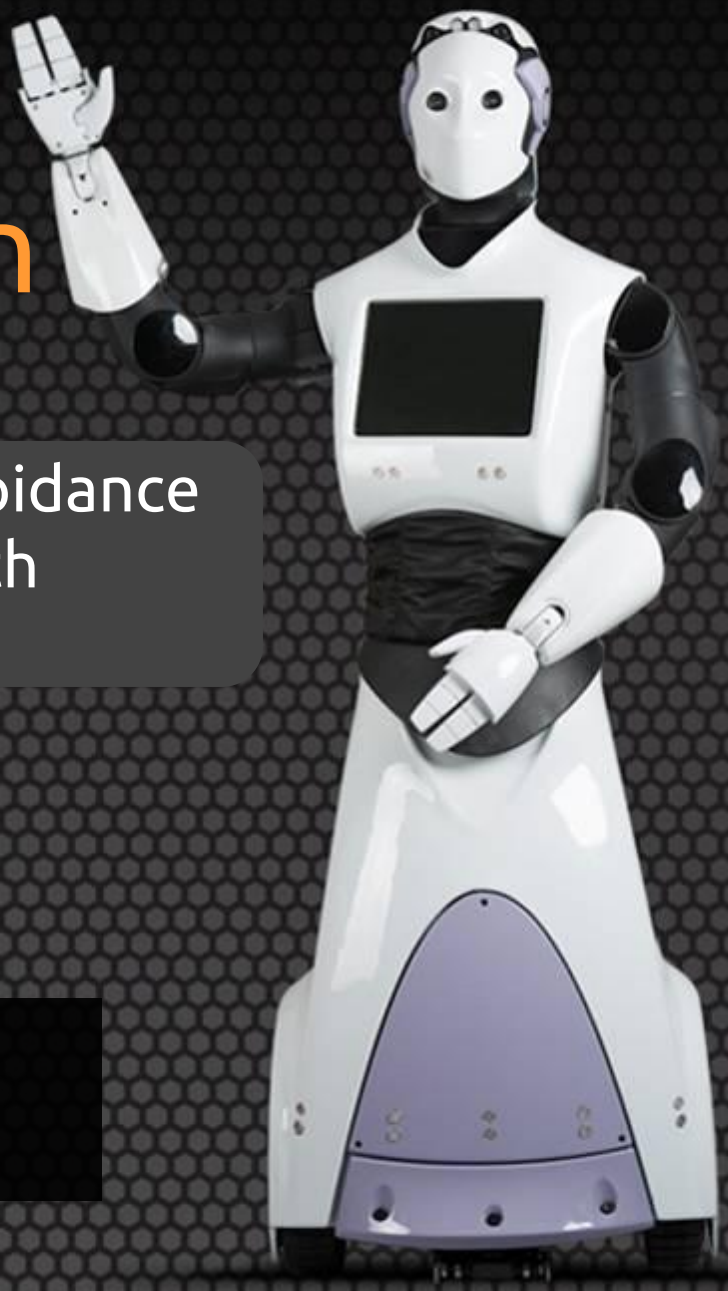
Final Presentation

Realizing Online (Self-)Collision Avoidance Based on Inequality Constraints with Hierarchical Inverse Kinematics

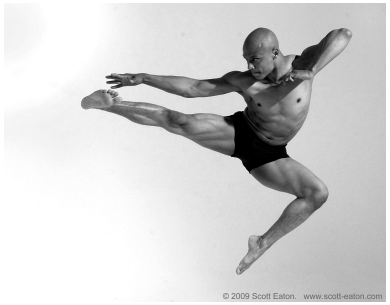


PAL ROBOTICS

Karsten Knese
09. May 2014

Index

1. Introduction



Dexterity

Multitasking

Real time

Collision avoidance

1. Introduction

Dexterity

- Full Body Motion Control
- Redundancy
- General Inverse Kinematics
- Least Squares

Multitasking

- Hierarchy of Tasks
- Priorities
- Quadratic Programming
- Equality Constraints
- Inequality Constraints

Real Time

- No Offline Planning
- IK Control Scheme
- Nullspace Optimization
- Hierarchical Complete Orthogonal Decomposition

Collision Avoidance

- Safety Distance
- Self-Collision Avoidance
- External Collision
- Capsule Decomposition

2. Related Work - Hardware

Degree of freedom:

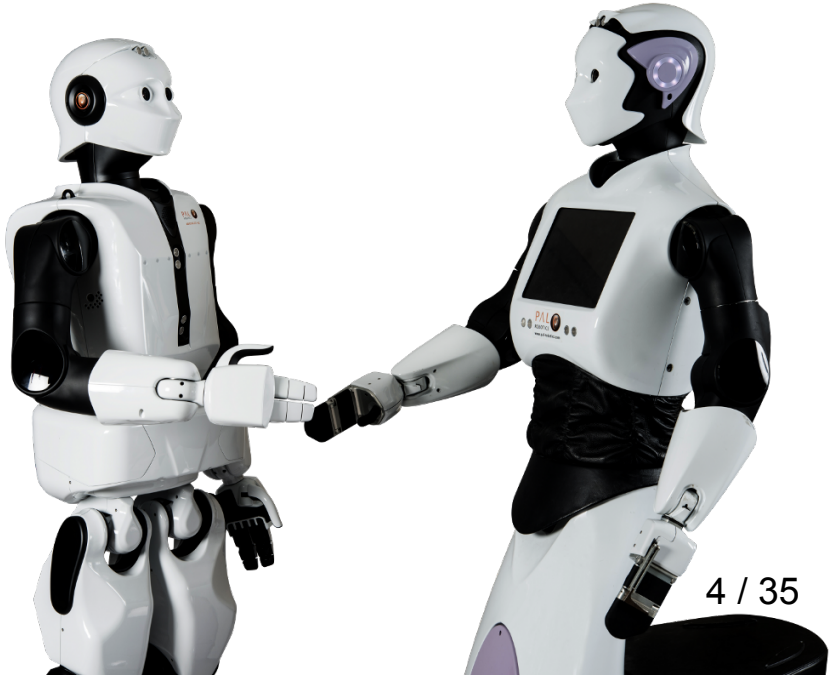
- 7 DoF arm
- 7(3) DoF hand
- 2 DoF Torso
- 2 DoF Head
- *6 DoF Leg (REEM-C only)*

Sensors:

- Stereo-Vision camera
- IMU
- Ultrasound
- Laser range
- *Force/Torque (REEM-C only)*

7*2 + 2 + 2 (+6*2) = 18 (30) DoF
→ Redundant System !

5 independent sensor types
→ Multiple Tasks !



Least Squares

$$Ax = b \quad \text{with } A \in \mathbb{R}^{m \times n}, m > n$$

$$\begin{aligned} r &= Ax - b \\ x^* &= \arg\min_x \|Ax - b\| \\ \|r\|^2 &= xA^TAx - 2b^TAx + b^Tb \\ \nabla_x \|r\|^2 &= 2A^TAx - 2A^Tb = 0 \end{aligned}$$

Least Squares - Hierarchy

$$x^* = A^+ b \quad \text{with } A^+ = (A^T A)^{-1} A^T$$

$$x^* = A^+ b + P x_p$$

→ Nullspace projection

$$x_2^* = \arg\min_{x_2} \|A_2 x_2 - b_2\|$$

$$\|A_2(A_1^+ b_1 + P_1 x_2) - b_2\|$$

$$\|A_2 P_1 x_2 - (b_2 - A_2 A_1^+ b_1)\|$$

→ Recursive hierarchy projection [Siciliano 1991]

$$x_2^* = (A_2 P_1)^+ (b_2 - A_2 A_1^+ b_1) + P_2 x_3$$

$$x_n^* = \sum_{k=1}^{n} (A_k P_{k-1})^+ (b_k - A_k A_{k-1}^+ b_{k-1}) + P_n x_{n+1}$$

PAL ROBOTICS

Quadratic Programming

$$\frac{1}{2}x^T Q x + c^T x + r_o$$
$$\text{subject to } Dx = e \quad \in \mathcal{E}$$
$$\text{subject to } Dx \leq e \quad \in \mathcal{I}$$

→ Lagrangian formulation

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^T Q x + c^T x + r_o + \lambda^T (Dx - e)$$

PAL ROBOTICS

Quadratic Programming cont

$$\mathcal{L}(x, \lambda) = \frac{1}{2} x^T Q x + c^T x + r_o + \lambda^T (Dx - e)$$

→ Karush-Kuhn-Tucker conditions

$$\begin{aligned}
\nabla \mathcal{L}(x, \lambda) &= 0 \\
d_i^T x &= e_i \quad \in \mathcal{E} \\
d_i^T x &\geq e_i \quad \in \mathcal{I} \\
\lambda &\geq 0 \quad \in \mathcal{I} \\
\lambda (d_i^T x - e_i) &= 0 \quad \in \mathcal{I}
\end{aligned}$$

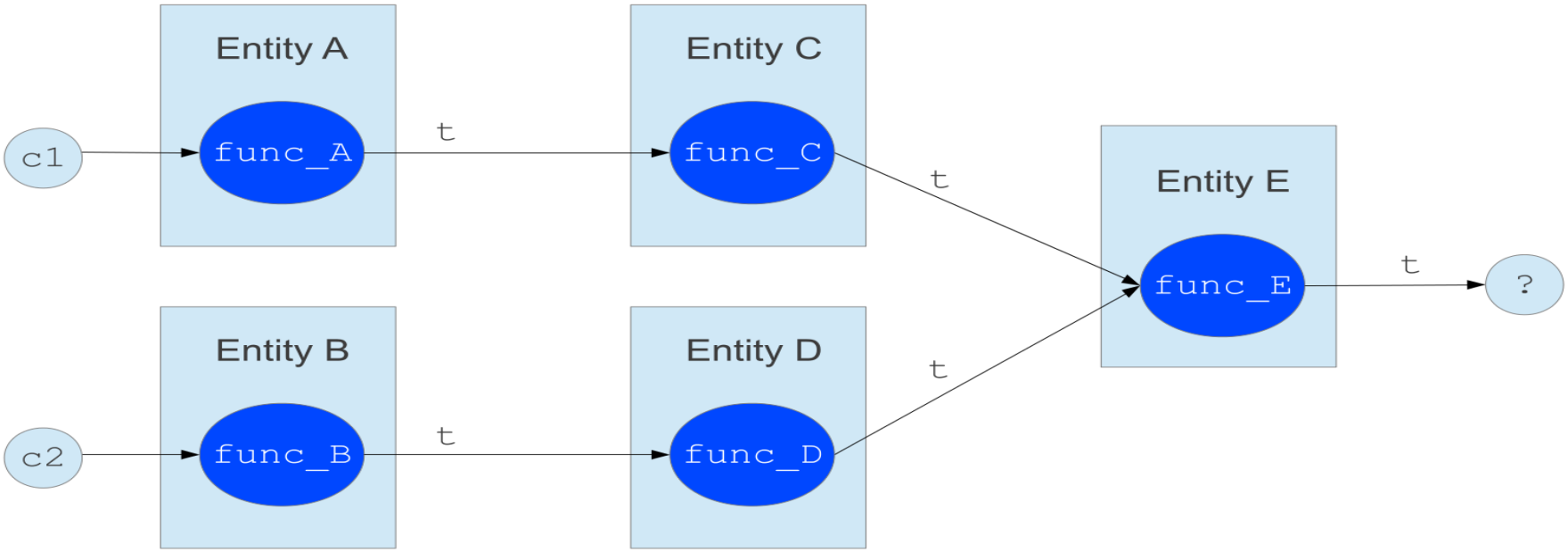
Stack-of-Tasks

Florent Lamiraux, Olivier Stasse and Nicolas Mansard
CNRS-LAAS, Toulouse, France

- control framework for real-time redundant manipulator control
- implementation of dynamic computational graph
  - push & pop tasks at runtime
  - connect entities through virtual signals
- hierarchy preserving quadratic program solver
  - efficient support for inequality constraints
  - no recursive search for solution (active set)

Stack-of-Tasks

Florent Lamiraux, Olivier Stasse and Nicolas Mansard
CNRS-LAAS, Toulouse, France



How to define a task

1.) Equality Constraint

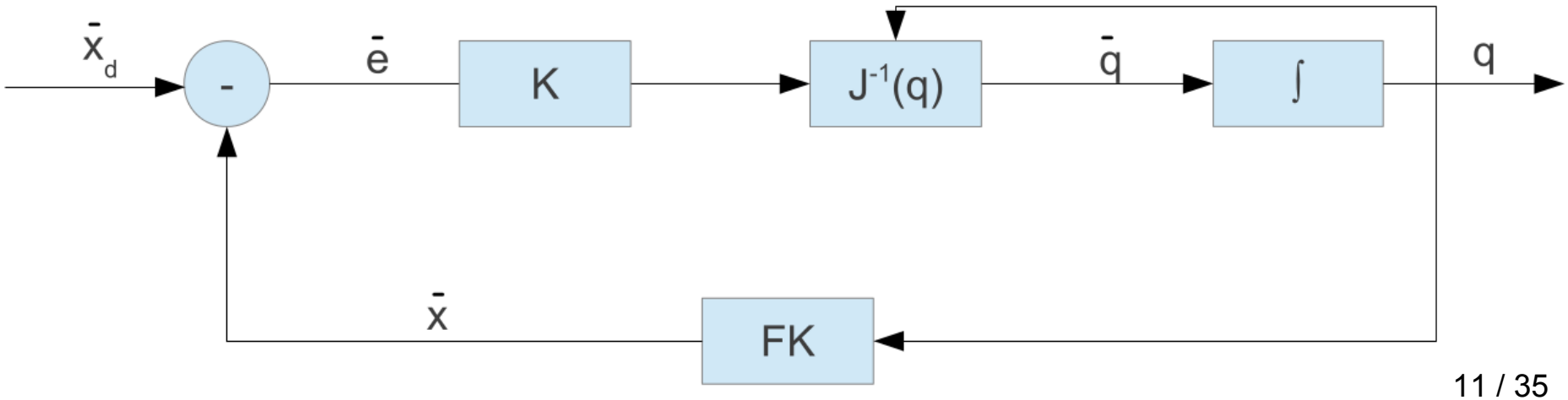
$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{e}}$$
$$= \mathbf{J}^{+}(\dot{\mathbf{x}}_d - \dot{\mathbf{x}})$$

2.) Inequality Constraint

$$\dot{\mathbf{q}} \leqslant \mathbf{J}^{+}\dot{\mathbf{e}}$$
$$\leqslant \mathbf{J}^{+}(\dot{\mathbf{x}}_d - \dot{\mathbf{x}})$$

3.) Joint Space

$$\dot{\mathbf{q}} = \mathbf{I}\dot{\mathbf{q}}_{des}$$
$$= \mathbf{I}\frac{\mathbf{q}_{des} - \mathbf{q}}{\Delta t}$$



2. Related Work

Basic
Video

3. Collision Avoidance - External

Velocity Damping Task

$$\mathbf{n}^T \mathbf{J}(\mathbf{p_1}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{d - ds}{\Delta t} \quad , \epsilon, d, d_s, \Delta t \in \mathbb{R}$$

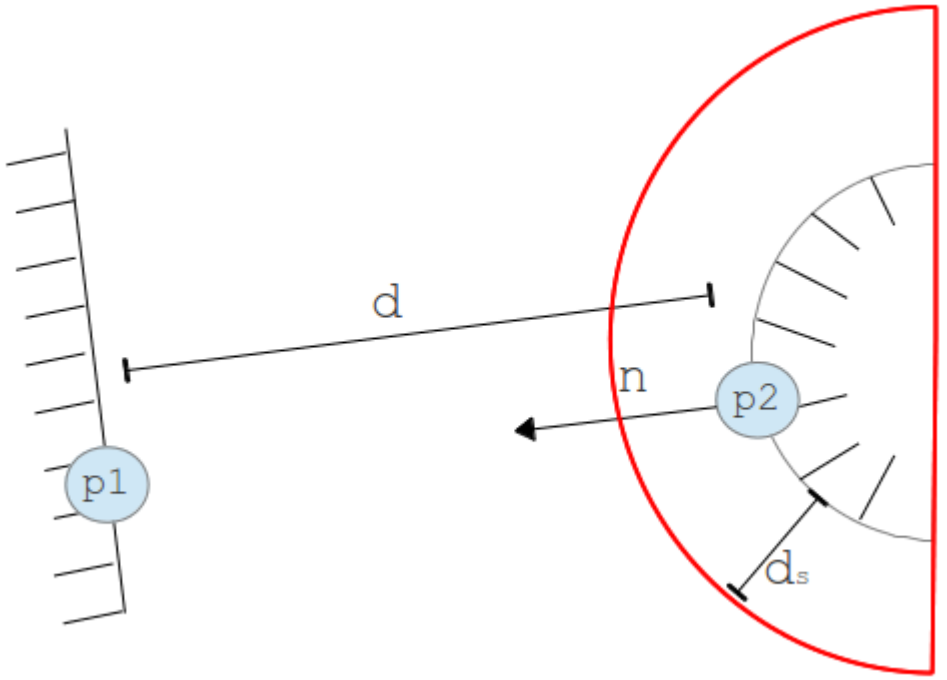
p_1 = actuated body part

p_2 = non-actuated obstacle

d_s = safety distance

$d = ||p_1 - p_2||$

$n = p_1 - p_2 \,/\, ||p_1 - p_2||$

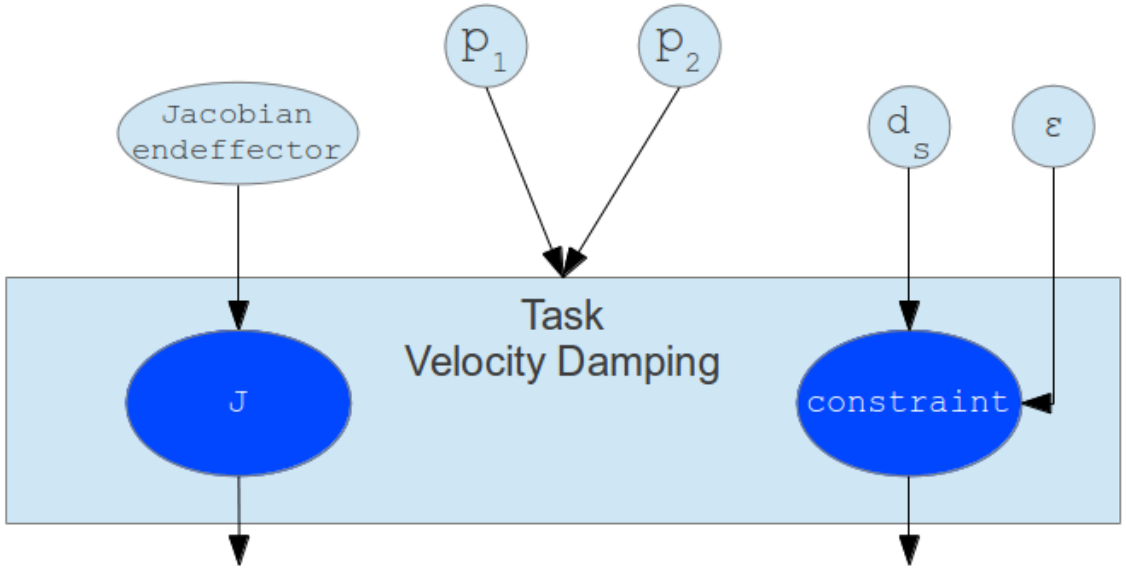


3. Collision Avoidance - External

Velocity Damping Task

$$\frac{\mathbf{p_1} - \mathbf{p_2}}{\|\mathbf{p_1} - \mathbf{p_2}\|}^T \mathbf{J}(\mathbf{p_1}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{d - ds}{\Delta t}$$

Dynamic Graph Integration

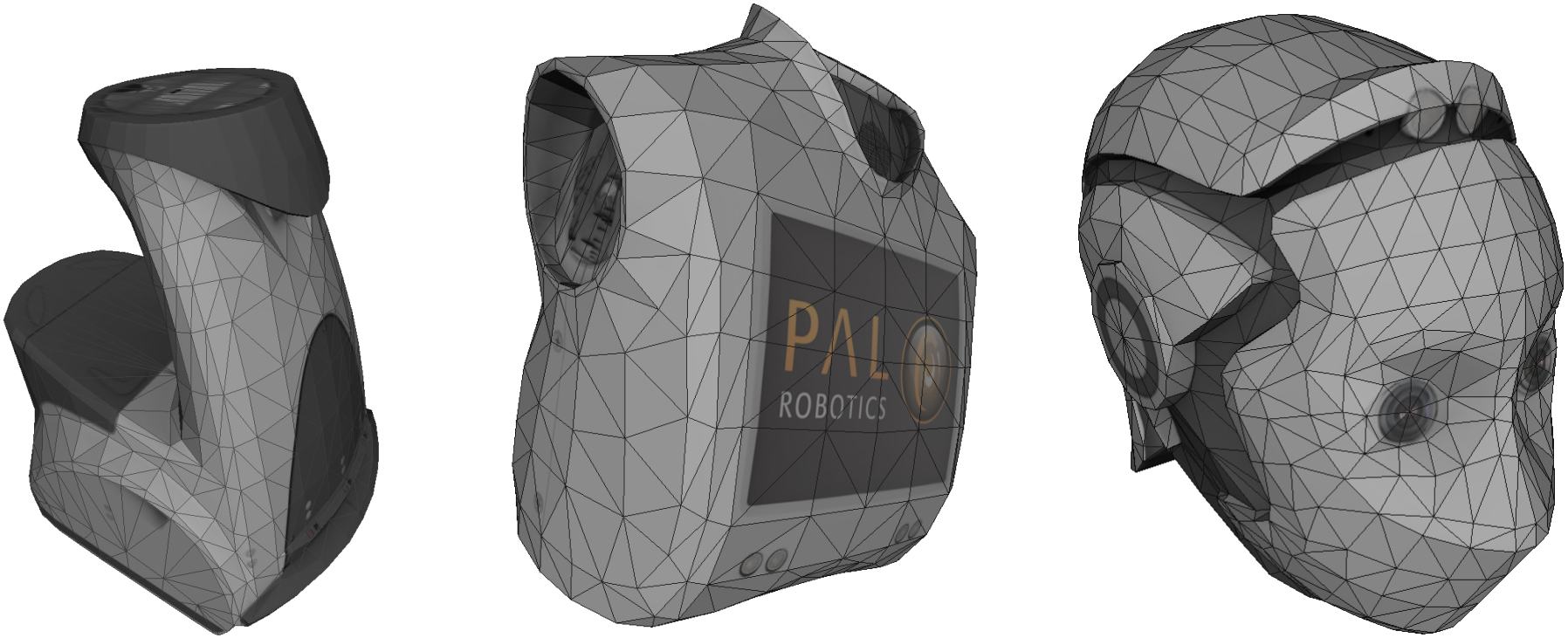


2. Related Work

External Collision Avoidance Video

Velocity Damping Task

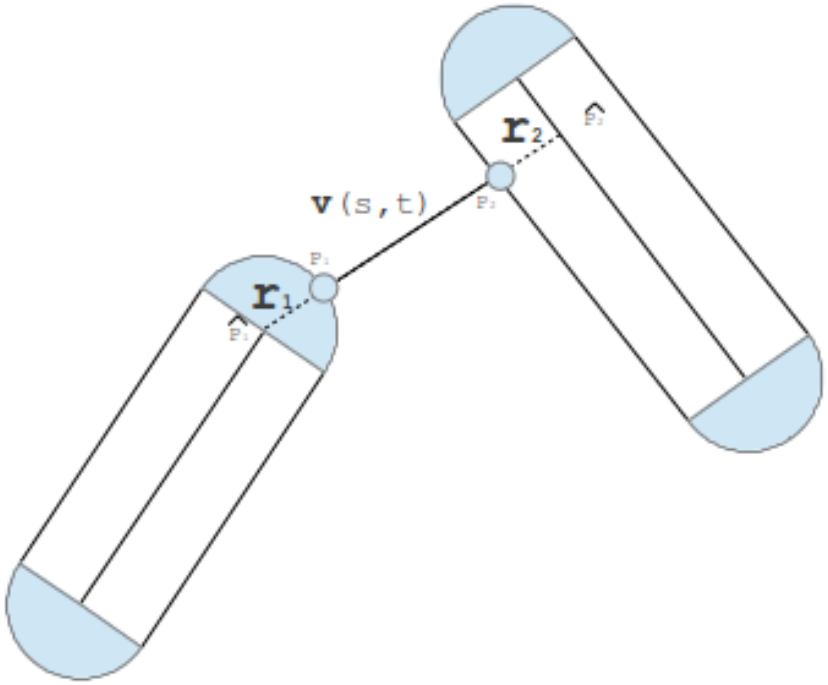
How do we find the closest point pair?

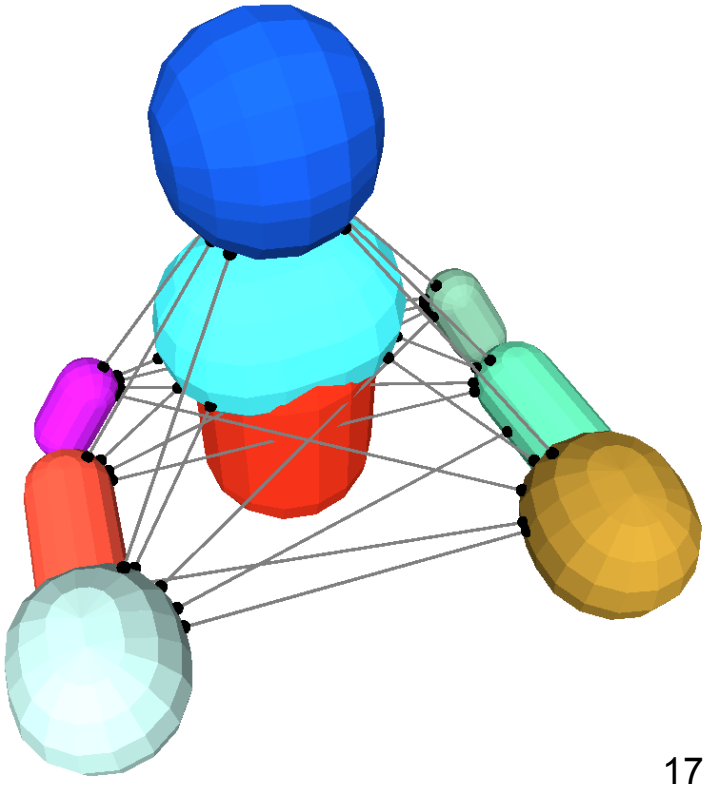


3. Collision Avoidance - Self

Velocity Damping Task

Capsule Decomposition !





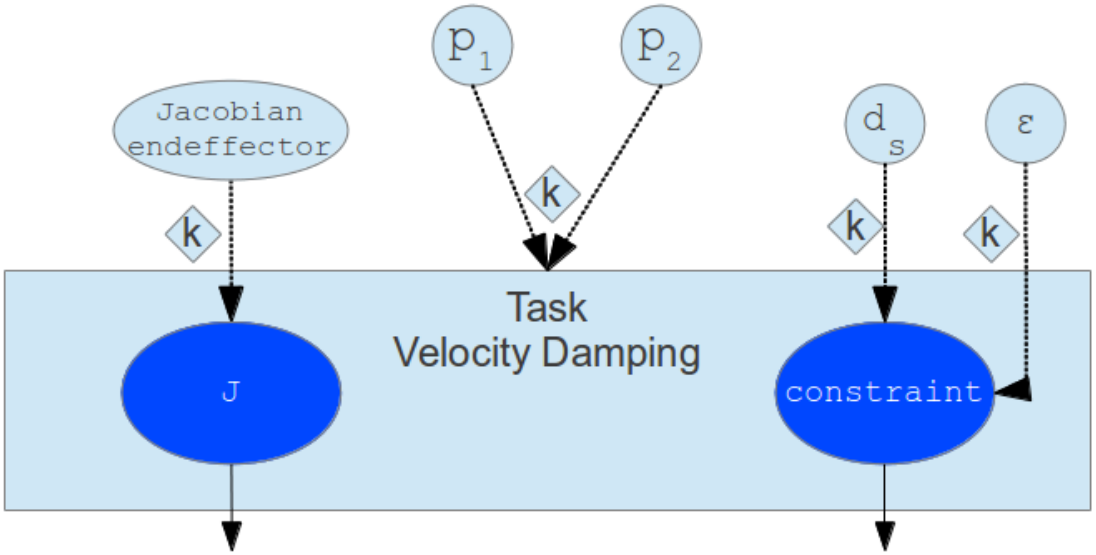
3. Collision Avoidance - Self

Velocity Damping Task

$$\mathbf{n_i}^T \mathbf{J}(\mathbf{p_{1,i}}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon_i \frac{d_i - d_{s,i}}{\Delta t} \quad \forall i \in \{1, \ldots, k\}$$

$$\mathbf{N}^T \mathbf{J}(\mathbf{p_1}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{\mathbf{d} - \mathbf{d_s}}{\Delta t}$$

Dynamic Graph Integration



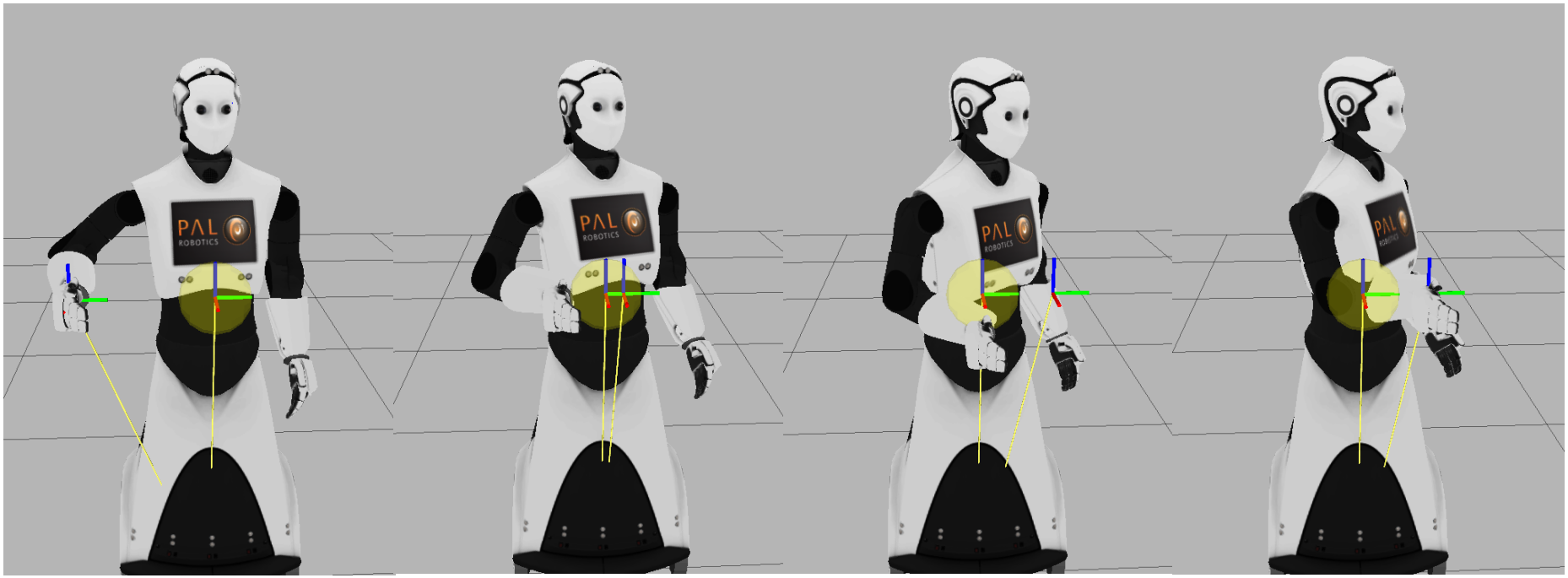
3. Collision Avoidance - Self

Self-Collision
Avoidance
Video

4. Experiments & Results

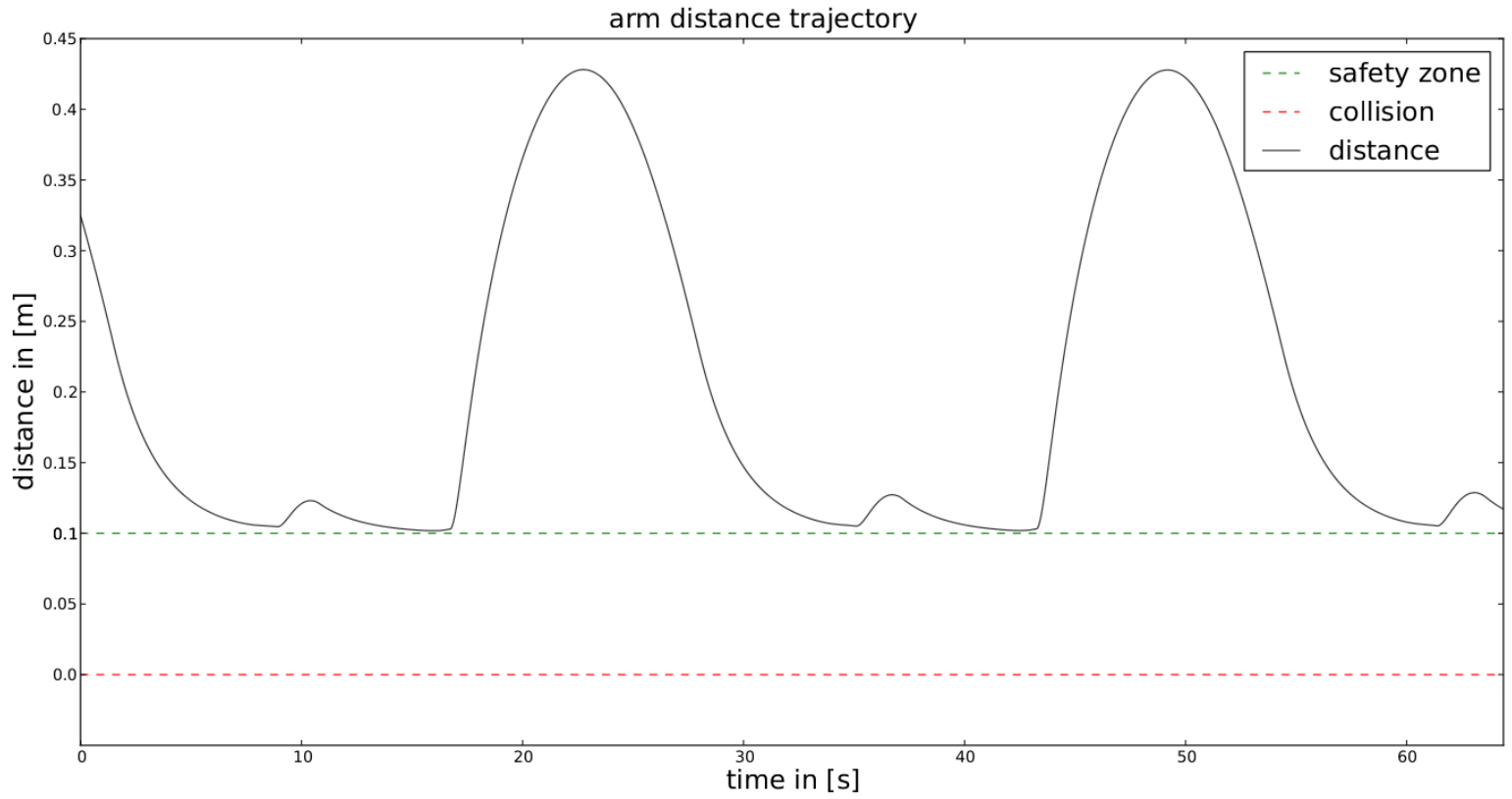
External Collision Object

→ Moving Arm, Static Object

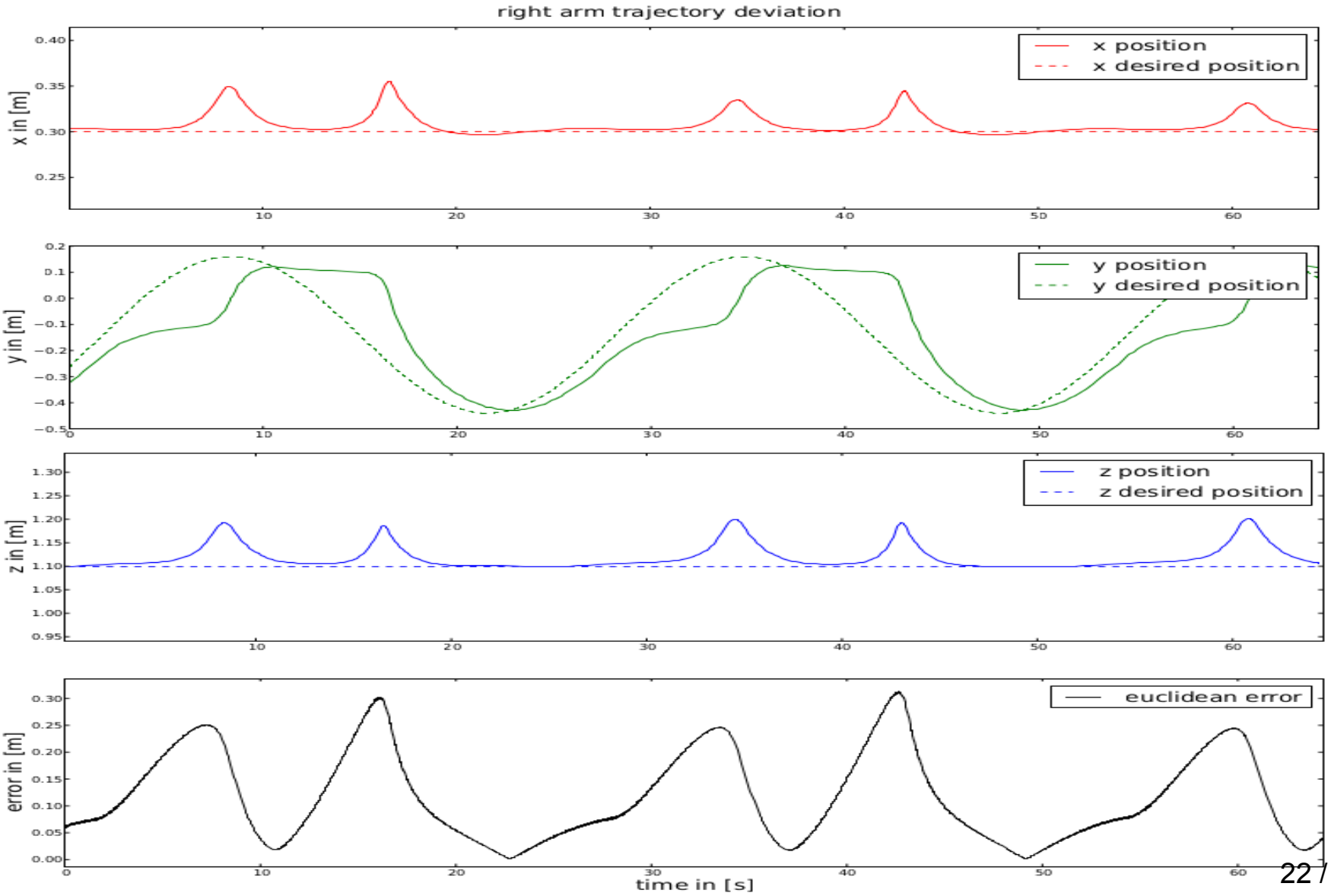


External Collision Object

→ Safety Distance does not get violated!

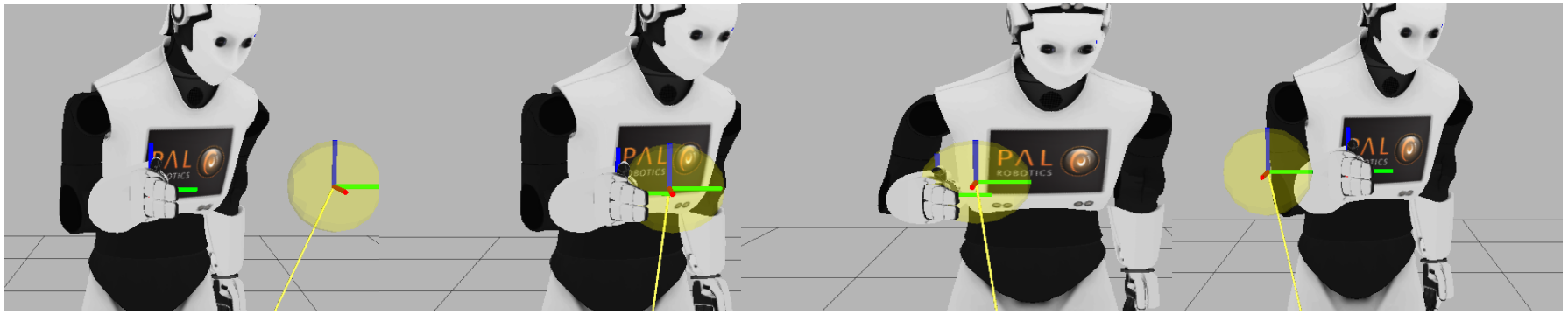


4. Experiments & Results



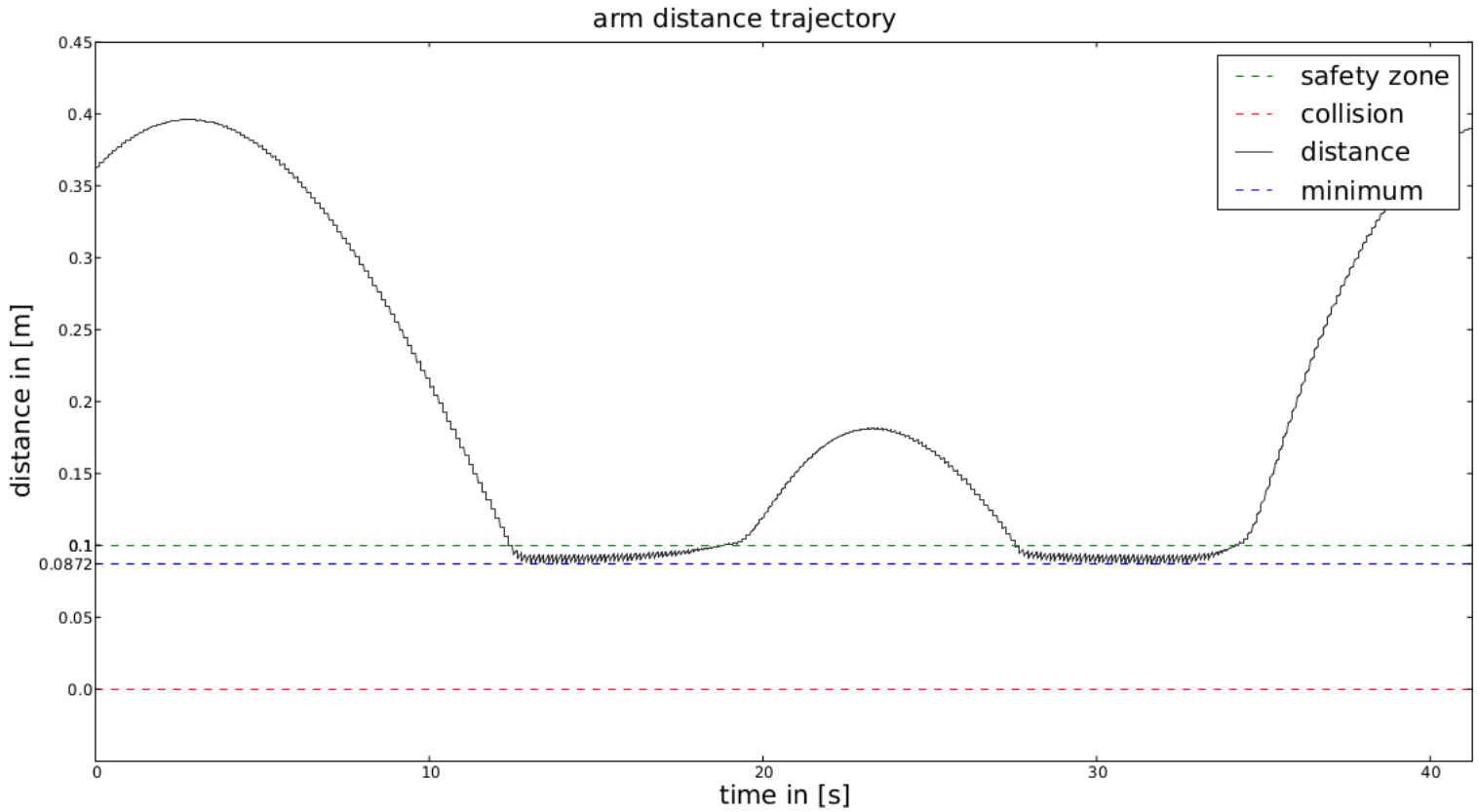
External Collision Object

→ Static Arm, Moving Obstacle



External Collision Object

→ Safety Distance is violated!

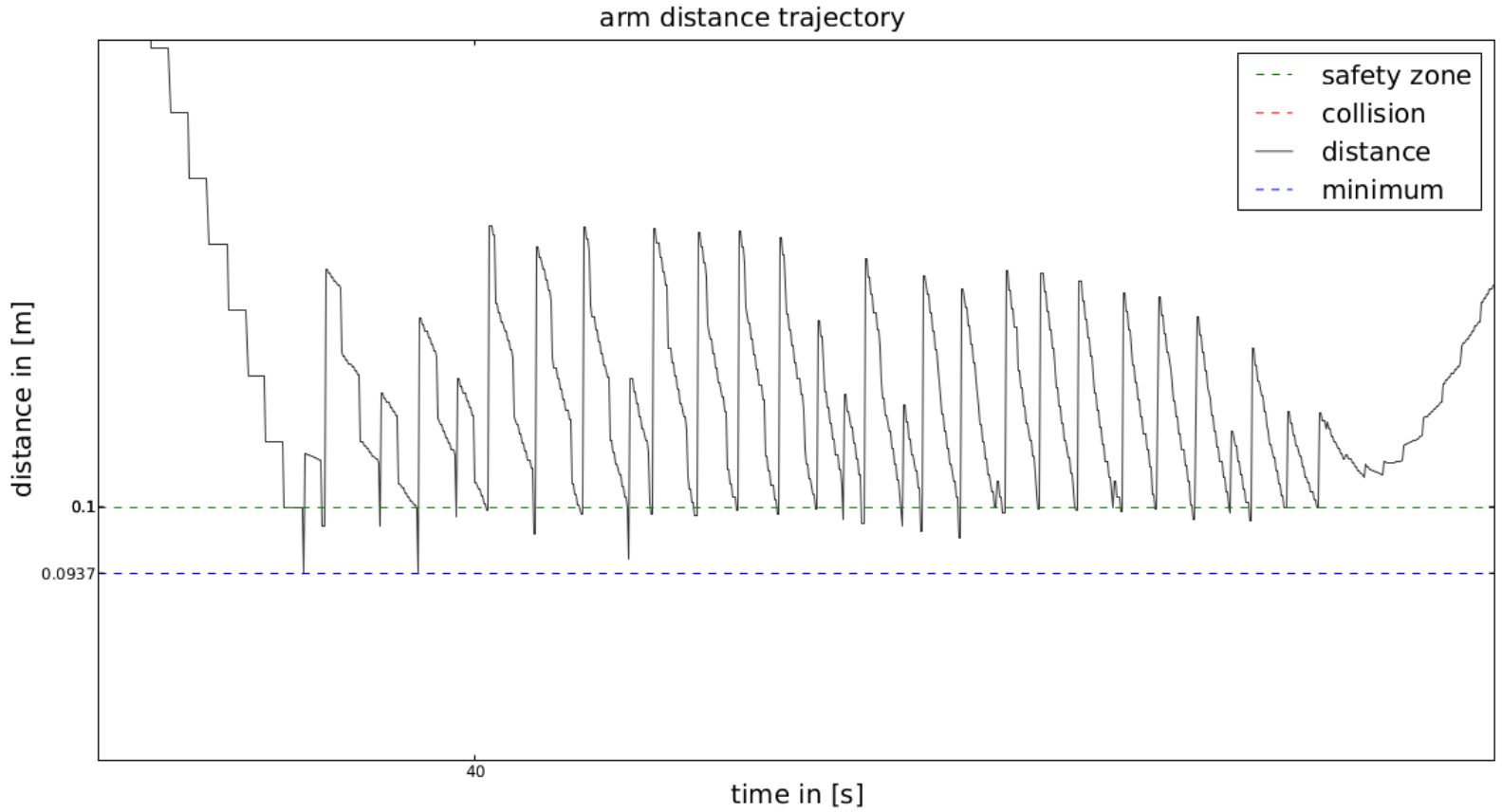


4. Experiments & Results

External Collision Object

$$\mathbf{n}^T \mathbf{J}(\mathbf{p_1}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{d - ds}{\Delta t}$$

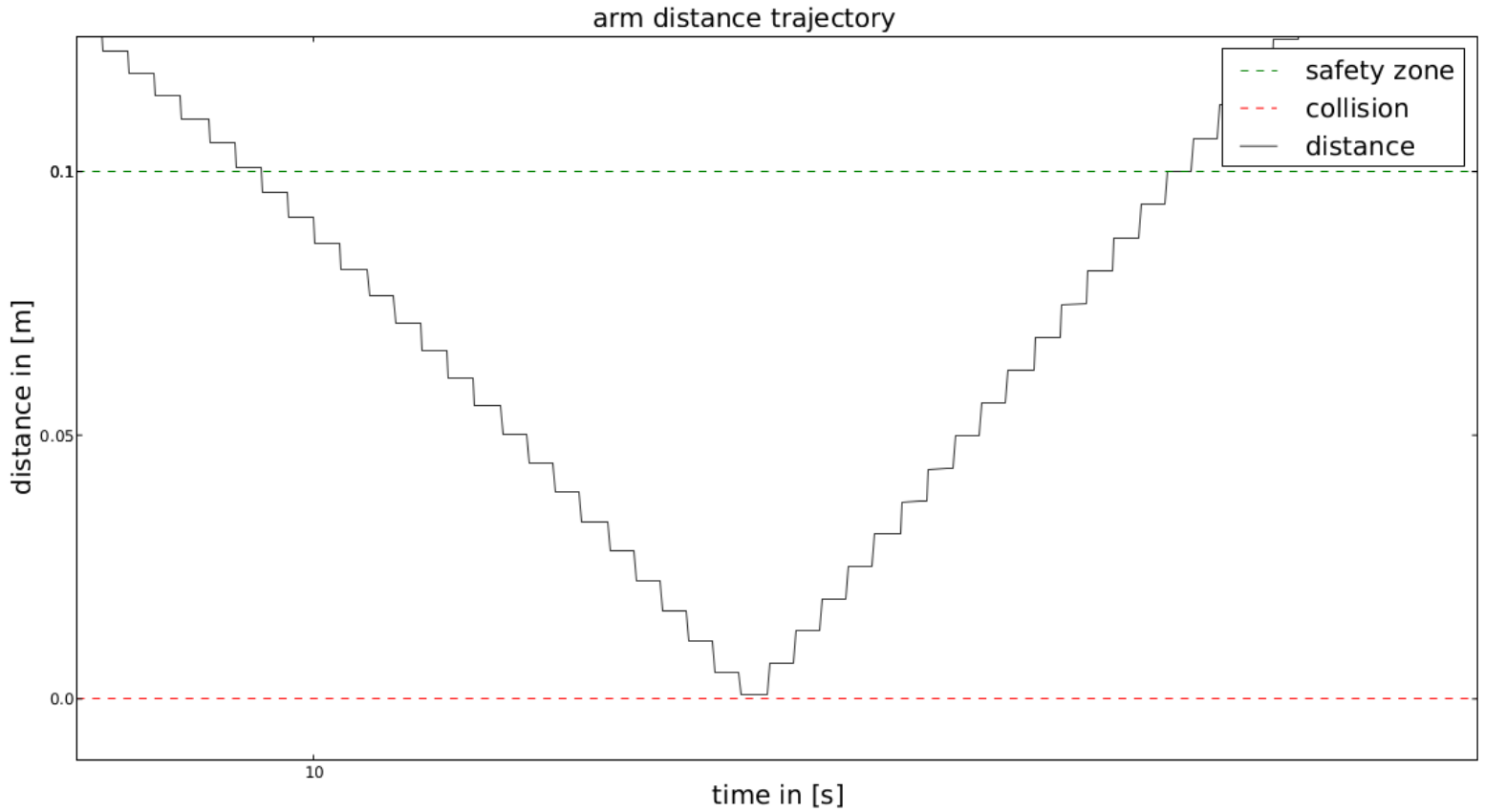
→ Gain tuning? ε = *10000*



External Collision Object

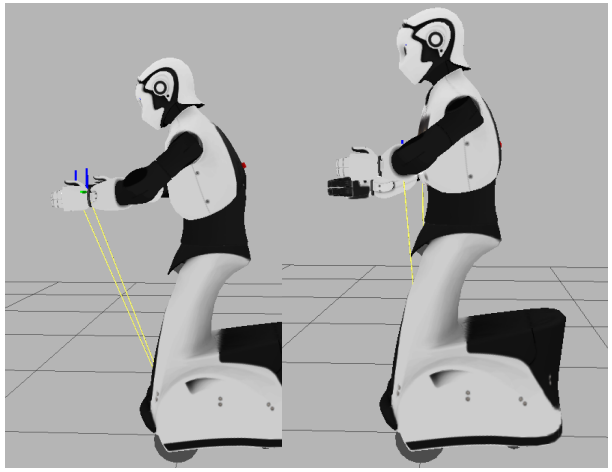
$$\mathbf{n}^T \mathbf{J}(\mathbf{p_1}, \mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{d - ds}{\Delta t}$$

→ Gain tuning? *ε = 0.01*

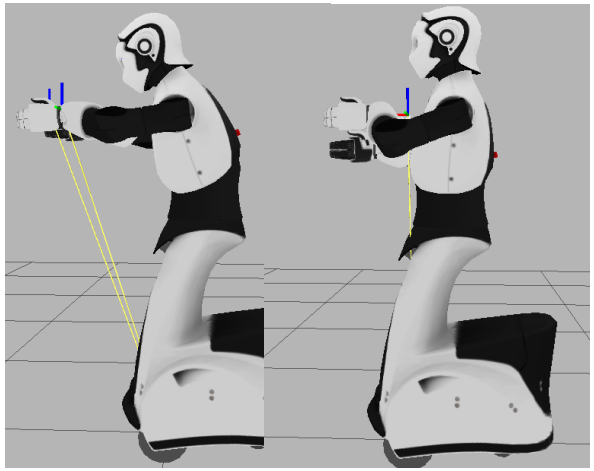


Self-Collision

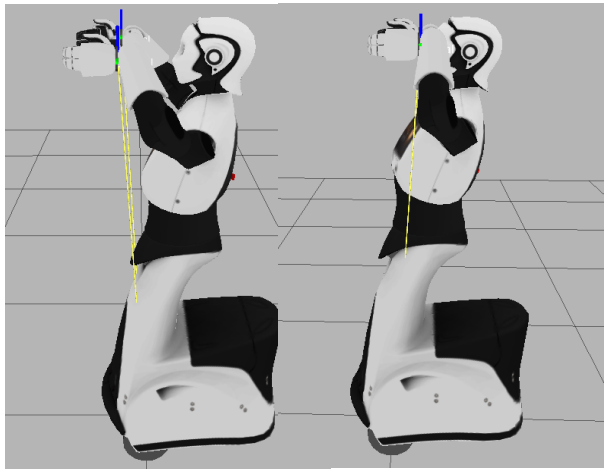
→ Full Collision Matrix *(everything against everything)*



Arms vs. Lower Torso



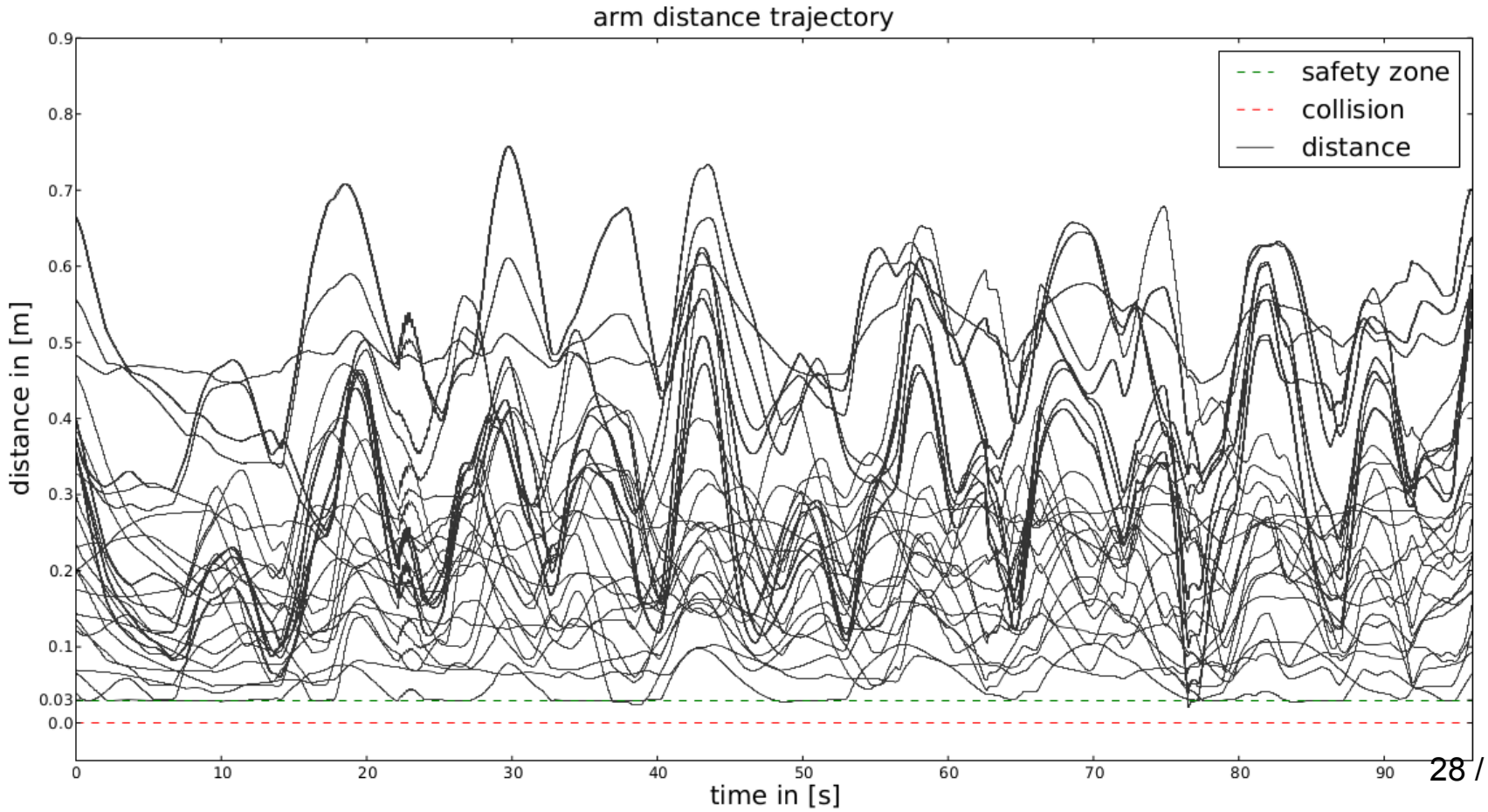
Arms vs. Upper Torso



Arms vs. Head

Self-Collision

→ Full Collision Matrix *(everything against everything)*



Inversion of Hierarchy

→ Why is the safety distance violated ?
→ Heavy oscillation between violation and overshoot !

$$\mathbf{n}^T \mathbf{J}(\mathbf{p},\mathbf{q})\dot{\mathbf{q}} \geq -\epsilon \frac{d - d_s}{\Delta t} \qquad for \quad d - d_s > 0$$

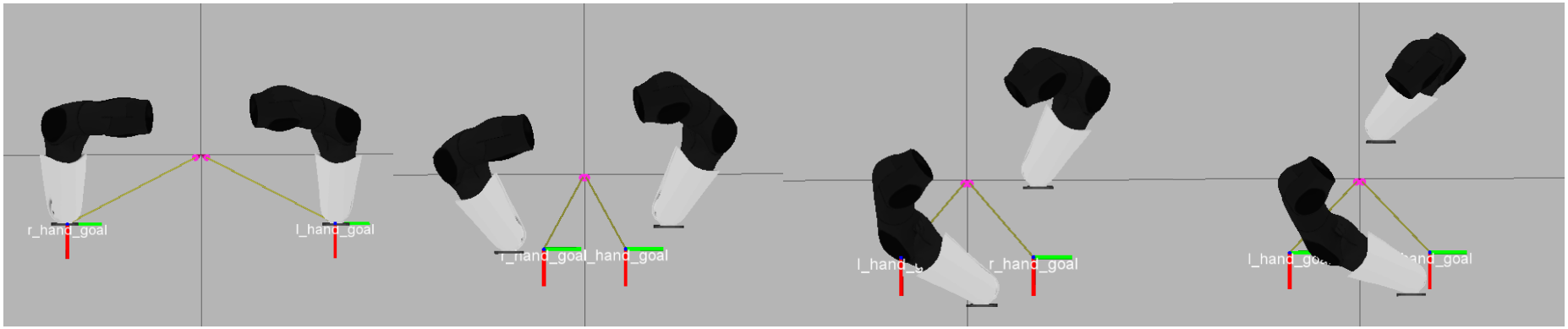
$$0 \geq -\epsilon \frac{d - ds}{\Delta t} \qquad for \quad d - d_s = 0$$

$$\mathbf{n}^T \mathbf{J}(\mathbf{p},\mathbf{q})\dot{\mathbf{q}} \leq \epsilon \frac{d - d_s}{\Delta t} \qquad for \quad d - d_s < 0$$

4. Experiments & Results

Inversion of Hierarchy cont.

→ Violation on zero velocity conditions
→ Lower priority tasks can produce clamping
  → zero velocity !
  → inversion of hierarchy !



Right arm has higher priority than left
→ left arm experiences clamping = zero velocity
→ right arm leaves trajectory = inversion

5. Conclusion & Outlook

Pro: Reliable tool for collision avoidance

- full self-collision prevention
- supports external obstacle avoidance (wrt. gain)
- dynamic graph allows minimal collision configuration
- multiple tasks in parallel (vision, both arms)
- fast and reactive (100Hz real time)

Cons: No look-ahead planning

- numerical solver instabilities
- no a priori path planning
- goal can be out of reach

5. Conclusion & Outlook

Outlook: More tasks in safe environment

- Integrate look-ahead component

- Visual Servoing
- Grasping (3D & 6D)
- Teleoperation
- Humanoid Walking

5. Conclusion & Outlook

Outlook
Video

THANK YOU FOR YOUR ATTENTION

Questions ?

PAL ROBOTICS