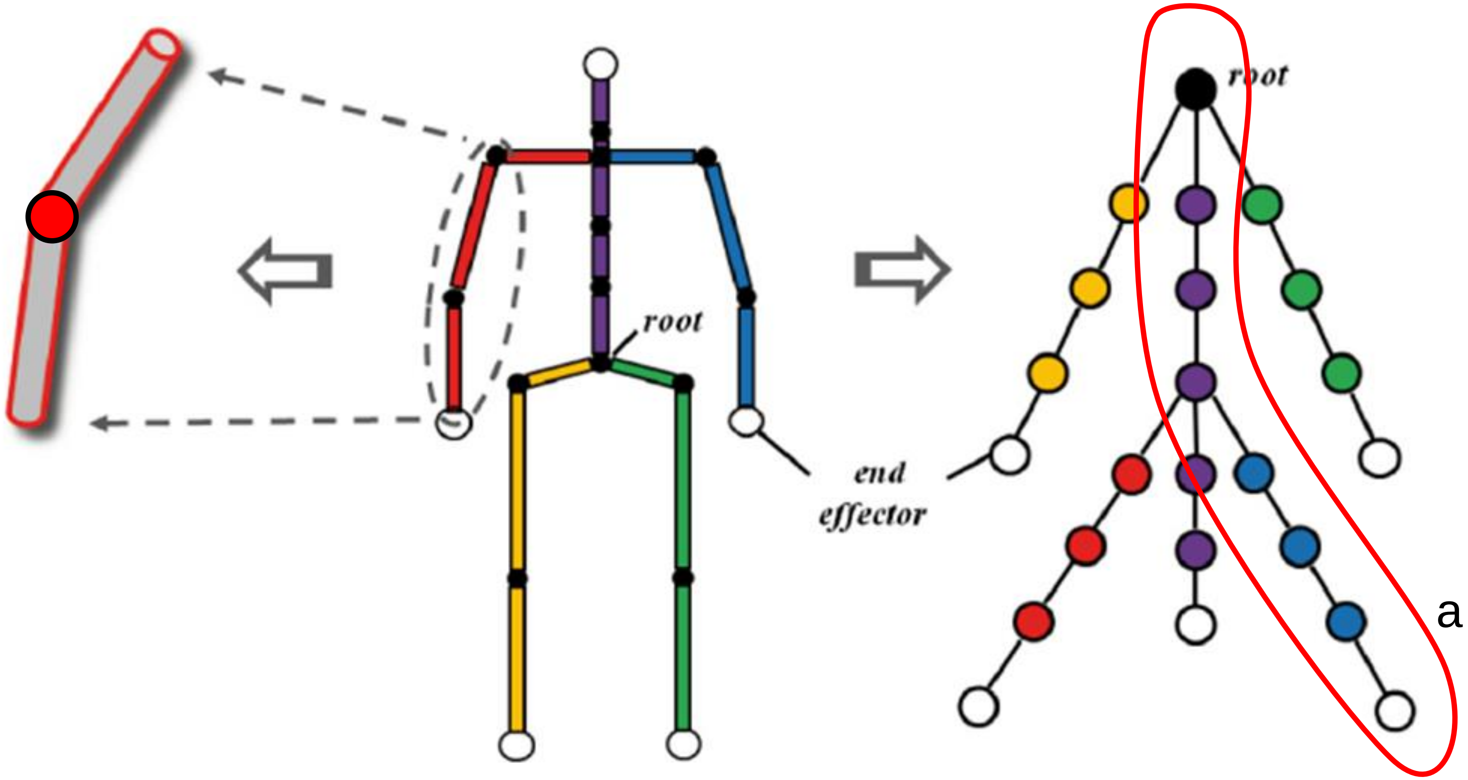


Character/robot model

Tree-based data structure
encodes hierarchical
connectivity of rigid bodies

Modeling articulated rigid body systems

Rigid Bodies and Joints

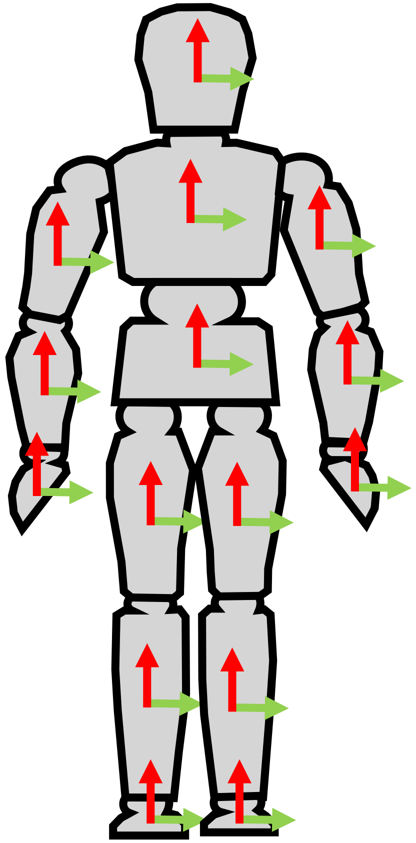


a kinematic chain

Character/robot model

Tree-based data structure encodes hierarchical connectivity of rigid bodies

How do we model a pose?



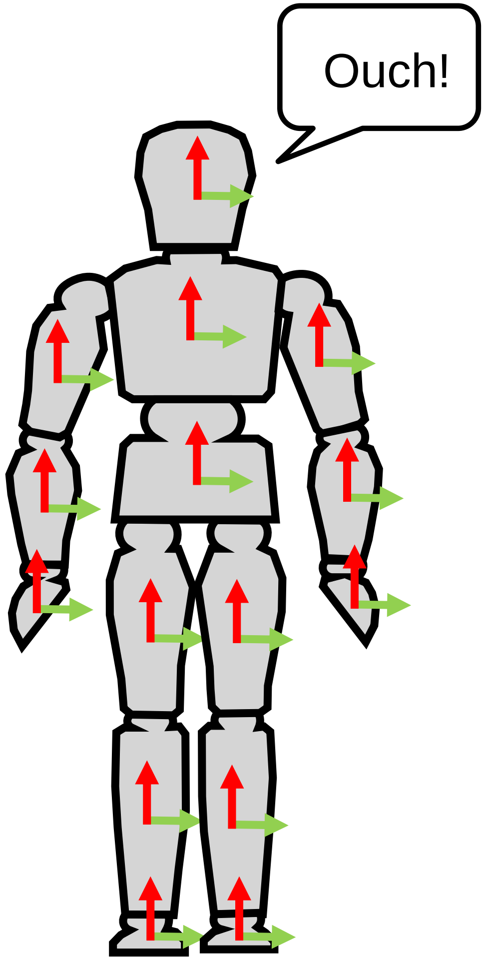
- Pose:
  - aggregate position and orientation of all *n* body parts into one large vector:

$$q = \left\{ \boldsymbol{x}_0, {}_W\boldsymbol{R}_0, \boldsymbol{x}_1, {}_W\boldsymbol{R}_1, \dots, \boldsymbol{x}_{n-1}, {}_W\boldsymbol{R}_{n-1} \right\}$$

    - Maximal coordinates (6**n* values, more on rotation parameterizations later)
    - Can represent any pose
    - Direct extension to rigid body kinematics/dynamics, easy to interface with physics engines for ragdoll effects, etc.
    - But…
      - Can represent MANY infeasible configurations also

Let’s try abducting the left leg (e.g. rotate in-plane by 45 degrees)

How do we model a pose?



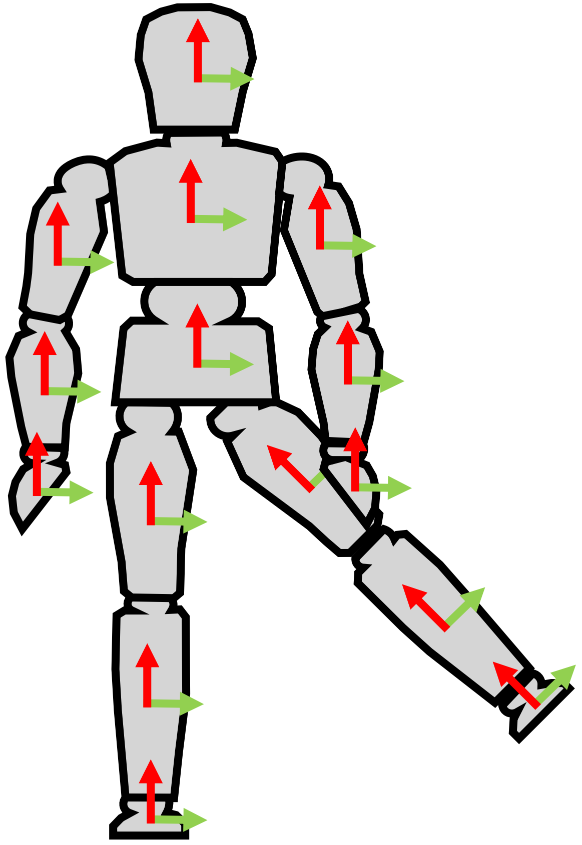
- Pose:
  - aggregate position and orientation of all *n* body parts into one large vector:

$$q = \left\{ \boldsymbol{x}_0, {}_W\boldsymbol{R}_0, \boldsymbol{x}_1, {}_W\boldsymbol{R}_1, \dots, \boldsymbol{x}_{n-1}, {}_W\boldsymbol{R}_{n-1} \right\}$$

  - Maximal coordinates (6*n values, more on rotation parameterizations later)
  - Can represent any pose
  - Direct extension to rigid body kinematics/dynamics, easy to interface with physics engines for ragdoll effects, etc.
  - But…
    - Can represent MANY infeasible configurations also
  - We probably wanted this!

Let’s try abducting the left leg (e.g. rotate in-plane by 45 degrees)

How do we model a pose?

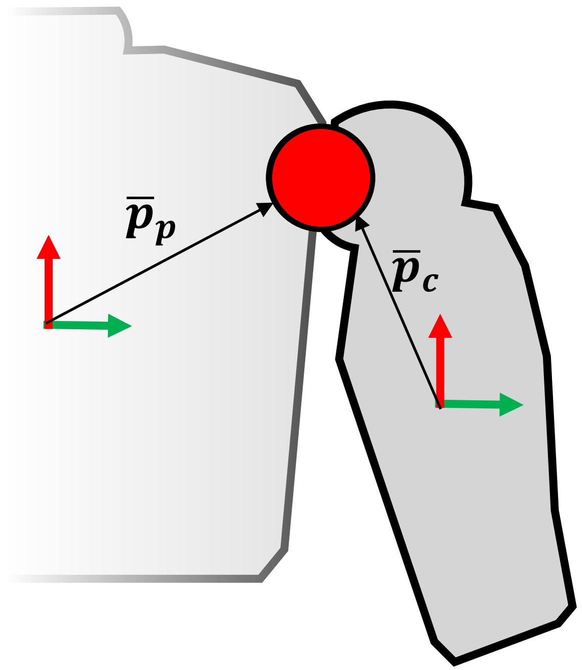


- Pose:
  - aggregate position and orientation of all *n* body parts into one large vector:

$$q = \left\{ \boldsymbol{x}_0, {}_W\boldsymbol{R}_0, \boldsymbol{x}_1, {}_W\boldsymbol{R}_1, \dots, \boldsymbol{x}_{n-1}, {}_W\boldsymbol{R}_{n-1} \right\}$$

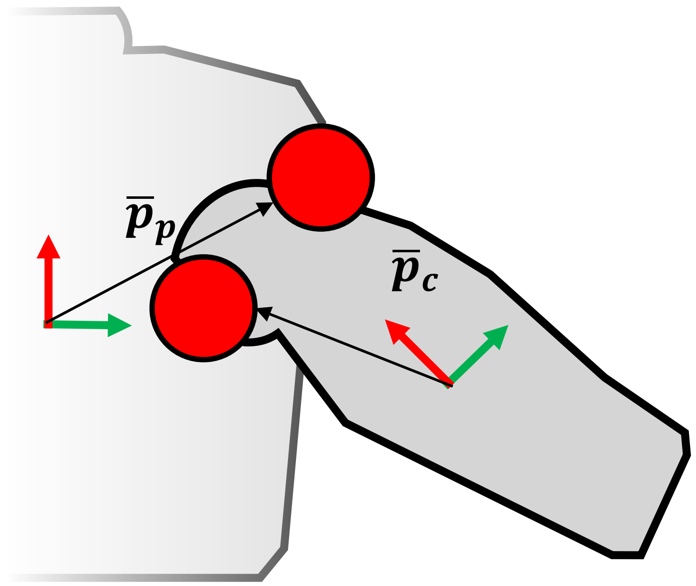
  - Maximal coordinates (6*n values, more on rotation parameterizations later)
  - Can represent any pose
  - Direct extension to rigid body kinematics/dynamics, easy to interface with physics engines for ragdoll effects, etc.
  - But…
    - Can represent MANY infeasible configurations also
  - We probably wanted this!
    - We can get there by recursively adjusting the position and orientation of all children

Fixing pose constraints – forward kinematics



- Assume we know:
  - Position $\boldsymbol{x}_p$ and orientation ${}_w\boldsymbol{R}_p$ of parent body
  - Relative orientation ${}_p\boldsymbol{R}_c$ due to joint motion (e.g. rotation about joint axis)
  - What is the position $\boldsymbol{x}_c$ and orientation ${}_w\boldsymbol{R}_c$ of child body?
- Answer:

Fixing pose constraints – forward kinematics



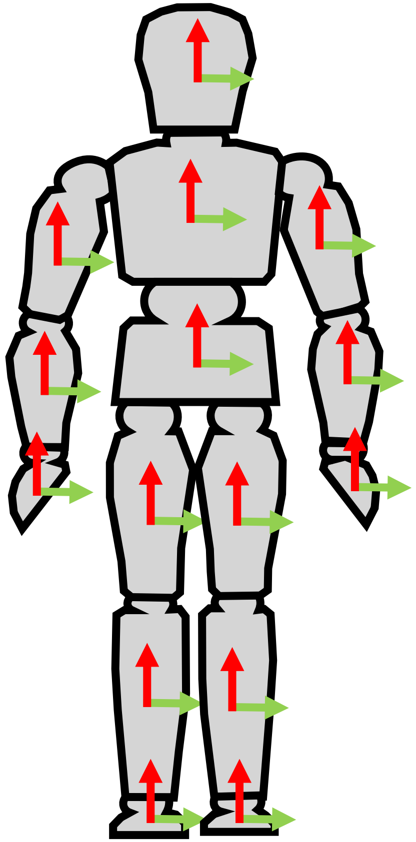
- Assume we know:
  - Position $\boldsymbol{x}_p$ and orientation ${}_w\boldsymbol{R}_p$ of parent body
  - Relative orientation ${}_p\boldsymbol{R}_c$ due to joint motion (e.g. rotation about joint axis)
  - What is the position $\boldsymbol{x}_c$ and orientation ${}_w\boldsymbol{R}_c$ of child body?
- Answer:

$$ {}_w\boldsymbol{R}_c = {}_w\boldsymbol{R}_p * {}_p\boldsymbol{R}_c $$

$$ \boldsymbol{x}_p + {}_w\boldsymbol{R}_p\bar{\boldsymbol{p}}_p = \boldsymbol{x}_c + {}_w\boldsymbol{R}_c\bar{\boldsymbol{p}}_c $$
$$ \Rightarrow \boldsymbol{x}_c = \boldsymbol{x}_p + {}_w\boldsymbol{R}_p\bar{\boldsymbol{p}}_p - {}_w\boldsymbol{R}_c\bar{\boldsymbol{p}}_c $$

- Repeat this procedure recursively for the child's children, its children's children, etc.
- So all we really need to know is the root's position and orientation, as well as each joint angle!

How do we model a pose? Take 2

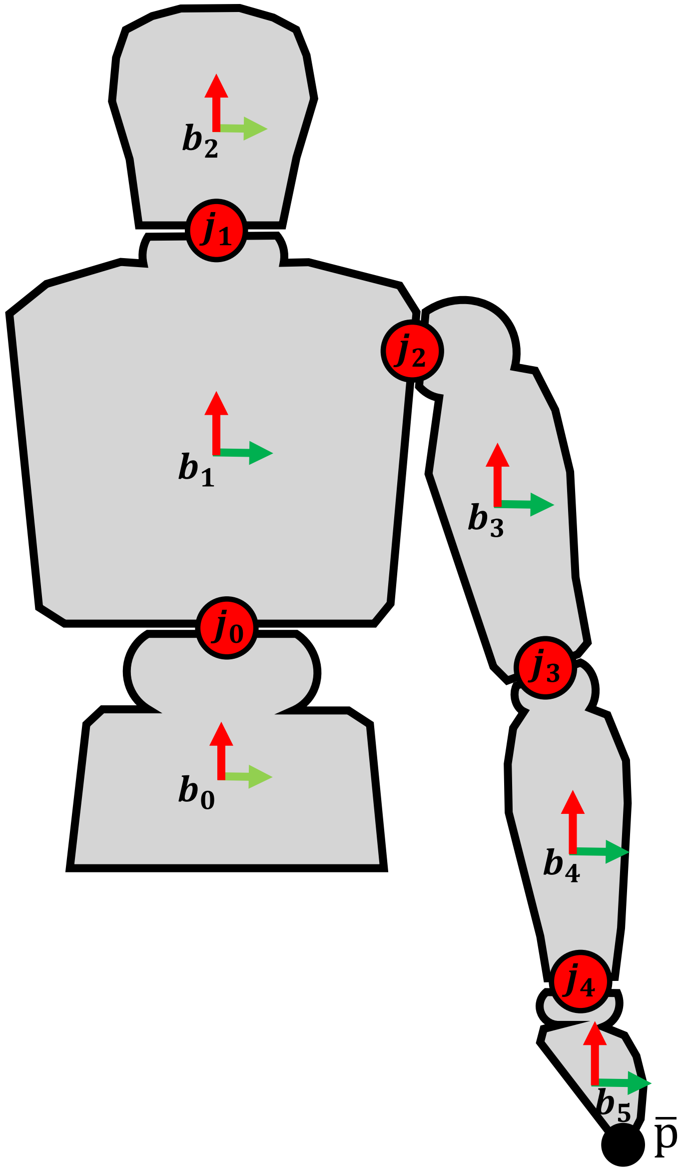


- Pose:
  - pose vector stores position and orientation of root + joint angles

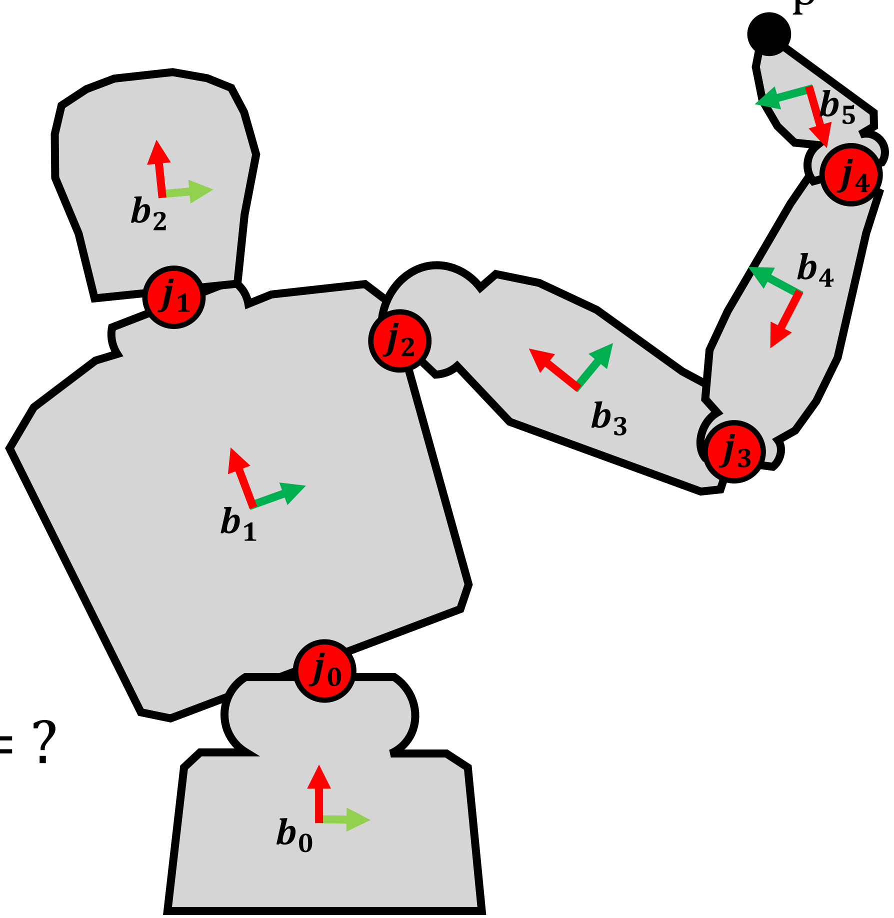
$$q = \{\overbrace{\boldsymbol{x}_{root}, \underbrace{\alpha, \beta, \gamma}_{{}_{w}\boldsymbol{R}_{root} = R_\alpha * R_\beta * R_\gamma}}^{\text{root DOFs}}, \overbrace{\theta_0, \theta_1, \dots, \theta_{n-2}}^{\text{joint angles}}\}$$

  - Reduced, or generalized coordinates: 6 + (*n* – 1) numbers
  - Can represent any feasible pose
  - Cannot represent infeasible poses (e.g. dislocated limbs)
  - Harder to derive equations of motion, but we'll do it anyway
  - How do we compute world coordinates of a point specified on a particular body part?
    - Compute that body part's position and orientation, then use formula from before
    - Or just perform computation directly

Forward kinematics using generalized coordinates



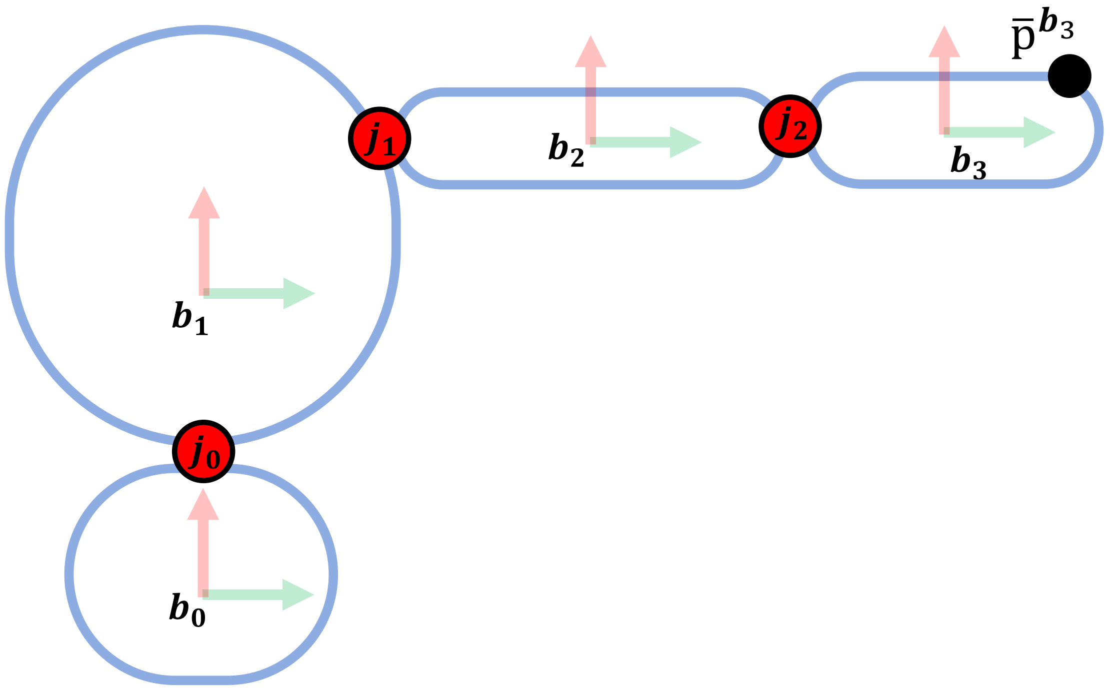
q



$$p(q, \bar{p}^{\boldsymbol{b}_5}) = FK(q, \bar{p}, \boldsymbol{b}_5) = ?$$

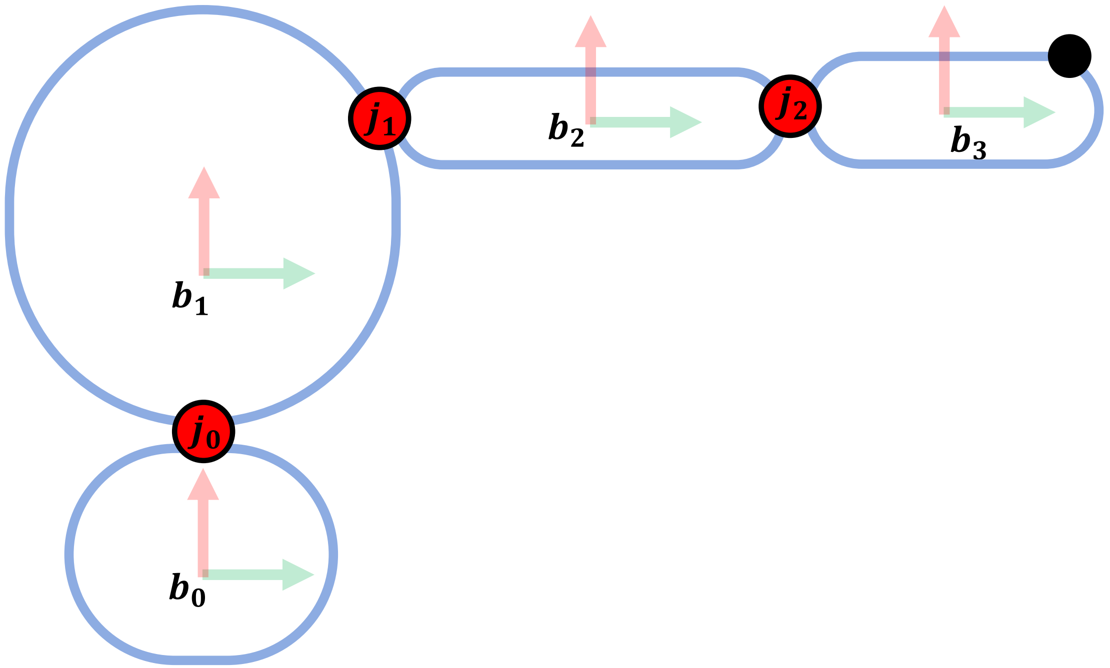
Forward kinematics using generalized coordinates

- Think of it as a sequence of rotations:
  - Start at the body the point is defined on, and move up the kinematic chain towards the root

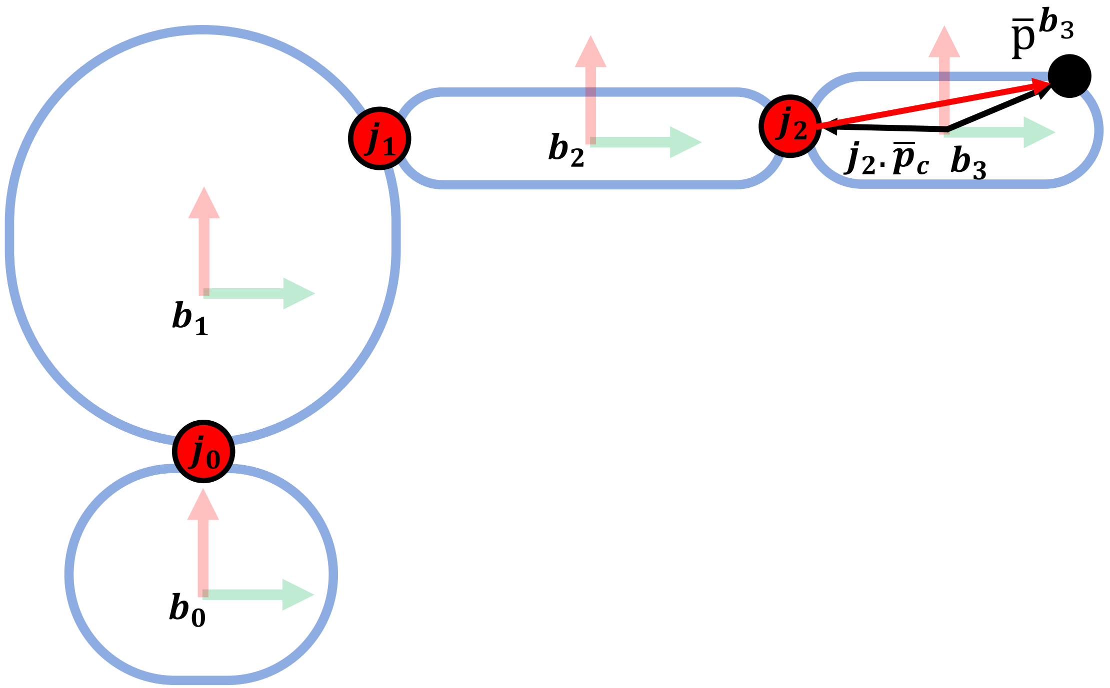


Forward kinematics using generalized coordinates

- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$

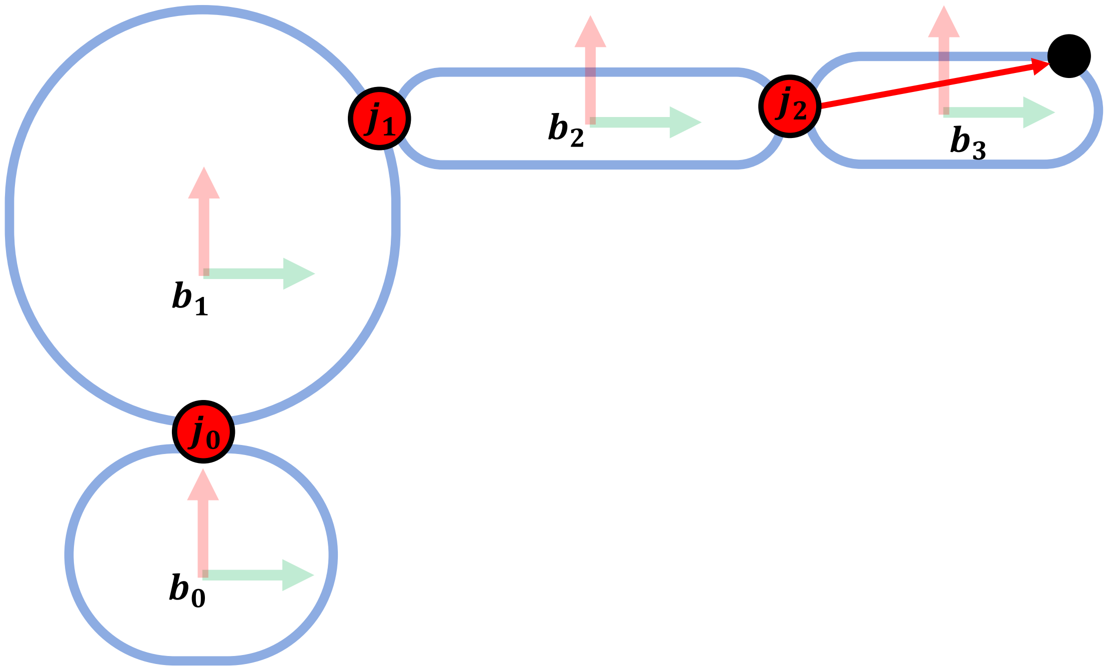


Forward kinematics using generalized coordinates



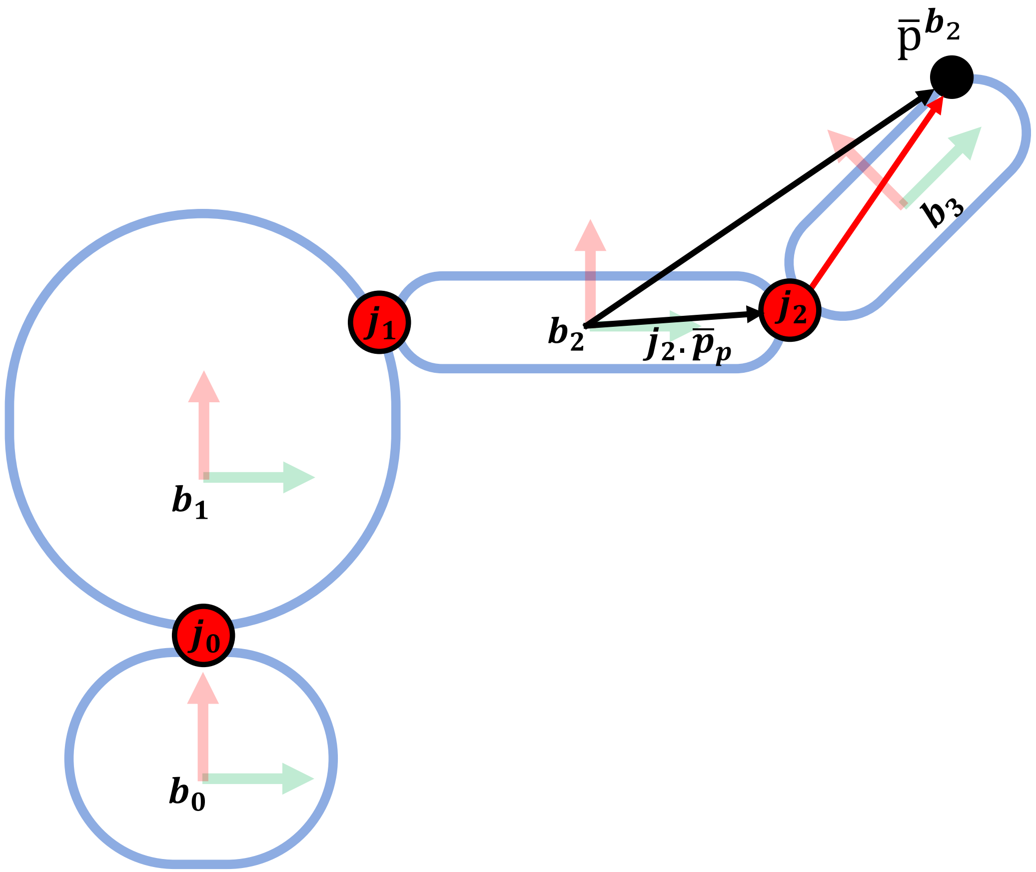
- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

Forward kinematics using generalized coordinates



- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\overline{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

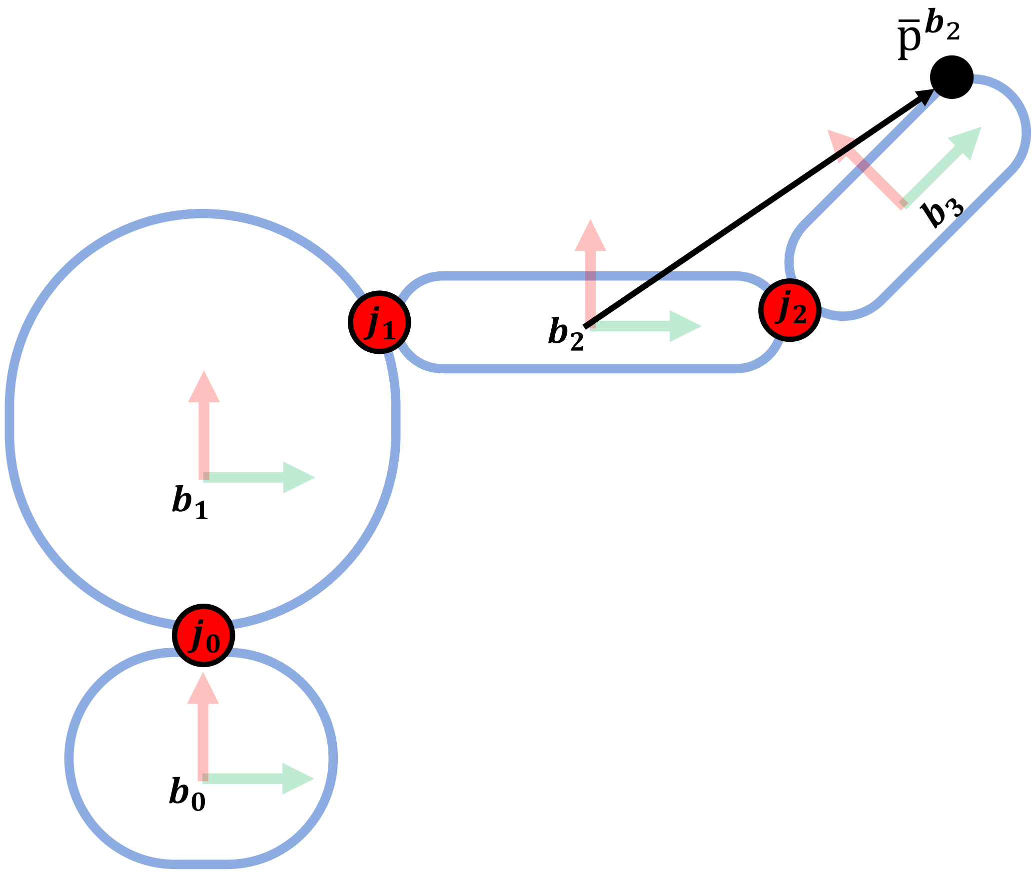
Forward kinematics using generalized coordinates



- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

$\boxed{}$

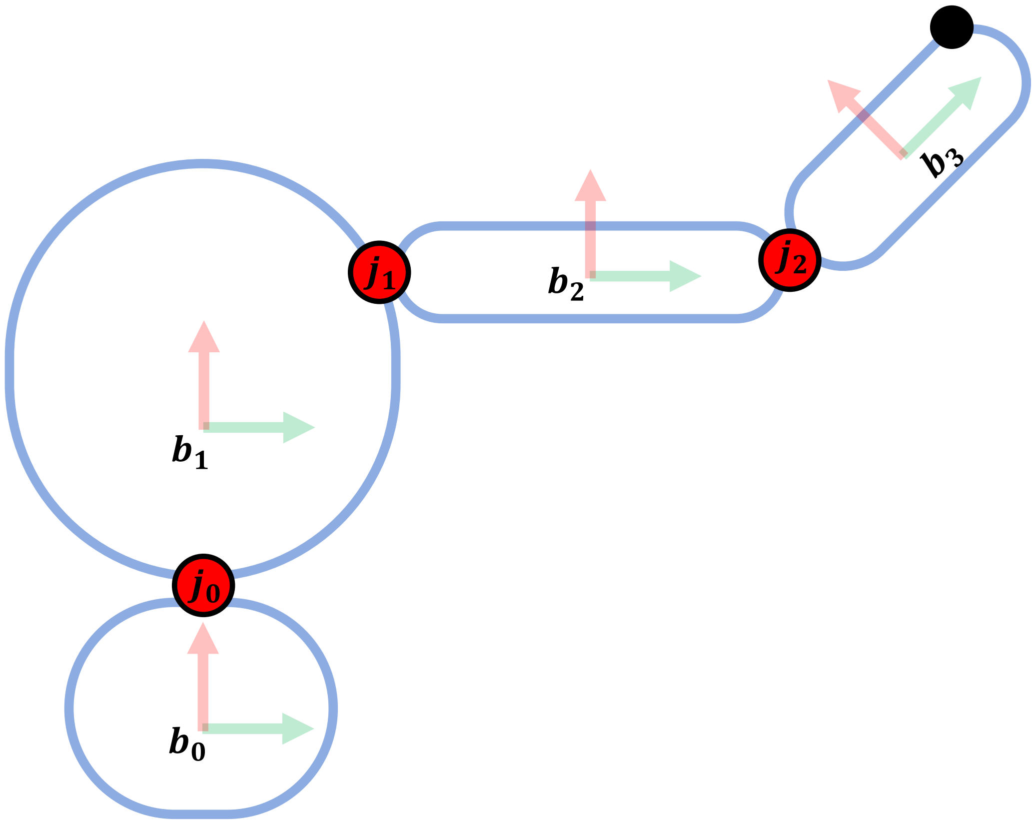
Forward kinematics using generalized coordinates



- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b}_3$ rotates about joint $\boldsymbol{j}_2$ relative to $\boldsymbol{b}_2$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b}_2$ after applying (relative) joint rotation $\boldsymbol{j}_2$?

$$\bar{\mathrm{p}}^{\boldsymbol{b}_2} = \boldsymbol{j}_2.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j}_2.\overline{\boldsymbol{v}}, \boldsymbol{\theta}_2) * \overrightarrow{(\boldsymbol{j}_2.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b}_3})}$$

Forward kinematics using generalized coordinates

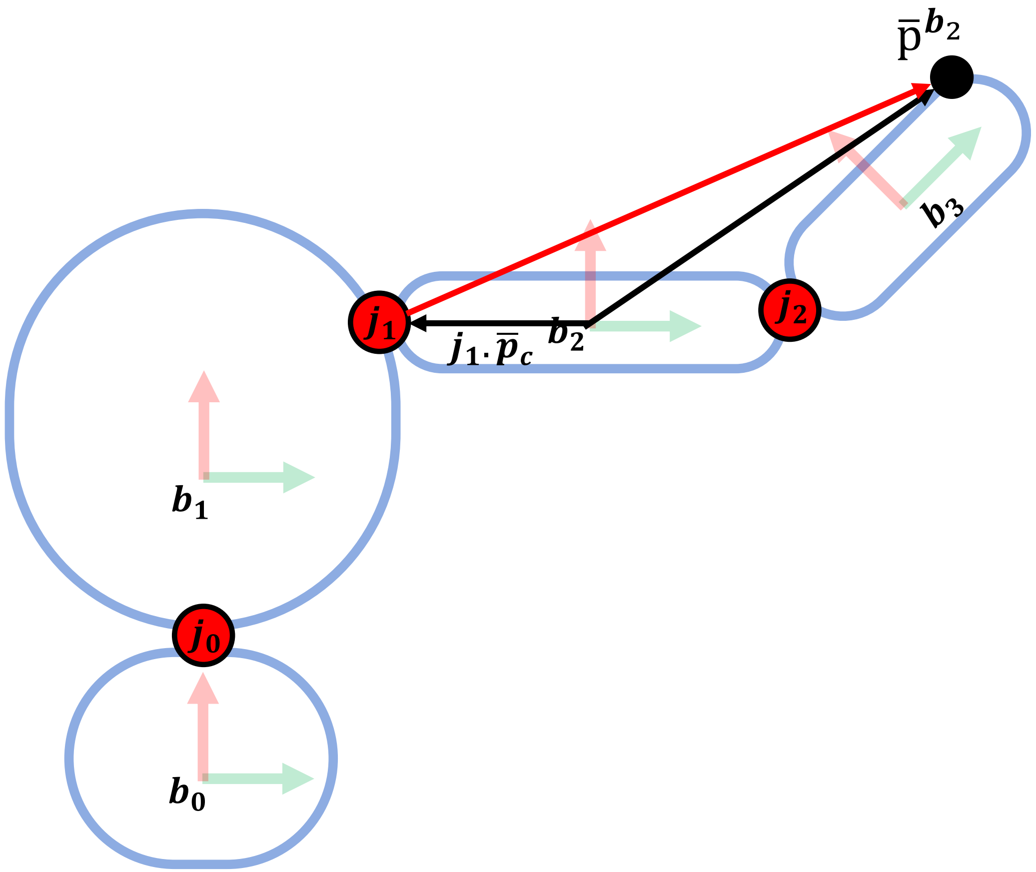


- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

$$\bar{\mathrm{p}}^{\boldsymbol{b_2}} = \boldsymbol{j_2}.\bar{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_2}.\bar{\boldsymbol{v}}, \boldsymbol{\theta_2}) * \overrightarrow{(\boldsymbol{j_2}.\bar{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b_3}})}$$

  - Now, think of a composite body $(\boldsymbol{b_2}, \boldsymbol{b_3})$ that rotates relative to $\boldsymbol{b_1}$ about joint $\boldsymbol{j_1}$

Forward kinematics using generalized coordinates

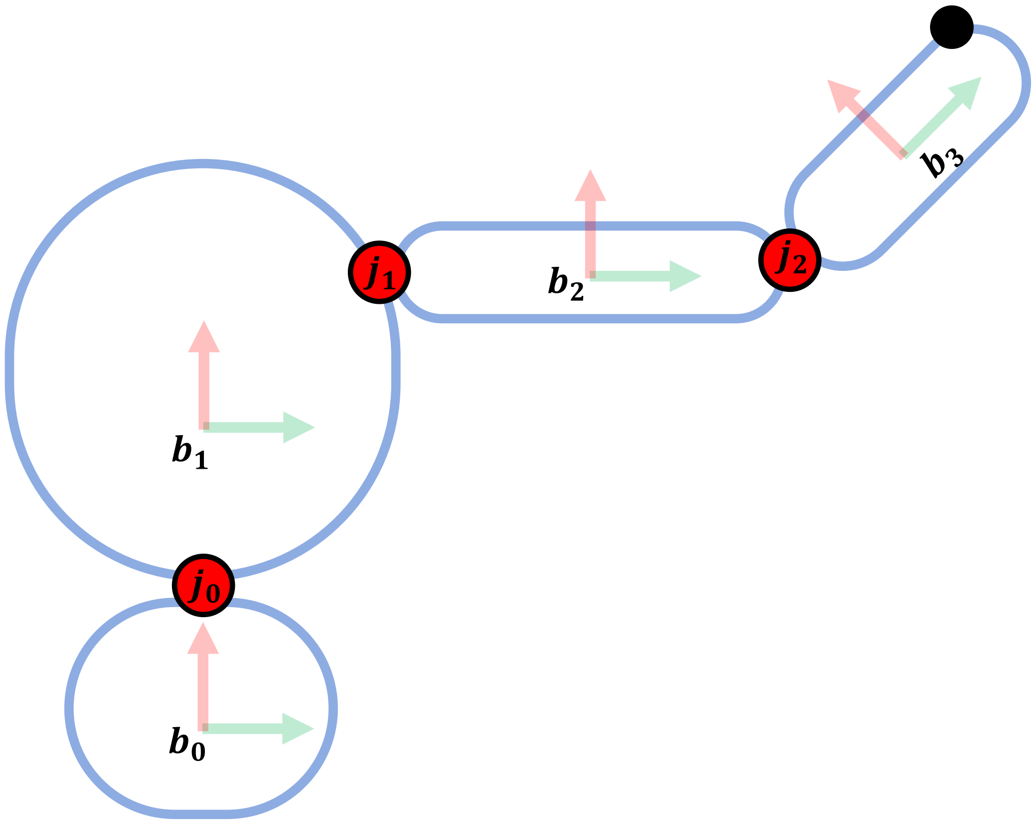


- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\overline{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

$$\overline{\mathrm{p}}^{b_2} = \boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_2}.\overline{\boldsymbol{v}}, \boldsymbol{\theta_2}) * \overrightarrow{(\boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \overline{\mathrm{p}}^{b_3})}$$

  - Now, think of a composite body $(\boldsymbol{b_2}, \boldsymbol{b_3})$ that rotates relative to $\boldsymbol{b_1}$ about joint $\boldsymbol{j_1}$
  - What are the coordinates of point $\overline{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_1}$ after applying joint rotation $\boldsymbol{j_1}$?

Forward kinematics using generalized coordinates



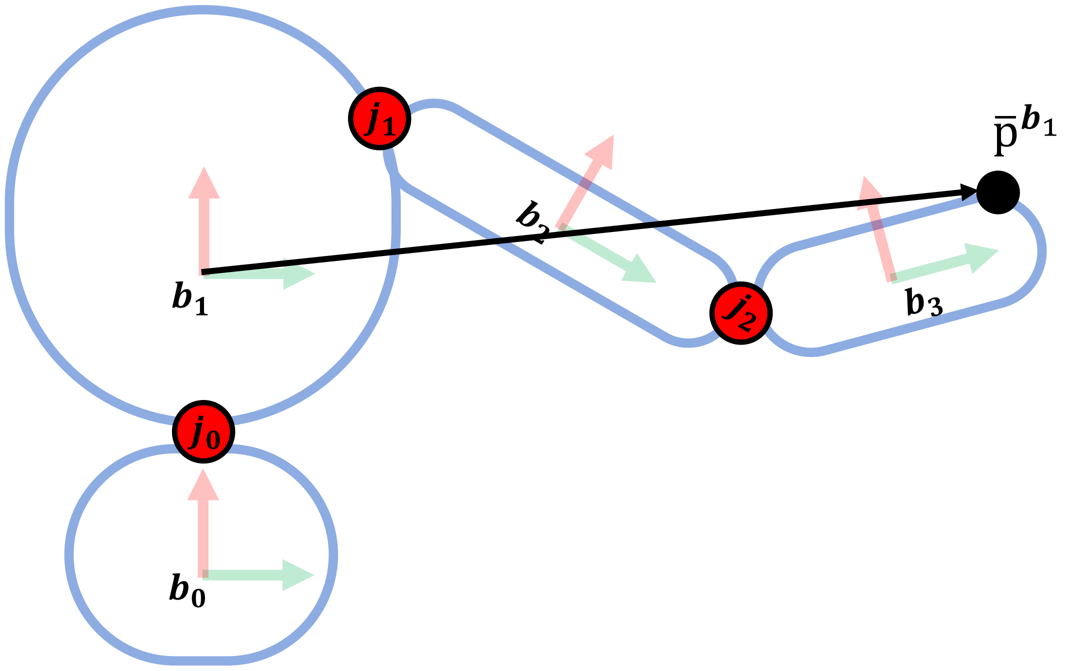
- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

$$\bar{\mathrm{p}}^{\boldsymbol{b}_2} = \boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_2}.\overline{\boldsymbol{v}}, \boldsymbol{\theta_2}) * \overrightarrow{(\boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b}_3})}$$

  - Now, think of a composite body $(\boldsymbol{b_2}, \boldsymbol{b_3})$ that rotates relative to $\boldsymbol{b_1}$ about joint $\boldsymbol{j_1}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_1}$ after applying joint rotation $\boldsymbol{j_1}$?

$$\bar{\mathrm{p}}^{\boldsymbol{b}_1} = \boldsymbol{j_1}.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_1}.\overline{\boldsymbol{v}}, \boldsymbol{\theta_1}) * \overrightarrow{(\boldsymbol{j_1}.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b}_2})}$$

Forward kinematics using generalized coordinates



- Think of it as a sequence of rotations:
  - First, $\boldsymbol{b_3}$ rotates about joint $\boldsymbol{j_2}$ relative to $\boldsymbol{b_2}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_2}$ after applying (relative) joint rotation $\boldsymbol{j_2}$?

$$\bar{\mathrm{p}}^{\boldsymbol{b}_2} = \boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_2}.\overline{\boldsymbol{v}}, \boldsymbol{\theta_2}) * \overrightarrow{(\boldsymbol{j_2}.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b}_3})}$$

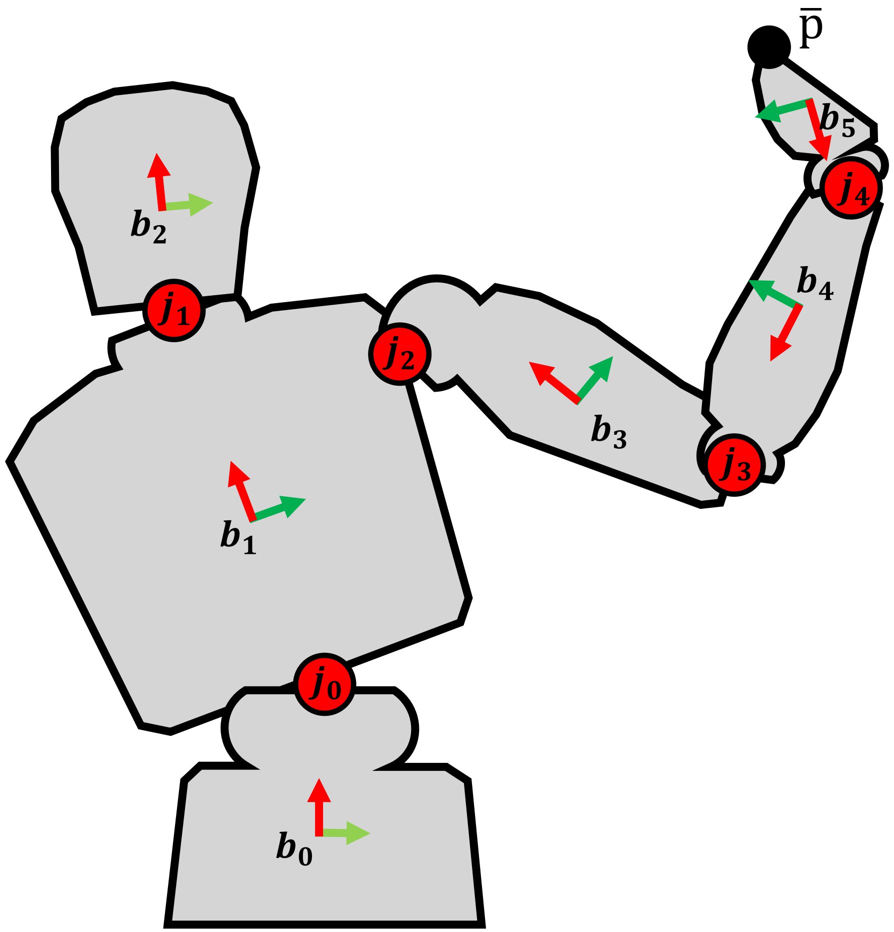
  - Now, think of a composite body $(\boldsymbol{b_2}, \boldsymbol{b_3})$ that rotates relative to $\boldsymbol{b_1}$ about joint $\boldsymbol{j_1}$
  - What are the coordinates of point $\bar{\mathbf{p}}$, in coordinate frame of body $\boldsymbol{b_1}$ after applying joint rotation $\boldsymbol{j_1}$?

$$\bar{\mathrm{p}}^{\boldsymbol{b}_1} = \boldsymbol{j_1}.\overline{\boldsymbol{p}}_{\boldsymbol{p}} + \boldsymbol{R}(\boldsymbol{j_1}.\overline{\boldsymbol{v}}, \boldsymbol{\theta_1}) * \overrightarrow{(\boldsymbol{j_1}.\overline{\boldsymbol{p}}_{\boldsymbol{c}}, \bar{\mathrm{p}}^{\boldsymbol{b}_2})}$$

  - Repeat recursively until reaching the root, then apply the final rotation + translation:

$$\mathrm{p}(\mathrm{q}) = \boldsymbol{x}_{root} + {}_{w}\boldsymbol{R}_{root}\bar{\mathrm{p}}^{\boldsymbol{b}_0}$$

Forward kinematics using generalized coordinates

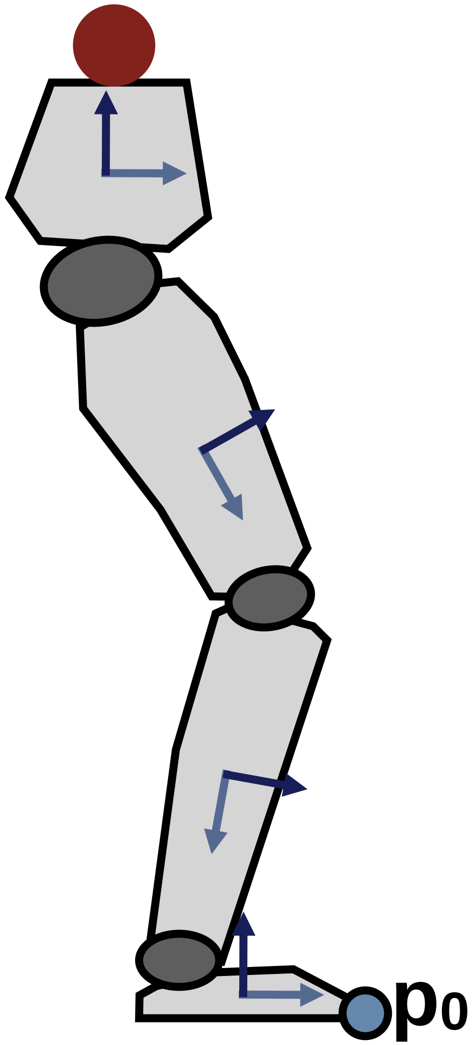


$$p(q, \bar{p}^{\boldsymbol{b}_5}) = FK(q, \bar{p}, \boldsymbol{b}_5) = ?$$

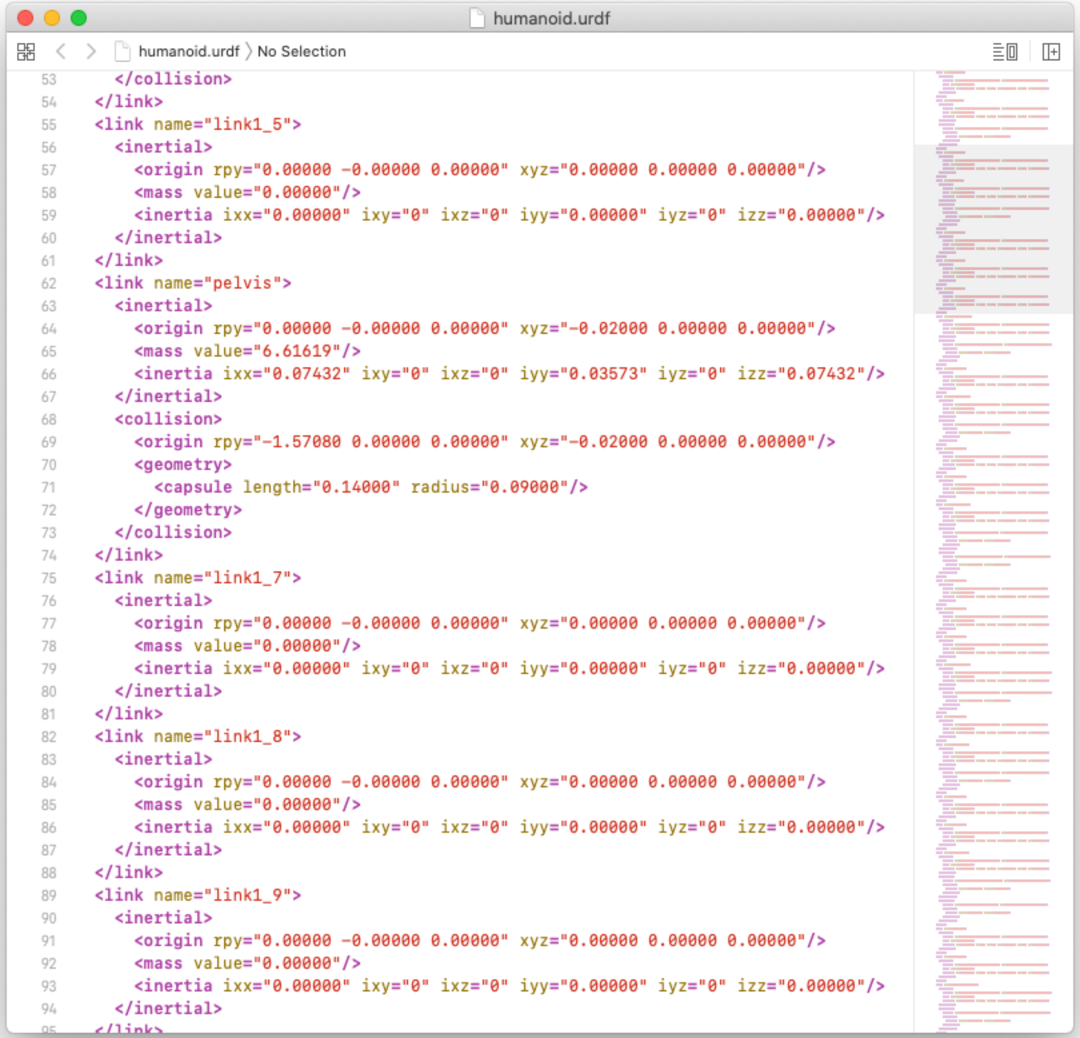
```
//NOTE 1: point p specified in the coord frame of body p
//NOTE 2: root_x(q), root_R(q) and j.R(q) extract data from
//appropriate indices in q
P3D FK(Pose q, P3D p, BodyPart b){
        Joint j = p.parentJoint;
        //check if b is root – if so, return world coordinates of p
        if (j == NULL)
                return root_x(q) + root_R(q) * vec(P3D(0,0,0), p);
        //return point expressed in coordinate frame of the parent
        return FK(j.p_p + j.R(q) * vec(j.p_c, p), j.parentBody);
}
```

Local and global coordinate frames

- We now know how to refer to a point in its own local frame and in the world frame.
- Within the kinematic transformation chain, we can refer to *any* point in *any* parent coordinate frame.
- Do you know how to compute the coordinates of the toe in the coordinate frame of the thigh?
- Do you know how to compute world coordinates of vectors which are specified in a local coordinate frame?
- Given a world-coordinates point p, do you know how to compute its coordinates in the local frame of reference of a body?



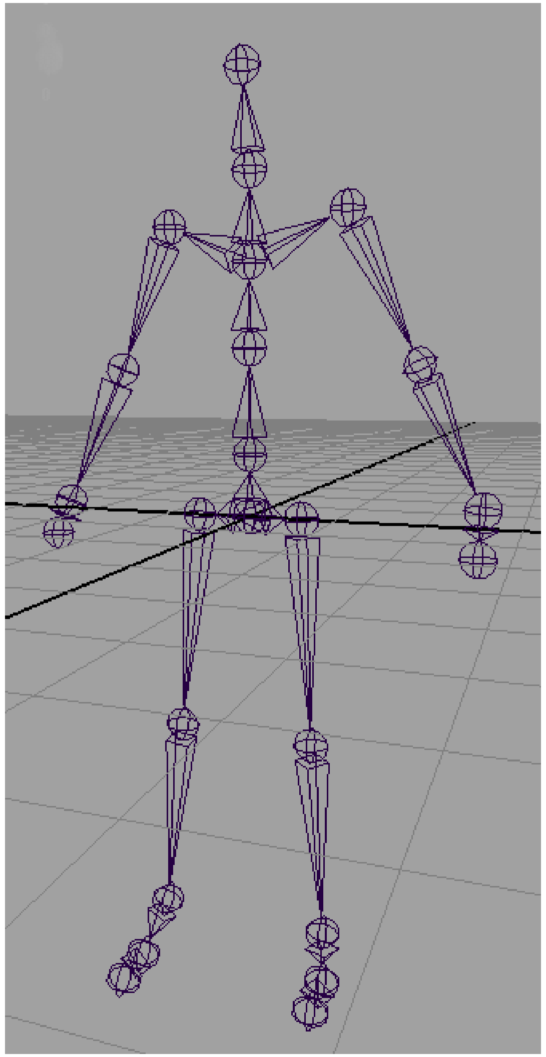
Hierarchical model description

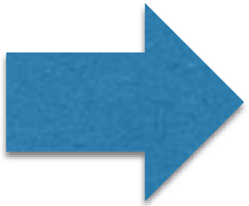


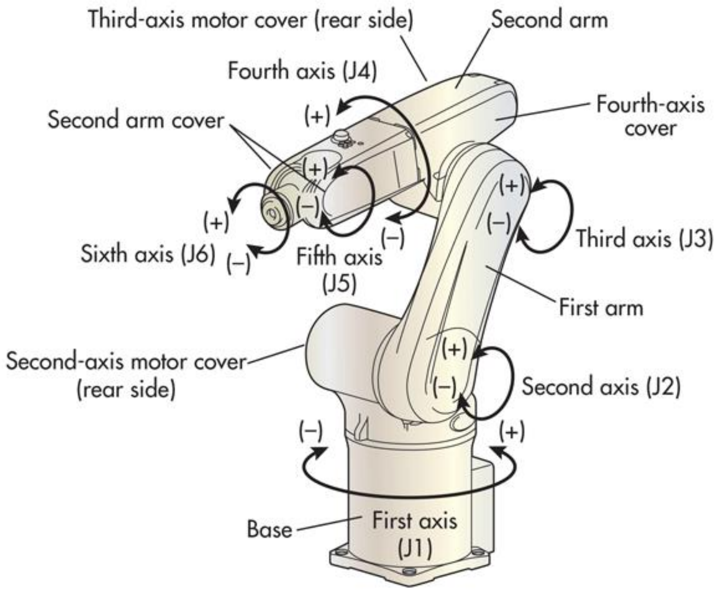

```xml
    </collision>
  </link>
  <link name="link1_5">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="pelvis">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <mass value="6.61619"/>
      <inertia ixx="0.07432" ixy="0" ixz="0" iyy="0.03573" iyz="0" izz="0.07432"/>
    </inertial>
    <collision>
      <origin rpy="-1.57080 0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <geometry>
        <capsule length="0.14000" radius="0.09000"/>
      </geometry>
    </collision>
  </link>
  <link name="link1_7">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_8">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_9">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
```

- Hierarchical modeling can be described compactly and precisely in agreed-upon formats.
- Common file formats used in CG applications include FBX and DAE.
- Most common file formats used in robotics are URDF and SDF.

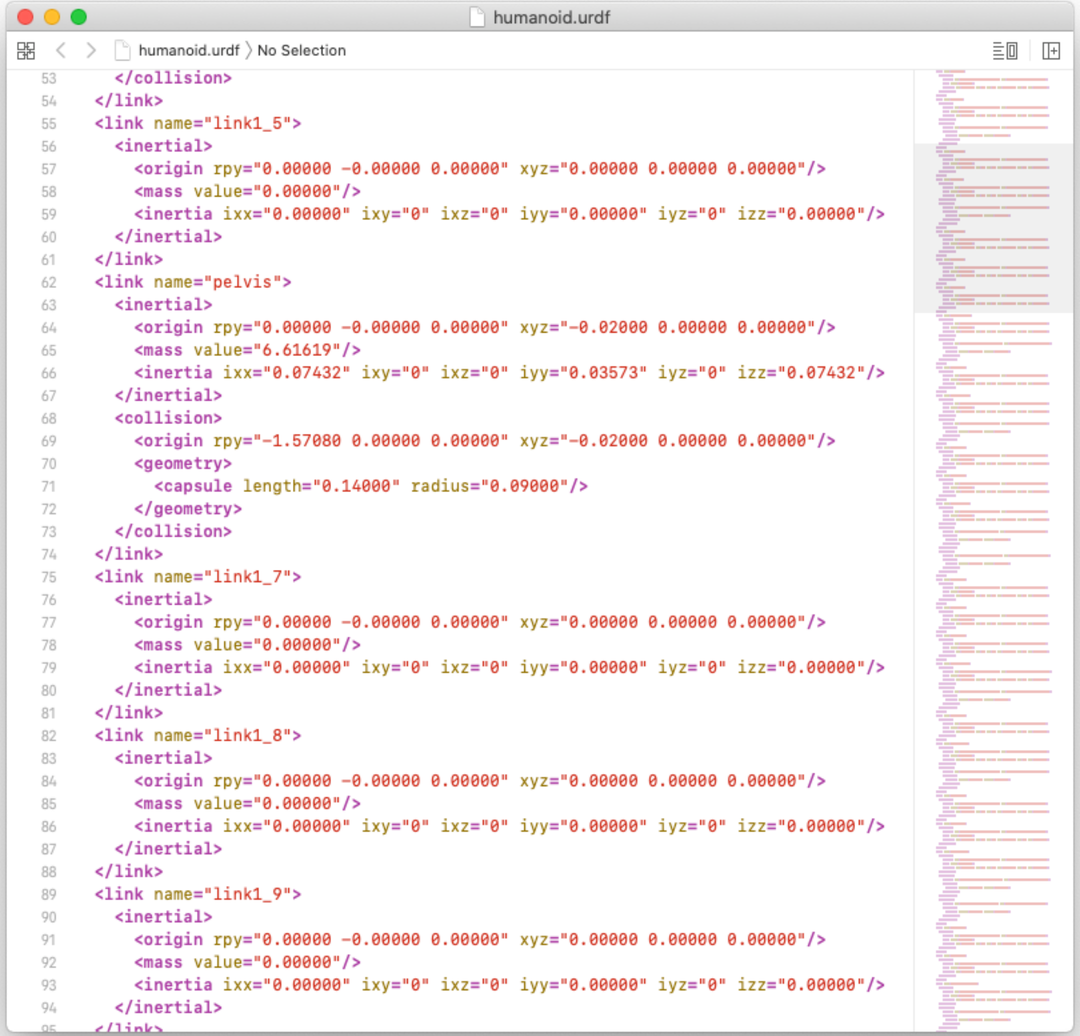
Articulated characters vs articulated robots







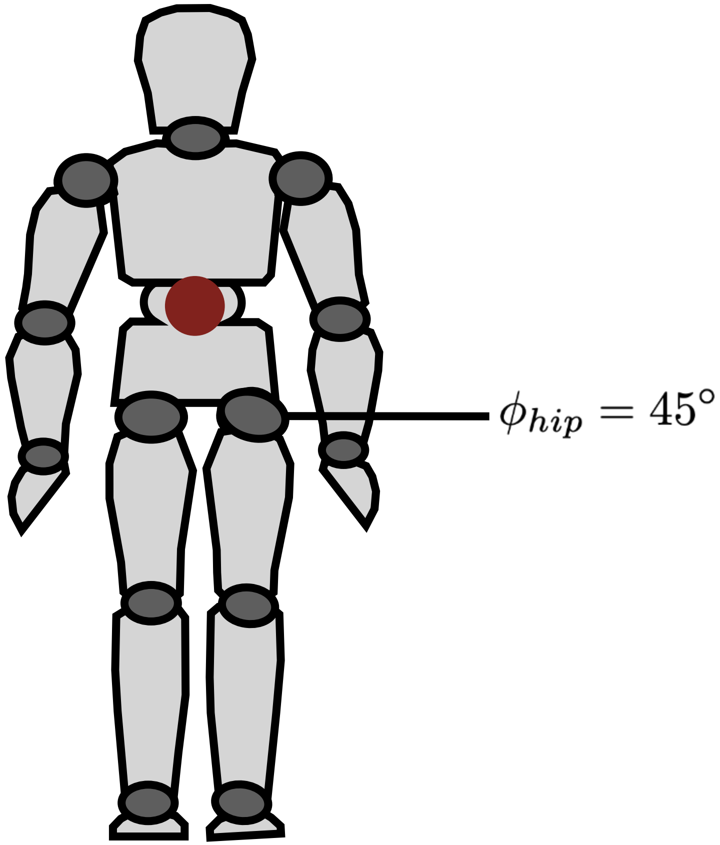
Hierarchical model description




```xml
    </collision>
  </link>
  <link name="link1_5">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="pelvis">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <mass value="6.61619"/>
      <inertia ixx="0.07432" ixy="0" ixz="0" iyy="0.03573" iyz="0" izz="0.07432"/>
    </inertial>
    <collision>
      <origin rpy="-1.57080 0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <geometry>
        <capsule length="0.14000" radius="0.09000"/>
      </geometry>
    </collision>
  </link>
  <link name="link1_7">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_8">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_9">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
```

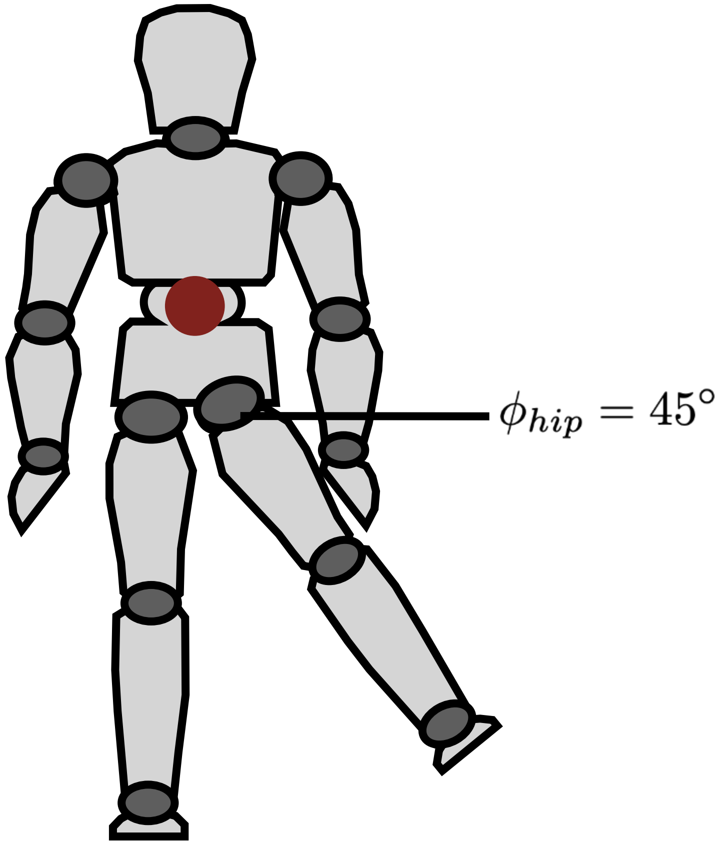
- Hierarchical modeling can be described compactly and precisely in agreed-upon formats.
- Common file formats used in CG applications include FBX and DAE.
- Most common file formats used in robotics are URDF and SDF.
- Many, many modeling choices

Root location

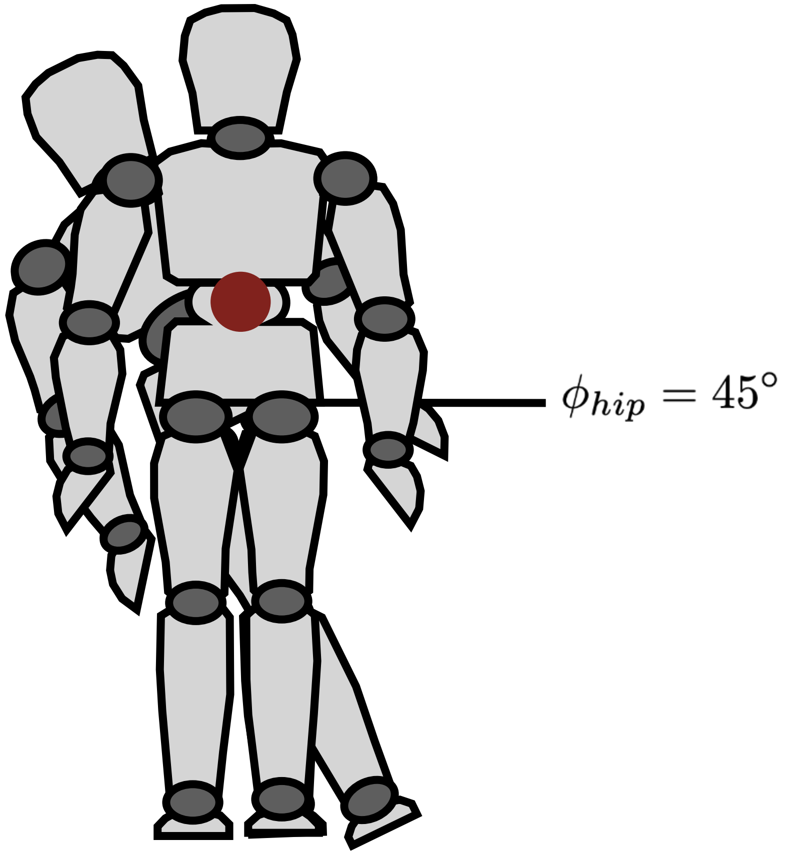


root: pelvis

Root location



root: abdomen



root: left foot

Hierarchical model description

humanoid.urdf

humanoid.urdf 〉 No Selection

```xml
    </collision>
  </link>
  <link name="link1_5">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="pelvis">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <mass value="6.61619"/>
      <inertia ixx="0.07432" ixy="0" ixz="0" iyy="0.03573" iyz="0" izz="0.07432"/>
    </inertial>
    <collision>
      <origin rpy="-1.57080 0.00000 0.00000" xyz="-0.02000 0.00000 0.00000"/>
      <geometry>
        <capsule length="0.14000" radius="0.09000"/>
      </geometry>
    </collision>
  </link>
  <link name="link1_7">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_8">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
  <link name="link1_9">
    <inertial>
      <origin rpy="0.00000 -0.00000 0.00000" xyz="0.00000 0.00000 0.00000"/>
      <mass value="0.00000"/>
      <inertia ixx="0.00000" ixy="0" ixz="0" iyy="0.00000" iyz="0" izz="0.00000"/>
    </inertial>
  </link>
```

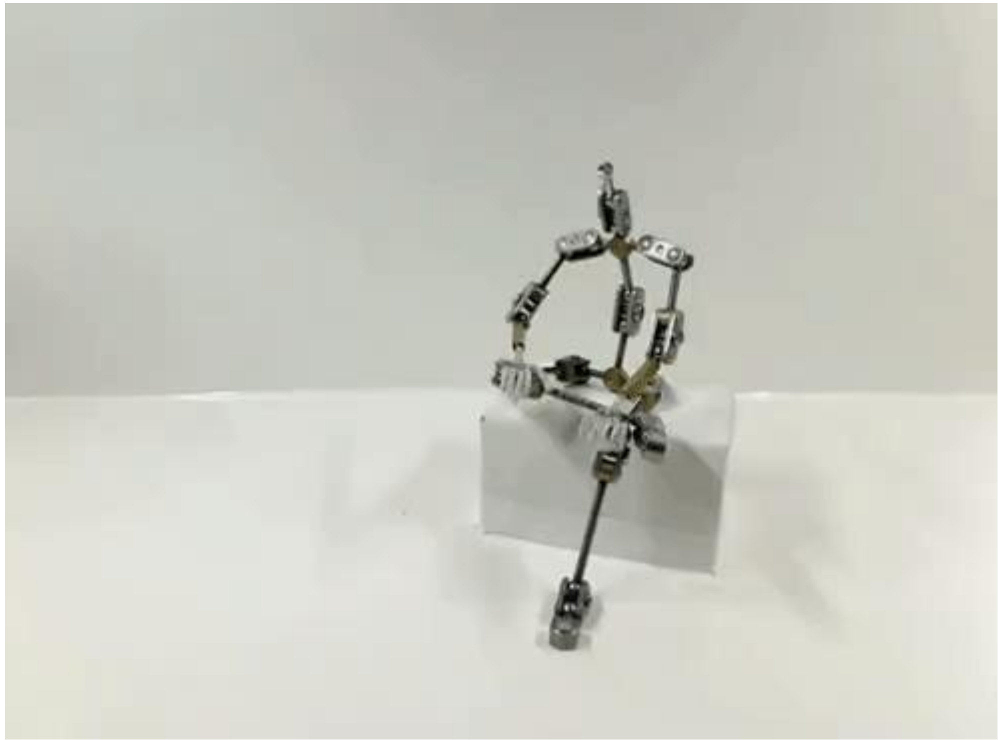
- Hierarchical modeling can be described compactly and precisely in agreed-upon formats.
- Common file formats used in CG applications include FBX and DAE.
- Most common file formats used in robotics are URDF and SDF.
- Many, many modeling choices
  - All of which should ultimately be equivalent!
  - Pick the one that's most convenient for your task.

This is Forward Kinematics in a nutshell

- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat

Keyframing

- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat

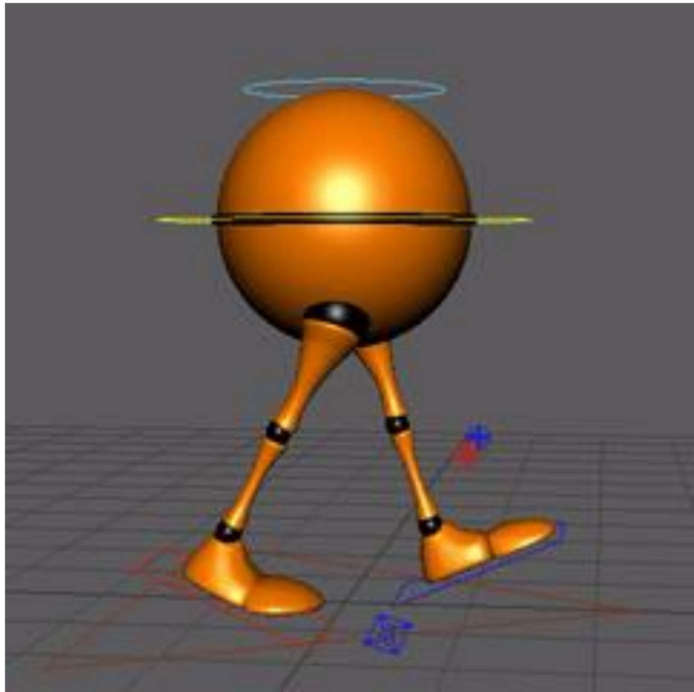


Keyframing

- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat

Keyframing

- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat



Keyframing

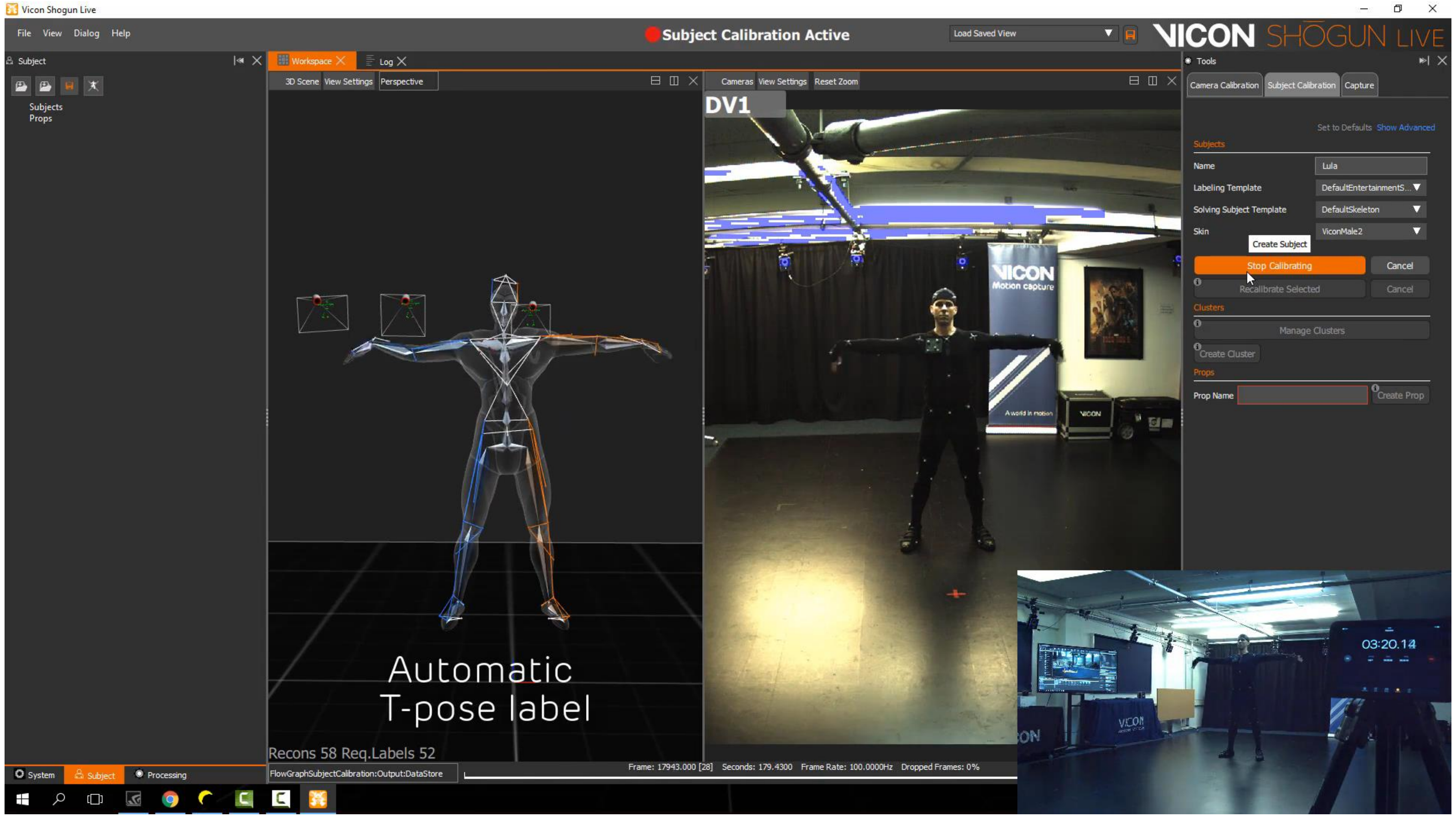
- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat

Keyframing

- But how do we use FK for animation?
  - Start with articulated structure
  - Specify a pose (e.g. root position/orientation, joint angles)
  - Draw/take a picture/etc.
  - Repeat
- Powerful but very tedious
  - Alternatives, of course, do exist…

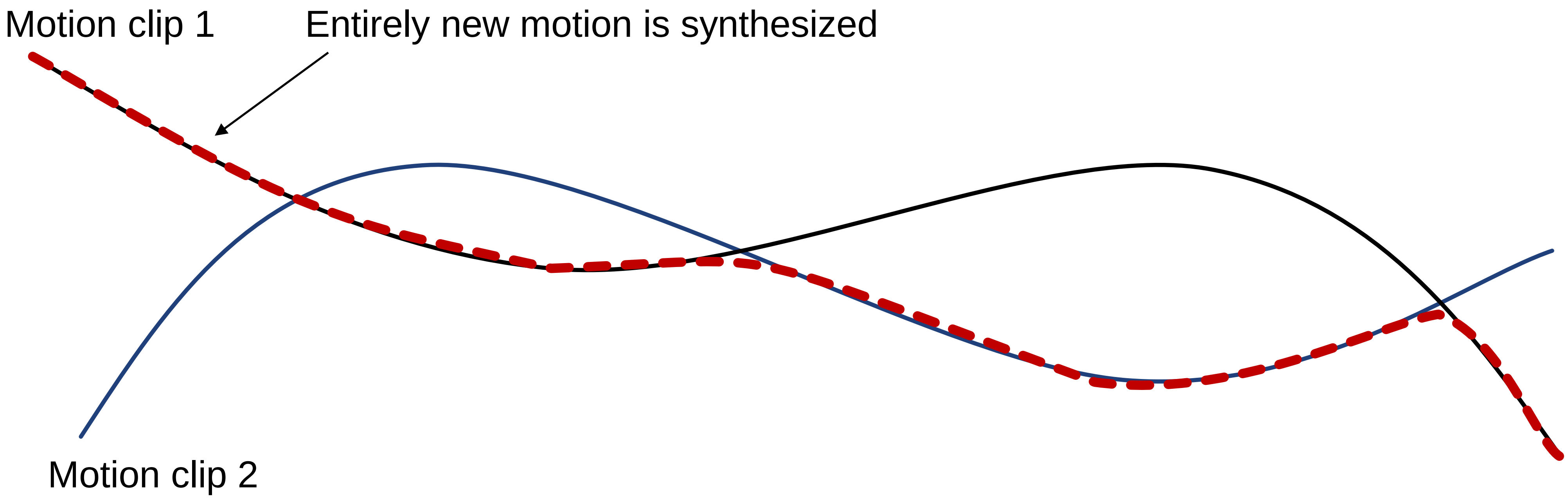
Motion capture

- Record poses from a live actor, apply those poses to an articulated character



Leveraging motion capture data

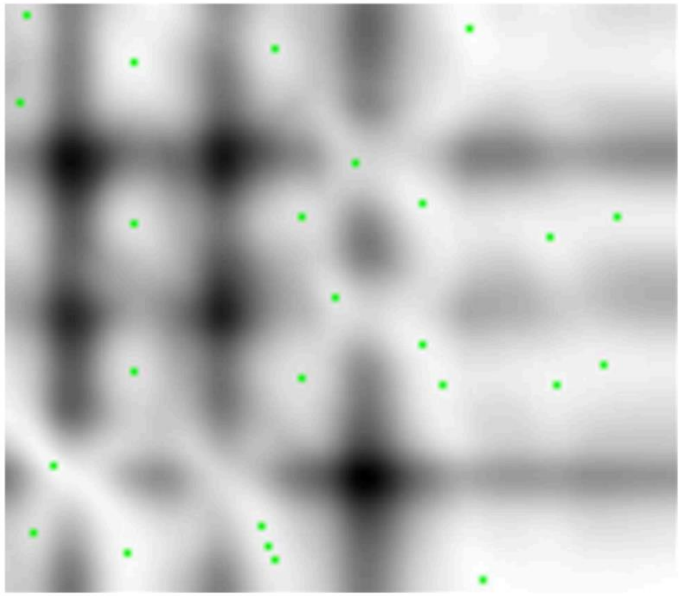
- Motion capture clips (sequence of poses over time) can either be played back directly, or they can be chopped up, re-sequenced and blended together to create new motions altogether



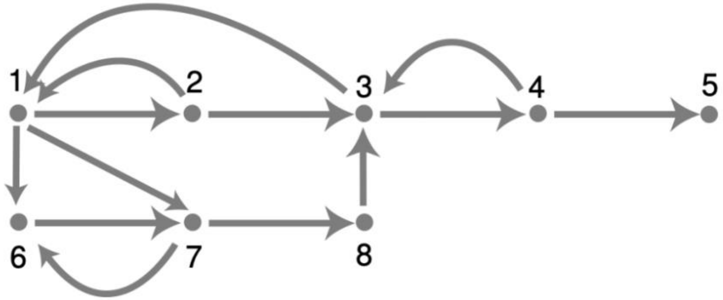
Leveraging motion capture data

- Motion capture clips (sequence of poses over time) can either be played back directly, or they can be chopped up, re-sequenced and blended together to create new motions altogether
- Generally speaking, we need:
  - A way of computing similarity between poses. The closer poses are, the more seamless the transition between two different motion clips.
  - A controller that decides which path to take whenever multiple possibilities exist.
  - Routines to smooth out/fix resulting motions.
- Subject of significant research efforts over the past two decades, and many successful implementations in video games and other interactive applications

Motion Graphs



Similarity matrix for two motion clips

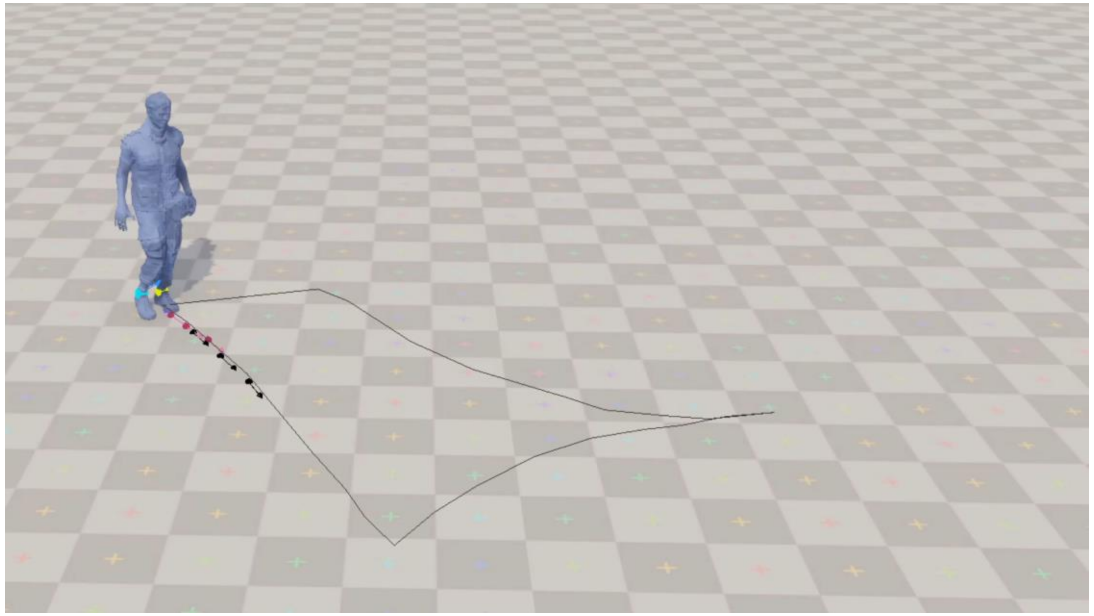


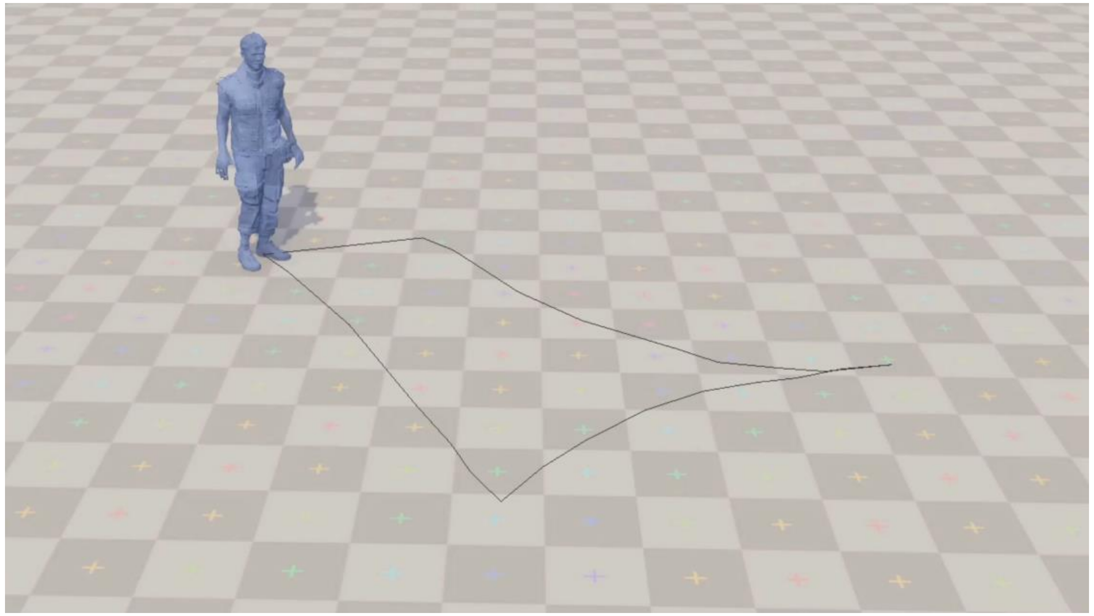
Motion Graph: edges represent motion clips; nodes represent transitions enabled by poses that are sufficiently similar

[1] Lucas Kovar, Michael Gleicher, Frédéric H. Pighin. **Motion Graphs.** SIGGRAPH 2002.

[2] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins,Nancy S. Pollard. **Interactive control of avatars animated with human motion data.** SIGGRAPH 2002.

Motion Fields/Motion Matching





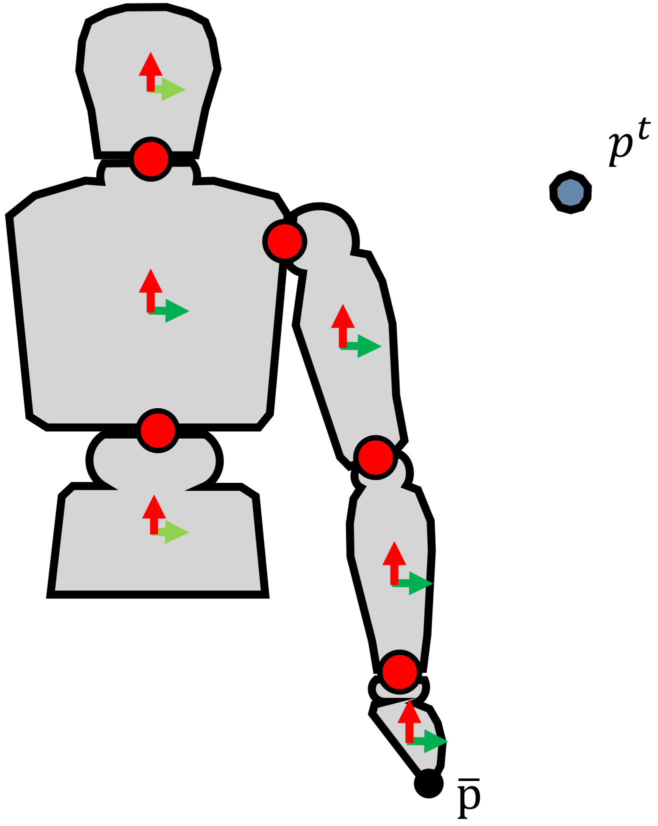
[1] Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popovic, Zoran Popovic. **Motion Fields for Interactive Character Animation.** SIGGRAPH Asia 2009.

[2] https://montreal.ubisoft.com/en/introducing-learned-motion-matching/

Forward Kinematics

- Combined with other tools and data sources, FK can take us far.
  - But of course, we are not done!
- Given that we talked about “forward” kinematics problems, are there also “inverse” kinematics problems?
  - Yes.

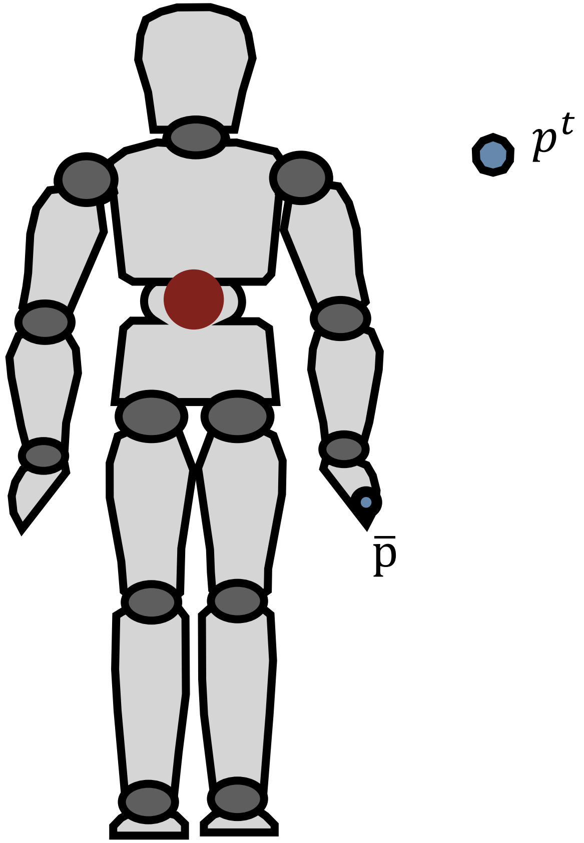
Forward Kinematics vs Inverse Kinematics



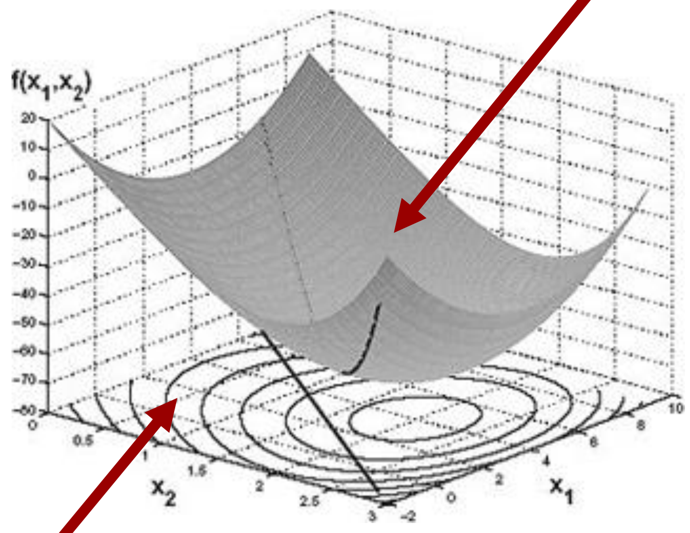
FK: given pose q, where does point $\bar{\mathrm{p}}$ end up in world coordinates, i.e. $\mathrm{p}(\mathrm{q}, \bar{\mathrm{p}})$?

IK: what is the pose q that makes point $\bar{\mathrm{p}}$ reach a target p^t, i.e. $\mathrm{q}(\mathrm{p}^t, \bar{\mathrm{p}})$?

Next class we will see how to formulate IK as an optimization problem



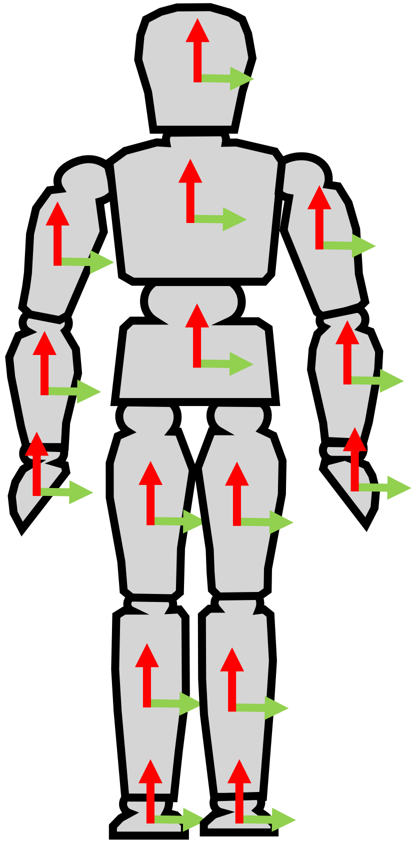
objective function



decision variable domain

$$q(p^t, \bar{p}) = \arg\min_{q^*} |p(q^*, \bar{p}) - p^t|_2^2$$

Any Questions?



Note: Many thanks to Prof. Karen Liu - some of these slides are adapted from her course.

Additional material

- Prof. Karen Liu: https://ckllab.stanford.edu/cs-348e-character-animation
- Prof. Nancy Pollard: http://graphics.cs.cmu.edu/nsp/course/15-462/Spring04/slides/05-hierarchy.pdf
- Radiopaedia: https://radiopaedia.org/articles/joints-1?lang=us
- Physiopedia: https://www.youtube.com/watch?v=l7h2FJnSXyw&feature=emb_logo
- Randale Sechrest: https://www.youtube.com/watch?v=D3GVKjeY1FM
- Affine transformations:
  - https://en.wikipedia.org/wiki/Affine_transformation,
  - http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/
- Rotation matrix:
  - https://www.haroldserrano.com/blog/rotations-in-computer-graphics