ETH zürich

# Control Challenges for Dexterous Manipulation


Robert Katzschmann

Soft Robotics Lab

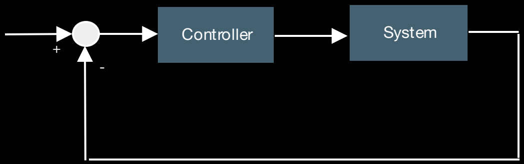

Toshimitsu et al., Getting the ball rolling, Humanoids  (2023)

Soft Robotics Laboratory

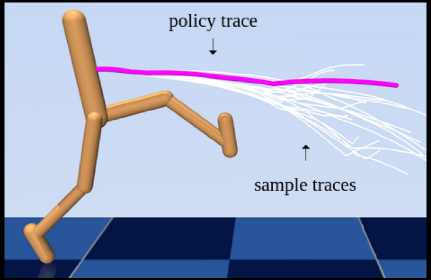
# Plan for Today

## 1. Sensing

## 2a. Feedback Control




## 2b. Model Predictive Control

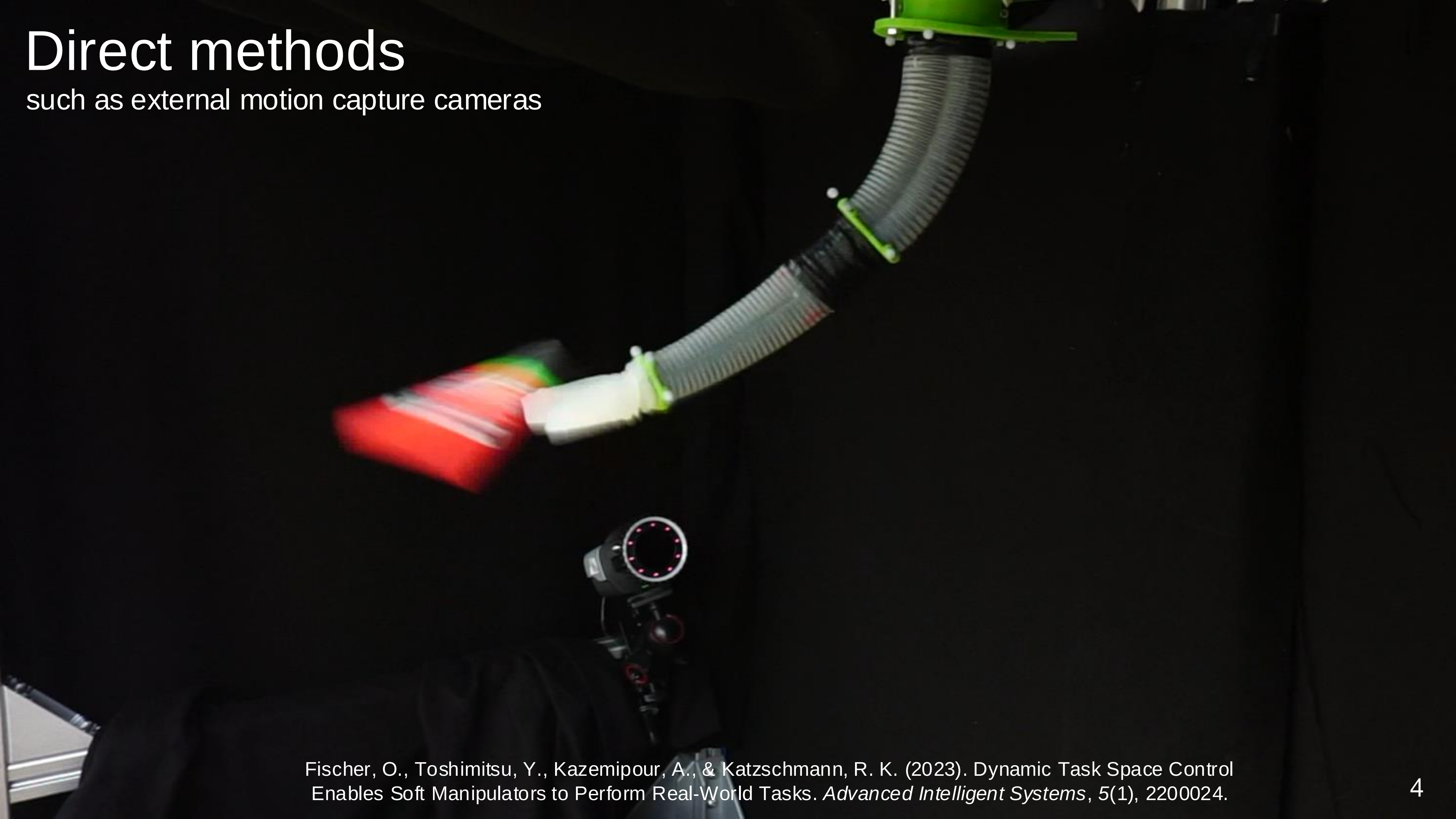



## 3. Challenges

# Part 1: Sensing

# Direct methods

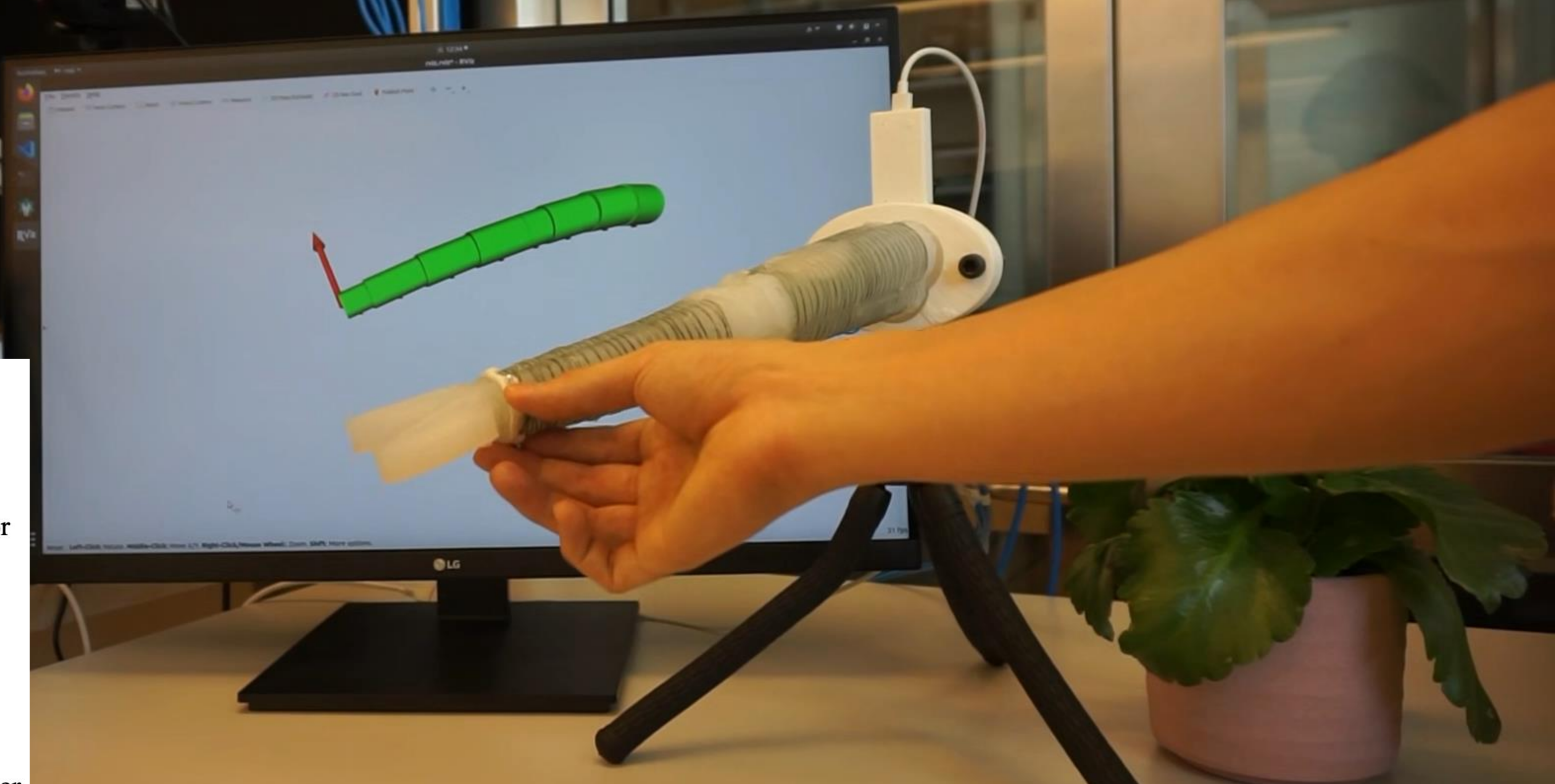
such as external motion capture cameras

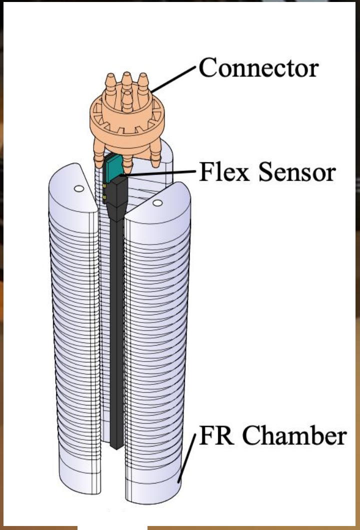


Fischer, O., Toshimitsu, Y., Kazemipour, A., & Katzschmann, R. K. (2023). Dynamic Task Space Control Enables Soft Manipulators to Perform Real-World Tasks. *Advanced Intelligent Systems*, *5*(1), 2200024.

# Indirect methods

such as built-in flex sensors

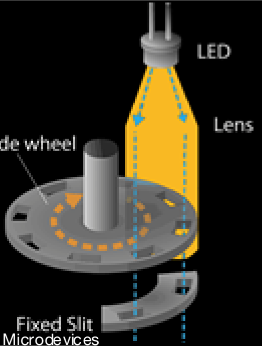





Toshimitsu, Y., Wong, K. W., Buchner, T., & Katzschmann, R. (2021, September). Sopra: Fabrication & dynamical modeling of a scalable soft continuum robotic arm with integrated proprioceptive sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 653-660). IEEE.

# Sensor options

## Rotary Encoders

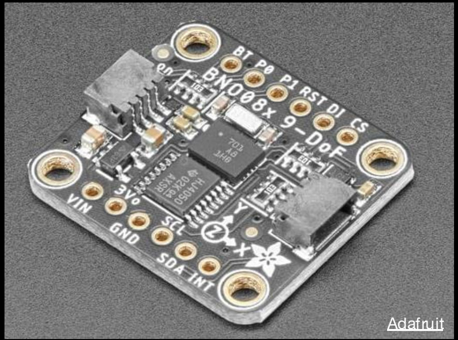



Asahi Kasei Microdevices

## Flex Sensors

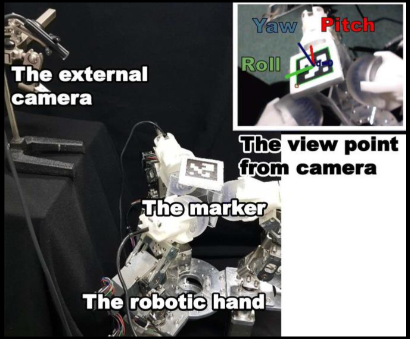
AdaFruit

## Inertial Measurement Unit




Adafruit

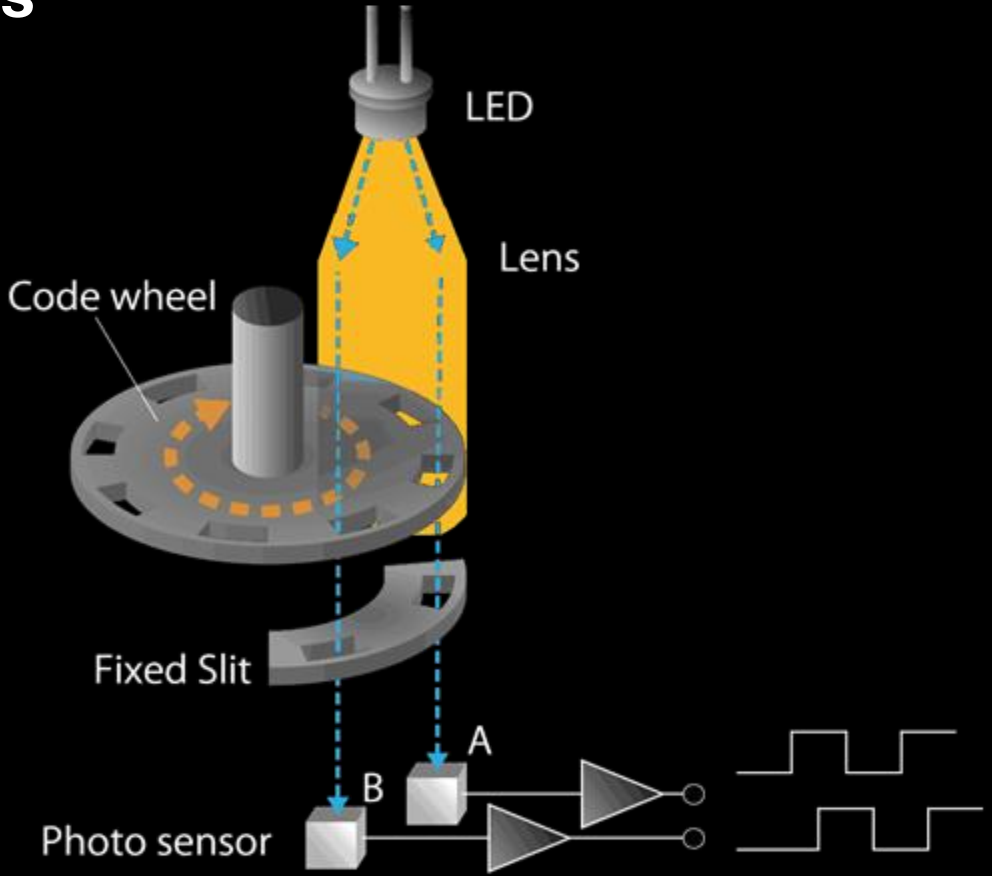
## Camera




Choi, Tahara, Robomech Journal (2020)

ETH zürich SoftRobotics Laboratory

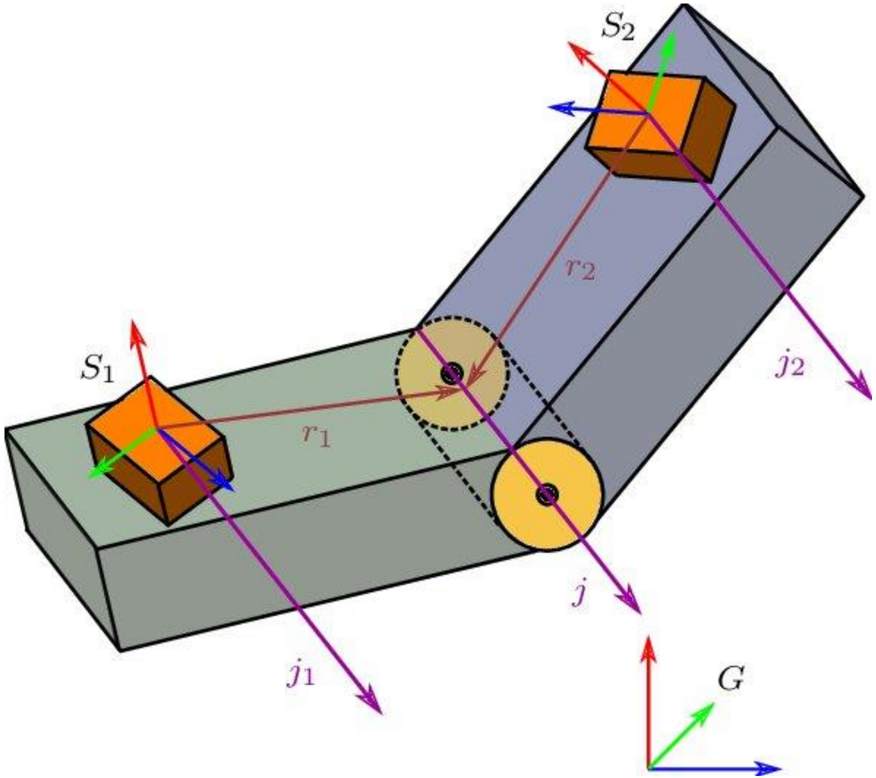
# Rotary Encoders



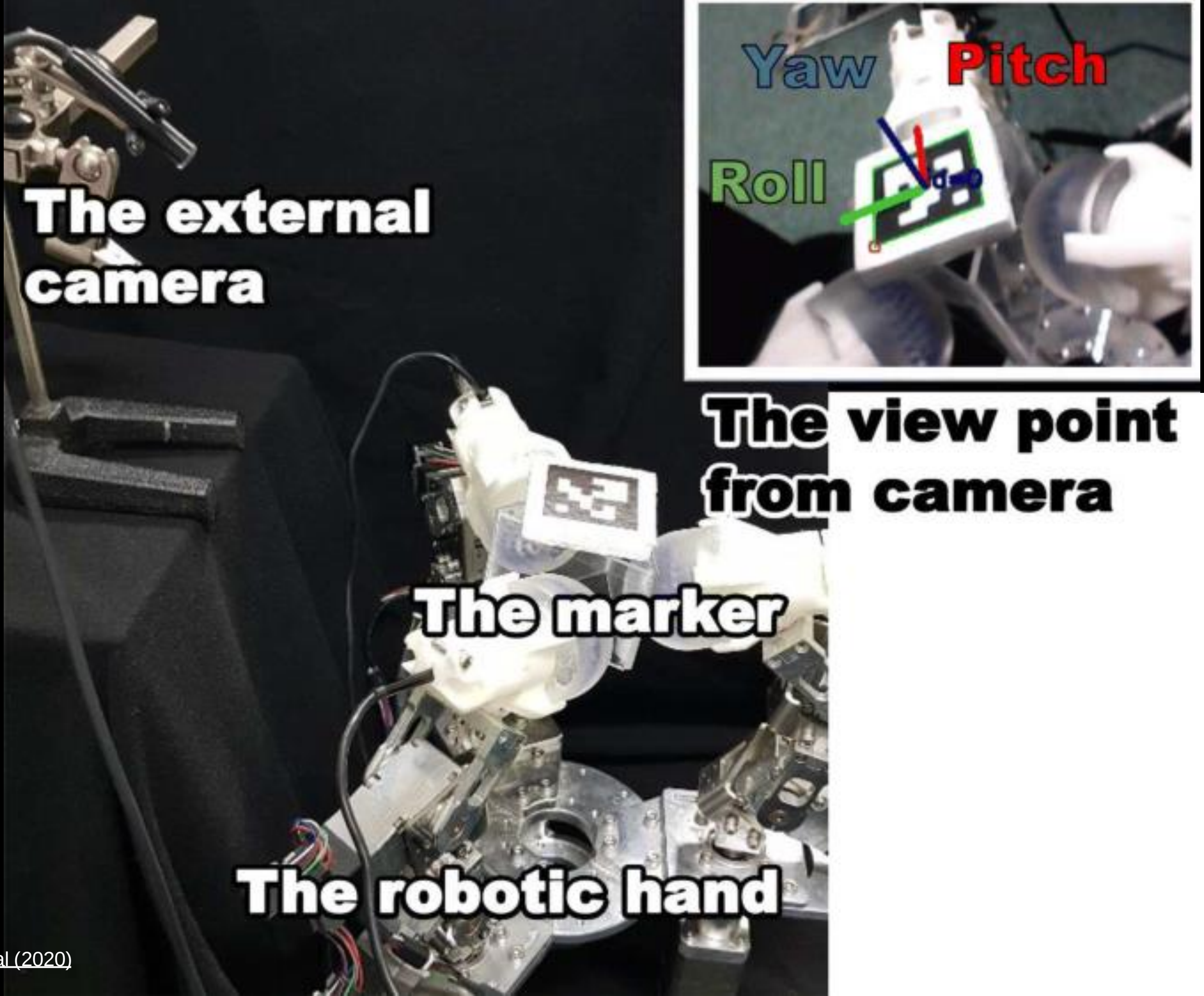
# Flex Sensors

Knecht et al. *Actuation, Sensing and Control of the Faive Robotic Hand*

# Inertial Measurement Unit




Olsson, F., Seel, T., Lehmann, D., & Halvorsen, K. (2019, July). Joint axis estimation for fast and slow movements using weighted gyroscope and acceleration constraints. In *2019 22th International Conference on Information Fusion (FUSION)* (pp. 1-8). IEEE.



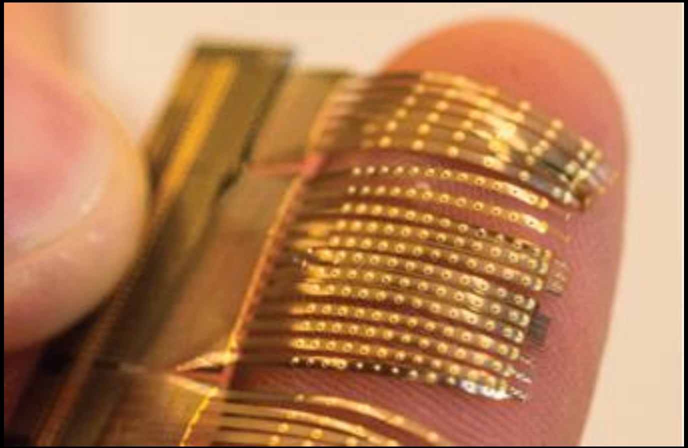

Choi, Tahara, Robomech Journal (2020)

# Sensing the touch:

## Force Sensing Resistors

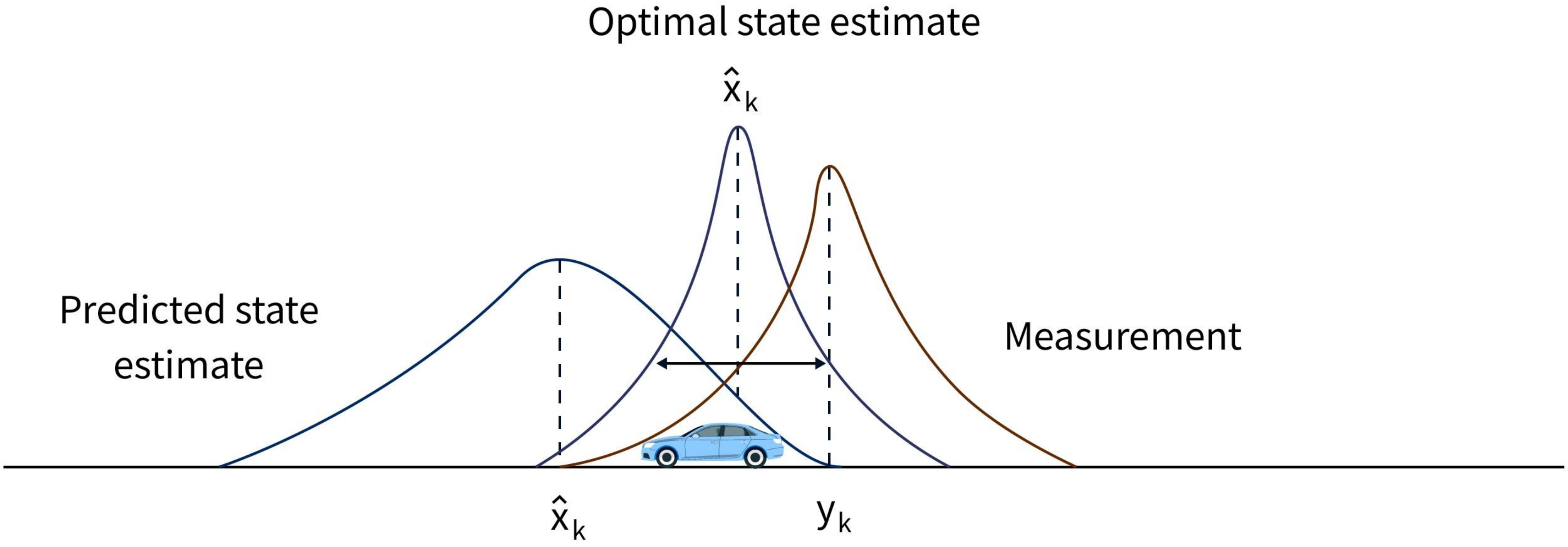

Ohmite

## Artificial Skin

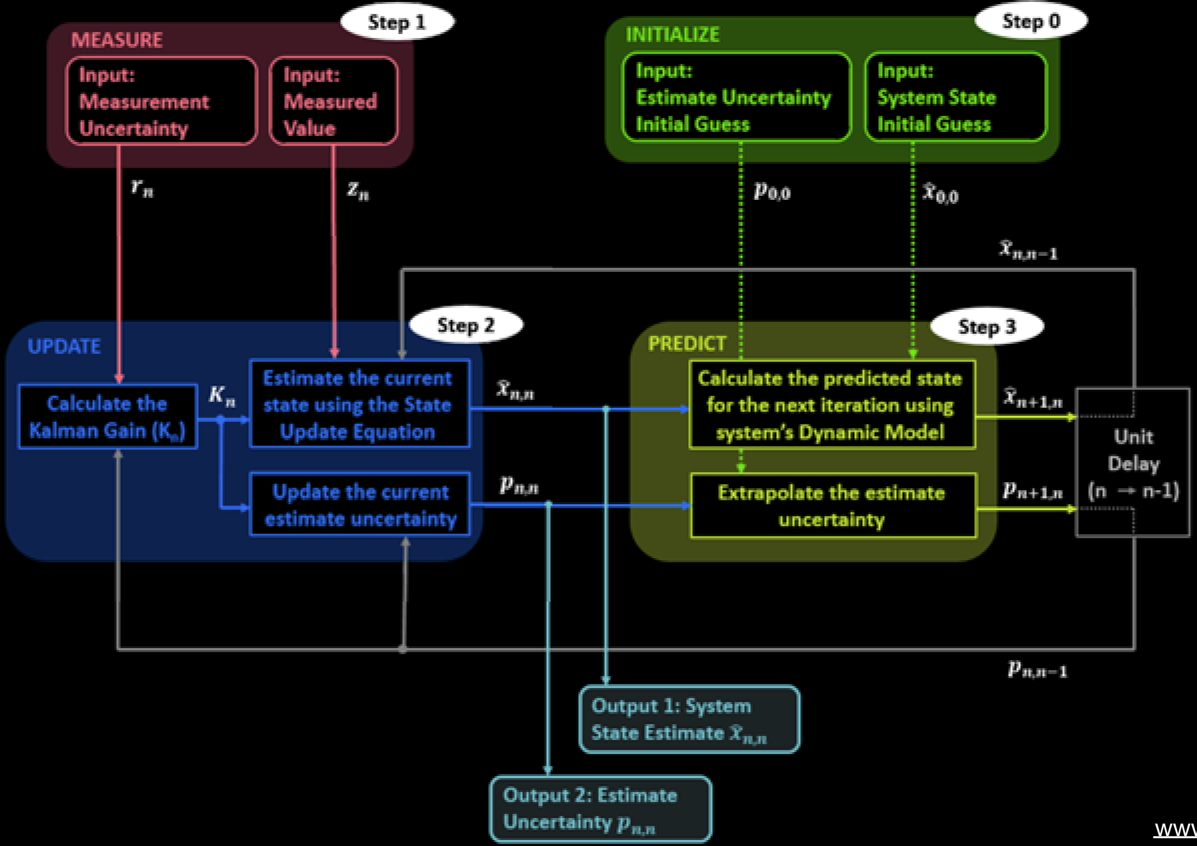


Weichart et al. *Tactile Sensing With Scalable Capacitive Sensor Arrays on Flexible Substrates (2021)*

# Kalman Filter – The Intuition



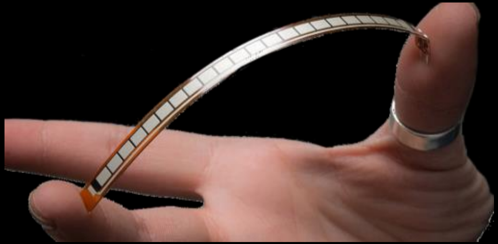
# Kalman Filter




www.kalmanfilter.net

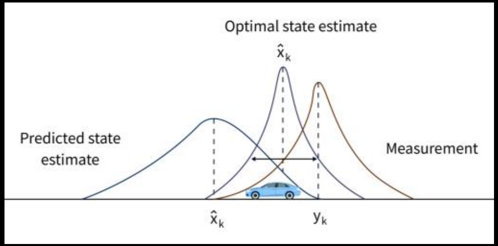
# Sensing Summary

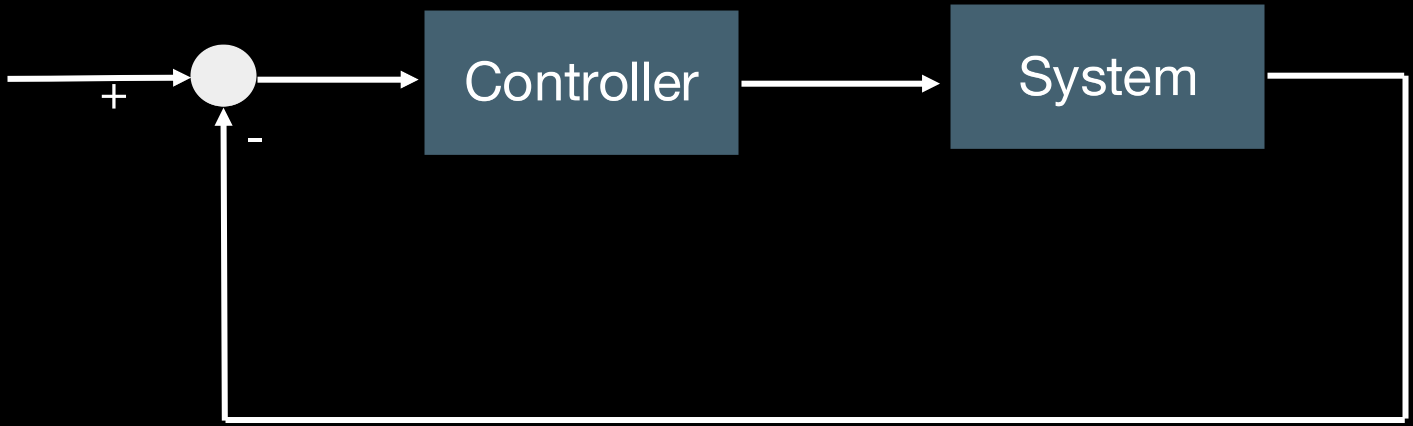
- Pose estimation
  - Measure absolute pose
  - Measuring relative pose



- Force estimation
  - Force Sensing Resistors
  - Artificial Skin

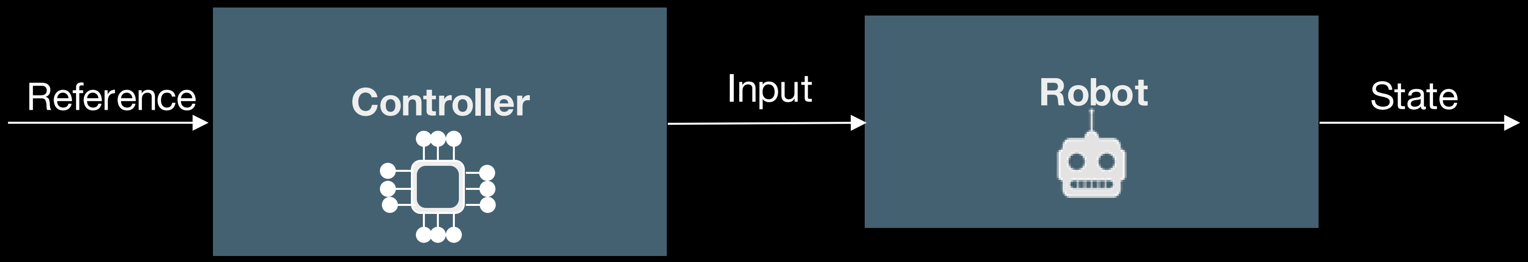
- Kalman Filter



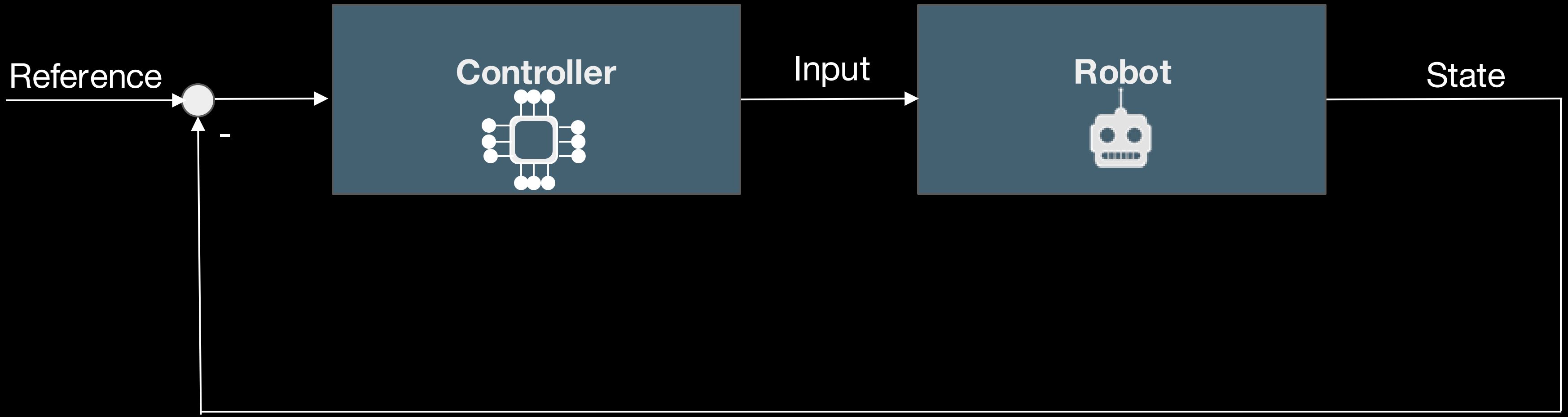



# Part 2a:
# Feedback Control

# Simplest controller possible: Open loop



# Closed Loop Controller



# Inverse Kinematics

- From greek ***kinema*** = motion
- In the past units we learnt that:

$$J(q)\dot{q} = \chi_e = \begin{bmatrix} \dot{p}_e \\ w_e \end{bmatrix}$$

- If we invert it we obtain:

$$\dot{q} = J^{+}\chi_e \text{ with } \mathrm{J}^{+} = J^{T}\left(JJ^{T}\right)^{-1}$$

- And in a differential form:

$$\Delta\chi_e = J^{+}\Delta q$$

**Algorithm 1** Numerical Inverse Kinematics

```
q ← q^0                                          ▷ Start configuration
while ||χ*_e − χ_e(q)|| > tol do                 ▷ While the solution is not reached
    J_eA ← J_eA(q) = ∂χ_e/∂q (q)                 ▷ Evaluate Jacobian for q
    J+_eA ← (J_eA)^+                             ▷ Calculate the pseudo inverse
    Δχ_e ← χ*_e − χ_e(q)                         ▷ Find the end-effector configuration error vector
    q ← q + J+_eA Δχ_e                           ▷ Update the generalized coordinates
end while
```

A possible inverse kinematics algorithm

Robot Dynamics Class @ ETH Zurich

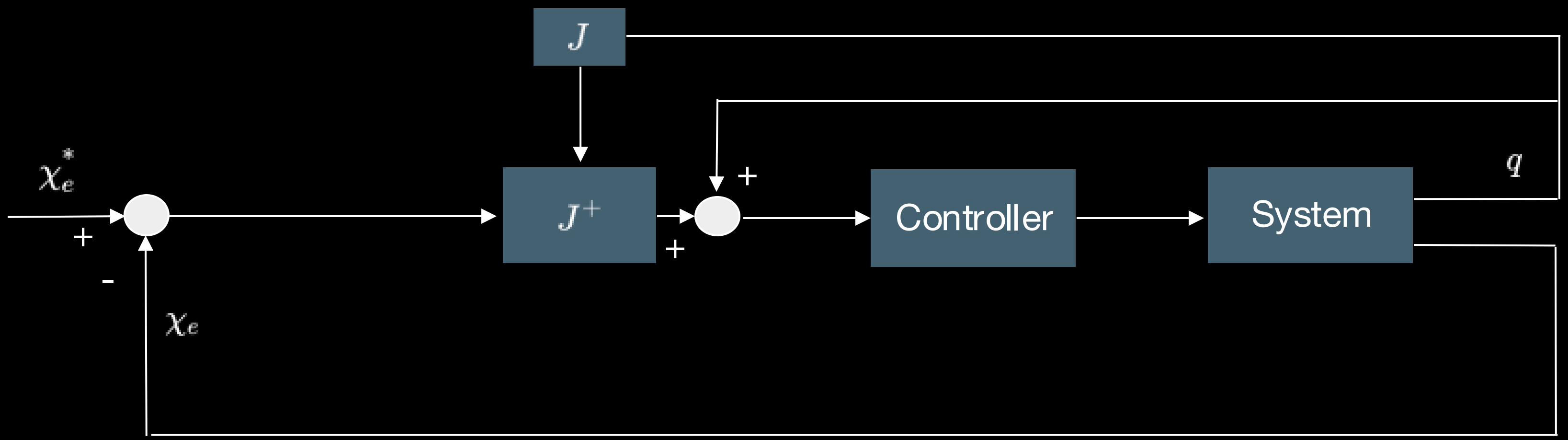
To overcome stability issues, the update can be scaled by a factor $k$

$$q \leftarrow q + kJ_{eA}^{+}\Delta\chi_e \text{ with } k \in (0,1)$$

-> slower convergence

ETH zürich SoftRobotics Laboratory

# Inverse Kinematics Control



# Trajectory Control

We can use a closed loop controller, but we need to add a component for the desired velocities

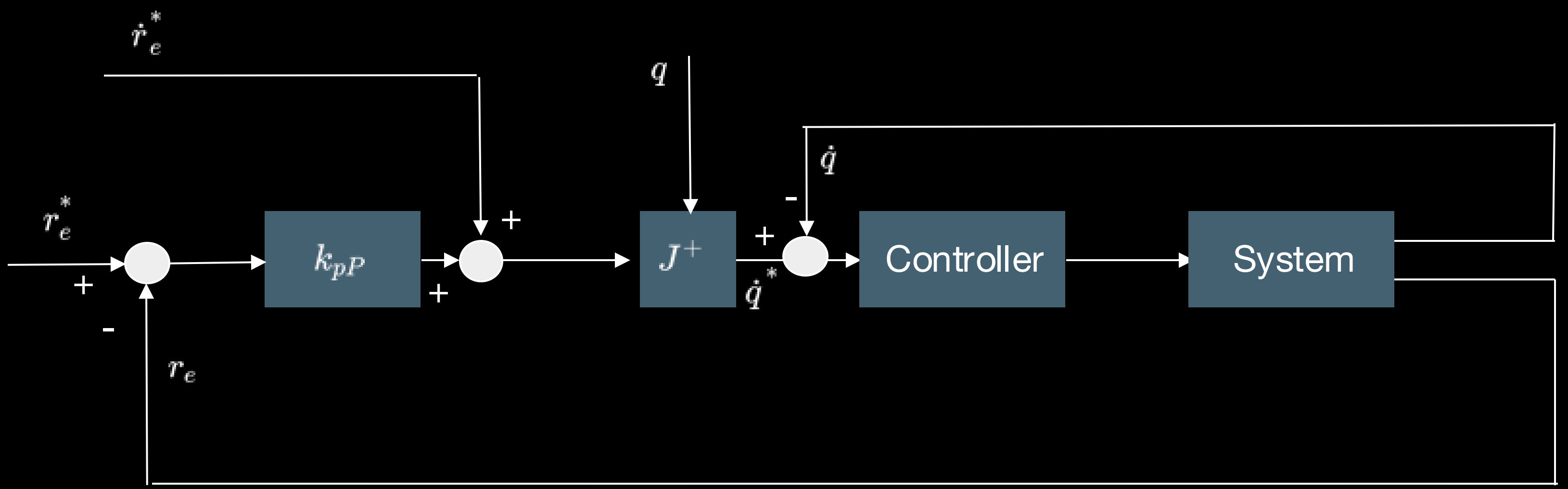
We define $\Delta r_e^t = r_e^*(t) - r_e(q^t)$

And the desired joint velocity $\dot{q}^* = J_{e0_P}^+(q^t) \cdot (\dot{r}_e^*(t) + k_{pP} \Delta r_e^t)$

If we have a desired rotation rate we write $\dot{q}^* = J_{e0_R}^+(q^t) \cdot (\omega_e^*(t) + k_{pR} \Delta \phi)$

Where $\phi$ are the angles used to represent the orientation of the end effector.

# Trajectory Control



# Dynamic control

The dynamic model is

$$M(q)\ddot{q} + b(q,\dot{q}) + g(q) = \tau + J_c(q)^T F_c$$

With:

$M(q)$ : Generalized mass matrix

$q, \dot{q}, \ddot{q}$ : Generalized position, velocity and acceleration vector

$b(q, \dot{q})$ : Coriolis and centrifugal terms

$g(q)$ : Gravitational terms

$\tau$ : External generalized forces

$F_c$ : External Cartesian forces

$J_c(q)$ : Geometric Jacobian corresponding to the external forces

# Dynamic control

The dynamic model is

$$M(q)\ddot{q} + b(q,\dot{q}) + g(q) = \tau + J_c(q)^T F_c$$

If we know the desired generalized accelerations, velocities and poses we can write

$$\ddot{q}^* = k_p(q^* - q) + k_d(\dot{q}^* - \dot{q})$$

Thus the joint torques will be

$$\tau^* = M(q)\ddot{q}^* + b(q,\dot{q}) + g(q)$$

# Task-space control

Remember that $J(q)\dot{q} = \chi_e = \begin{bmatrix} \dot{p}_e \\ w_e \end{bmatrix}$

If you derive that with respect to time: $\dot{\chi}_e = J(q)\ddot{q} + \dot{J}(q)\dot{q}$

And if we solve the dynamics equation for the joint acceleration and substitute in the equation above we get: $\dot{\chi}_e = JM^{-1}(\tau - b - g) + \dot{J}\,\dot{q}$

Finally, remembering that $\tau = J_e^T F_e$

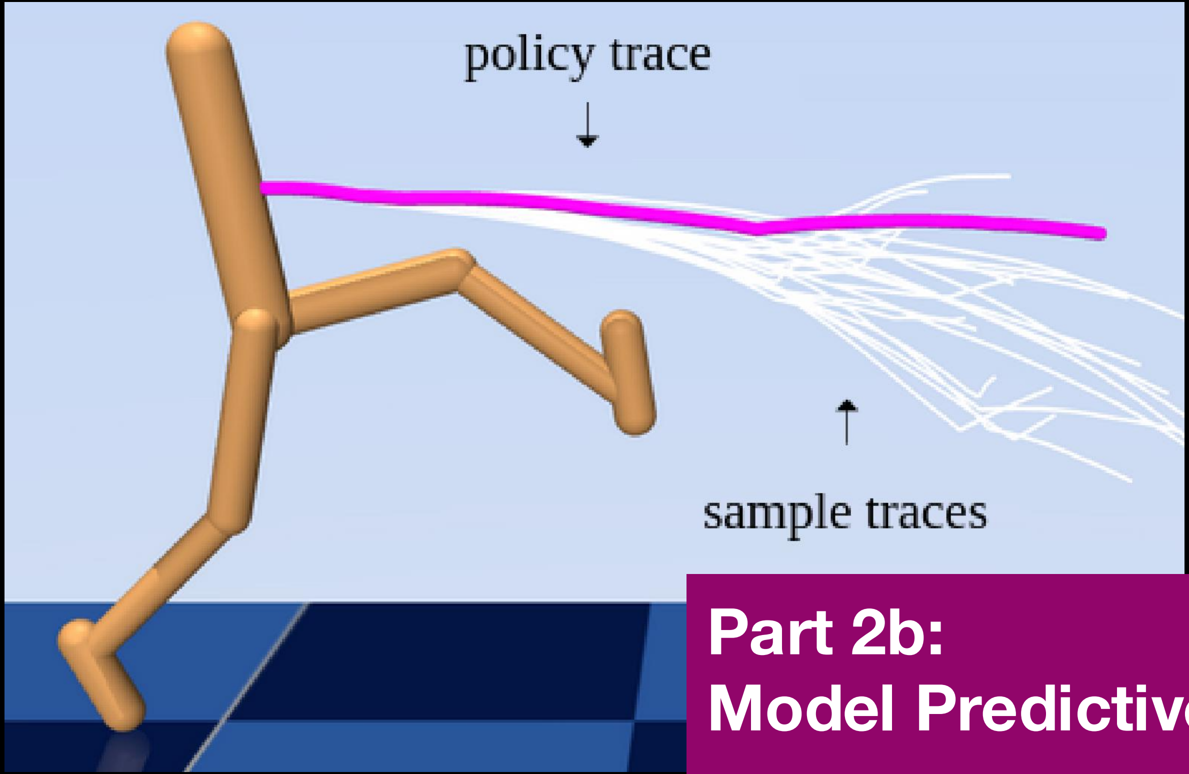
We can write $\Lambda_e \dot{\chi}_e + \mu + p = F_e$

$$\begin{aligned} \Lambda_e &= (J_e M^{-1} J_e^T)^{-1} \\ \mu &= \Lambda_e J_e M^{-1} b - \Lambda_e \dot{J}_e \dot{q} \\ p &= \Lambda_e J_e M^{-1} g \end{aligned}$$

# Task-space control

Defining the dynamics uniquely depending on the state of the end effector allows us to design a control loop

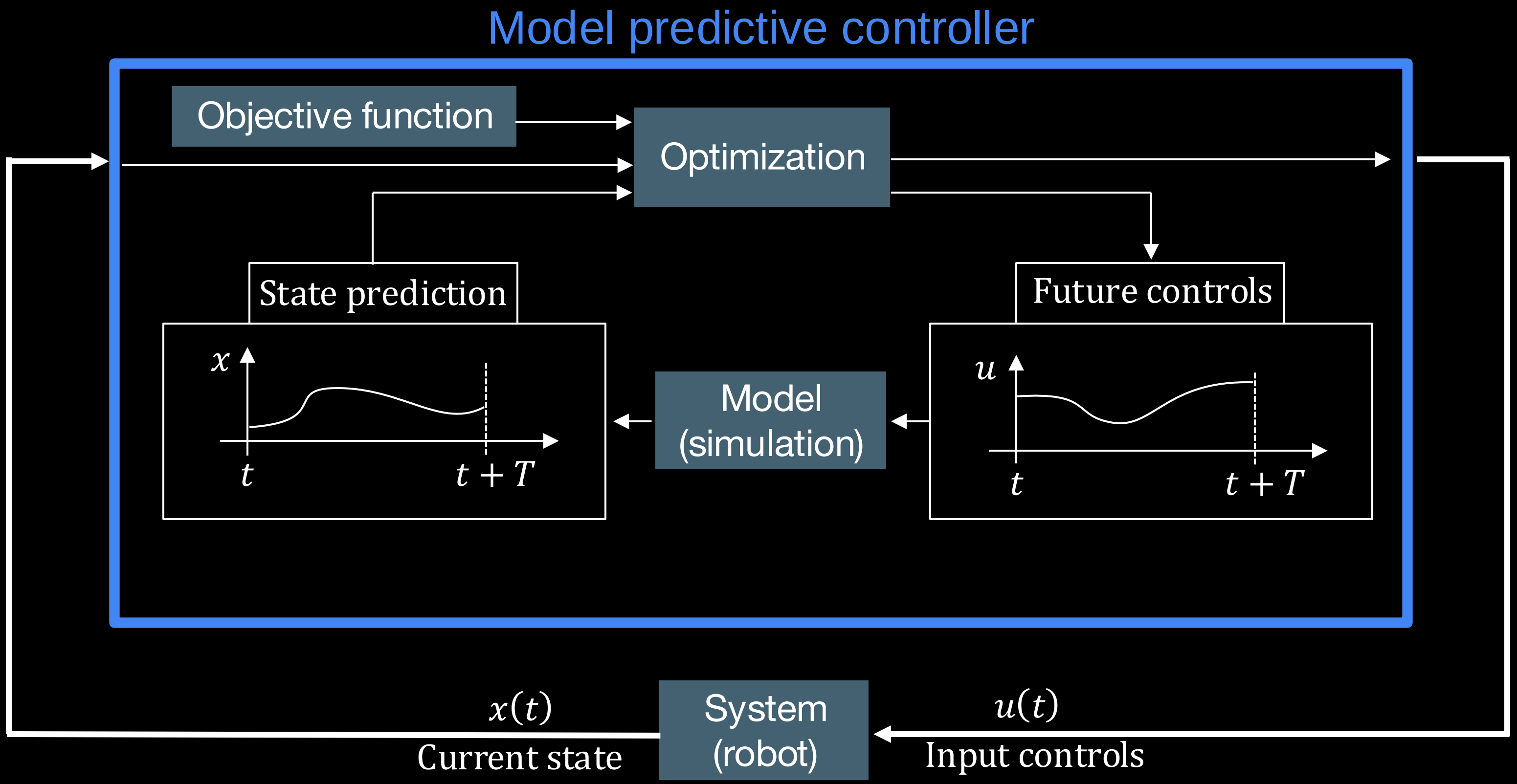
$$\dot{\chi}_e^* = \begin{pmatrix} r_e^* - r_e \\ \Delta\phi_e \end{pmatrix} + k_d(\chi_e^* - \chi_e)$$



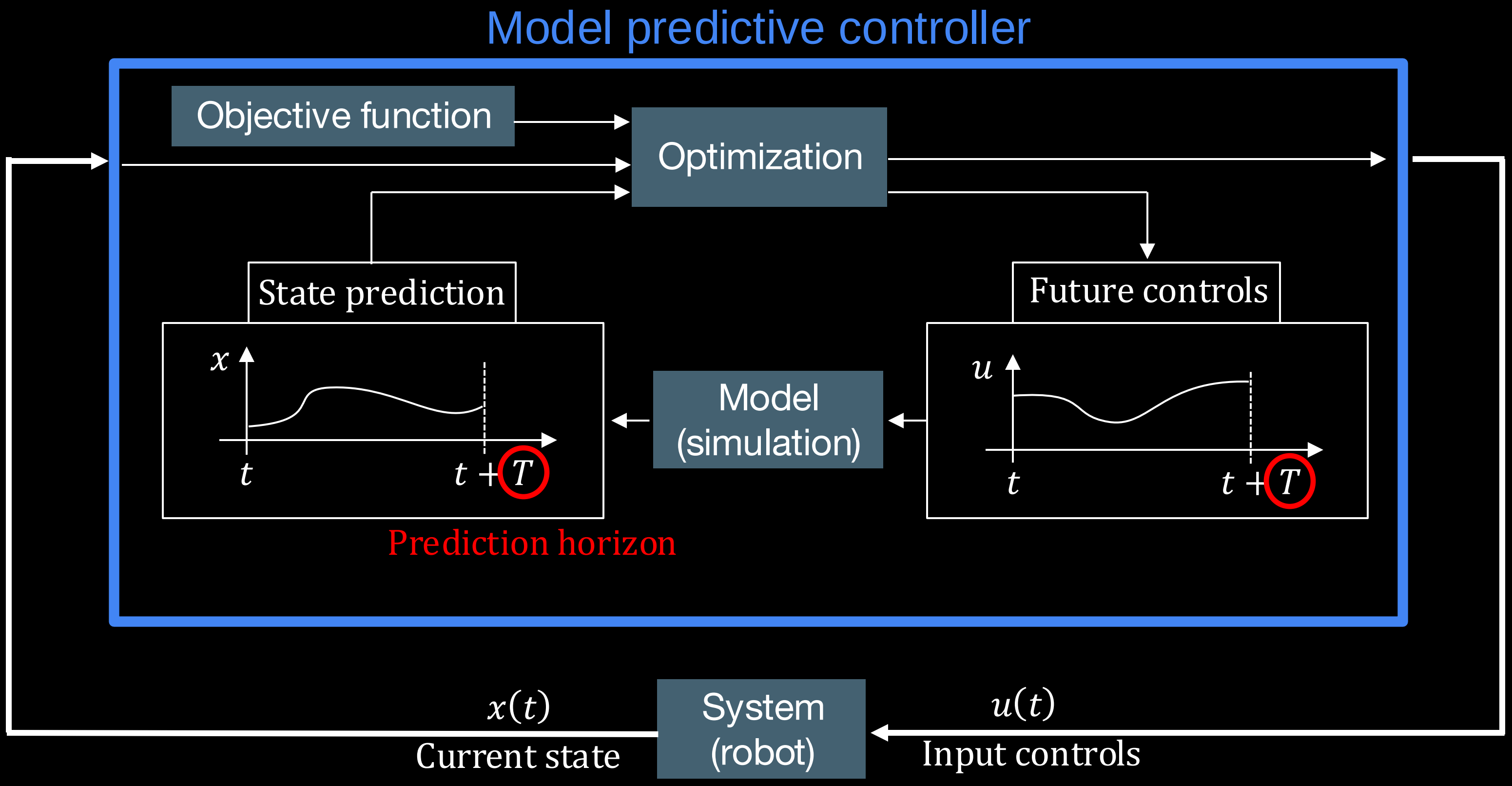

Mujoco PC

# Part 2b: Model Predictive Control

# Model predictive control

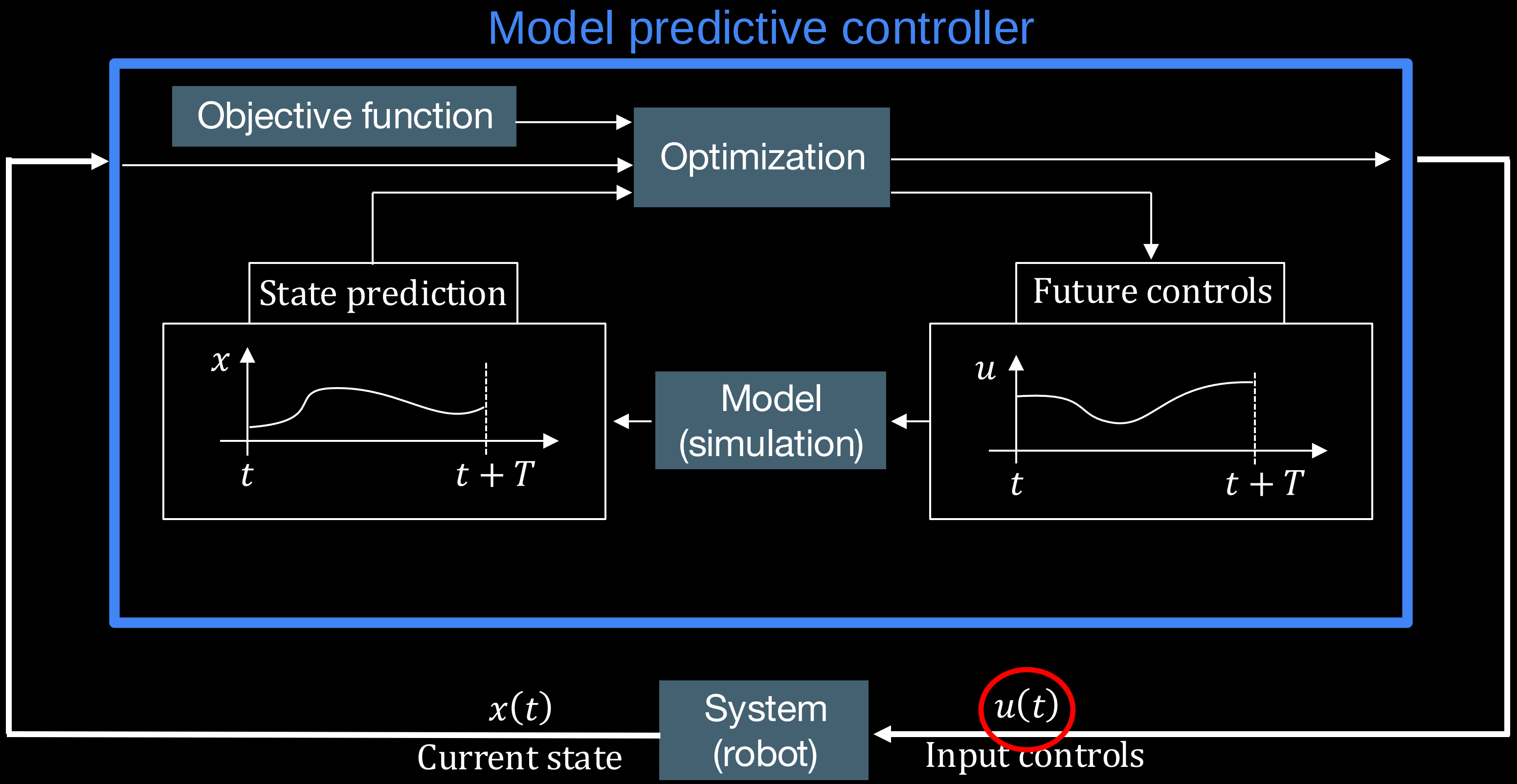


# Model predictive control



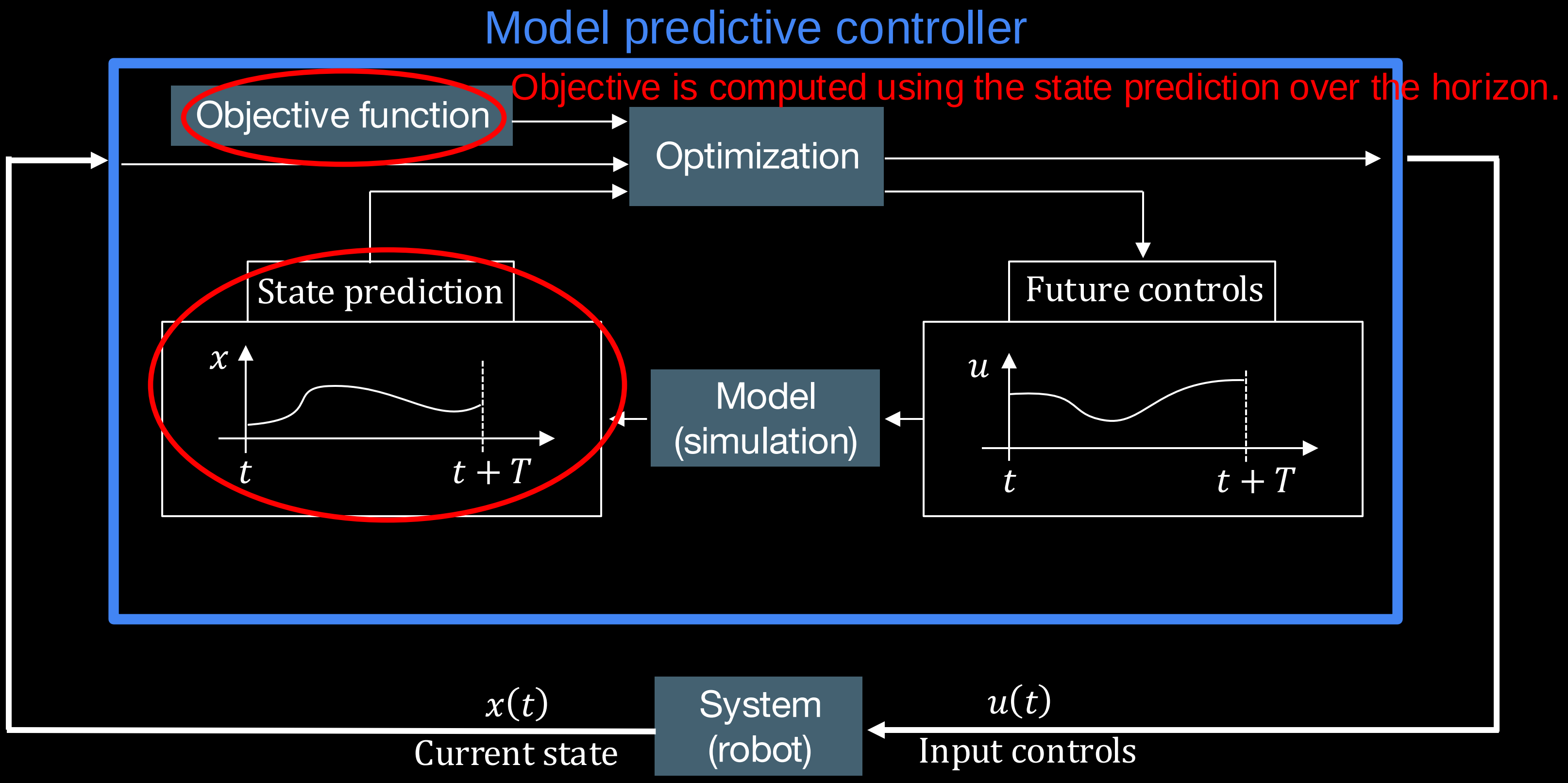

ETH zürich SoftRobotics Laboratory

# Model predictive control

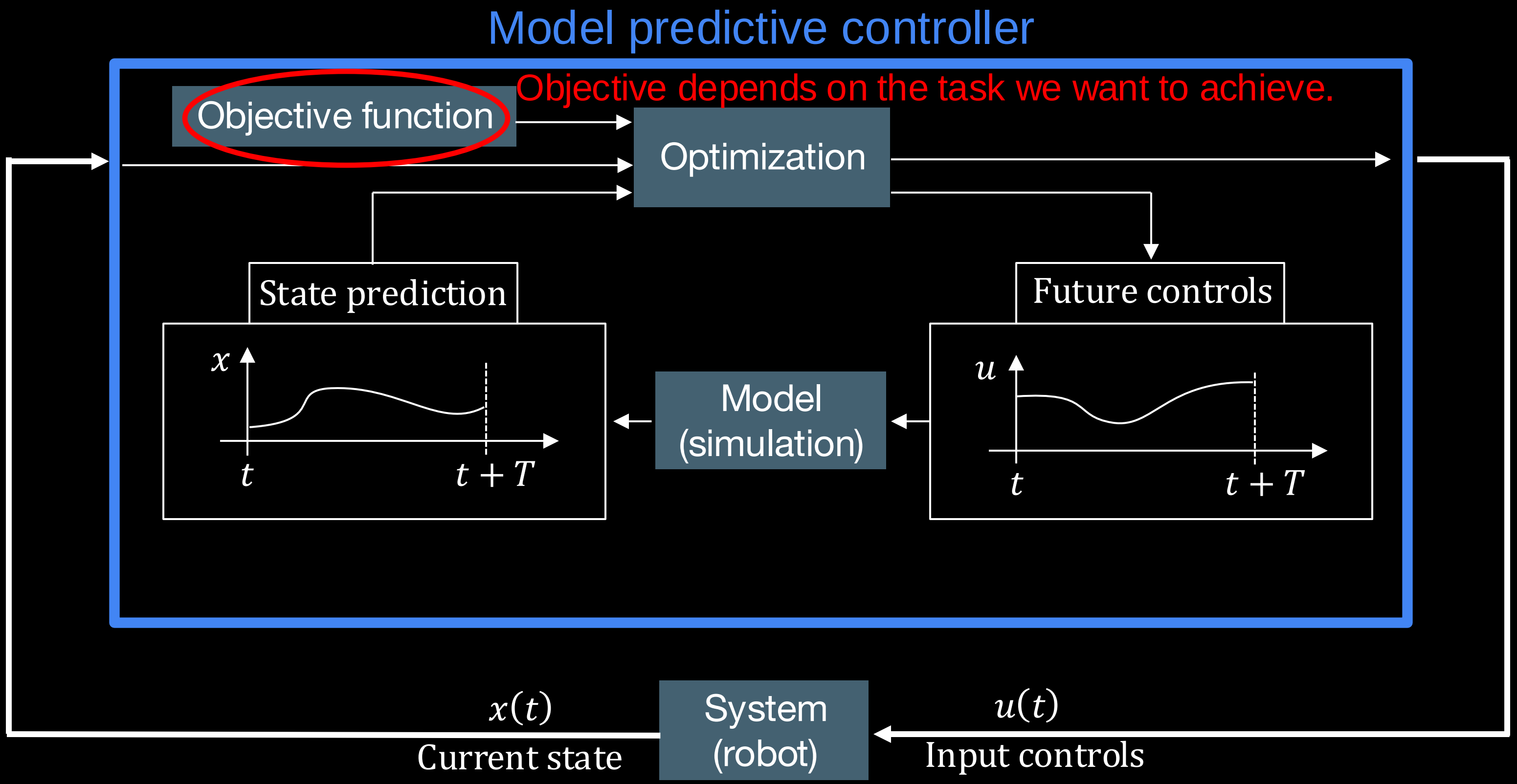



Execute only the "next step",
not the whole prediction horizon.

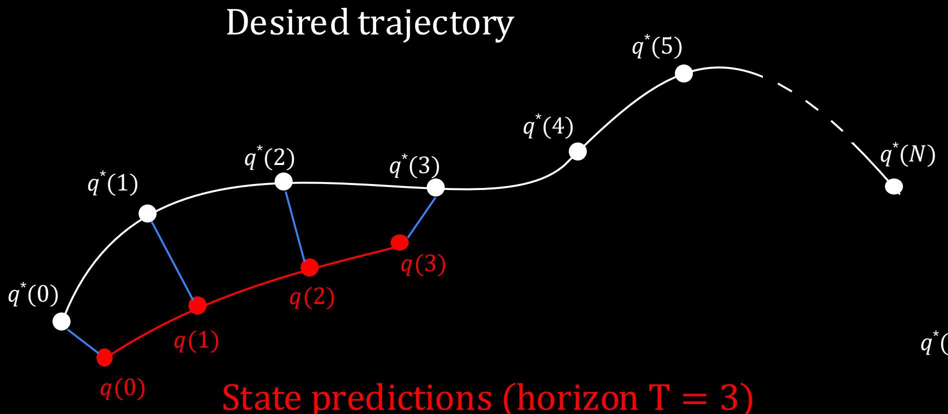
# Model predictive control

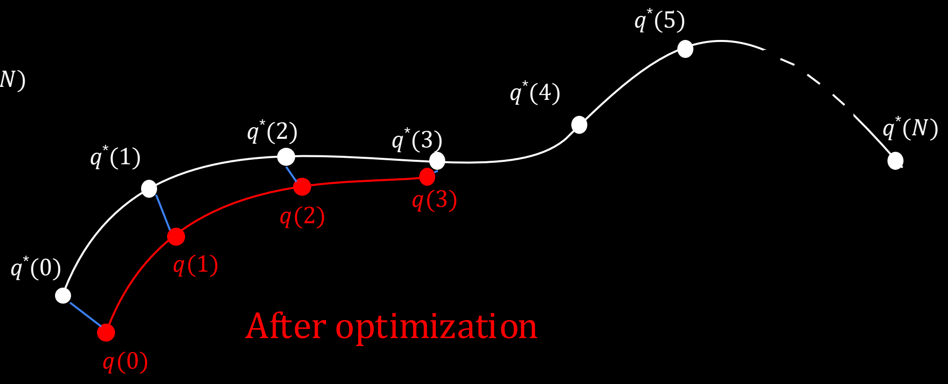


# Model predictive control



# Trajectory following with MPC



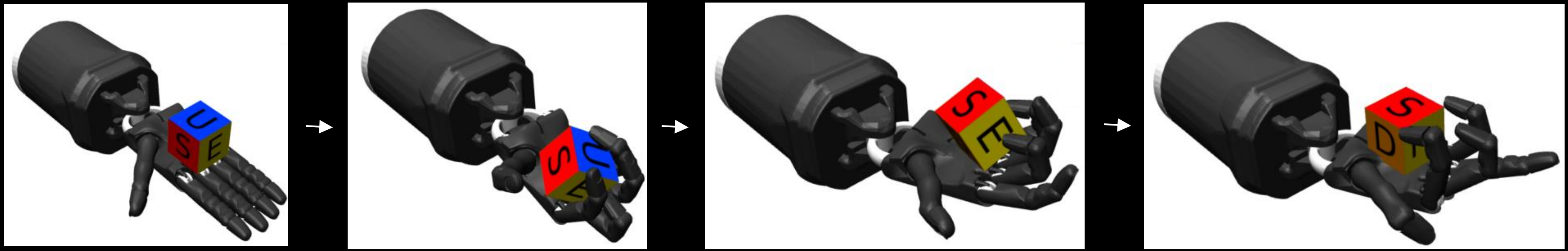

Objective $J = \sum_{t=0}^{T} c(t)$

where each step cost $c(t) = ||q^*(t) - q(t)||_2$

# Cube reorientation with MPC

Goal orientation: 

Howell et al. 2022, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo"

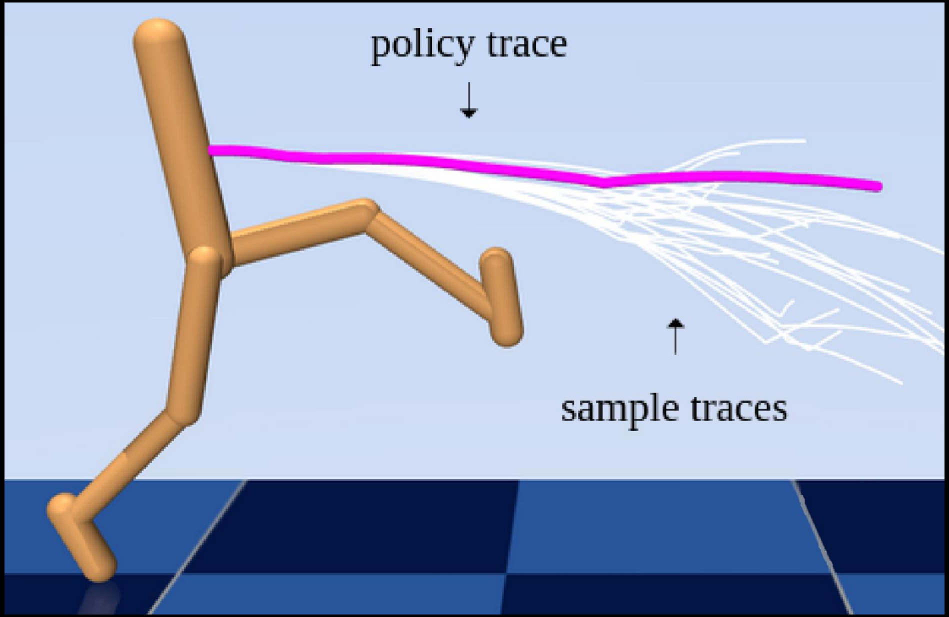


System state $x(t)$ includes robot state $q(t)$, but also the object state.

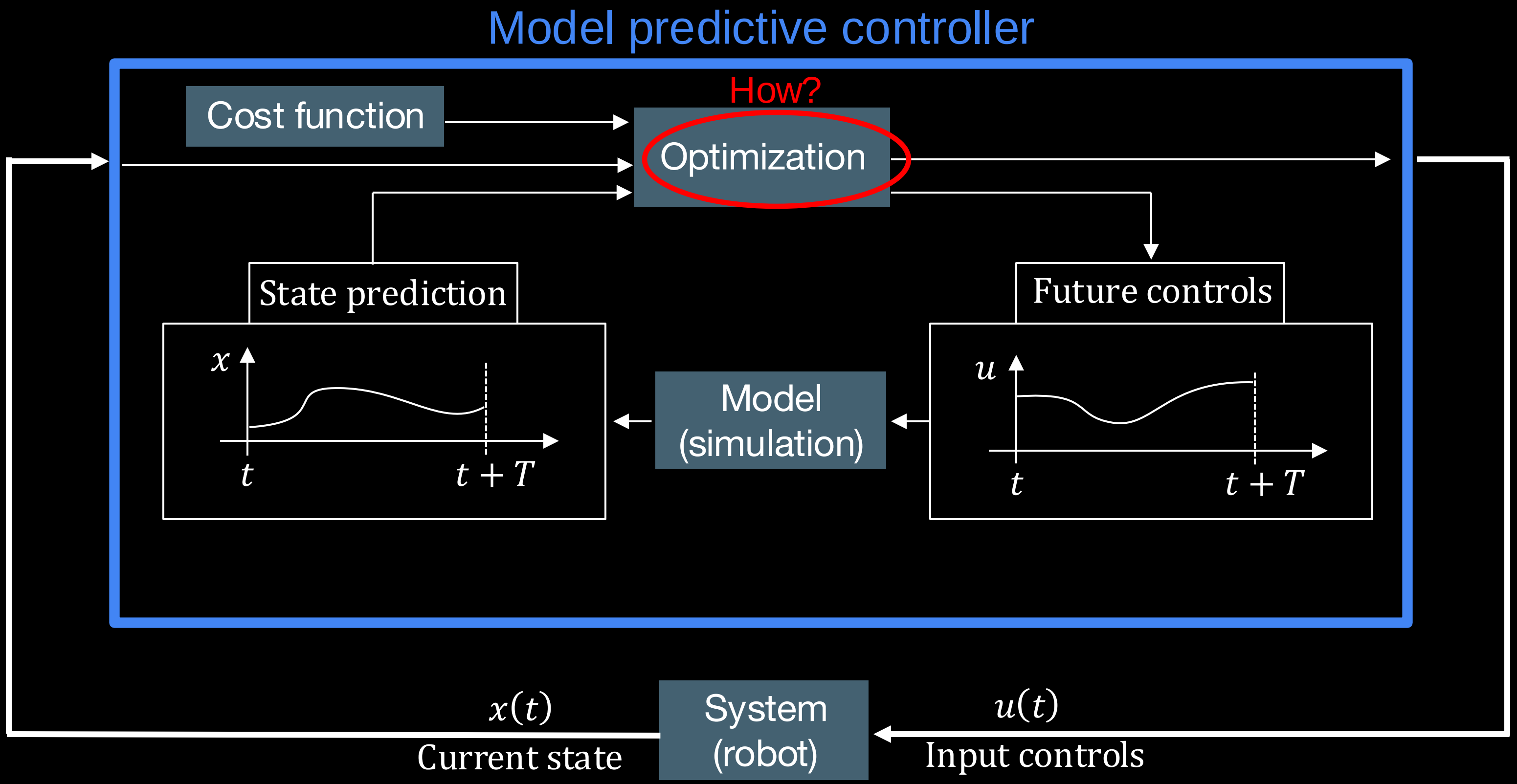
Objective $J = \sum_{t=0}^{T} c(t)$

where $c(t) = ||cube\ orientation\ (t) - goal\ orientation||_2 + ||cube\ position\ (t) - palm\ center||_2$

# Mujoco Predictive Control (MJPC) Demo



# Model predictive control



# Trajectory optimization

Given current state $x(0)$

$$\min_{u(0),u(1),\dots,u(T-1)} J$$

such that $x(t+1) = f(x(t), u(t))$

# Linear Quadratic Regulator (LQR)

Given current state $x(0)$

Quadratic

$$\min_{u(0),u(1),\ldots,u(T-1)} J$$

Linear

such that $x(t+1) = f(x(t), u(t))$

An optimal feedback law exists.

# Linear Quadratic Regulator (LQR)

Given current state $x(0)$

Quadratic

$$\min_{u(0),u(1),\ldots,u(T-1)} J$$

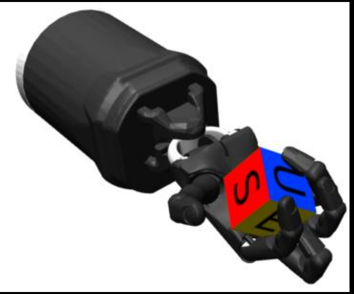
Linear

such that $x(t+1) = f(x(t), u(t))$

An optimal feedback law exists.

But for robotics, dynamics is rarely linear.

Highly non-linear!



# Trajectory optimizers in MJPC

- Derivative-based methods

**iLQR**:

Requires $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial u}, \frac{\partial c}{\partial x}, \frac{\partial c}{\partial u}, \frac{\partial^2 c}{\partial x^2}, \frac{\partial^2 c}{\partial u^2}, \frac{\partial^2 c}{\partial x \partial u}$

**Gradient descent**:

Requires $\frac{\partial J}{\partial u}$

Derivative computation is expensive!

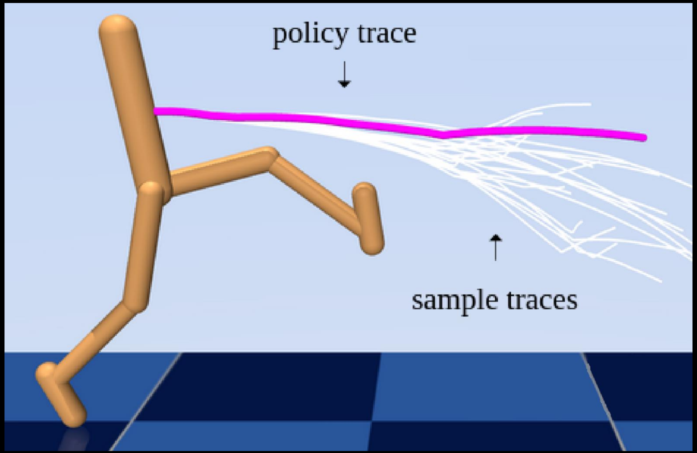
# Trajectory optimizers in MJPC

- A derivative-free method

**Predictive Sampling Algorithm:**

**Step 1:** Rollout all $N$ noisy trajectories

Step **2:** Pick the best one




- Performs surprisingly well!
- Parallelizable!

# Feedback Control vs MPC

**Feedback control**

**MPC**

# Feedback Control vs MPC

**Feedback control**

- Computationally cheap.

**MPC**

- Expensive.

# Feedback Control vs MPC

**Feedback control**

- Computationally cheap.

- Reacts to immediate residual.

**MPC**

- Expensive.

- Longer horizon. But still myopic after horizon $T$.

# Feedback Control vs MPC

**Feedback control**

- Computationally cheap.
- Reacts to immediate residual.
- Doesn't require a model.

**MPC**

- Expensive.
- Longer horizon. But still myopic after horizon $T$.
- Requires a computational model.
  - Sim2Real gap.

# Feedback Control vs MPC

**Feedback control**

- Computationally cheap.
- Reacts to immediate residual.
- Doesn't require a model.
- Limited to regulation/tracking.

**MPC**

- Expensive.
- Longer horizon. But still myopic after horizon $T$.
- Requires a computational model.
  - Sim2Real gap.
- Can encode higher-level tasks.

# MPC vs Reinforcement Learning

**MPC**

**Reinforcement Learning**

# MPC vs Reinforcement Learning

**MPC**

- No offline training.

**Reinforcement Learning**

- Offline training needed.

# MPC vs Reinforcement Learning

**MPC**

- No offline training.
- Requires a model.

**Reinforcement Learning**

- Offline training needed.
- Does not require a model.

# MPC vs Reinforcement Learning

**MPC**

- No offline training.
- Requires a model.
- Limited to our state representations.

**Reinforcement Learning**

- Offline training needed.
- Does not require a model.
- Can discover latent representations, and “intelligent” behavior.

# MPC vs Reinforcement Learning

**MPC**

- No offline training.
- Requires a model.
- Limited to our state representations.
- Slower during execution.

**Reinforcement Learning**

- Offline training needed.
- Does not require a model.
- Neural network representations and more "intelligent" behavior.
- Learns a policy, a direct mapping from state to action.

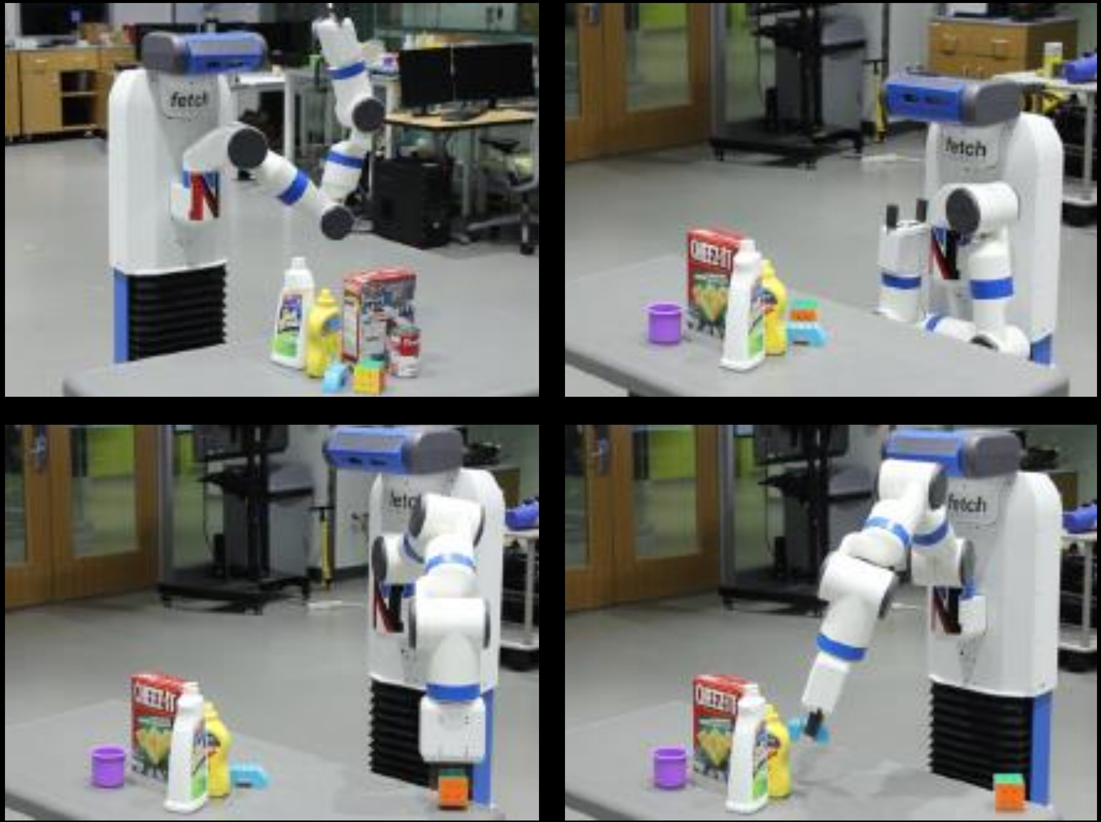
Maria College

# Part 3: Challenges

# What should you expect?

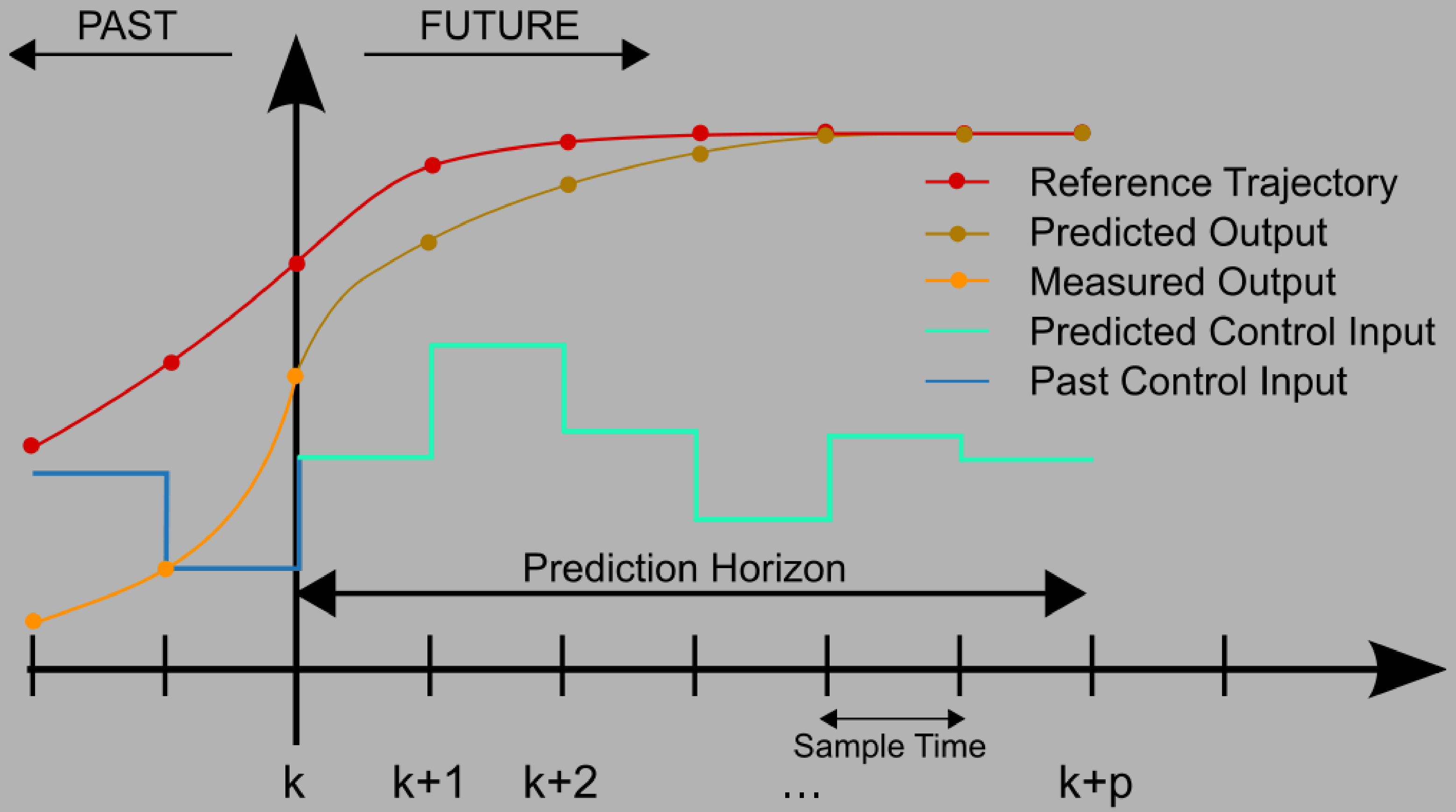
- Uncertainty and Partial Observability
- Long Horizon
- Under/Over actuation
- Sim-to-real gap
- Tendon strain + skin non-linearity
- Encoder's sensibility

# Uncertainty and Partial Observability



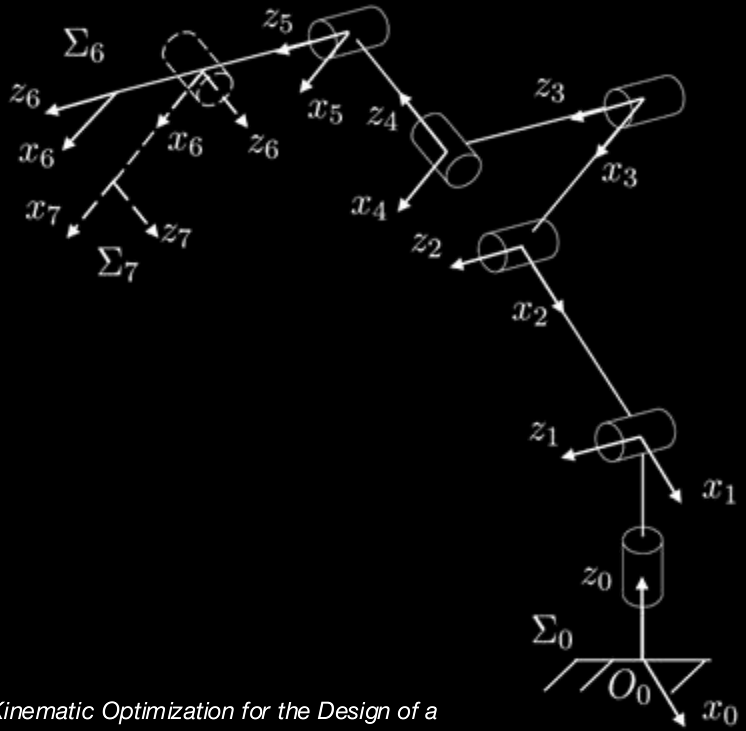

Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, Christopher Amato.
**Online Planning for Target Object Search in Clutter under Partial Observability.**
IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, May 2019.

# Long Horizon



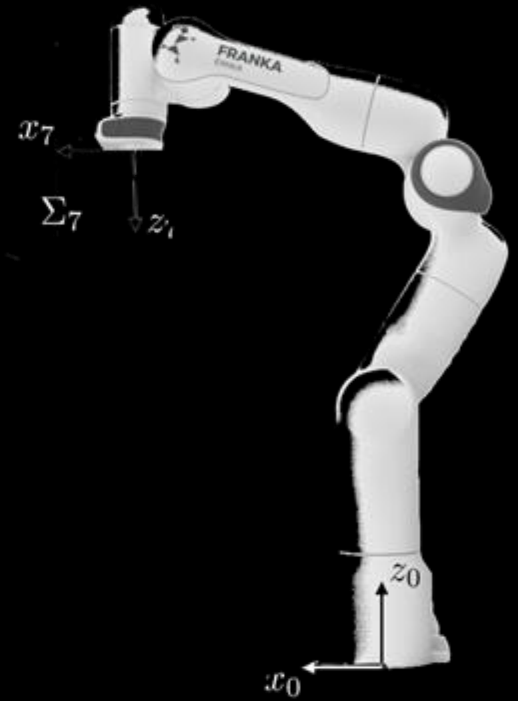

MPC and constrained systems , TU Eindhoven

# Underactuation and Overactuation




Filippeschi et al. *Kinematic Optimization for the Design of a Collaborative Robot End-Effector for Tele-Echography (2021)*

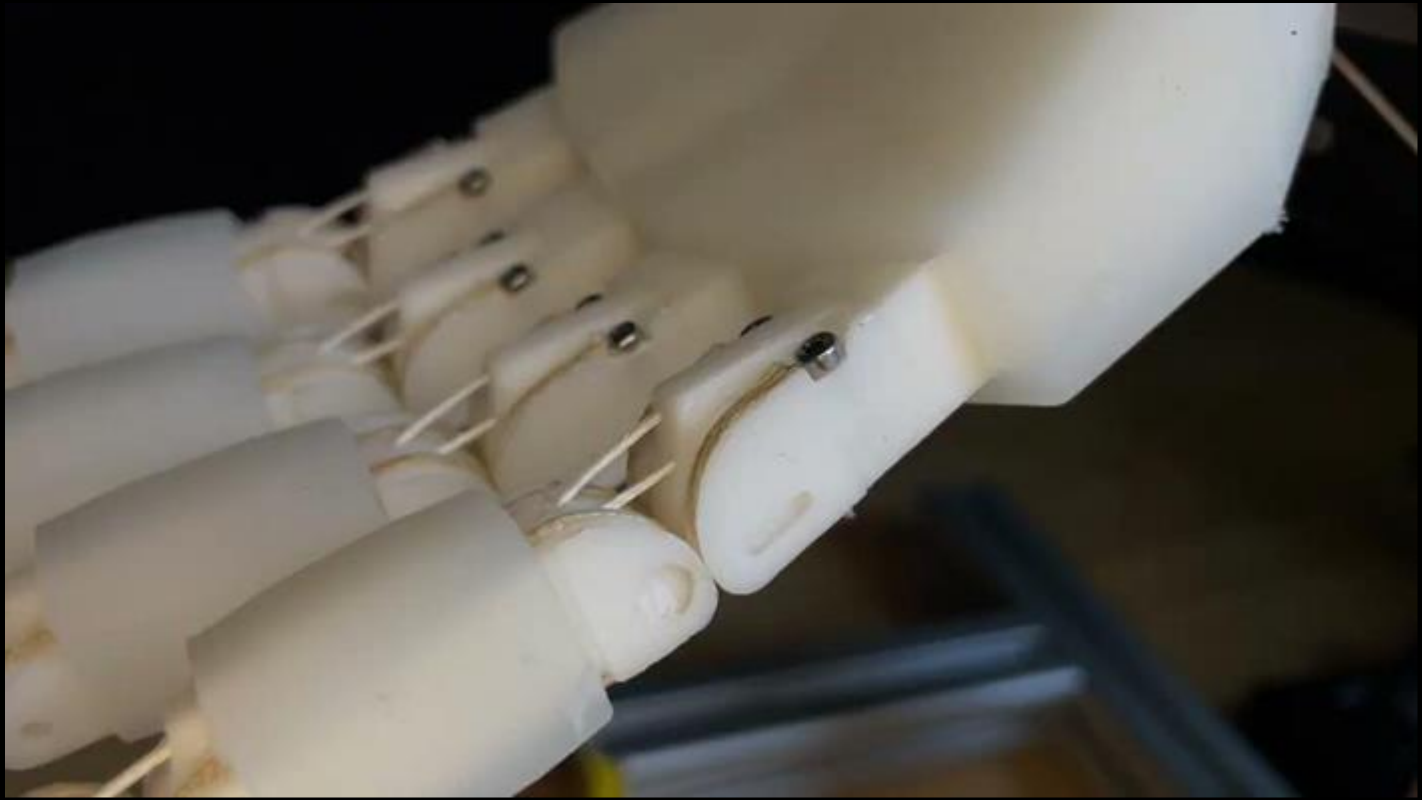
(a)



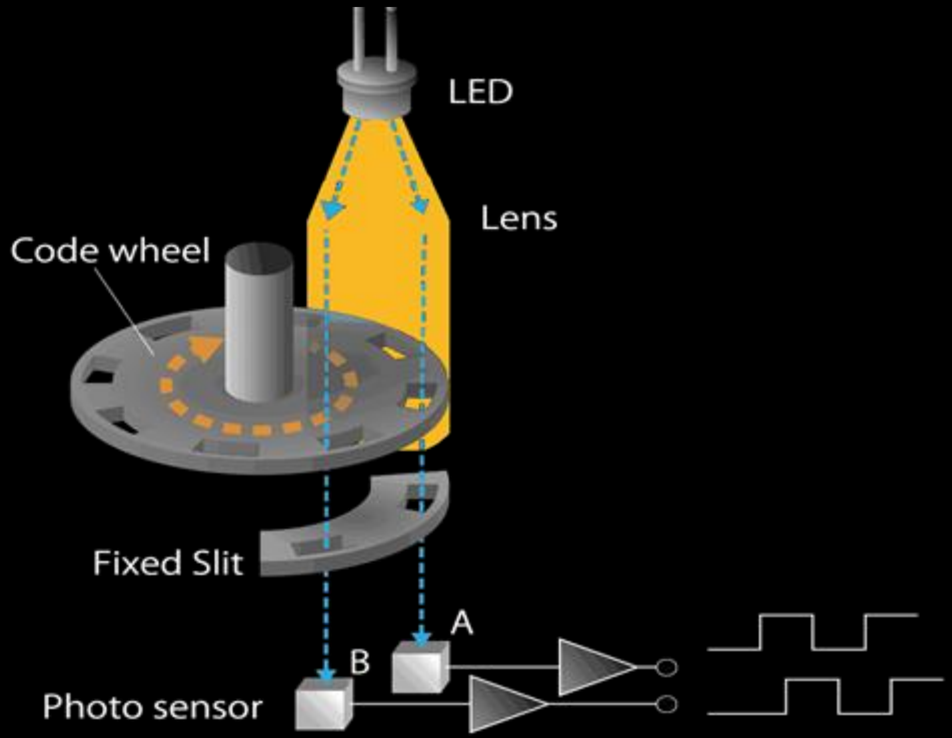

(b)

# Sim-to-real gap

# Tendon strain + skin non-linearity



# Encoder's sensibility

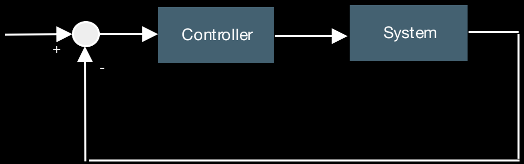



Asahi Kasei Microdevices

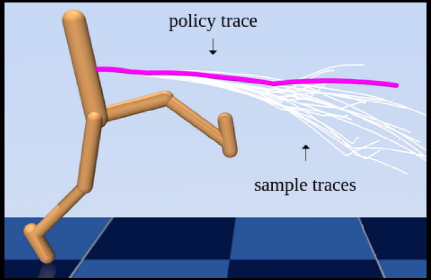
# Wrap up

## 1. Sensing

## 2a. Feedback Control




## 2b. Model Predictive Control




## 3. Challenges

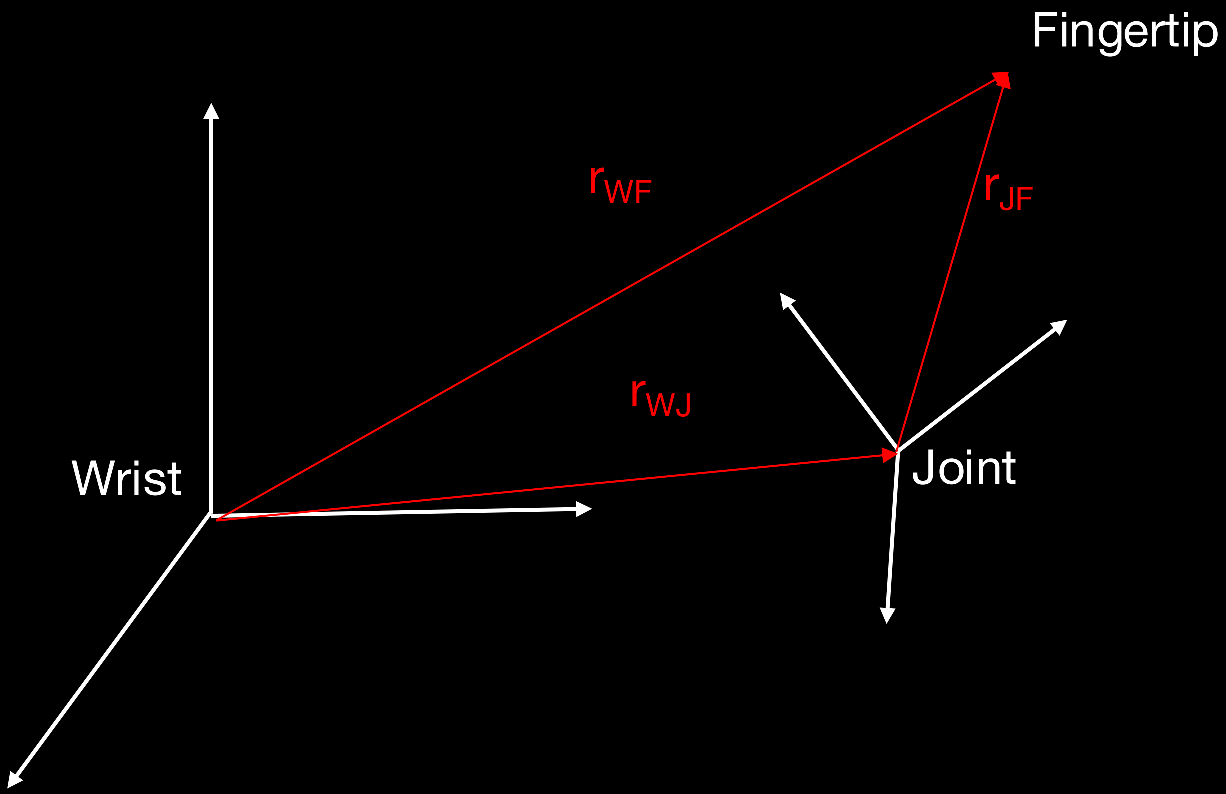

ETH zürich SoftRobotics Laboratory

# Backup Slides

# Sensing the pose: two methods

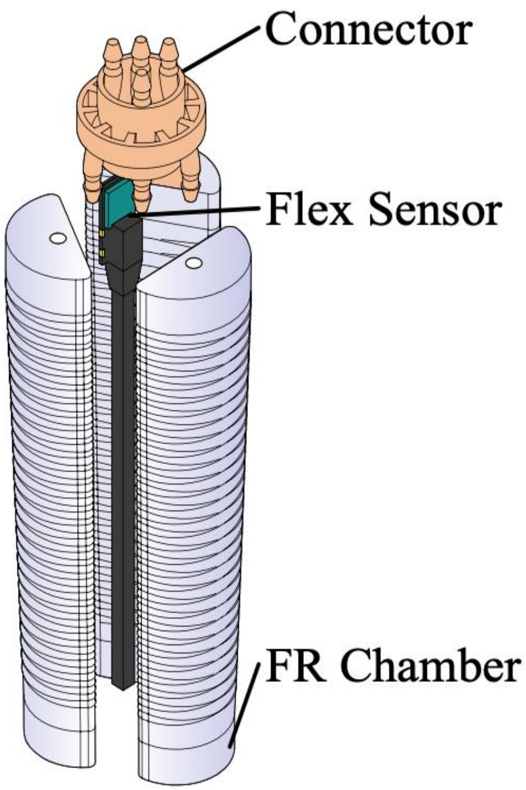
- Direct methods: Direct reference to the world reference frame
  - The sensors obtain the absolute value of the state we are measuring
- Indirect methods: Obtain a measurement with reference to a second frame
  - The sensors will estimate a relative measurement that can be transformed into an absolute measurement

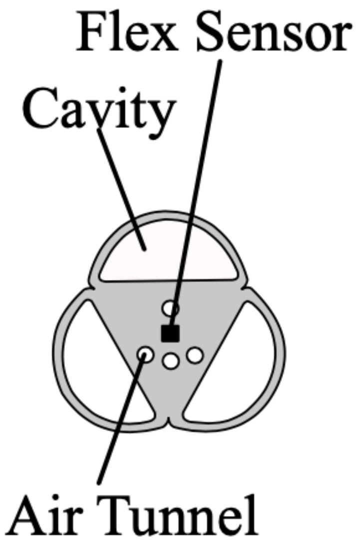
# Second solution

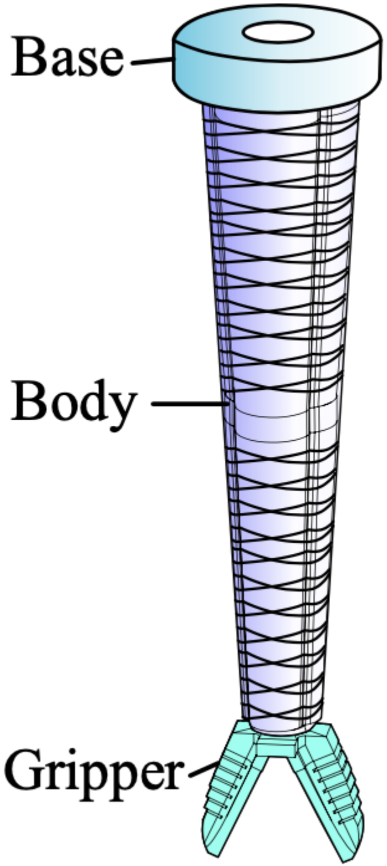


# Indirect methods

e.g., Built-in Flex Sensor

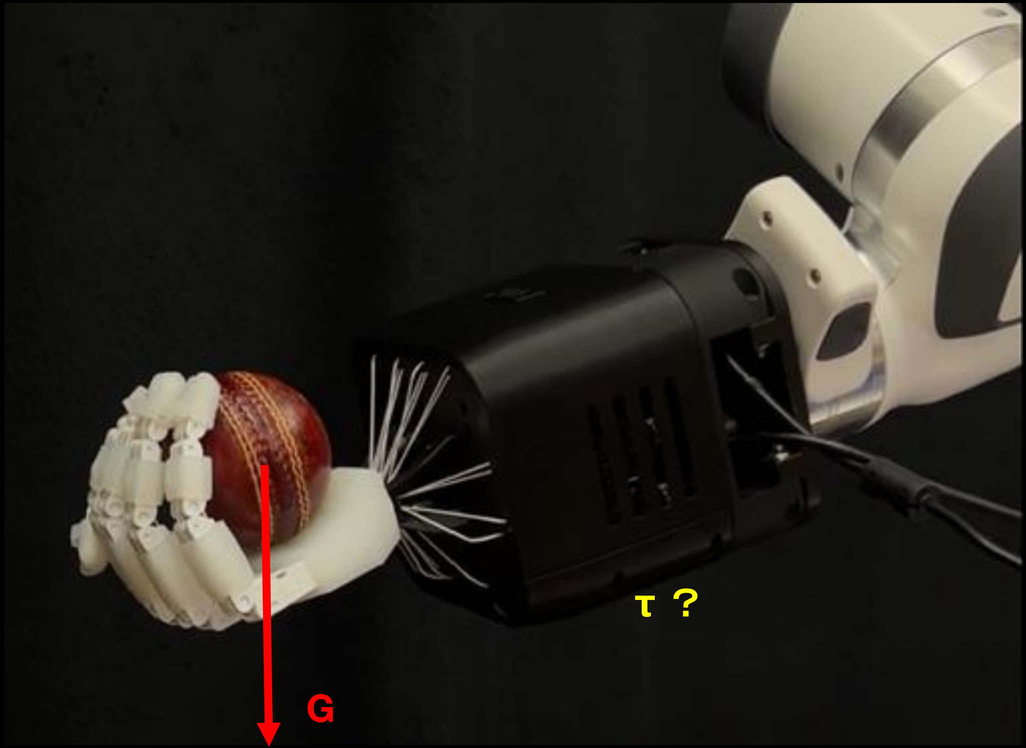



Toshimitsu, Y., Wong, K. W., Buchner, T., & Katzschmann, R. (2021, September). Sopra: Fabrication & dynamical modeling of a scalable soft continuum robotic arm with integrated proprioceptive sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 653-660). IEEE.

# Force control for tendon actuation




73



Toshimitsu et al. (2023) https://srl-ethz.github.io/get-ball-rolling/

# Force control for tendon actuation

Tendon
Lengths

$$p = g(l) = g\big(f(q)\big) = F(q)$$

Motor
Positions

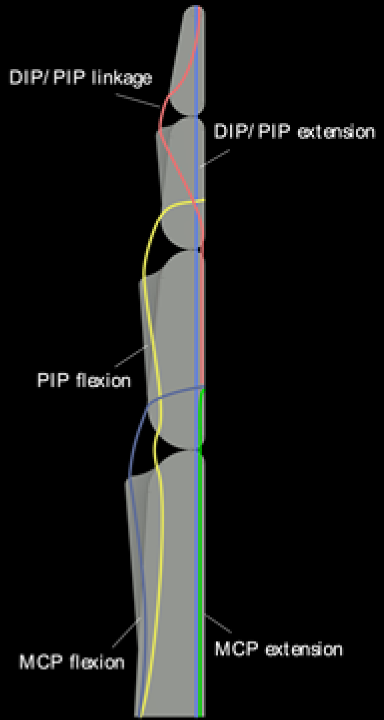
Joint
Angles



ETH zürich SoftRobotics Laboratory

# Force control for tendon actuation

$$J_m = \begin{bmatrix} \dfrac{\partial p_1}{\partial q_1} & \dfrac{\partial p_1}{\partial q_2} \\ \dfrac{\partial p_2}{\partial q_1} & \dfrac{\partial p_2}{\partial q_2} \end{bmatrix}$$




75

# Force control for tendon actuation

Velocity of the finger joints

$$\dot{p} = J_m \cdot \dot{q}$$

Velocity of the motors

$$\tau^T \cdot \dot{q} = T^T \cdot \dot{p}$$

Conservation of Power

$$\tau^T \cdot \dot{q} = T^T \cdot J_m \cdot \dot{q} \rightarrow \tau = {J_m}^T \cdot T$$

76

# Force control for tendon actuation

Previous slide: $\tau = {J_m}^T \cdot T$

$$\dot{X}_{fingertip} = J_{fingertip} \cdot \dot{q}$$

$$\tau^T \cdot \dot{q} = {F_{fingertip}}^T \cdot \dot{X}_{fingertip}$$

$$\tau^T \cdot \dot{q} = {F_{fingertip}}^T \cdot J_{fingertip} \cdot \dot{q}$$

$$\tau = {J_{fingertip}}^T \cdot F_{fingertip}$$

77

# Force control for tendon actuation

$$\tau = {J_m}^T \cdot T$$

$$\tau = {J_{fingertip}}^T \cdot F_{fingertip}$$

$$T = \left({J_m}^T\right)^{-1} \cdot {J_{fingertip}}^T \cdot F_{fingertip}$$

78

ETH zürich SoftRobotics Laboratory

# Outro no slide

# Useful links

https://link.springer.com/book/10.1007/978-3-319-54413-7
https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c
https://underactuated.csail.mit.edu/index.html
https://www.kalmanfilter.net/default.aspx