COURS MASTER IVI

HAPTIC RENDERING & SIMULATION SOFTWARE ARCHITECTURE

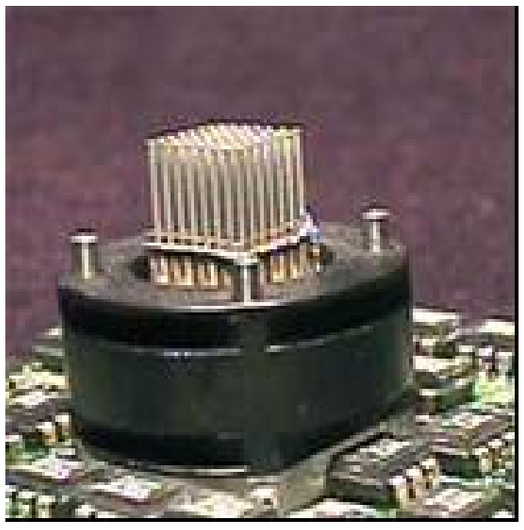
IMPORTANCE & PARTICULARITY OF HAPTIC FEEDBACK

- Visual & Audio channels = unidirectional information
- Haptic modality = bidirectional
  - Information exchange
  - Energy exchange

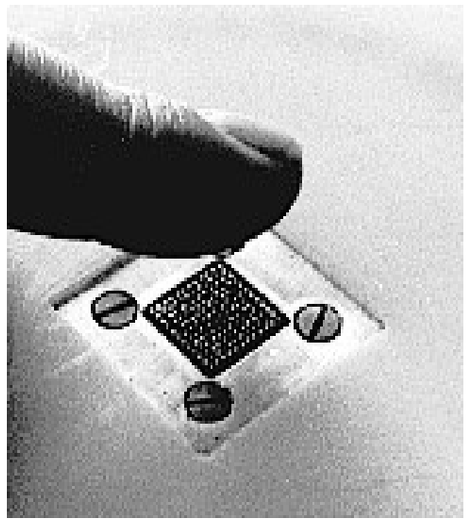
L'Incrédulité de saint Thomas, Le Caravage

HAPTIC = TACTILE & KINESTHESIS

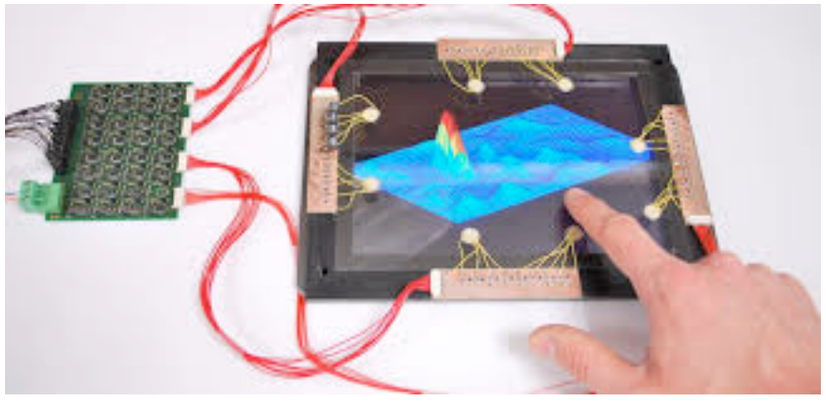
- Tactile devices…



STReSS tactile display



http://newton.ex.ac.uk/research/biomedical/tactile/



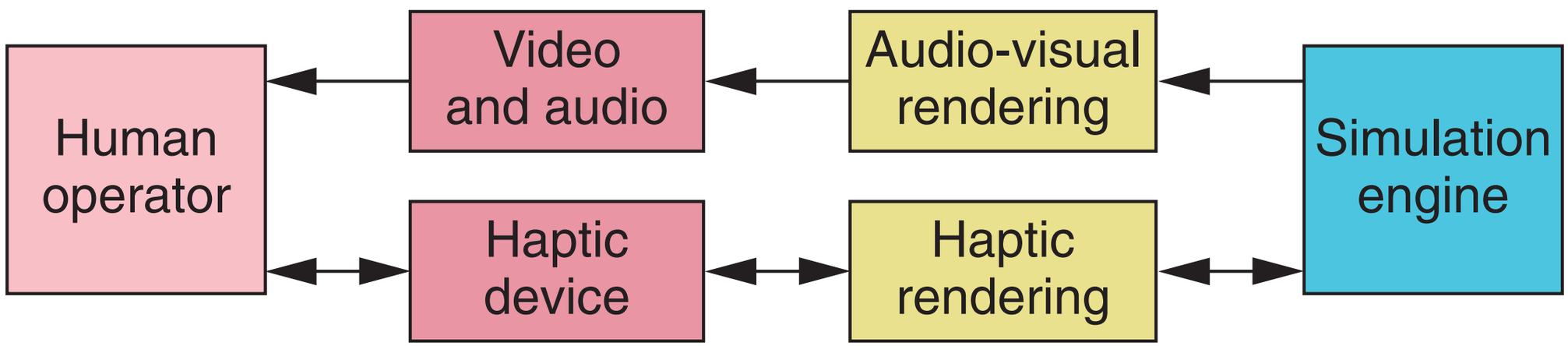
CEA

HAPTIC: FROM REAL TO VIRTUAL ENVIRONMENTS

- Teleoperation with haptic feedback

ARCHITECTURE FOR HAPTIC FEEDBACK

- Simulation engine
- Visual and haptic rendering algorithms
- Transducers



TOPICS

- Haptic interface devices & control basics
- Haptic Algorithms
- Software architecture for real-time simulation

HAPTIC INTERFACE DEVICES & CONTROL BASICS

HAPTIC INTERFACE DEVICES

- Robotic reversible (serial) arms

3dofs / 6dofs

6dofs

HAPTIC INTERFACE DEVICES

- Parallel architecture

3dofs

6dofs (+1)

HAPTIC INTERFACE DEVICES

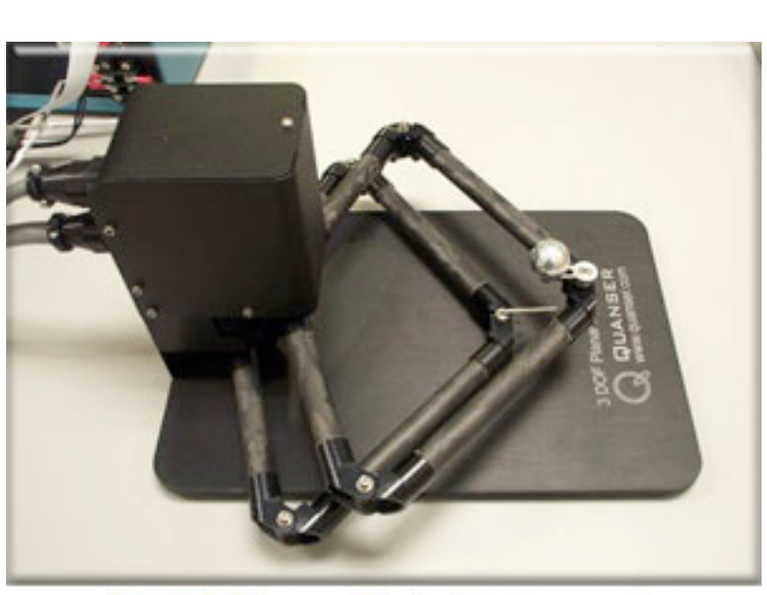
- Parallel architecture

HAPTIC INTERFACE DEVICES

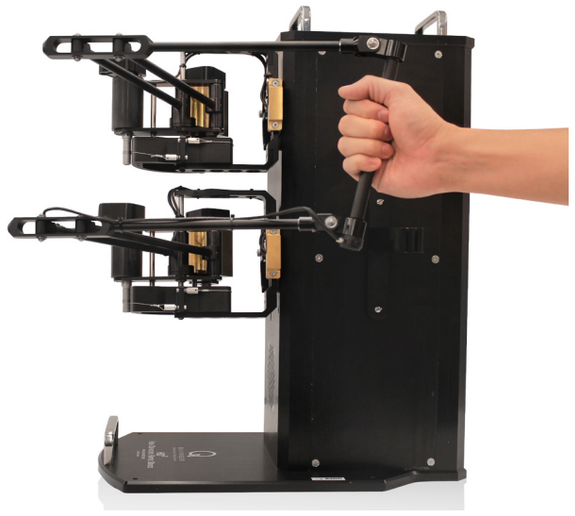
- Serial arms
  - large workspace, « simple » models
  - low stiffness
- Parallel robots
  - stiff and light robots
  - complex model and workspace

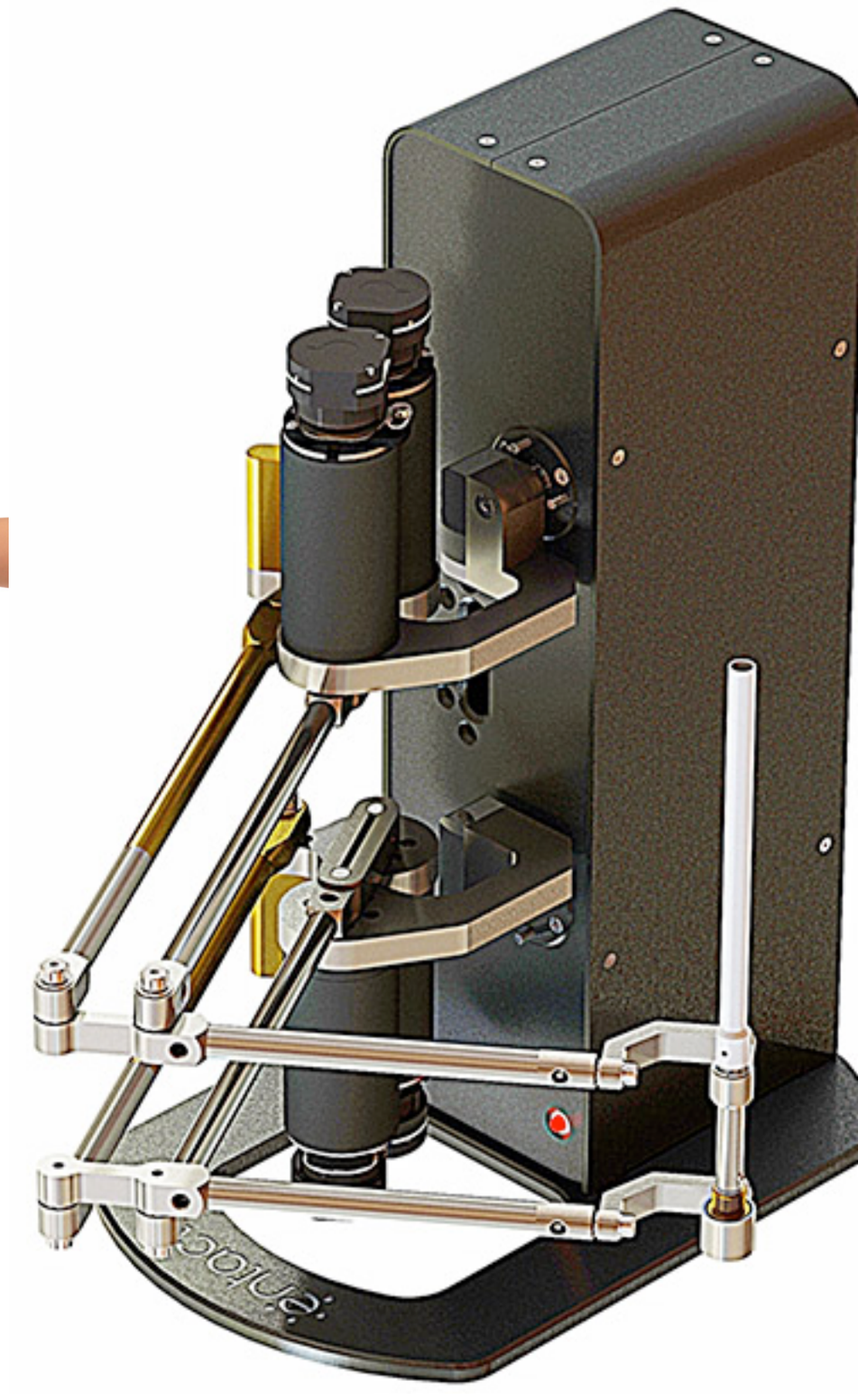
HAPTIC INTERFACE DEVICES

- Hybrid architectures



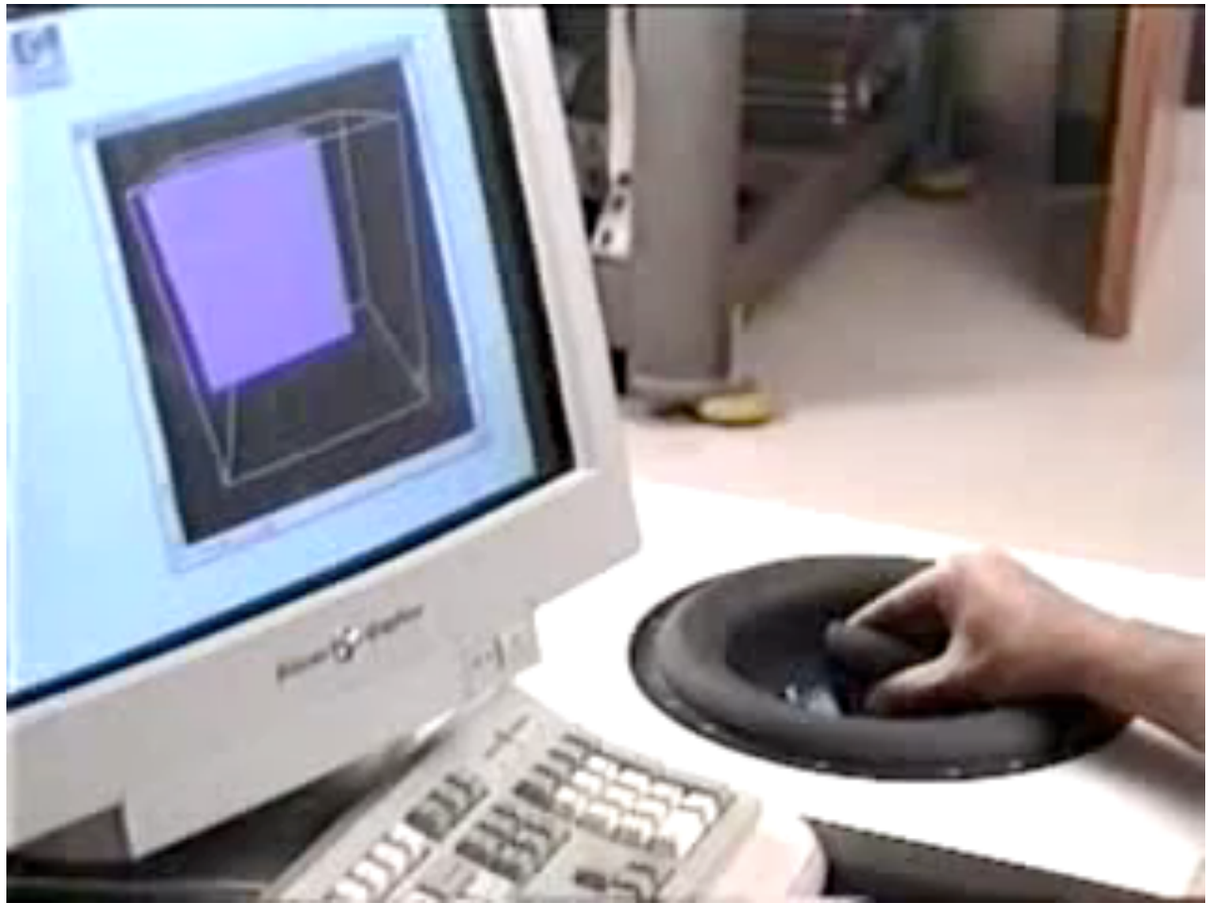
3DOF Planar Twin Pantograph





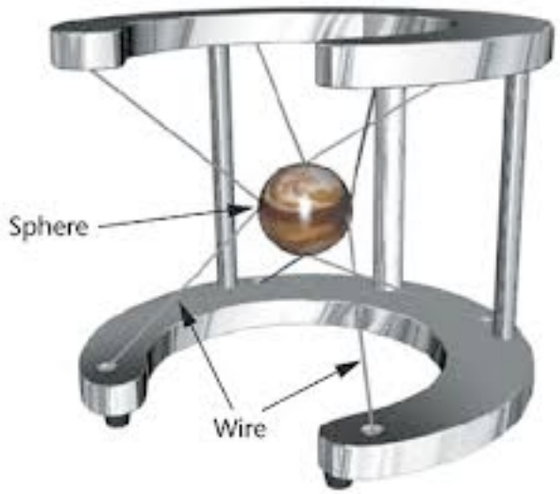
HAPTIC INTERFACE DEVICES

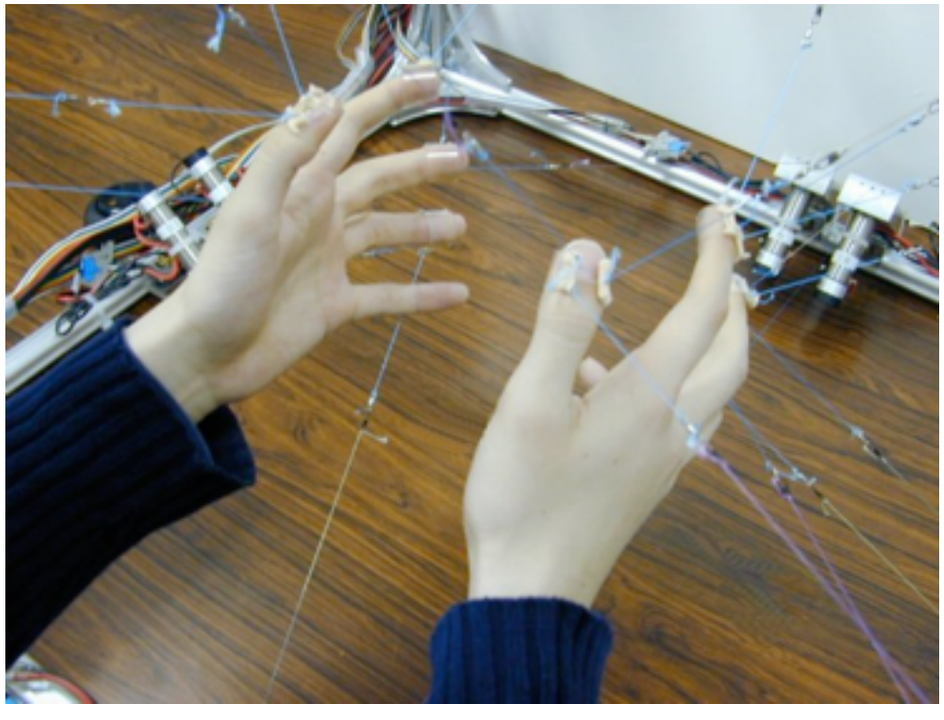
- Magnetic Levitation Haptic Interfaces



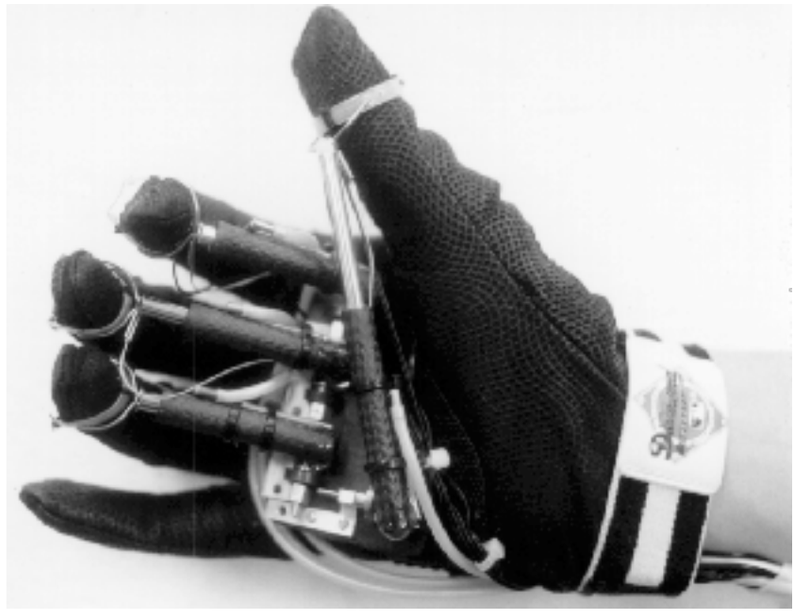
HAPTIC INTERFACE DEVICES

- Cables

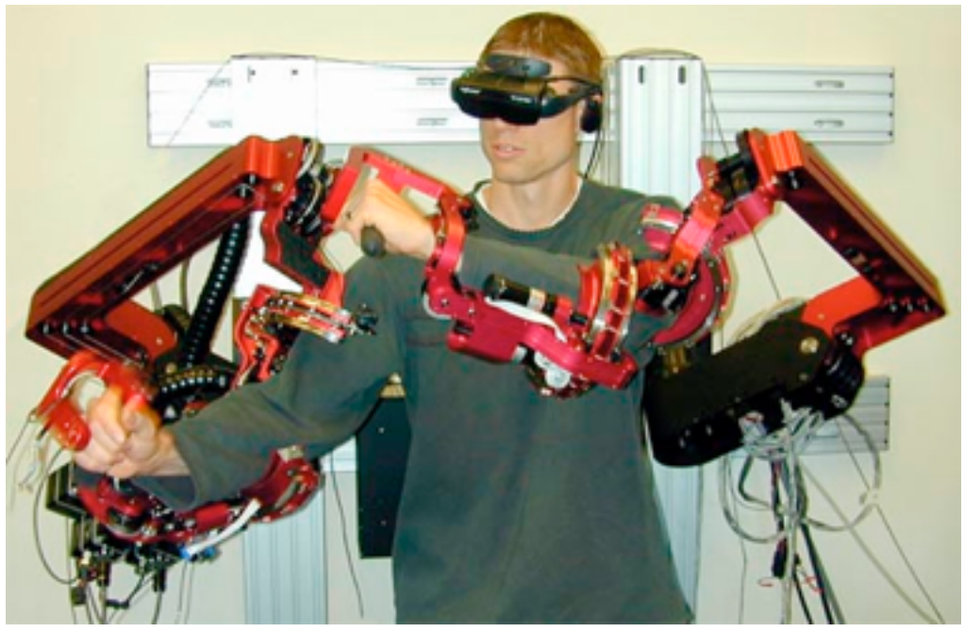


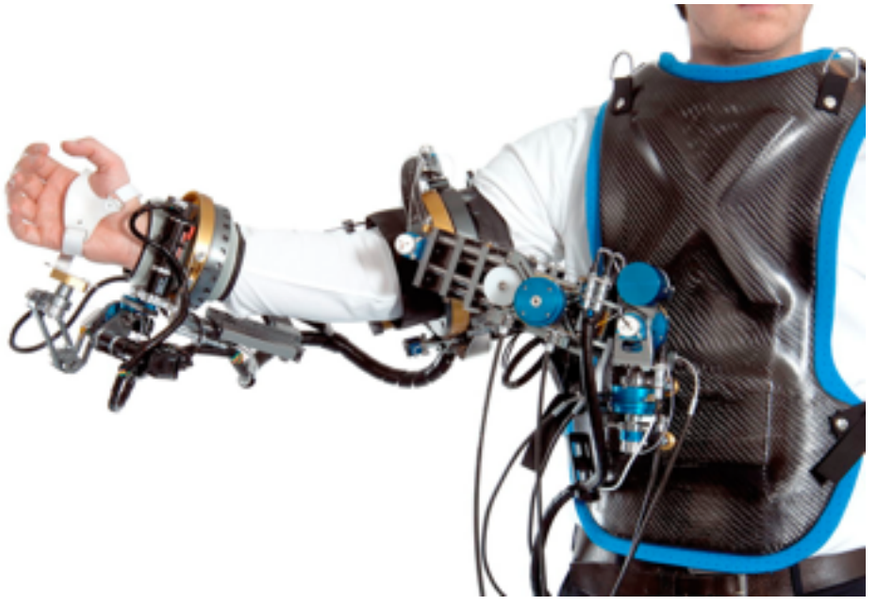


GLOVES



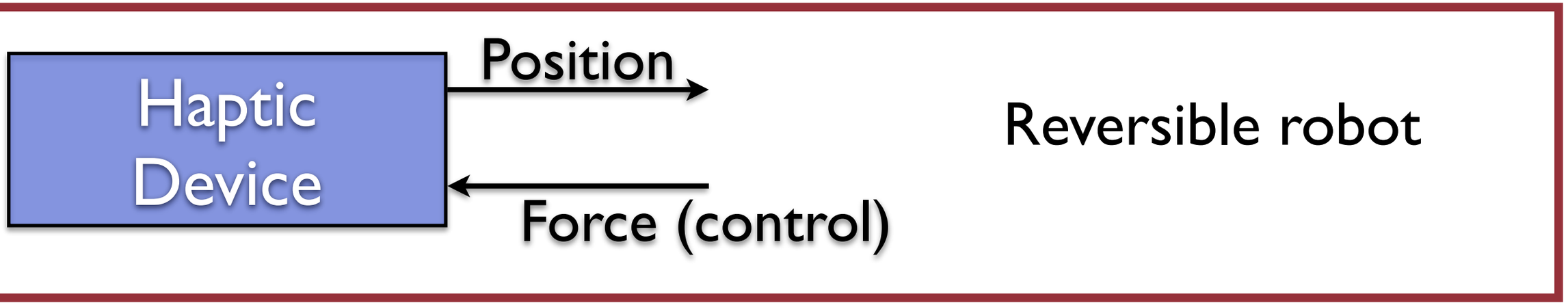
EXOSKELETONS

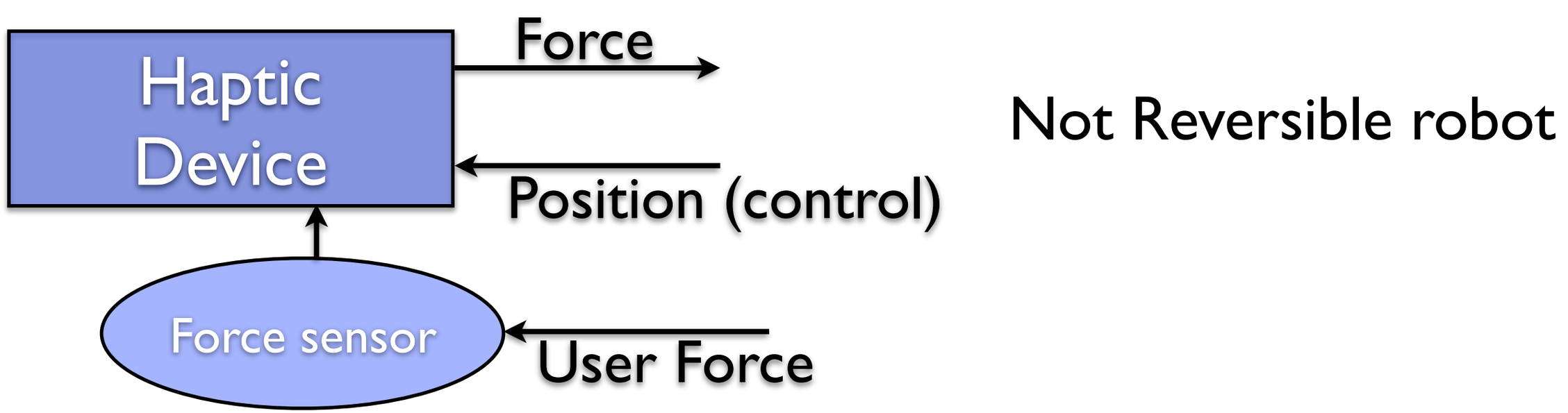




HAPTIC INTERFACE DEVICES

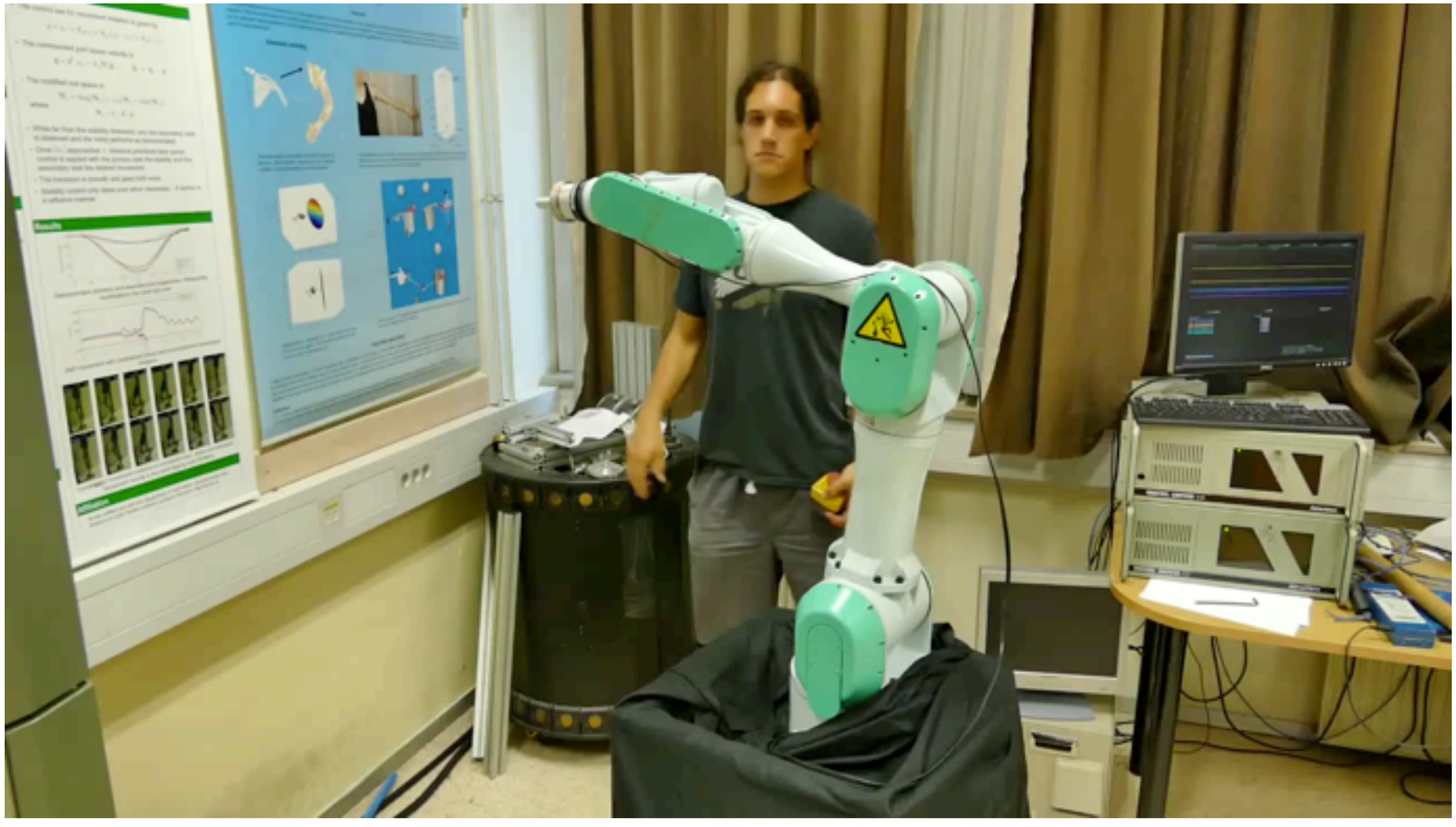
- Impedance / Admittance devices

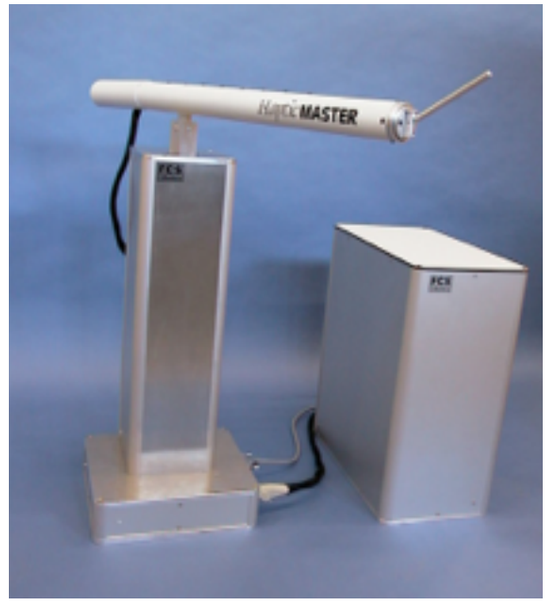




ADMITTANCE DEVICES

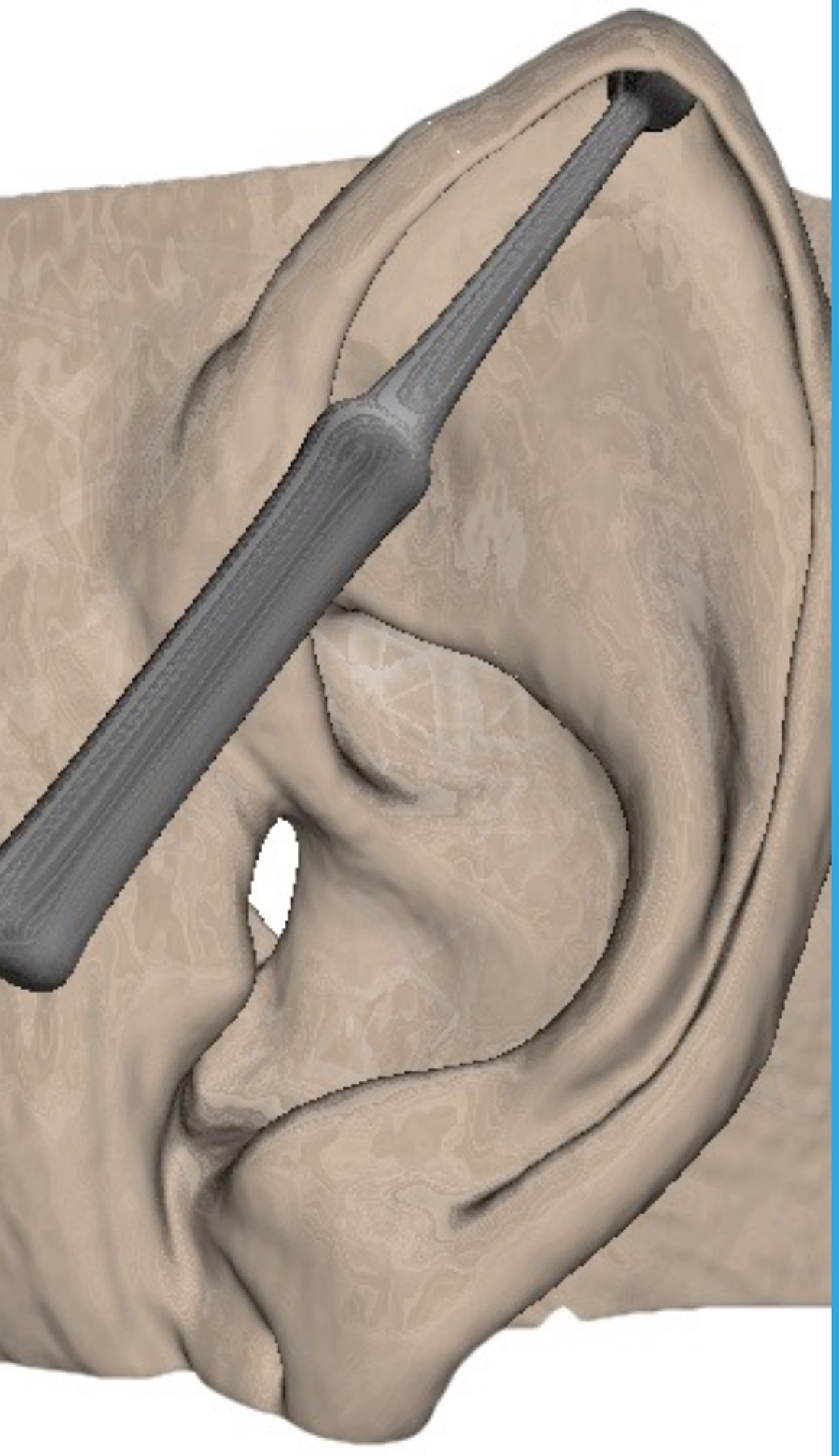
- Use of force sensor





CONSTRAINTS ON DESIGN !!

- low inertia and friction
- Minimal constraints on motion (« feels free »)
- Symmetric inertia, friction, stiffness, resonate-frequency
- Balanced range, resolution and bandwidth of position sensing and force reflection
- Ergonomic



HAPTIC ALGORITHMS

HAPTIC RENDERING ISSUES

- **Stability**: Haptic device is a robotic arm !! The control must be stable
  - When coupling, both simulation and control must be stable !
- **Transparency**: the force that is transmitted to the user is supposed to be the actual force, computed in the simulation
  - Computation of forces in the simulation must be correct + the control should not perturb the rendering

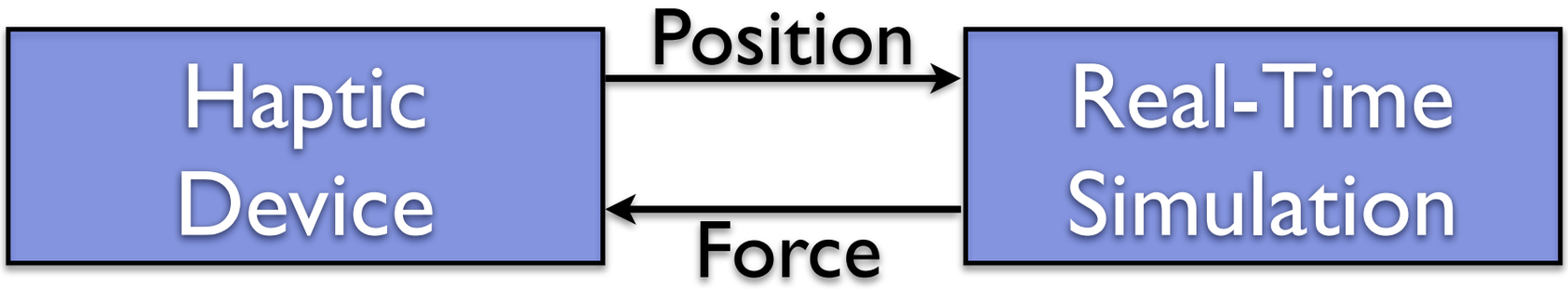
HAPTIC RENDERING ISSUES

- Stability
  - Related to passivity: the simulation and the control should never add energy to the system… (Add damping… but…)
  - Time step must be as small as possible (as delay creates energy): often use of 500Hz / 1kHz
  - Related to robustness: the simulation must be robust to any gesture of the user…
  - The simulation can be stable without haptic and becomes instable with haptic rendering !

HAPTIC RENDERING ISSUES

- Transparency
  - Problem with Damping forces
    - Damping helps the stability but is very bad for transparency...
  - Depends on contact response model
  - Penalty methods / Constraint-based approaches.

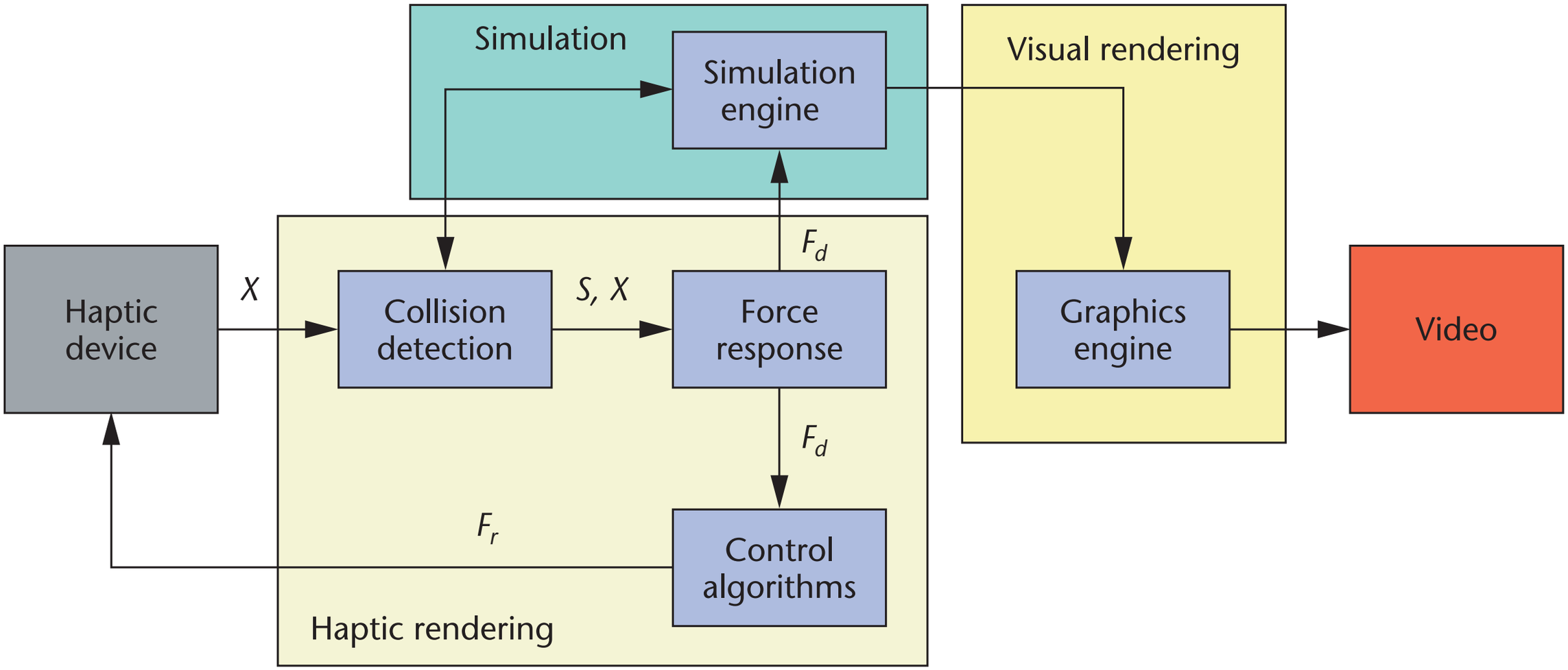
DIRECT COUPLING



No !

CLOSED-LOOP BETWEEN USER AND HAPTIC

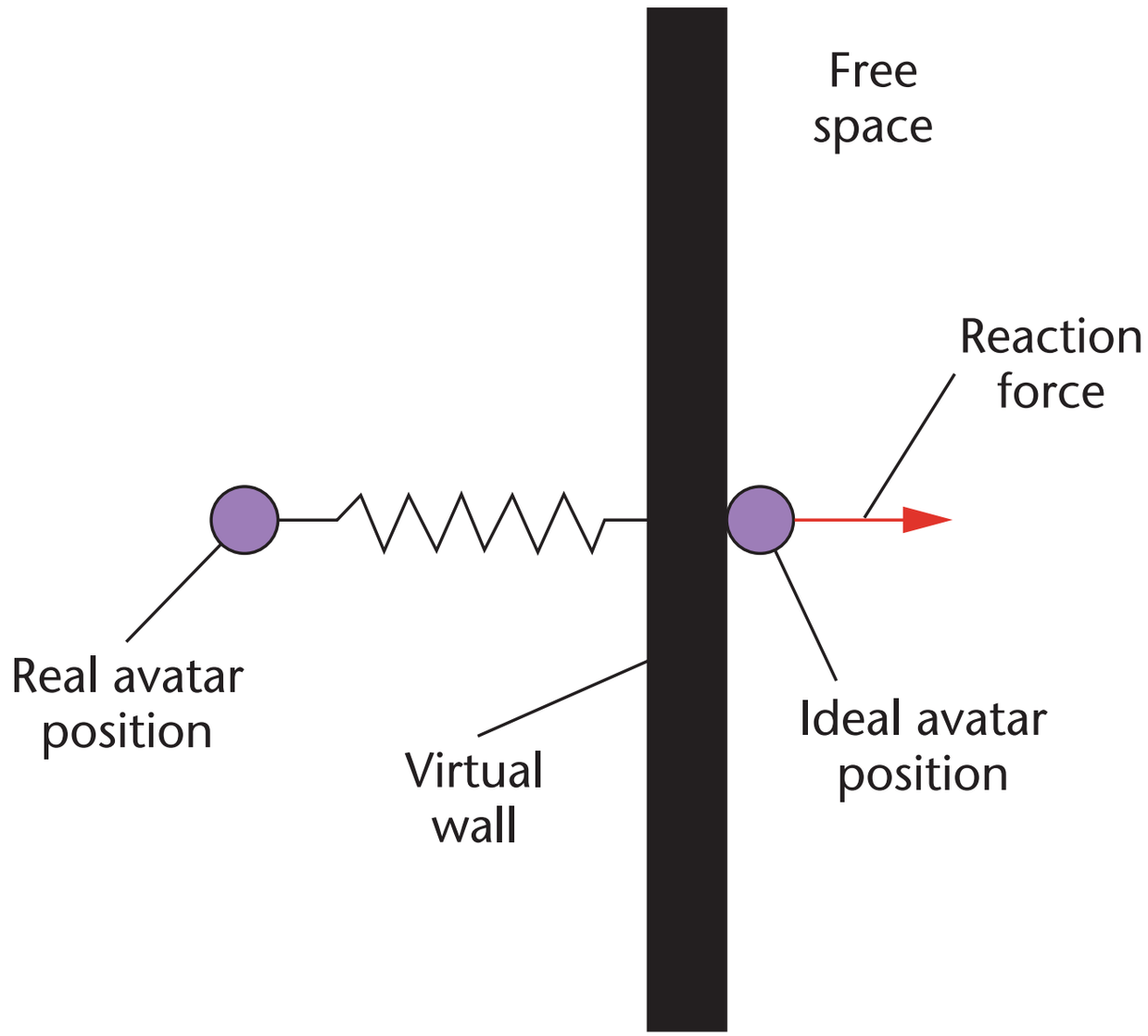
HAPTIC RENDERING



AVATAR

- virtual representation of the haptic interface through which the user interacts with the virtual environment
- Depends on what’s being simulated
- The operator controls the avatar’s position

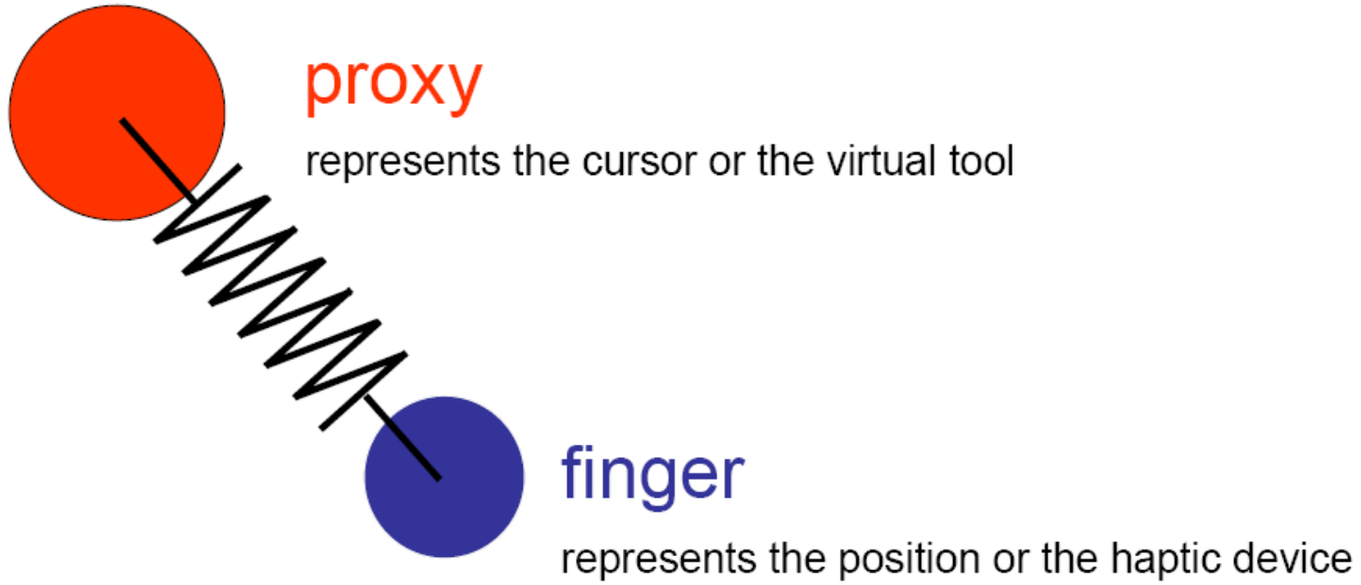
VIRTUAL WALL CONCEPT



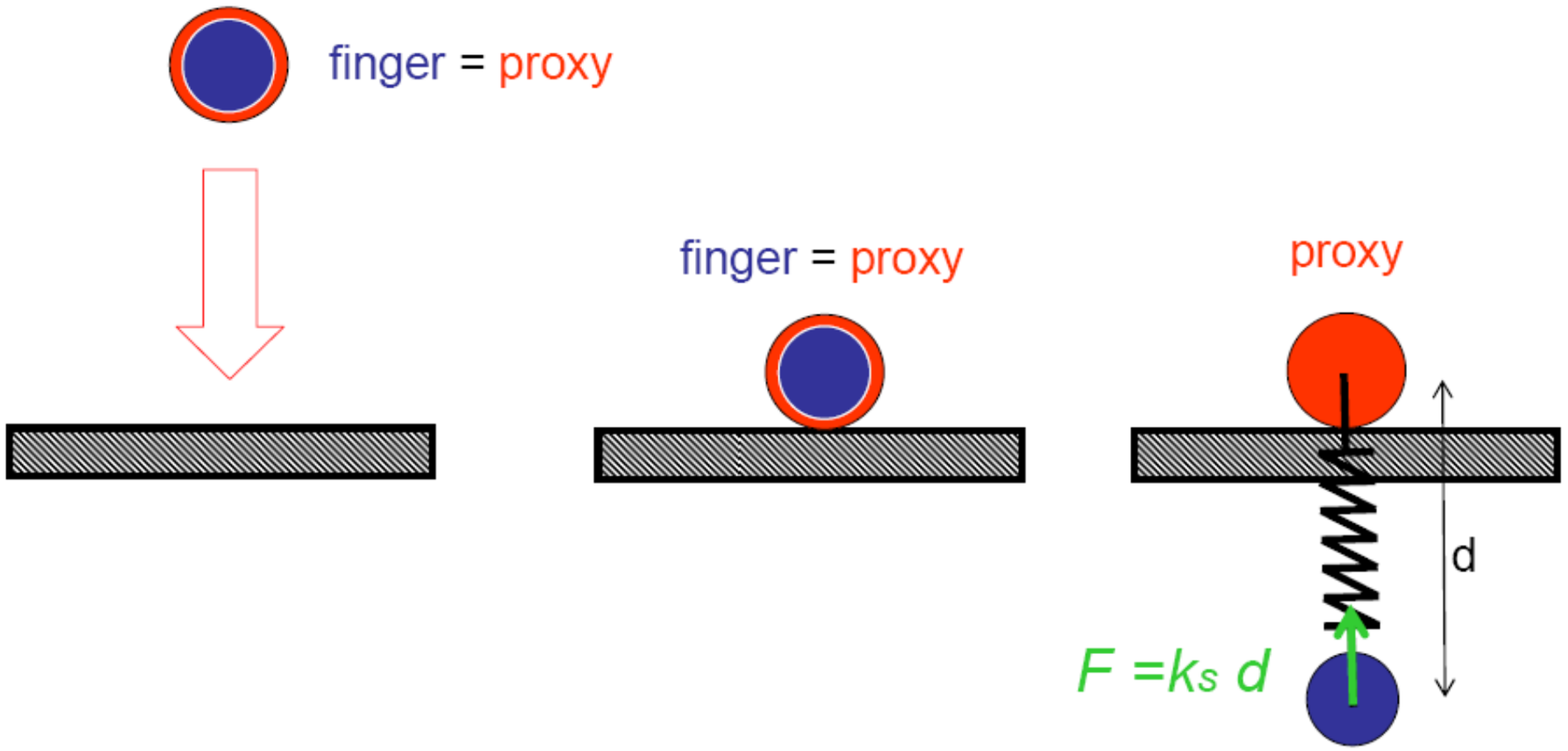
$$F = \begin{cases} 0 & x > x_W \\ K(x_W - x) & x \le x_W \end{cases}$$

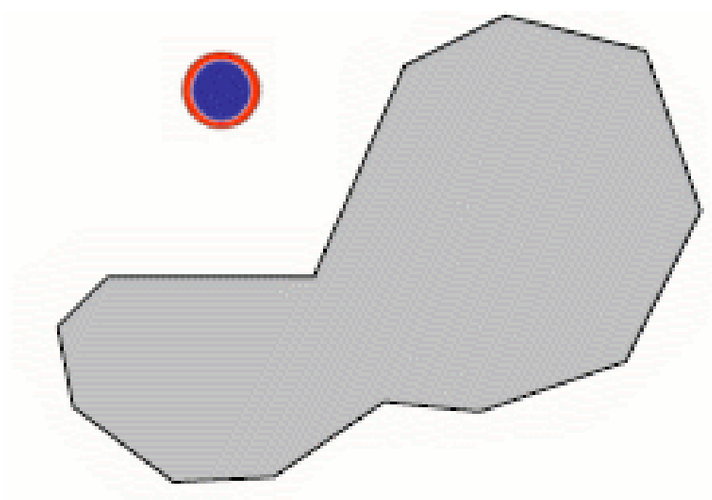
PROXY

- Constrain a virtual proxy of the haptic interface to remain on the surfaces:



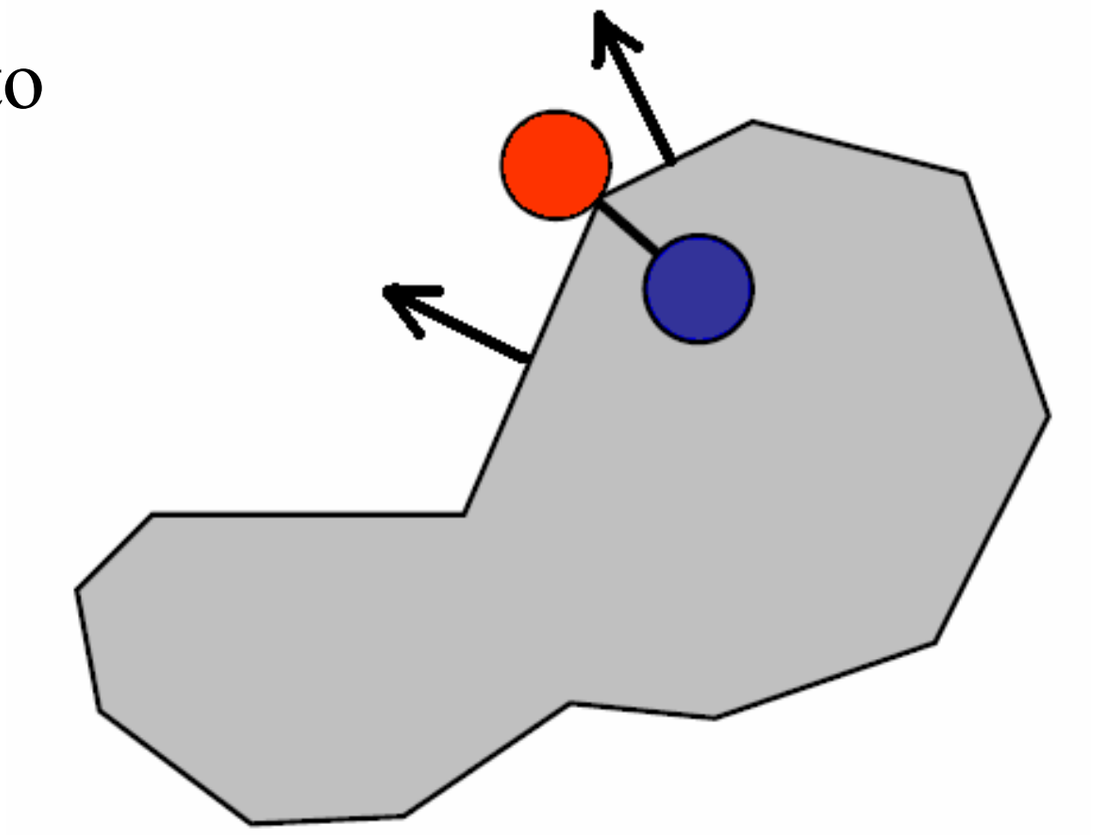
PROXY





PROXY

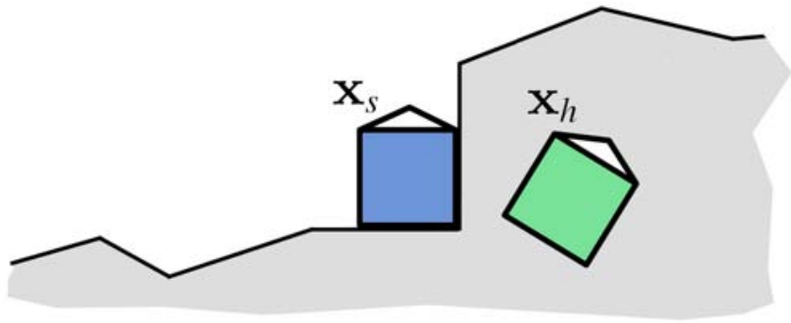
$$\begin{aligned}
&\text{minimize} \left\| x - p \right\| \text{subject to} \\
&\hat{n}_1^T x \geq 0, \\
&\hat{n}_2^T x \geq 0, \\
&\vdots \\
&\hat{n}_m^T x \geq 0,
\end{aligned}$$

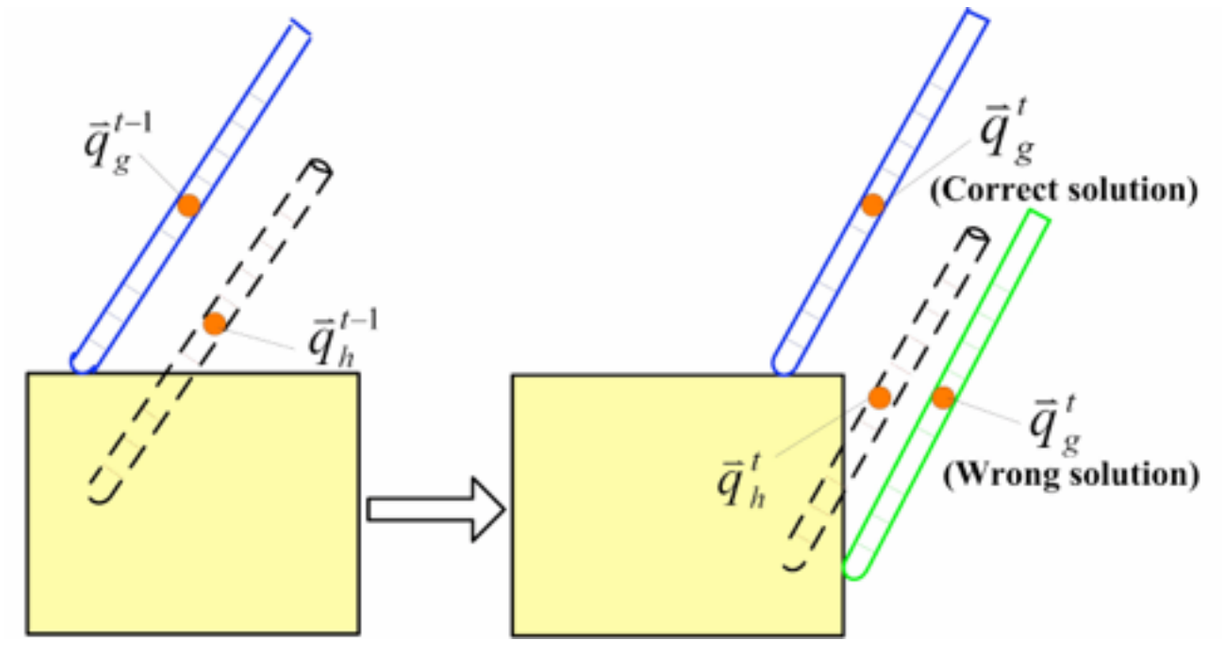


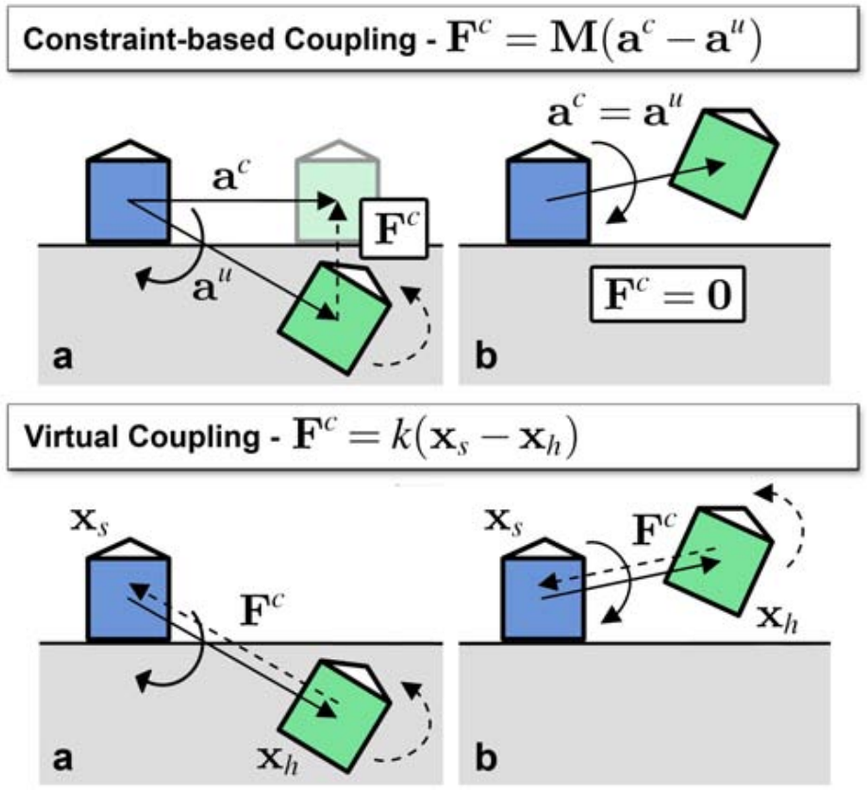
Still no simulation…

6DOF GOD-OBJECT

- More complex…

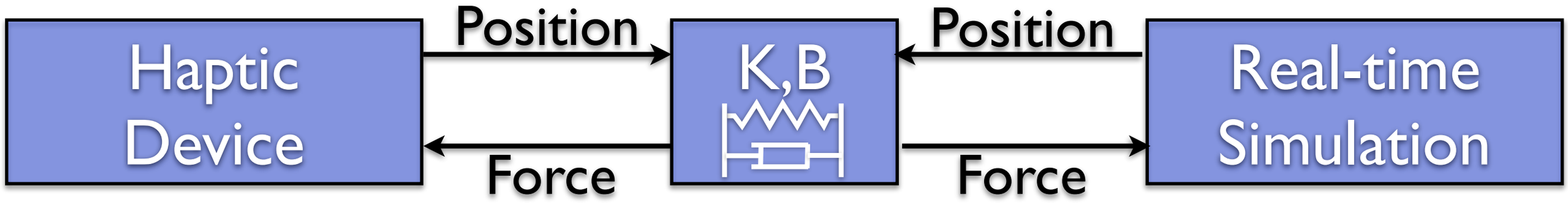






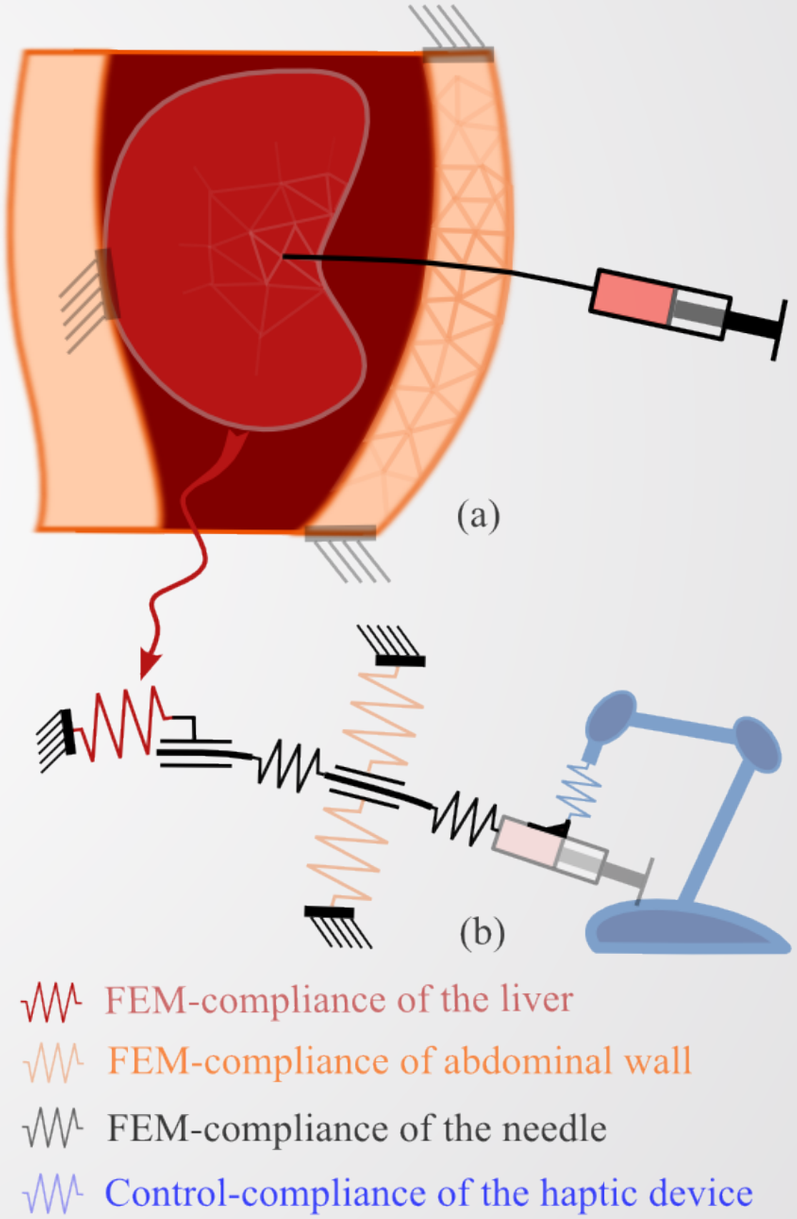
VIRTUAL COUPLING METHOD

- A 6-DoF damped spring is placed between the haptic loop and the simulation.



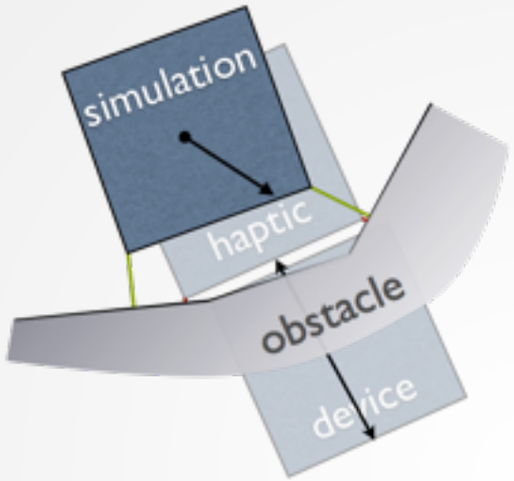
MULTIRATE COMPLIANT MECHANISMS

- Mechanisms
  - Support an extensive number of interaction types,
  - Versatile definition of constraint laws (adequate force/ motion transmission model).
- Multirate
  - Build and simulate the mechanisms at low rates
  - Share with the haptic loop
  - Recompute at high rates for an intuitive and passive control.
- Compliant
  - Use the mechanical coupling between interaction spots,
  - Build compliance matrices based on physical models,
  - Handle both deformable and rigid objects.

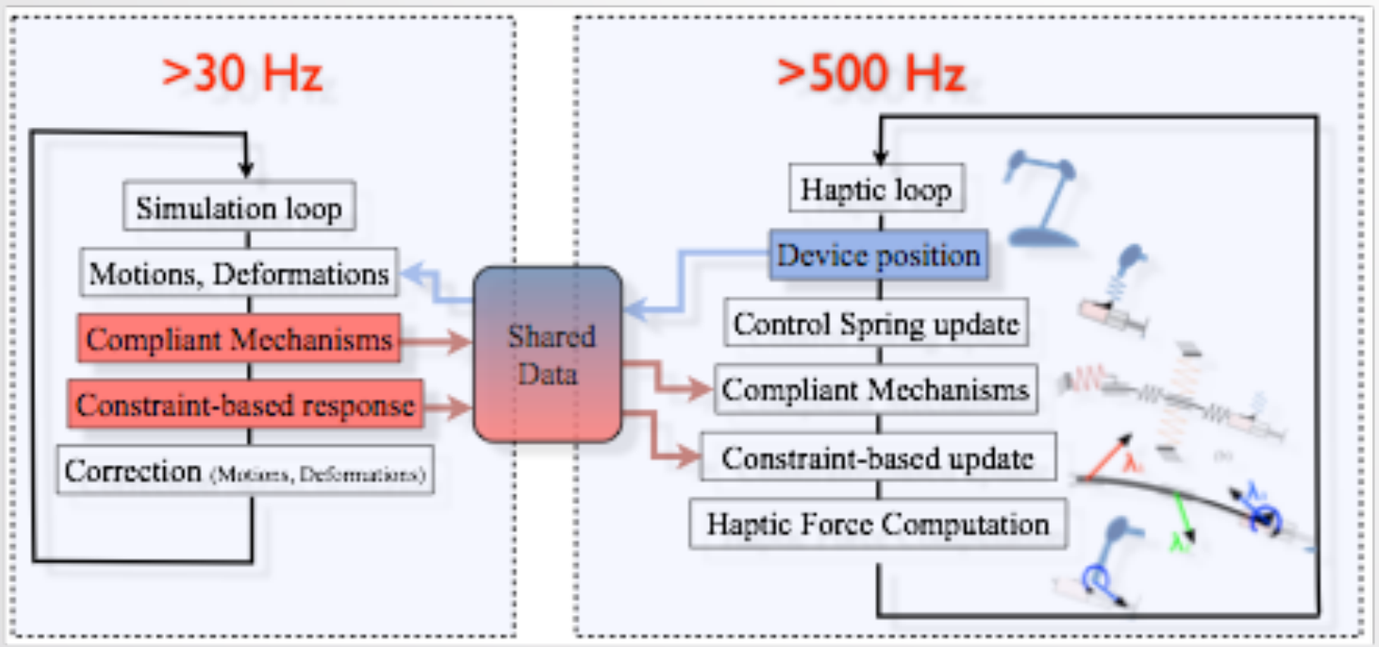


MULTIRATE COMPLIANT MECHANISMS

- 3 ≠ positions

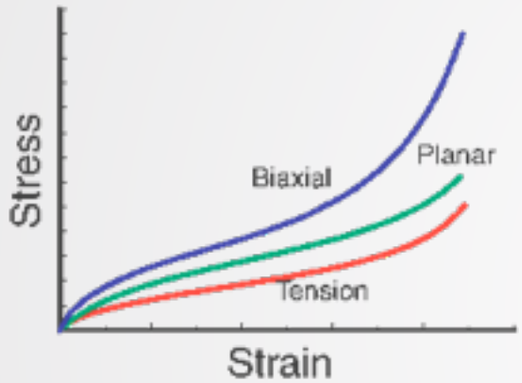


- Shared data to maintain coherency



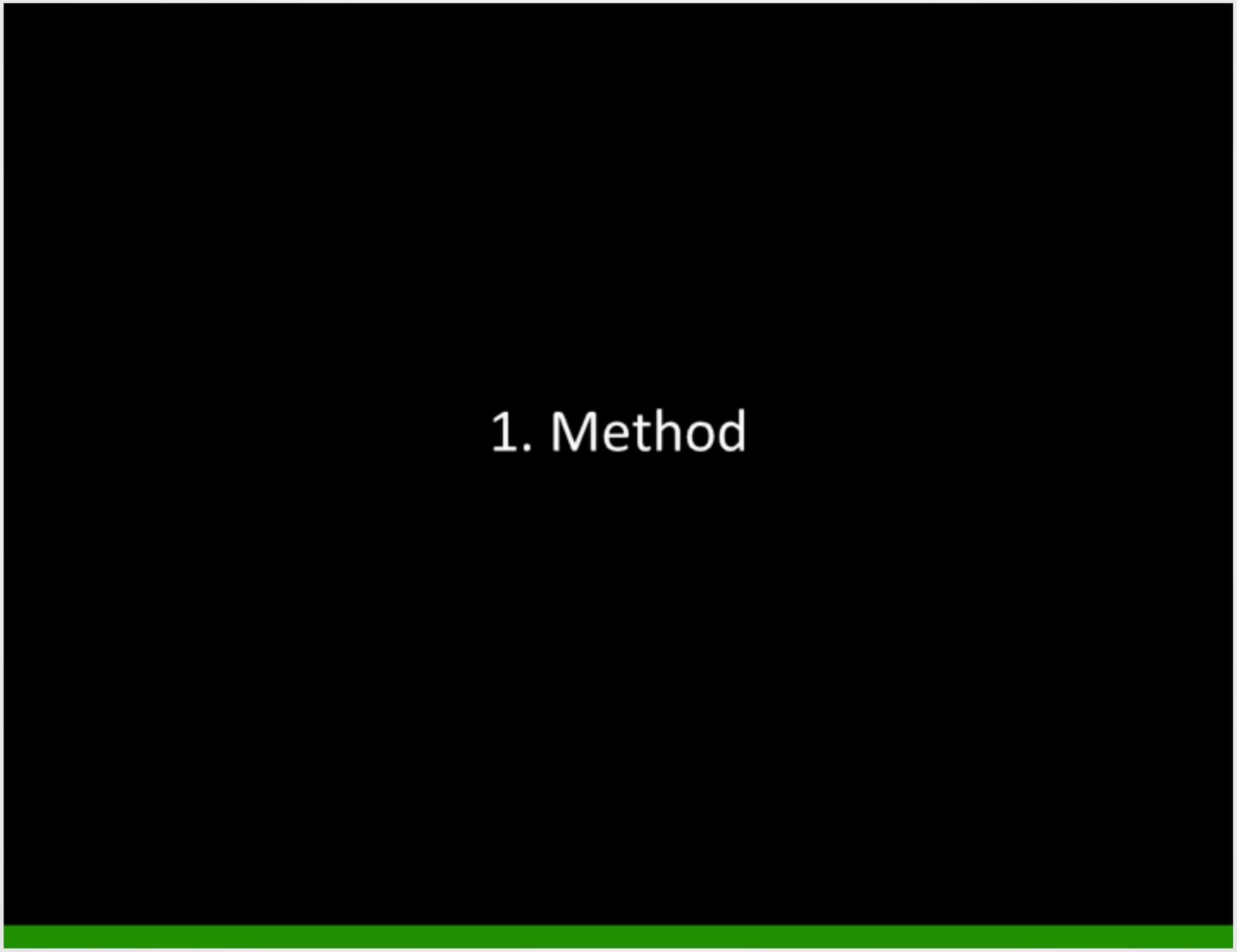
EXAMPLE: HYPERELASTIC MODELS WITH FRICTION

- What is « hyperelastic » ?



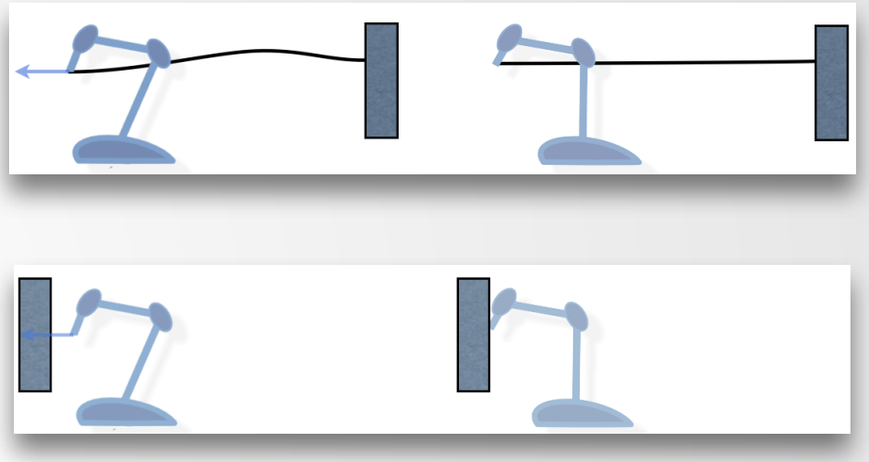
- 3 phenomenons at 3 different rates
  - Velocity of deformations (non-linearities): *low*
  - Velocity of user motions: *medium*
  - Velocity of sensitive (haptic) feedback: *high*

HAPTIC RENDERING OF HYPERELASTIC MODELS WITH FRICTION



TOWARDS ASYNCHRONOUS SIMULATION

- Fast bending/streching transition
  - Need to compute the hole wire model at high rates
  - Compliance of the wire is fast to compute !
- Asynchronous strategy:
  - The thread is simulated at haptic rates (> 500 Hz)
  - The remaining of simulation at low rates (> 25 Hz)
- Preliminary results:
  - Constraints computed at both low and high rates !
  - Follows action/reaction principle
  - about 20 beams at 1000Hz
- Limitations:
  - Works well with quasi-static behaviors
  - More limitations on dynamic models (for now ?)

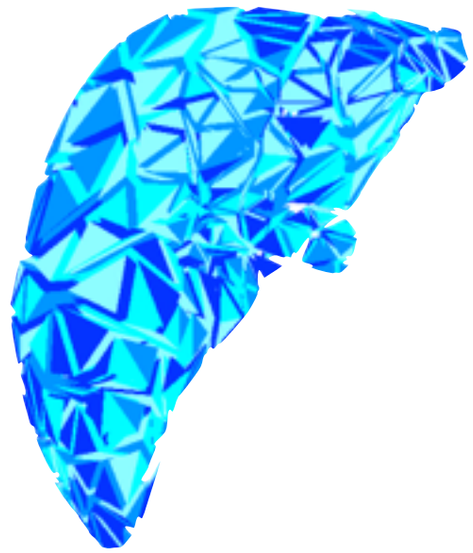


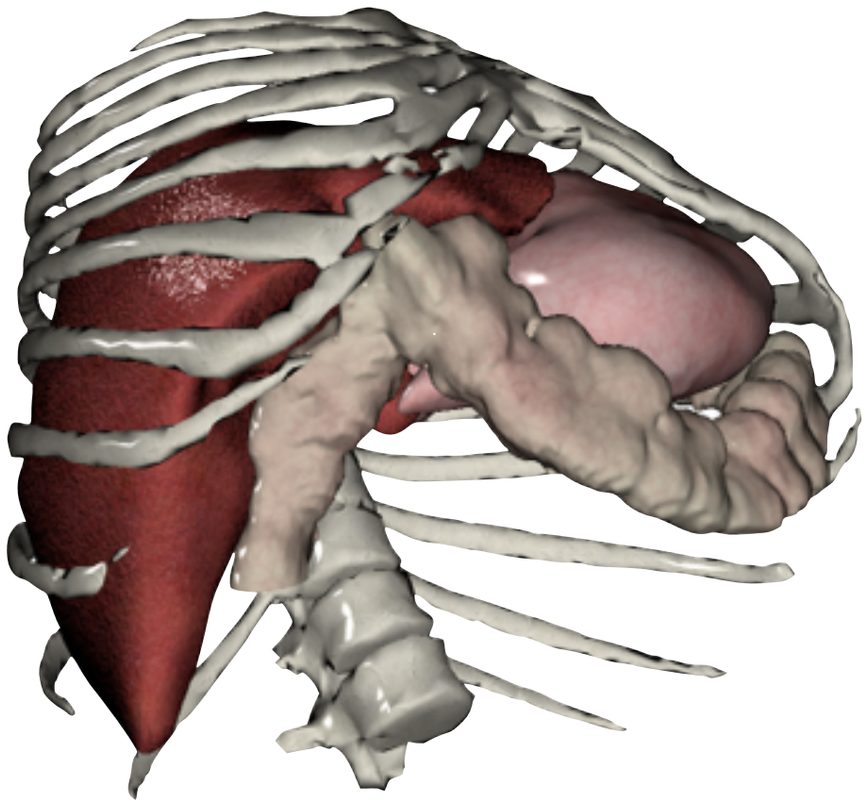
SOFTWARE ARCHITECTURE FOR REAL-TIME SIMULATION

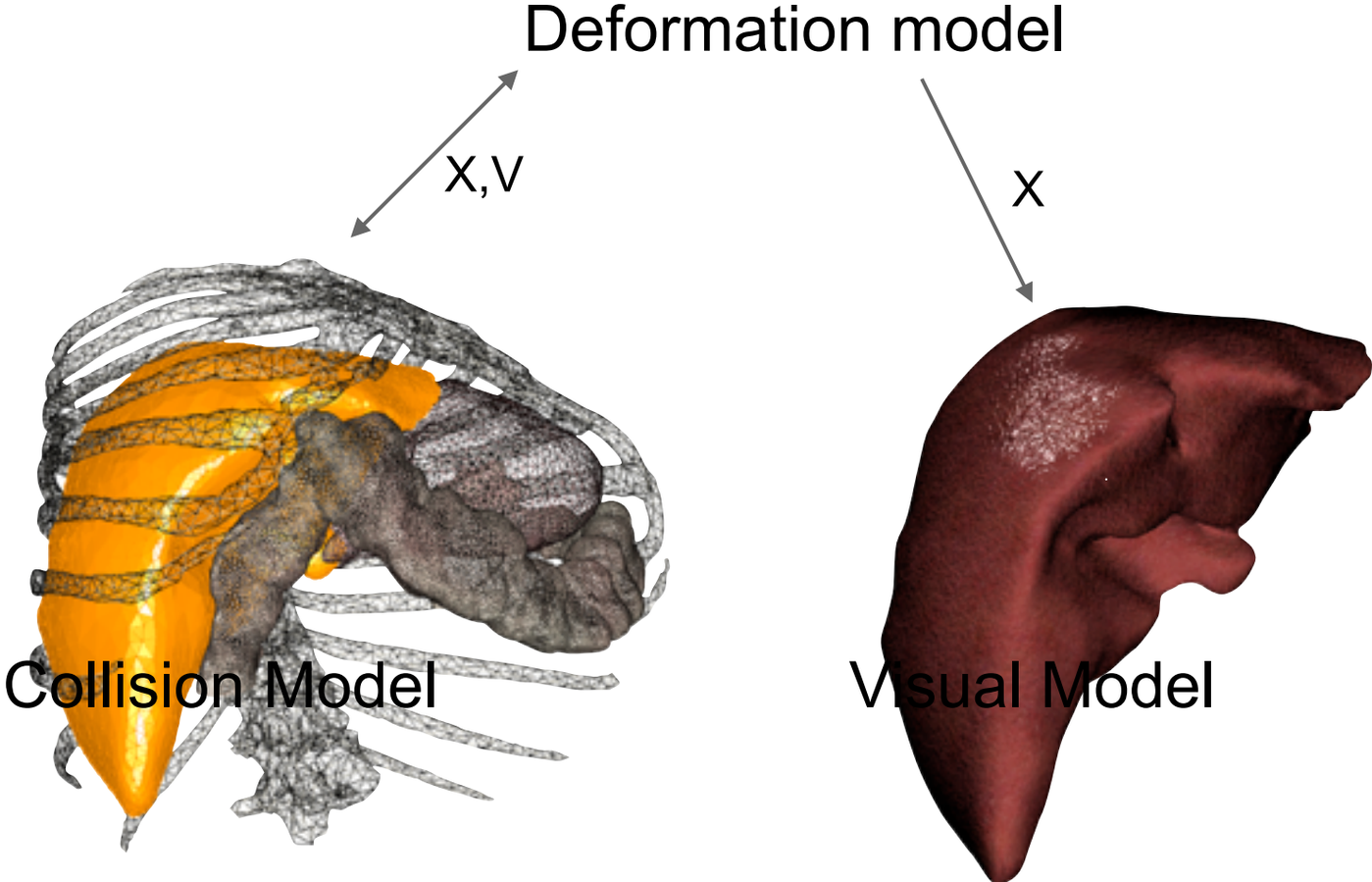
Frank Gehry

EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION

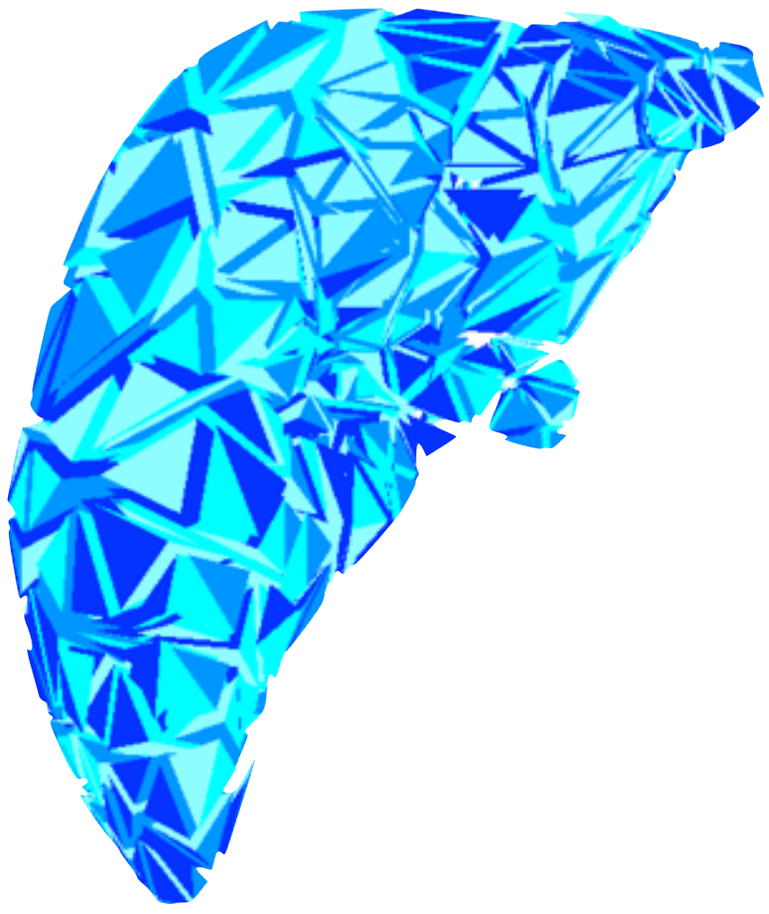
- Split computations into independent parts
  - Separate problems
  - Improve reusability







EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION



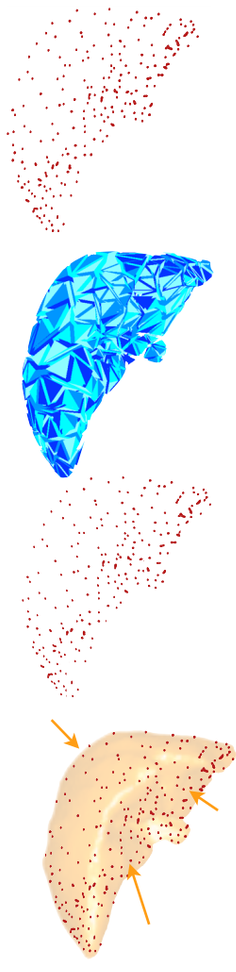
Deformation model

$$\mathbb{M}\dot{\mathbf{v}} = \mathbf{p} - \mathbf{f}(\mathbf{v}, \mathbf{x}) + \mathbb{H}^T\lambda$$

ODE Solver

$$\mathbf{A}x = \mathbf{b}$$

Linear Solver



Degrees of Freedom

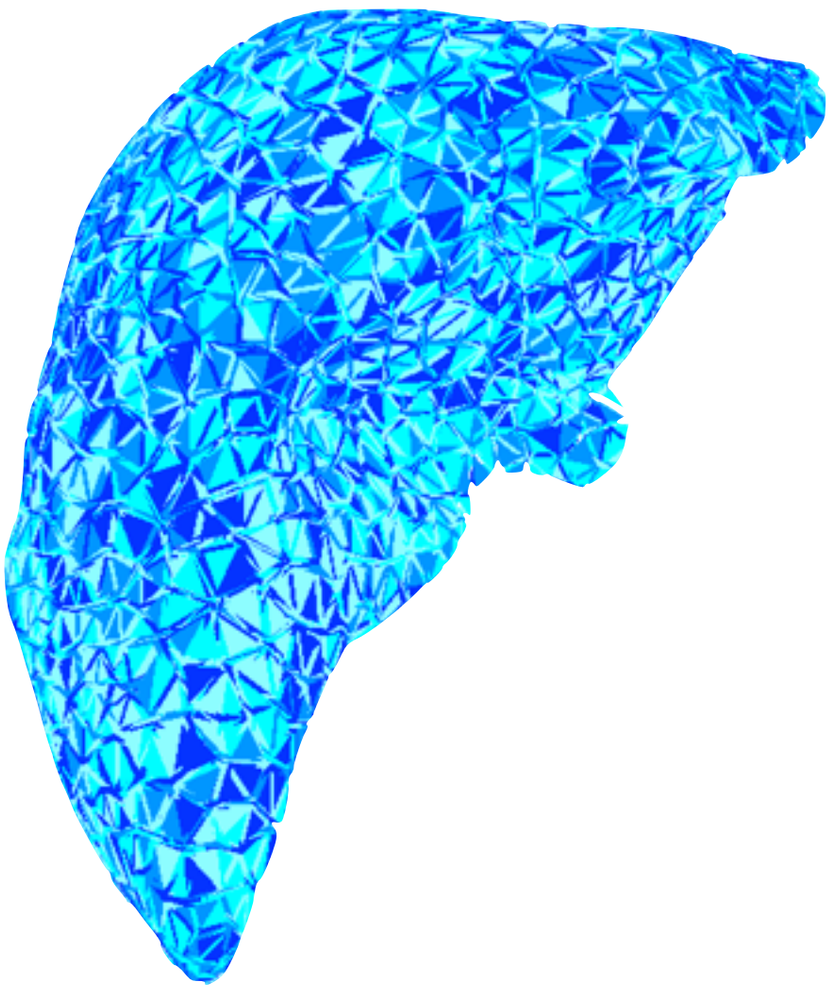
Finite Element Forces and Mass

External Loads

Mapped Forces

...

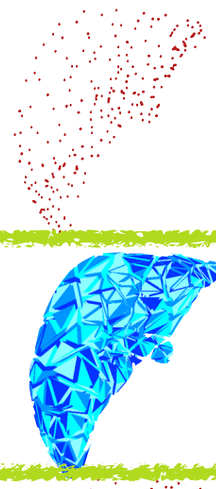
EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION



computation on GPU

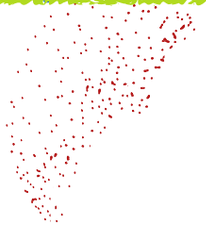
$\mathbb{M}\dot{\mathbf{v}} = \mathbf{p} - \mathbf{f}(\mathbf{v}, \mathbf{x}) + \mathbb{H}^T\lambda$ ODE Solver

$\mathbf{A}x = \mathbf{b}$ Linear Solver

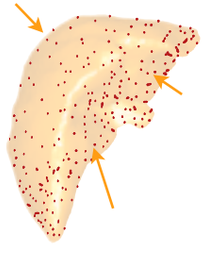


Degrees of Freedom

Finite Element Forces and Mass



External Loads

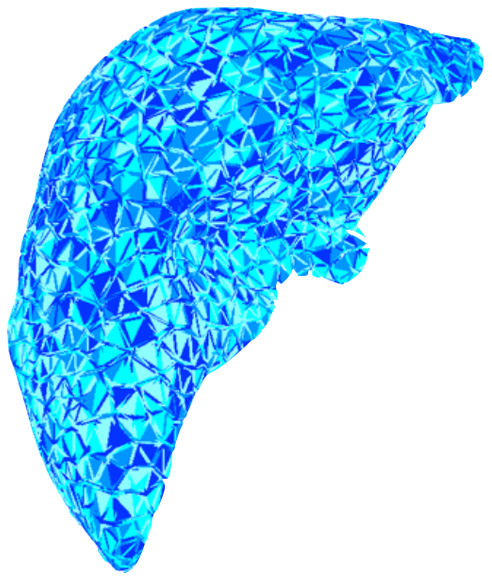


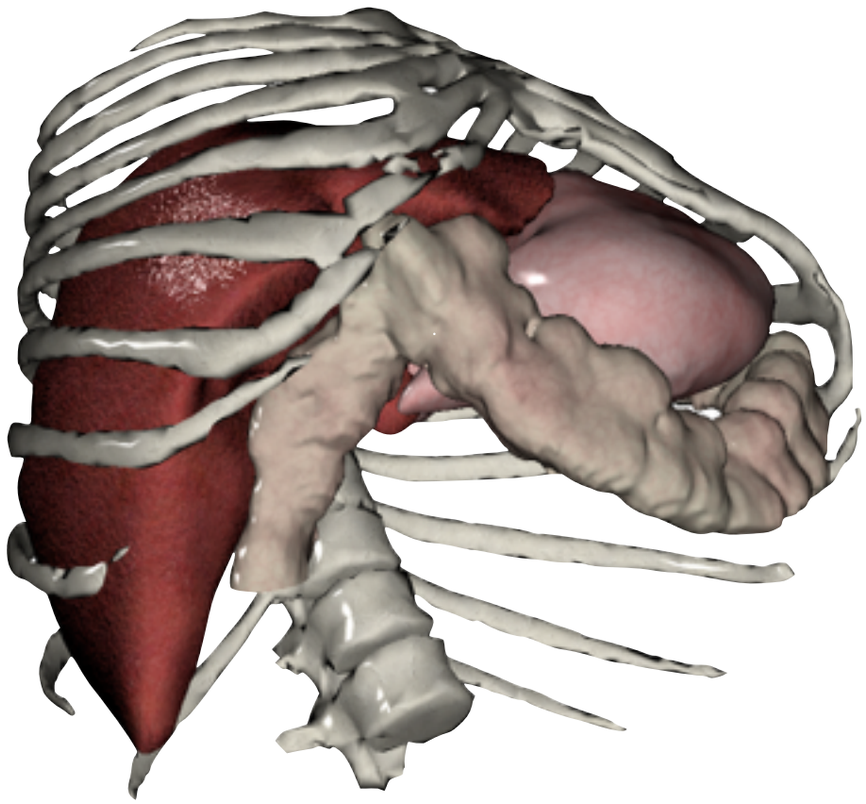
Mapped Forces

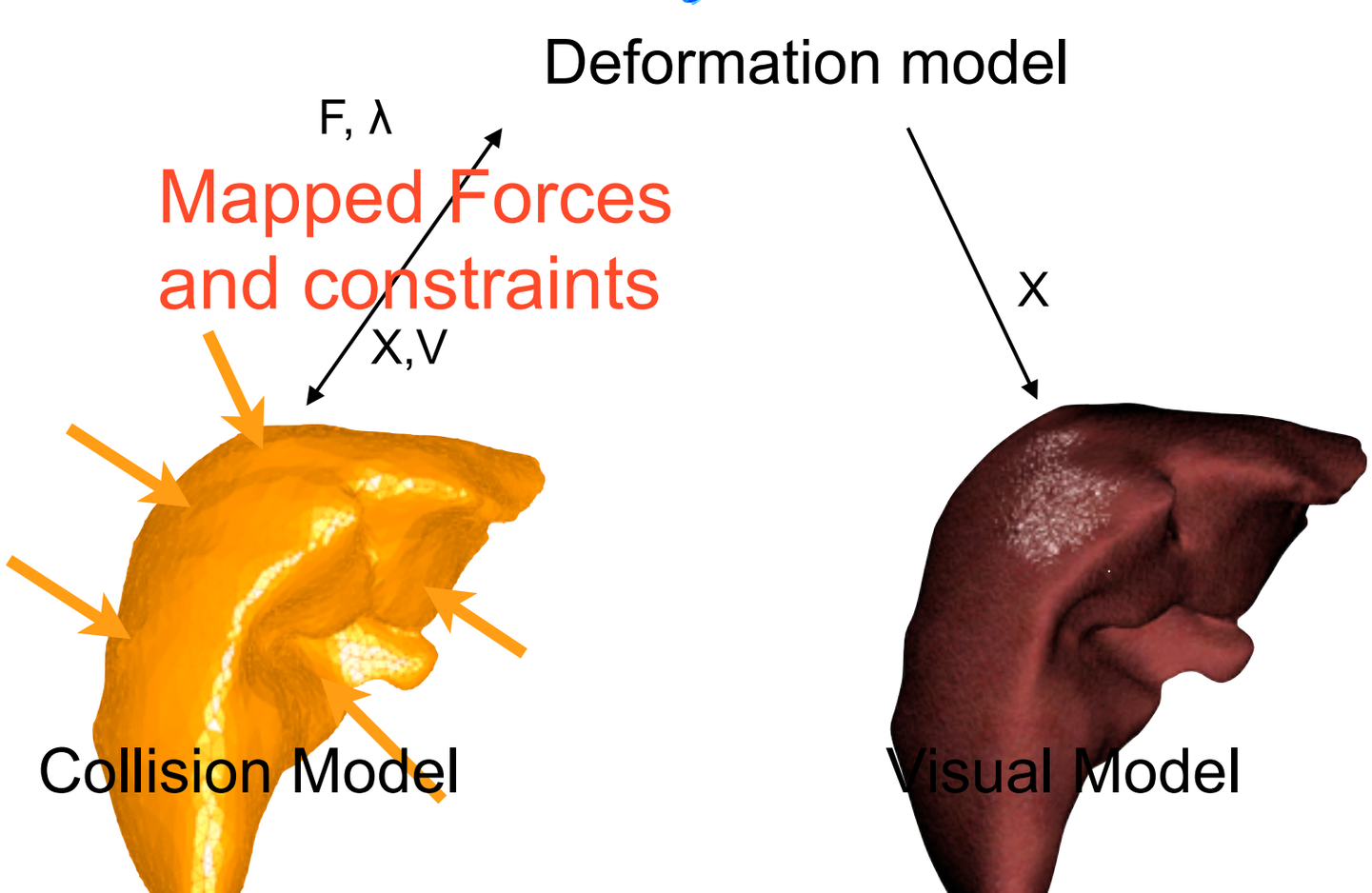
...

EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION

- Mapping allows for precise transmission of the contact forces or constraints

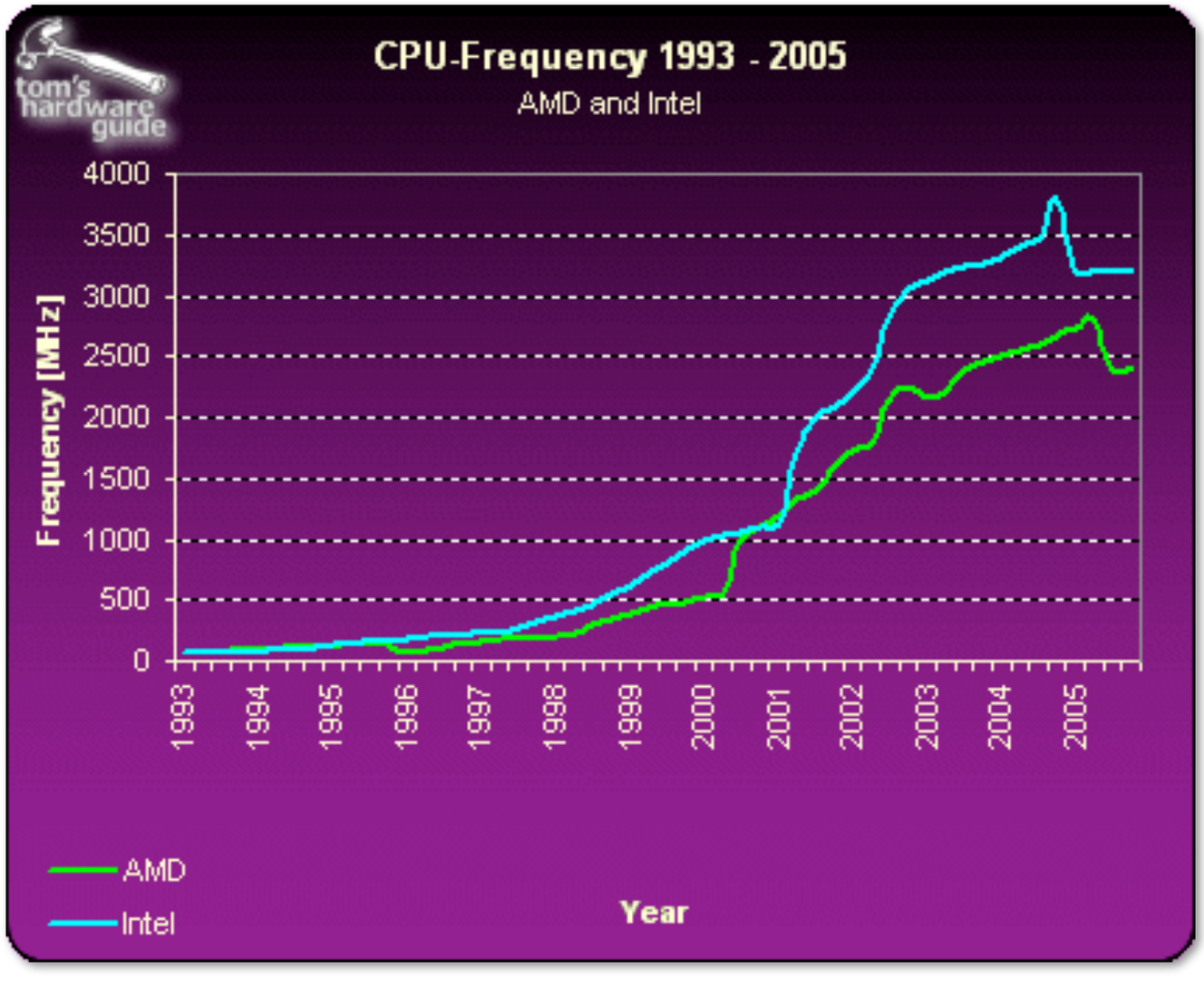


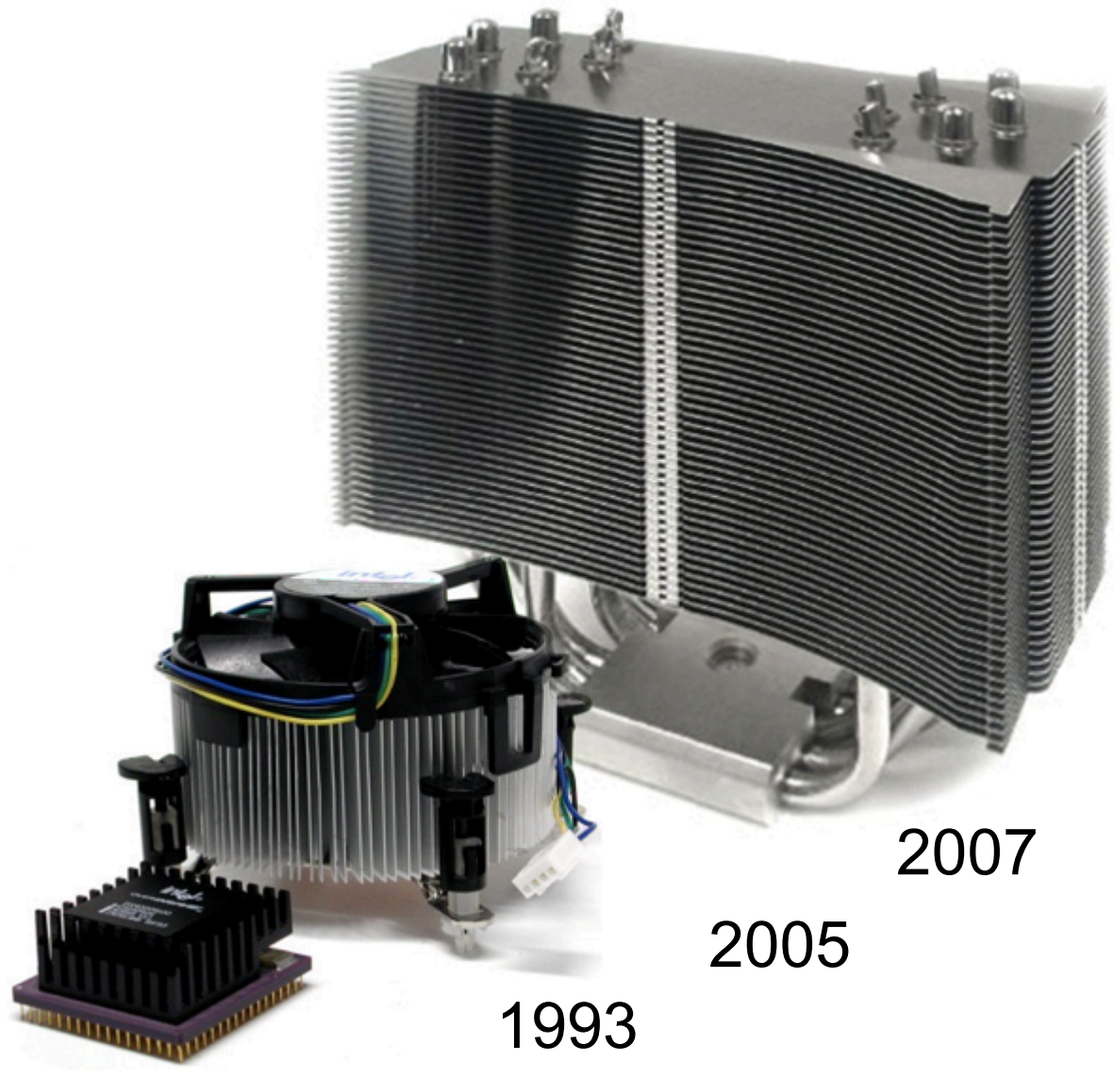




COMPUTATION TIME OPTIMISATION

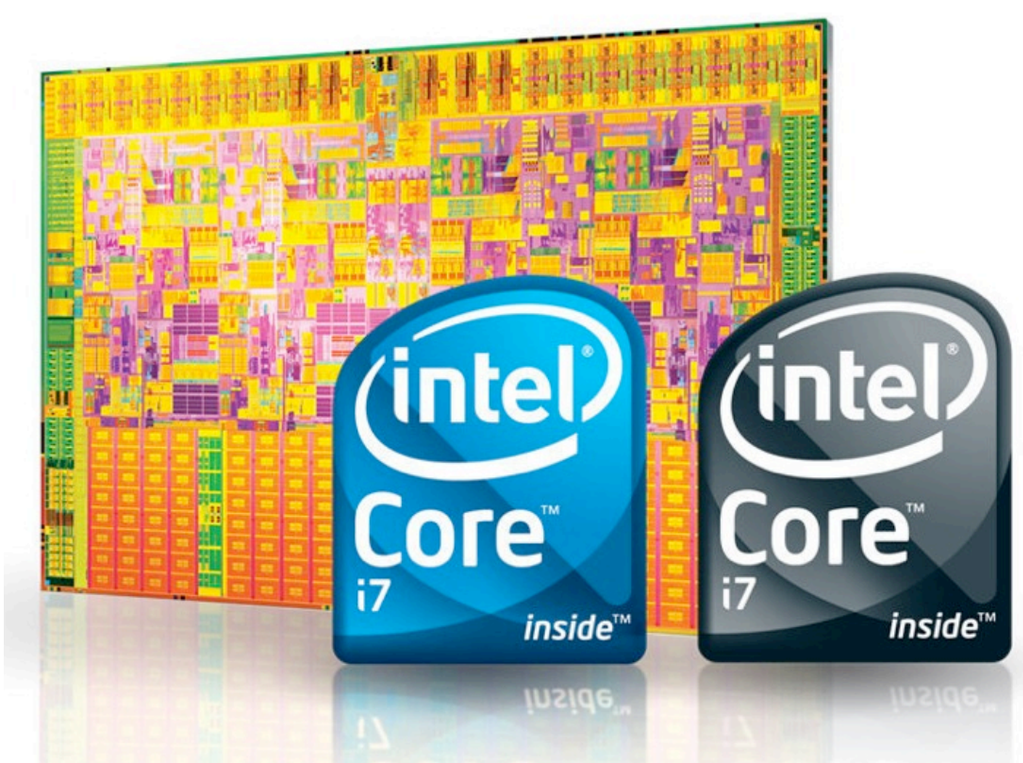
- Why parallel computing ?

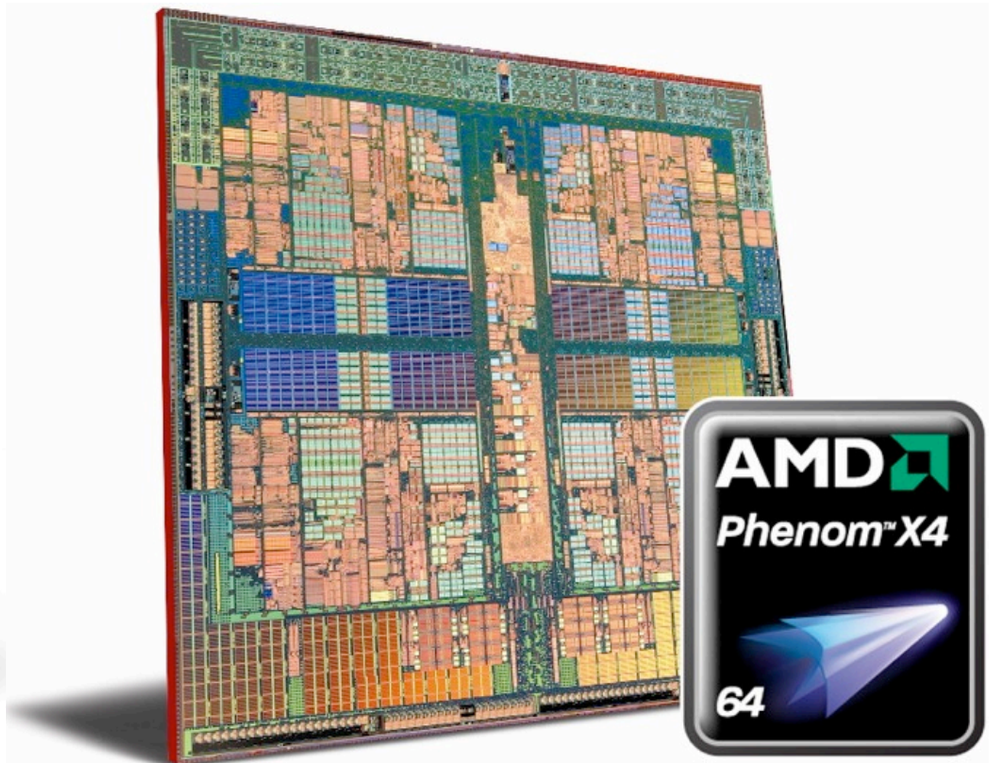




COMPUTATION TIME OPTIMISATION

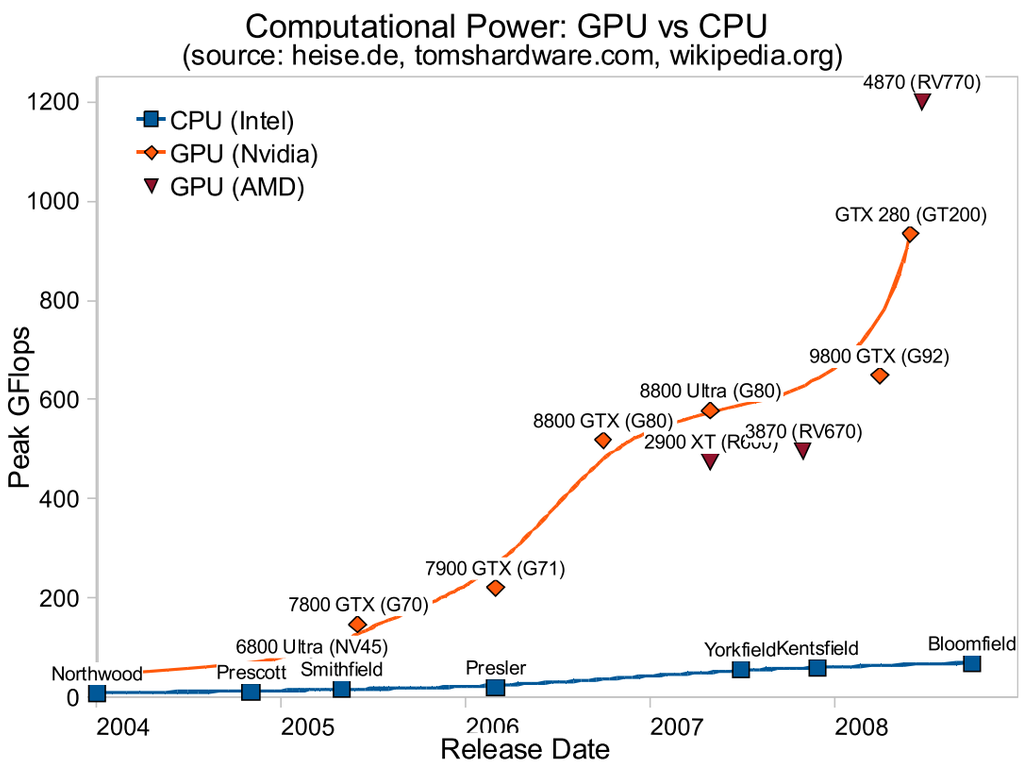
- Classical Parallel Computing
  - Solution : parallelize computations among CPU processing units (2 to 8 cores using 1 or 2 CPUs)
  - Issues : separate tasks, synchronizations, load balancing
    Improve performances, but not dramatically



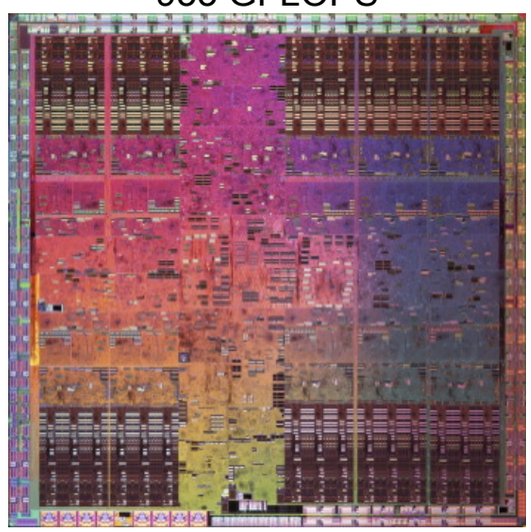


COMPUTATION TIME OPTIMISATION

- *Extreme* Parallel Computing
  - Alternate solution : exploit other available units
  - GPU : hundreds of units, new general purpose languages
  - Issues : massive parallelism, varying functionalities

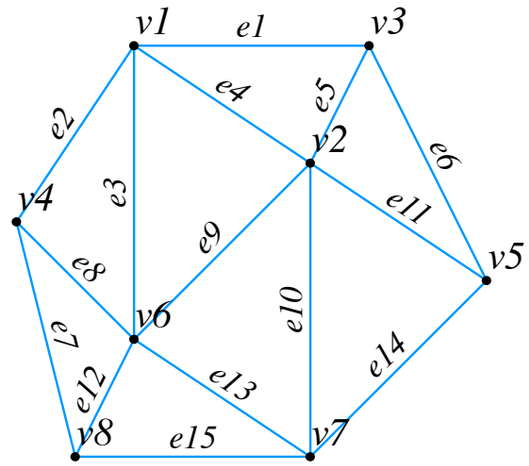


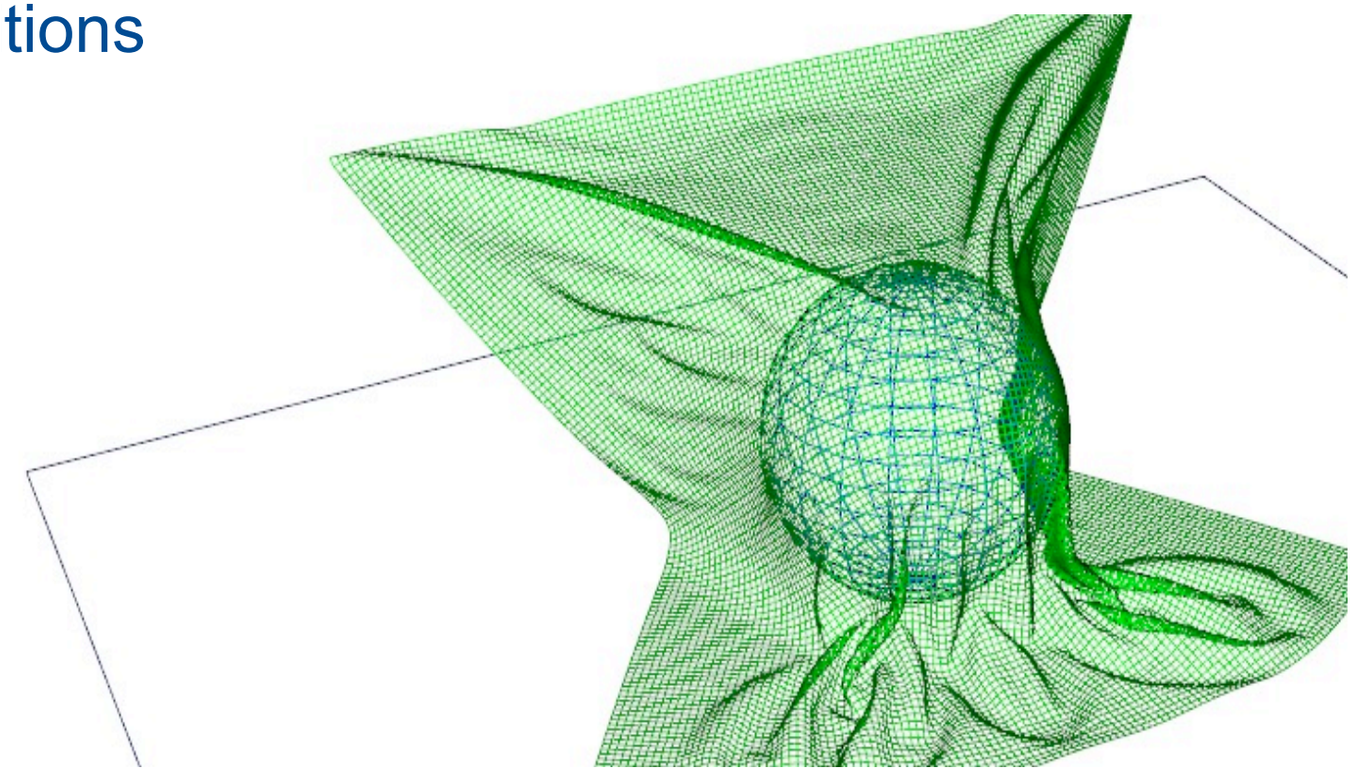
NVIDIA GT 200: 240 units,
933 GFLOPS



Mesh Example

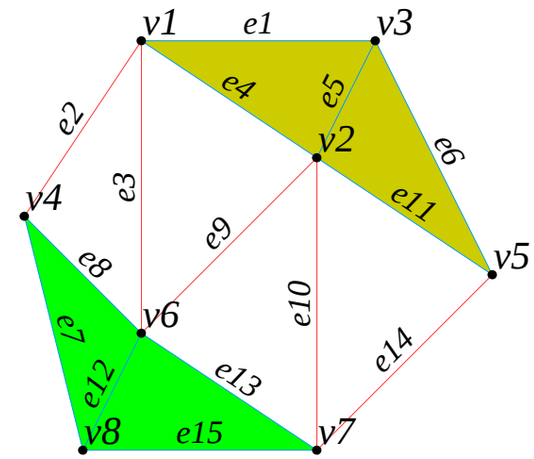
- Let's define a graph with *n* vertices and *m* edges
- We want to accumulate on vertices some values computed on each edge
  - Mass-Spring systems
  - Constraints impulses
  - …
- Problem : several edges connected to the same vertex
- Simple solution : use atomic additions
  - Not available on all architectures
  - Not available on all data-types
  - Slower than regular additions

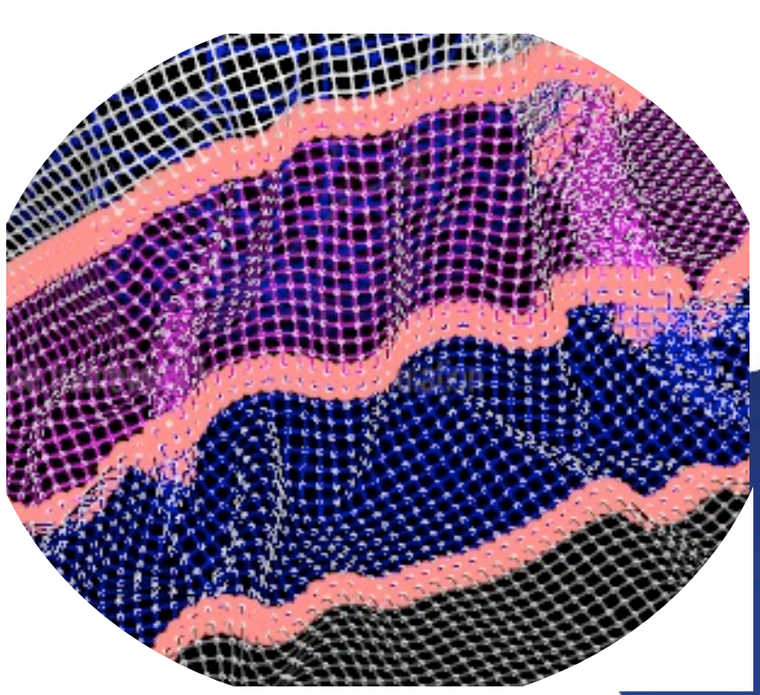


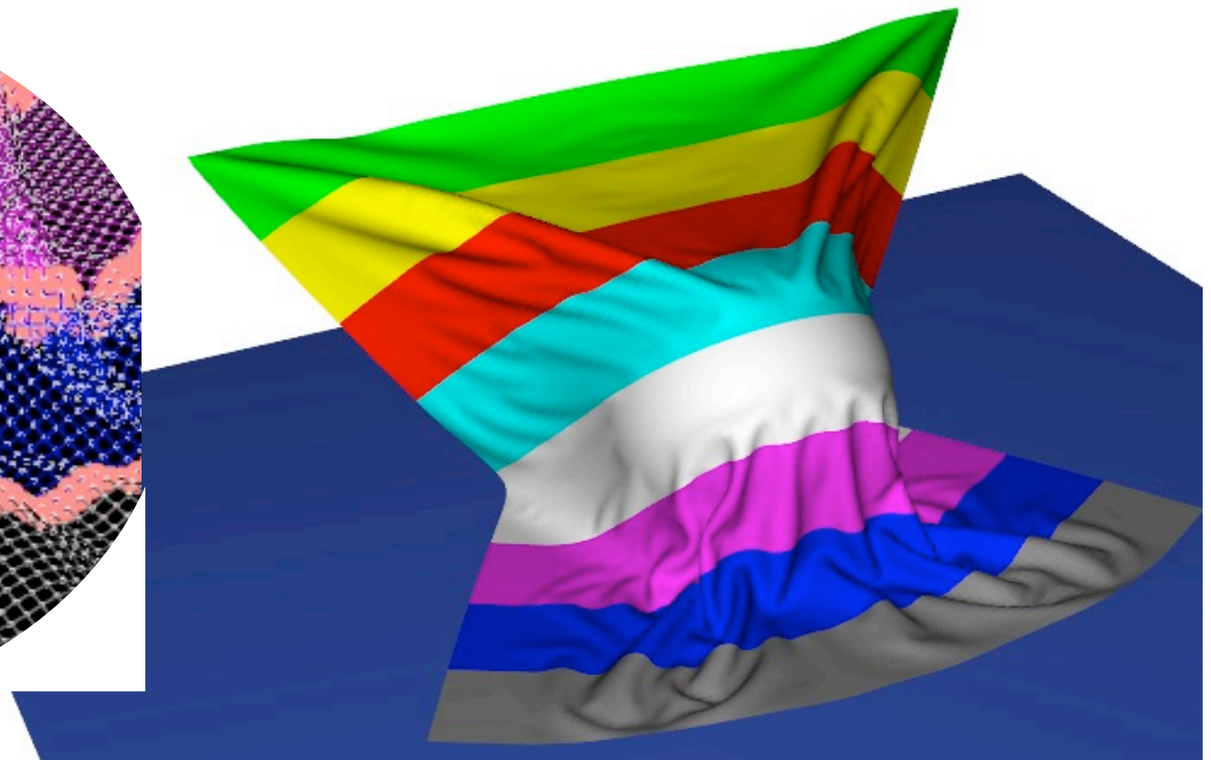


Graph Partitioning

- Classical parallelization : split the mesh in *p* partitions
- *p* threads can process each partition in parallel
  - *boundary* edges between partitions requires exchanges between threads
- Problem : only efficient if *p* << *n*
  - *Due to boundaries overhead*

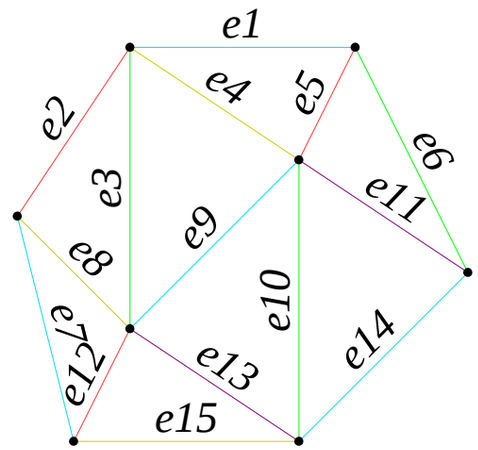


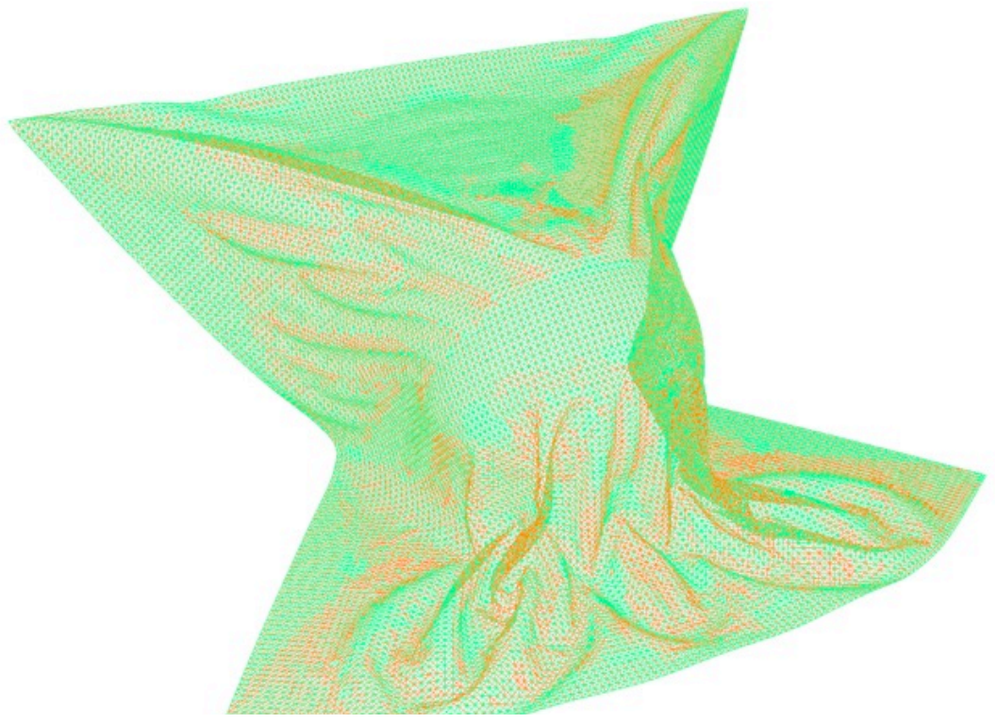




Graph Coloring

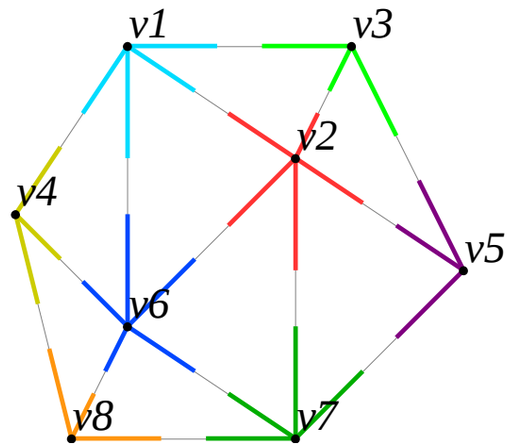
- Alternate parallelization : find *c* sets of unconnected edges
- Edges within a set can be processed in parallel
  - *m* / *c* average parallelism
  - requires *c-1* synchronization barriers
- Problem : c is at least equal to the max arity in the graph
  - 12 in typical mass-spring cloth meshes
- Needs to be recomputed if graph changes





Gather instead of Scatter

- Remove write conflicts by splitting each edge in two
- One thread per vertex, looping over all connected edges
  - *n* total threads
- Everything can be computed in one step
  - Computation of an edge duplicated on 2 threads
- Or an intermediate buffer can be used to :
  - First compute and store values of each edge (*m* threads)
  - Then accumulate results on each vertex (*n* threads)



END...