WS5 Dynamic Simulation of Assistive Robotics and Human Motion

Day 1: Inverse Modelling

- Intro to simulation, engines and approaches (forward/inverse). (15 m)
- Neuromechanics basics. (15 m)
- Introduction to OpenSim, musculoskeletal modelling and inverse modelling (1 h)
- Q&A + IT help (30 m)

Day 2: Forward Sim + Control

- MuJoCo and forward modelling basics. (25 m)
- Control basics and introducing programming challenge (15 m)
- Arm prototyping exercise (1 h)
- Q&A (10 m)

Day 3: Advanced use-cases

- Reinforcement learning & motion synthesis (10 m)
- Simulations for vision-based robotics (15 m)
- Virtual prostheses, sim2real concerns and methods. (20 m)
- Ankle exo challenge (1h)
- Q&A + buffer (15 m)

WS5 Dynamic Simulation of Assistive Robotics and Human Motion

Day 1: Inverse Modelling

- Intro to simulation, engines and approaches (forward/inverse). (15 m)
- Neuromechanics basics. (15 m)
- Introduction to OpenSim, musculoskeletal modelling and inverse modelling (1 h)
- Q&A + IT help (30 m)

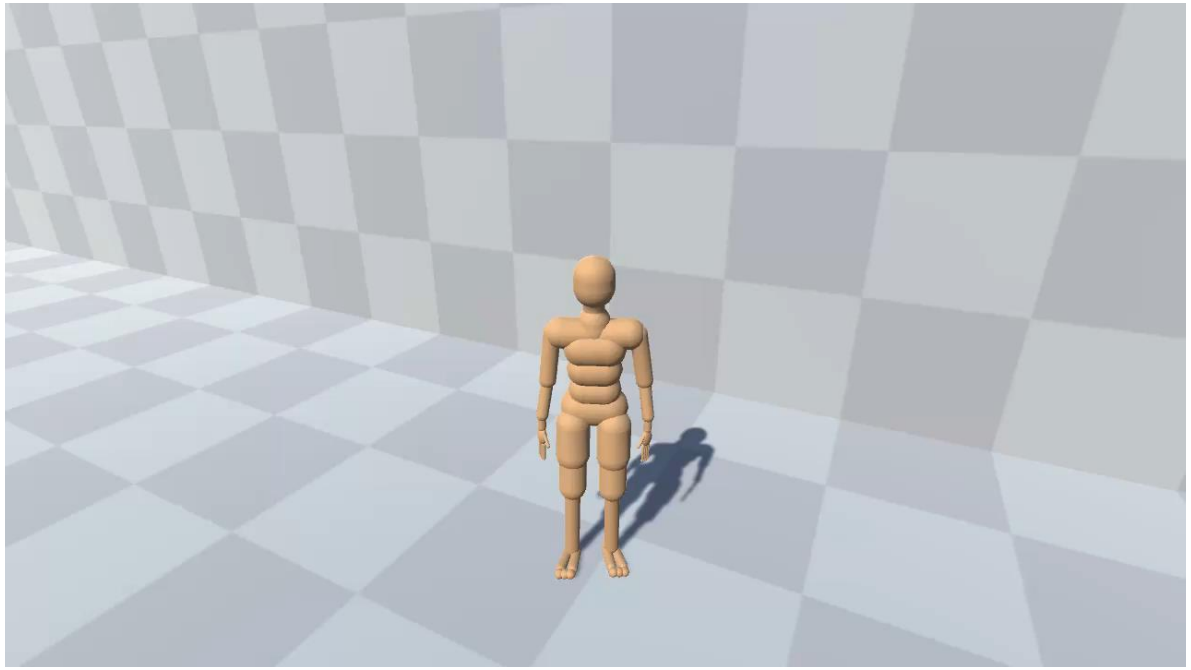
Day 2: Forward Sim + Control

- MuJoCo and forward modelling basics. (25 m)
- Control basics and introducing programming challenge (15 m)
- Arm prototyping exercise (1 h)
- Q&A (10 m)

Day 3: Advanced use-cases

- Reinforcement learning & motion synthesis (10 m)
- Simulations for vision-based robotics (15 m)
- Virtual prostheses, sim2real concerns and methods. (20 m)
- Ankle exo challenge (1h)
- Q&A + buffer (15 m)

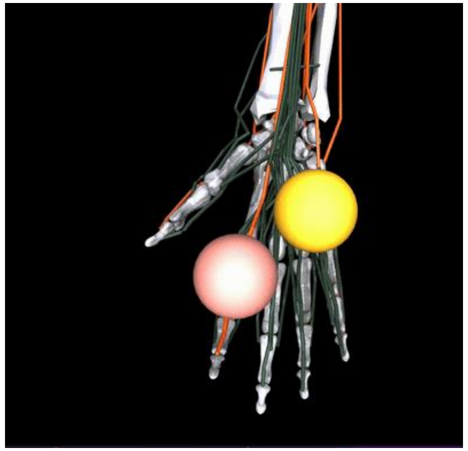
Why go through the trouble of physics-based animation?



Every interaction needs to be set up to not violate our expectations.

Can we enforce them implicitly?

Quick history of MuJoCo



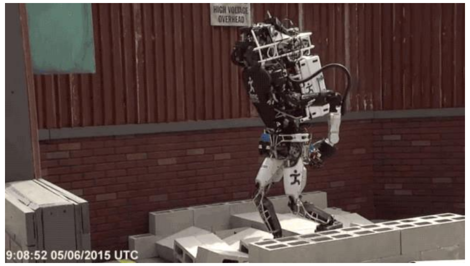
MyoSuite

Optimal control in biomechanics

Robotics Community

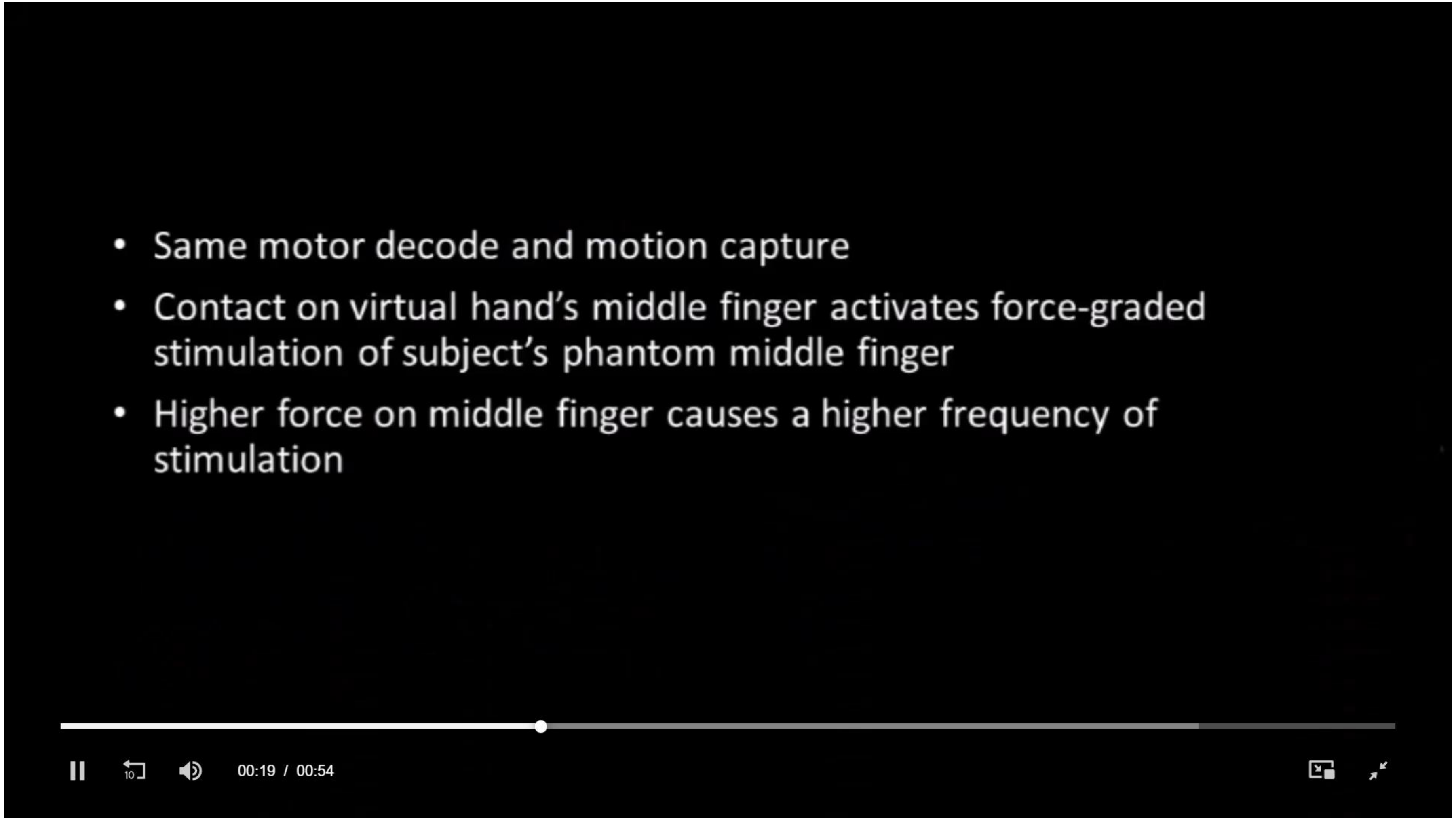
RL Community

DeepMind Acquisition + Open Sourcing



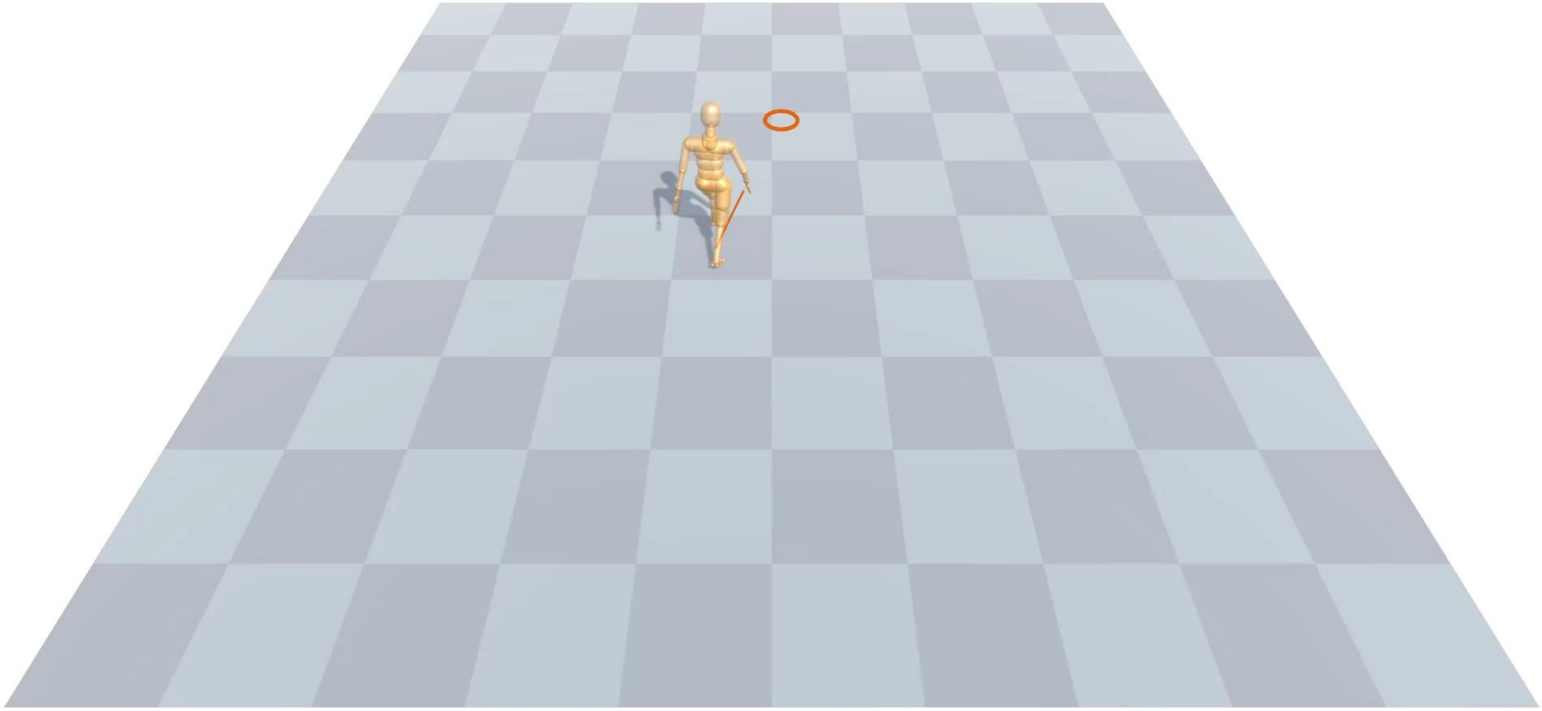
DARPA Robotics challenge

Example use case: Virtual Prototyping

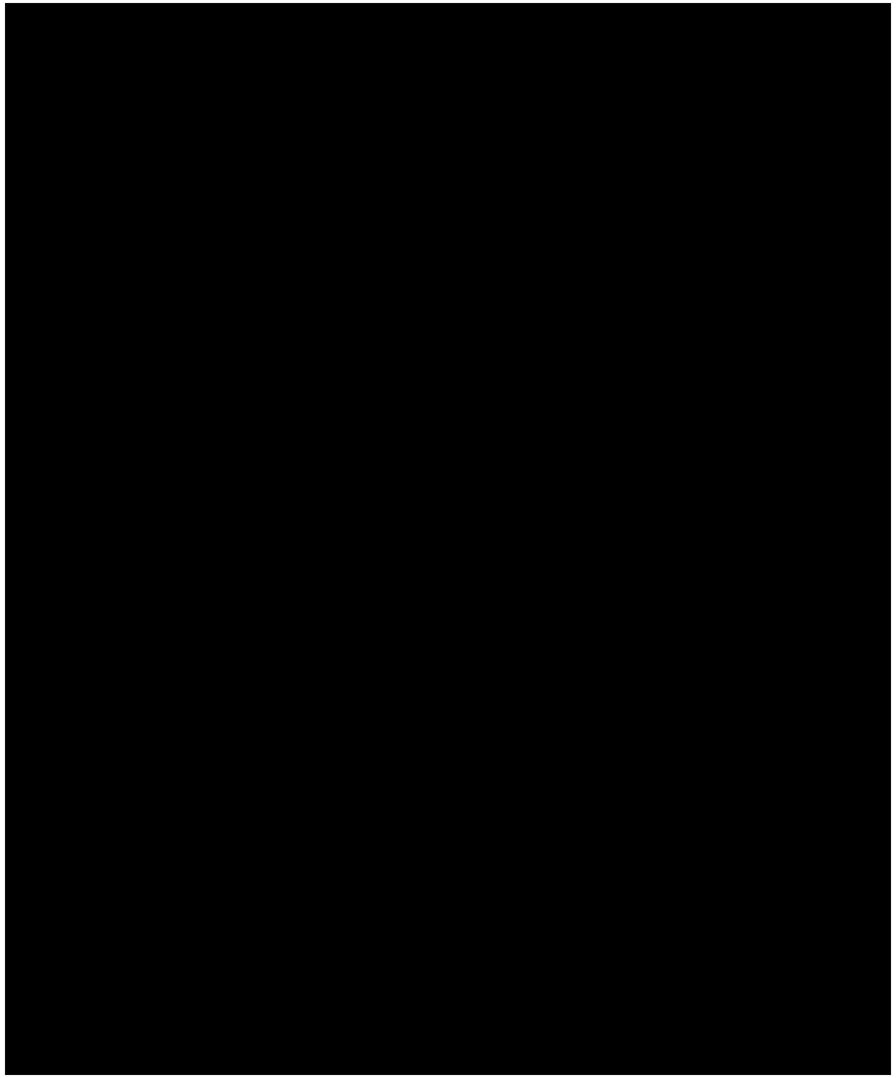


D. T. Kluger et al., "Virtual Reality Provides an Effective Platform for Functional Evaluations of Closed-Loop Neuromyoelectric Control," in IEEE TNSRE 2019

Example use case: Motion Synthesis



Example use case: Digital twin



Example use case: Shared autonomy control with vision based control

Automatically controlled:

- Hand Preshape
- Hand Aperture
- Wrist Rotation

Myoelectrically controlled:

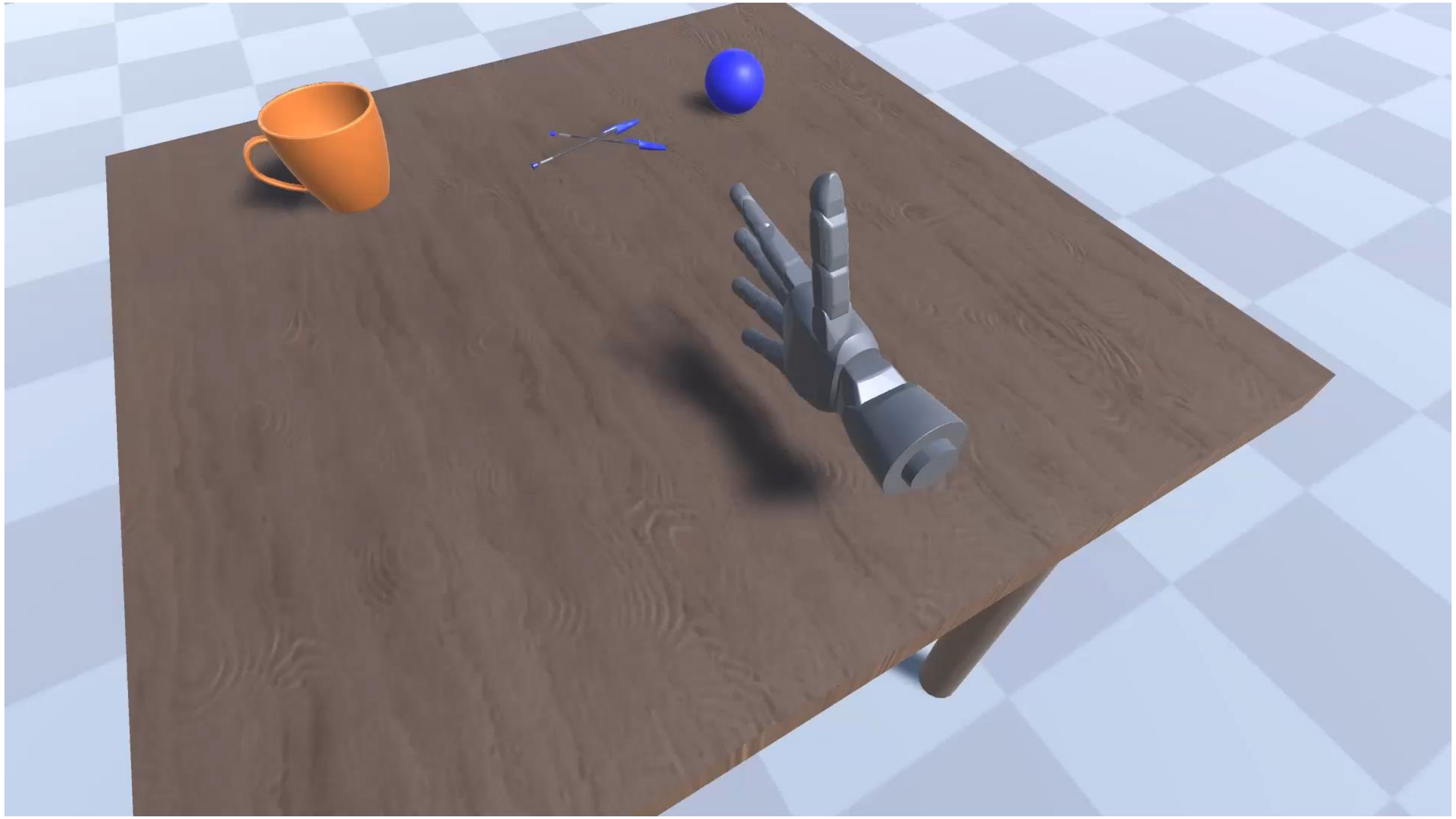
- Grasping/releasing

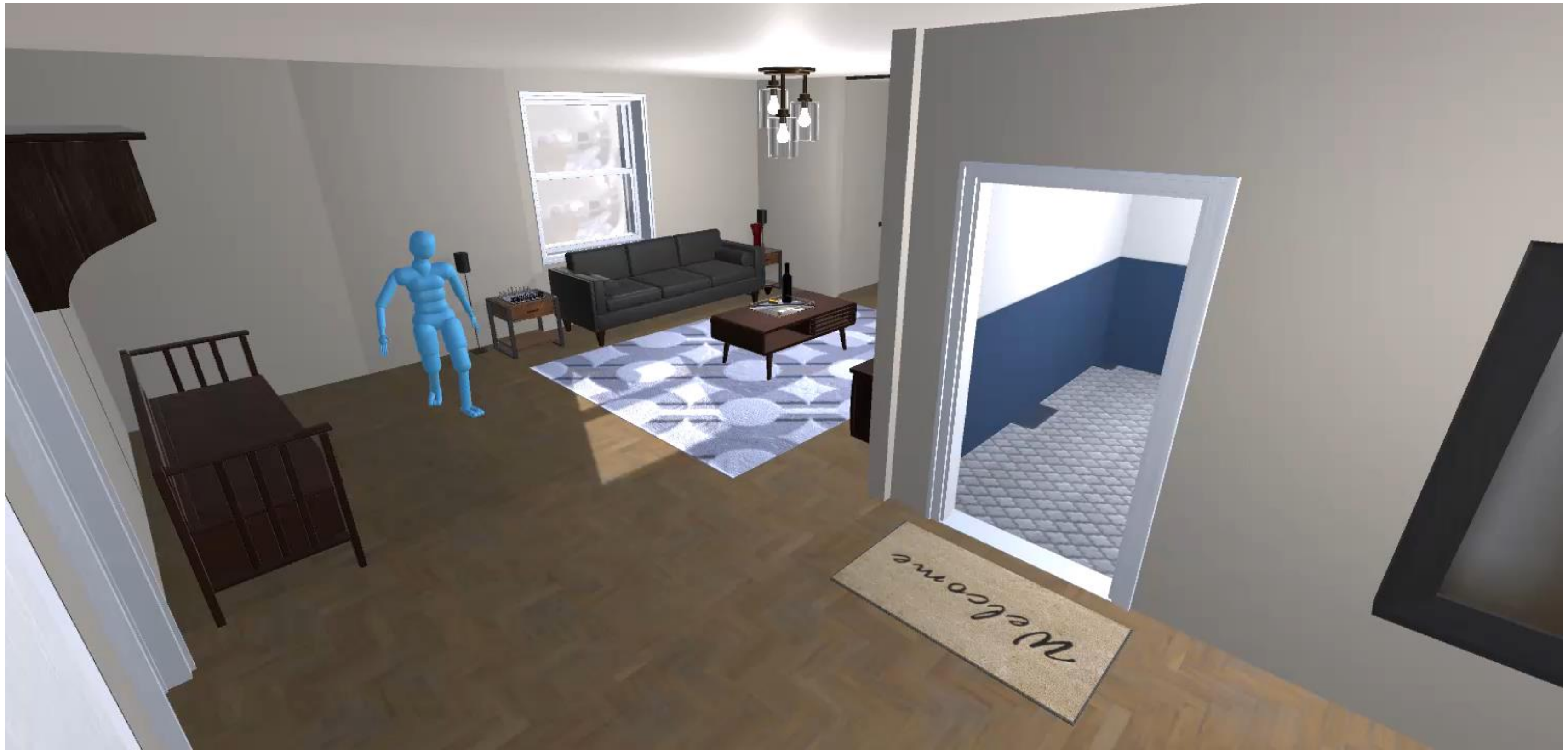
Reduce:

- Compensatory movement
- Muscle activation

Markovic et.al. (2015). Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis. *Journal of Neural Engineering*

Generate datasets or evaluate performance in simulation





Locomotion agent from: Hodossy, B. and Farina, D., 2022. Shared Autonomy Locomotion Synthesis with Virtual Powered Prosthetic Ankle (2023).

Differences from OpenSim

OpenSim has a dedicated GUI (Wrapping the Simbody physics engine)	↔	MuJoCo needs to be combined with a development environment for a proper UI (e.g., Coppelia, Unity)
Elastic Foundations contact model	↔	Optimisation based constraint resolution with force impulses (constraints in velocity vs acceleration)
Muscle-tendon equilibrium resolved dynamically	↔	Tendon assumed to be unstretchable, other parameters adapted to match behaviour
Well explored and understood MSk model library	↔	System-identified robotic model library (but MSk models are being translated from OpenSim)

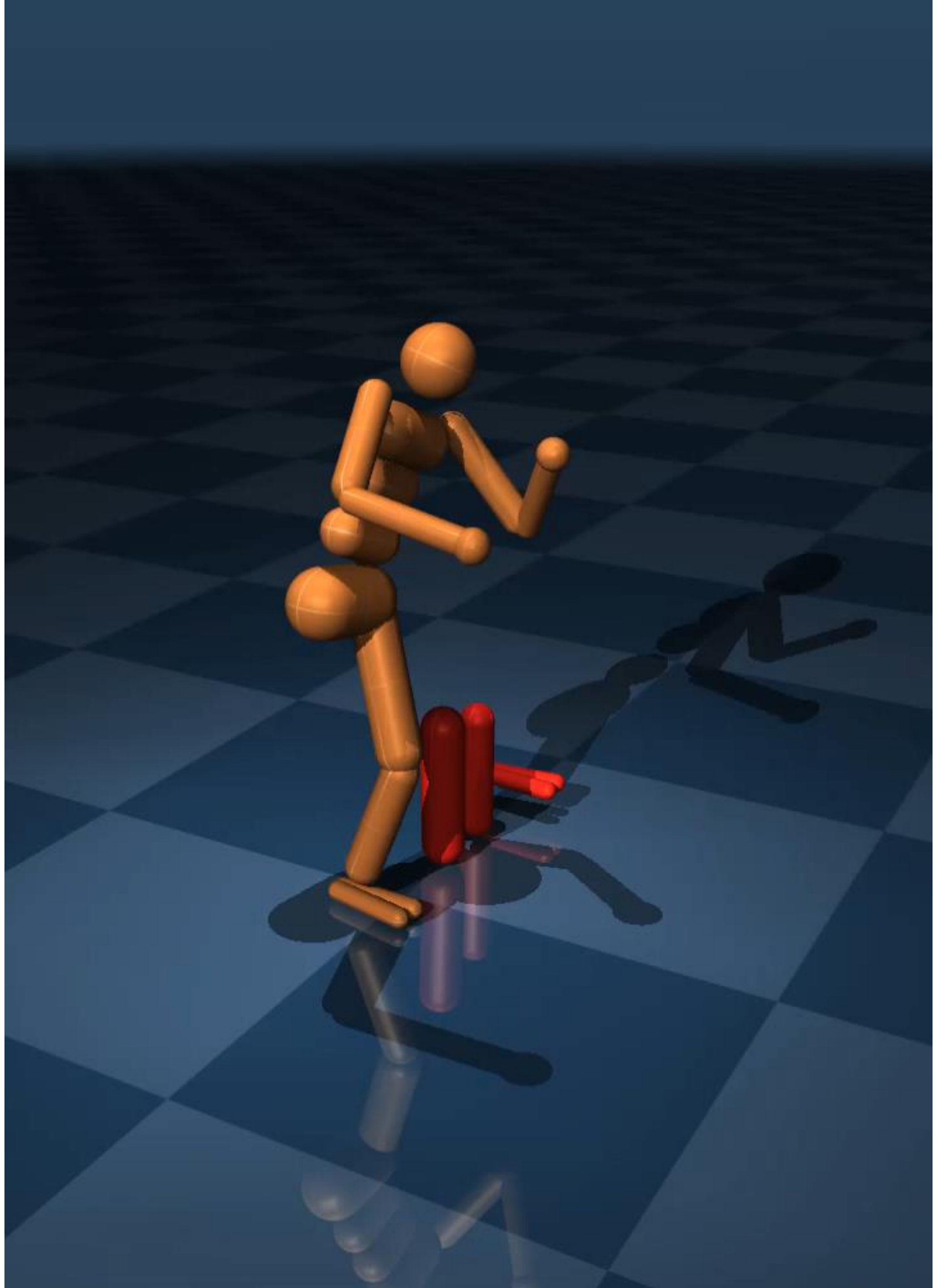
MuJoCo is usually* 60-4000 times faster

Developing with MuJoCo

- Python, Simulink, C++, C# API / wrappers
- Visual GUI in Coppelia Sim or in Unity
- Custom GUIs
- GPU accelerated learning, engine rewritten in Jax (MJX)

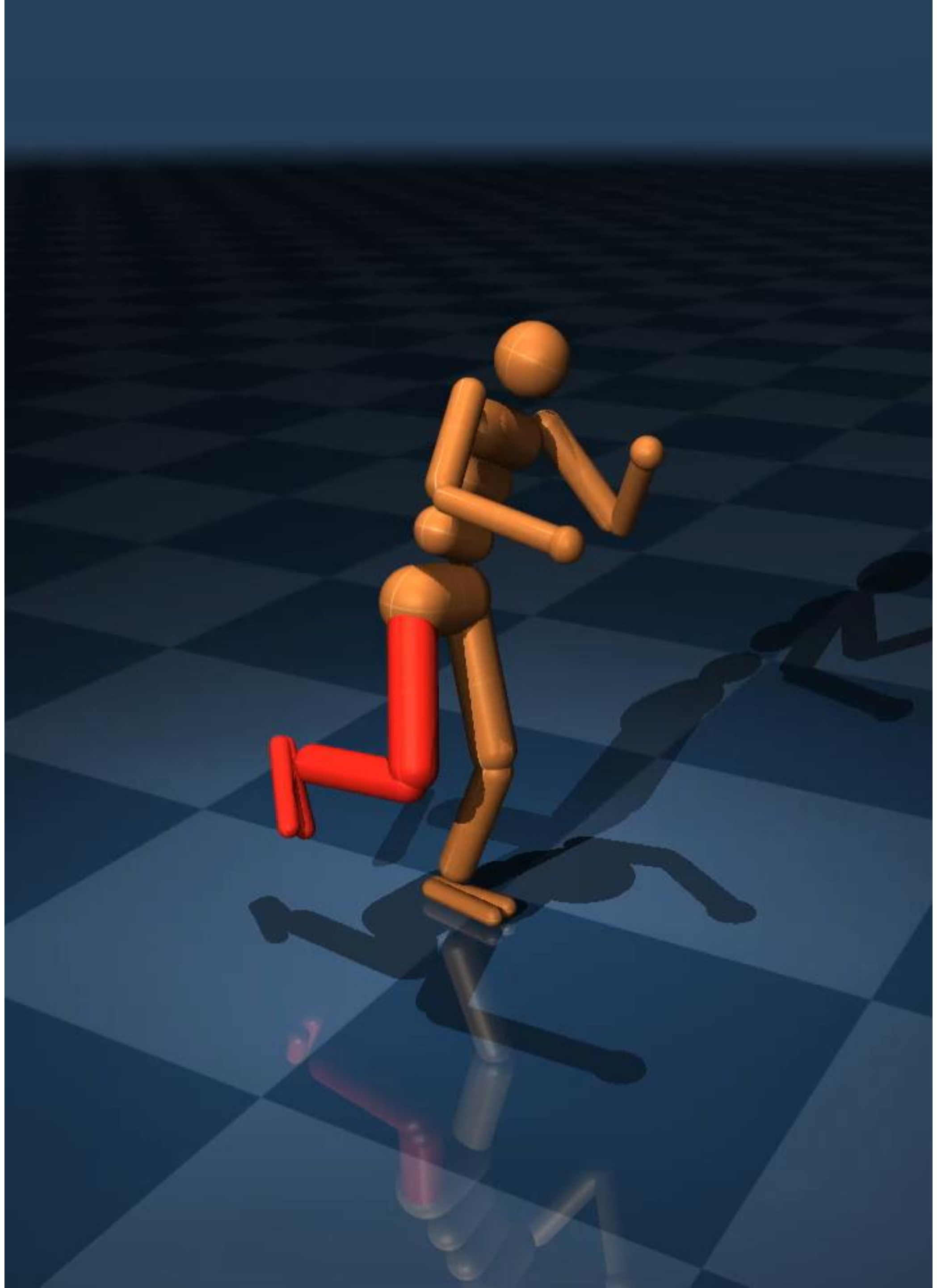
Simulation State

Global Coordinates



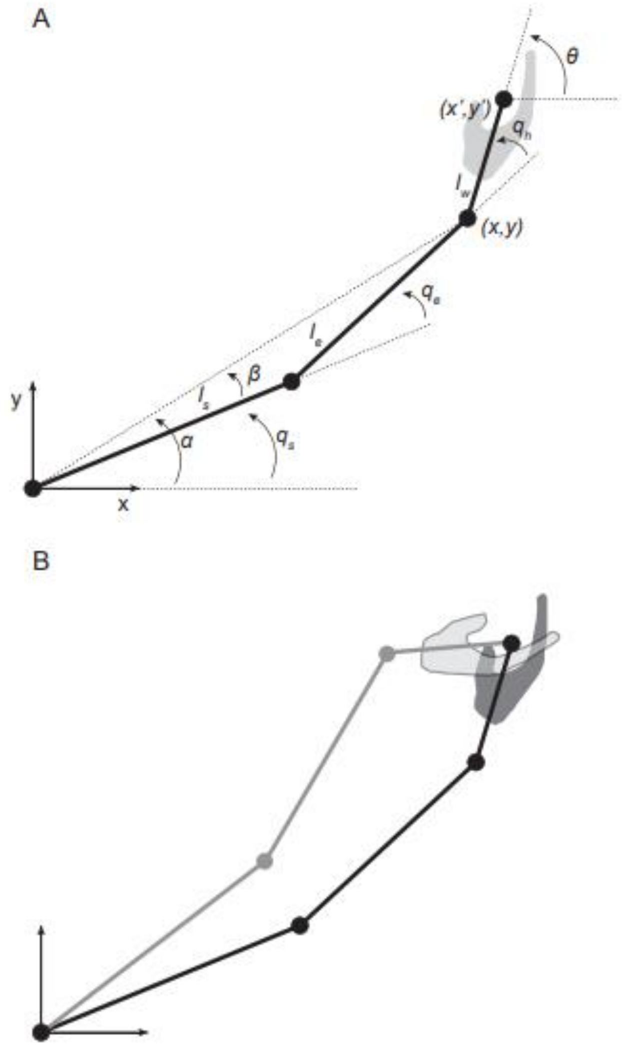
Simulation State

Generalised Coordinates



Simulation State

Not always 1:1 mapping between different state representations



Simulation State

Convert between state representations

$$\mathbf{J}(\mathbf{q}) = \left(\frac{\partial x_i}{\partial q_j} \right) \longrightarrow \partial x = \mathbf{J} \partial q$$

Hand		Joint		Muscle	
$\mathbf{x}$	$\longleftarrow$	$\mathbf{q}$	$\longrightarrow$	$\boldsymbol{\lambda}$	position
$\dot{\mathbf{x}}$	$\xleftarrow{\mathbf{J}}$	$\dot{\mathbf{q}}$	$\xrightarrow{\mathbf{J}_\mu}$	$\dot{\boldsymbol{\lambda}}$	velocity
$\mathbf{F}$	$\xrightarrow{\mathbf{J}^T}$	$\boldsymbol{\tau}$	$\xleftarrow{\mathbf{J}_\mu{}^T}$	$\boldsymbol{\mu}$	force
$\mathbf{K}_x$	$\xrightarrow{\mathbf{J}^T \mathbf{K}_x \mathbf{J} + \frac{d\mathbf{J}^T}{d\mathbf{q}} \mathbf{F}}$	$\mathbf{K}$	$\xleftarrow{\mathbf{J}_\mu{}^T \mathbf{K}_\mu \mathbf{J}_\mu + \frac{d\mathbf{J}_\mu{}^T}{d\mathbf{q}} \boldsymbol{\mu}}$	$\mathbf{K}_\mu$	stiffness
$\mathbf{D}_x$	$\xrightarrow{\mathbf{J}^T \mathbf{D}_x \mathbf{J}}$	$\mathbf{D}$	$\xleftarrow{\mathbf{J}_\mu{}^T \mathbf{D}_\mu \mathbf{J}_\mu}$	$\mathbf{D}_\mu$	damping

Evolving state

Integration based simulation

Equation of motion

$$M\dot{v} \quad = \sum \tau$$

$$M\dot{v} + c = \tau + J^T f$$

Forward modelling:

τ known, what is $\dot{v}$?

$$\dot{v} = M^{-1}(\tau + J^T f - c)$$

Inverse modelling:

$\dot{v}$ known, what is τ?

$$\tau = M\dot{v} + c - J^T f$$

Symbol	Size	Description
n_Q		number of position coordinates
n_V		number of degrees of freedom
n_C		number of active constraints
q	n_Q	joint position
v	n_V	joint velocity
τ	n_V	applied force: passive, actuation, external
$c(q, v)$	n_V	bias force: Coriolis, centrifugal, gravitational
$M(q)$	$n_V \times n_V$	inertia in joint space
$J(q)$	$n_C \times n_V$	constraint Jacobian
$r(q)$	n_C	constraint residual
$f(q, v, \tau)$	n_C	constraint force

Evolving state

Integration based simulation

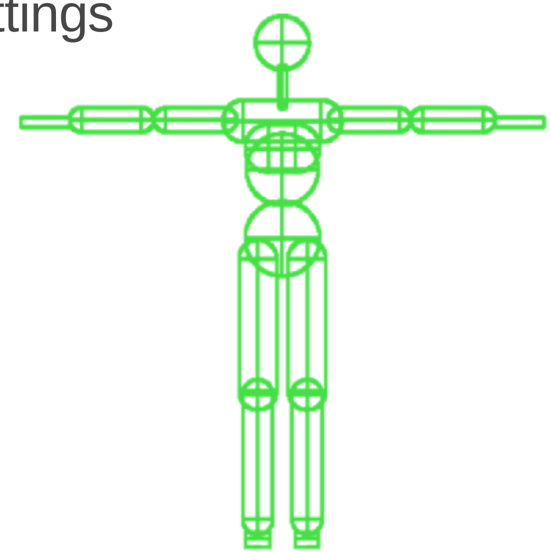
Timestep of h seconds (e.g. 0.005 s)

$$\begin{aligned}\text{velocity:}\quad & v_{t+h} = v_t + h\dot{v}_t \\ \text{position:}\quad & q_{t+h} = q_t + hv_t\end{aligned}$$

System representation in MuJoCo

Model

- Static information about model
- Reference frames, DoFs
- Inertia, geometry, colliders, constraints
- Passive properties, actuation setup
- Sim settings



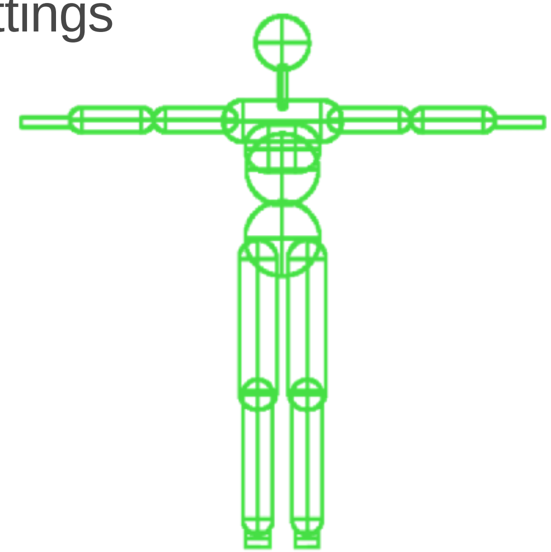
Data

- Dynamic information about the model
- Current pose and derivatives
- Jacobians, mass matrix, contact state
- Actuation and control logic

System representation in MuJoCo

Model

- Static information about model
- Reference frames, DoFs
- Inertia, geometry, colliders, constraints
- Passive properties, actuation setup
- Sim settings



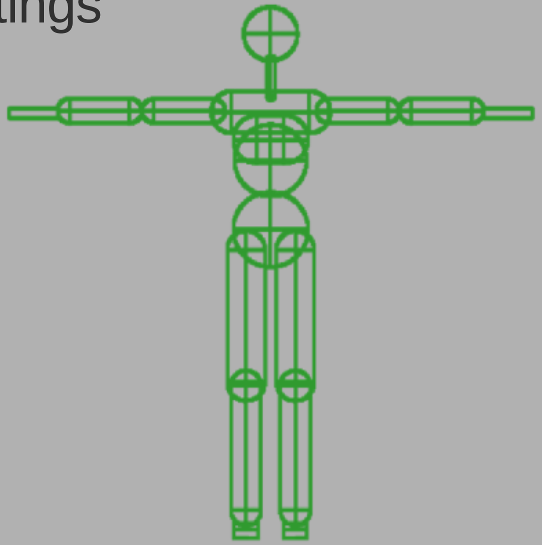
Data

- Dynamic information about the model
- Current pose and derivatives
- Jacobians, mass matrix, contact state
- Actuation and control logic

System representation in MuJoCo

Model

- Static information about model
- Reference frames, DoFs
- Inertia, geometry, colliders, constraints
- Passive properties, actuation setup
- Sim settings



Data

- Dynamic information about the model
- Current pose and derivatives
- Jacobians, mass matrix, contact state
- Actuation and control logic

Simulation Loop

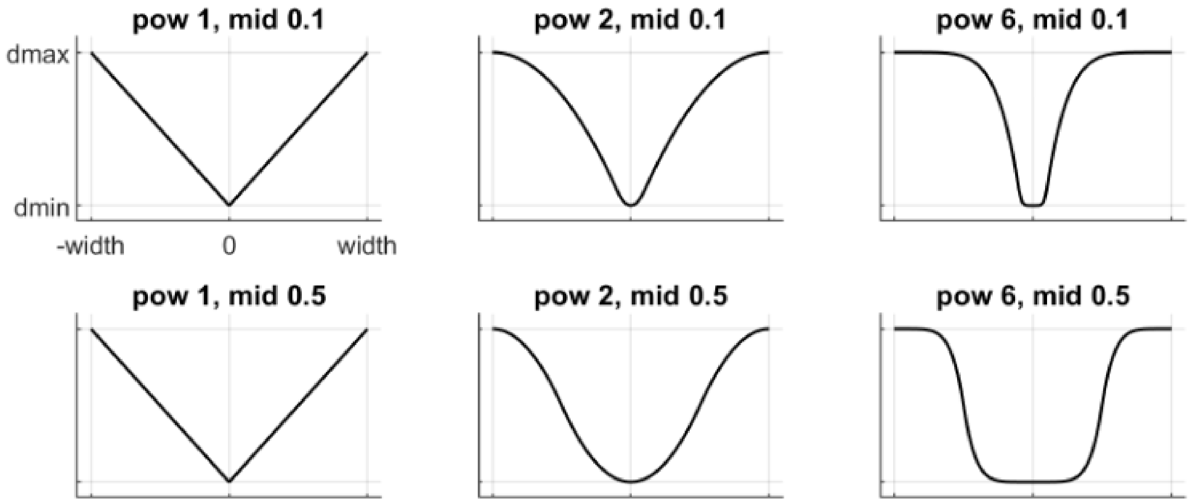
```
void mj_step1(const mjModel* m, mjData* d)
{
  mj_checkPos(m, d);
  mj_checkVel(m, d);
  mj_fwdPosition(m, d);
  mj_sensorPos(m, d);
  mj_energyPos(m, d);
  mj_fwdVelocity(m, d);
  mj_sensorVel(m, d);
  mj_energyVel(m, d);

  // if we had a callback we would be using mj_step, but call it anyway
  if( mjcb_control )
    mjcb_control(m, d);
}

void mj_step2(const mjModel* m, mjData* d)
{
  mj_fwdActuation(m, d);
  mj_fwdAcceleration(m, d);
  mj_fwdConstraint(m, d);
  mj_sensorAcc(m, d);
  mj_checkAcc(m, d);

  // compare forward and inverse solutions if enabled
  if( mjENABLED(mjENBL_FWDINV) )
    mj_compareFwdInv(m, d);

  // integrate with Euler; ignore integrator option
  mj_Euler(m, d);
}
```

Resolving constraints

- Contacts assumed soft, but not purely spring-damper-like

Relaxed complementarity → Allow penetration of rigid bodies

- Can solve as convex optimization, much faster
- “It’s a feature, not a bug.” Model compressibility, recover contact forces from kinematics.
- Choose parameter set to model the interaction you are looking for.



Thank you for the attention!

Feel free to reach out to me if you have questions!
bkh16@ic.ac.uk

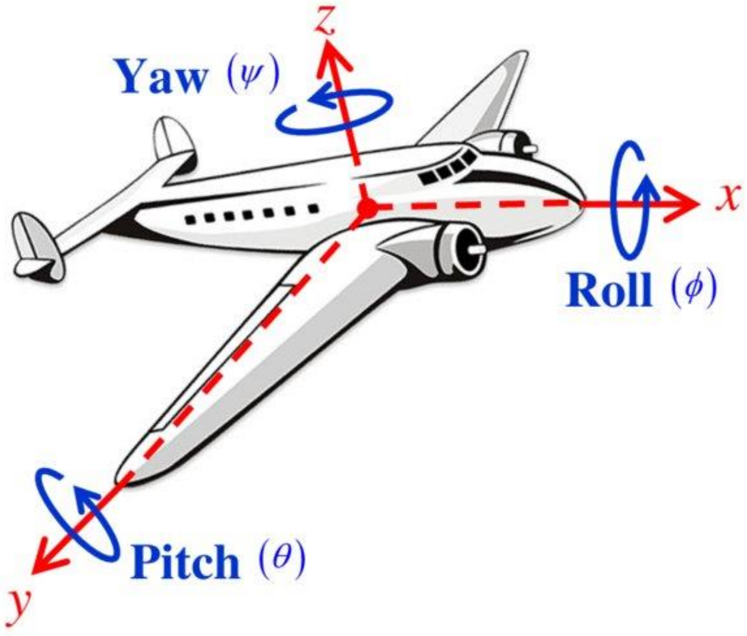
Both OpenSim and MuJoCo has easy to follow tutorials and documentation to getting started, as well as communities that can answer questions and solve problems:
https://github.com/opensim-org/opensim-core
https://github.com/google-deepmind/mujoco

Recommended reading:

De Groote, F. and Falisse, A., 2021. Perspective on musculoskeletal modelling and predictive simulations of human movement to assess the neuromechanics of gait. *Proceedings of the Royal Society B*, *288*(1946), p.20202432.

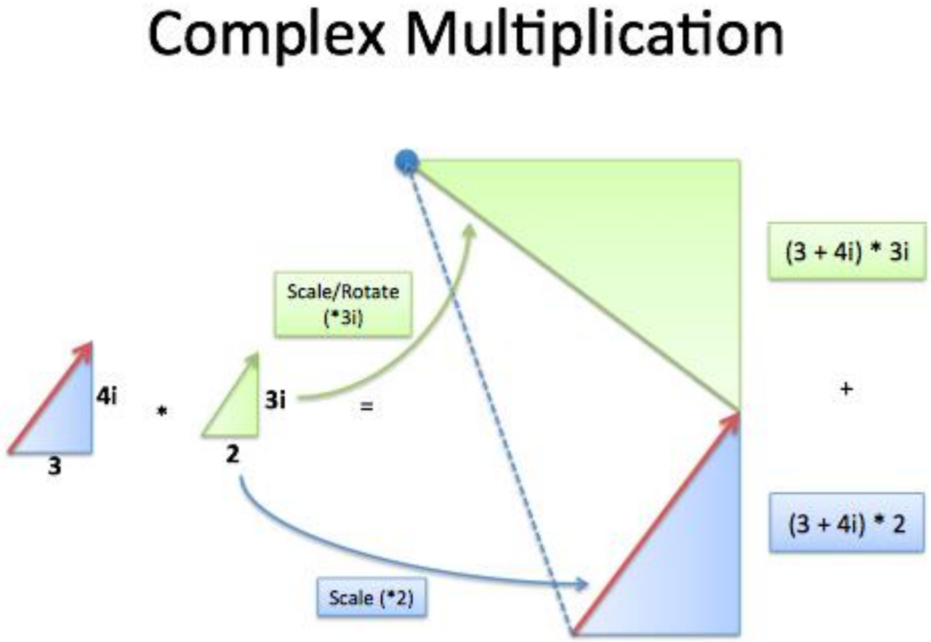
Representing 3 DoF rotations

Euler angles are a simple and intuitive way, but suffer from a few key issues



Representing 3 DoF rotations

Unintuitively, you need more than three numbers to accurately represent 3 DoFs with shared center of rotation (e.g., ball joints or free joints).



Representing 3 DoF rotations

- 4 numbers for position, 3 for velocity

Quaternions extend complex numbers to represent 3D rotations continuously, with familiar, multiplication-based notation.