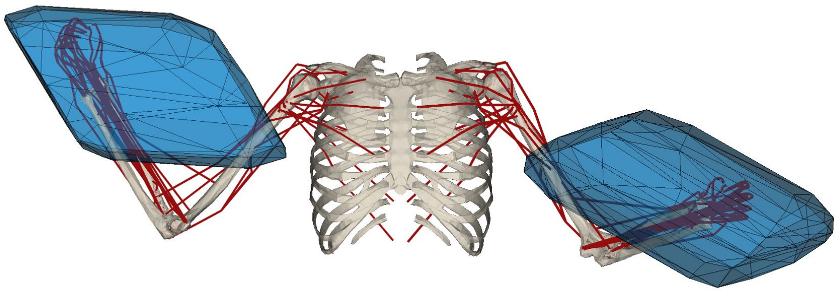
Efficient calculation of human wrench capacity based on human musculoskeletal models Application in collaborative robot control

Antun Skuric, Vincent Padois, Nasser Rezzoug, David Daney



Efficient calculation of human wrench capacity based on human musculoskeletal models

Application in collaborative robot control

Antun Skuric, Vincent Padois, Nasser Rezzoug, David Daney

Assistive robotics perspective

- Human-centered robot control
  - Robot constantly adapts to the ***need*** of the human counterpart
  - Example: Assist-as-needed paradigm
    - Exoskelts, Medical and collaborative robots

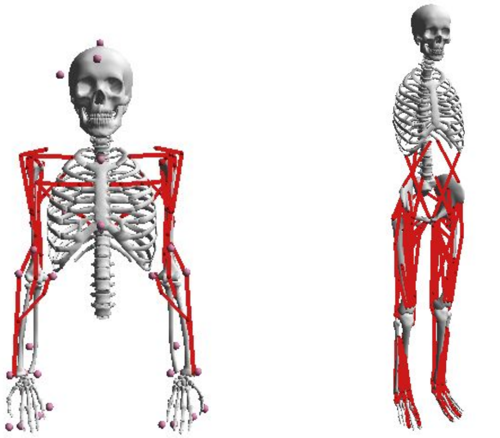
- Fundamental challenge/assumption
  Real-time knowledge about human performance and capacity metrics

Assistive robotics perspective

- Human-centered robot control
  - Robot constantly adapts to the ***need*** of the human counterpart
  - Example: Assist-as-needed paradigm
    - Exoskelts, Medical and collaborative robots

- Fundamental challenge/assumption
  Real-time knowledge about human performance and capacity metrics

- Musculoskeletal models
  - Most complete human body models
  - Mapping muscular dynamics to human body dynamics



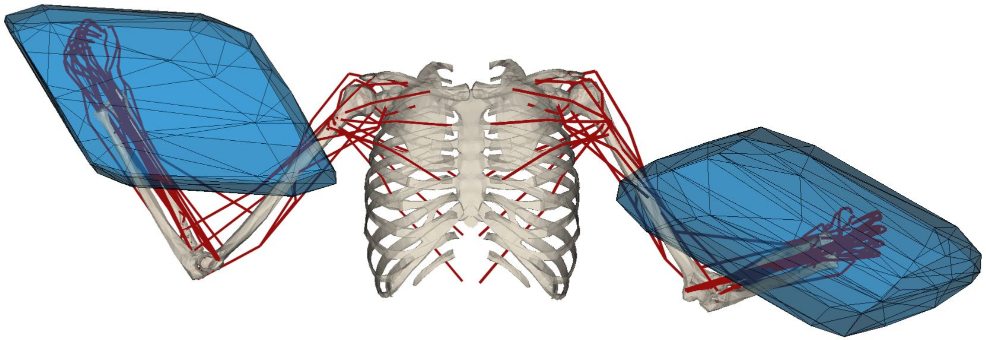
Wrench capacity of human musculoskeletal models

- *“Ability to resist and apply forces and moments in arbitrary directions.”*
  - Important for performance and safety analysis

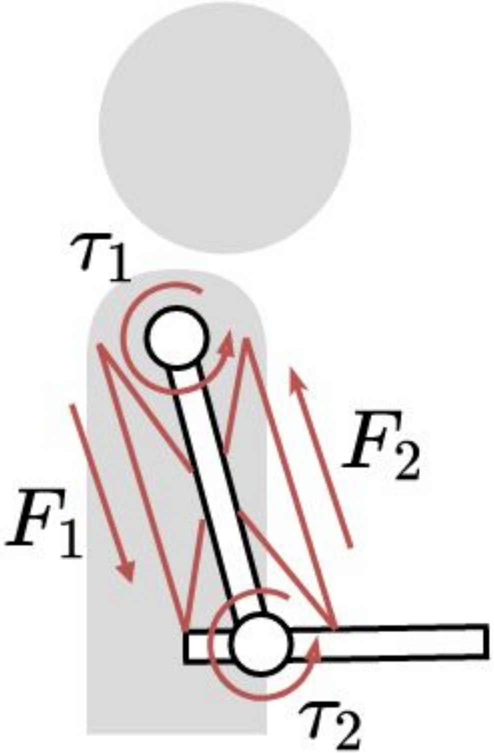
- Ellipsoids
  - Efficient to calculate - online capable
  - Approximative measure

- ***Polytopes***
  - Exact solution
  - More complete metric
  - Computationally intensive

    So far not real-time capable



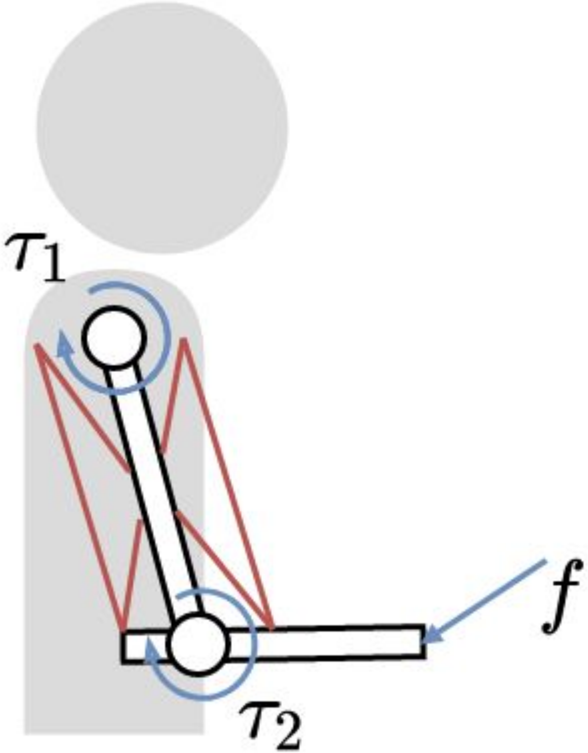
Problem geometry: 6 muscle 2 dof planar example



$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad F = \begin{bmatrix} F_1 \\ F_2 \\ \cdots \end{bmatrix} \qquad F \in [\underline{F},\ \overline{F}]$$

$$\tau = N(q)F$$

Problem geometry: 6 muscle 2 dof planar example



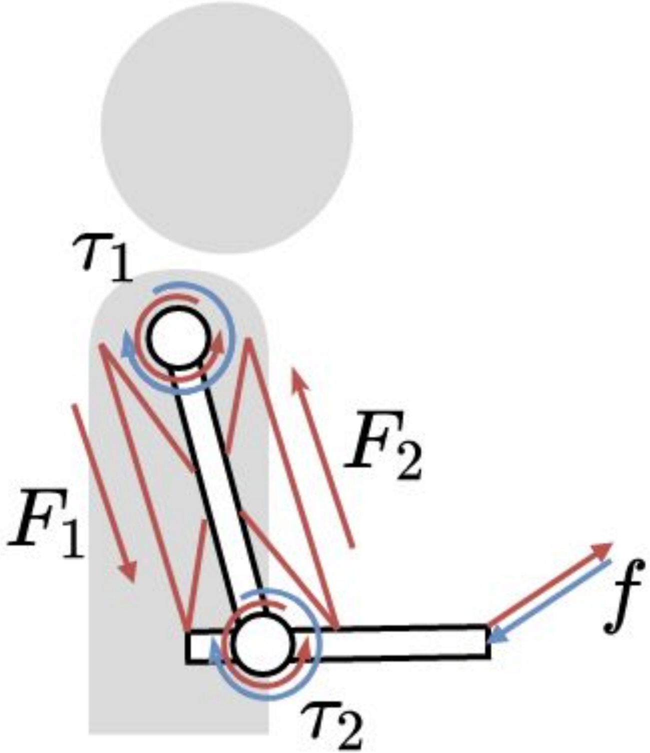
$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \dots \end{bmatrix} \quad F \in [\underline{F},\ \overline{F}]$$

$$\tau = N(q)F$$

$$\tau = J^T(q)f$$

Moment arm and Jacobian matrix
Calculated using: OpenSim, Pyomeca, ...

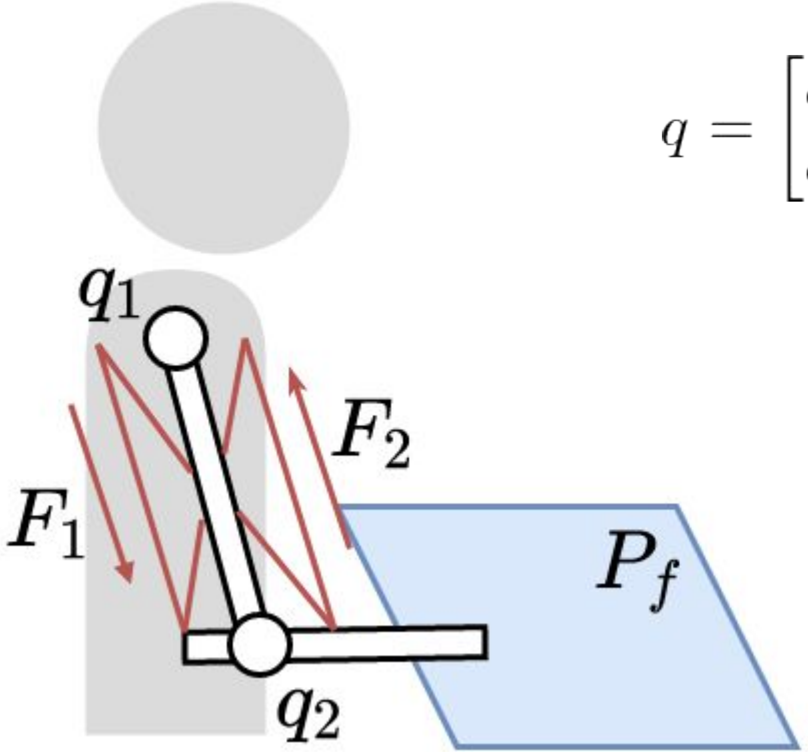
Problem geometry: 6 muscle 2 dof planar example



$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad F = \begin{bmatrix} F_1 \\ F_2 \\ \dots \end{bmatrix} \quad F \in [\underline{F},\ \overline{F}]$$

$$J^T(q)f = N(q)F$$

Problem geometry: 6 muscle 2 dof planar example



$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \qquad F = \begin{bmatrix} F_1 \\ F_2 \\ \cdots \end{bmatrix} \qquad F \in [\underline{F},\ \overline{F}]$$

$$J^T(q)f = N(q)F$$

$$P_f = \{f \in R^m \,|\, J^T f = NF,\ F \in [\underline{F}, \overline{F}]\}$$

Wrench polytope calculation challenges

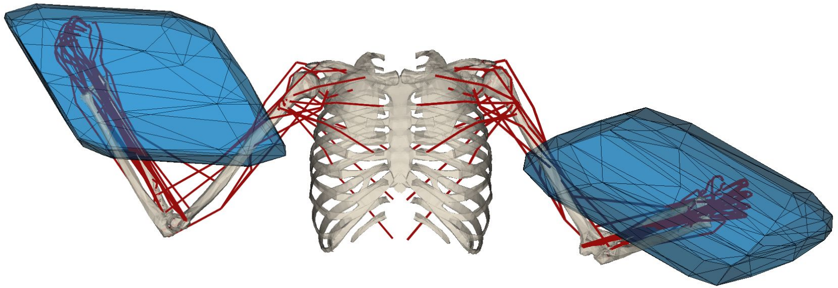
$$J^T(q)f = N(q)F \qquad \underline{F} \leq F \leq \overline{F}$$

Class of problems: $$Ax = By \qquad \underline{y} \leq y \leq \overline{y}$$

Standard algorithms challenges:

- Implicit polytope definition
  - Combination of multiple methods necessary $Ax \leq b$ and $y = Ax$
- Exponential execution time with number of muscles
  - Number of muscles is usually high (ex. human arm 20-50 muscles)
- Exact polytope often impractical
  - Very high number of faces and vertices

Proposed Algorithm



- Using implicit definition directly

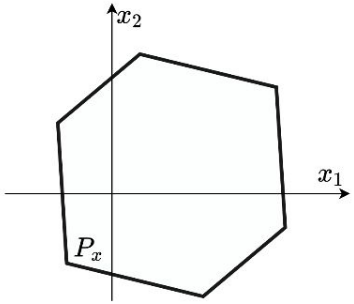
$$Ax = By \qquad \underline{y} \leq y \leq \overline{y}$$

- Enables polytope approximation with user defined precision **ε**
- Near-linear execution time
  - With respect to the number of muscles
  - With respect to the precision **ε**
- Enables intuitive performance vs precision trade-off - *real-time capable*
  - Execution time: under 200ms for 50 muscle 7 dof human arm, precision **ε** = 10N
  - Execution time: under 100ms for 32 muscle 7 dof human arm, precision **ε** = 10N

Proposed Algorithm

- Iterative method
  - *Linear programming* to find polytope vertices
  - *Convex hull algorithm* to group them
- Successively augmenting the precision of the approximation

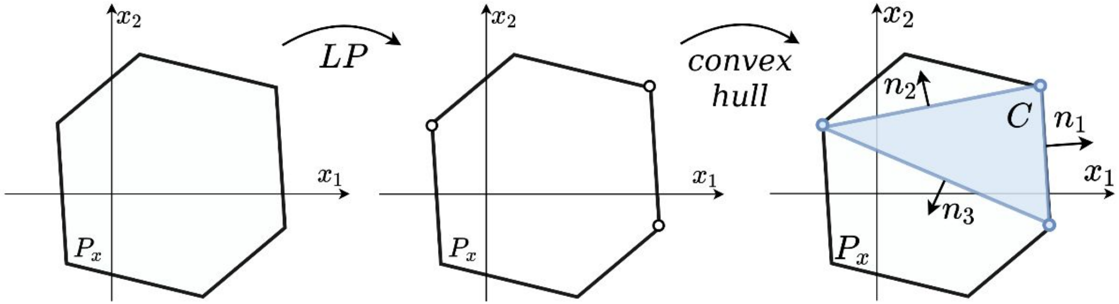
Geometric interpretation:



Proposed Algorithm

- Iterative method
  - *Linear programming* to find polytope vertices
  - *Convex hull algorithm* to group them
- Successively augmenting the precision of the approximation

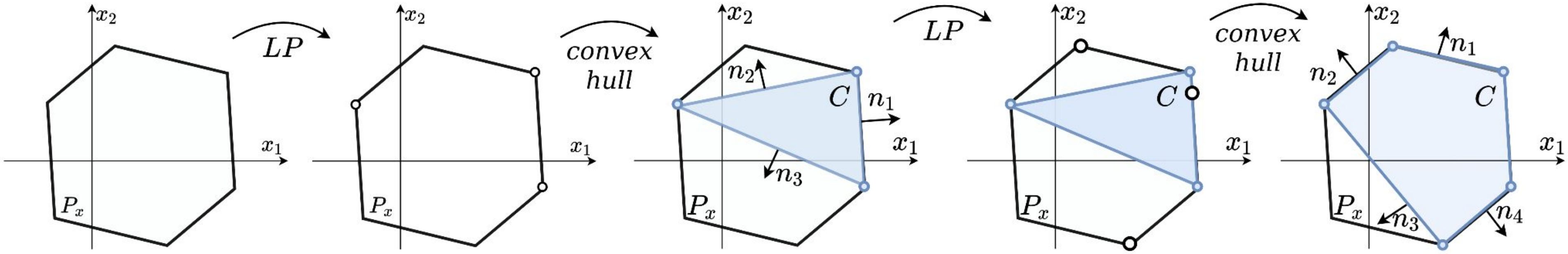
Geometric interpretation:



Proposed Algorithm

- Iterative method
  - *Linear programming* to find polytope vertices
  - *Convex hull algorithm* to group them
- Successively augmenting the precision of the approximation

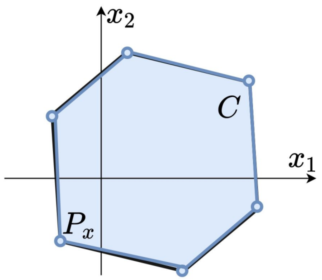
Geometric interpretation:



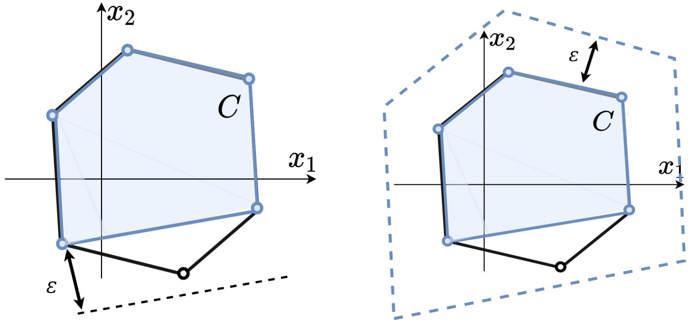
Proposed Algorithm

- Iterative method
  - *Linear programming* to find polytope vertices
  - *Convex hull algorithm* to group them
- Successively augmenting the precision of the approximation

Stopping condition:

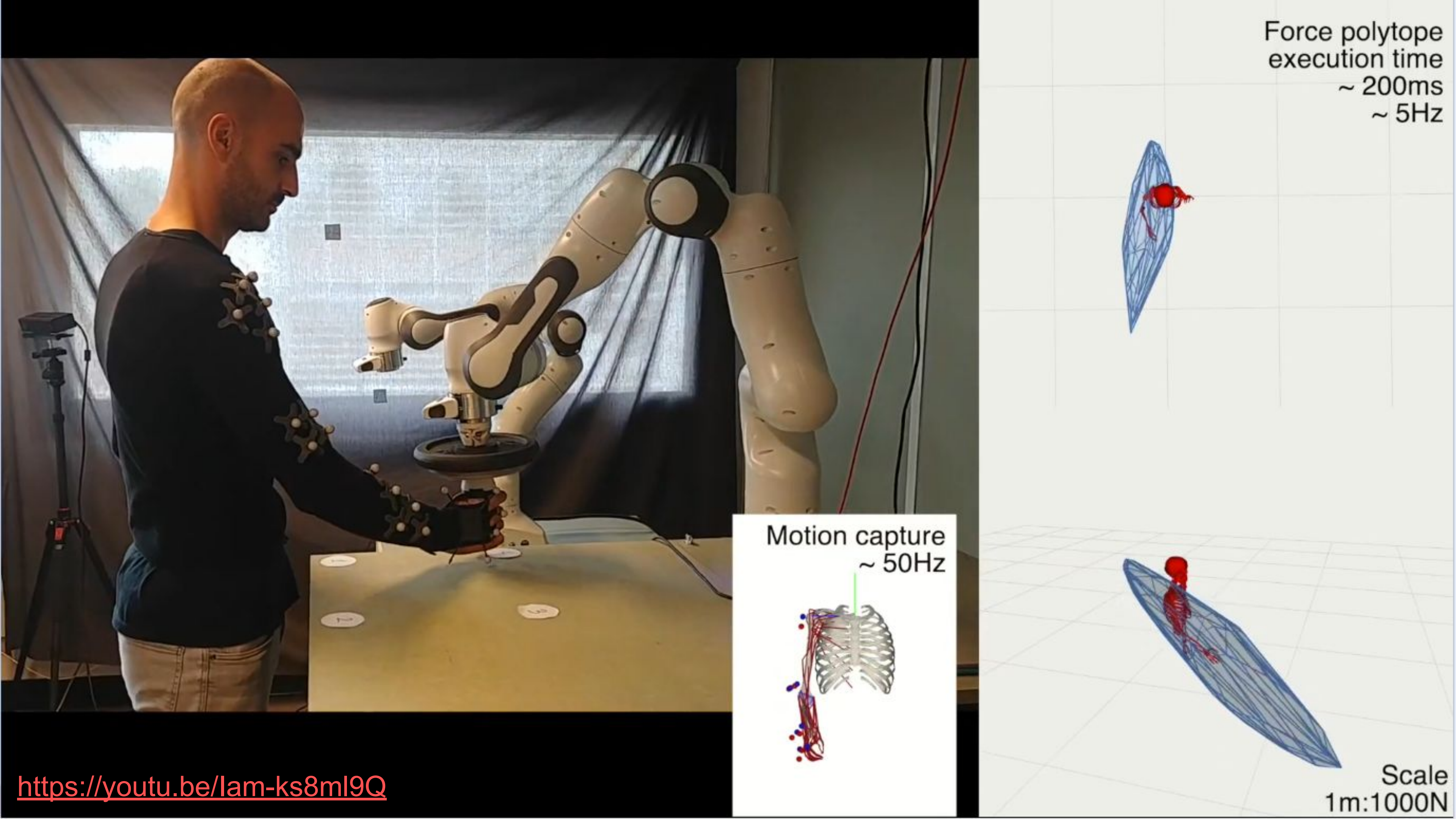


Exact solution found

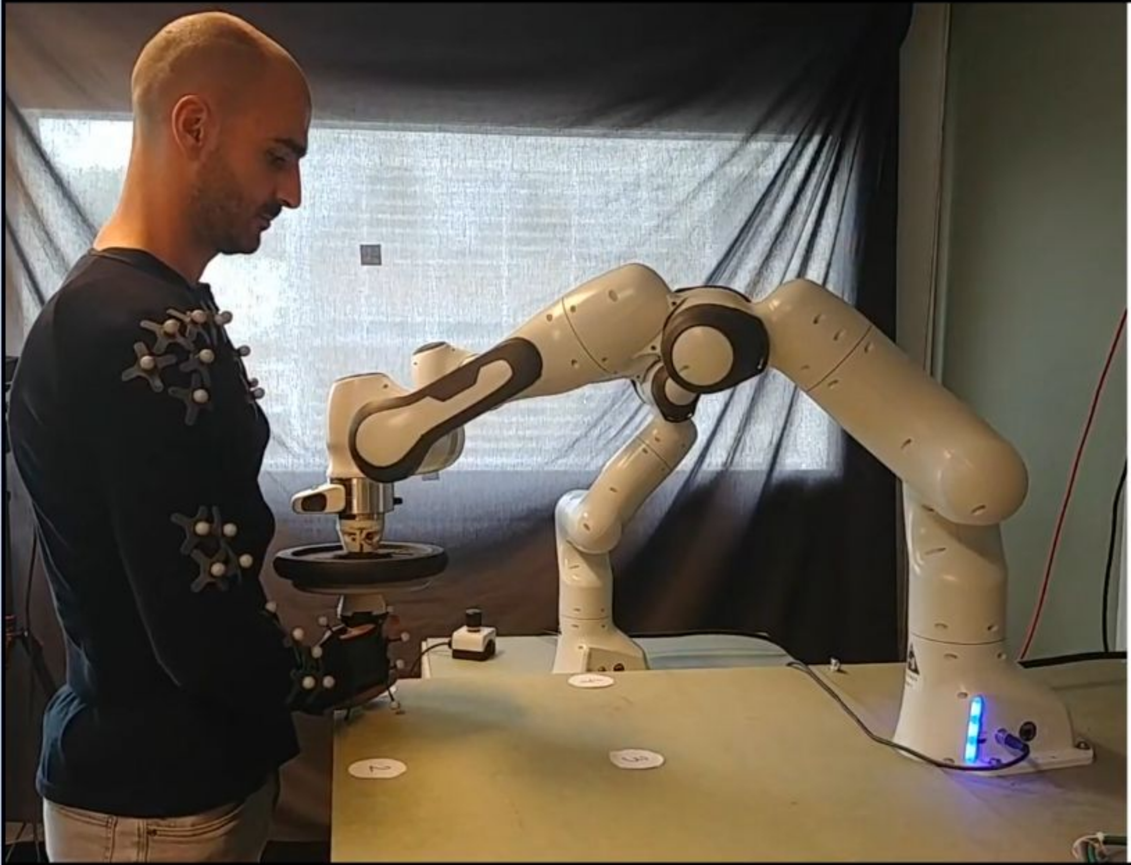


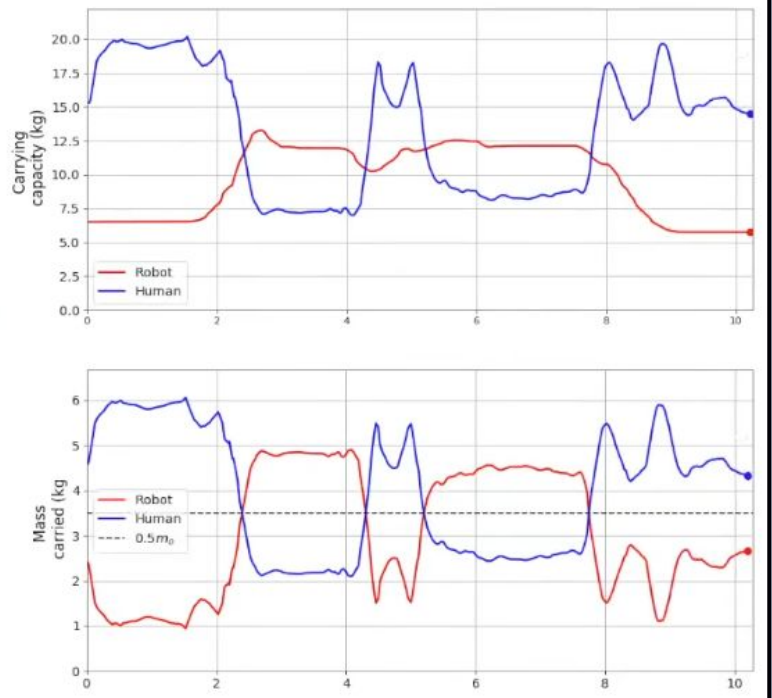
Precision ε satisfied

Experiment
Collaborative carrying task



https://youtu.be/Iam-ks8ml9Q





https://youtu.be/URxJO2FHnKY

Conclusion

- Novel polytope evaluation algorithm for class of problems $Ax = By$
- Efficient human wrench polytope calculation
    - Enables polytope approximation with user defined precision **ε**
    - Can be used for real-time robot control - *example*: *Collaborative carrying task*

Email: *antun.skuric@inria.fr*
GitLab: *https://gitlab.inria.fr/askuric/human_wrench_capacity*
HAL: *https://hal.inria.fr/hal-03369576*

This work has been funded by the BPI France Lichie project