上海科技大学
ShanghaiTech University

CS289: **Mobile Manipulation Fall 2025**

Sören Schwertfeger

ShanghaiTech University

Final

- Thursday Dec 11!
  - 15:00 – 17:00 in 1D-104

- Content:
  - All lectures
    - Take a look at facts, algorithms, concepts

- You are allowed to bring **3** A4 sheets (so 6 pages) of info to the exams. You can write/ print anything on those sheets. On top of **every page** (so 6 times) there needs to be your **name (pinyin), student ID and ShanghaiTech email** address. We will check every cheat sheet before the exam and **confiscate** every sheet without name or with a name that is not yours.

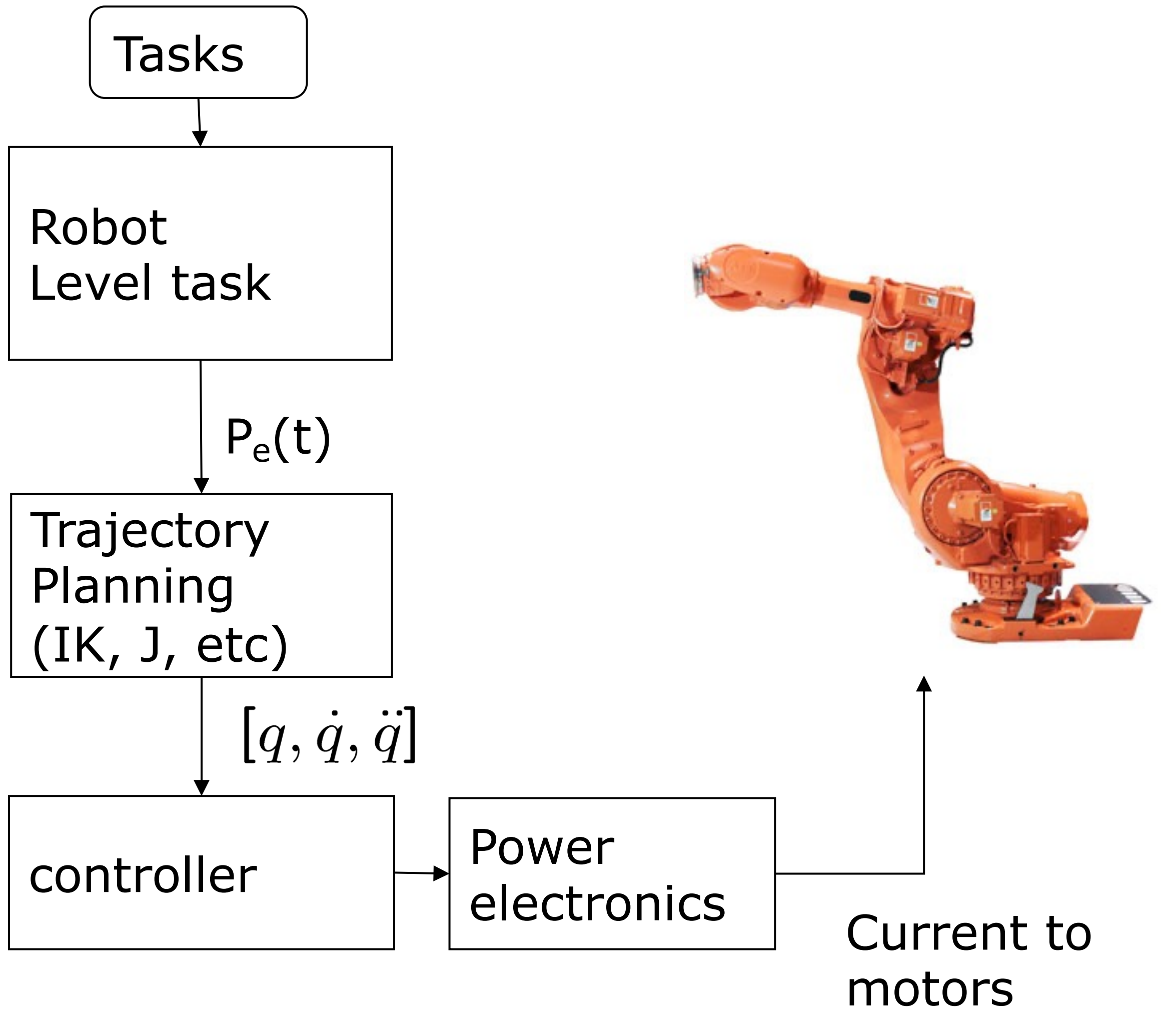
- No electronics/ calculator/ smartwatch allowed

HW 4

- HW 4 is out
- Due date: in 1 month!
  - Start early – quite a lot of work…
  - It’s about mission planning with:
    - Finite State Machine
    - Behaviour Tree
    - Symbolic Planning
    - LLM
  - Work in groups of 4 – one person for each approach
  - Work together WITHIN the group
  - Do NOT share work between groups (cheating)
  - Let the TA know your group members by Dec 1st via email!

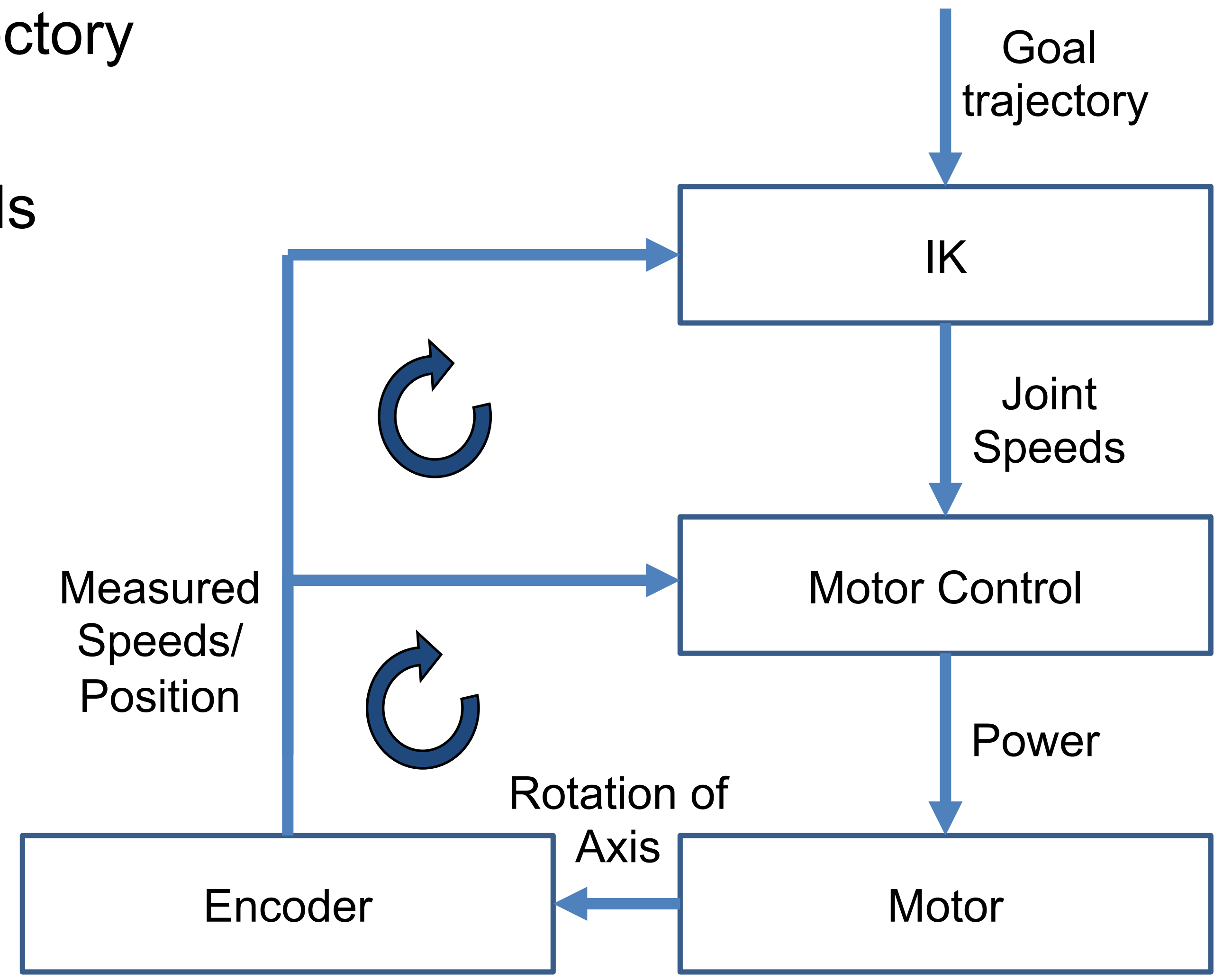
Motivation & Overview

- We covered Kinematics, Planning, Perception, etc.
- How to make the robot actually move?
- **Control** the robot motion
  - Dynamics (forces, mass, inertia etc.) =>
  - Kinematics of speeds: Jacobian
  - Control Introduction
  - PID
- Hardware
  - PWM
  - Motor Drivers
  - Motor
  - Gears

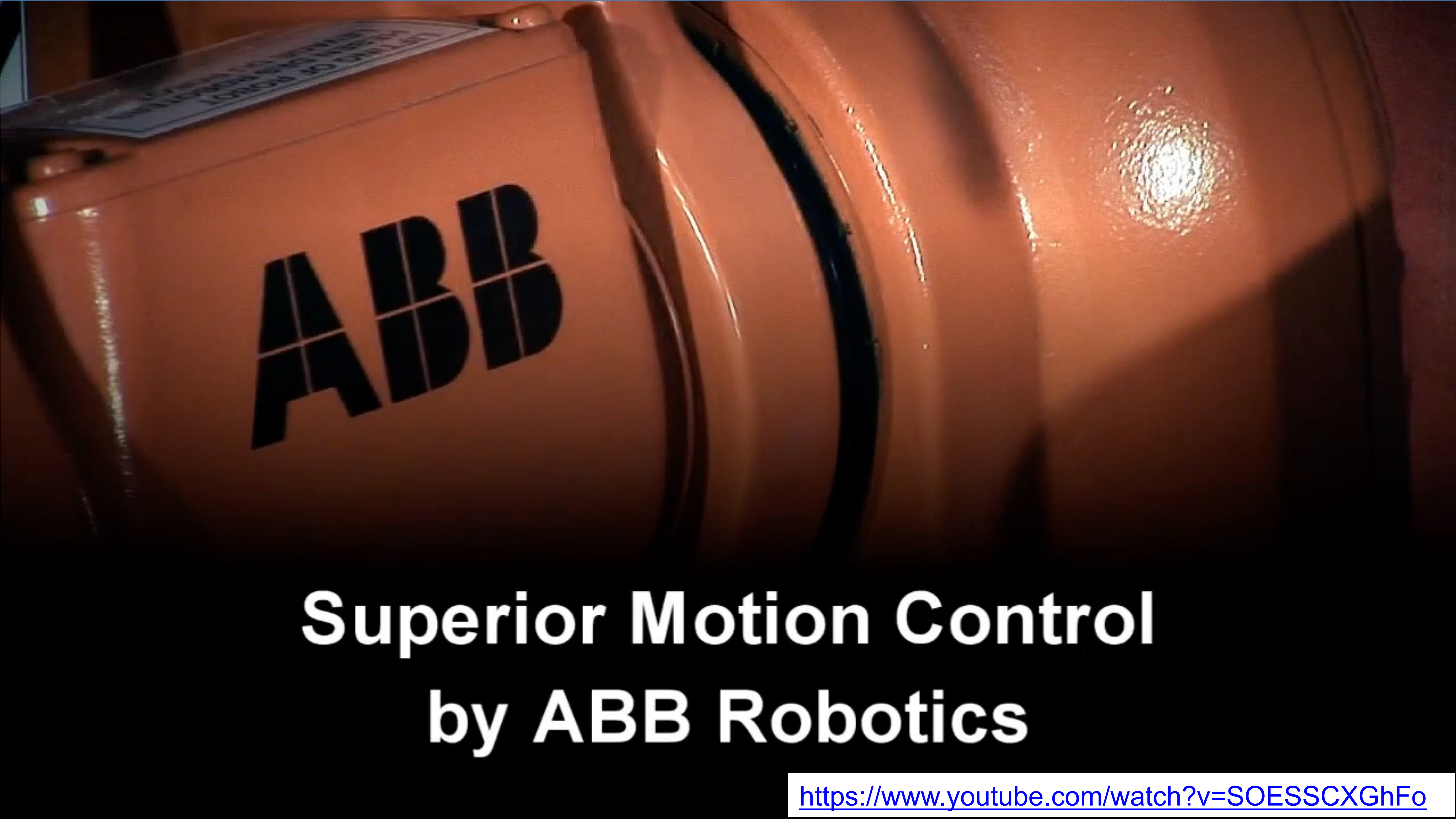


Control Hierarchy

- Assume we have a goal trajectory
- Calculate needed joint speeds using Kinematics =>
- Desired joint speeds
  - Typically not just one joint =>
  - Many motor controllers, motors, encoders
- Motor control loop
- Pose control loop



MULTIPLE MANIPULATORS

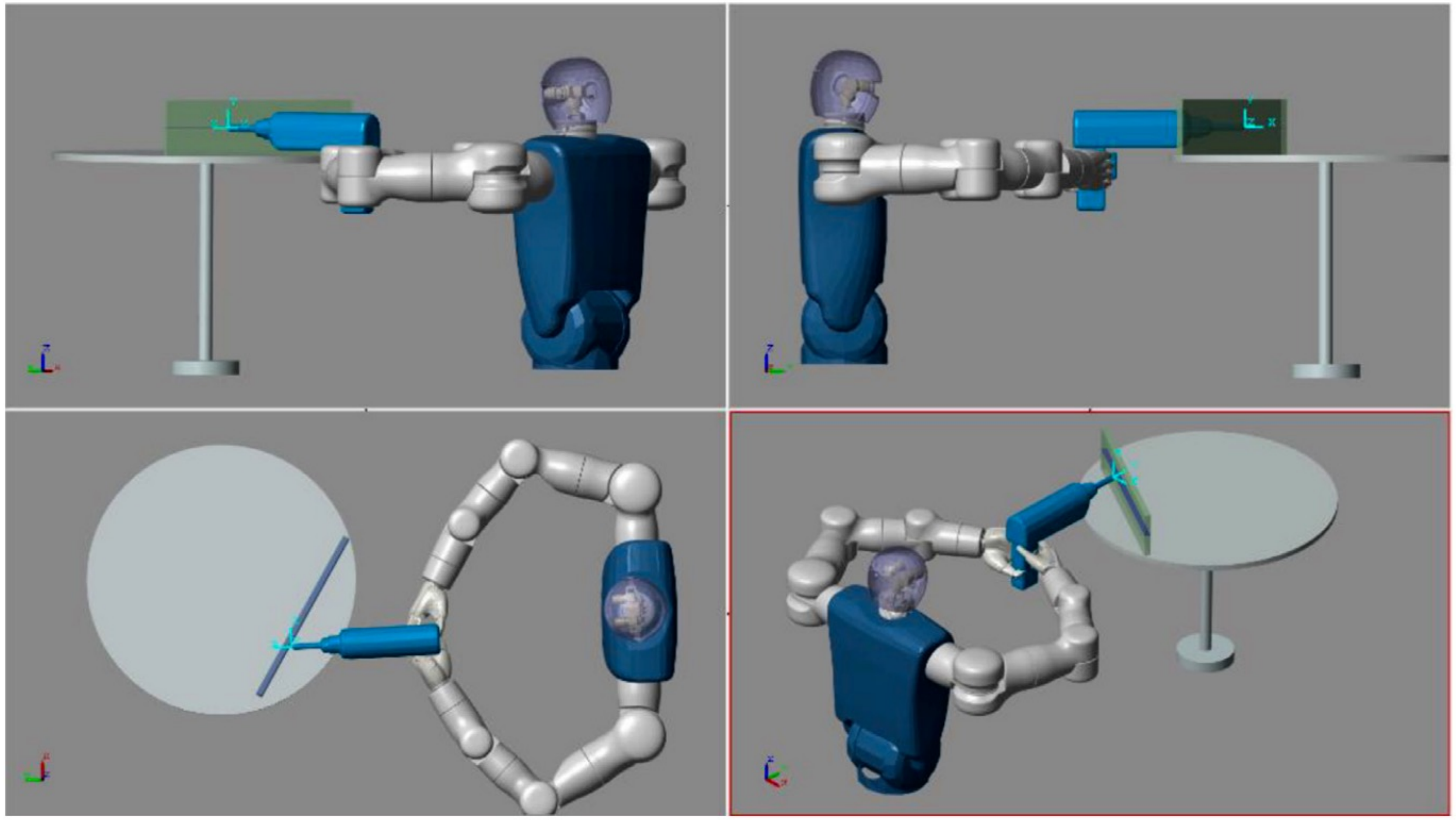


Superior Motion Control
by ABB Robotics

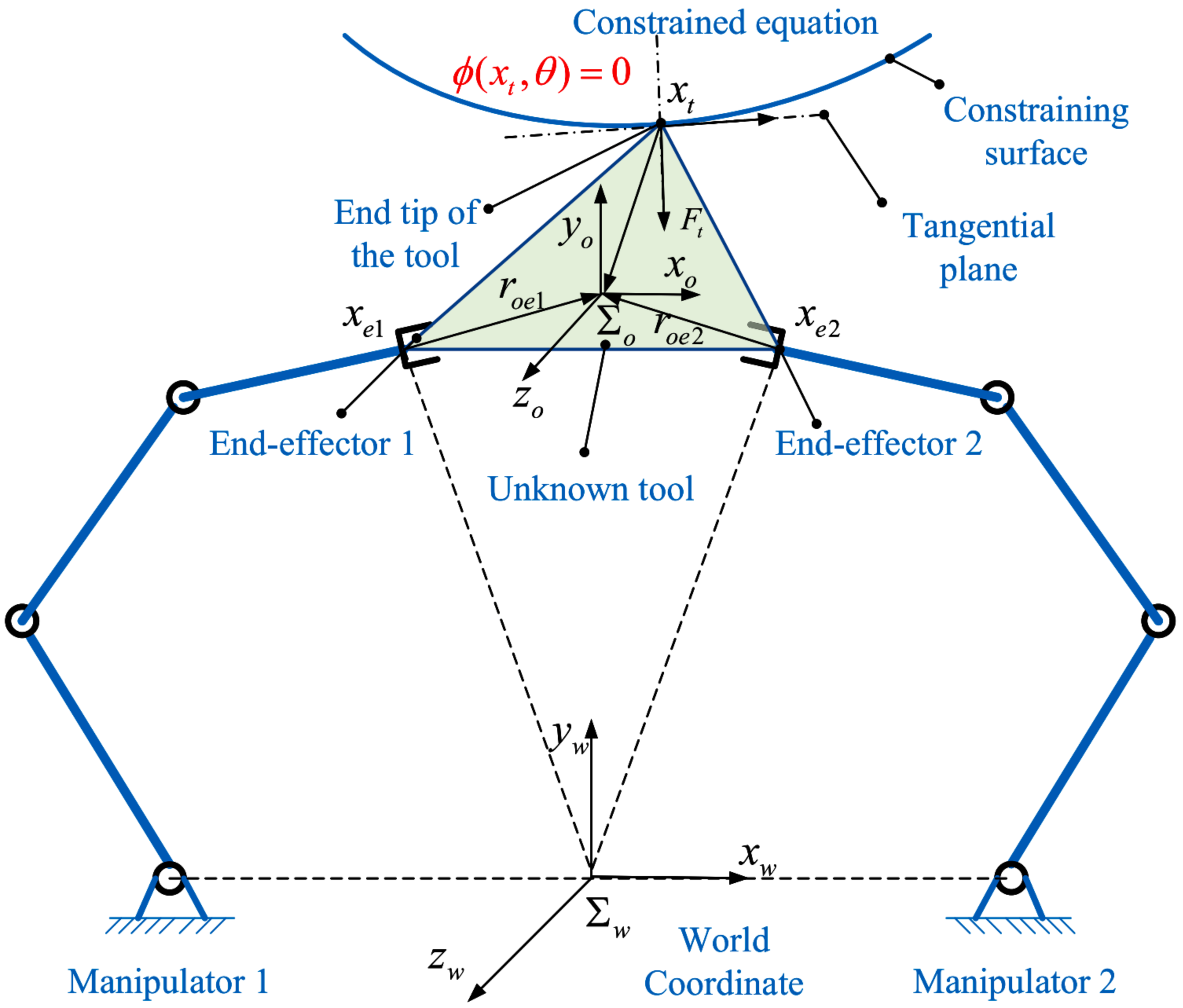
https://www.youtube.com/watch?v=SOESSCXGhFo

DUAL-ARM FORCE CONTROL

Adaptive hybrid position/force control of dual-arm cooperative manipulators with uncertain dynamics and closed-chain kinematics.



Ren, Y., Chen, Z., Liu, Y., Gu, Y., Jin, M., & Liu, H. (2017). Adaptive hybrid position/force control of dual-arm cooperative manipulators with uncertain dynamics and closed-chain kinematics. Journal of the Franklin Institute, 354(17), 7767-7793. https://doi.org/10.1016/j.jfranklin.2017.09.015


Constrained equation
$\phi(x_t, \theta) = 0$
x_t
Constraining surface
Tangential plane
End tip of the tool
y_o
F_t
x_o
r_{oe1}
x_{e1}
Σ_o
r_{oe2}
x_{e2}
z_o
End-effector 1
Unknown tool
End-effector 2
y_w
x_w
Σ_w
z_w
World Coordinate
Manipulator 1
Manipulator 2

Kinematics

$x_e = [x_{e1}^T \quad x_{e2}^T]^T \in R^{2m\times 1}$ Pose vector of two end-effectors

$\dot{x}_e = J_D \dot{q}_D$

$q_D = [q_1^T, q_2^T]^T \in R^{(n_1+n_2)\times 1}$ Joint angles vector

$J_D = \text{blockdiag}[J_1, J_2] \in R^{2m\times(n_1+n_2)}$ Jacobian Matrix

$\dot{x}_e = Y_k(q_D, \dot{q}_D)\theta_k$ $\quad \theta_k = [\theta_{k1}, \theta_{k2}, ..., \theta_{kj}]^T \in R^j$ Kinematic parameters, e.g. joint offsets & link lengths

$\dot{x}_e = J_o \dot{x}_o$ $\quad x_o \in R^p$ Object's center of mass $\quad J_o$ Grasp matrix

$\dot{x}_o = \mathcal{R}(x_t)\dot{x}_t$ Velocity of the tip of tool $\quad \mathcal{R}(x_t)$ Mapping matrix from the task space to object space

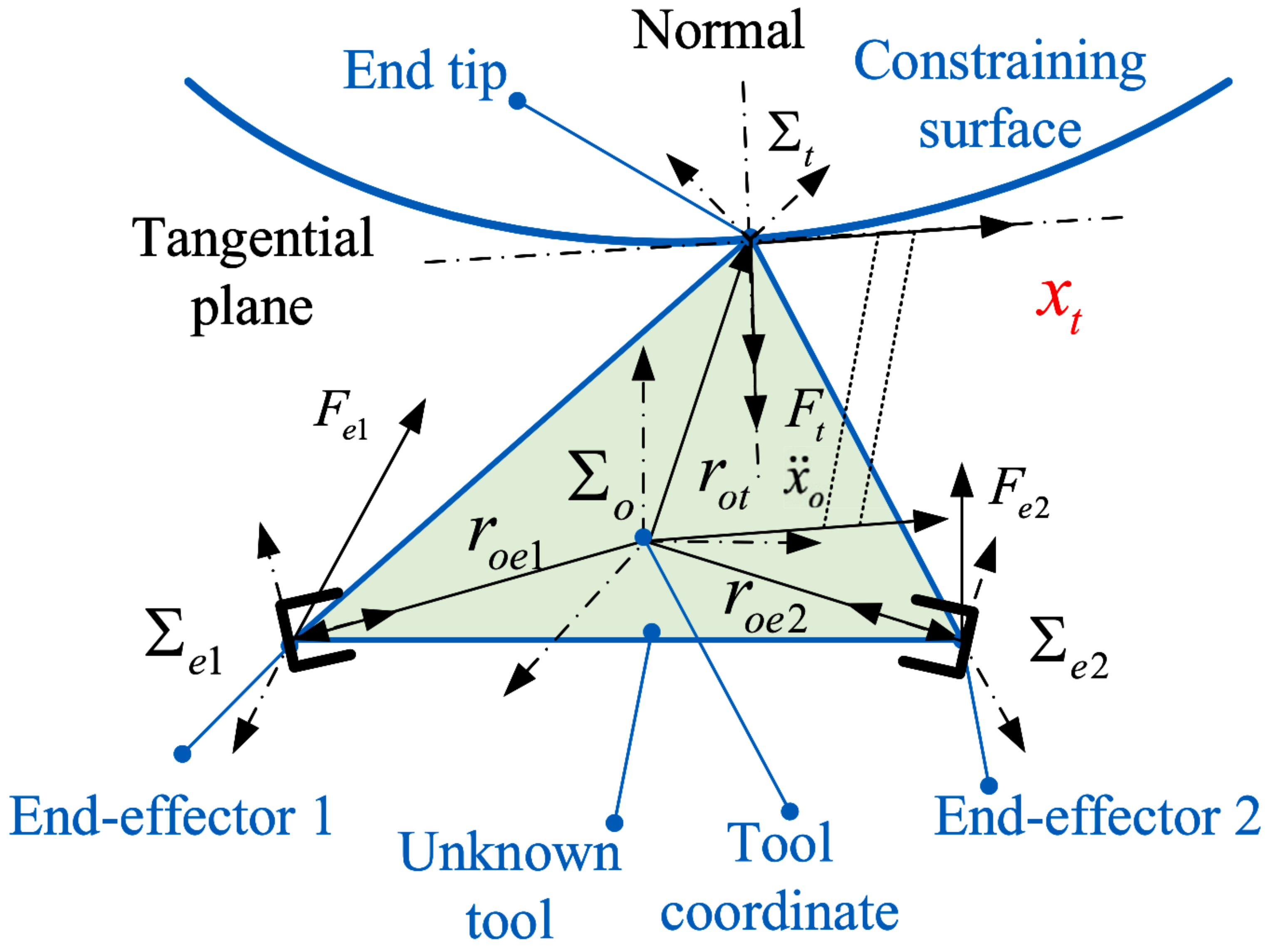


Fig. 2. Sketch of the forces acting on the unknown tool.

Dynamics

$\tau \in R^{(n_1+n_2)}$ Applied joint torques

$F_e = [F_{e1}^T \quad F_{e1}^T]^T \in R^{2m\times 1}$ Interacting forces on object

$$M_D(q_D)\ddot{q}_D + C_D(q_D, \dot{q}_D)\dot{q}_D + g_D(q_D) = \tau - J_D^T F_e$$

$M_D(q_D) = \text{blockdiag}[M_1(q_1), M_2(q_2)] \in R^{(n_1+n_2)\times(n_1+n_2)}, \; M_i(q_i) \in R^{n_i \times n_i}$ Inertial matrix

$C_D\dot{q}_D = [(C_1\dot{q}_1)^T, (C_2\dot{q}_2)^T]^T \in R^{(n_1+n_2)\times 1}$ Coriolis & Centrifugal forces

$g_D = [g_1^T \quad g_2^T]^T \in R^{(n_1+n_2)\times 1}$ Gravitational forces

Hybrid position/ force control: velocities & accelerations

Reference joint velocities

$$\dot{q}_r = \hat{J}_D^{\dagger}\left(J_o\mathcal{R}\dot{x}_{tr} + \kappa N_{J_o^{\dagger}}\mathcal{F}^T\lambda_{FI}\right) + \left(I - \hat{J}_D^{\dagger}\hat{J}_D\right)\psi$$

$$= \hat{J}_D^{\dagger}J_o\mathcal{R}\Big[\underbrace{\dot{x}_{td} + \alpha(x_{td} - x_t)}_{\text{Tip position term}} - \underbrace{\beta R_t J_t^T\Delta\lambda_{Ft}}_{\text{Contact force term}}\Big] + \underbrace{\kappa\hat{J}_D^{\dagger}N_{J_o^{\dagger}}\mathcal{F}^T\Delta\lambda_{FI}}_{\text{Internal force term}} + \left(I - \hat{J}_D^{\dagger}\hat{J}_D\right)\psi$$

Reference joint accelerations

$$\ddot{q}_r = \left(I - \hat{J}_D^{\dagger}\hat{J}_D\right)\dot{\psi} - \frac{d\left(\hat{J}_D^{\dagger}\hat{J}_D\right)}{dt}\psi + \frac{d\left(\hat{J}_D^{\dagger}J_o\mathcal{R}\right)}{dt}\left[\dot{x}_{td} + \alpha(x_{td} - x_t) - \beta R_t J_t^T\Delta\lambda_{Ft}\right]$$

$$+\hat{J}_D^{\dagger}J_o\mathcal{R}\left[\ddot{x}_{td} + \alpha(\dot{x}_{td} - \dot{x}_t) - \beta\left(R_t J_t^T\Delta\lambda_t + \frac{d(R_t J_t^T)}{dt}\Delta\lambda_{Ft}\right)\right]$$

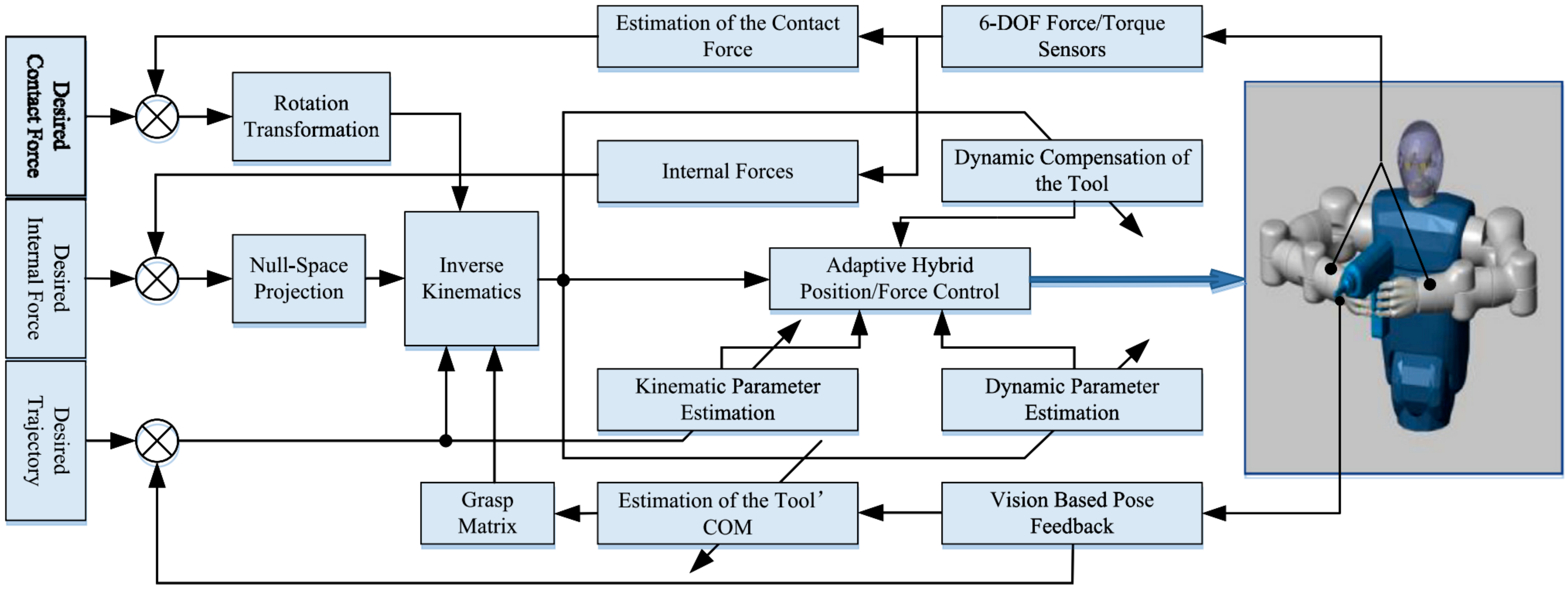
$$+\kappa\frac{d\left(\hat{J}_D^{\dagger}N_{J_o^{\dagger}}\mathcal{F}^T\right)}{dt}\Delta\lambda_{FI} + \kappa\hat{J}_D^{\dagger}N_{J_o^{\dagger}}\mathcal{F}^T\Delta\lambda_I$$

Hybrid position/ force control: Adaptive torque controller

$$\tau = K_p s + \underbrace{Y_{mdr}\hat{\theta}_{mdr} + Y_{Jod}\hat{\theta}_{Jod} - Y_{ft}\hat{\theta}_{ft}\lambda_t + Y_{fI}\hat{\theta}_{fI}\lambda_I}_{\text{Dynamic compensation}} + \underbrace{\hat{J}_D^T \mathcal{F}^T (\Delta\lambda_I + \gamma\,\Delta\lambda_{FI})}_{\text{Internal force control}}$$

$$+\left(\mathcal{R}^- J_o^\dagger \hat{J}_D\right)^T \left\{ \underbrace{K\left(\Delta\dot{\hat{x}}_t + \alpha\tilde{x}_t\right)}_{\text{Tip position control}} - \underbrace{R_t J_t^T (\Delta\lambda_t + \gamma\,\Delta\lambda_{Ft})}_{\text{Contact force control}} \right\}$$

Hybrid position/ force control: Block Diagram


Desired Contact Force
Desired Internal Force
Desired Trajectory
Rotation Transformation
Null-Space Projection
Inverse Kinematics
Estimation of the Contact Force
Internal Forces
6-DOF Force/Torque Sensors
Dynamic Compensation of the Tool
Adaptive Hybrid Position/Force Control
Kinematic Parameter Estimation
Dynamic Parameter Estimation
Grasp Matrix
Estimation of the Tool' COM
Vision Based Pose Feedback

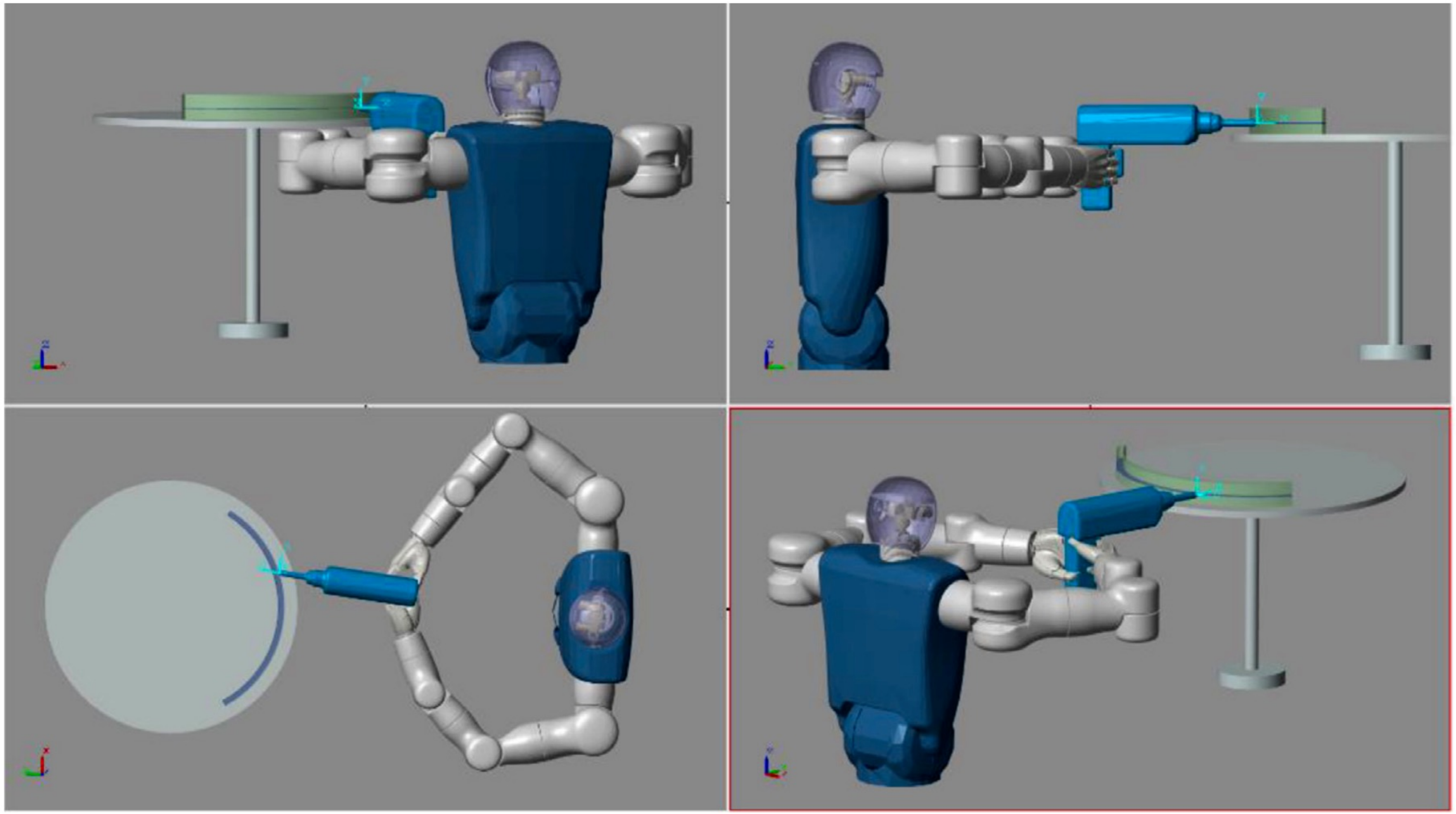
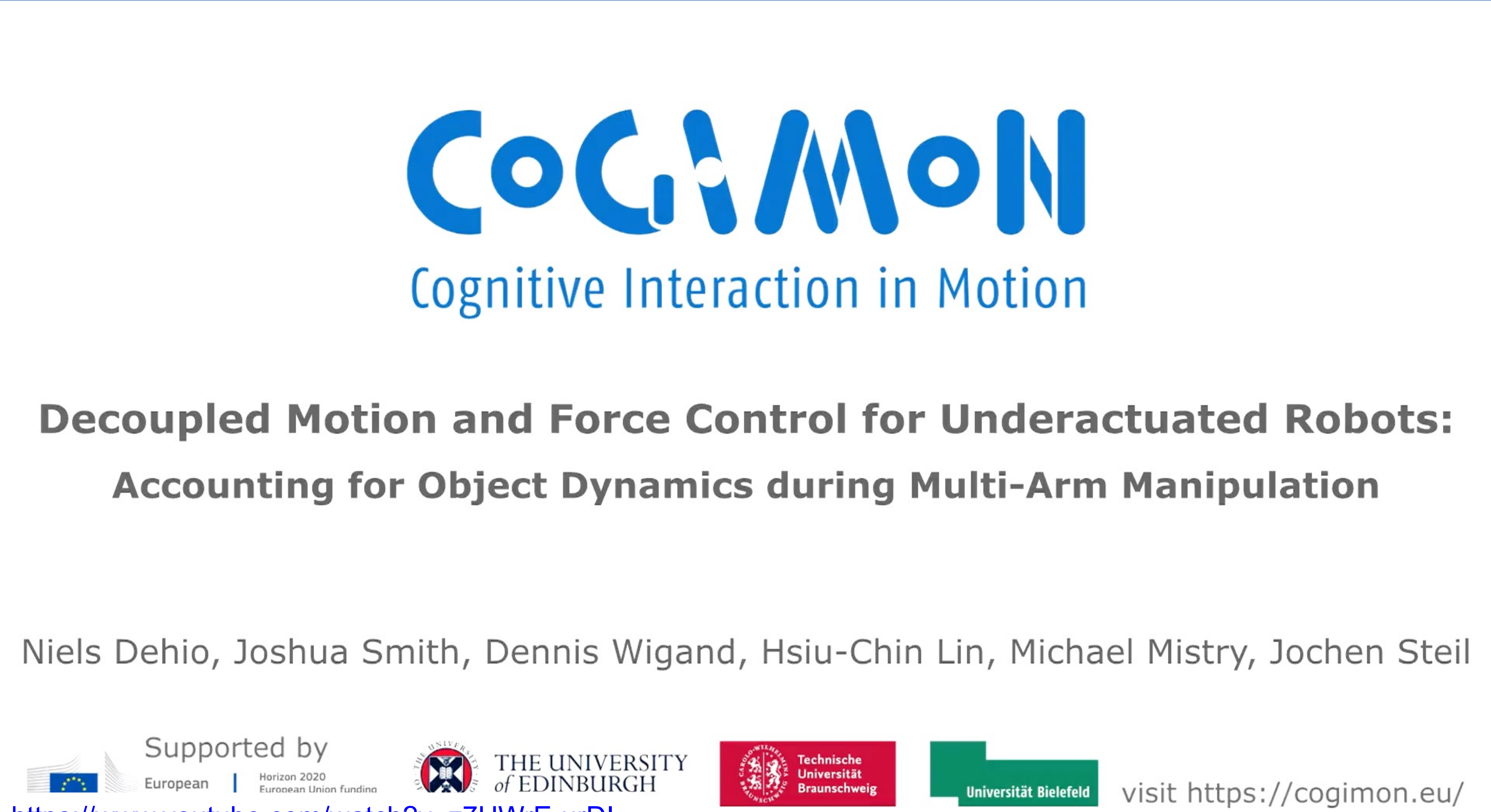


Fig. 8. Snapshot of the curved contact simulation.



COGIMON

Cognitive Interaction in Motion

Decoupled Motion and Force Control for Underactuated Robots:

Accounting for Object Dynamics during Multi-Arm Manipulation

Niels Dehio, Joshua Smith, Dennis Wigand, Hsiu-Chin Lin, Michael Mistry, Jochen Steil

Supported by

European | Horizon 2020 European Union funding

THE UNIVERSITY of EDINBURGH

Technische Universität Braunschweig

Universität Bielefeld

visit https://cogimon.eu/

https://www.youtube.com/watch?v=z7HWrE-urDI

DISTRIBUTED COOPERATION

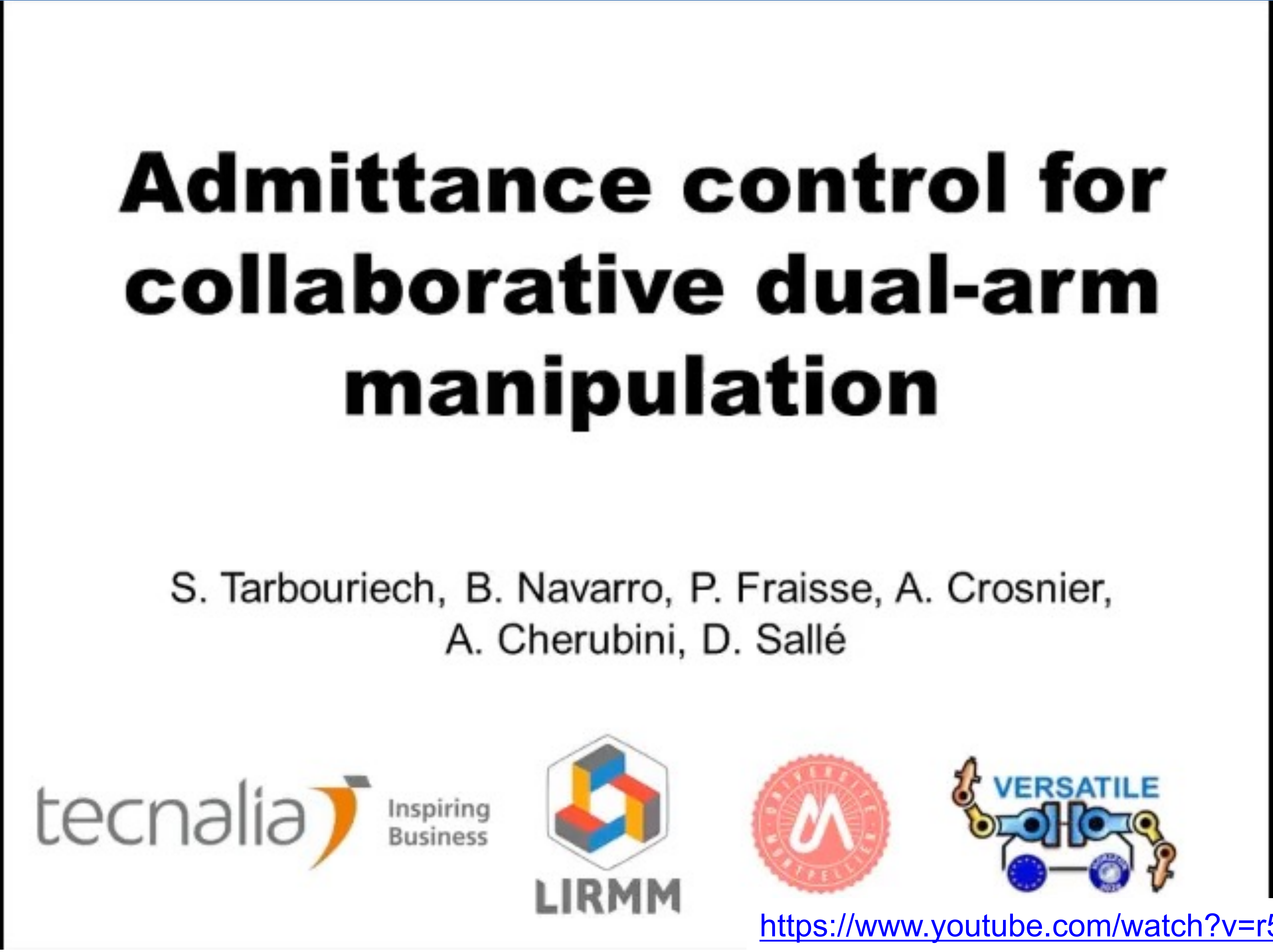
Distributed Multi-Robot Cooperative Manipulation with Obstacle Avoidance and Internal Performance Optimisation

(Part 1 - Coordination OFF vs. ON)

Yanhao He
Institute of Control Systems
University of Kaiserslautern

https://www.youtube.com/watch?v=8PN7bQok8_w

COLLABORATIVE CONTROL



Admittance control for collaborative dual-arm manipulation

S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, D. Sallé

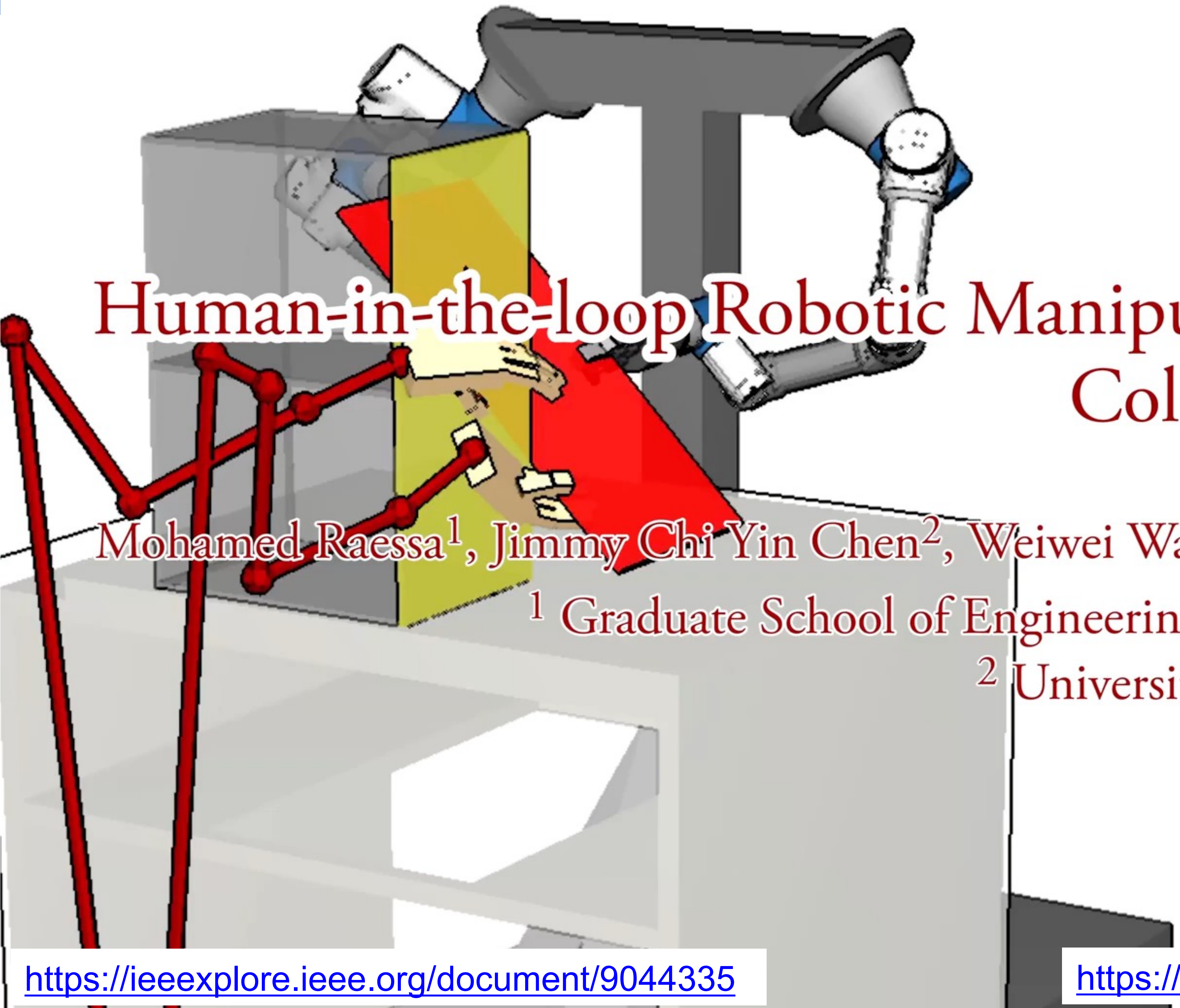
tecnalia Inspiring Business

LIRMM

VERSATILE

https://www.youtube.com/watch?v=r5FeUCIPwfw

COLLABORATION: TASK PLANNING



Human-in-the-loop Robotic Manipulation Planning for Collaborative Assembly

Mohamed Raessa[1], Jimmy Chi Yin Chen[2], Weiwei Wan*[13], and Kensuke Harada[13]

[1] Graduate School of Engineering Science, Osaka University

[2] University of California, Santa Cruz

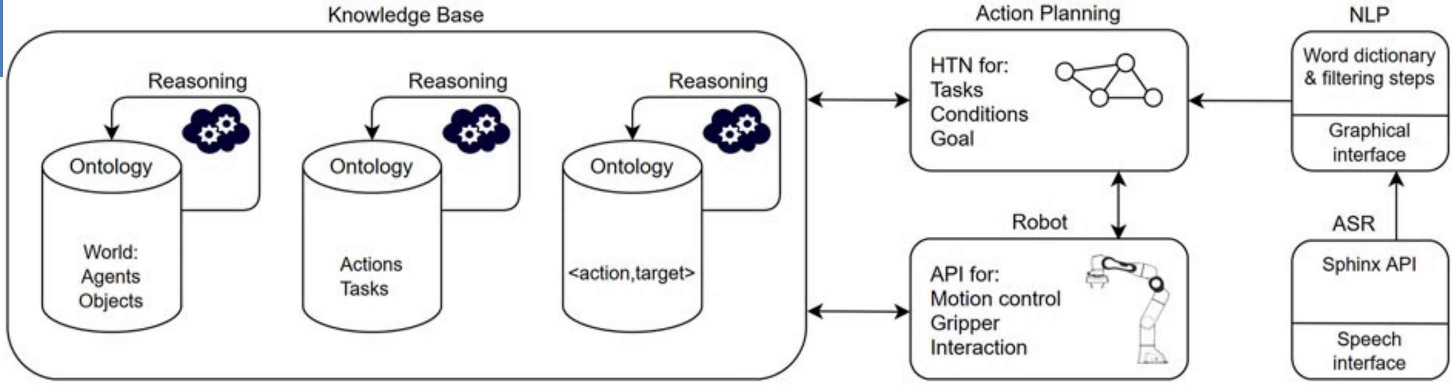
[3] National Inst. of AIST

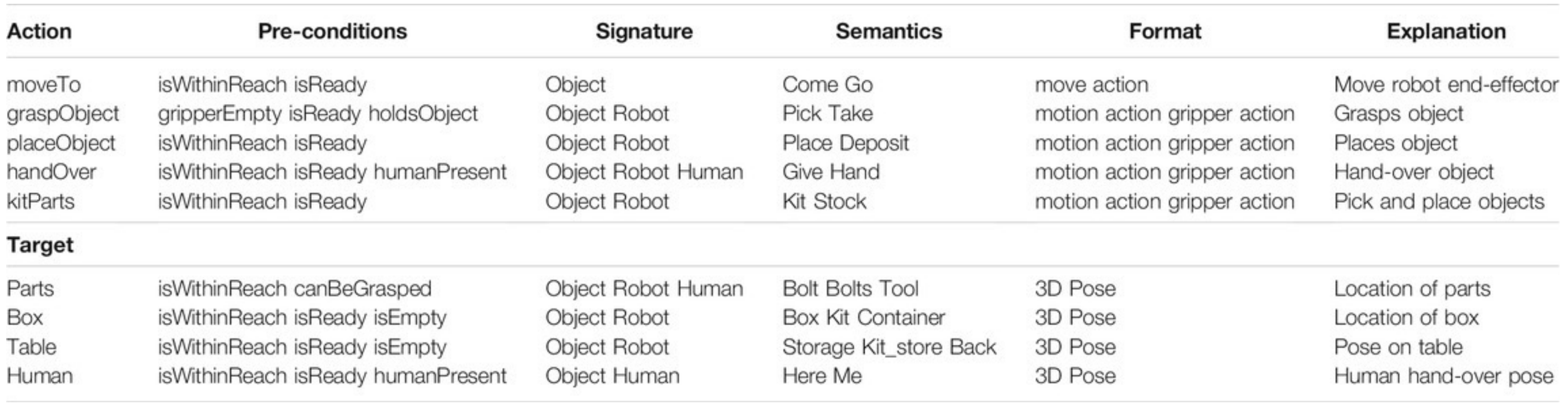
https://ieeexplore.ieee.org/document/9044335

https://www.youtube.com/watch?v=t_-89-N_RgM

Coordinating Shared Tasks in Human-Robot Collaboration by Commands

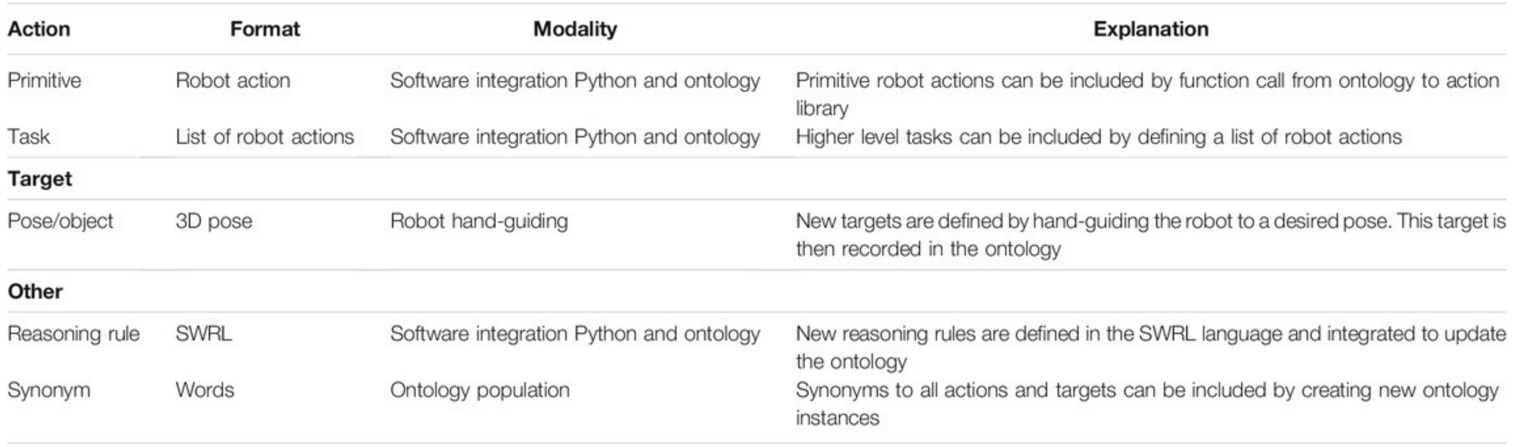
- Knowledge-based system architecture: supports reasoning, planning and knowledge integration
- Shared task coordination by human commands, either by a graphical interface or by speech
- Hierarchical Task Networks (HTN): another symbolic AI planning approach – can often be translated to PDDL





Action	Pre-conditions	Signature	Semantics	Format	Explanation
moveTo	isWithinReach isReady	Object	Come Go	move action	Move robot end-effector
graspObject	gripperEmpty isReady holdsObject	Object Robot	Pick Take	motion action gripper action	Grasps object
placeObject	isWithinReach isReady	Object Robot	Place Deposit	motion action gripper action	Places object
handOver	isWithinReach isReady humanPresent	Object Robot Human	Give Hand	motion action gripper action	Hand-over object
kitParts	isWithinReach isReady	Object Robot	Kit Stock	motion action gripper action	Pick and place objects
Target					
Parts	isWithinReach canBeGrasped	Object Robot Human	Bolt Bolts Tool	3D Pose	Location of parts
Box	isWithinReach isReady isEmpty	Object Robot	Box Kit Container	3D Pose	Location of box
Table	isWithinReach isReady isEmpty	Object Robot	Storage Kit_store Back	3D Pose	Pose on table
Human	isWithinReach isReady humanPresent	Object Human	Here Me	3D Pose	Human hand-over pose

Integrating new knowledge =>

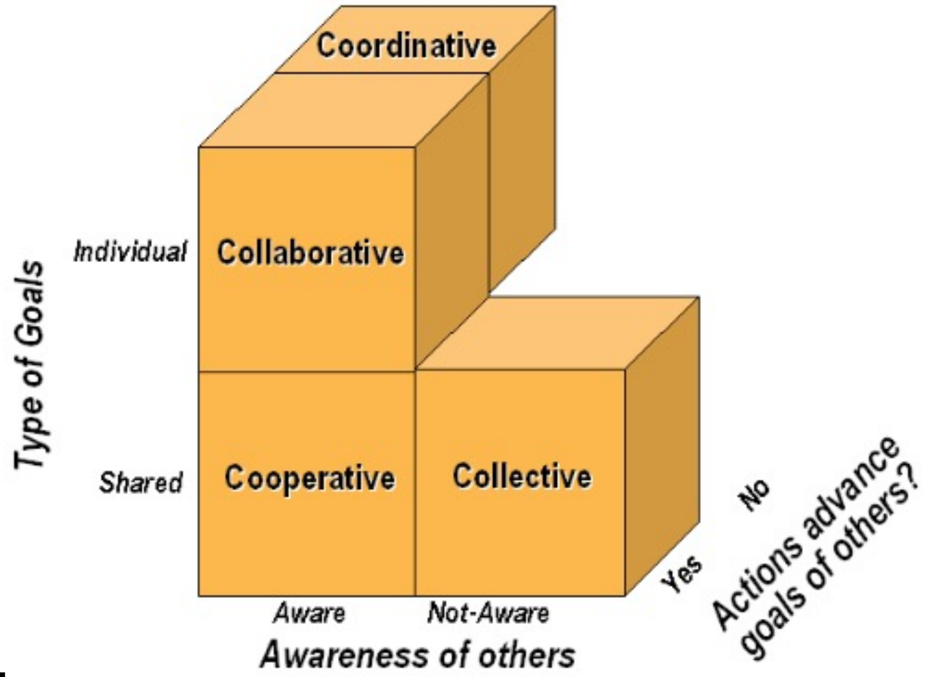


Action	Format	Modality	Explanation
Primitive	Robot action	Software integration Python and ontology	Primitive robot actions can be included by function call from ontology to action library
Task	List of robot actions	Software integration Python and ontology	Higher level tasks can be included by defining a list of robot actions
Target			
Pose/object	3D pose	Robot hand-guiding	New targets are defined by hand-guiding the robot to a desired pose. This target is then recorded in the ontology
Other			
Reasoning rule	SWRL	Software integration Python and ontology	New reasoning rules are defined in the SWRL language and integrated to update the ontology
Synonym	Words	Ontology population	Synonyms to all actions and targets can be included by creating new ontology instances

Angleraud, A., Mehman Sefat, A., Netzev, M., & Pieters, R. (2021). Coordinating shared tasks in human-robot collaboration by commands. Frontiers in Robotics and AI, 8, 734548.

Summary Multi Manipulator Manipulation

- Force Control (e.g. carry a heavy load together)
  - Centralized
  - De-centralized -> multi-agent control -> collaboration
  - Distributed Cooperation: share some information
- Position Control
  - Precisely follow pre-programmed trajectories
  - Motion planning: on-the-fly plan new trajectories for cooperation
- Sequential manipulation
- Task level coordination, collaboration & cooperation

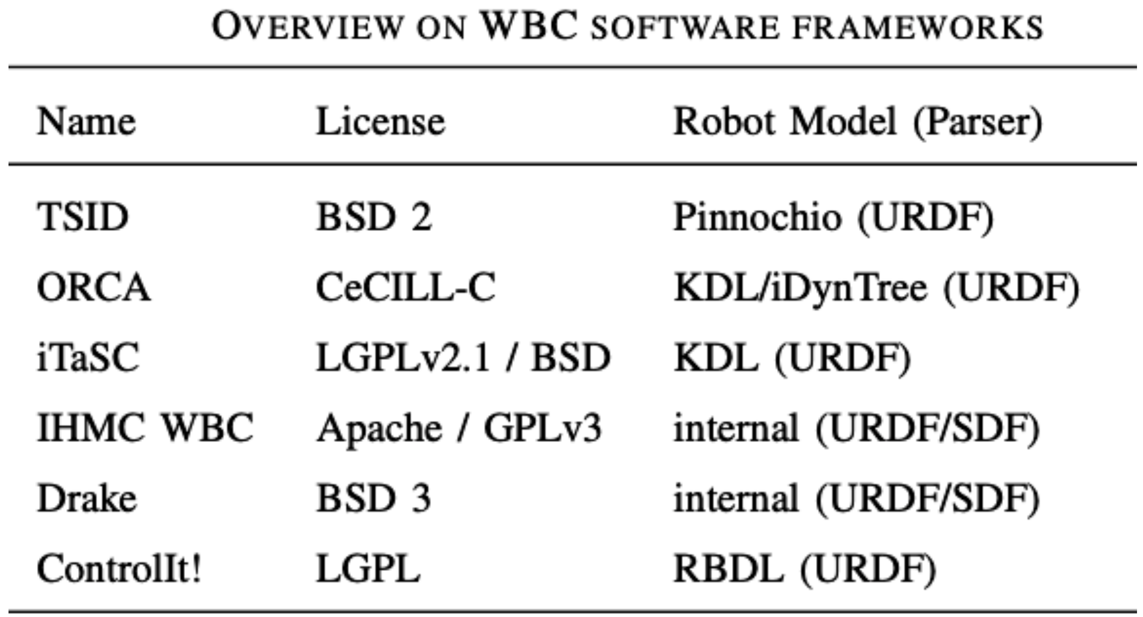


- Whole-body control (e.g. dual-arms & mobile base)

WHOLE-BODY CONTROL

Whole-Body Control

- Plan & control for combined motion of manipulator and mobile base
- Particular popular for legged, especially humanoid robots
  - Tree-like kinematic structure – no loops!
- Also needed for aerial, underwater, surface vehicles and space robots:
  - Manipulation forces move the mobile base!
- Ground vehicles: non-holonomic kinematics restricts possible motions -> difficult and unpopular
  - Alternative: holonomic ground robots!
- MPC popular
- Reinforcement Learning very popular



OVERVIEW ON WBC SOFTWARE FRAMEWORKS

Name	License	Robot Model (Parser)
TSID	BSD 2	Pinnochio (URDF)
ORCA	CeCILL-C	KDL/iDynTree (URDF)
iTaSC	LGPLv2.1 / BSD	KDL (URDF)
IHMC WBC	Apache / GPLv3	internal (URDF/SDF)
Drake	BSD 3	internal (URDF/SDF)
ControlIt!	LGPL	RBDL (URDF)

Mronga, D., Kumar, S., & Kirchner, F. (2022, May). Whole-body control of series-parallel hybrid robots. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 228-234). IEEE.

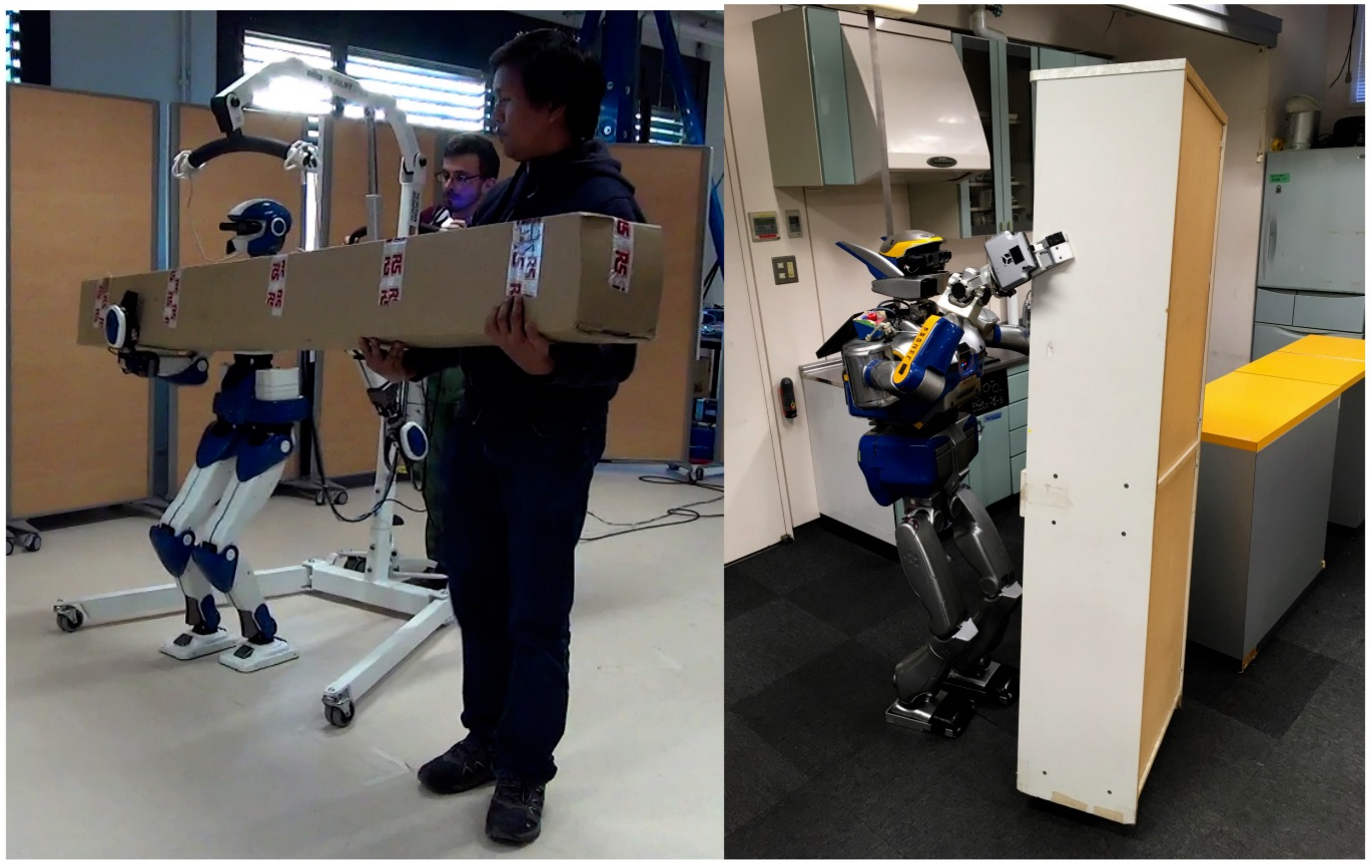
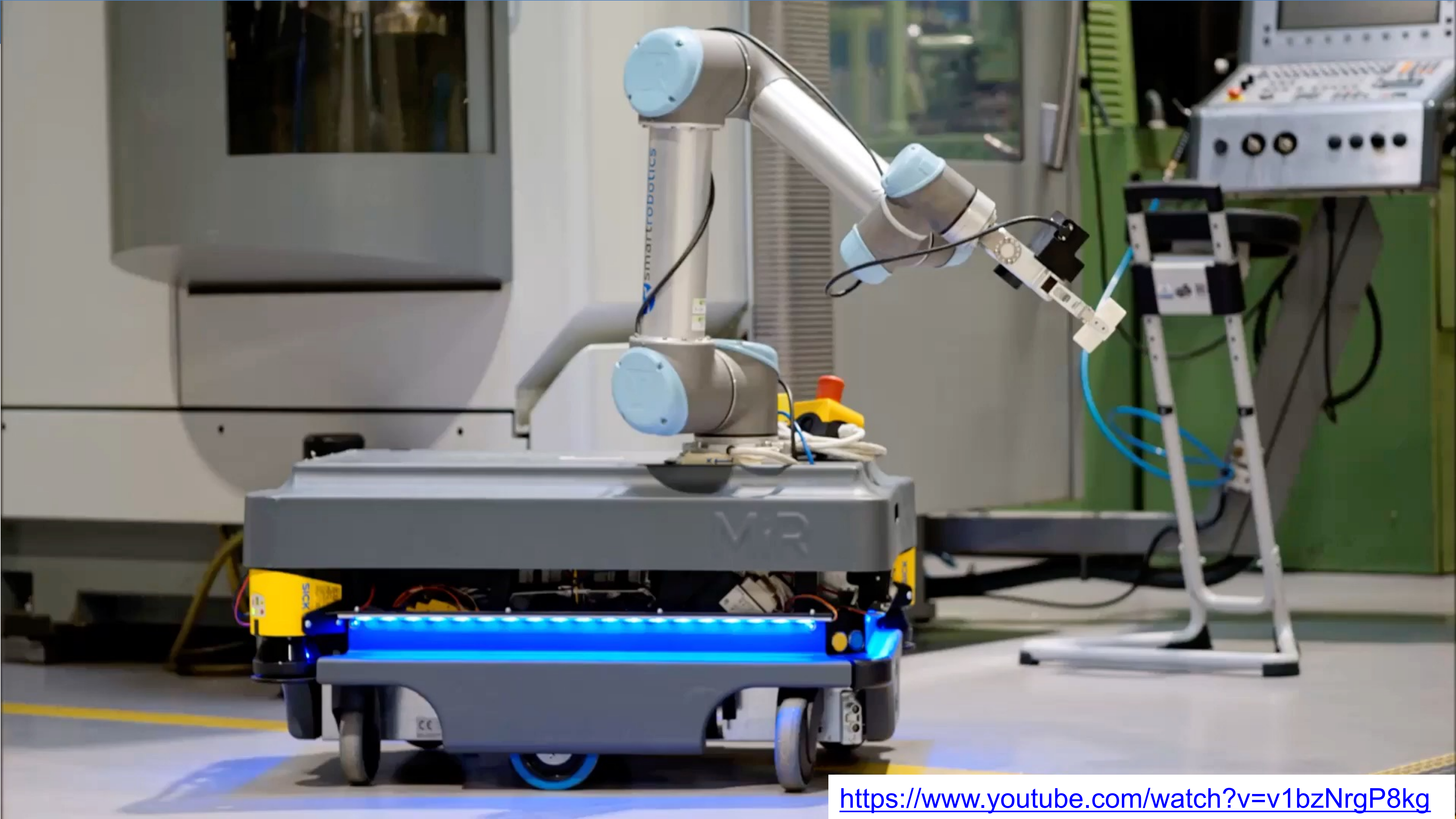
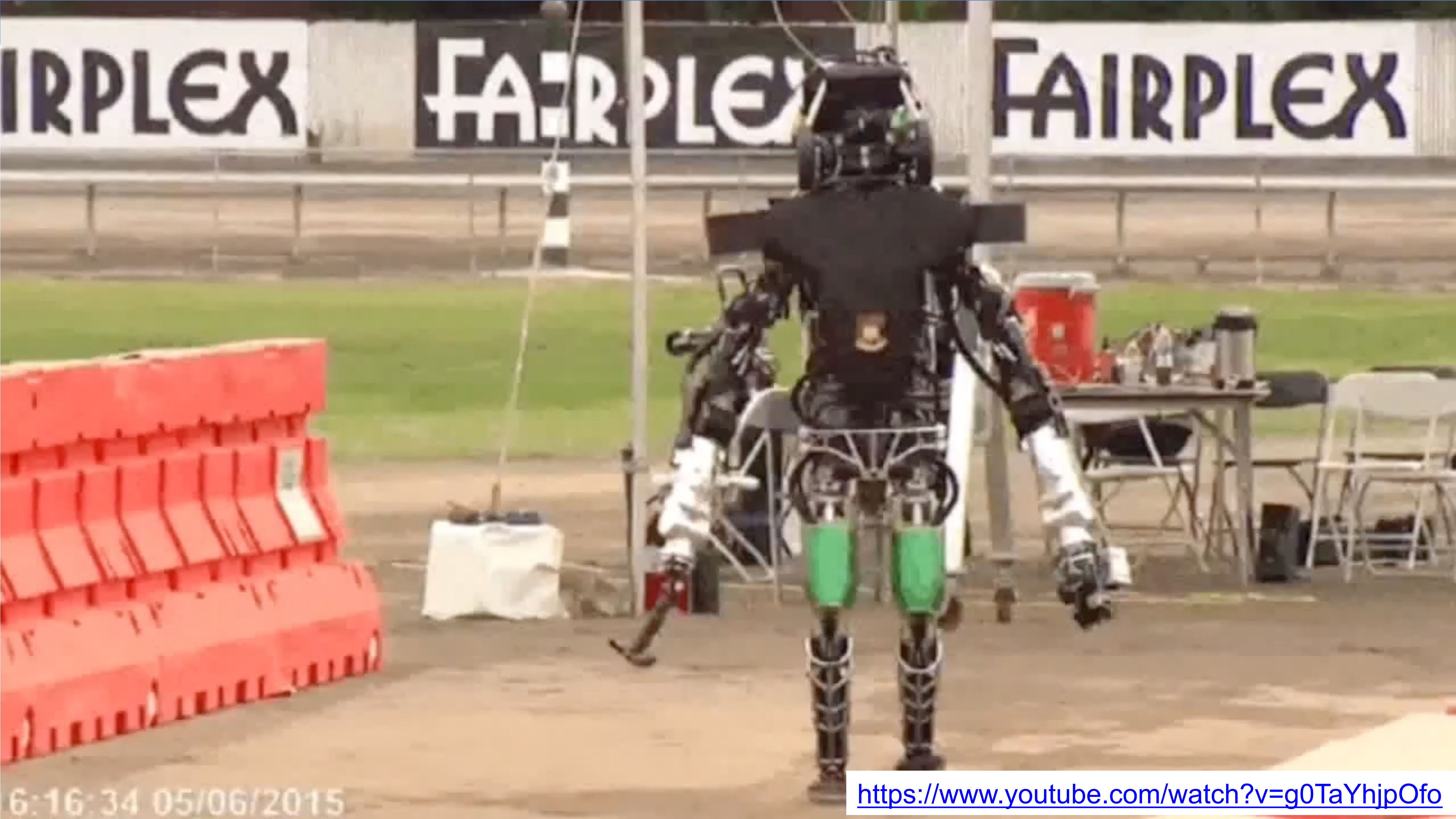


Fig. 3: (left) HRP-4 holding a large box with a human while walking (Agravante et al. (2019)) (right) HRP-2 pivoting a furniture (Murooka et al. (2017)).

Stasse, O., & Righetti, L. (2020). Whole-body manipulation. *Encyclopedia of Robotics*, 1-9.

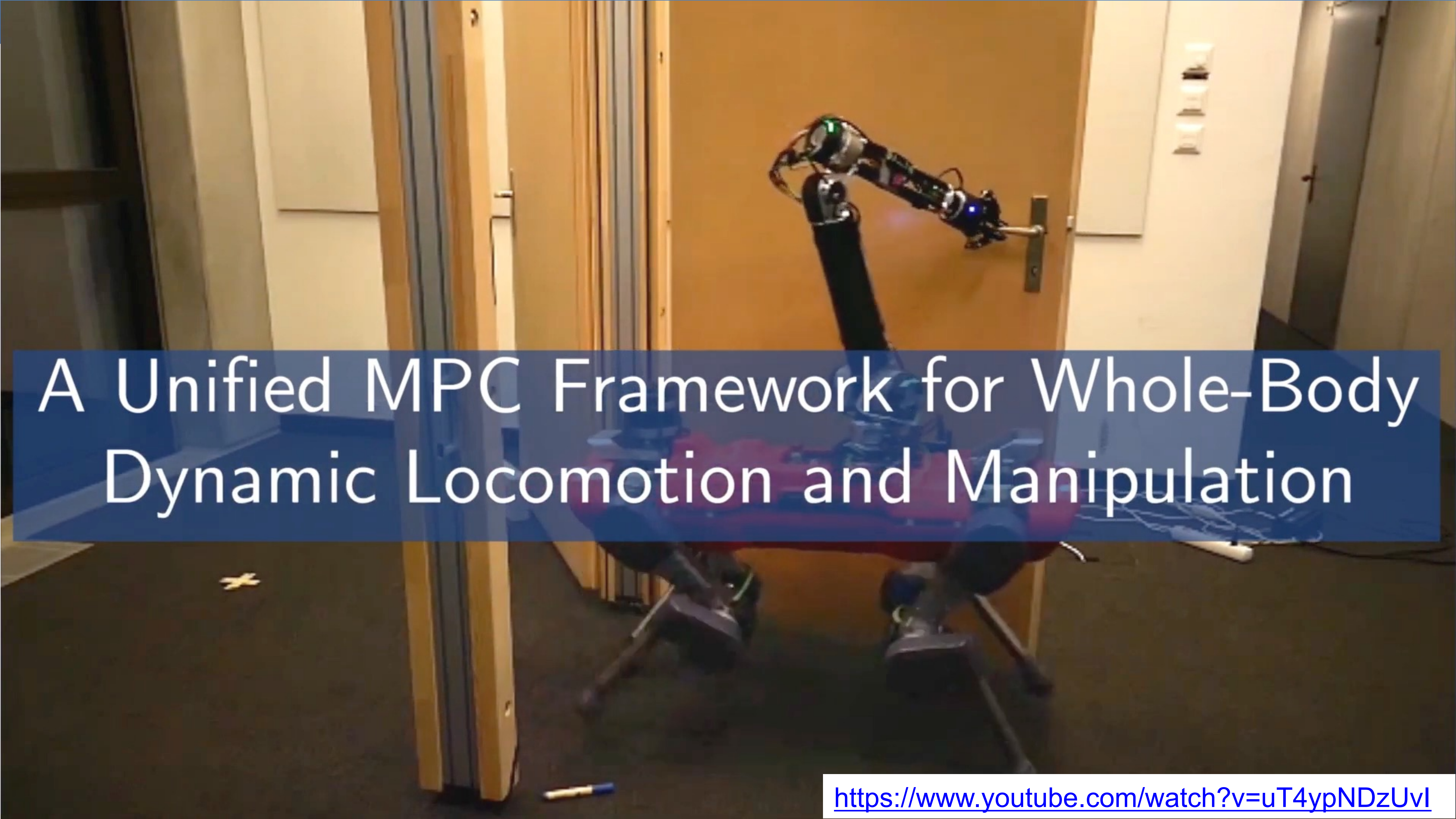


https://www.youtube.com/watch?v=v1bzNrgP8kg



https://www.youtube.com/watch?v=g0TaYhjpOfo

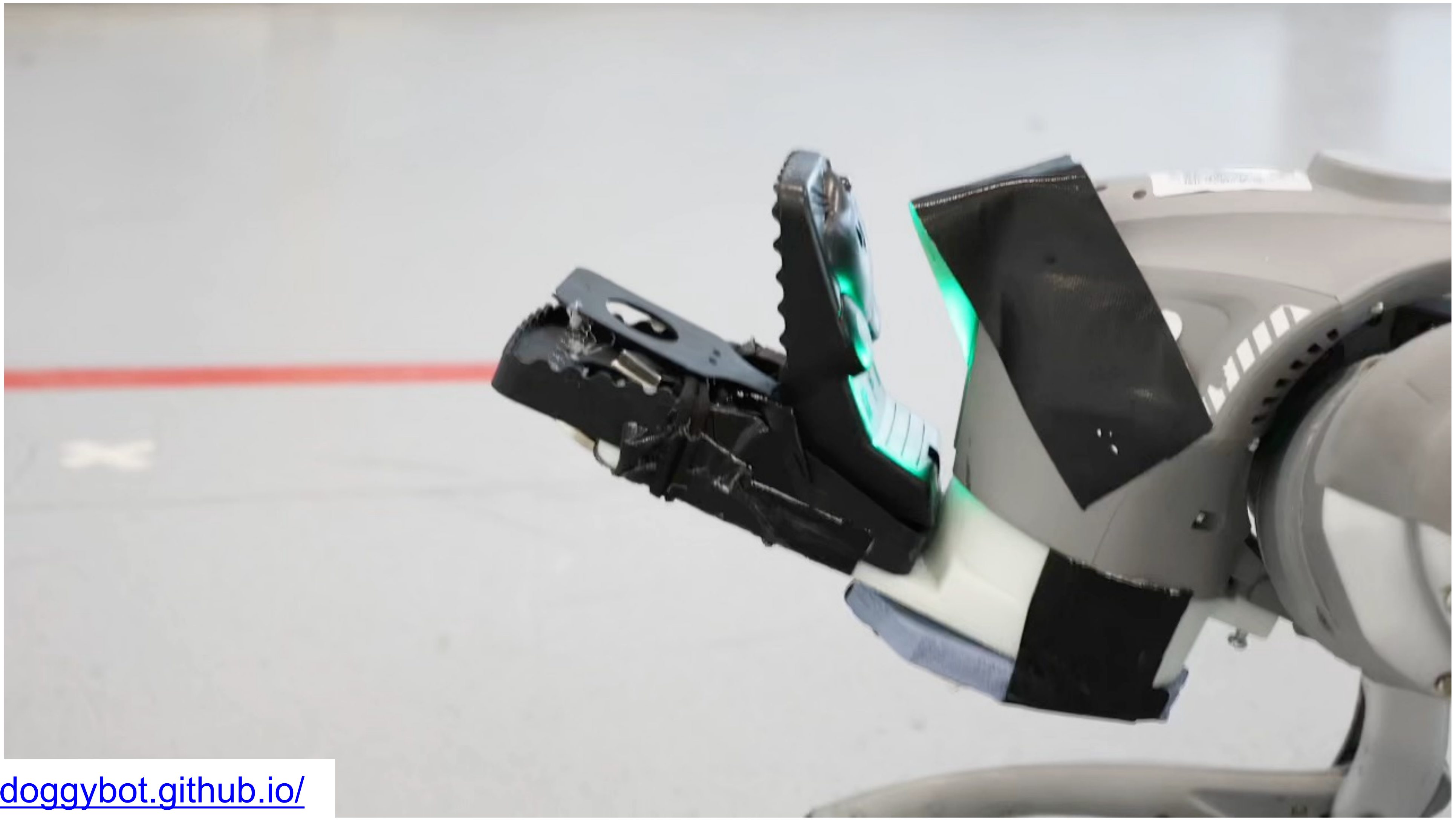
https://www.youtube.com/watch?v=QXS1gBGc23A



A Unified MPC Framework for Whole-Body Dynamic Locomotion and Manipulation

https://www.youtube.com/watch?v=uT4ypNDzUvI

Playful DoggyBot: Learning Agile and Precise Quadrupedal Locomotion



https://playful-doggybot.github.io/

ROBOTIC LEARNING

Overview of Learning Approaches in Robotics

- **Goal:** To explore various learning paradigms that enable robots to perform tasks autonomously.
- **Categories:**
  - Model-Free vs. Model-Based Learning
  - Supervised vs. Unsupervised Learning
  - Passive vs. Active Learning
  - Reinforcement Learning (RL)
  - Imitation Learning
  - End-to-End Deep Learning
  - Actor-Critic Learning
  - Evolutionary Algorithms
  - Transfer Learning
  - Self-Supervised Learning
  - Few-Shot and Zero-Shot Learning
  - Multi-Agent Learning
  - Curriculum Learning
  - LLM
  - Foundation Models
  - Other types of “learning”

Next few slides
With help from ChatGPT :D

Model-Based vs Model-Free Learning

Model-Based Learning:

- Involves learning a model of the environment or dynamics (e.g., using physics or system dynamics).
- Robot can plan and predict actions based on this model.
- **Example:** Planning with a learned dynamics model in robotic control tasks.

Model-Free Learning:

- Directly learns a mapping from states to actions or rewards without modeling the environment.
- **Example**: Q-learning or policy gradient methods in Reinforcement Learning.

Supervised vs. Unsupervised Learning

Supervised Learning:

- Learning from labeled data (input-output pairs).
- Requires large datasets and human supervision.
- **Example:** Image classification for object detection in robotics, such as recognizing "graspable" objects in a scene.

Unsupervised Learning:

- Learning from unlabeled data to find hidden patterns (e.g., clustering or representation learning).
- **Example:** A robot exploring its environment autonomously to cluster sensory data (e.g., LIDAR or visual data) into distinct regions like walls, furniture, or open spaces. This clustering can later help the robot map its environment for navigation.

Passive vs Active Learning

Passive Learning:

- The robot learns from a fixed dataset (either labeled or unlabeled).
- **Example:** Supervised learning with a fixed dataset.

Active Learning:

- The robot queries the environment for more informative data based on its current knowledge or uncertainty.
- **Example:** A robot selects which objects to interact with in order to maximize learning.

End-to-End Deep Learning in Robotics

- **Definition:** Learning a direct mapping from raw input (e.g., images, sensory data) to the output (e.g., control commands).
- **Example:** A robot controlling a gripper using only camera images.
- **Advantages:** Simplifies the pipeline by learning a direct mapping.
- **Challenges:** Requires large amounts of labeled data.

Reinforcement Learning (RL)

- **Definition:** An agent learns to take actions in an environment to maximize cumulative reward over time.
- **Key Components:** States, actions, rewards, policy.
- **Example:** Training a robot to navigate using trial-and-error.
- **Types:**
  - **Model-Free:** Methods like Q-learning, policy gradients.
  - **Model-Based:** Use of learned models to simulate and plan actions.

Imitation Learning

- **Definition:** Robots learn by observing and imitating human demonstrations or expert behaviors.

- **Approaches:**
  - **Behavior Cloning:** Supervised learning from demonstrations.
  - **Inverse Reinforcement Learning (IRL):** Learning the underlying reward function from expert demonstrations.

- **Example:** Teaching a robot to grasp objects by mimicking human actions.

Actor-Critic Learning

- **Definition:** A type of Reinforcement Learning where two models (actor and critic) are trained simultaneously.
  - **Actor:** Decides which action to take based on the current state.
  - **Critic:** Evaluates the chosen action based on the reward.
- **Advantages:** More stable training by using both policy and value functions.
- **Example:** Robotic manipulation tasks requiring fine-grained control.

Evolutionary Algorithms in Robotics

- **Definition:** Optimization methods inspired by natural selection, such as genetic algorithms or neuroevolution.
- **Example:** Optimizing robot locomotion for uneven terrains or designing neural network architectures for control.
- **Advantages:** Effective for tasks where gradient-based optimization struggles or is infeasible.
- **Challenges:** Computationally expensive and may require many iterations.

Transfer Learning in Robotics

- **Definition:** Leveraging knowledge from one domain/task to accelerate learning in another.
- **Example:** Transferring knowledge from simulation-trained robots to real-world environments.
- **Advantages:** Reduces training time and reliance on large datasets.
- **Challenges:** Ensuring the transfer is effective despite domain differences.

Self-Supervised Learning in Robotics

- **Definition:** Robots generate their own training signals from raw, unlabeled data.
- **Example:** Learning to predict the next sensory input (e.g., next video frame) for tasks like navigation or manipulation.
- **Advantages:** Removes dependence on human-labeled data, making learning scalable.
- **Challenges:** Designing effective self-supervised objectives.

Few-Shot and Zero-Shot Learning in Robotics

- **Definition:** Learning to generalize with few or no examples of the target task.
- **Example:** Teaching a robot to recognize and manipulate a novel object with only one image or no prior direct training.
- **Advantages:** Crucial for real-world scalability.
- **Challenges:** Relies heavily on robust pre-trained models.

Multi-Agent Learning in Robotics

- **Definition:** Learning strategies for robots that interact and collaborate or compete in shared environments.
- **Example:** Swarms of drones coordinating for mapping or delivery tasks.
- **Advantages:** Enables cooperative behaviors in teams of robots.
- **Challenges:** Complex interactions and scalability.

Curriculum Learning in Robotics

- **Definition:** Gradually increasing the complexity of tasks during training.
- **Example:** Training a robot arm to first stack one block, then multiple blocks, before solving general stacking problems.
- **Advantages:** Stabilizes and accelerates learning.
- **Challenges:** Designing a proper curriculum and transitions between tasks.

LLM for Robotics

- **Definition:** LLMs are AI systems trained on massive text corpora to process, understand, and generate human-like text.
- **Key Capabilities in Robotics:**
  - **Natural Language Understanding:** Interpreting commands and queries.
  - **Knowledge Integration:** Retrieving and applying knowledge to tasks (e.g., assembly instructions).
  - **Reasoning and Task Decomposition:** Breaking down complex instructions into actionable steps.
- **Advantages:**
  - Provides high-level reasoning and task planning.
  - Reduces the need for detailed programming in language-based tasks.
  - Can handle diverse instructions using pre-trained knowledge

How LLMs Are Used in Robotics

- **Applications:**
  - **Human-Robot Interaction:** Robots can interpret and execute natural language instructions (e.g., “Bring me a cup of water”).
  - **Task Planning:** Combining linguistic reasoning with real-world task execution.
  - **Multi-Modal Integration:** Enhancing decision-making by linking text, vision, and sensory inputs.
- **Challenges:**
  - Ensuring grounding in physical environments (e.g., interpreting "left" in a spatial context).
  - Real-time response constraints due to the size of models.
  - Domain-specific fine-tuning for robotics applications.

Robotics Foundation Models

- **Definition:** Large-scale AI models pre-trained on diverse, multi-modal datasets (e.g., text, images, videos).
- **Core Characteristics:**
  - **Versatile Pre-training:** Serve as a base for fine-tuning on specific tasks.
  - **Multi-Modal Understanding:** Integrate text, vision, and other sensory inputs for broader applicability.
- **Key Advantages for Robotics:**
  - Generalize across multiple tasks with minimal retraining.
  - Simplify the training pipeline by leveraging shared representations.
  - Adaptable to new tasks without extensive data collection.

How Foundation Models Empower Robotics

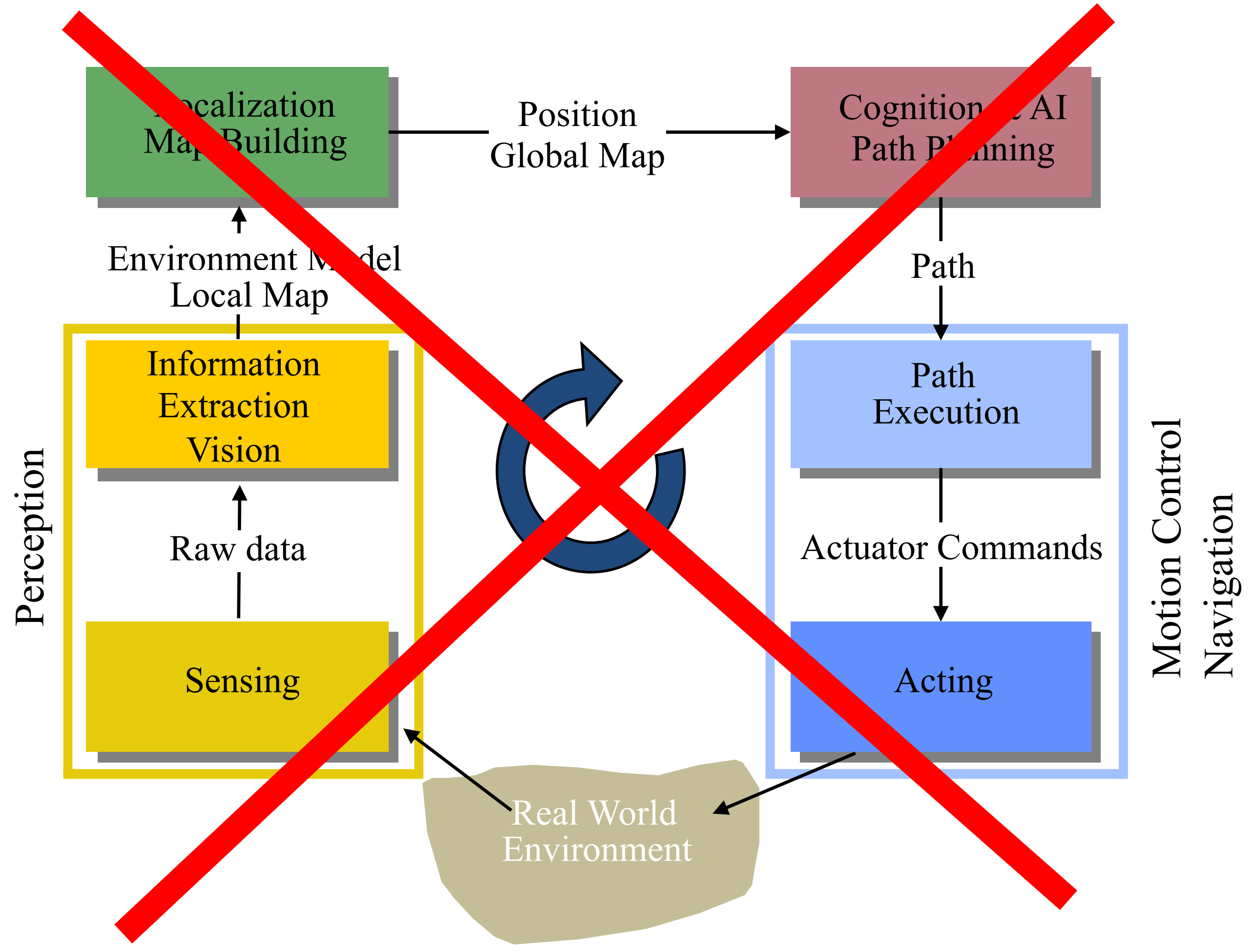
- **Applications:**
  - **Perception:** Models like CLIP interpret visual data for scene understanding.
  - **Control:** Leveraging shared representations for motion planning and actuation.
  - **Task Generalization:** Performing varied tasks without task-specific training.
  - **Simulation-to-Real Transfer:** Reducing the gap between simulated and real-world performance.
- **Challenges:**
  - High computational costs for pre-training and fine-tuning.
  - Limited grounding in physical dynamics without additional modeling.
  - Potential biases from pre-training on non-robotic data.

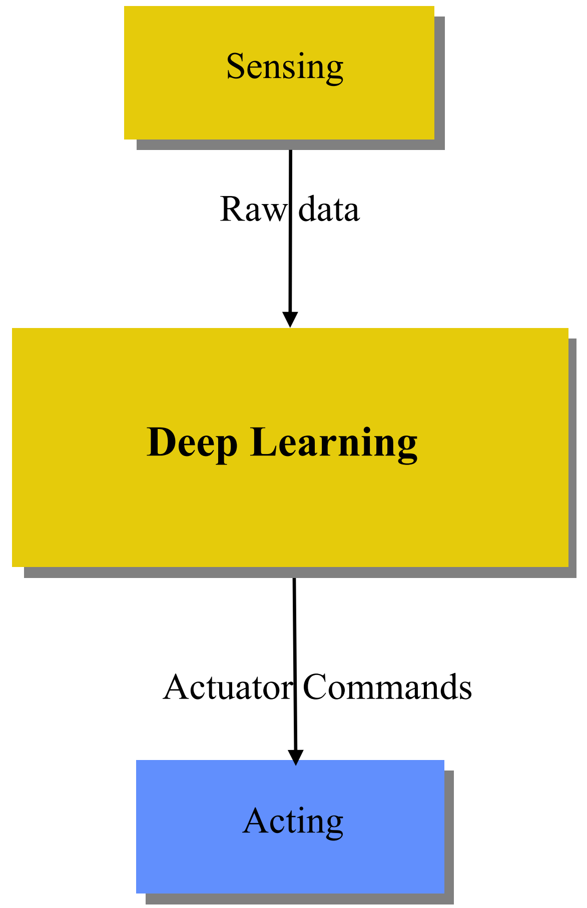
Other types of learning

- Term “learning” also used in different contexts in robotics, e.g.
- SLAM: the robot “learns” the map (by SLAM algorithm)
- Adaptive Control Learning
- Motion Planning
- Meta-learning: learn to learn
- Lifelong Learning (Continuous Learning)
- …

REINFORCEMENT LEARNING

End-to-End Deep Learning



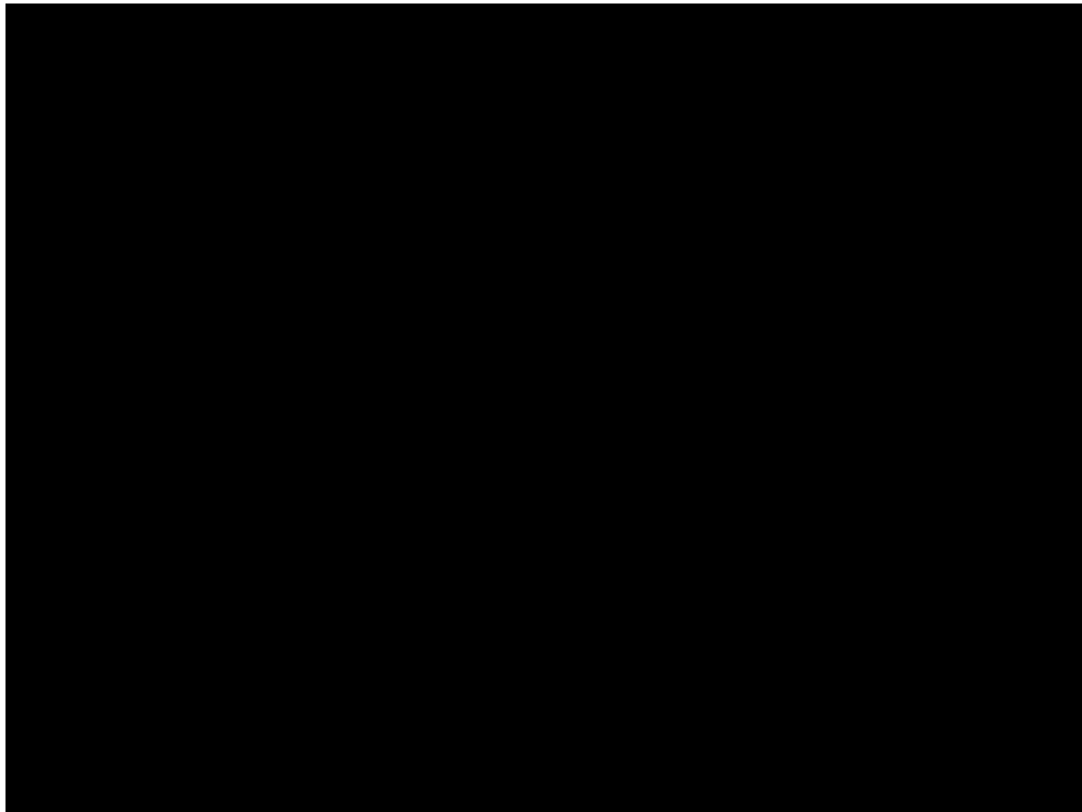


End-to-End Deep Reinforcement Learning

- From sensors to actuation: one layered or recurrent neural network! =>
  - NOT classical general control scheme (Perception, SLAM, Cognition & Planning, Navigation)

- Needs reward signal: sparse, noisy, delayed!
- Take time into account: input frames are related!

- Gained interest 2013 again with:
  - Deep Mind (google) playing ATARI 2600 games
  - Video: Breakout
  - Learned 7 games
  - Surpasses human expert in 3



https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

What is Reinforcement Learning?

- Learning from interaction with an environment to achieve some long-term goal that is related to the state of the environment
- The goal is defined by reward signal, which must be maximized
- Agent must be able to partially/fully sense the environment state and take actions to influence the environment state
- The state is typically described with a feature-vector

Material adapted from
Karan Kathpalia
https://www.cs.princeton.edu/courses/archive/spring17/cos598F/lectures/RL.pptx

Exploration versus Exploitation

- We want a reinforcement learning agent to earn lots of reward
- The agent must prefer past actions that have been found to be effective at producing reward
- The agent must exploit what it already knows to obtain reward
- The agent must select untested actions to discover reward-producing actions
- The agent must explore actions to make better action selections in the future
- Trade-off between exploration and exploitation

Reinforcement Learning Systems

- Reinforcement learning systems have 4 main elements:
  - Policy
  - Reward signal
  - Value function
  - Optional model of the environment

Policy

- A policy is a mapping from the perceived states of the environment to actions to be taken when in those states

- A reinforcement learning agent uses a policy to select actions given the current environment state

Reward Signal

- The reward signal defines the goal
- On each time step, the environment sends a single number called the reward to the reinforcement learning agent
- The agent’s objective is to maximize the total reward that it receives over the long run
- The reward signal is used to alter the policy

Value Function (1)

- The reward signal indicates what is good in the short run while the value function indicates what is good in the long run
- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting in that state
- Compute the value using the states that are likely to follow the current state and the rewards available in those states
- Future rewards may be time-discounted with a factor in the interval [0, 1]

Value Function (2)

- Use the values to make and evaluate decisions
- Action choices are made based on value judgements
- Prefer actions that bring about states of highest value instead of highest reward
- Rewards are given directly by the environment
- Values must continually be re-estimated from the sequence of observations that an agent makes over its lifetime

Model-free versus Model-based

- A model of the environment allows inferences to be made about how the environment will behave
- Example: Given a state and an action to be taken while in that state, the model could predict the next state and the next reward
- Models are used for planning, which means deciding on a course of action by considering possible future situations before they are experienced
- Model-based methods use models and planning. Think of this as modelling the dynamics
- Model-free methods learn exclusively from trial-and-error (i.e. no modelling of the environment)
- Followoing: model-free methods

On-policy versus Off-policy

- An on-policy agent learns only about the policy that it is executing
- An off-policy agent learns about a policy or policies different from the one that it is executing

Credit Assignment Problem

- Given a sequence of states and actions, and the final sum of time-discounted future rewards, how do we infer which actions were effective at producing lots of reward and which actions were not effective?

- How do we assign credit for the observed rewards given a sequence of actions over time?

- Every reinforcement learning algorithm must address this problem

Reward Design

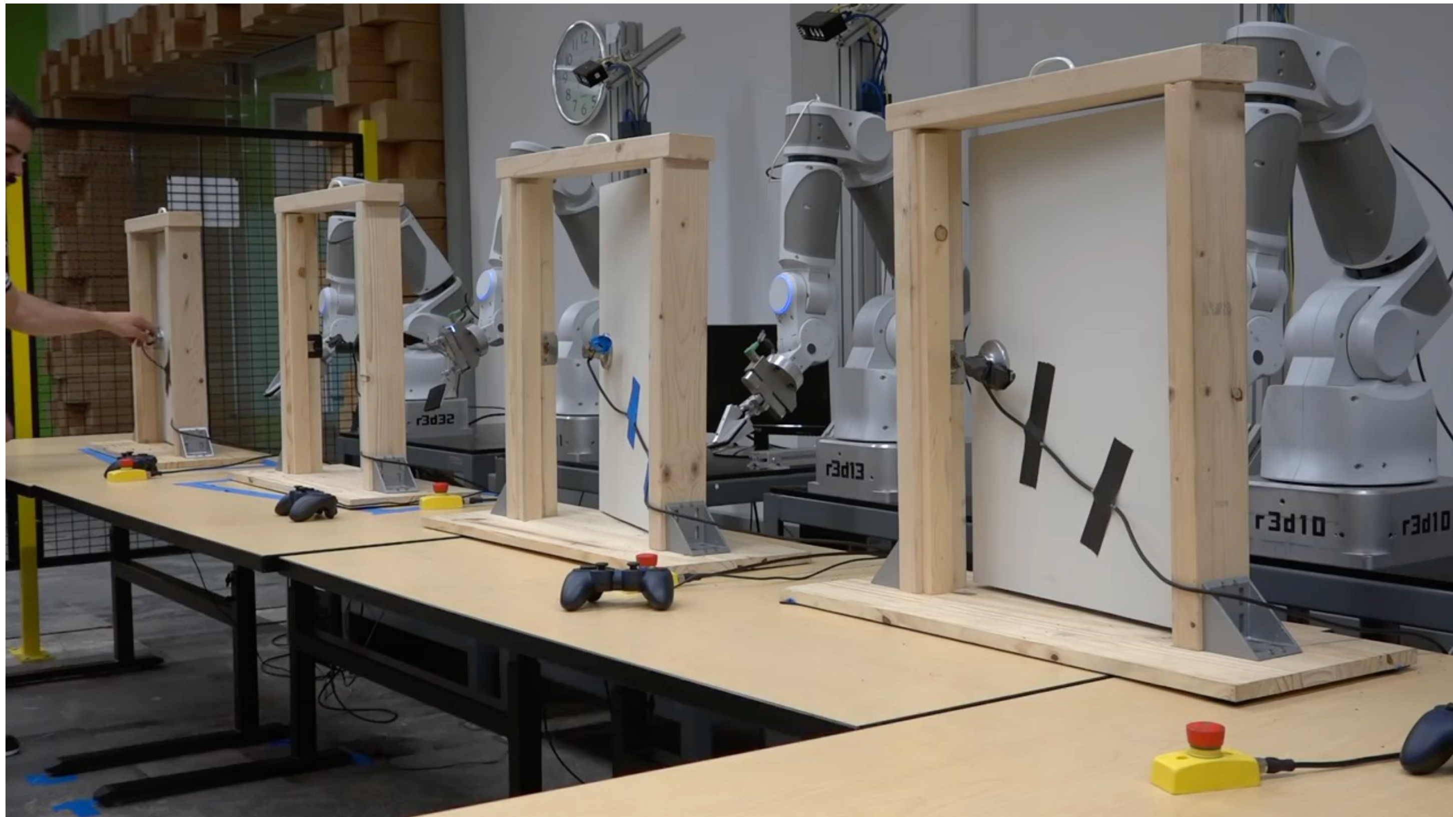
- We need rewards to guide the agent to achieve its goal
- Option 1: Hand-designed reward functions
- This is a black art
- Option 2: Learn rewards from demonstrations
- Instead of having a human expert tune a system to achieve the desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration

What is Deep Reinforcement Learning?

- Deep reinforcement learning is standard reinforcement learning where a deep neural network is used to approximate either a policy or a value function
- Deep neural networks require lots of real/simulated interaction with the environment to learn
- Lots of trials/interactions is possible in simulated environments
- We can easily parallelize the trials/interaction in simulated environments
- We cannot do this with robotics (no simulations) because action execution takes time, accidents/failures are expensive and there are safety concerns

Google Door Opening Project

- Learn to open doors using Reinforcement learning
  - Learning reward: opening the door
  - Much harder than purely digital learning: very slow iterations!
  - Simulation only helps a bit:
    real world much more complex
- Google and
  UC Berkeley Sergey Levine
- Google very secretive …



https://www.wired.com/2017/01/googles-go-playing-machine-opens-door-robots-learn/

RL Algorithms

• Finite Markov Decision Processes	MDP
• Temporal-Difference Learning	TD Learning
• State-Action-Reward-State-Action	SARSA TD Learning
• Q-learning: Off-policy TD Control	
• Deep Q-Networks	DQN

- Policy Gradient Methods
  - Actor-Critic Methods

- Asynchronous Reinforcement Learning