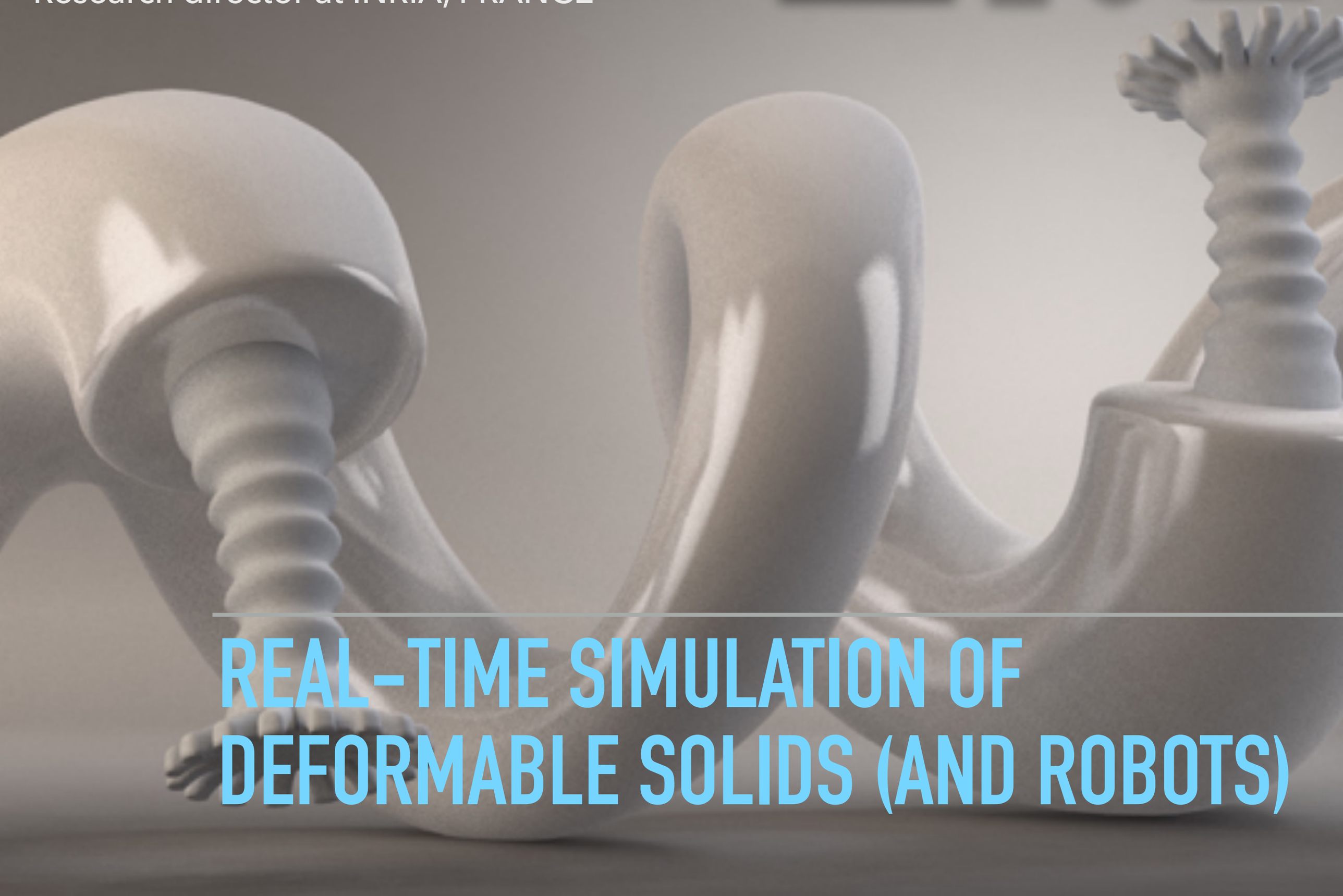
Christian Duriez, head of DEFROST team
Research director at INRIA, FRANCE



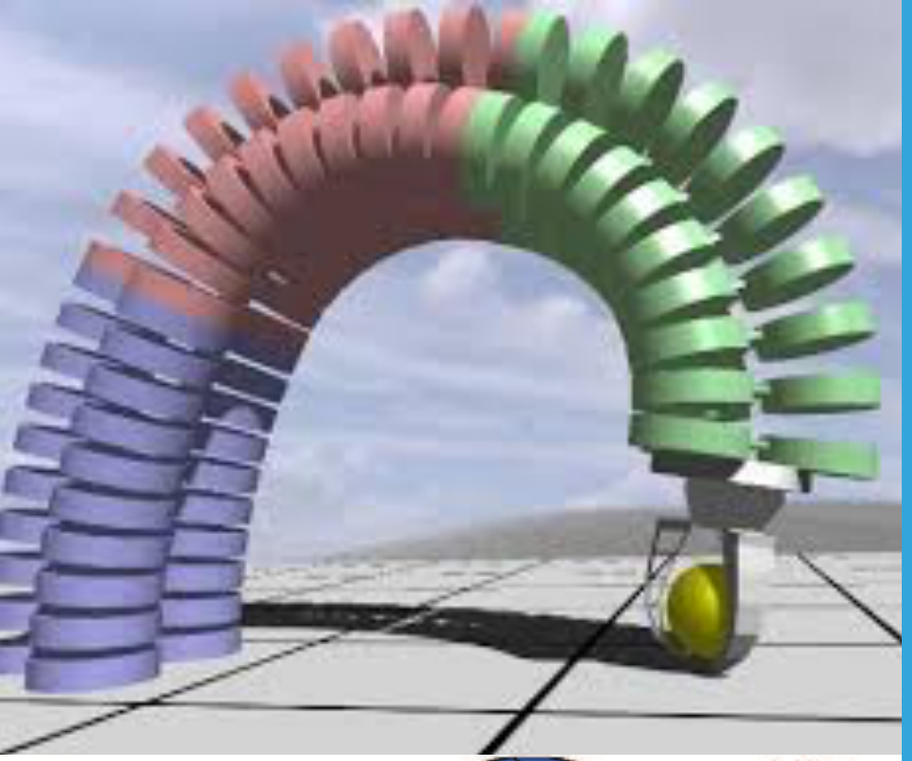
REAL-TIME SIMULATION OF DEFORMABLE SOLIDS (AND ROBOTS)

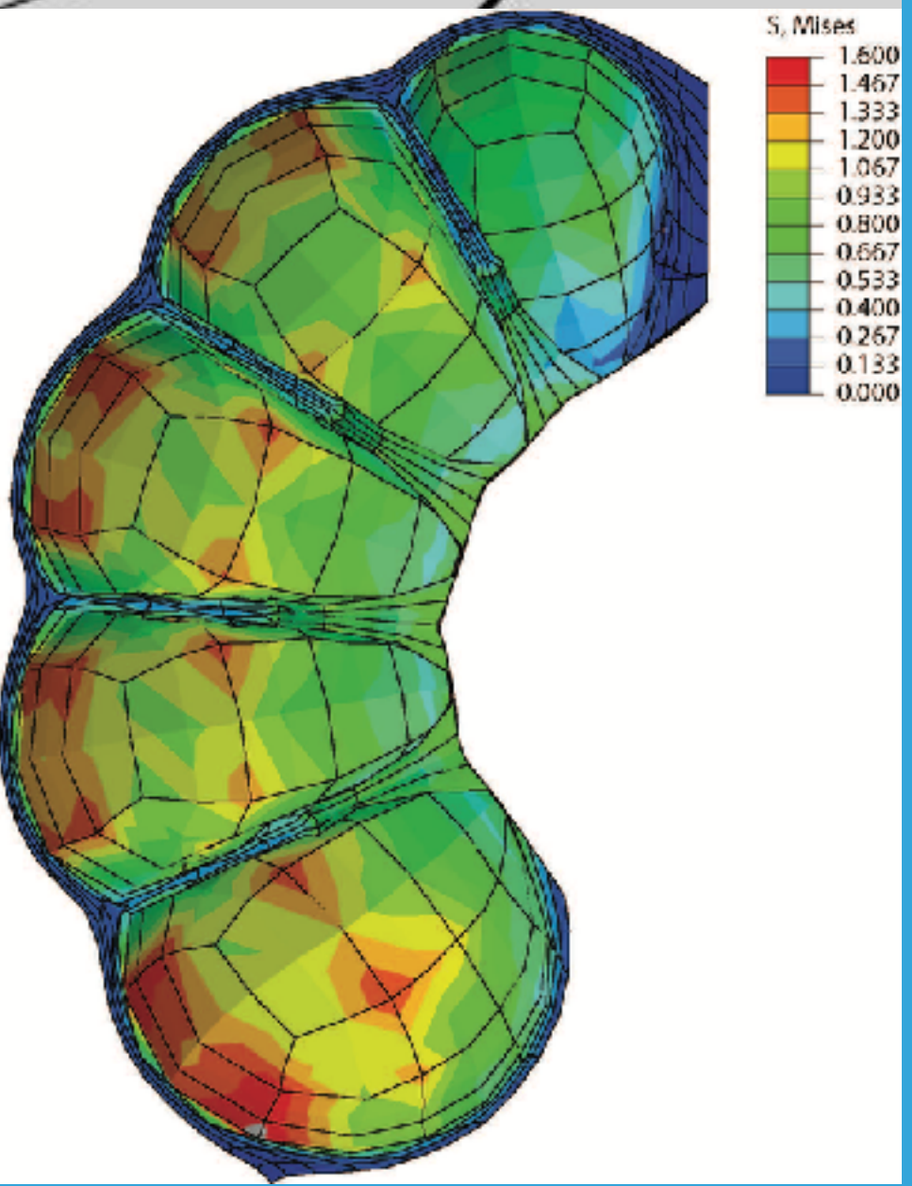
SUMMARY

- Context and motivations
- Mechanical models for real-time computation
- Collision detection and response
- Applications: surgical training and robotics

SUMMARY

- Context and motivations
- Mechanical models for real-time computation
- Collision detection and response
- Applications: surgical training and robotics



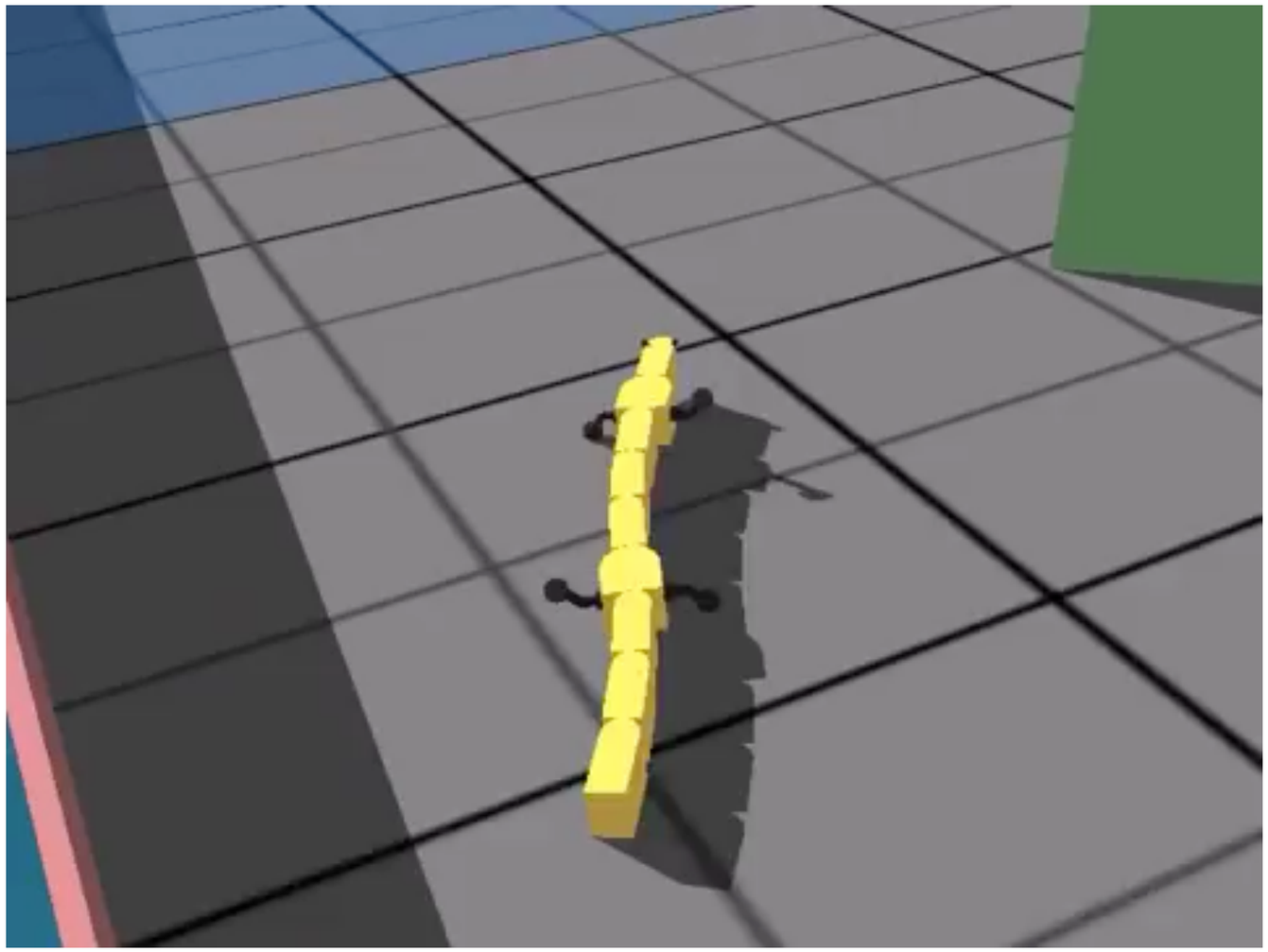


CONTEXT & MOTIVATIONS

HOW SIMULATION IS USEFUL IN ROBOTICS ?

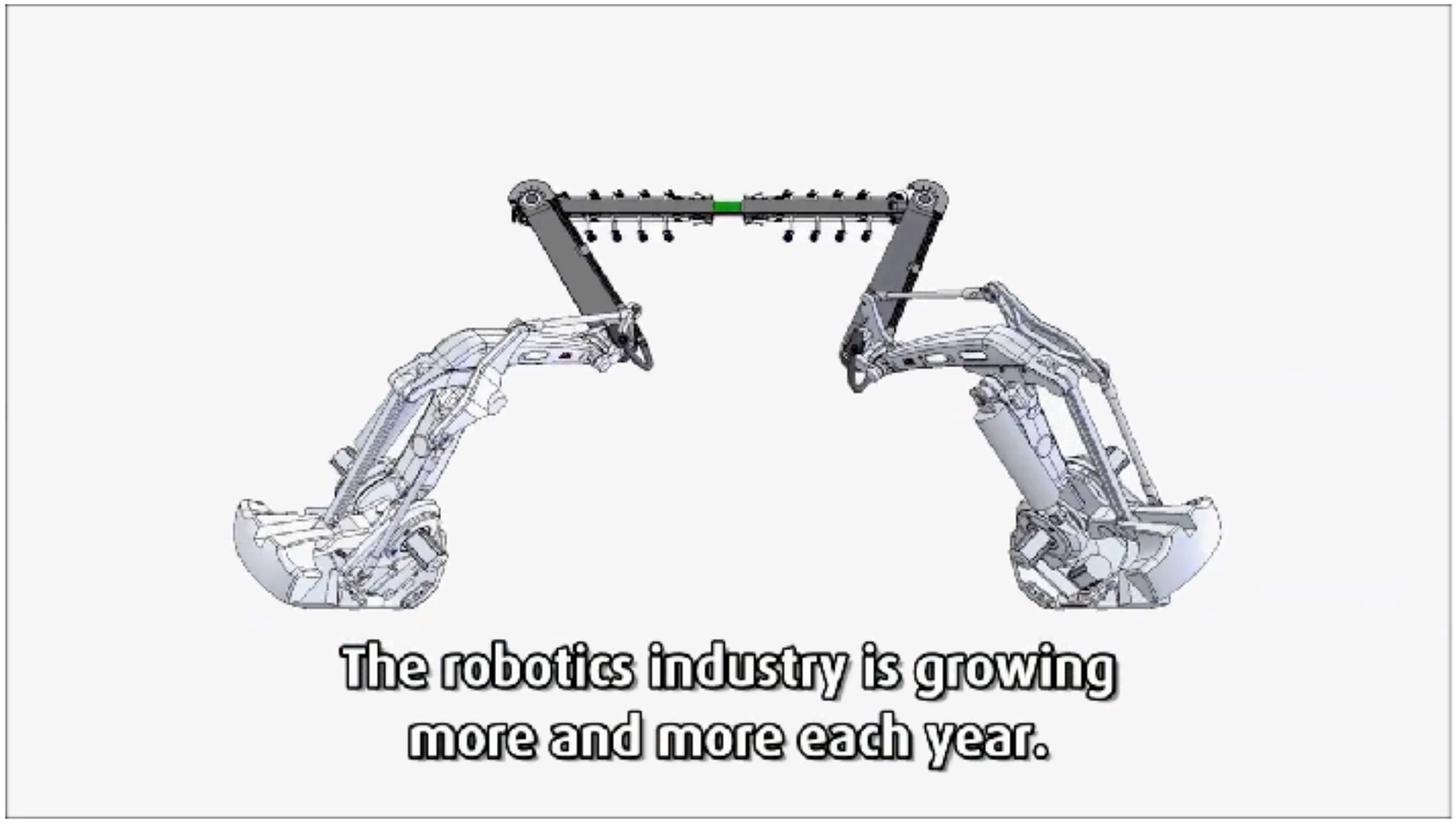
HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 1. Accelerated and safe design environment



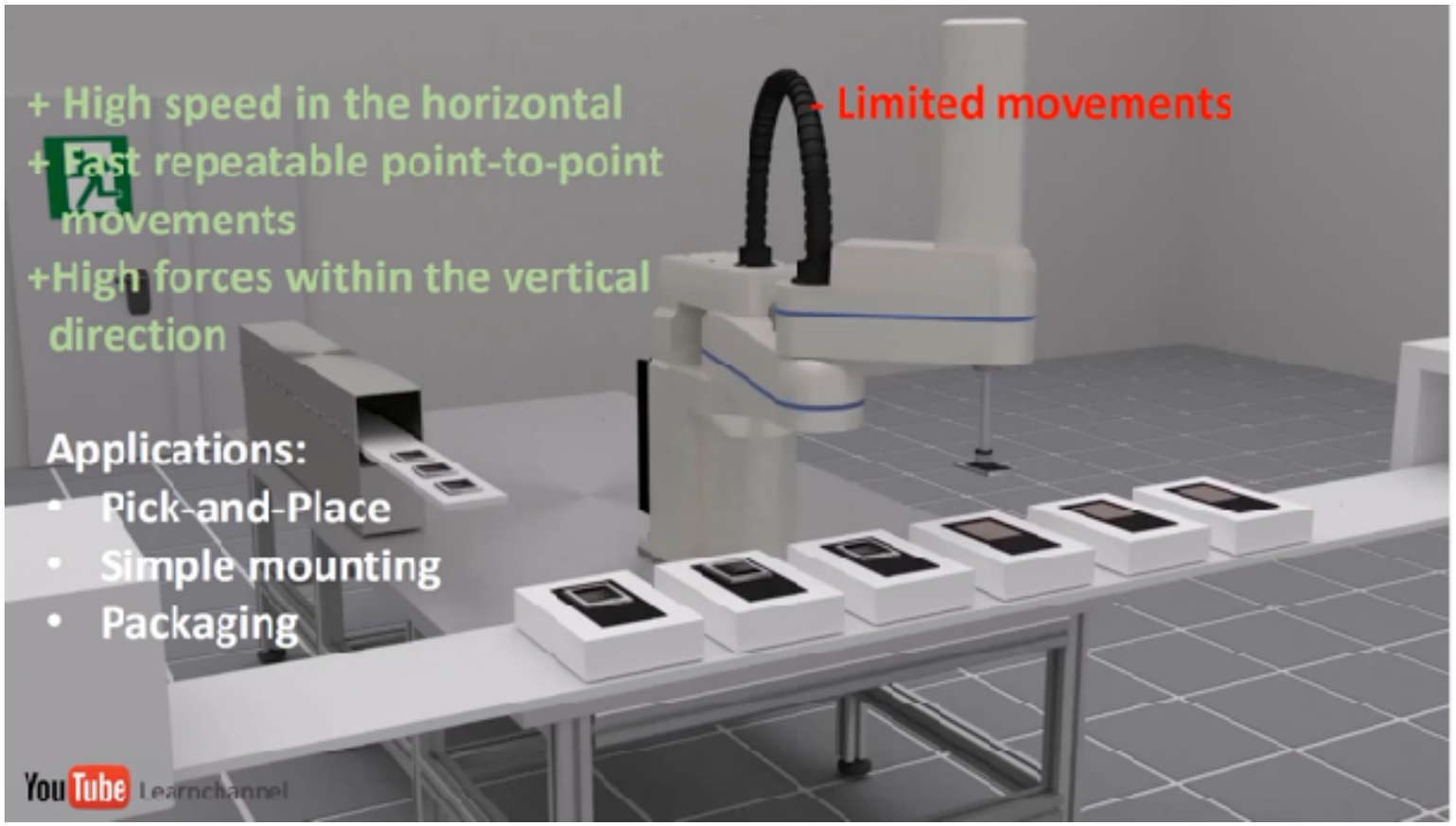
HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 2. Controlling solution costs & durability



HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 3. Accelerated and safe verification environment



HOW SIMULATION IS USEFUL IN ROBOTICS ?

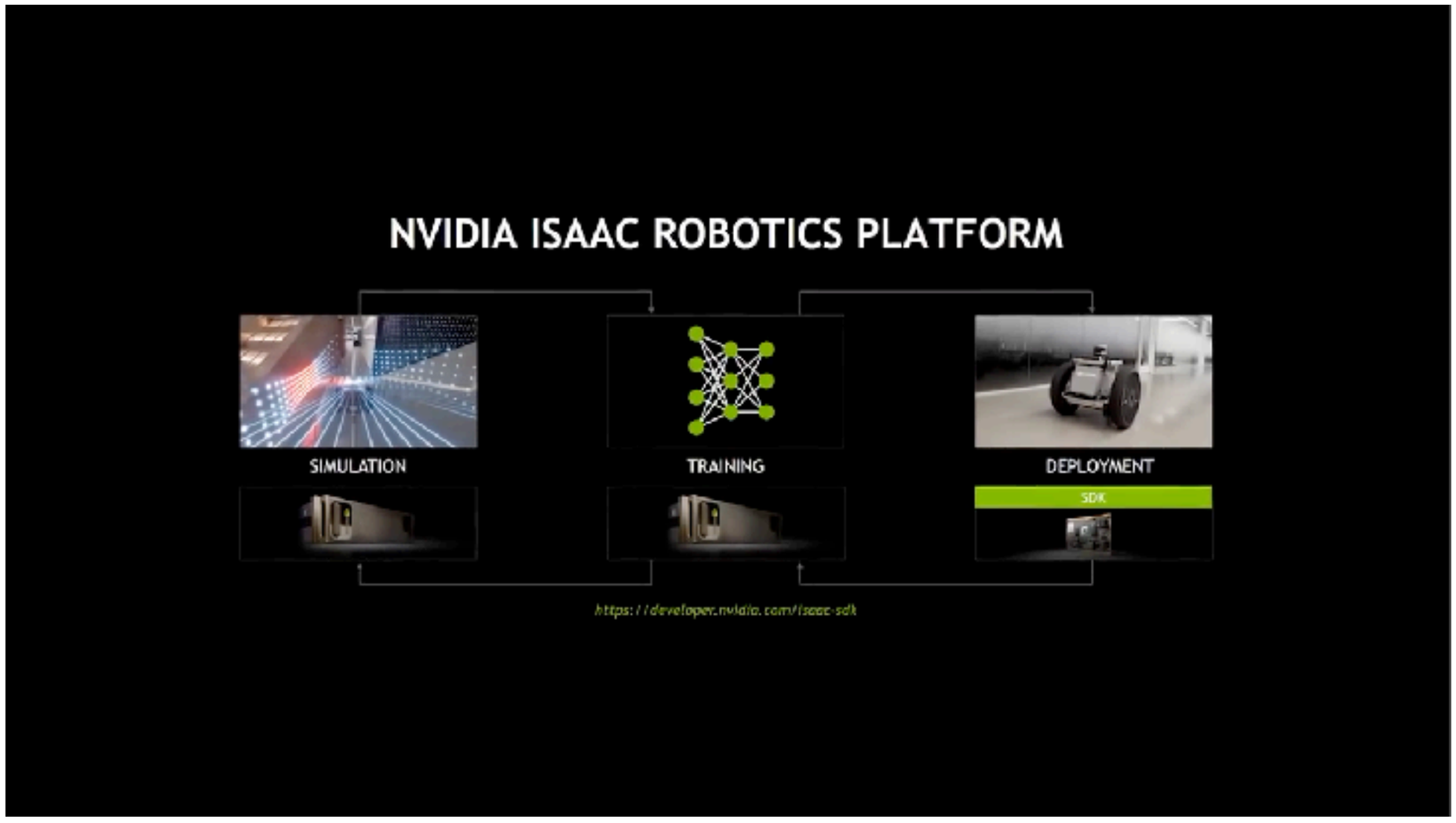
- 4. Generate large amounts of data for Machine Learning

HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 4. Generate large amounts of data for Machine Learning

HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 5. Making decisions

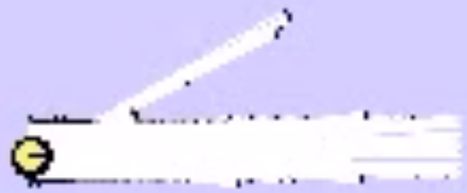


HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 5. Making decisions

HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 6. Multi-robot, collaborative scenarios



HOW SIMULATION IS USEFUL IN ROBOTICS ?

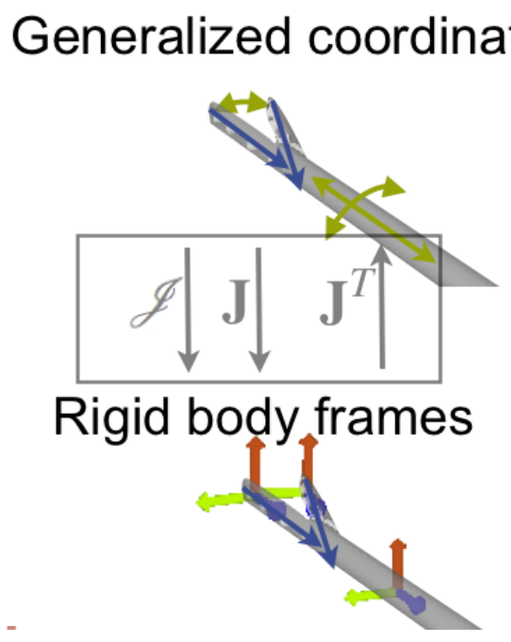
- 7. Facilitate human-robot interaction

HOW SIMULATION IS USEFUL IN ROBOTICS ?

- 7. Facilitate human-robot interaction

CHALLENGE: RIGID VS. DEFORMABLE MECHANICS

- Articulated Rigid Models
  - Geometry based model
  - Kinematics (Jacobian)



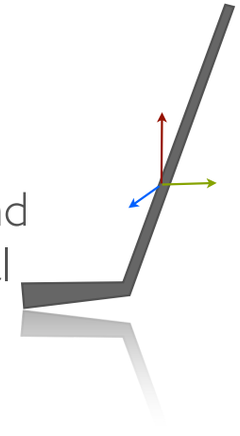
- Dynamics

$$\mathbb{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbb{P}(t) - \mathbb{F}(\mathbf{q}, \mathbf{v})$$

Mass, Inertia

Gravity

Coriolis and centrifugal Forces



- Deformable models
  - Kinematic behavior depends on material properties…

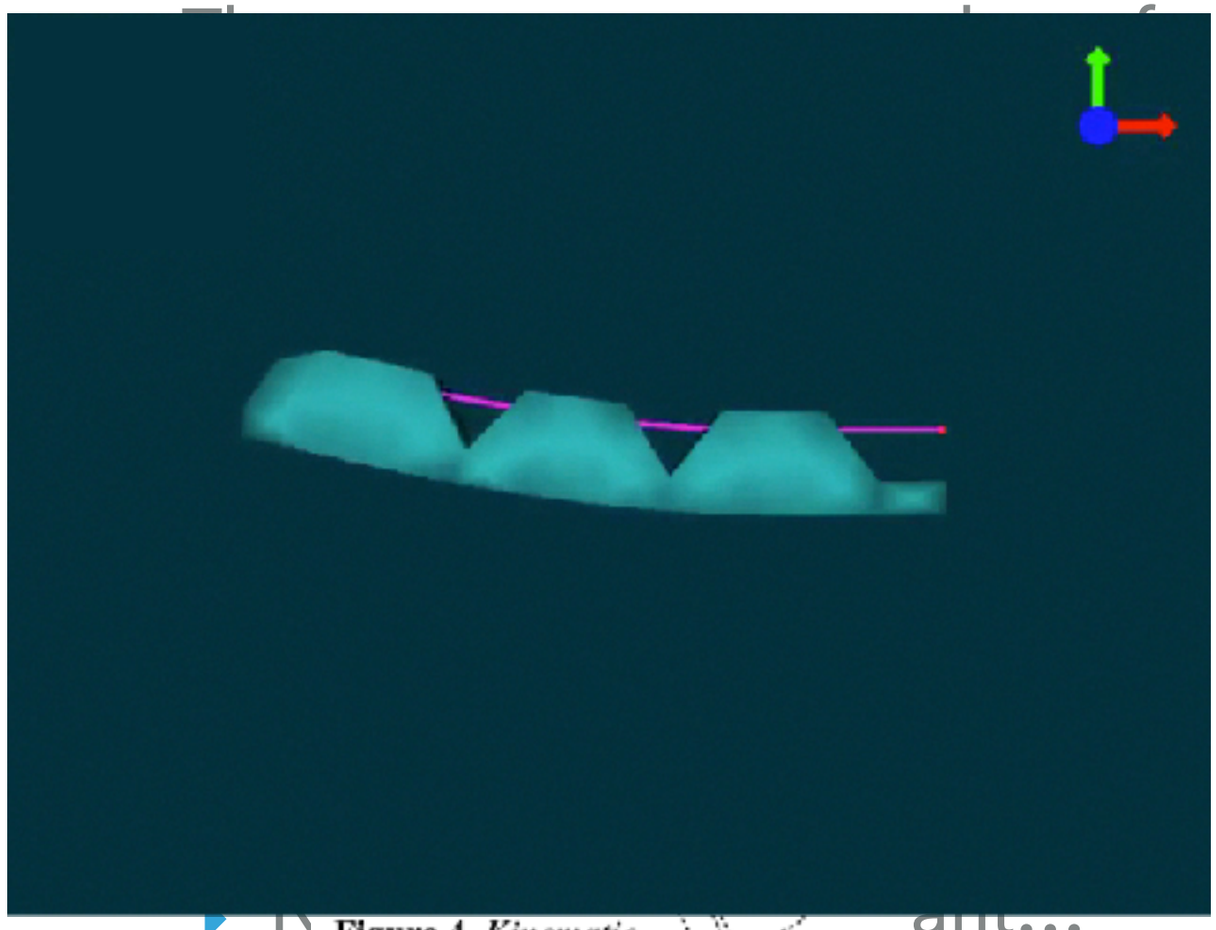
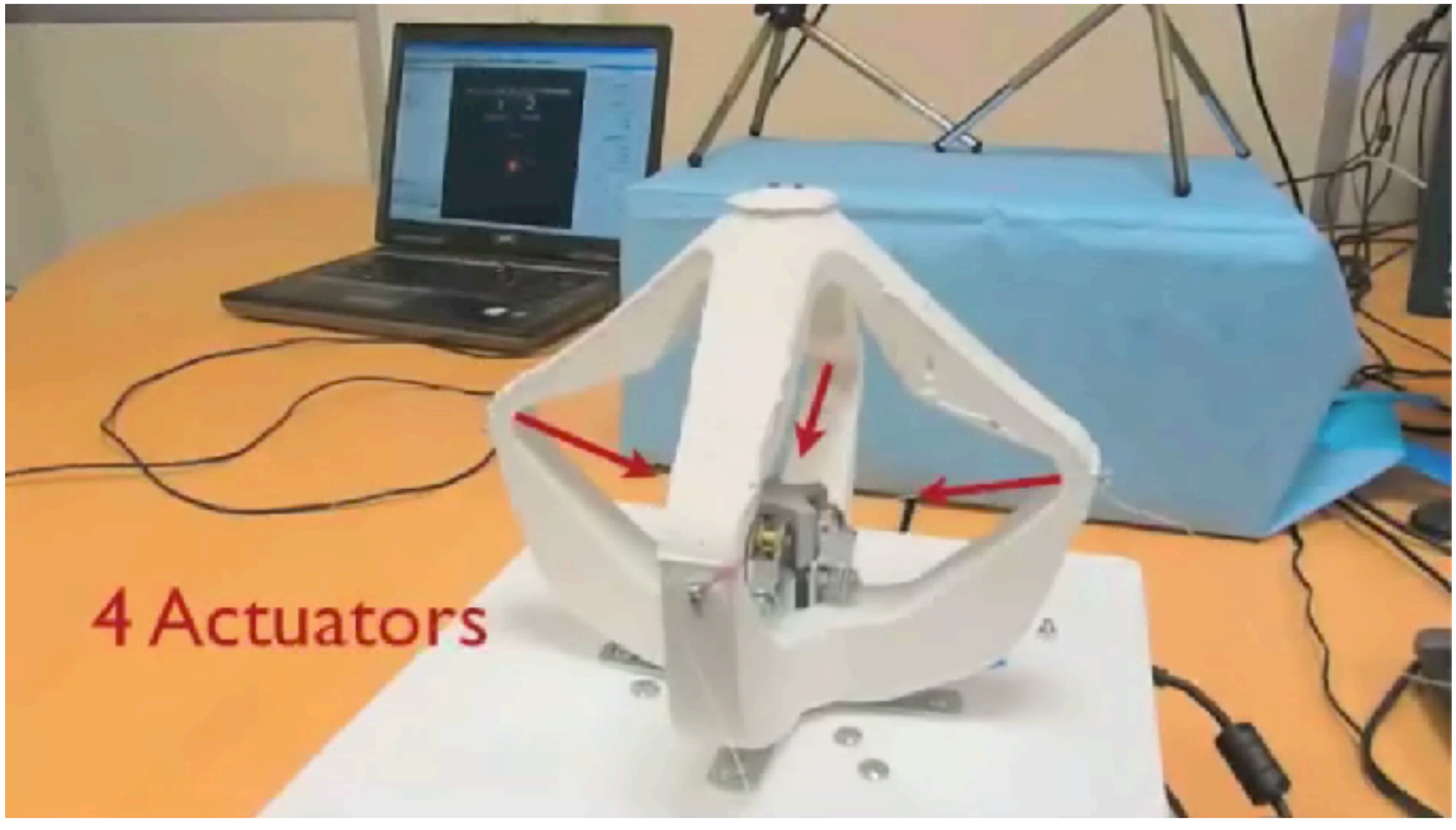
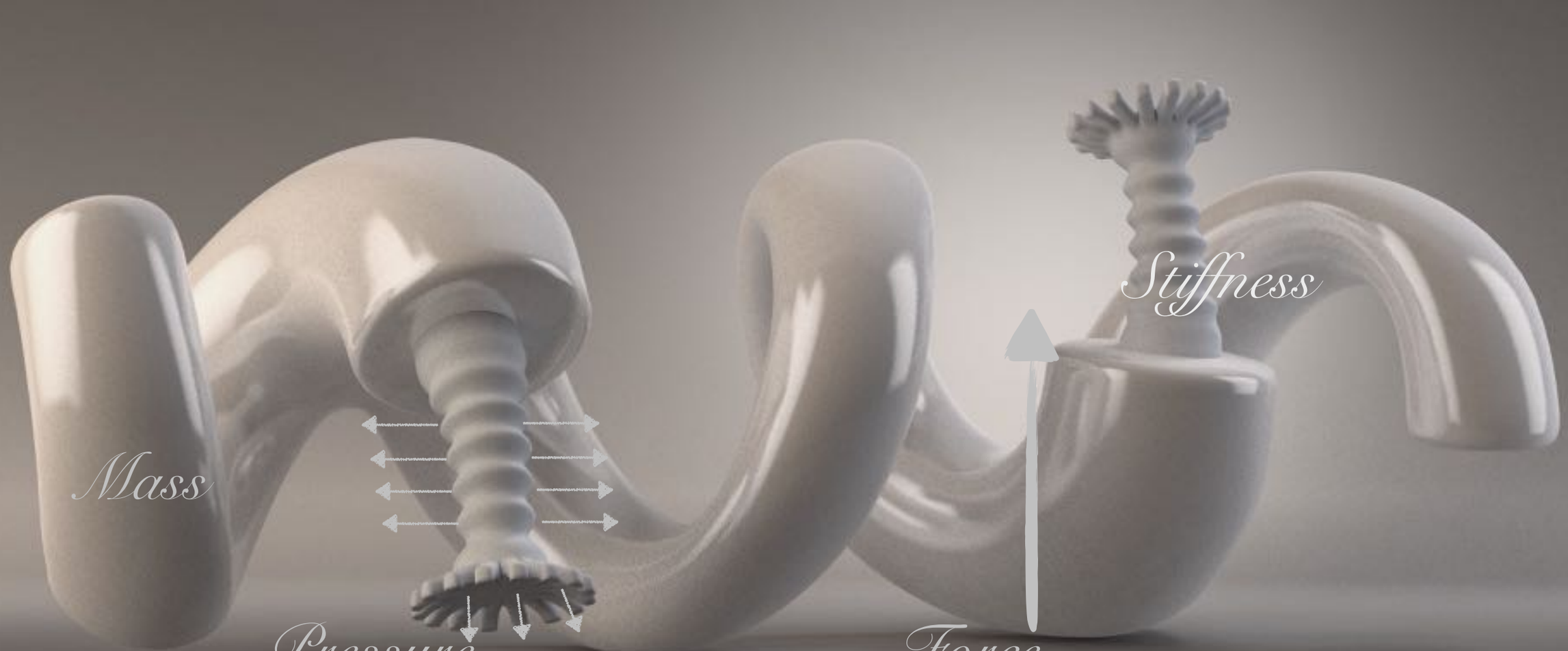


Figure 4. *Kinematic behavior depends on material properties.*

APPLICATIONS

- Test bench: parallel soft robot

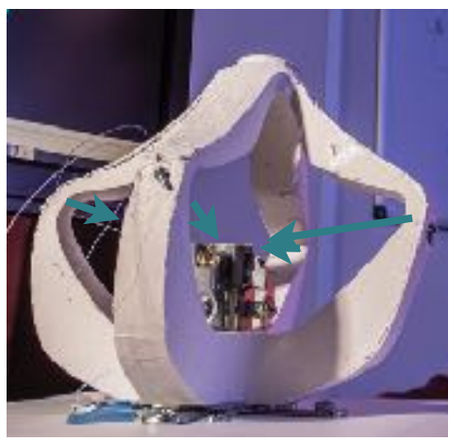




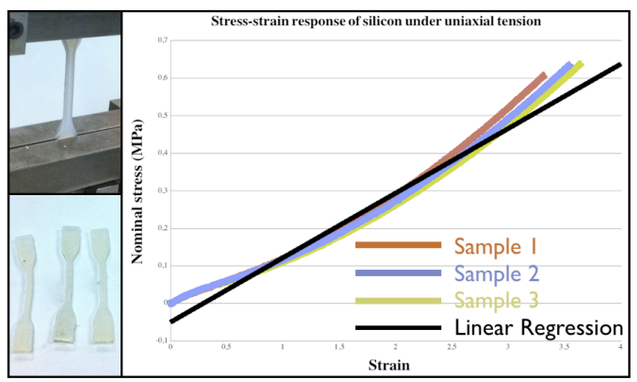
MODELING FOR REAL-TIME SIMULATION

MECHANICAL DEFORMABLE MODELS

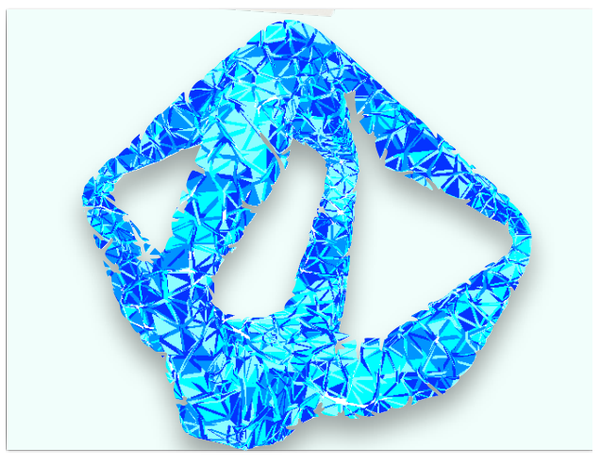
- FEM model



Deformable robot



Constitutive law



FEM mesh

- Newton's second law

$$\mathbb{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbb{P}(t) - \mathbb{F}\left(\mathbf{q}, \mathbf{v}\right) + \mathbf{H}^T\lambda$$

$\mathbf{q} \in \mathbb{R}^n$ Vector of generalized degrees of freedom (nodes of a deformable model)

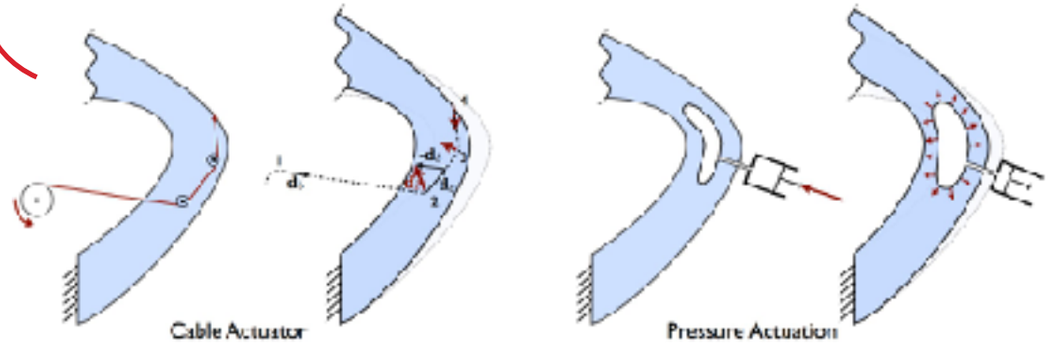
$\mathbf{v} \in \mathbb{R}^n$ Vector of velocities

$\mathbb{M}(\mathbf{q}) : \mathbb{R}^n \mapsto \mathcal{M}^{n\times n}$ Inertia Matrix

$\mathbb{F}\left(\mathbf{q}, \mathbf{v}\right)$ Internal forces (non-linear model)

$\mathbb{P}(t)$ External forces

$\mathbf{H}^T\lambda \in \mathbb{R}^n$ Constraint force contribution



TIME INTEGRATION SCHEMES

linear problem

COLLISION RESPONSE

What will happen in a few ms ?

COLLISION RESPONSE

A non-smooth event !

V- <0 before impact and V+ >0 after impact.. between them, an infinite small time step

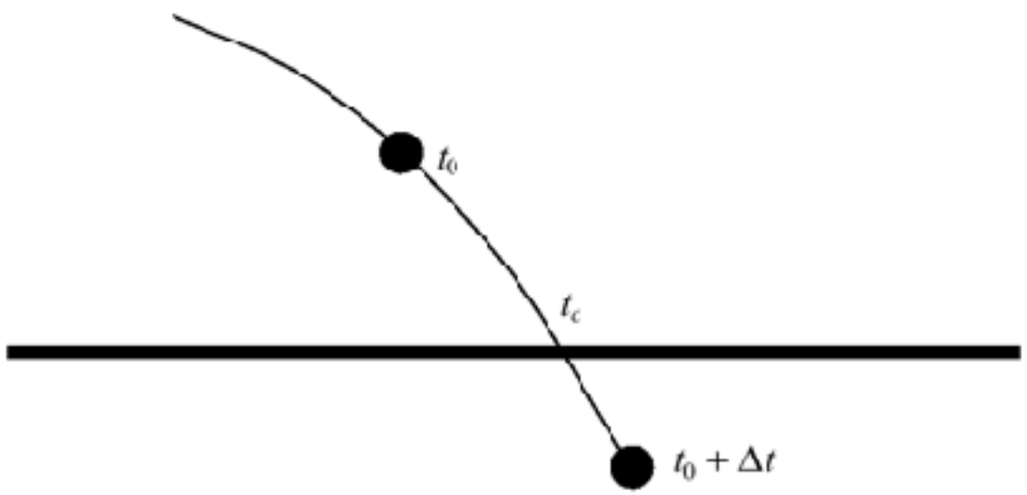
HOW TO DEAL WITH NON-SMOOTH EVENTS

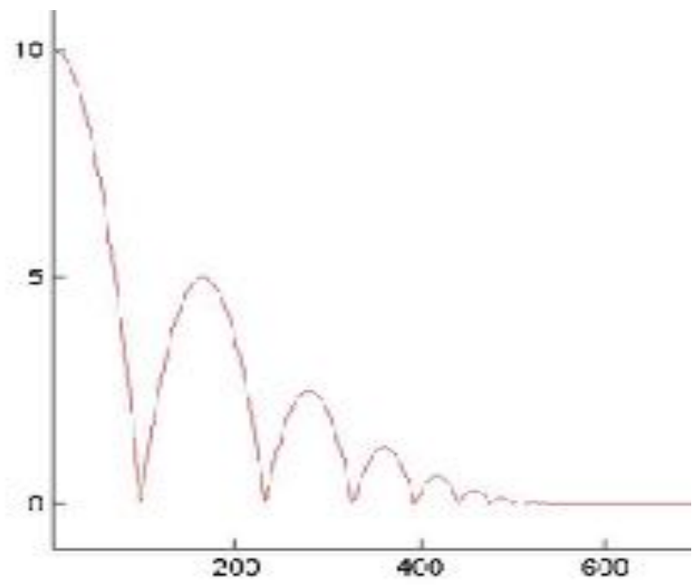
- EVENT-DRIVEN
  - At each new collision:
    - Determine the moment of the first impact,
    - Stop integration in time,
    - Solve the impact,
    - Restart the integration in time

- Advantage:
  - Between two events, we respect the necessary continuity to "high", and therefore precise, integration schemes.
- Drawbacks:
  - Costly in resolution if you have a lot of contacts,
  - Rebound "infinite" ... we have to give a stop criterion

HOW TO DEAL WITH NON-SMOOTH EVENTS

- EVENT-DRIVEN
  - Collision against a wall
    - Collision time tc
    - Restitution factor e of Newton
  - Problem of infinite rebound





HOW TO DEAL WITH NON-SMOOTH EVENTS

- TIME-STEPPING
  - Fixed time steps
    - We detect all the collisions that appeared during the time step,
    - We solve all these collisions "at the same time",
    - "Forced" movement with all the forces

- Advantage:
  - Faster if there are a lot of contacts,
- Drawbacks:
  - Use of low order integration scheme (Contact forces become impulses)
  - Small integration time steps

X

- Time-Stepping method

$$\begin{aligned}\mathbf{M}(\mathbf{v}_f - \mathbf{v}_i) &= h\left(\mathbb{P}(t_f) - \mathbb{F}(\mathbf{q}_f, \mathbf{v}_f)\right) + h\mathbf{H}^T\lambda \\ \mathbf{q}_f &= \mathbf{q}_i + h\mathbf{v}_f\end{aligned}$$

- 1 Linearization per step

$$\mathbb{F}\left(\mathbf{q}_i + d\mathbf{q}, \mathbf{v}_i + d\mathbf{v}\right) = \mathbf{f}_i + \frac{\delta\mathbb{F}}{\delta\mathbf{q}}d\mathbf{q} + \frac{\delta\mathbb{F}}{\delta\mathbf{v}}d\mathbf{v}$$

- Matrix system to be solved:

$$\underbrace{\left(\mathbf{M} + h\frac{\delta\mathbb{F}}{\delta\mathbf{v}} + h^2\frac{\delta\mathbb{F}}{\delta\mathbf{q}}\right)}_{\mathbf{A}}\underbrace{d\mathbf{v}}_{\mathbf{x}} = \underbrace{-h^2\frac{\delta\mathbb{F}}{\delta\mathbf{q}}\mathbf{v}_i - h\left(\mathbf{f}_i + \mathbf{p}_f\right)}_{\mathbf{b}} + h\mathbf{H}^T\lambda$$

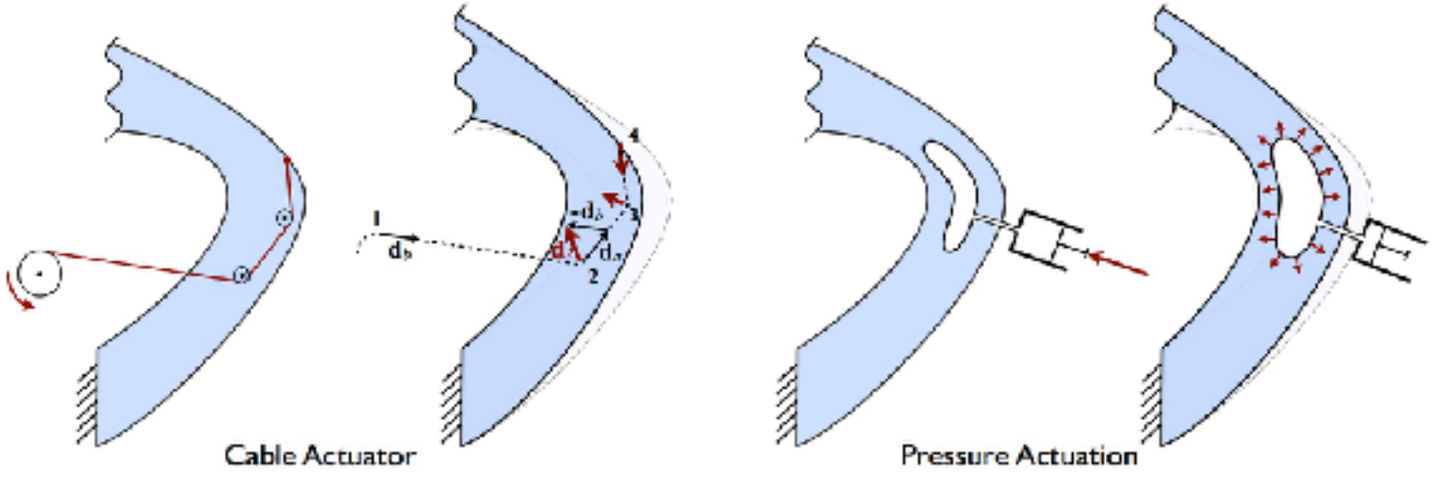
or (quasi static case)

$$\underbrace{\frac{\delta\mathbb{F}}{\delta\mathbf{q}}}_{\mathbf{A}}\underbrace{d\mathbf{q}}_{\mathbf{dx}} = \underbrace{\mathbb{P} - \mathbf{f}_i}_{\mathbf{b}} + \mathbf{H}^T\lambda$$

- « Direct simulation »

$h\mathbf{H}^T\lambda$ → actuators $\boldsymbol{\delta}_a(\mathbf{q}) = u$

$h\mathbf{H}^T\lambda$ → contacts $0 \leq \boldsymbol{\delta}_c \perp \boldsymbol{\lambda}_c \geq 0$



REAL-TIME INTEGRATION

- Interactive Simulation = the physician can modify the course of the simulation
- Time derivatives in model equation = integration scheme (notion of time step…)
- Between these two steps, the user do a certain motion in a REAL interval of time
- The time elapsed in the simulation must be EQUAL to the REAL interval of time

$h = dt$

$t + h$

$\mathfrak{t} + dt$

HOW TO MAKE IT REAL-TIME ?

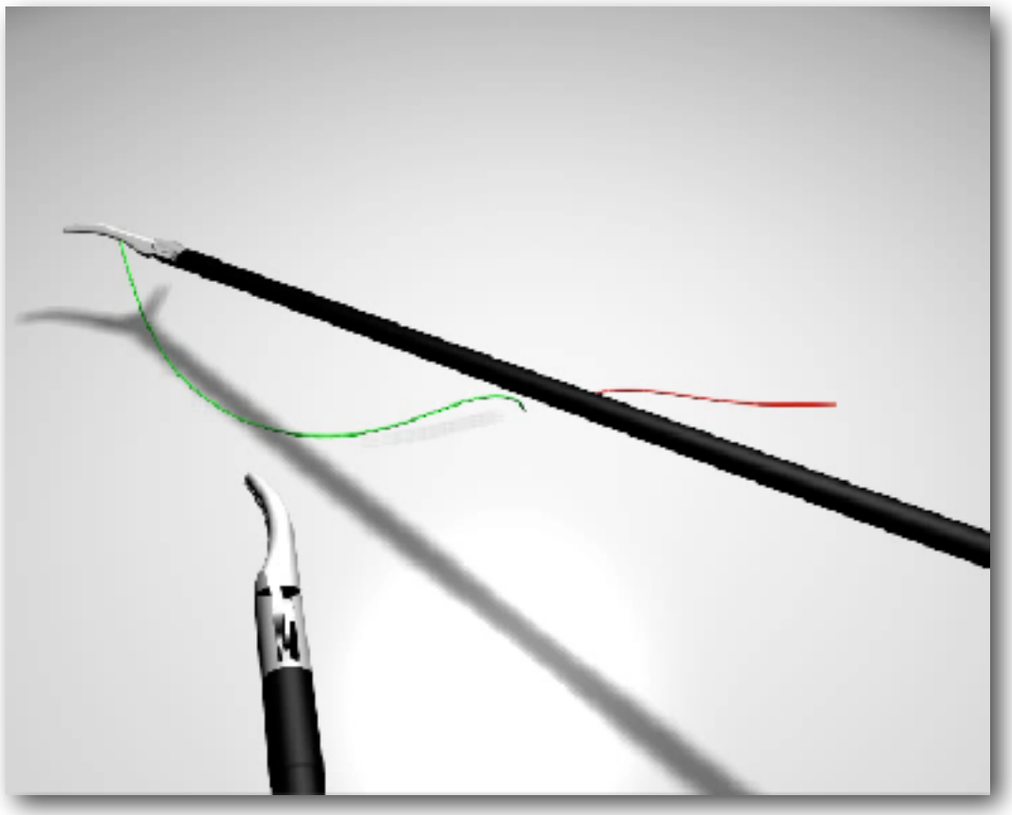
- Use of numerical recipes
  - Optimisation of the structure of the matrix (see D.James *et al.*,...)
  - Preconditioners (see H.Courtecuisse *et al.*)
  - Domain decomposition (see Barbic *et al.*, Kry *et al.*, ...)
  - ...
- Code & formulation optimisation
  - GPU implementation (see J.Allard *et al.*, ...)
  - Multigrid or adaptive methods (Georgii *et al.*, Debune *et al.*,...)
  - Fast hyperelastic models (see S.Marchesseaux, H.Delingette *et al.*)
  - ...
- Precomputation
  - Condensation methods (Cotin *et al.*,...)
  - Reduced-order methods (Barbic *et al.*, Goury *et al.*...)
  - ...

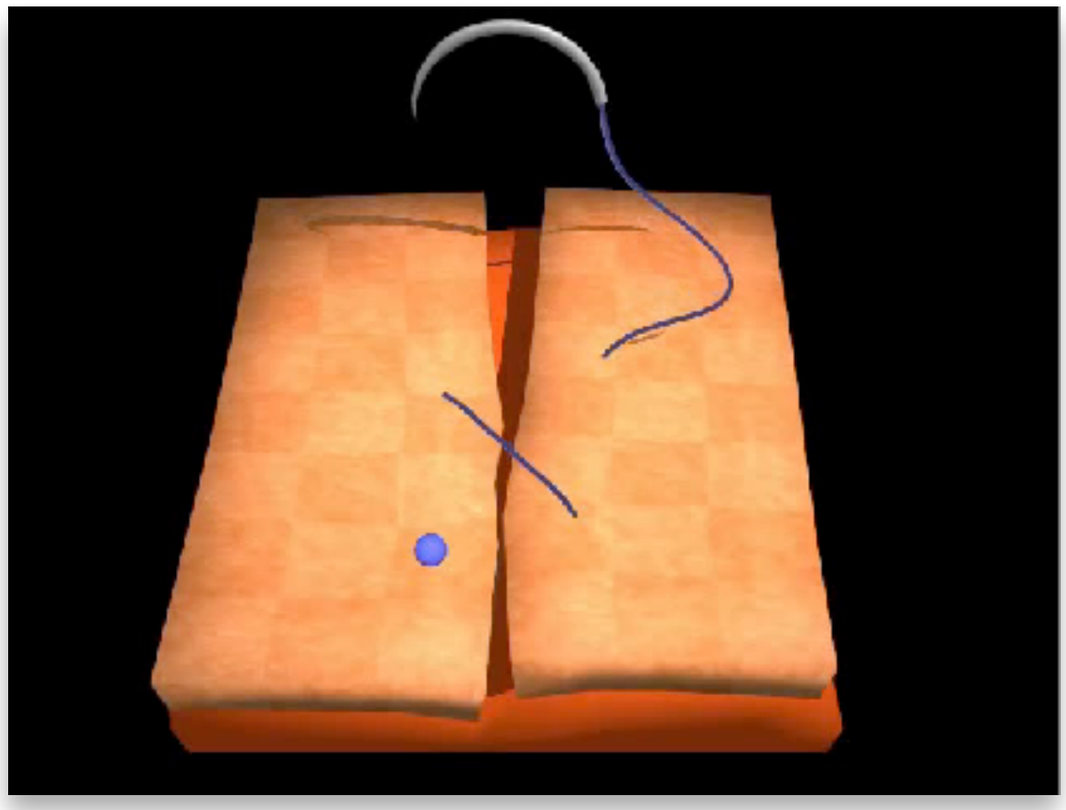
Different communities: computer animation, biomechanics, numerical methods,...

HOW TO MAKE IT REAL-TIME ?

- One example…
  - Block Tri-Diagonal Matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & & & \cdots & & 0 \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & & & & \\ & \ddots & \ddots & \ddots & & & \vdots \\ & & \mathbf{A}_k & \mathbf{B}_k & \mathbf{C}_k & & \\ \vdots & & & \ddots & \ddots & \ddots & \\ & & & & \mathbf{A}_{n-1} & \mathbf{B}_{n-1} & \mathbf{C}_{n-1} \\ 0 & & \cdots & & & \mathbf{A}_n & \mathbf{B}_n \end{bmatrix}$$



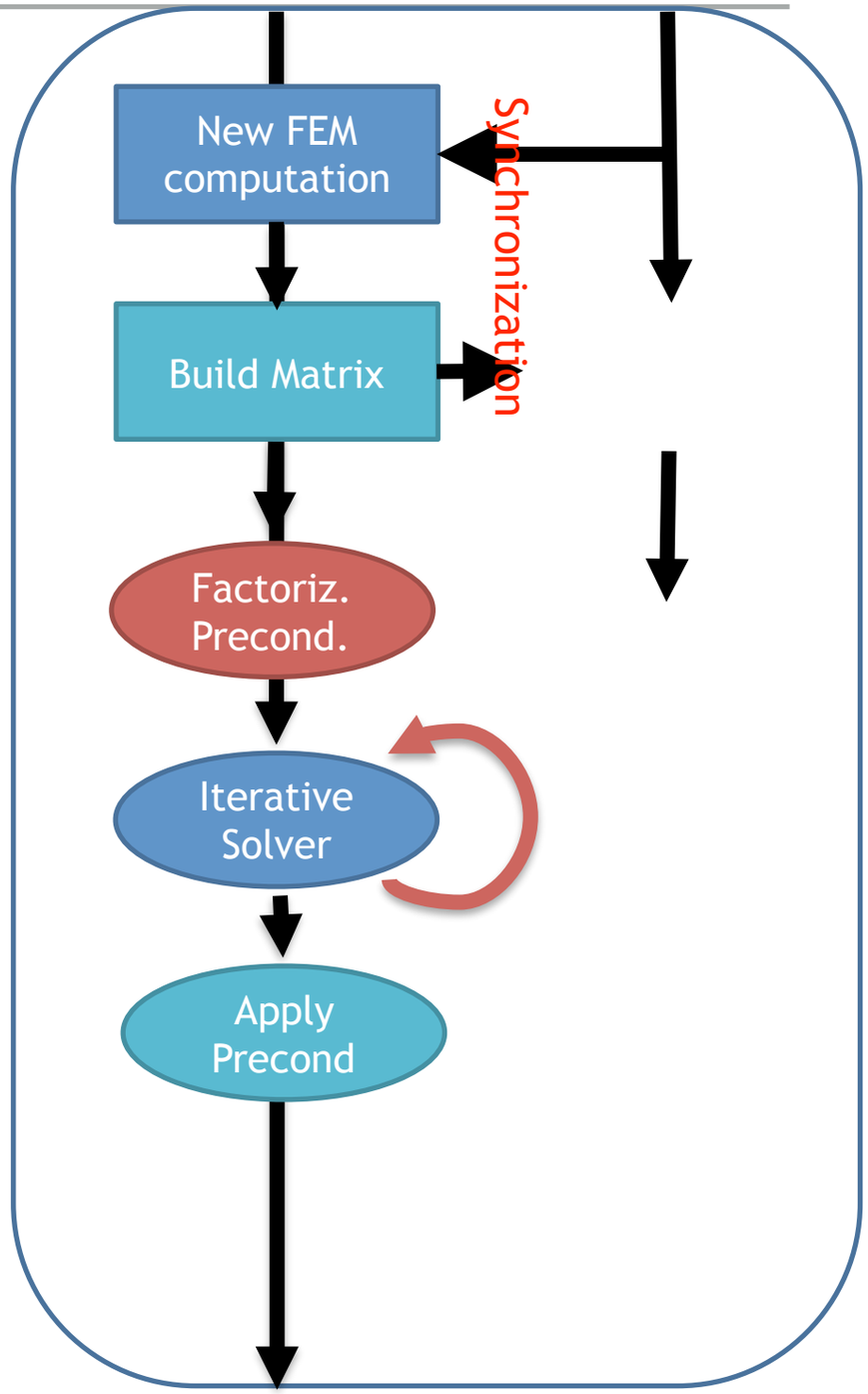


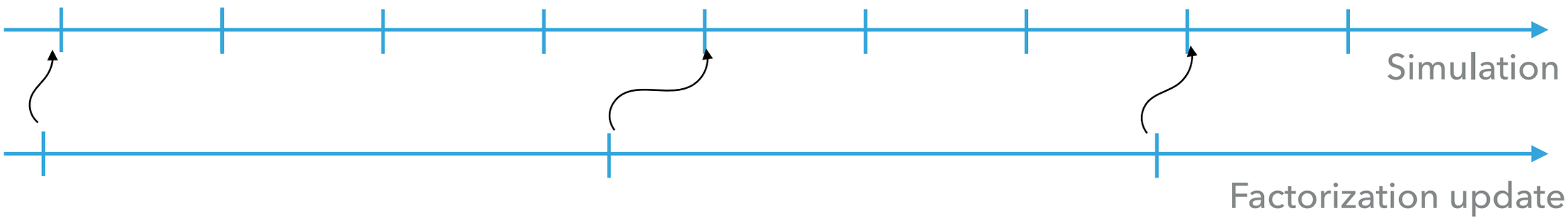
HOW TO MAKE IT REAL-TIME ?

- Asynchronous Preconditionner
  - Solve system A x = b (A, b known, x unknown)
    - Factorization costly / Iterative solvers poor convergence when bad conditioning of matrix A
  - Motivations for asynchronous preconditionner:
    - Temporal Coherency of A: an « old » value of A is a good preconditionner
  - Use of a preconditionner:

$$\underbrace{P^{-1}A}_{\approx I} x = P^{-1}b$$

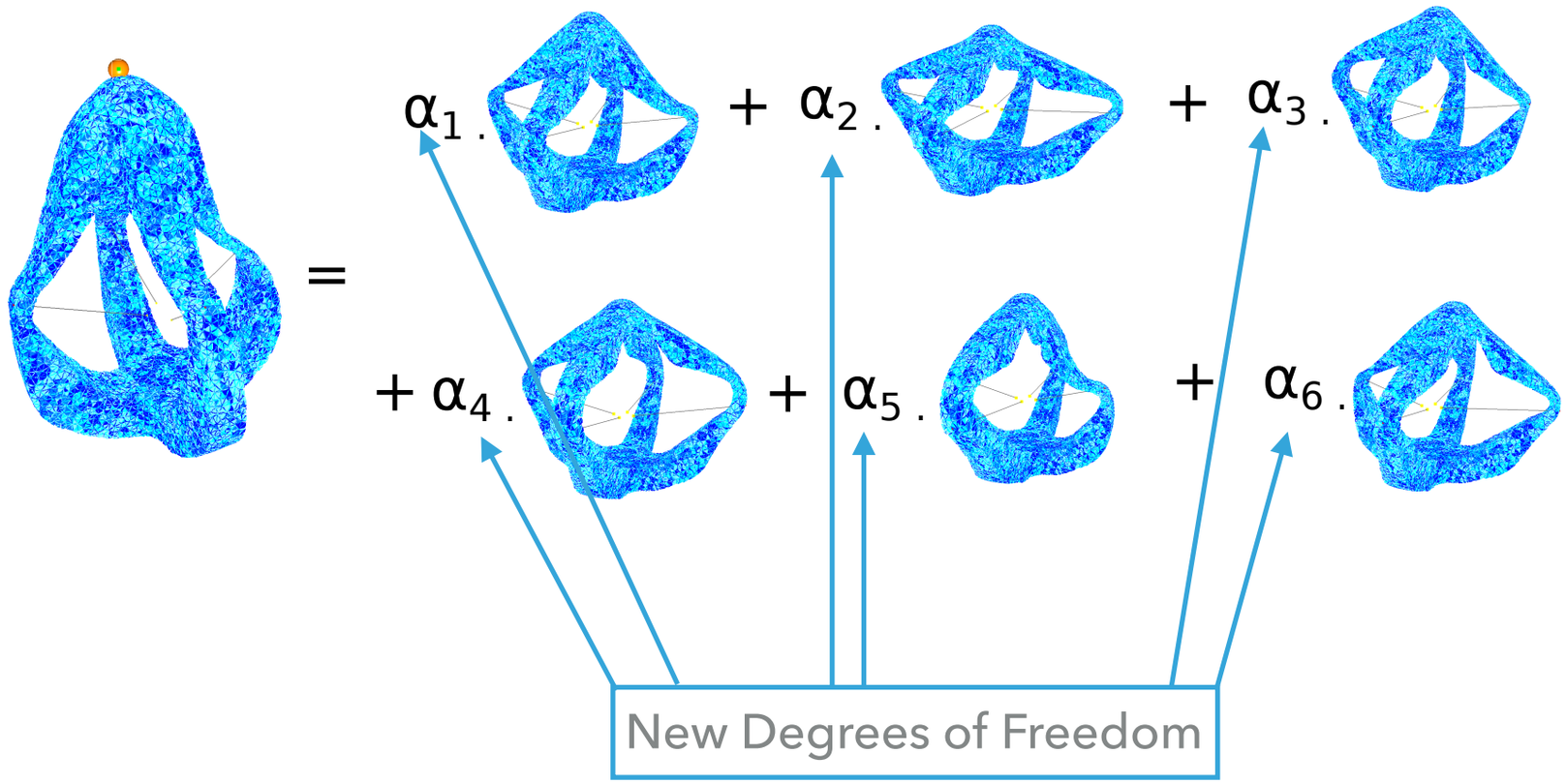
  - The cost of factorization of P is not supported by the real-time loop !!





MODEL ORDER REDUCTION

$$\mathbf{q}\left(t, \boldsymbol{\lambda}(t)\right) \approx \mathbf{q}(0) + \sum_{i=1}^{N} \phi_i \alpha_i(t, \boldsymbol{\lambda}(t)) = \mathbf{q}(0) + \boldsymbol{\Phi}\, \boldsymbol{\alpha}(t, \boldsymbol{\lambda}(t)) \quad (4)$$



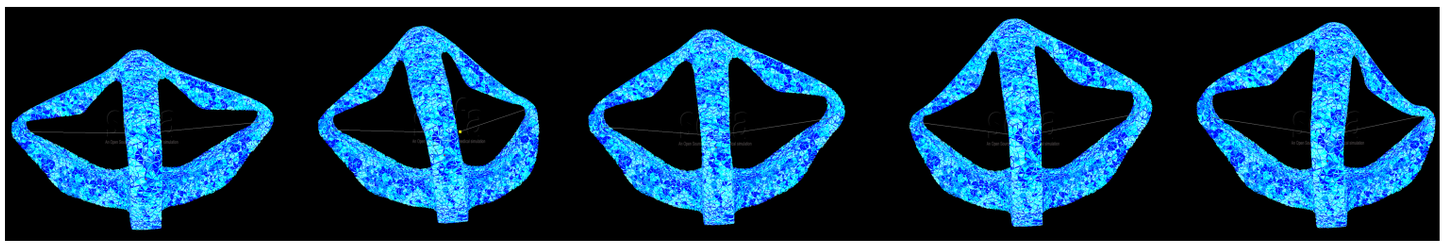
Goury & Duriez IEEE Trans on Robotics, 2018

MODEL ORDER REDUCTION

- This leads to the reduced equations

$$\underbrace{\mathbf{\Phi}^T \mathbf{A}(\mathbf{q}_t, \mathbf{v}_t)\mathbf{\Phi}}_{\mathbf{A}_r} \dot{\boldsymbol{d}}\alpha(t+1) = \underbrace{\mathbf{\Phi}^T \mathbf{b}(\mathbf{q}_t, \mathbf{v}_t)}_{\mathbf{b}_r} + \underbrace{(\mathbf{H}\,\mathbf{\Phi})^T}_{\mathbf{H}_r} \boldsymbol{\lambda},$$

- Offline stage:
  - shake the robot within the range of its actuators => compute a snapshot space $\mathbf{S}$
  - Find basis that minimize the cost function $\widehat{J}(\mathbf{\Phi})^2 = \sum_{\boldsymbol{\lambda}^* \in \widehat{\boldsymbol{\Lambda}}} \sum_{t=t_0}^{t=t_{n_t}} \left\| \mathbf{q}(t, \boldsymbol{\lambda}^*(t)) - \sum \left({\phi_i}^T \mathbf{q}(t, \boldsymbol{\lambda}^*(t)) \right) \phi_i \right\|_2^2 .$
  - After SVD $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ (and truncation), we obtain the reduced basis:



Goury & Duriez IEEE Trans on Robotics, 2018

MODEL ORDER REDUCTION

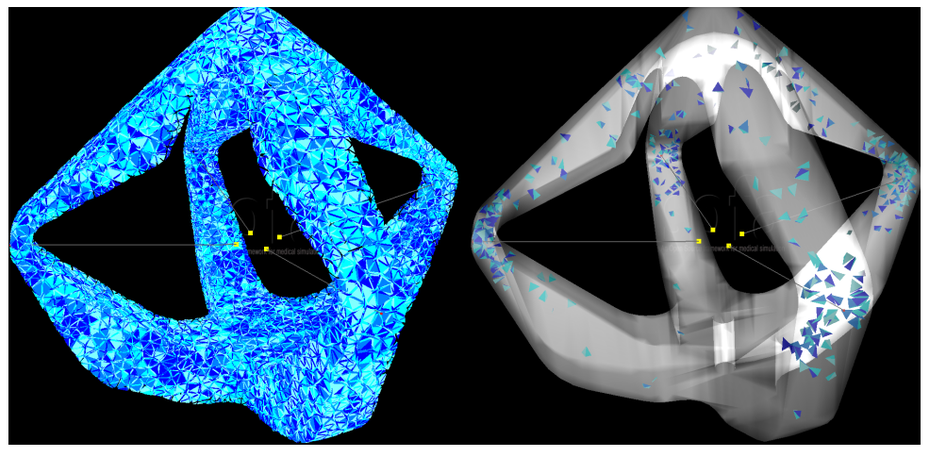
- Hyper reduction

$$\underbrace{\mathbf{\Phi}^T \mathbf{A}(\mathbf{q}_t, \mathbf{v}_t)\mathbf{\Phi}}_{\mathbf{A}_r} \dot{\boldsymbol{d}}\alpha(t+1) = \underbrace{\mathbf{\Phi}^T \mathbf{b}(\mathbf{q}_t, \mathbf{v}_t)}_{\mathbf{b}_r} + \underbrace{(\mathbf{H}\,\mathbf{\Phi})^T}_{\mathbf{H}_r} \boldsymbol{\lambda},$$

To evaluate $\mathbf{A}_r$, we still need to compute the matrix $\mathbf{A}(\mathbf{q}_t, \mathbf{v}_t)$.

[Farhat 2014]. Go back to the element level to express the reduced forces as a weighed sum of a small subset of elements:

$$\begin{aligned} \mathbf{\Phi}^T \mathbf{g}(t;\boldsymbol{\lambda}) &= \sum_{e \in \mathcal{E}} \mathbf{\Phi}_e^T \mathbf{g}_e(t;\boldsymbol{\lambda}) \\ &\approx \sum_{e \in \widehat{\mathcal{E}}} \xi_e \mathbf{\Phi}_e^T \mathbf{g}_e(t;\boldsymbol{\lambda}) \end{aligned}$$



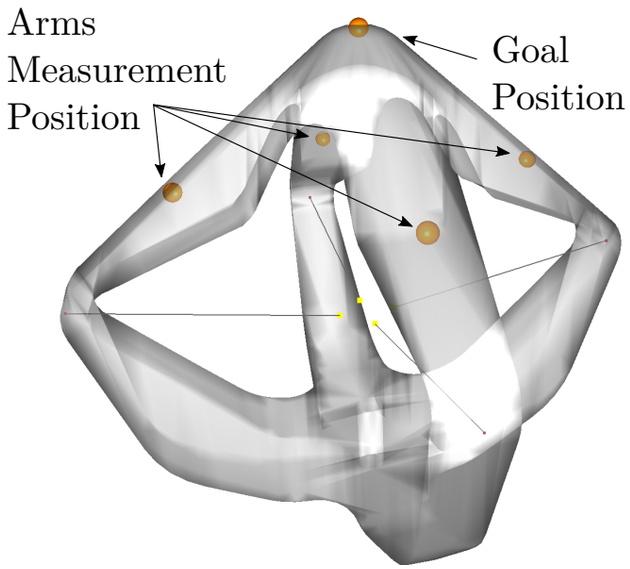
Goury & Duriez IEEE Trans on Robotics, 2018

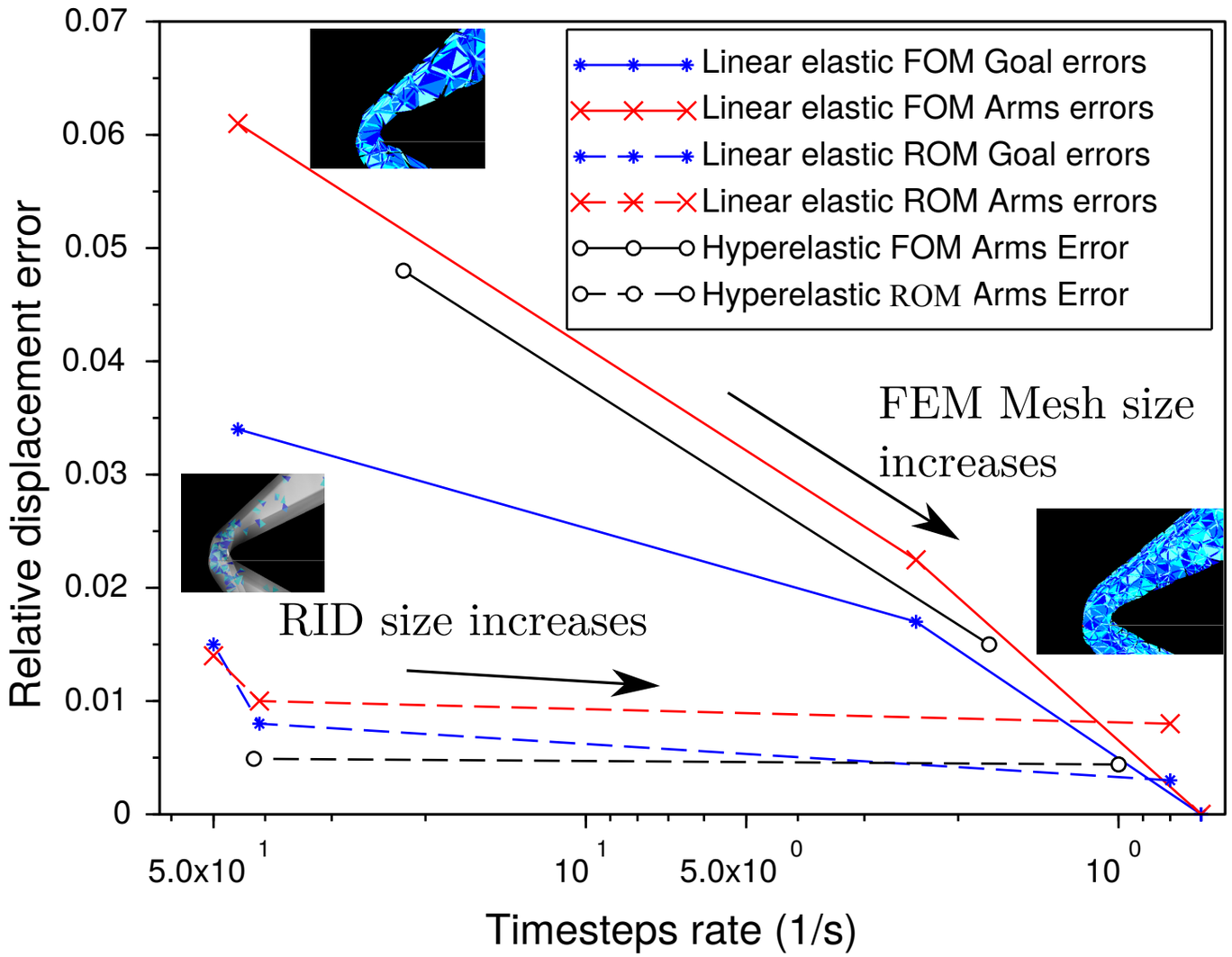
MODEL ORDER REDUCTION

- Performance

Full Order Model (FOM)

Reduced Order Model (ROM)

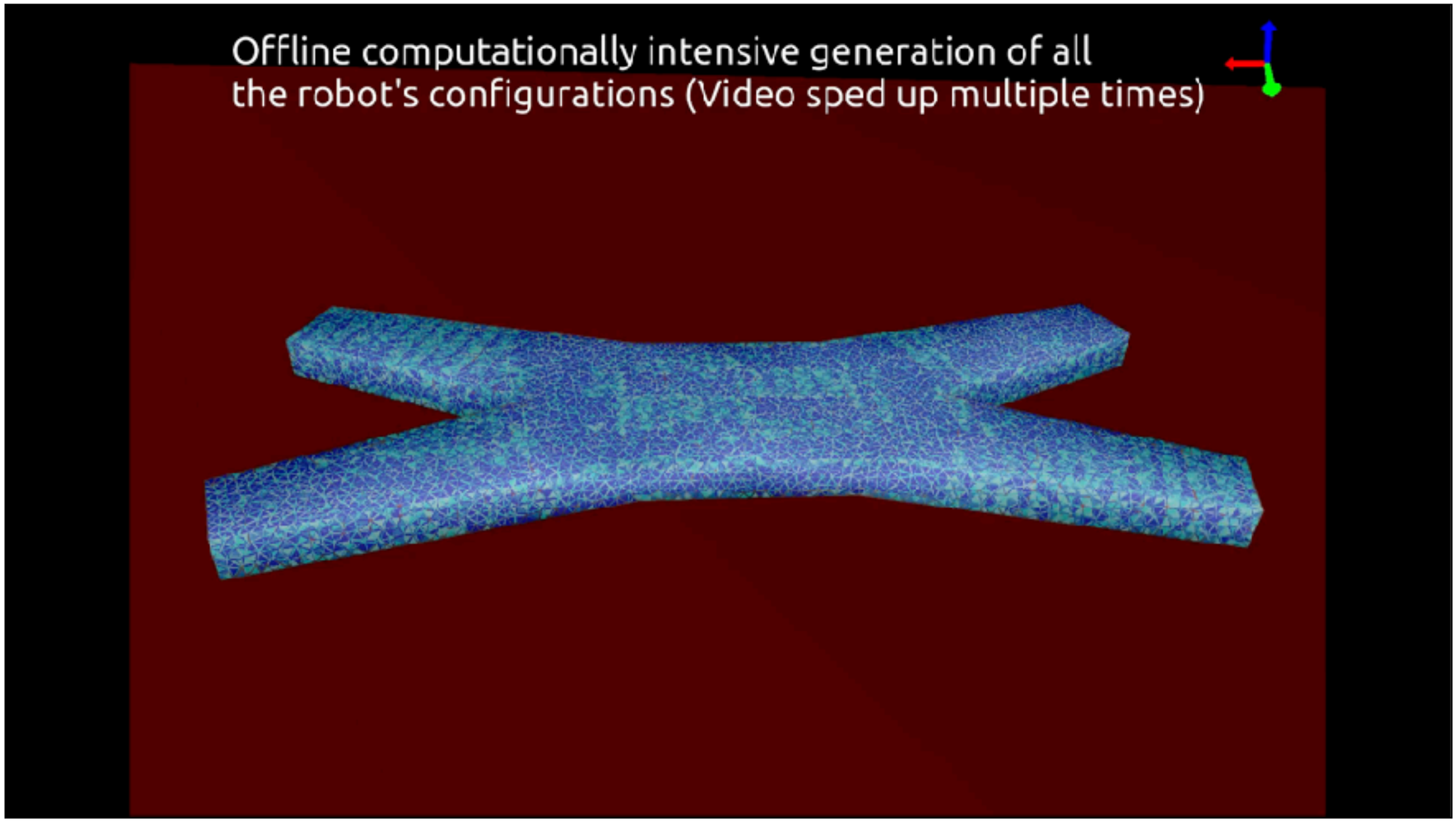




Better tradeoff between accuracy and performance

Goury & Duriez IEEE Trans on Robotics, 2018

MODEL ORDER REDUCTION

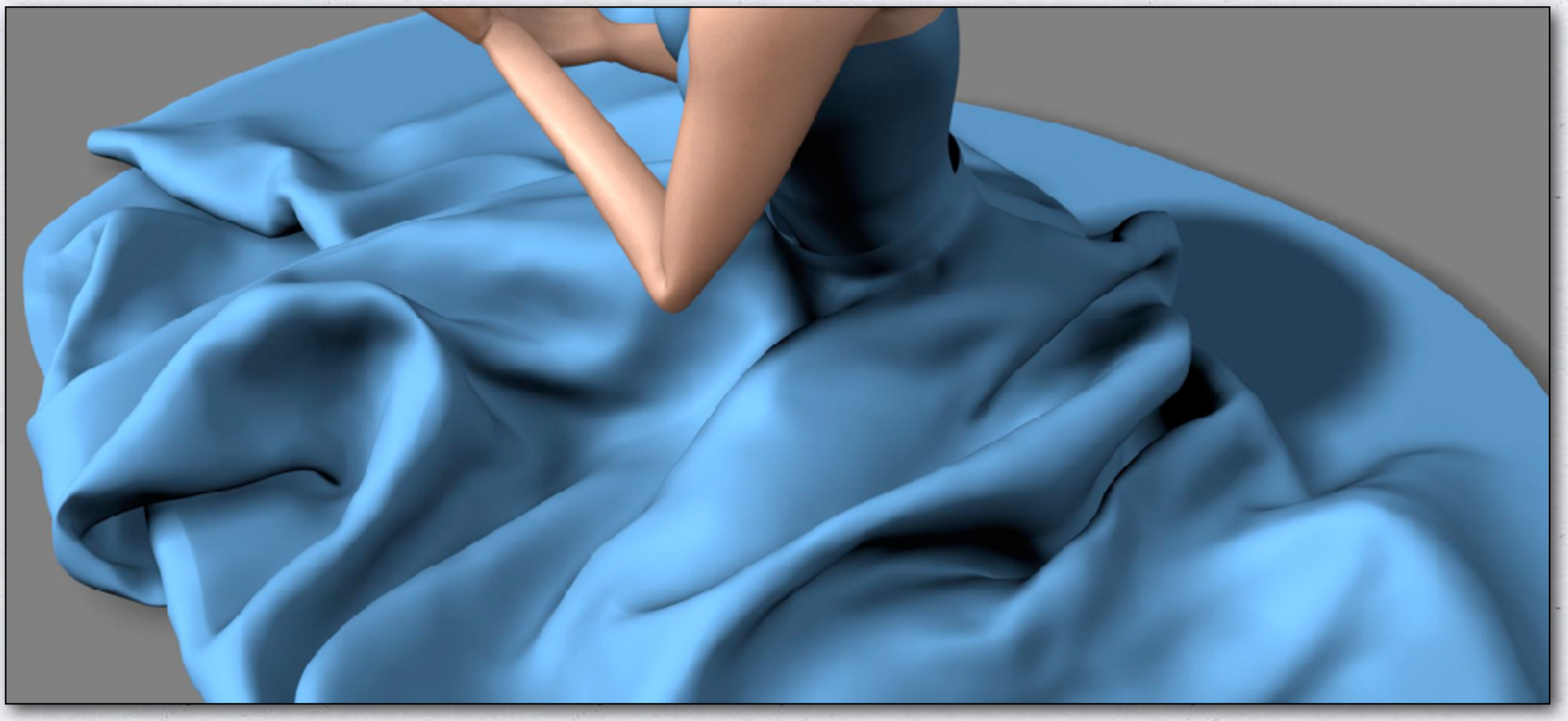


Goury & Duriez IEEE Trans on Robotics, 2018

MODELING SOFT-ROBOTS IN THEIR ENVIRONMENT

DÉTECTION DE COLLISIONS

Jérémie Dequidt / jeremie.dequidt@inria.fr / http://dequidt.plil.net



Deux “Stratégies”

* Détection Spatiale (détection à chaque *dt* de la simulation)

* Détection Spatio-Temporelle (prise en compte du temps dans la détection)

Principe de la détection spatiale

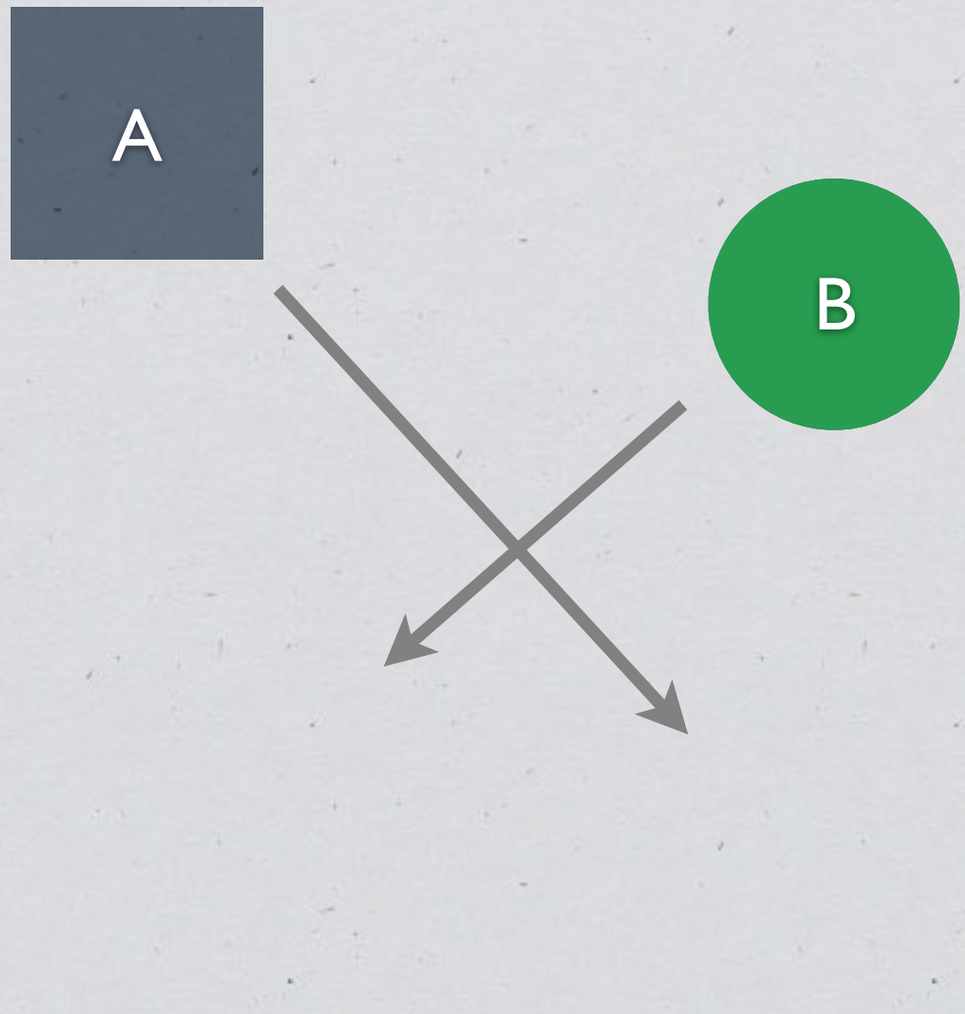
* On observe le système à des instants donnés (*i.e.* tous les pas de temps de la simulation)

* On détecte les intersections non-vides entre objets

* Basé sur la géométrie algorithmique

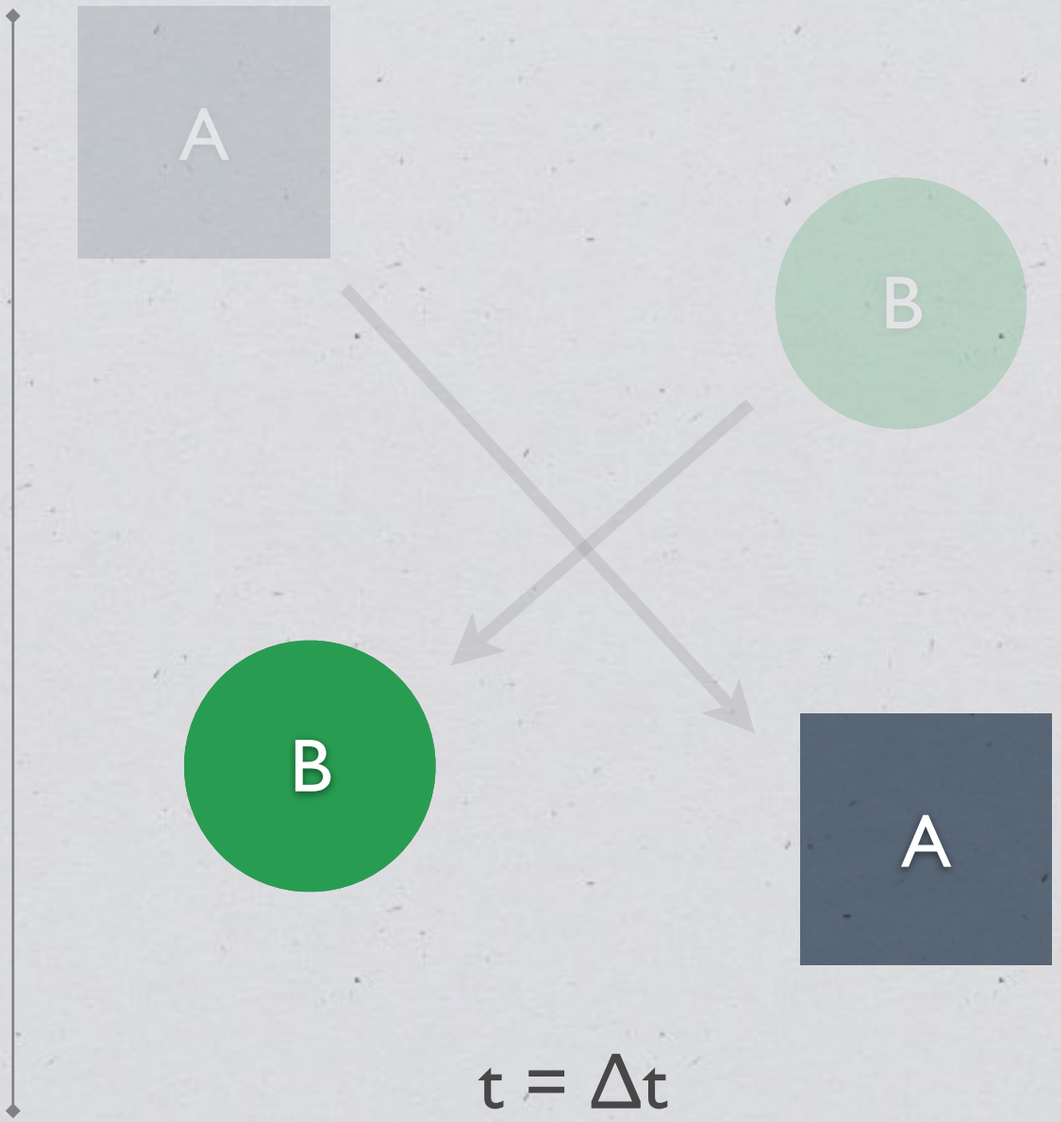
Limite

* Possibilité de “*manquer*” des collisions



Limite

* Possibilité de “*manquer*” des collisions … si vitesse trop importante



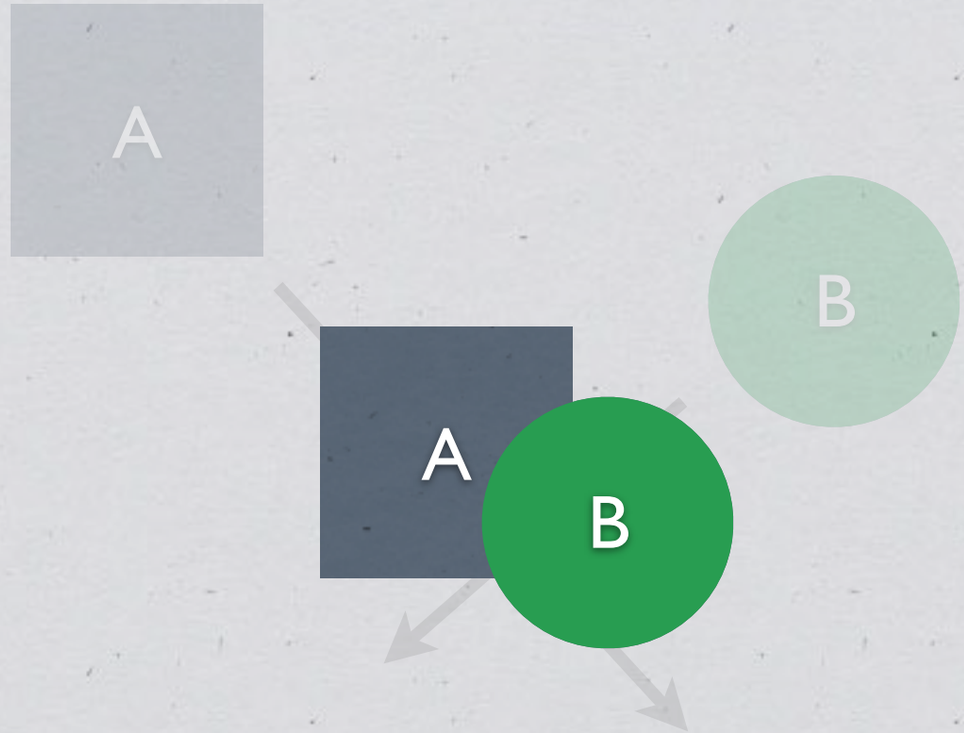
Limite

* Possibilité de “*manquer*” des collisions … si vitesse trop importante

* Notion de longueur critique:

$$l_c = v_{\max} \times \Delta t$$

* Les objets pourront passer l’un au travers de l’autre si leurs dimensions sont plus petites que *l_c*



Classes d'algorithmes

* Différents types d'informations peuvent caractériser une collision

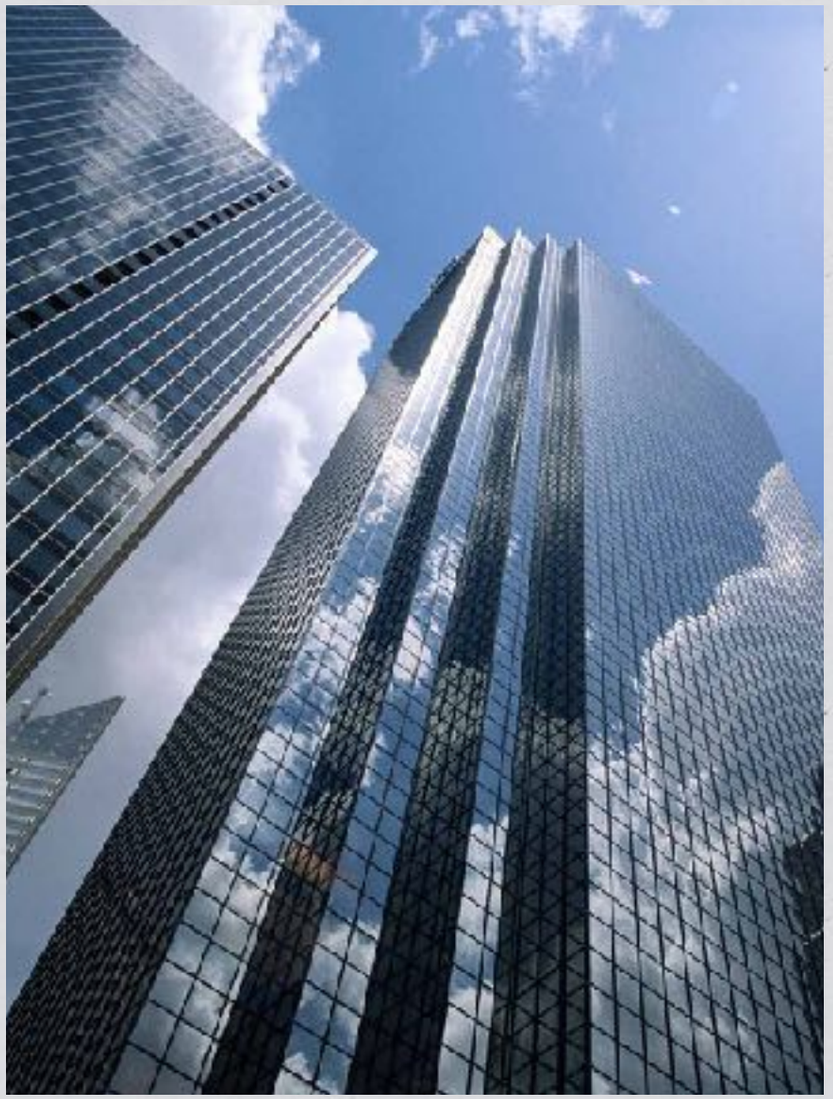
* De manière générale, plus l'information est complète, plus l'algorithme pour la fournir est complexe

* Classification des méthodes par complexité croissante

Classes d’algorithmes

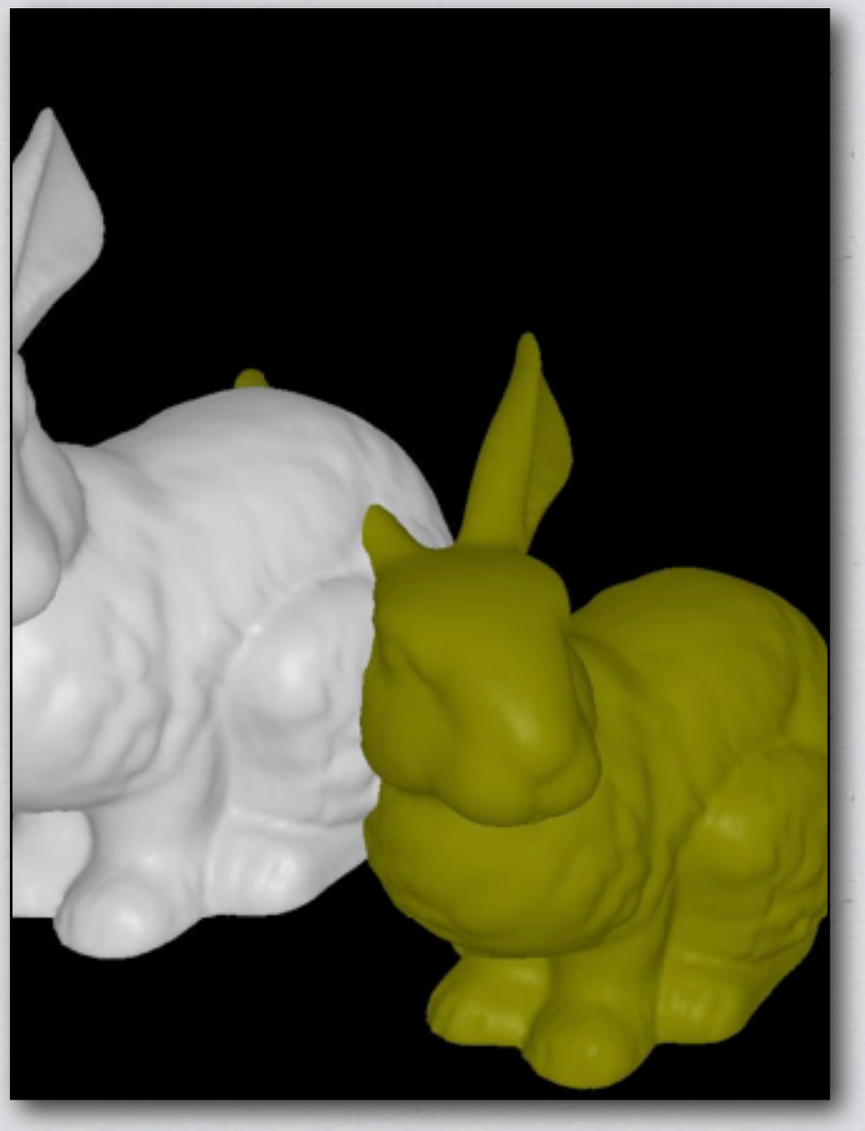
* Détection d’intersection non-vide

  * réponse booléenne



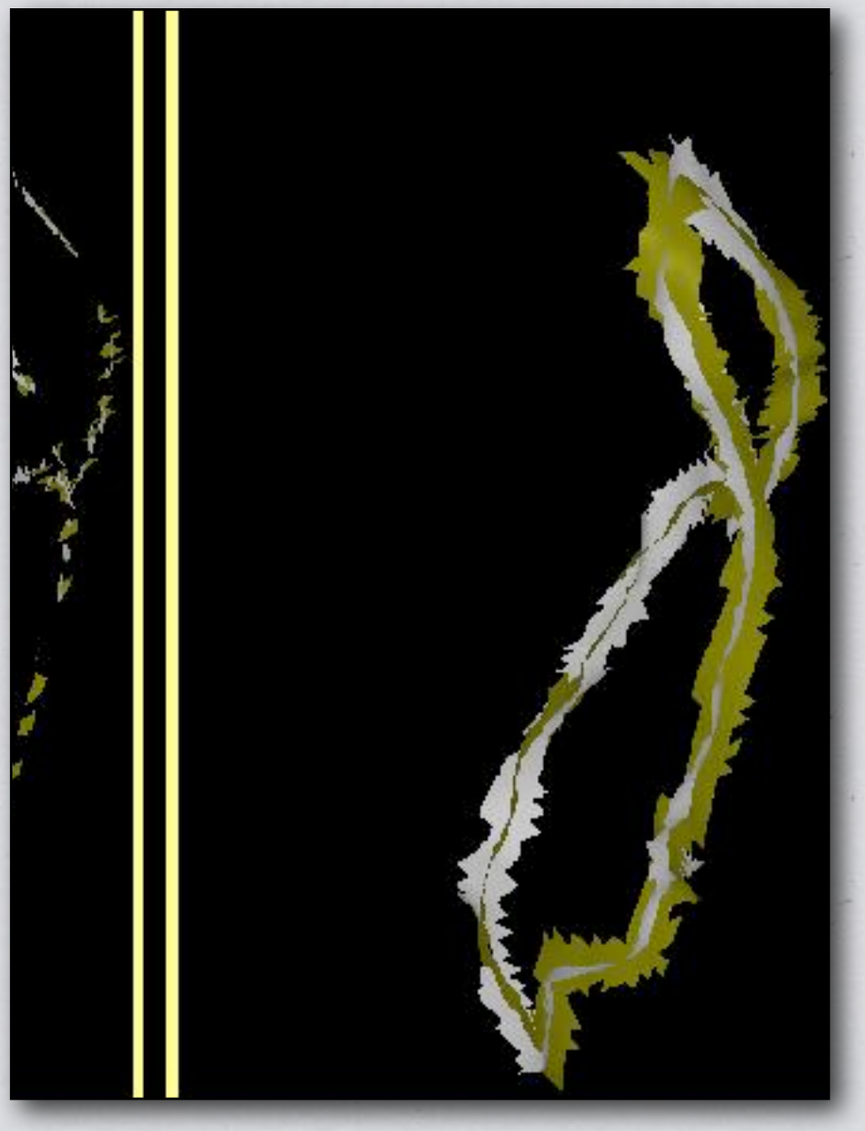
Classes d’algorithmes

* Intersection des surfaces des objets:
    * recherche des primitives d’un objet qui intersectent un autre objet



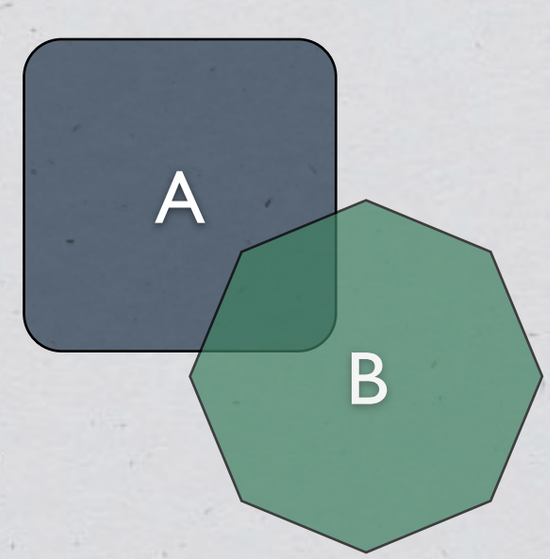
Classes d’algorithmes

* Intersection des surfaces des objets:
  * recherche des primitives d’un objet qui intersectent un autre objet



Classes d’algorithmes

* Estimation de l’amplitude de l’intersection
  * volume de l’intersection
  * distance d’interpénétration (plus petite translation qui permet de séparer les objets)

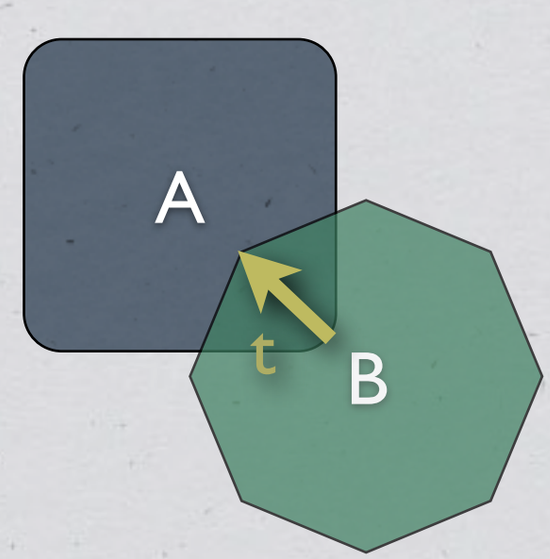


Classes d’algorithmes

* Estimation de l’amplitude de l’intersection

  * distance d’interpénétration (plus petite translation qui permet de séparer les objets)

  * volume de l’intersection



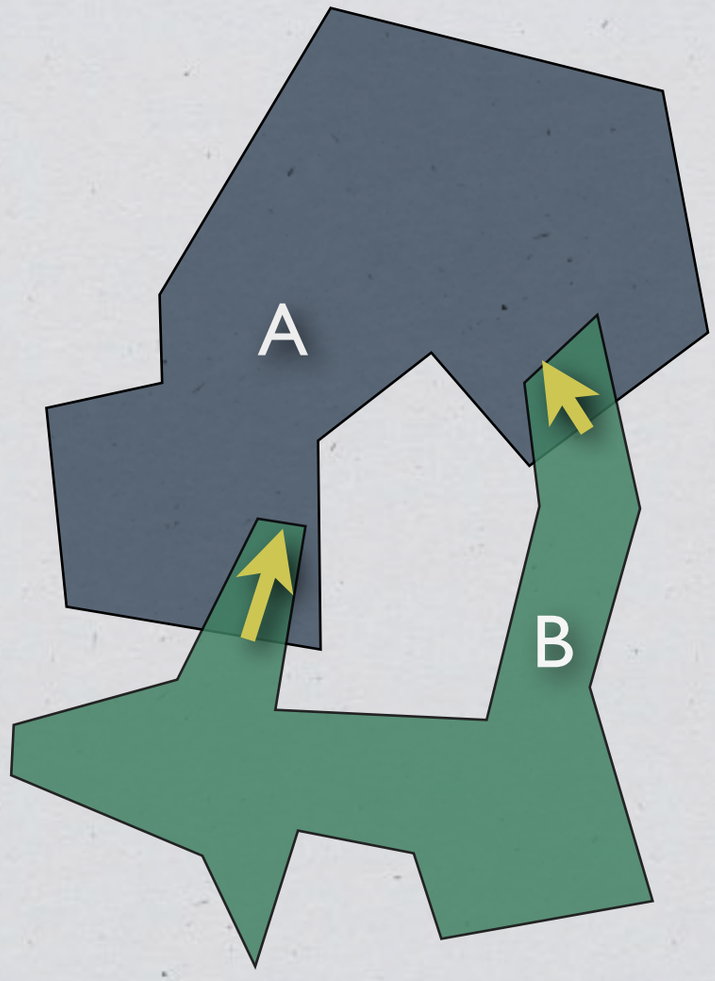
Classes d’algorithmes

* Aparté sur la distance d’interpénétration …

* … cas d’un volume d’intersection

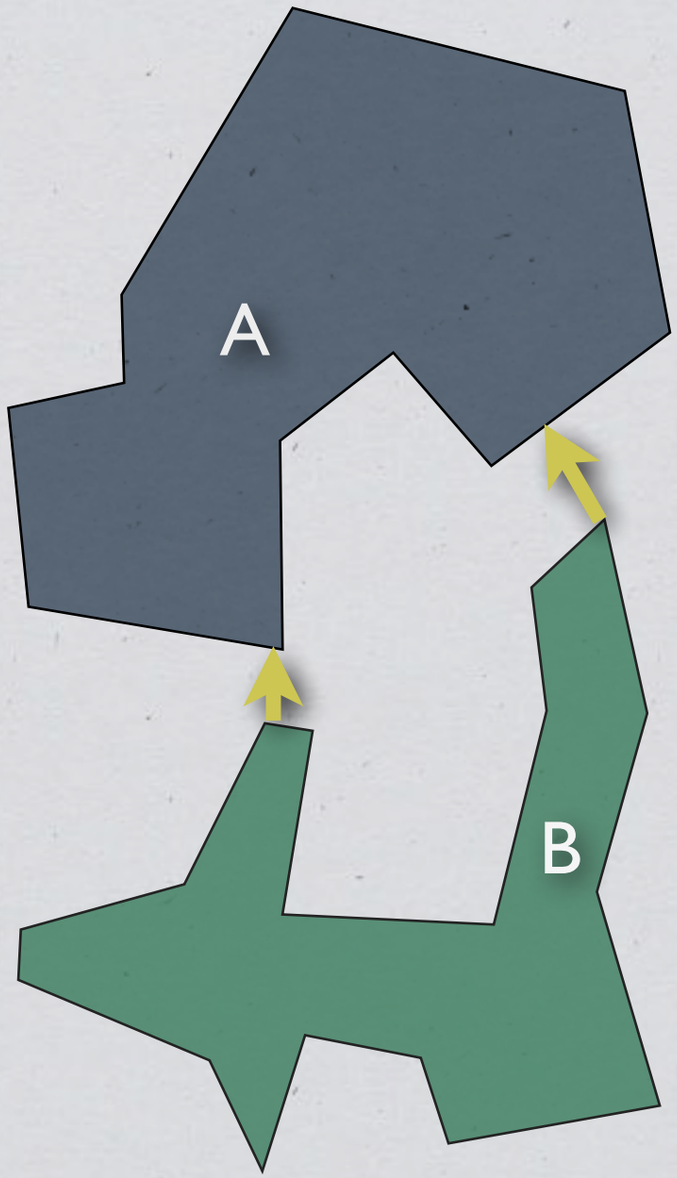
  * convexe

  * non-convexe



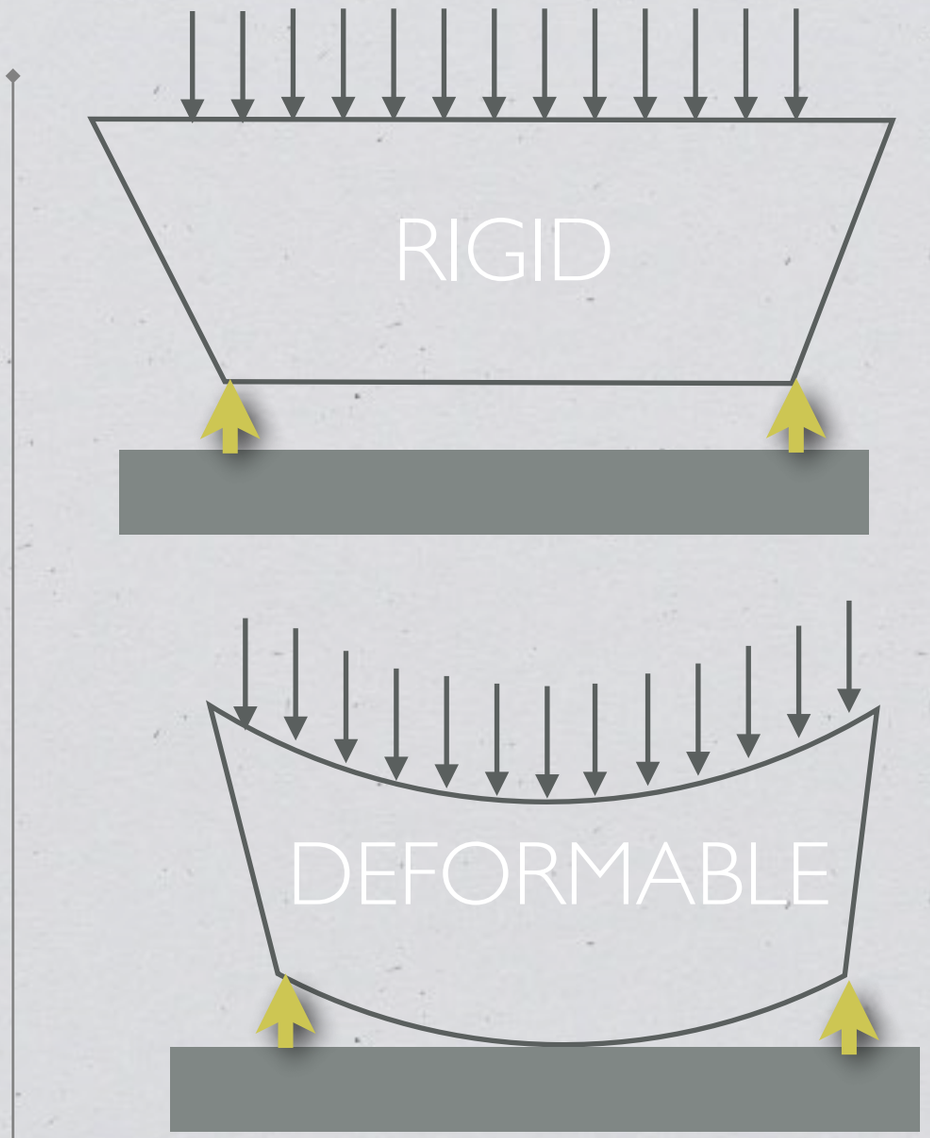
Classes d'algorithmes

- Détection de proximité
  - Calcul de minimum global (cas convexe) ou locaux (cas concave) de la distance entre 2 objets



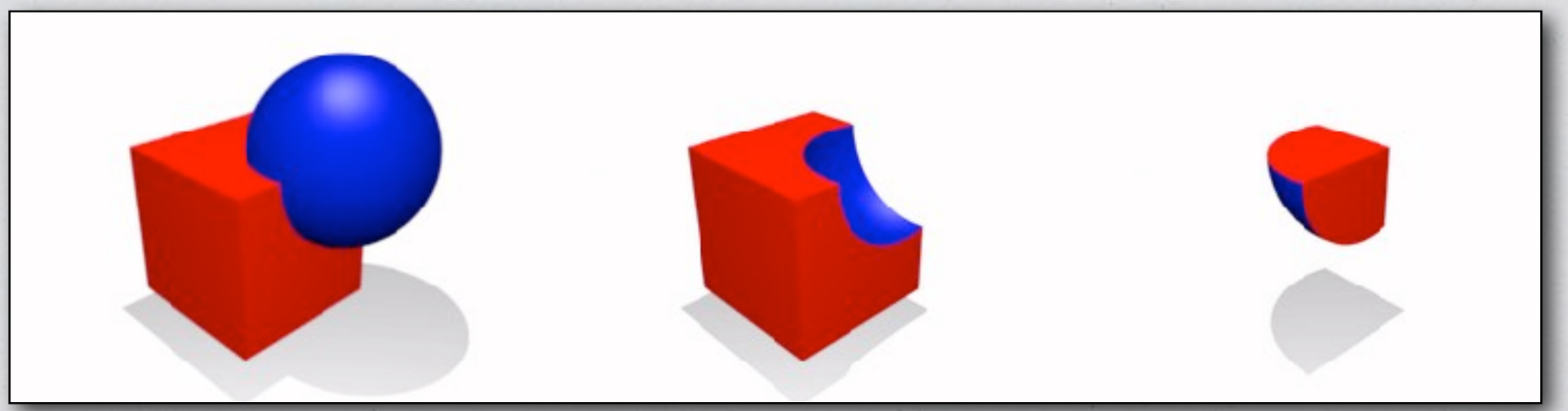
Classes d'algorithmes

* Détection de proximité
  * Calcul de minimum global (cas convexe) ou locaux (cas concave) de la distance entre 2 objets
* Attention: pas le même problème pour les objets rigides et déformables



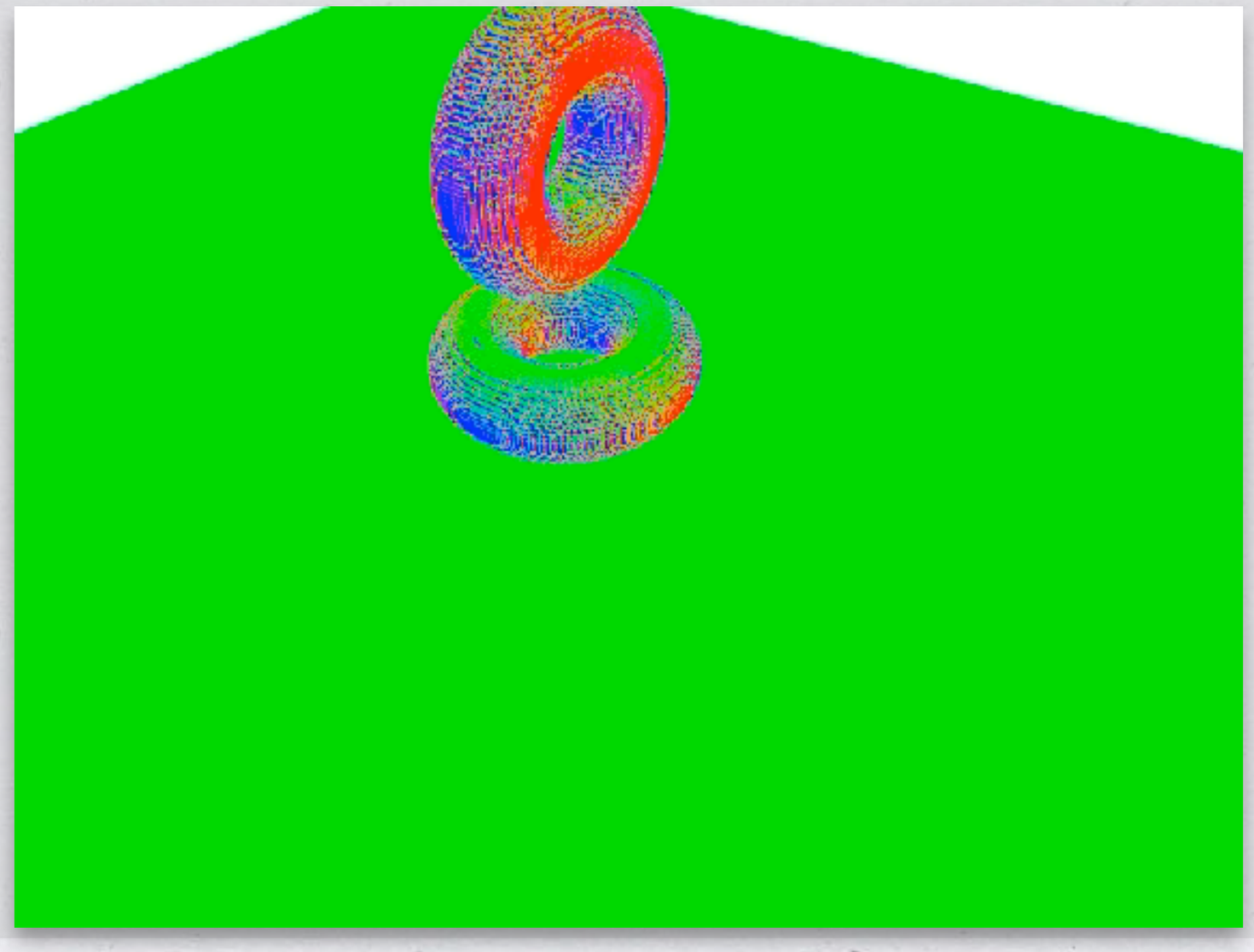
Classes d'algorithmes

* Calcul de la forme du volume d'intersection (calcul formel très difficile, possibilité d'approximation numérique)

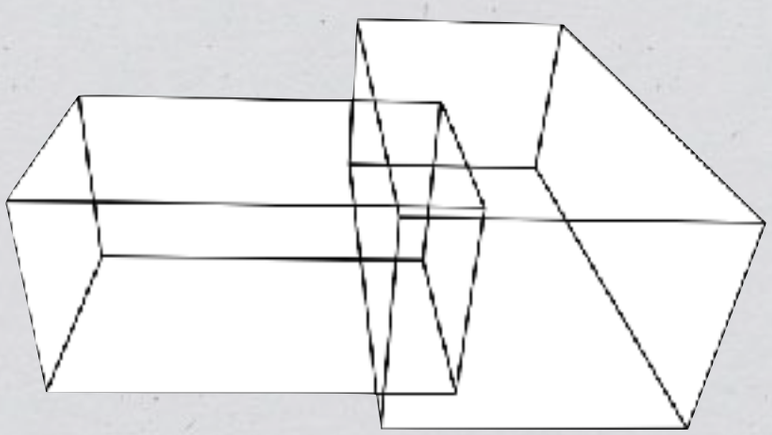


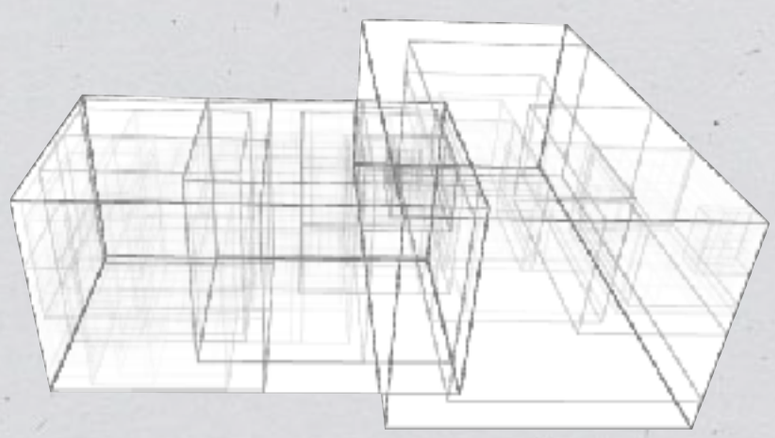
Classes d'algorithmes

* Exemple: approximation numérique sur GPU

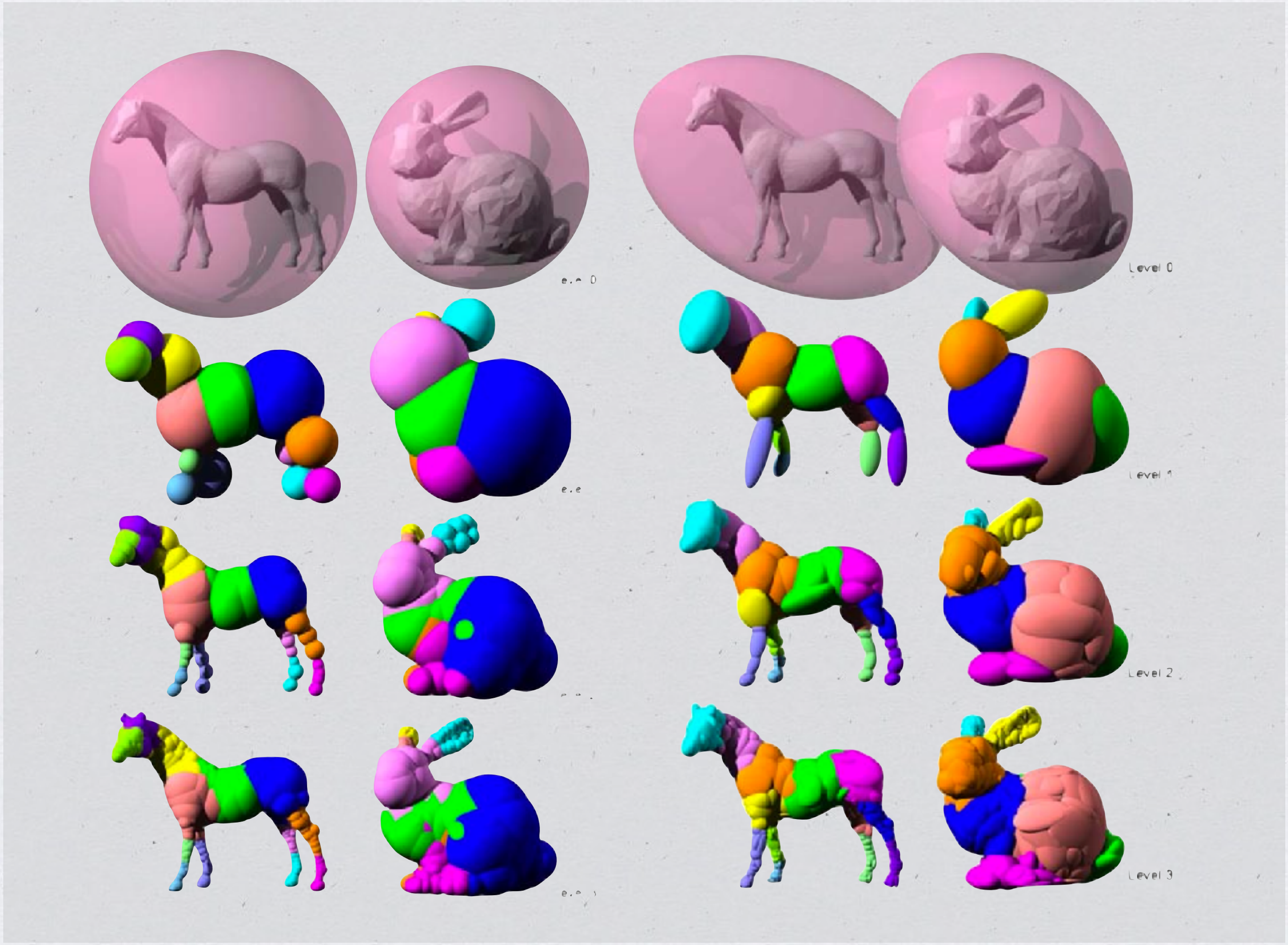


Acceleration des calculs



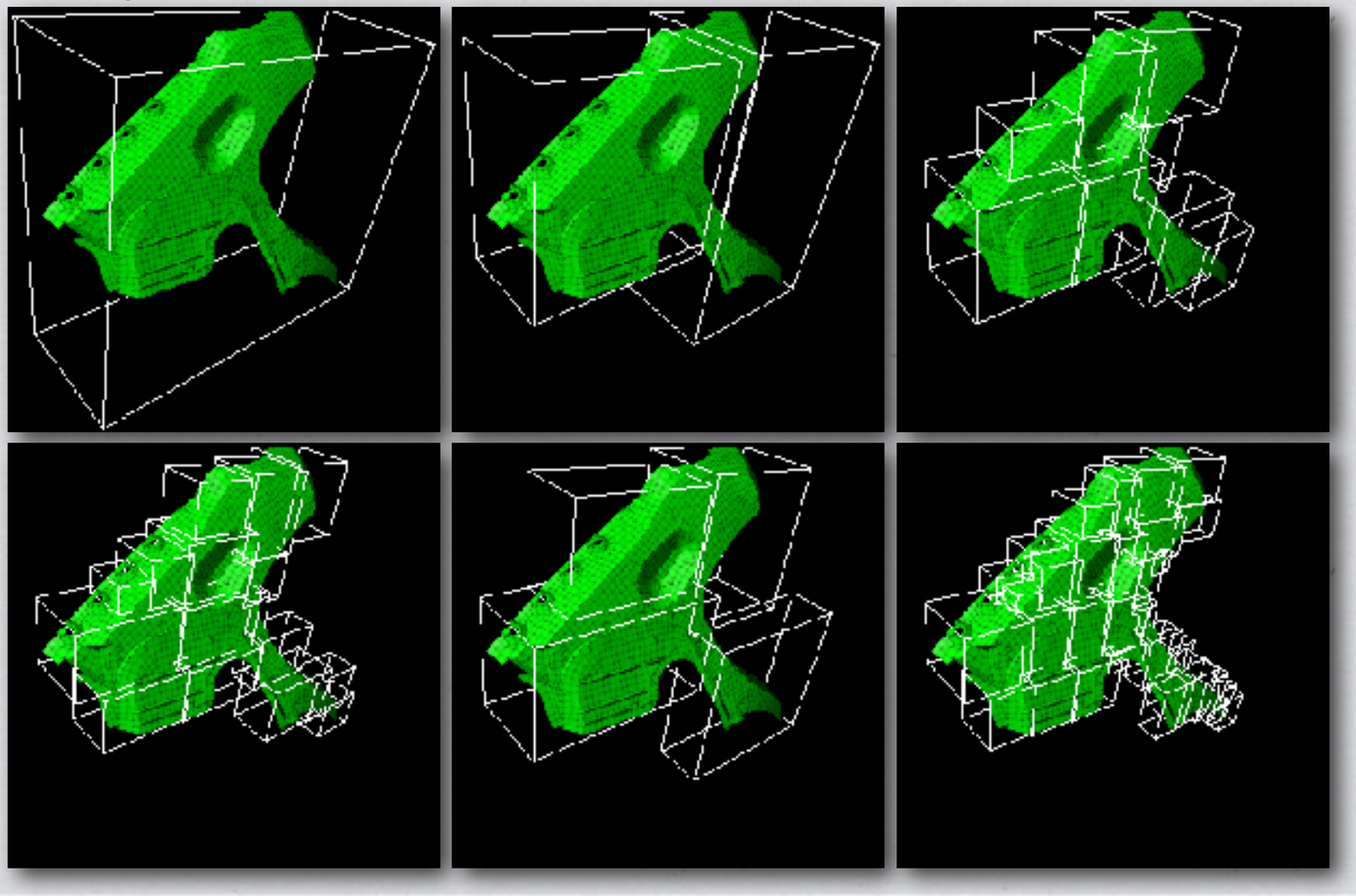


Broad Phase → **Narrow Phase** → **Exact Phase** →


Level 0
Level 1
Level 2
Level 3

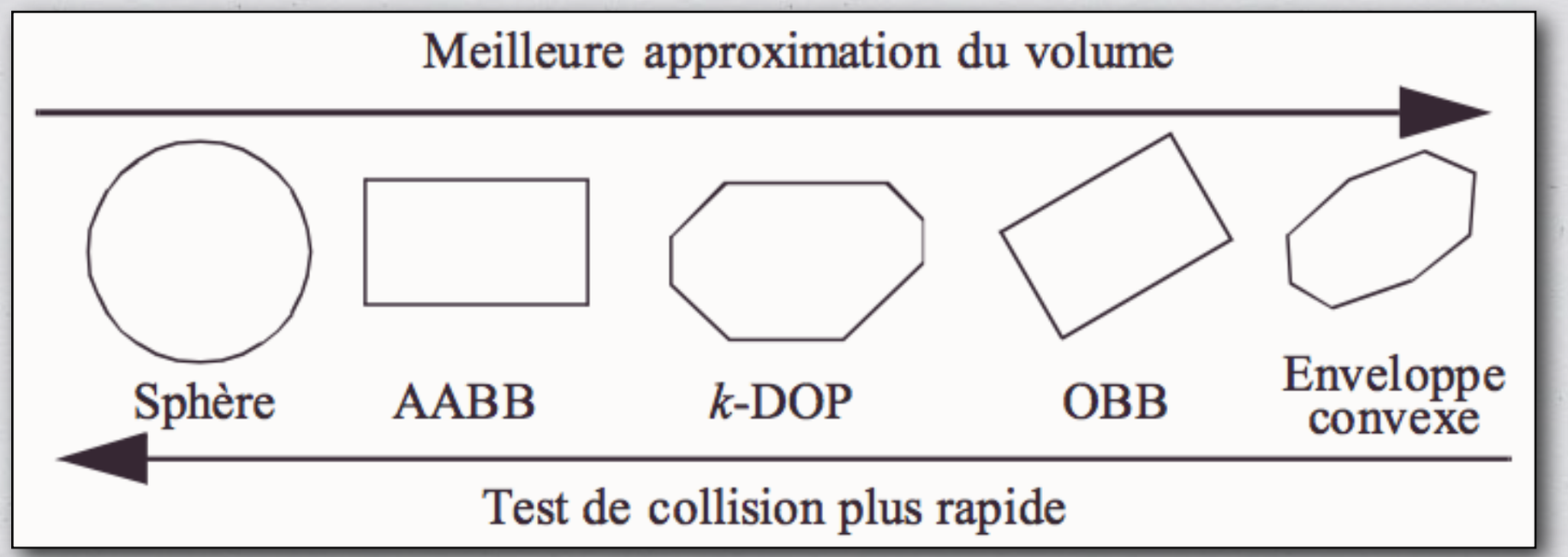
Narrow Phase

* Hiérarchie de Volumes Englobants (Bounding Volume Hierarchy)



Broad Phase

* Volume englobant. Principe: réaliser sur des volumes simples qui approchent la géométrie des objets

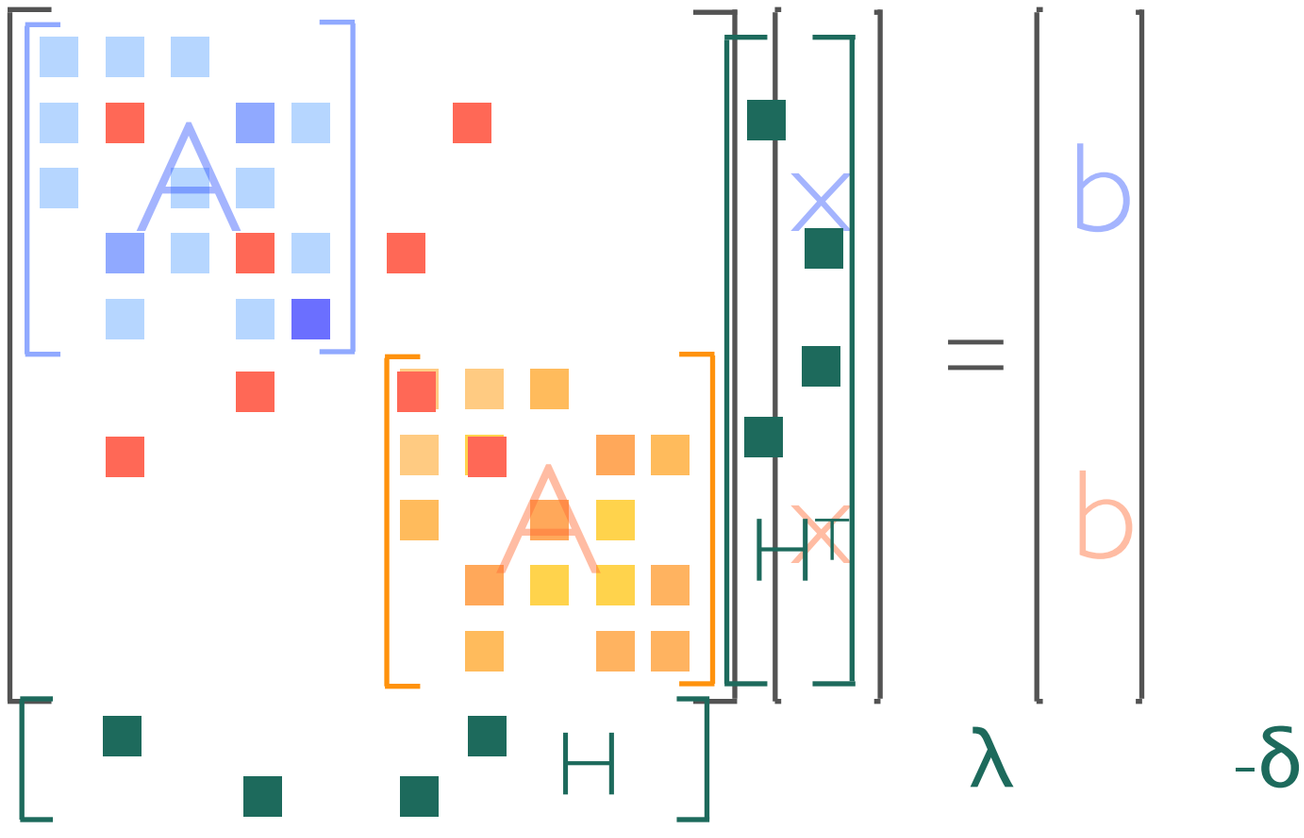


Bilan (rapide) de la détection de collision

- Etape la plus coûteuse d'une boucle de simulation, primordial d'optimiser
- Pas de panacée, dépend fortement de l'application (modélisation, nature des objets, type de réponse souhaitée)
- Prise en compte d'objets déformables empêche / limite certaines optimisations.
- Conséquence: état de l'art très vaste, difficile de benchmarker toutes les méthodes (études parfois contradictoires)

CHALLENGES

- Coupled equations (penalty, Lagrange multipliers)
- Conditioning problem
- Large Systems of equation



Direct approach

INTERACTION BETWEEN TWO DEFORMABLE MODELS

- Indirect approach
  - Free Motion
  - Constraint setting
  - Constraint solving
  - Constraint correction:
    $x = x^{free} + Dx$

$$\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} Dx^{free} \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} + \begin{bmatrix} H^T \end{bmatrix} \begin{bmatrix} \lambda \end{bmatrix}$$

$$\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} Dx \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} + \begin{bmatrix} H^T \end{bmatrix} \begin{bmatrix} \lambda \end{bmatrix}$$

$$0 \leq \delta^{free} + \left[\underbrace{\quad}_{W} + \underbrace{\quad}_{W} \right]^{-1} \perp \lambda \geq 0$$

[Compliance in constraint space]

Indirect approach

INTERACTION BETWEEN TWO DEFORMABLE MODELS

$$0 \leq \delta^{free} + \left[\quad W \quad + \quad W \quad \right] \lambda \qquad \perp \lambda \geq 0$$

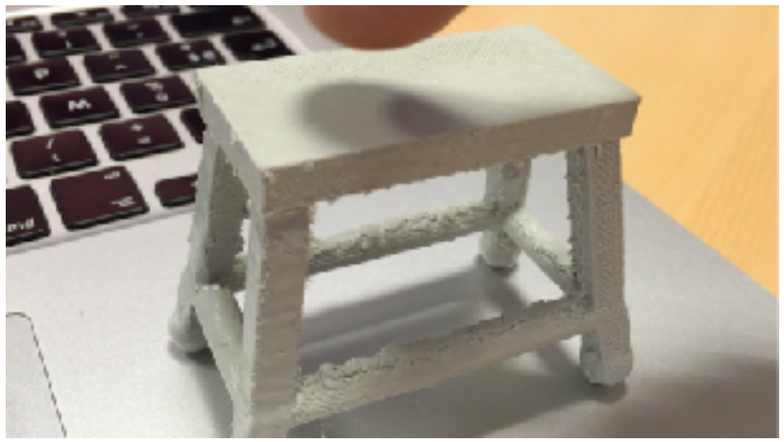
- Complementarity problem:
  - linear (no friction) / non-linear (friction)
- Iterative (block-Gauss-Seidel)

$$\underbrace{\delta_\alpha - \mathbf{W}_{\alpha\alpha}\ \lambda_\alpha}_{\text{unknown}} = \underbrace{\sum_{\beta=1}^{\alpha-1} \mathbf{W}_{\alpha\beta}\ \lambda_\beta + \sum_{\beta=\alpha+1}^{m} \mathbf{W}_{\alpha\beta}\ \lambda_\beta + \ \delta_\alpha^{\text{free}}}_{\text{frozen}}$$

  - Block contact+friction (no simplification of friction cone)
  - Generic implementation: works for **other interactions**
  - Slow convergence (importance of warm start…)

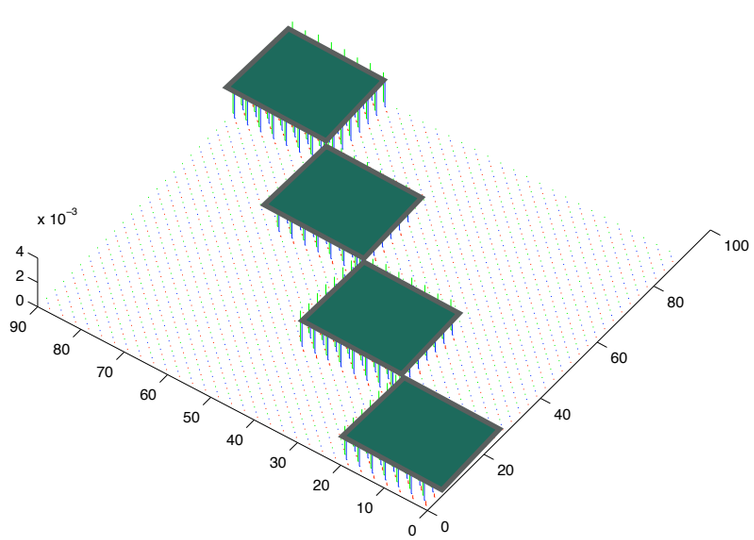
COMPLIANCE IN CONSTRAINT SPACE

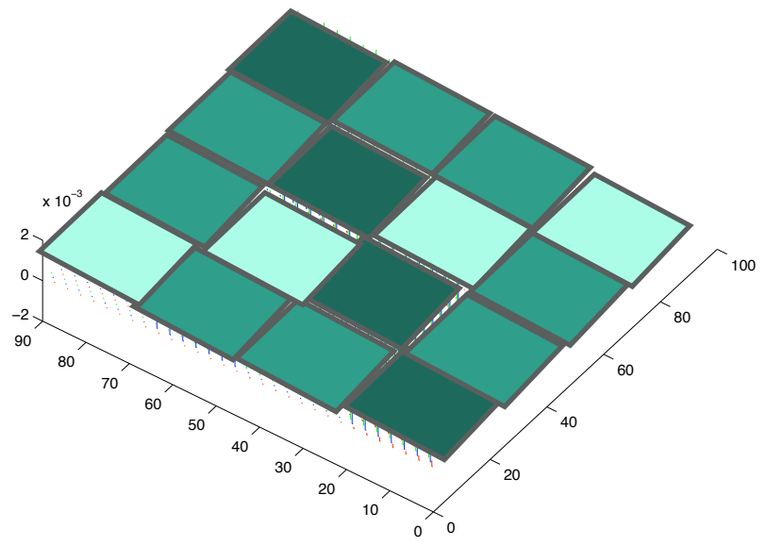
- What represents W ?
  - Matrix in the contact space
  - **Mechanical coupling between constraints**
  - Table example



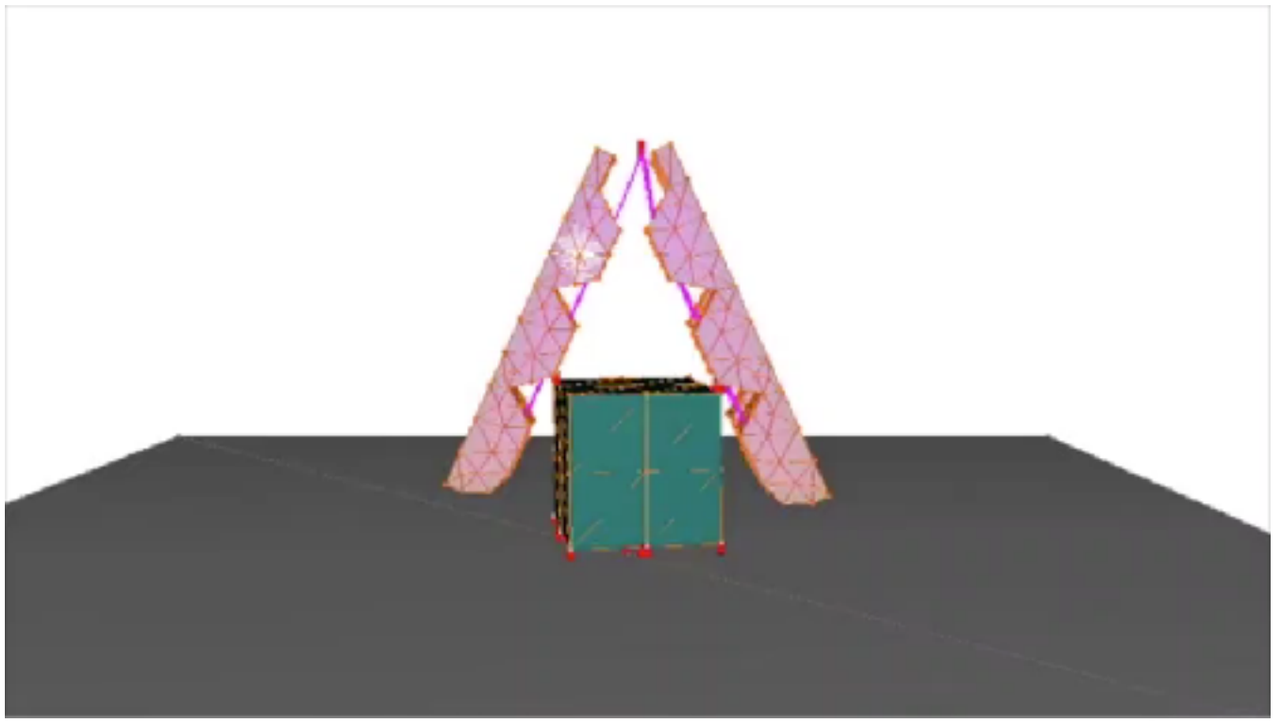
$$\underbrace{h^2 \left[\mathbf{H}_i \mathbf{A}^{-1} \mathbf{H}_j^T\right]}_{\mathbf{W}_{ij}}$$

- 2 issues for the computation time:
  - size of A
  - size of H



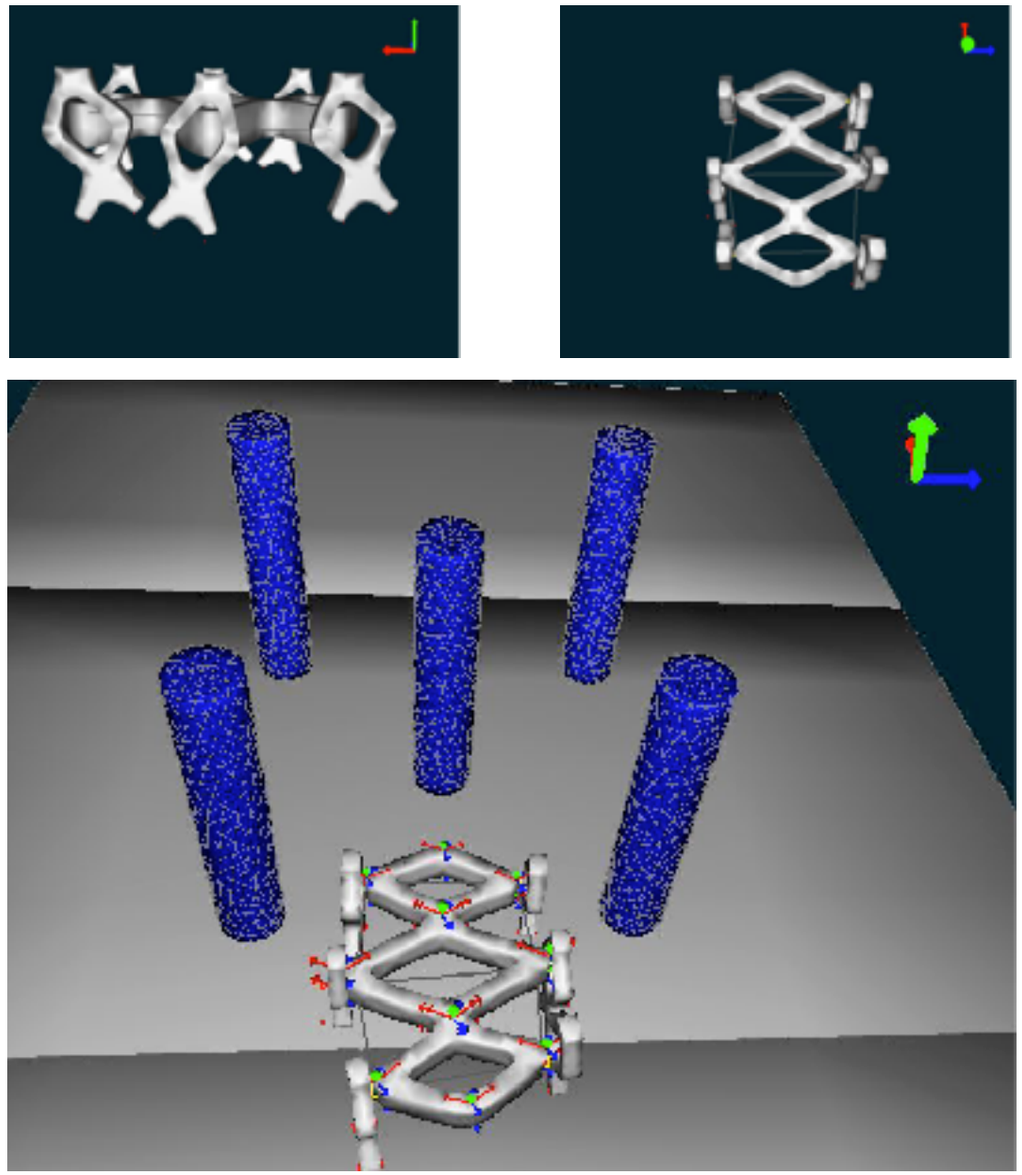


EXAMPLES



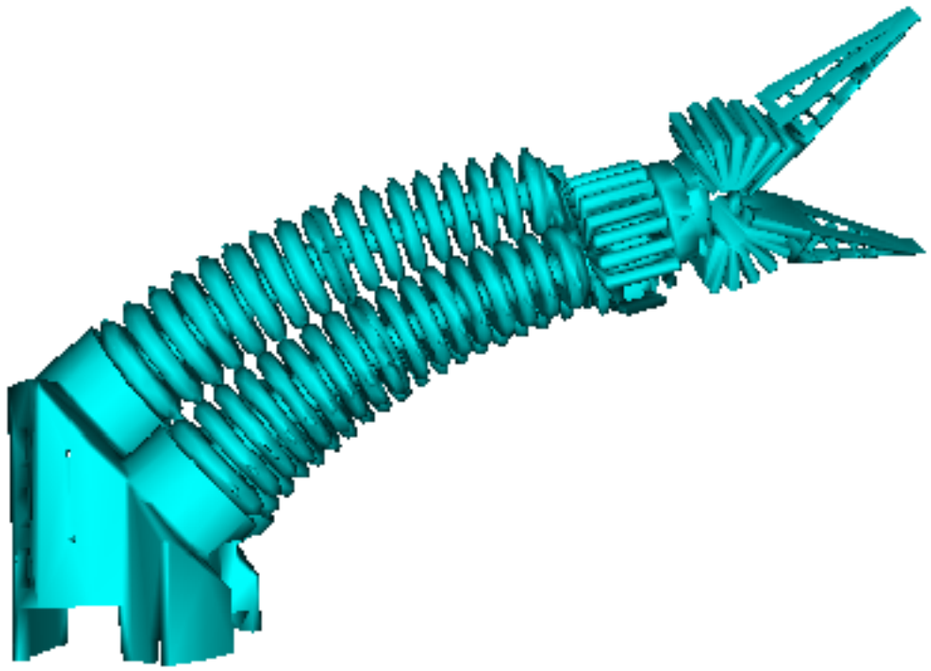
Speed x3

Grasping simulation

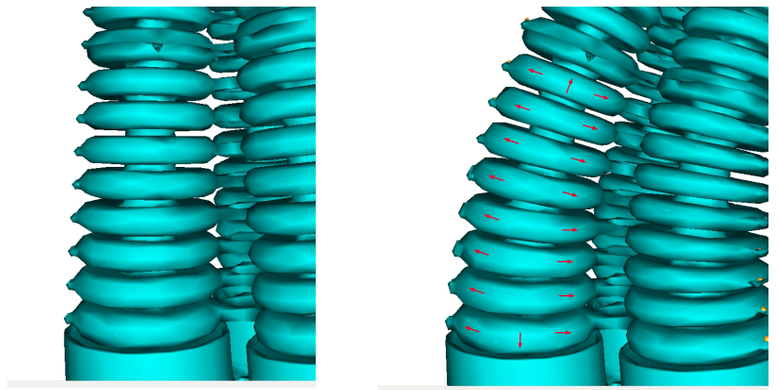


Locomotion

CABLE AND PRESSURE

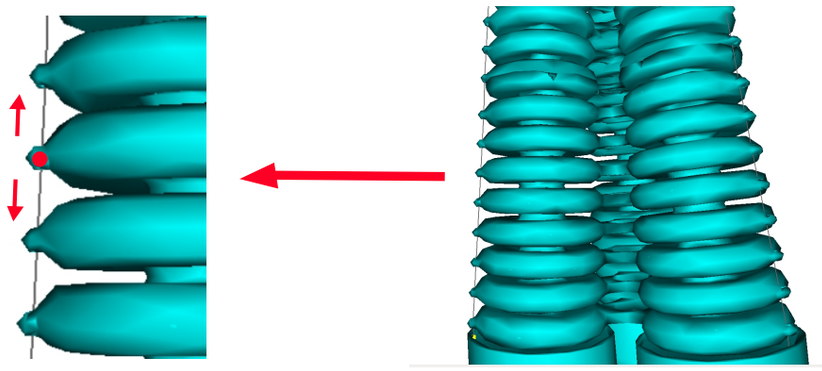


Festo deformable trunk



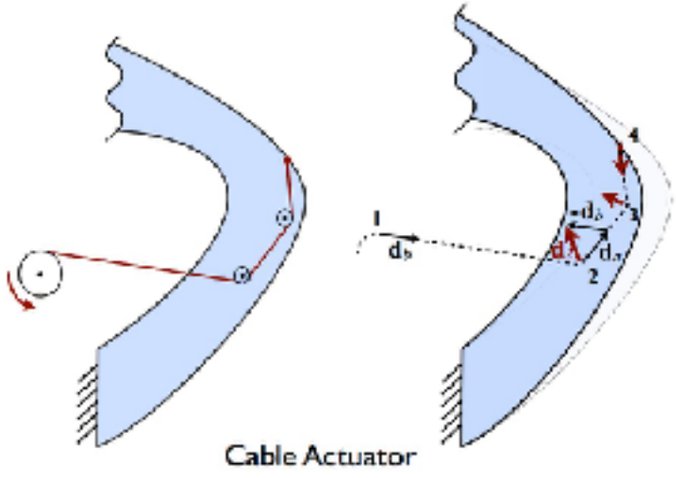
δ = volume change

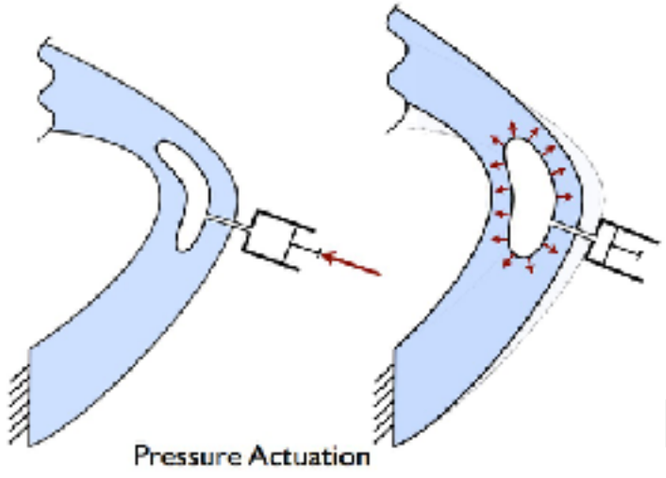
Pressure actuation



δ = length change

Cable actuation



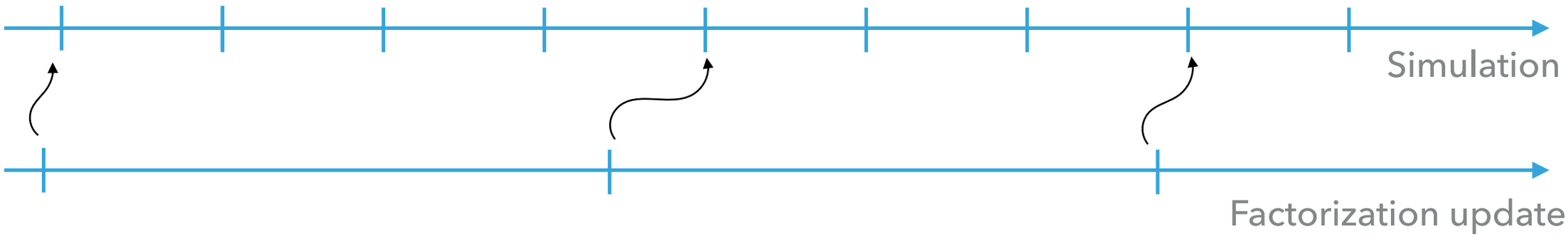


$\mathbf{H}^T \lambda$ ← cable force pressure

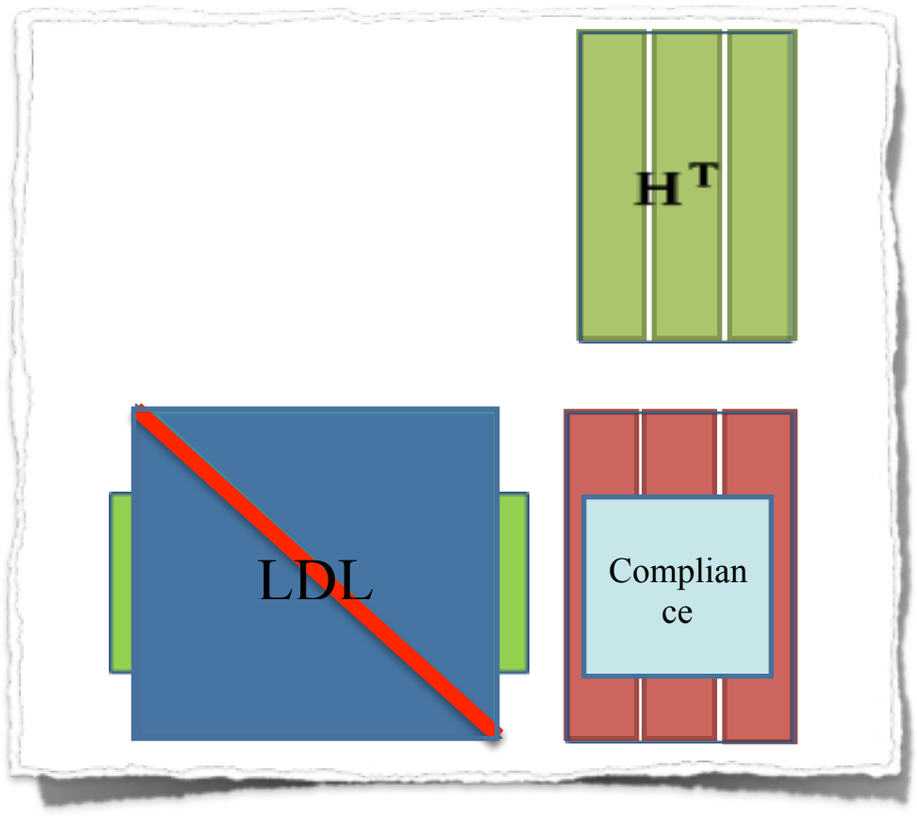
Force distribution

COMPLIANCE IN CONSTRAINT SPACE IN REAL-TIME

- The compliance matrix can be approximated by preconditionning technique:



$$W \approx h^2 H (LDL)^{-1} H^T$$

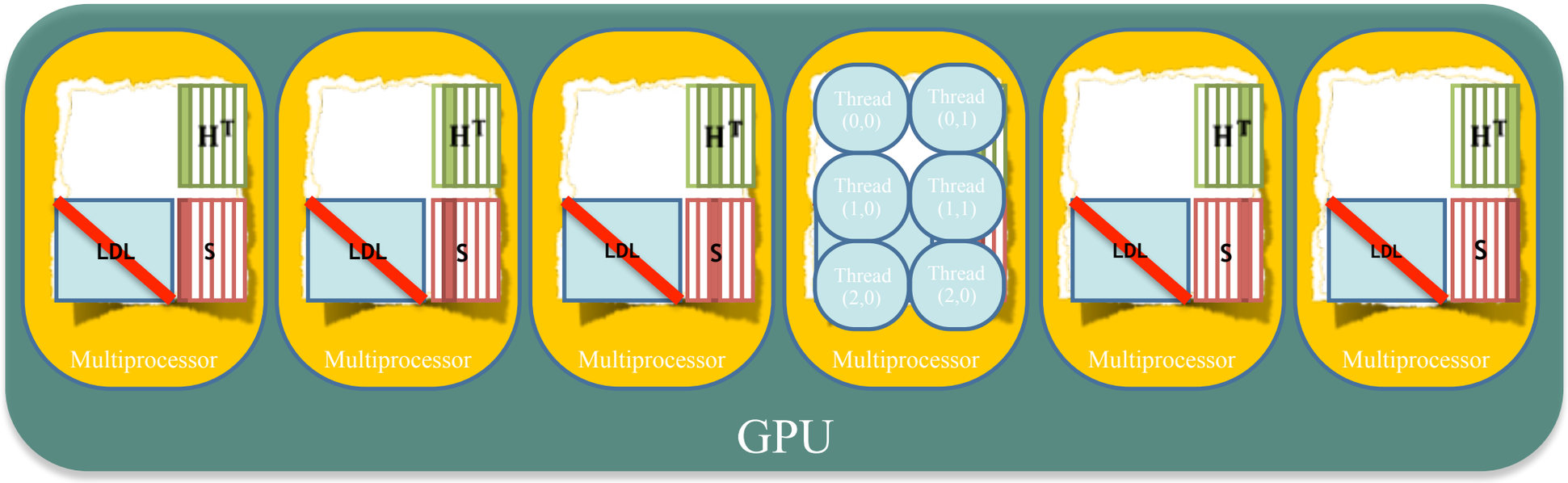


- Repeat this operation, until get $S = (LDL)^{-1} H^T$
- Finally, obtain the compliance matrix with:

$$W = H\,(LDL)^{-1} H^T = H\,S$$

COMPLIANCE IN CONSTRAINT SPACE IN REAL-TIME

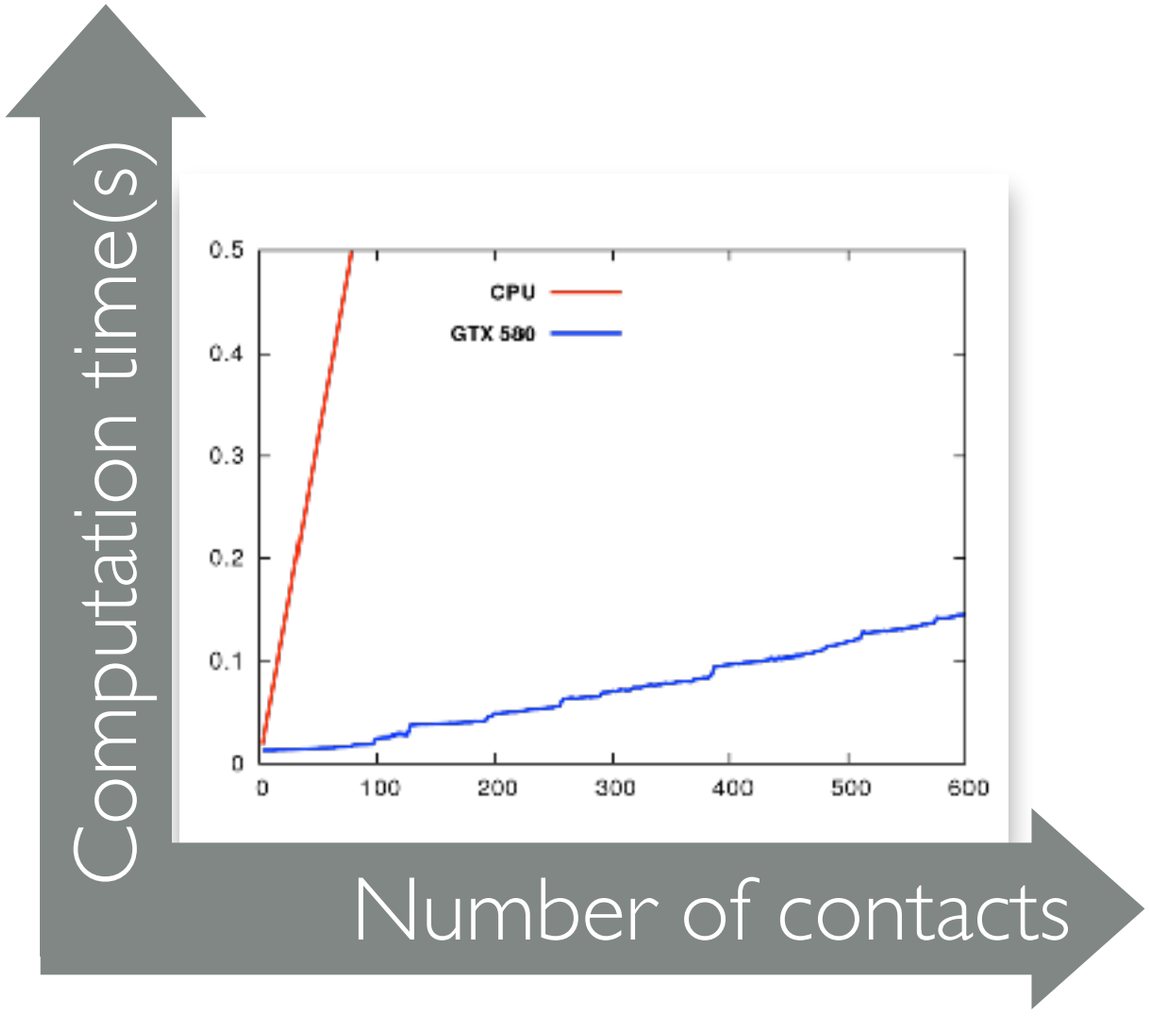
- The most expensive task is the computation of S
- But each column can be solved in parallel

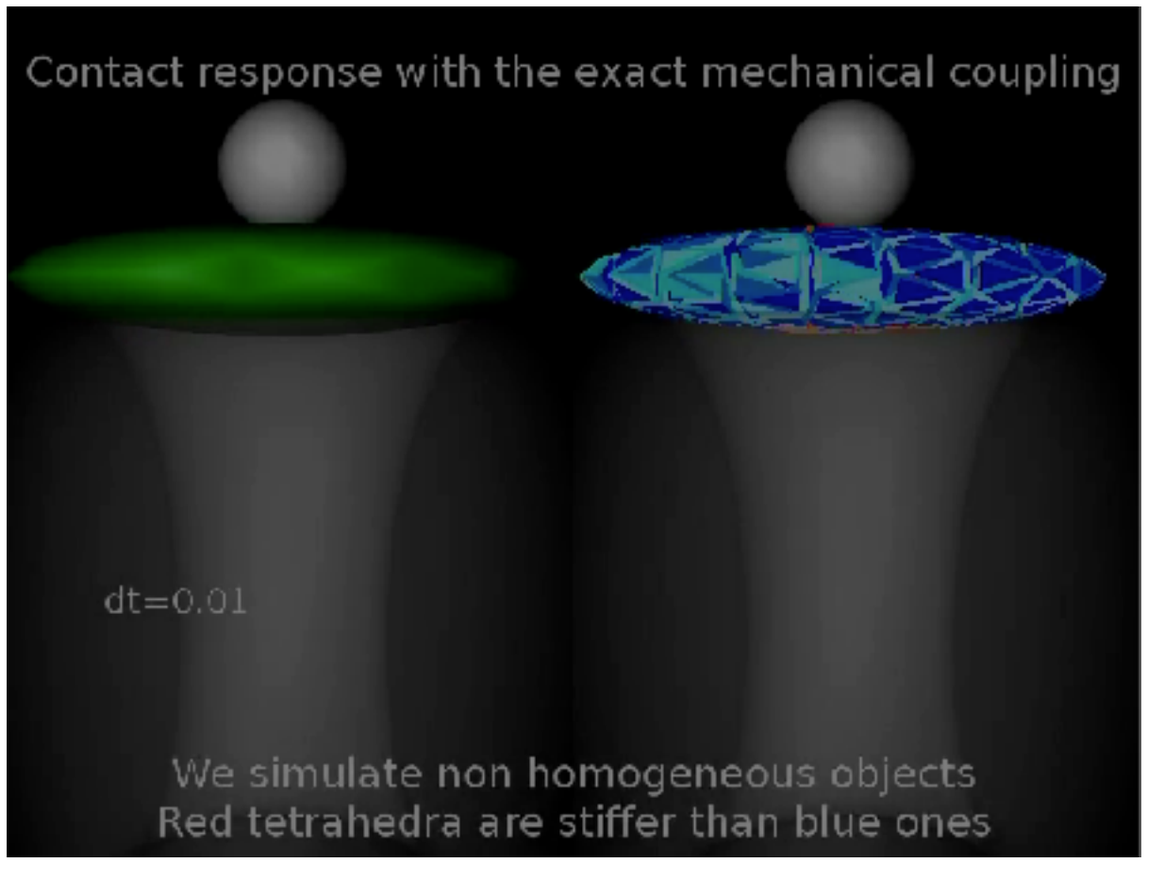


- Each resolution is performed by a single multiprocessor on GPU
- Second level of parallelism within each linear system resolution

COMPLIANCE IN CONSTRAINT SPACE IN REAL-TIME

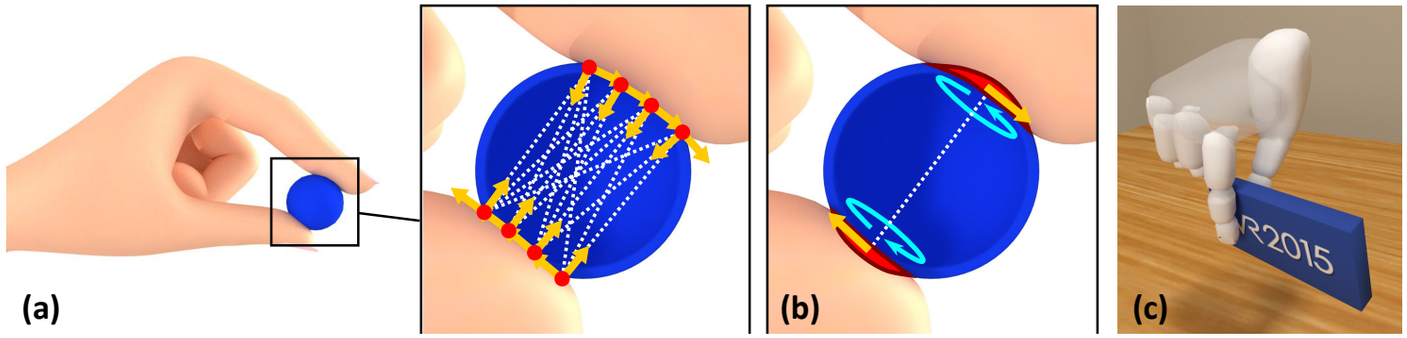
- Validation on solid with inhomogeneities
  - The green shape is the reference / blue & red mesh uses asynchronous precon.
  - Red tetrahedra are stiffer than blue ones.
- High speed-up of the computation time for W



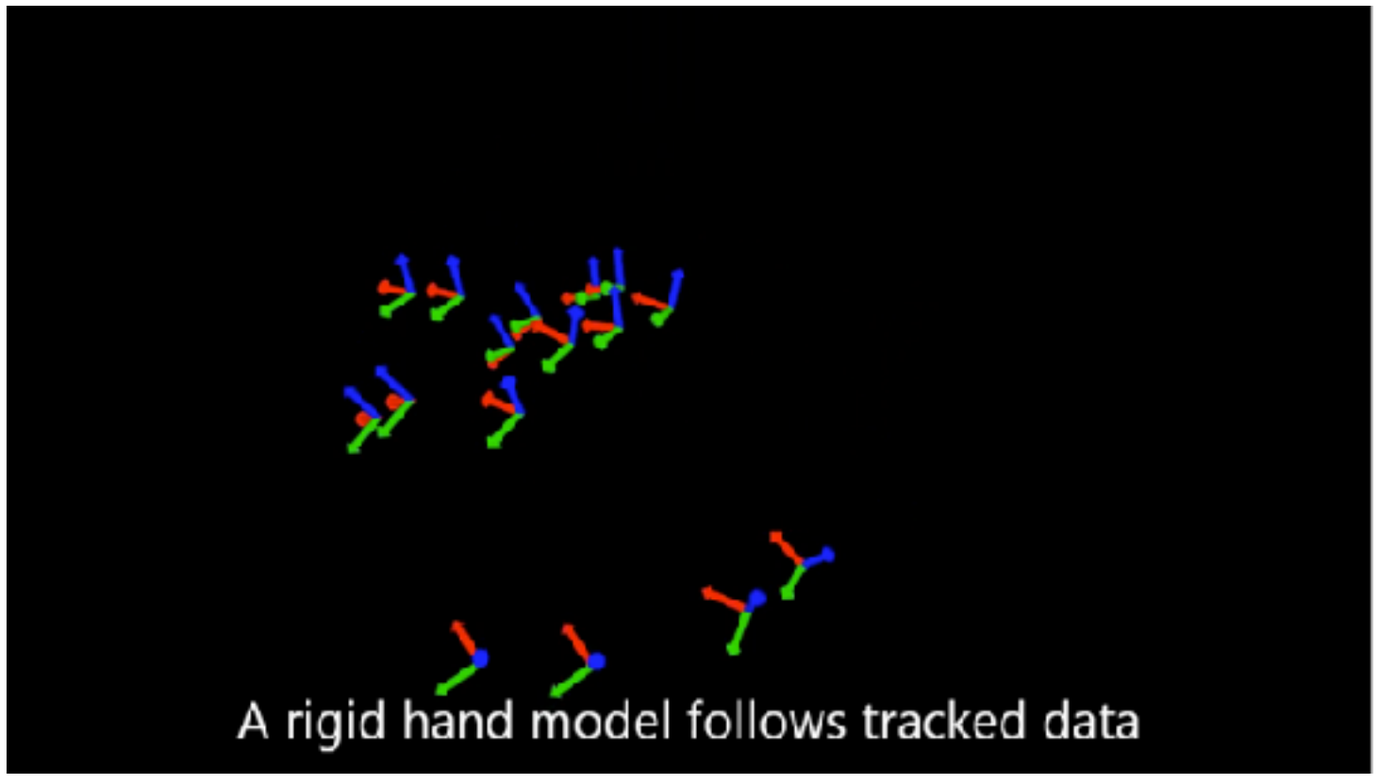


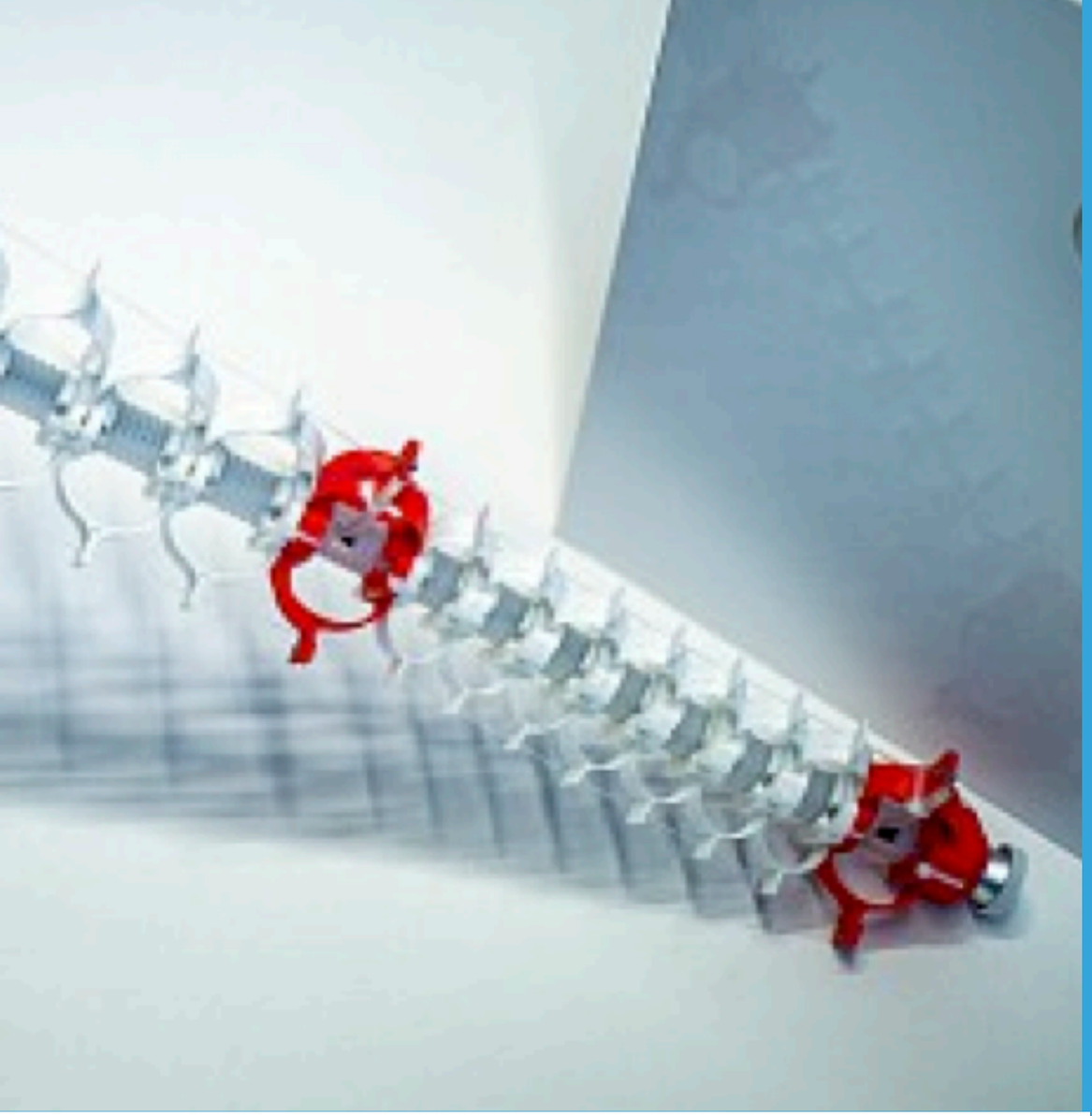
Courtecuisse *et al.* MICCAI 2011

AGGREGATE CONSTRAINTS

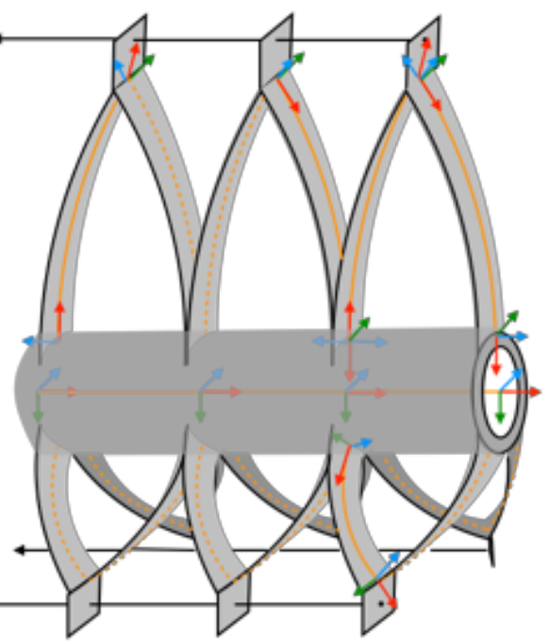


- Goal: reduce the size of W !!
  - Aggregate the contact constraints in the same zones
  - Contact distance vs. contact volume !
- Application to grasping and manipulation:

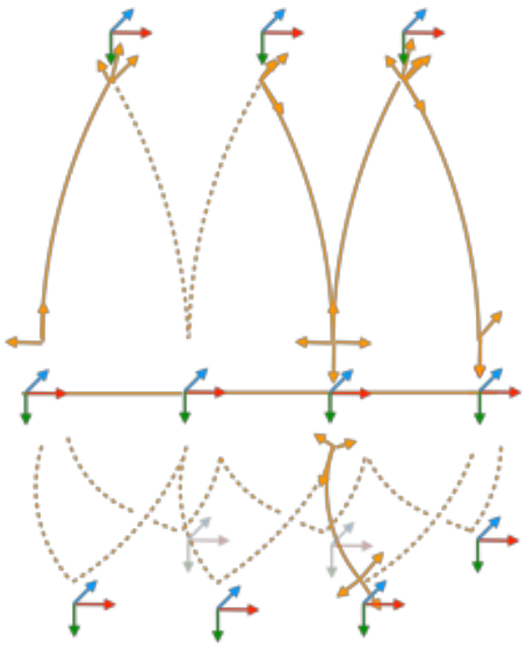




SOFT ROBOTIC SIMULATIONS AND CONTROL

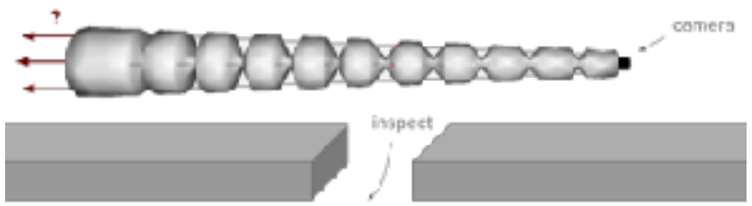


(a)

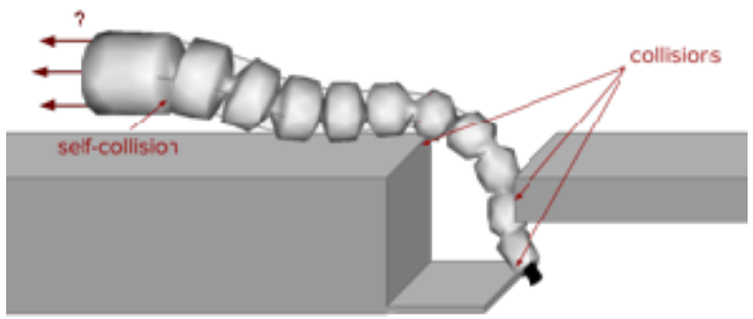


(b)

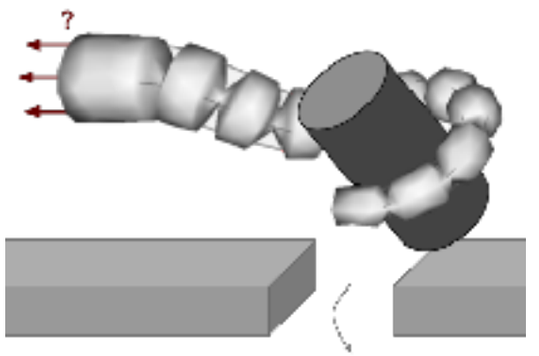
THREE TYPES OF TASKS



1. Inverse kinematics of soft robots



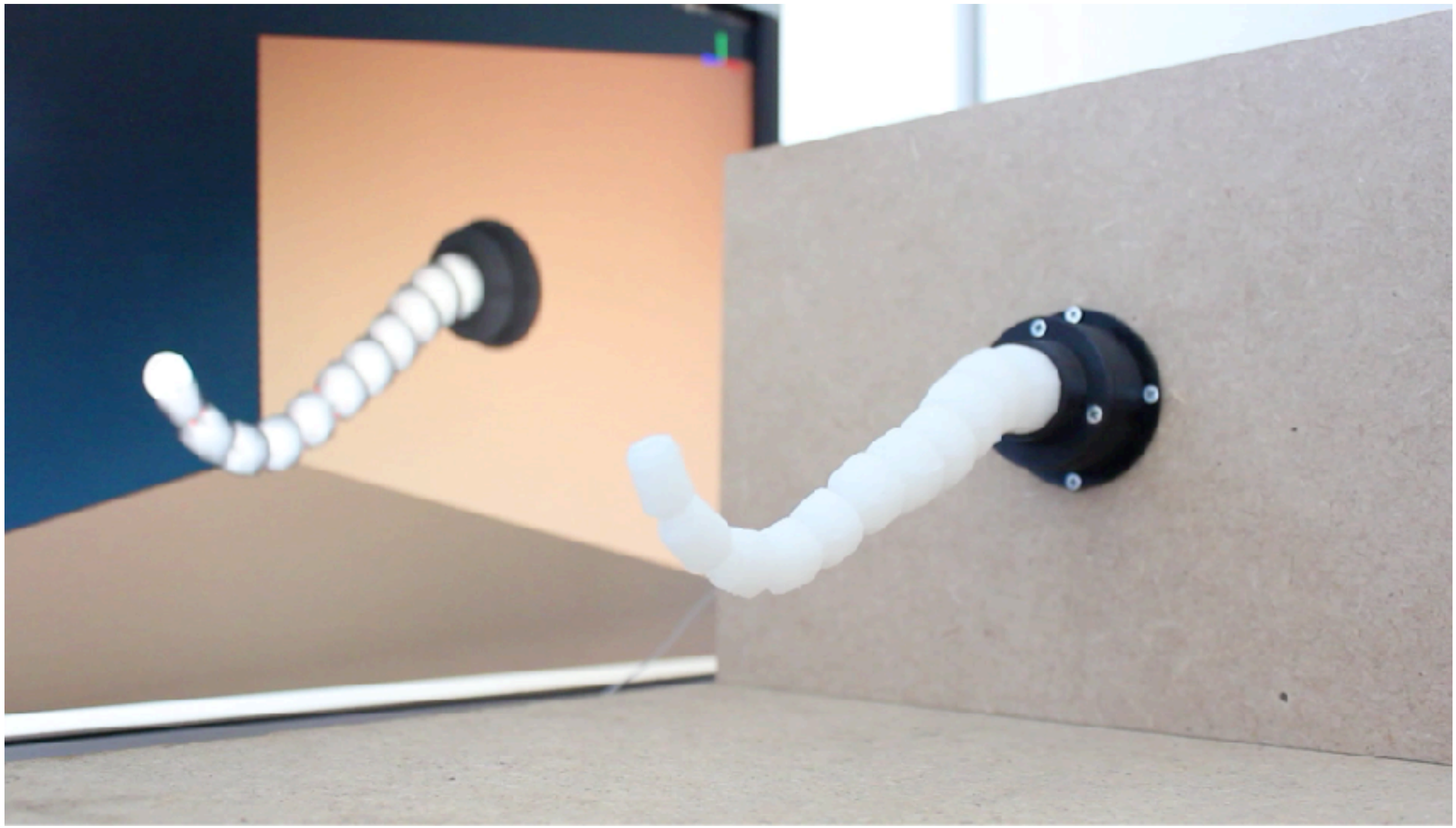
2. IK with contact handling



3. IK with stick contact handling

EXPERIMENTS

	#DoFs	#Elem	W	QP	Sim.
trunk	2127	1972	2.8 *ms*	< 0.1 *ms*	24.1 *ms*



EXPERIMENTS

	#DoFs	#Elem	#Conct.	W	QP(s)	Sim.
trunk	2127	1972	51	10.09 *ms*	4.91 *ms*	34.52 *ms*

EXTENSION TO STICK CONTACT

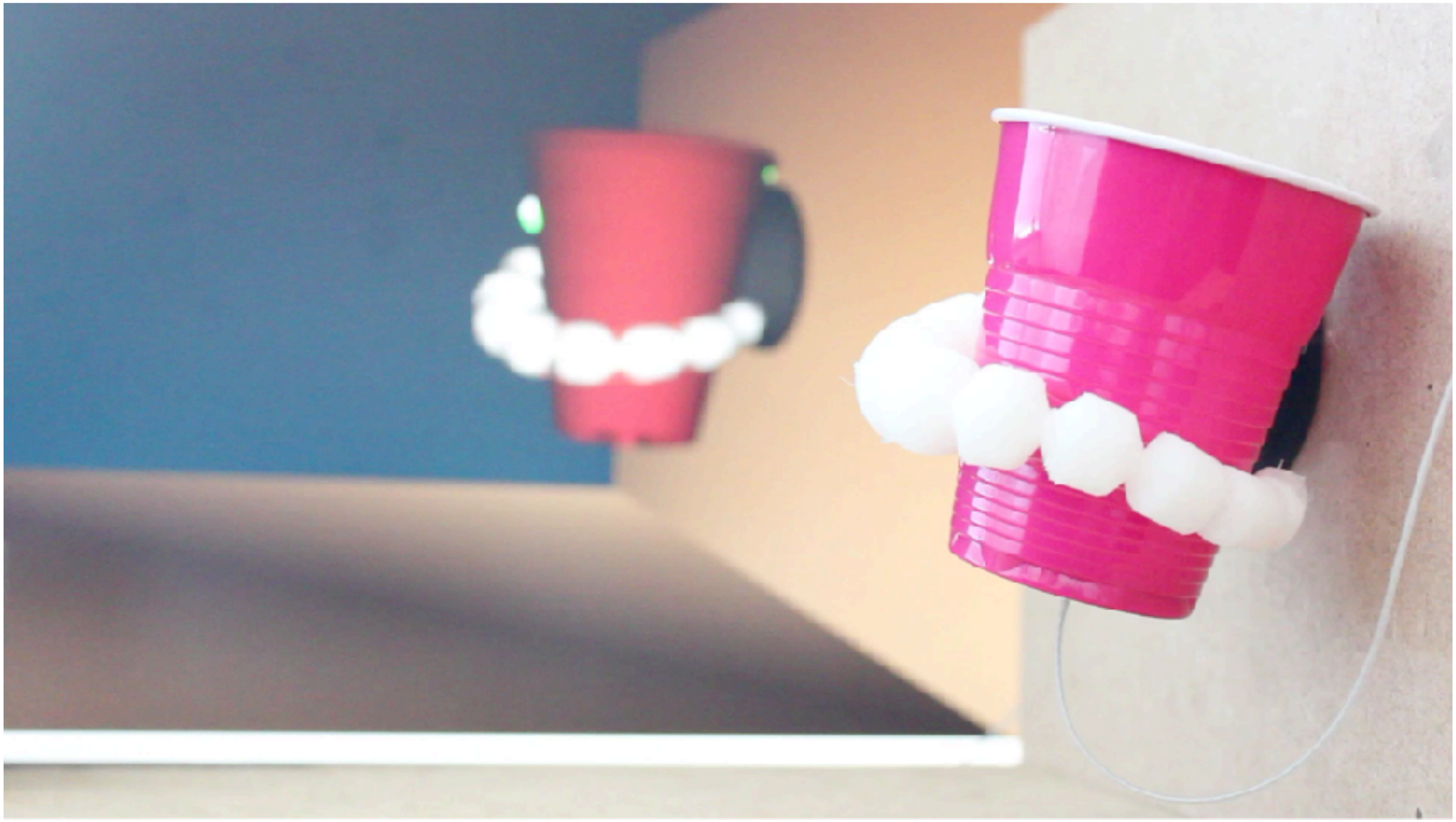
- Grasping

	#DoFs	#Elem	#Conct.	W	QP(s)	Sim.
trunk	2127	1972	81	76.5 *ms*	35.5 *ms*	185.1 *ms*
cup	3765	3800				

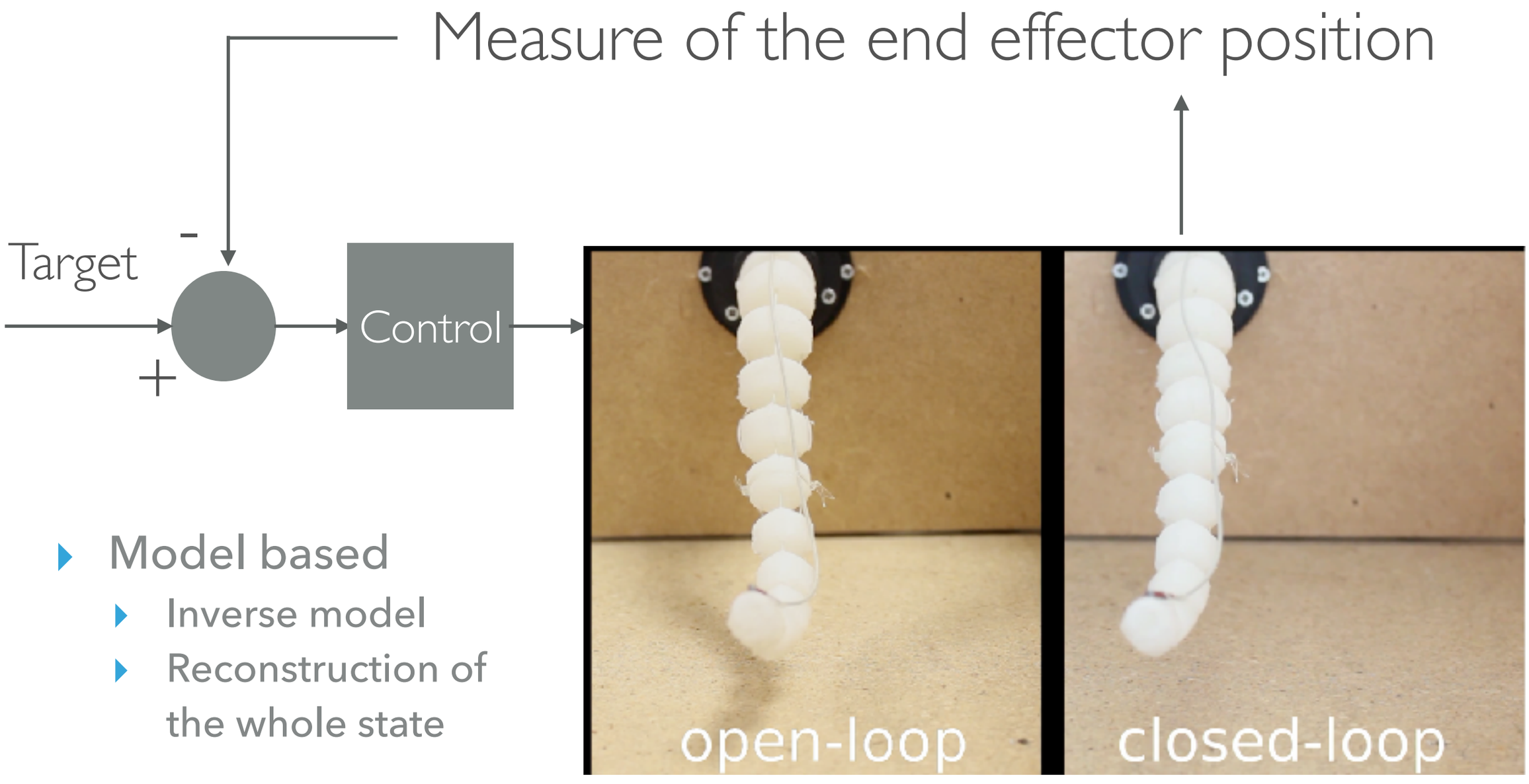
EXTENSION TO STICK CONTACT

- Grasping

	#DoFs	#Elem	#Conct.	W	QP(s)	Sim.
trunk	2127	1972	81	76.5 *ms*	35.5 *ms*	185.1 *ms*
cup	3765	3800				



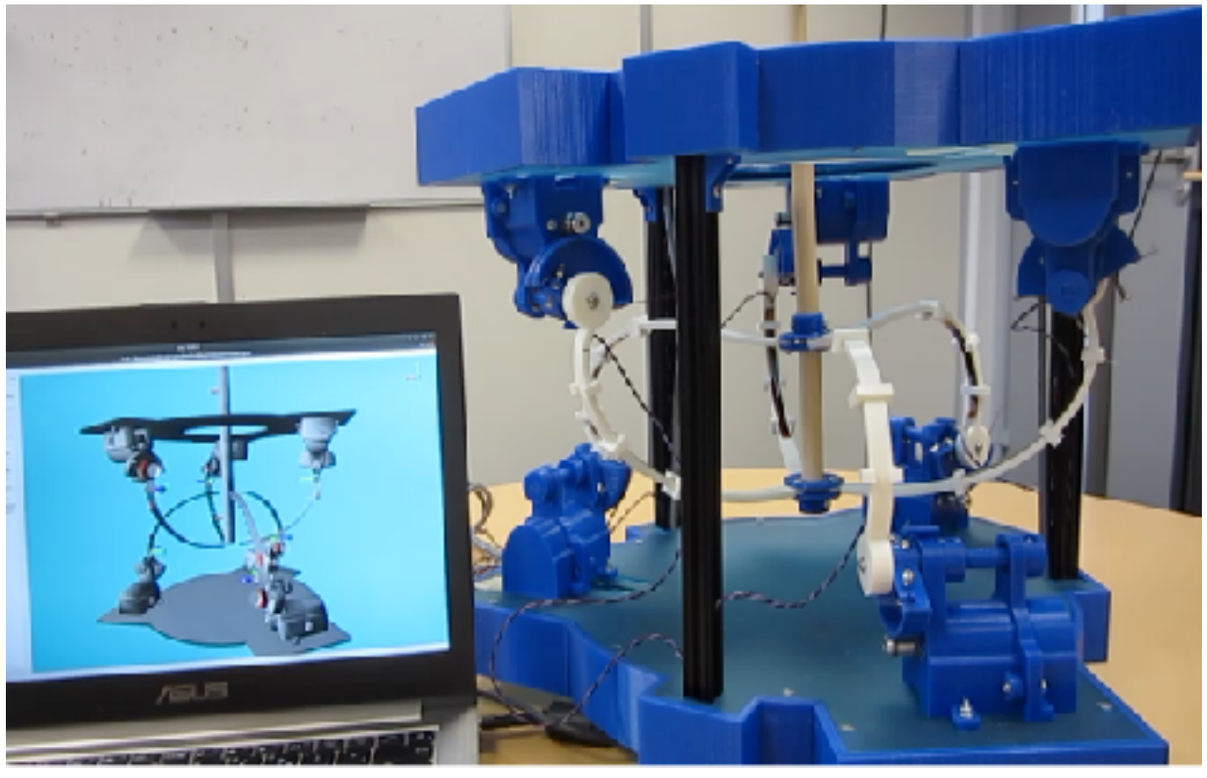
FEEDBACK CONTROL



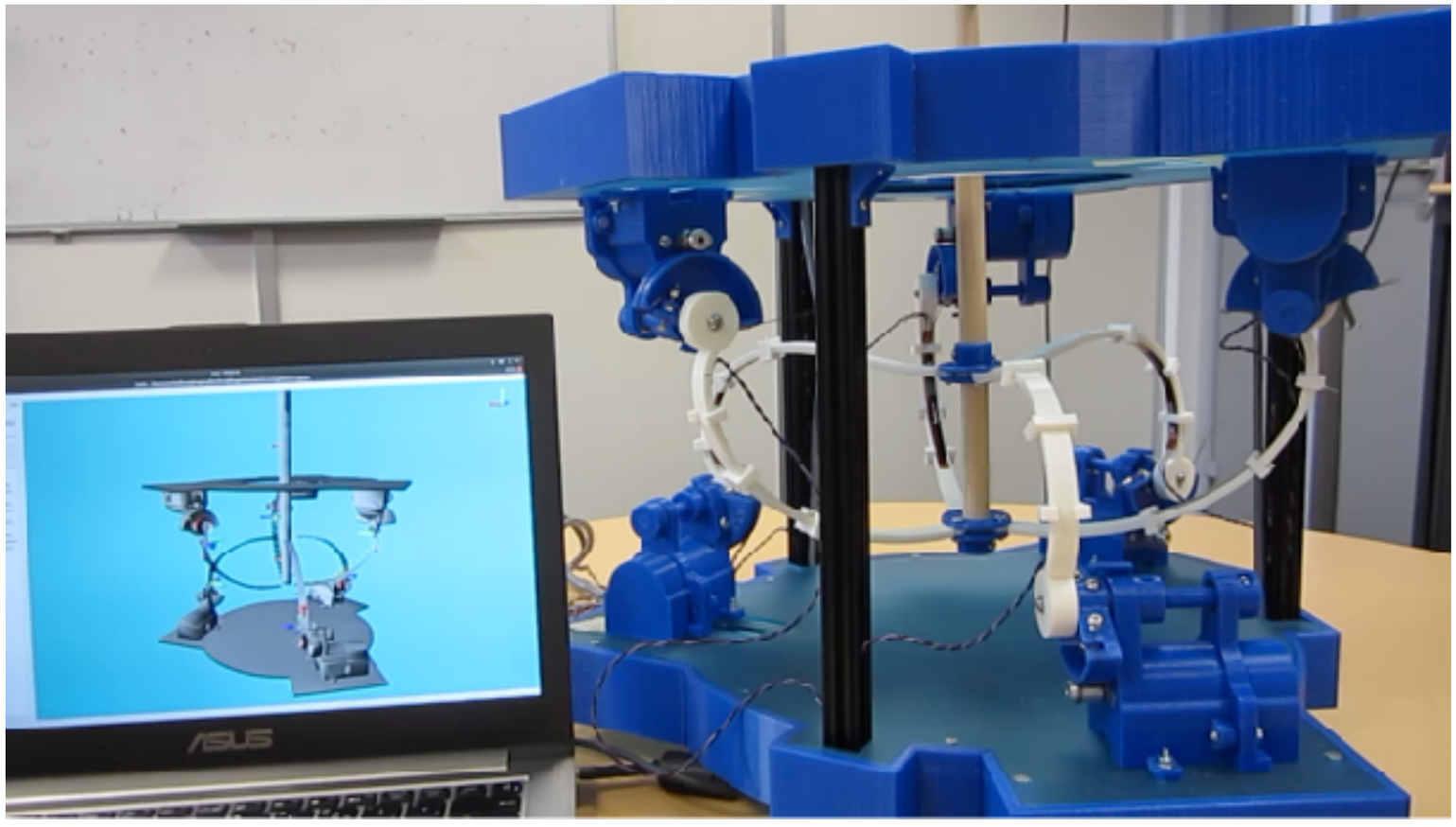
- Model based
  - Inverse model
  - Reconstruction of the whole state

NEW DESIGN WITH 6 MOTORS & SENSING

- 5-DOF end-effector (no twist)
- 6 motors, capstan drives, attached to the end-effector via deformable beams
- Position control on each motor
- High resolution encoders
- Flex sensors on each beam
- Communication between motor controller and high rate thread (CAN), ~500 Hz

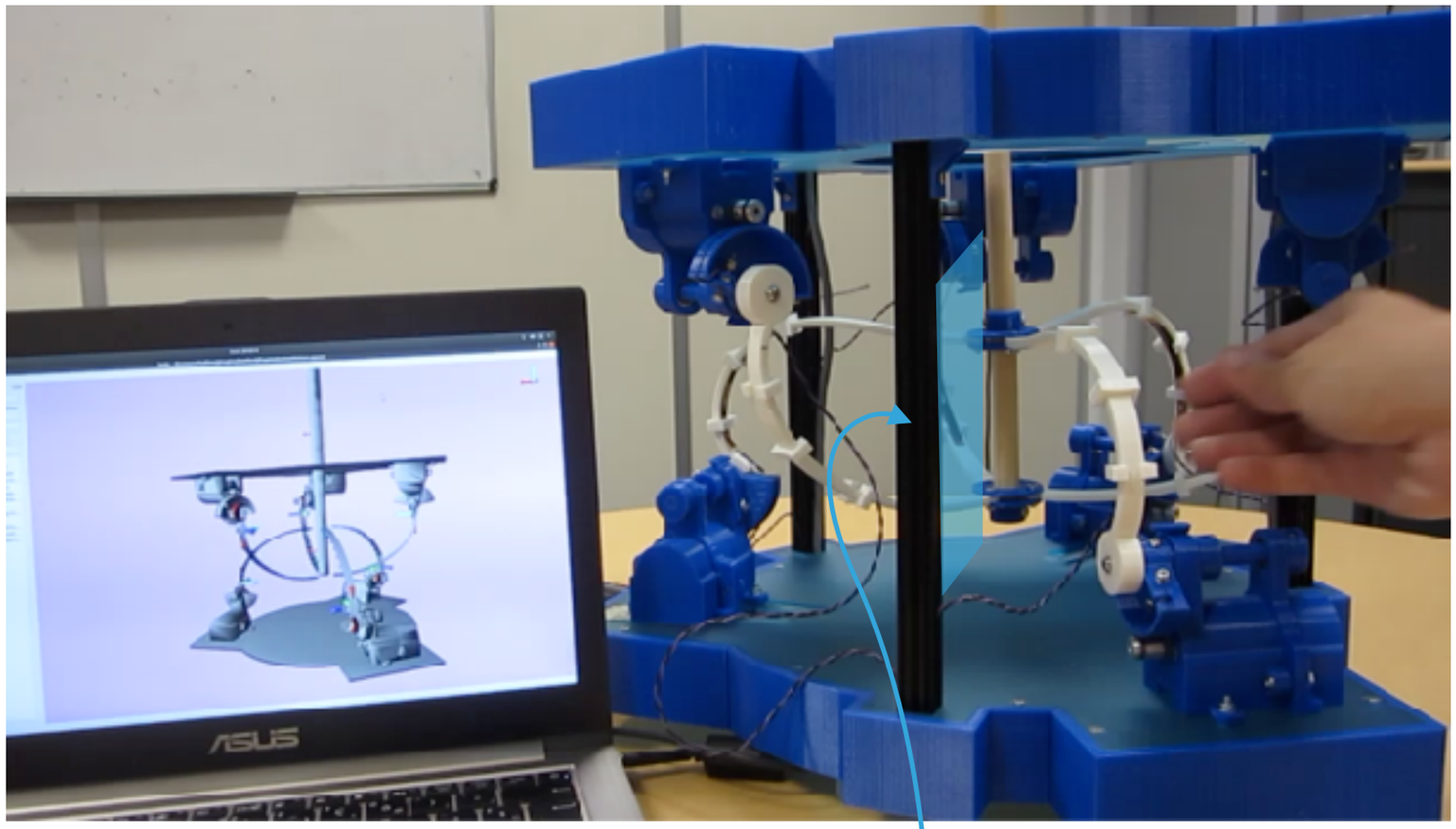


END-EFFECTOR SENSING



PROMISING FIRST TESTS

- Use as an haptic device…



END...

christian.duriez@inria.fr