COURS MASTER IVI

HAPTIC RENDERING & SIMULATION SOFTWARE ARCHITECTURE

Sources: Ming Lin, Pierre Frédérick Villard

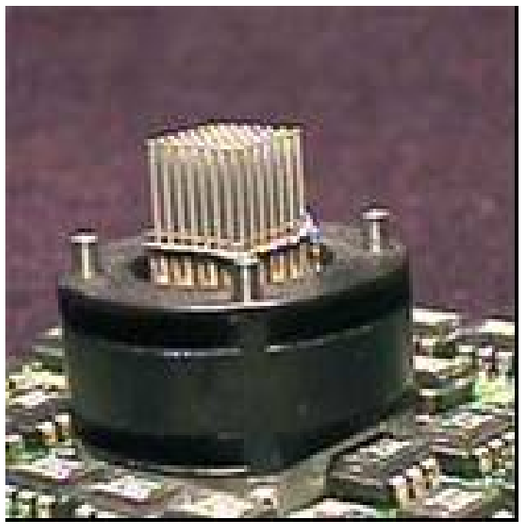
IMPORTANCE & PARTICULARITY OF HAPTIC FEEDBACK

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

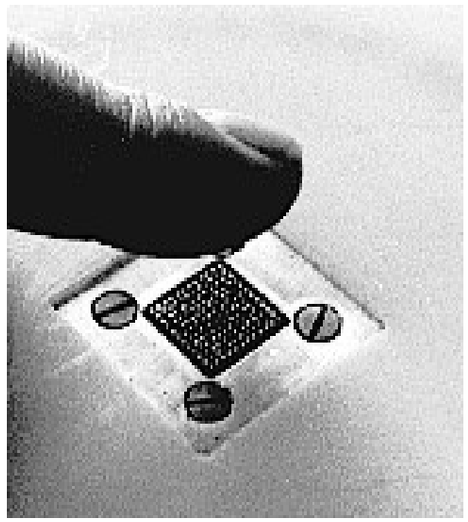
L'Incrédulité de saint Thomas, Le Caravage

HAPTIC = TACTILE & KINESTHESIS

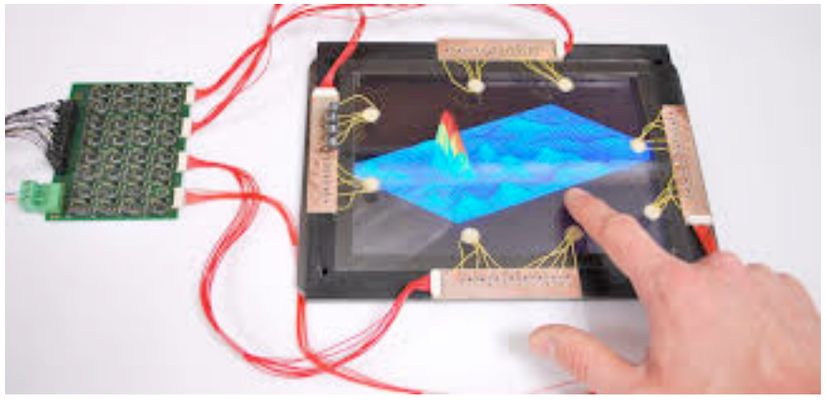
- Tactile devices…



STReSS tactile display



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



CEA

HAPTIC: FROM REAL TO VIRTUAL ENVIRONMENTS

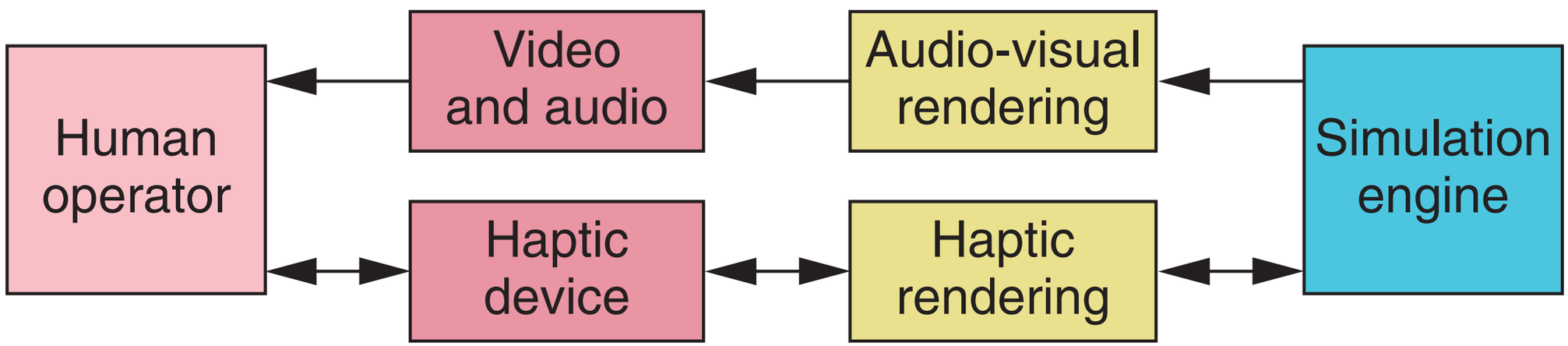
- Teleoperation with haptic feedback

HAPTIC: FROM REAL TO VIRTUAL ENVIRONMENTS

- Haptic on virtual environment

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

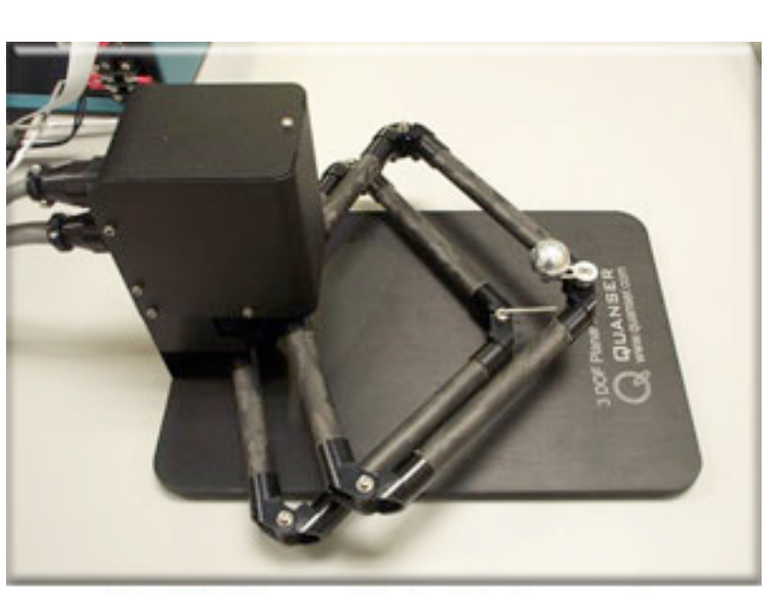
- Parallel architecture

HAPTIC INTERFACE DEVICES

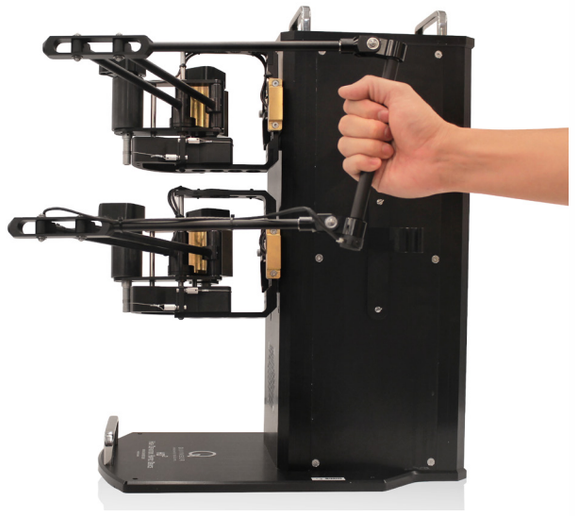
- Serial arms
  - large workspace, « simple » models
  - low stiffness
- Parallel robots
  - stiff and light robots
  - complex model and workspace

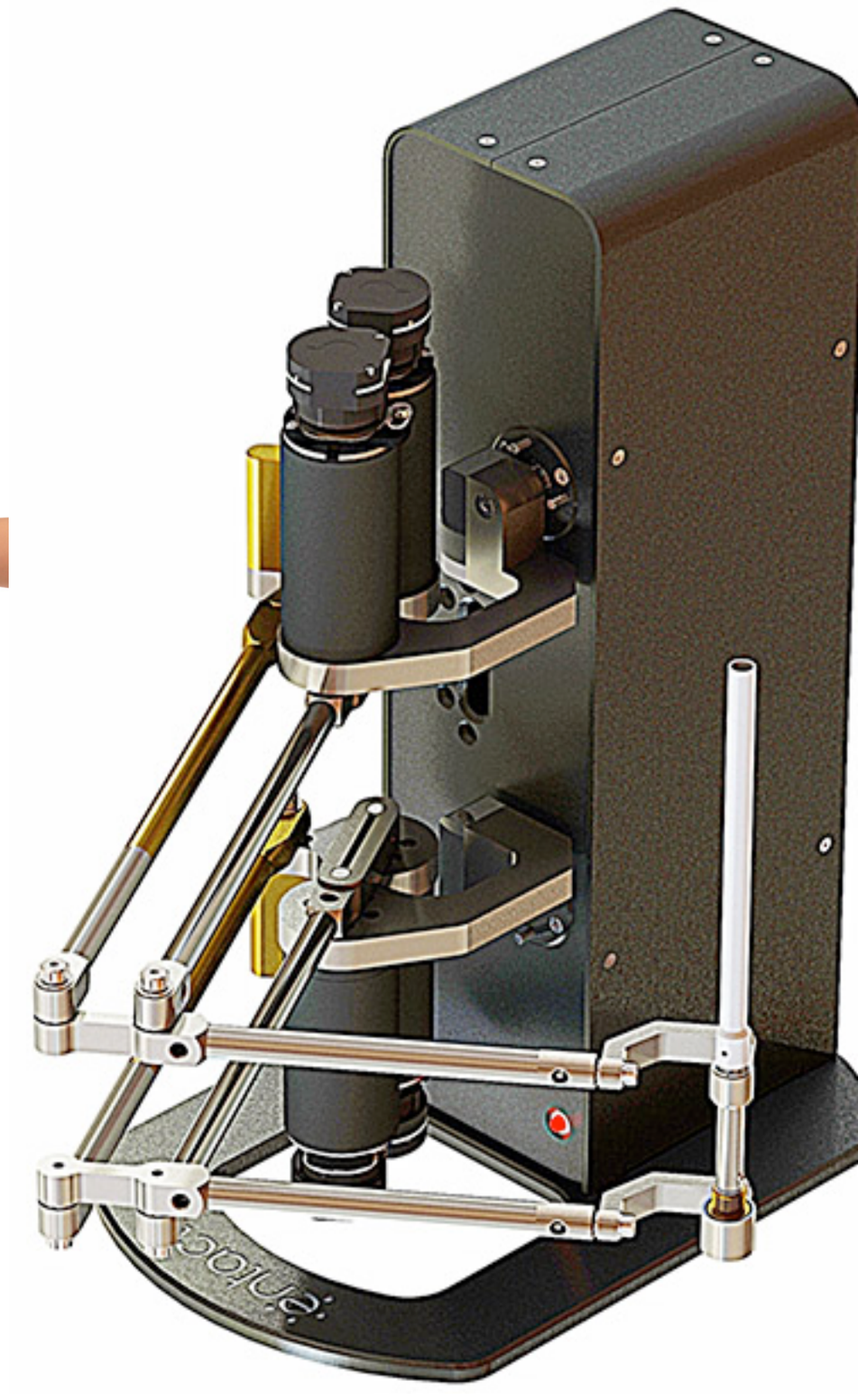
HAPTIC INTERFACE DEVICES

- Hybrid architectures



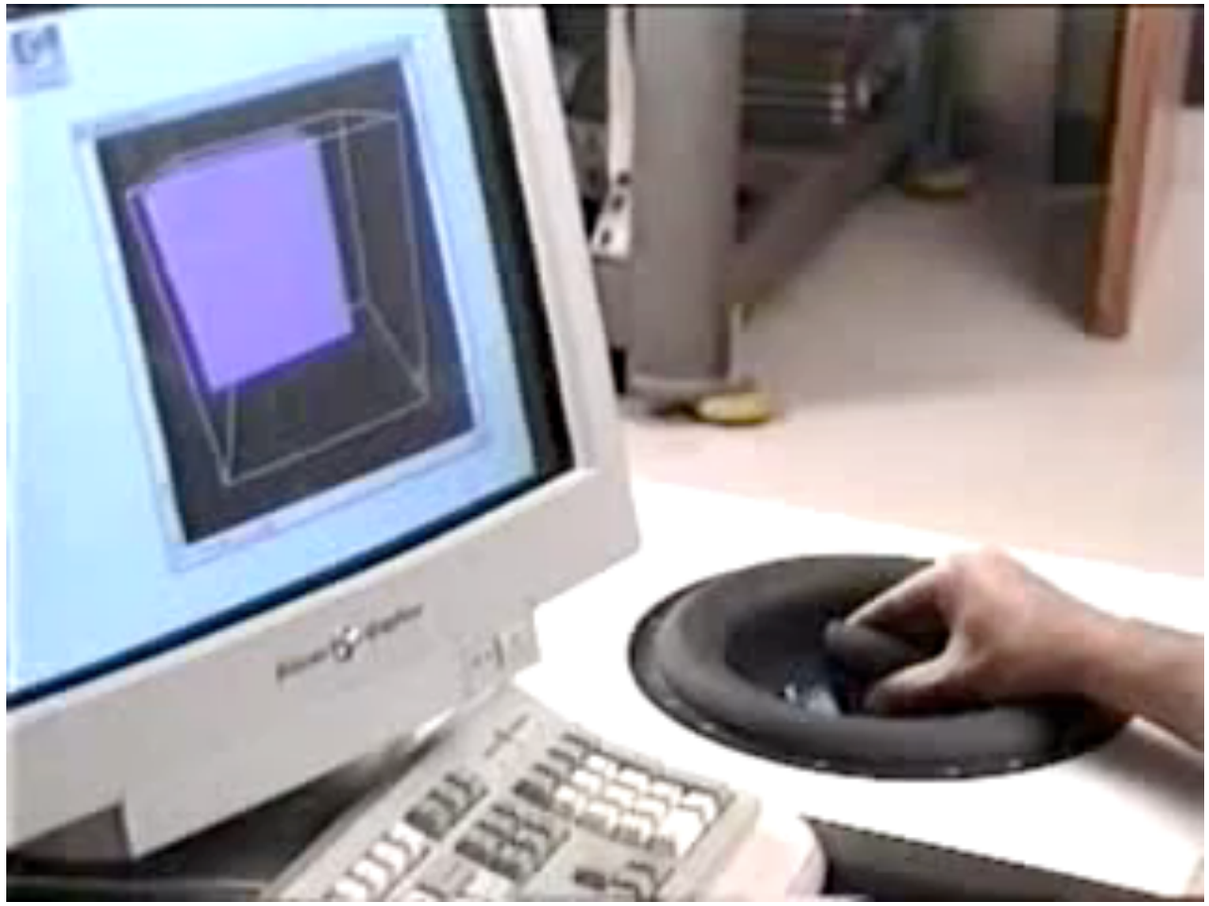
3DOF Planar Twin Pantograph





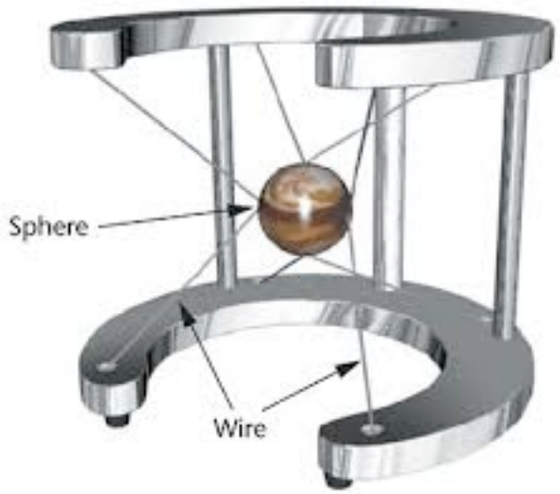
HAPTIC INTERFACE DEVICES

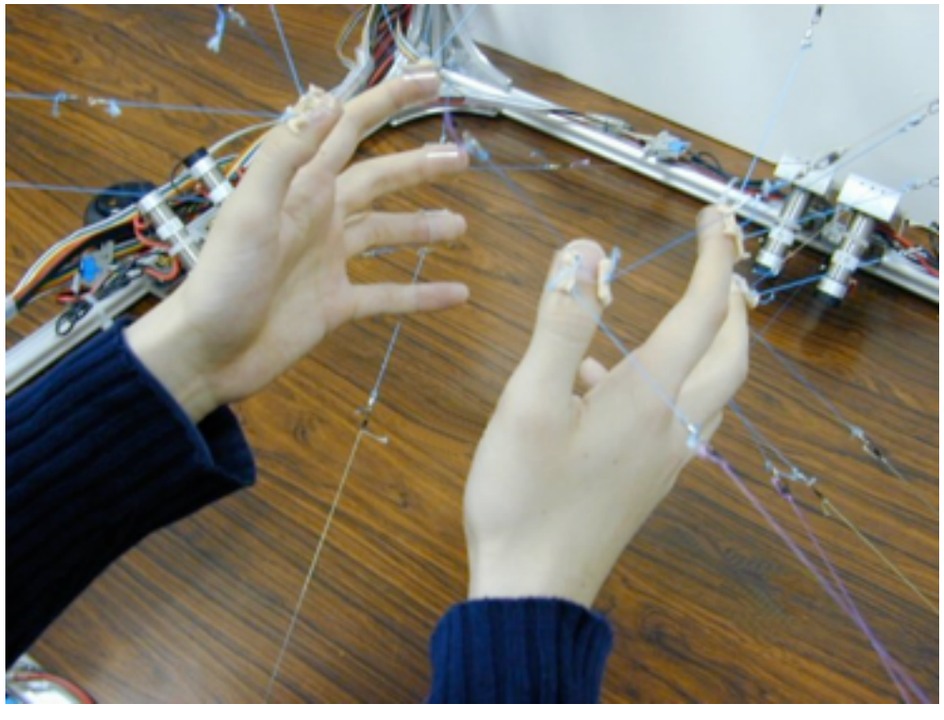
- Magnetic Levitation Haptic Interfaces



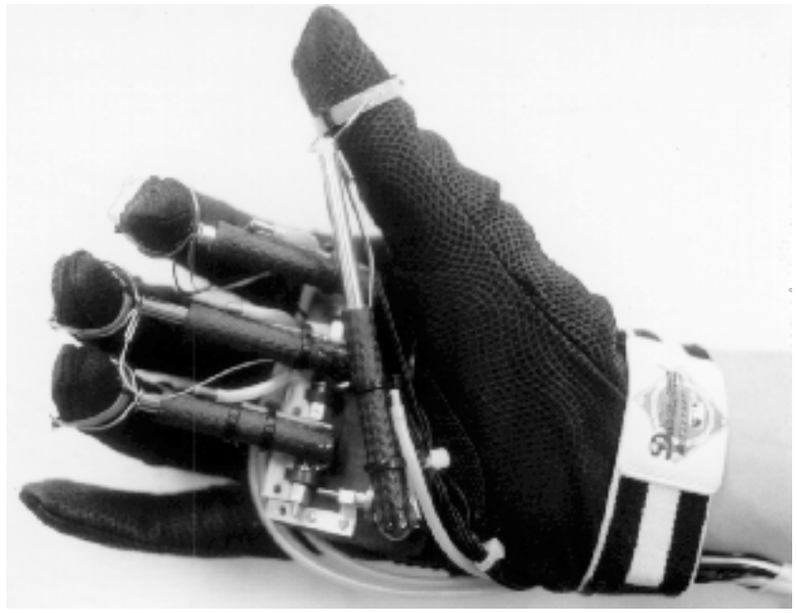
HAPTIC INTERFACE DEVICES

- Cables

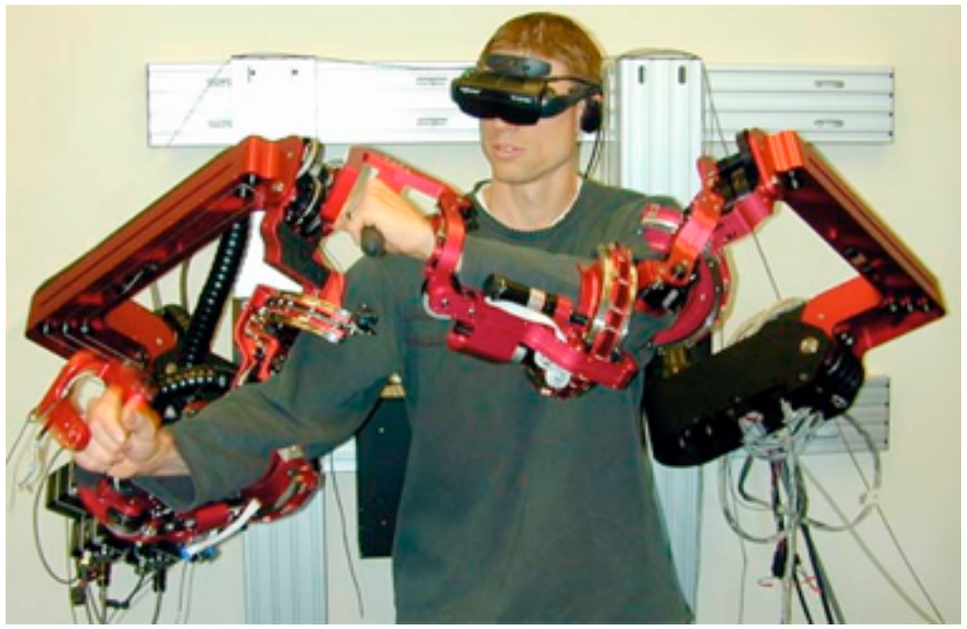


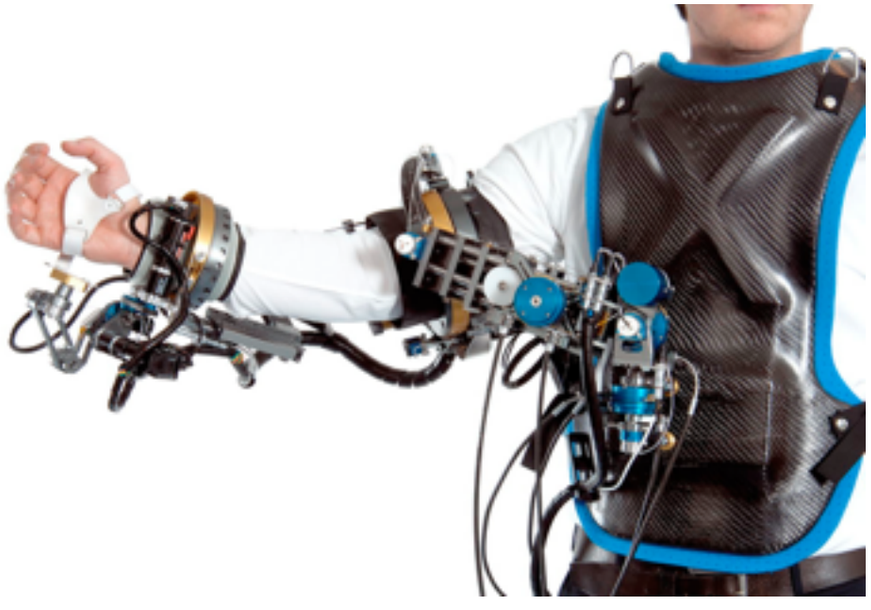


GLOVES



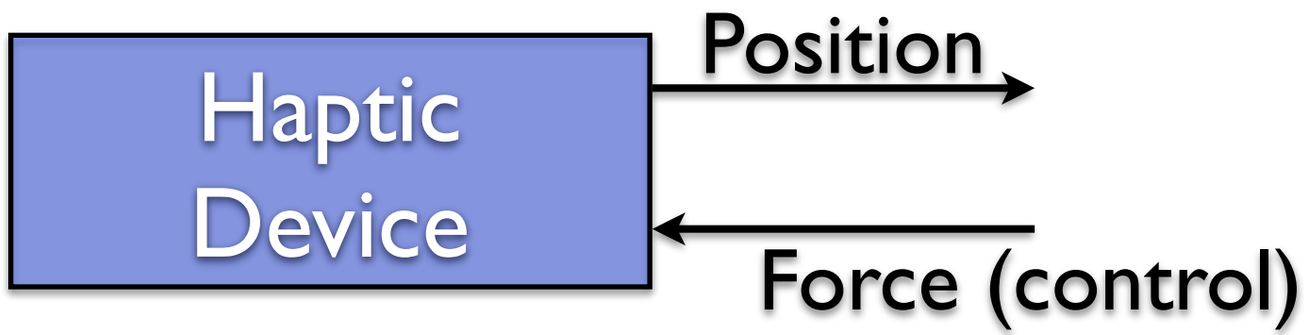
EXOSKELETONS



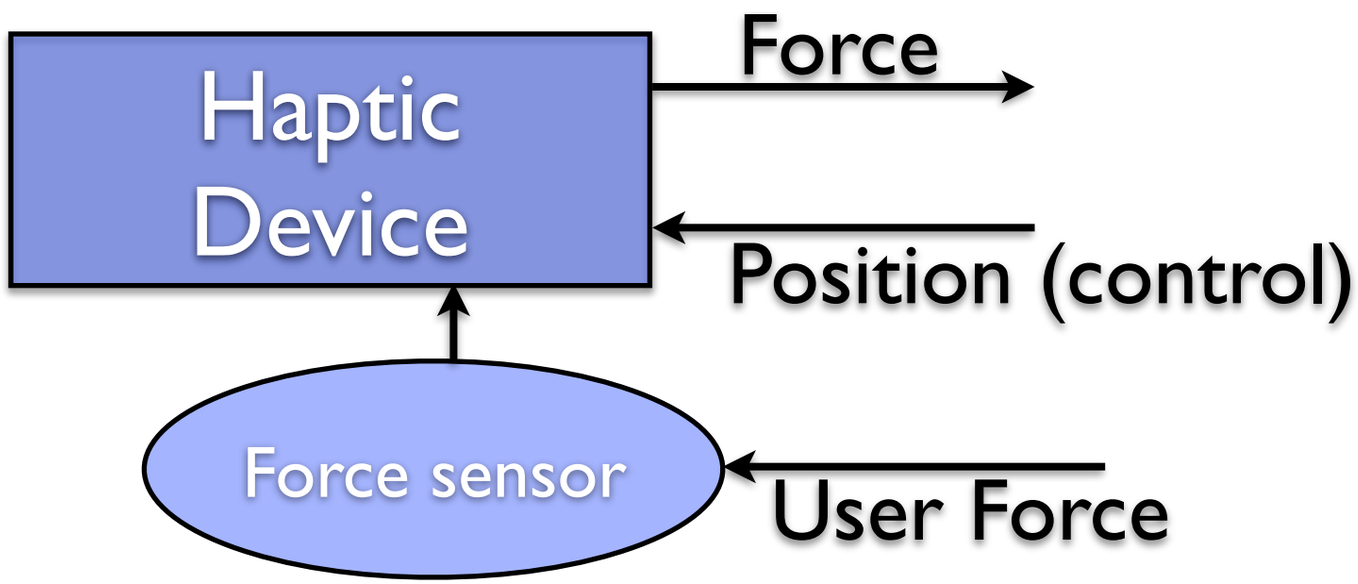


HAPTIC INTERFACE DEVICES

- Impedance / Admittance devices



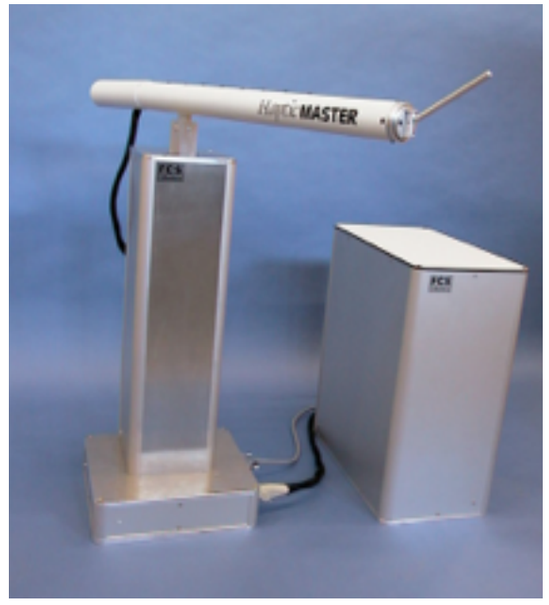
Reversible robot



Not Reversible robot

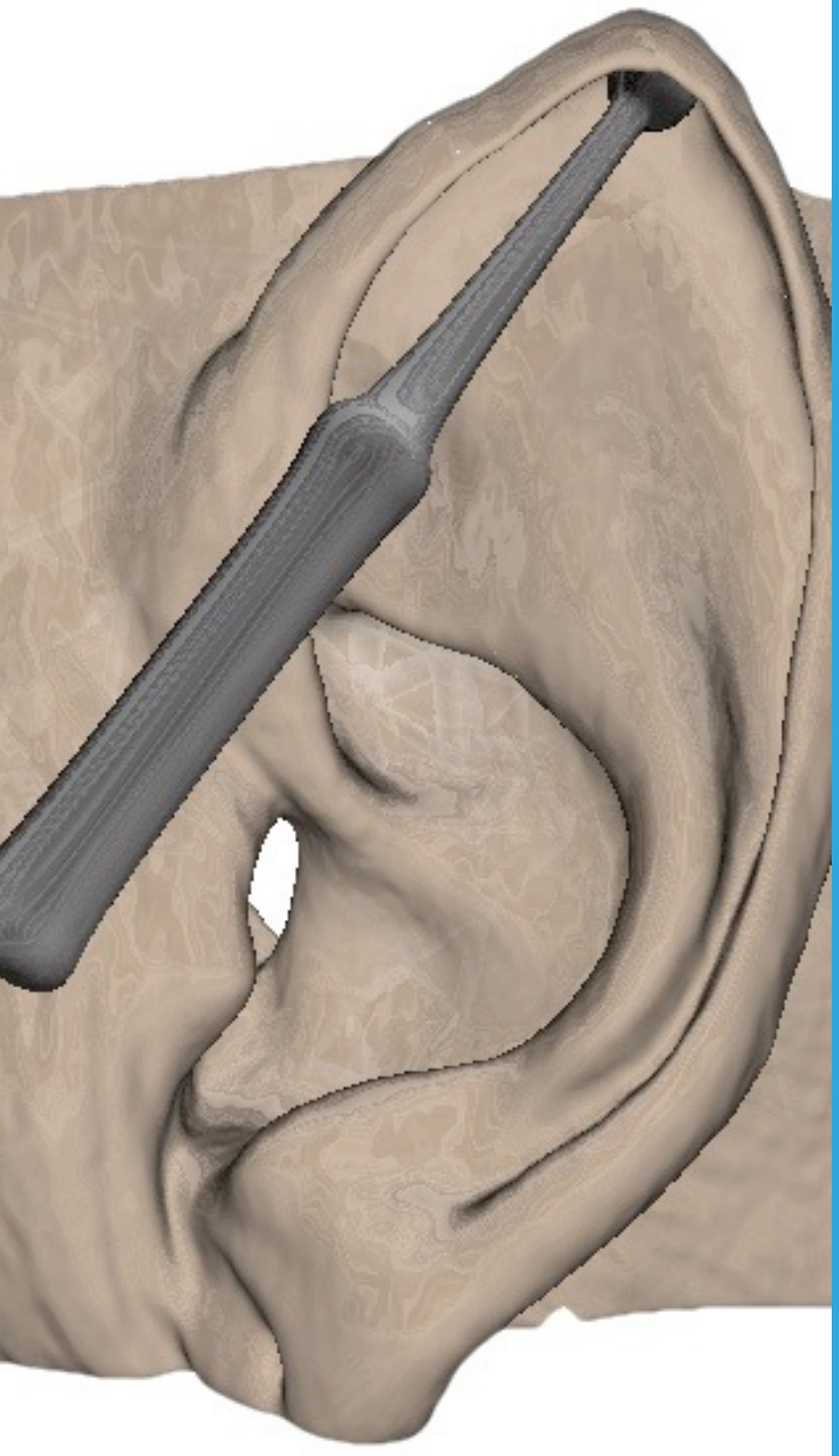
ADMITTANCE DEVICES

- 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

THE USER IS IN THE LOOP !!!

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

HAPTIC RENDERING ISSUES

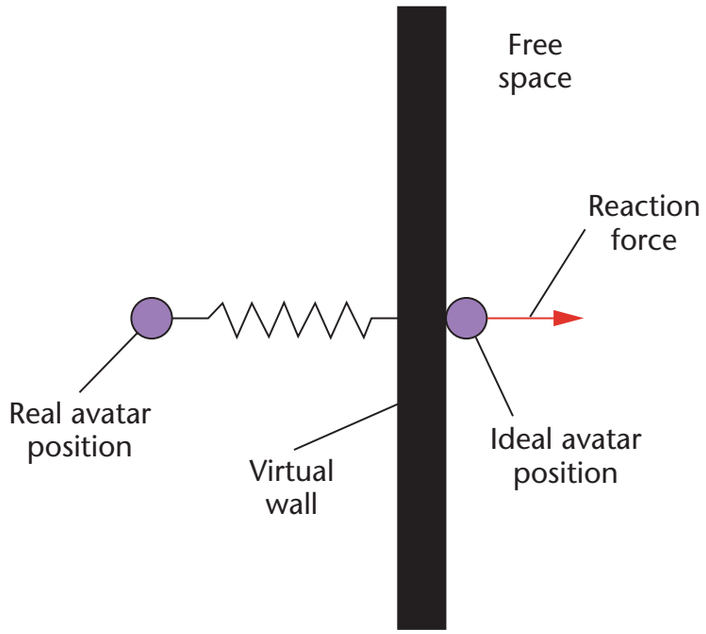
- **Stability**: Haptic device is a robotic arm !! The control must be stable
  - When coupling, both simulation and control must be stable !
- **Transparency**: the force that is transmitted to the user is supposed to be the actual force, computed in the simulation
  - Computation of forces in the simulation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

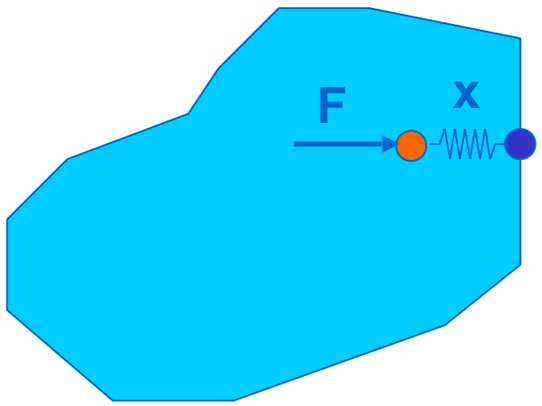
1KHZ PERFORMANCE REQUIREMENT

- The user becomes part of the simulation loop.
- 1KHz is necessary so that the whole system doesn't suffer from disturbing oscillations.
  - Think of the analogy with numerical integration of a system with spring, mass and damper, where the frequency of the haptic loop sets the integration step.
- The Phantom haptic devices run their control loop at 1KHz.
- Consequence: we are very limited on the amount of computation that we can do.

VIRTUAL WALL CONCEPT (BASIC RENDERING)

- Check if point penetrates an object.
- Find closest point on the surface.
- Penalty-based force.



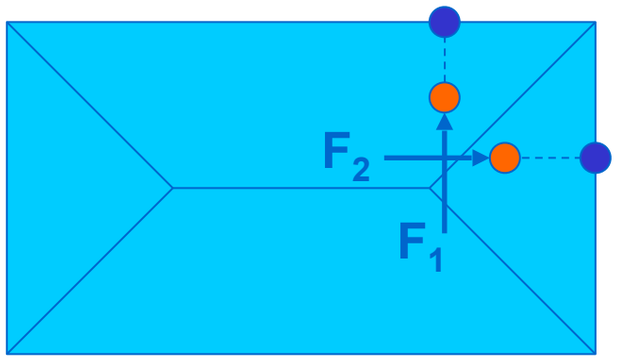


$$F = k \cdot x$$

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

HAPTIC RENDERING ISSUES

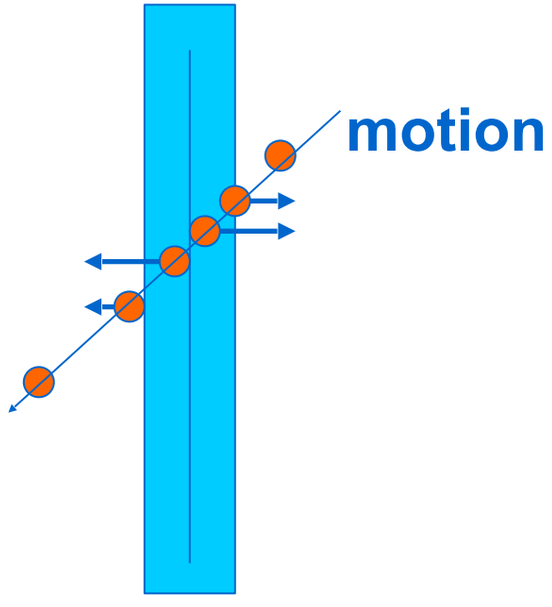
- Force discontinuities when crossing boundaries of internal Voronoi cells.



Unexpected force discontinuities (both in magnitude and direction) are very disturbing!

HAPTIC RENDERING ISSUES

- Pop-through thin objects.



After the mid line is crossed, the force helps popping through.

HAPTIC RENDERING ISSUES

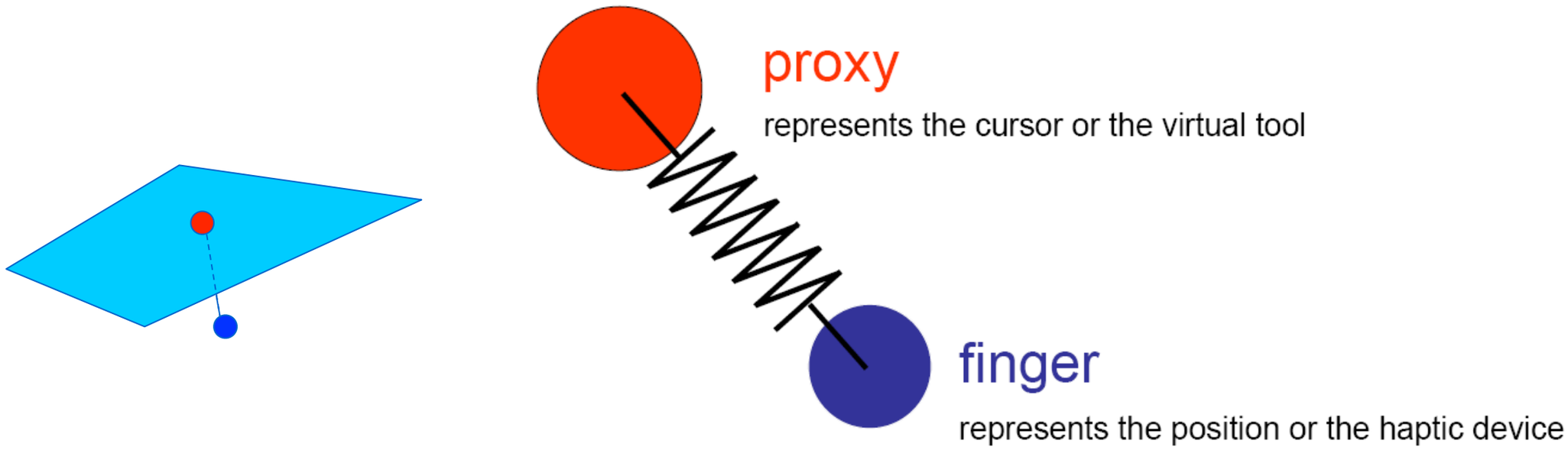
- Stability
  - Related to passivity: the simulation and the control should never add energy to the system… (Add damping… but…)
  - Time step must be as small as possible (as delay creates energy): often use of 500Hz / 1kHz
  - Related to robustness: the simulation must be robust to any gesture of the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

HAPTIC RENDERING ISSUES

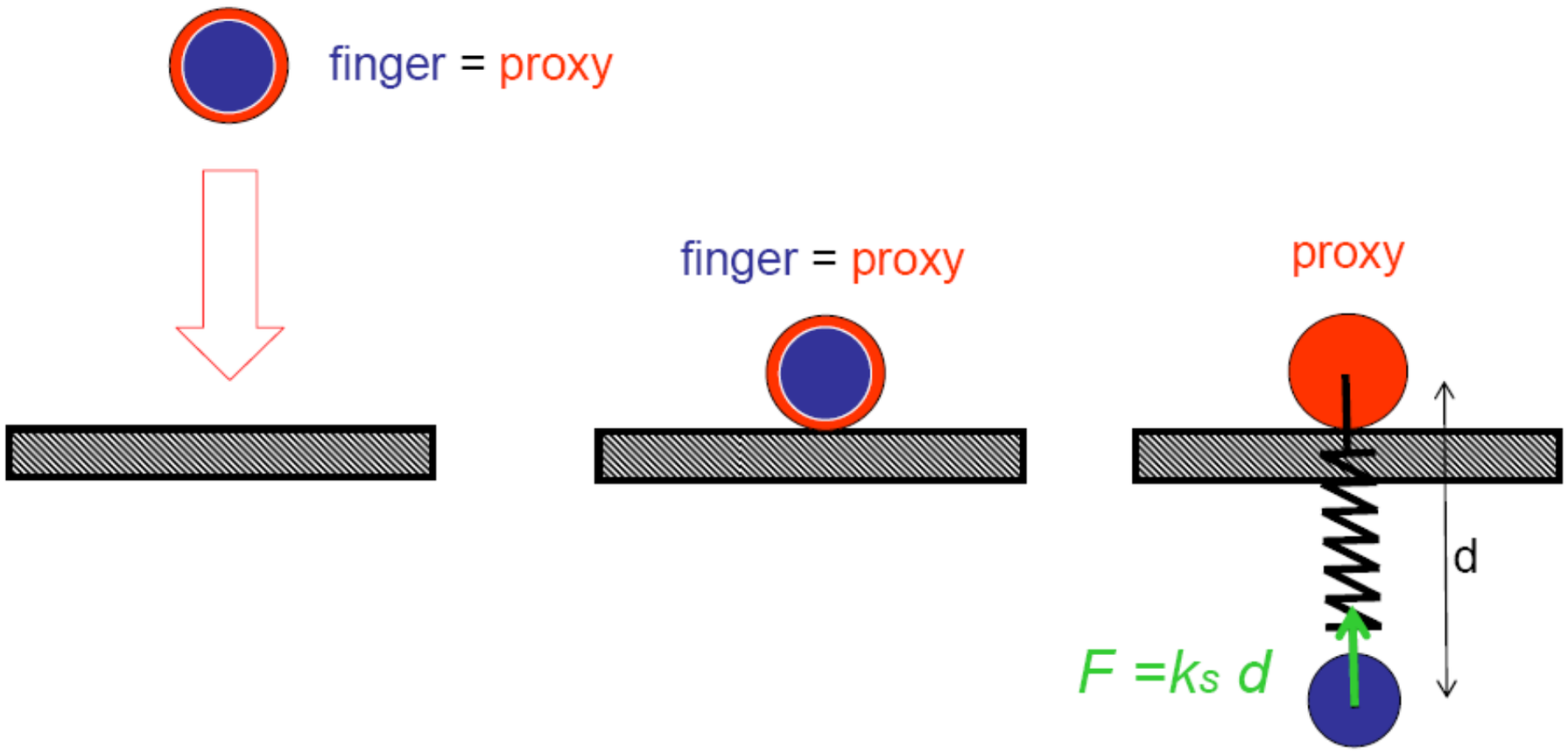
- Transparency
  - Problem with Damping forces
    - Damping helps the stability but is very bad for transparency...
  - Depends on contact response model
  - Penalty methods / Constraint-based approaches.

POINT TO PLANE METHOD (MARK ET AL., 1996.)

- For distributed applications. The simulator sends the equation of a plane to the haptic loop. The update of the plane is asynchronous.
- Forces are computed between the haptic point and the plane.
- Possible stiffness is limited, because too stiff would bring a jerky behavior at low update rates.



SIMPLE PROXY



GOD-OBJECT (ZILLES AND SALISBURY, 1995).

- Use the position of the haptic interface point (HIP) and a set of local constraint surfaces to compute the position of god-object (GO).
- Compute GO as the point that minimizes the distance from HIP to the constraint surfaces. Lagrange multipliers.

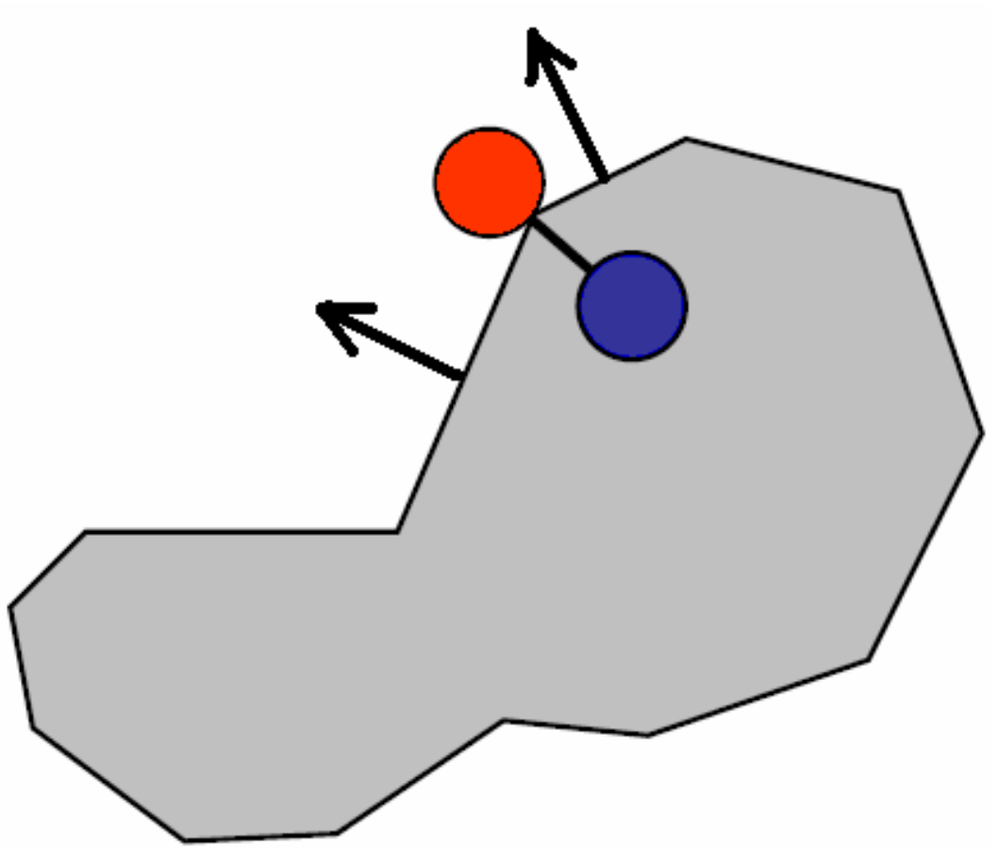
$$\text{minimize} \|x - p\| \text{subject to}$$

$$\hat{n}_1^T x \geq 0,$$

$$\hat{n}_2^T x \geq 0,$$

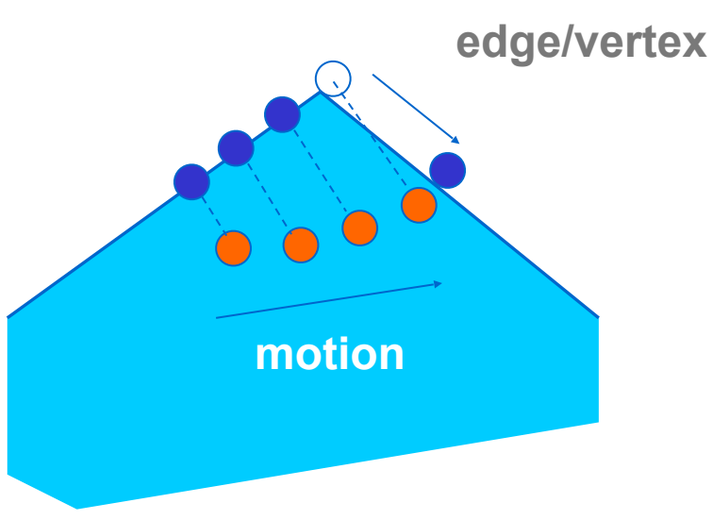
$$\vdots$$

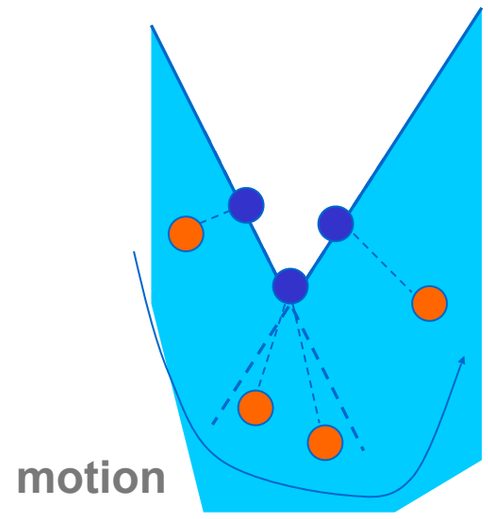
$$\hat{n}_m^T x \geq 0,$$



GOD-OBJECT

- Constraint surfaces:
  - Surfaces impeding motion
  - GO is outside (orientation test) and in the extension of the surface.
  - The HIP is inside the surface.





Concavity: 2/3 constraint planes in 3D

GOD-OBJECT

- Constraint plane equations:

$$A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i = 0$$

- Energy function that will account for the distance.

$$E = \frac{1}{2} \cdot k \cdot \left((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2 \right)$$

- Define cost function using Lagrange multipliers.

$$C = \frac{1}{2} \cdot k \cdot \left((x - x_{HIP})^2 + (y - y_{HIP})^2 + (z - z_{HIP})^2 \right) + \sum_{i=1}^{3} \lambda_i \cdot (A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i)$$

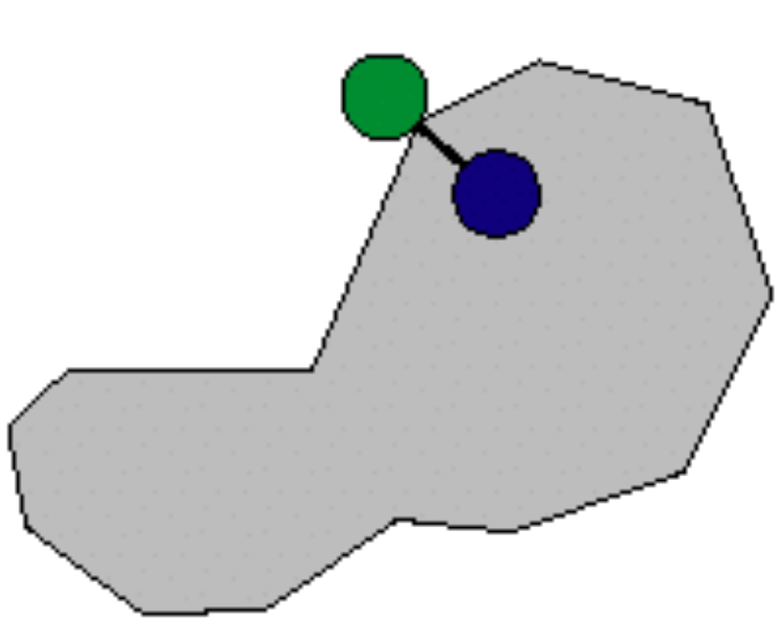
- Minimize, solving for x, y, z, λ1, λ2 and λ3.

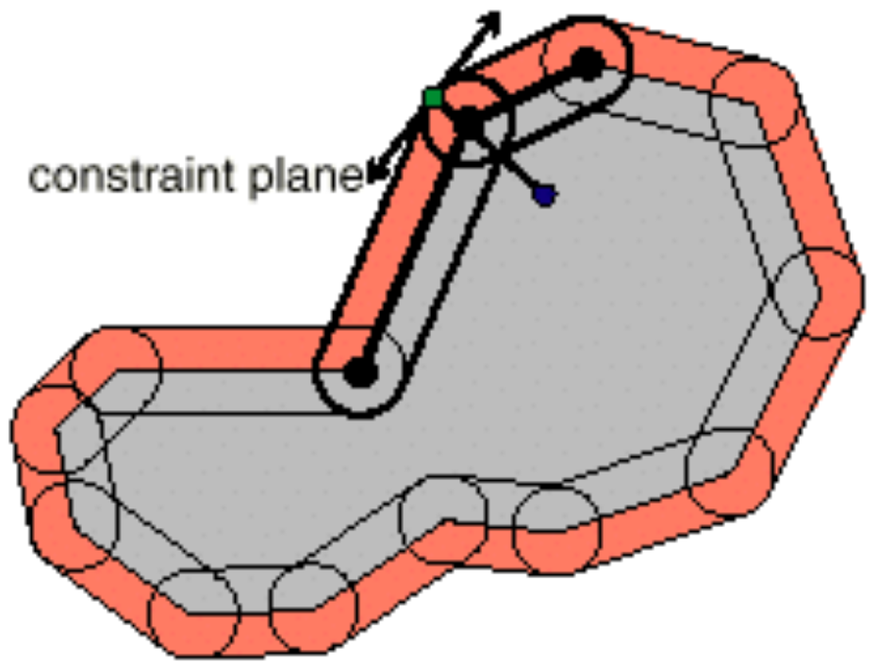
VIRTUAL PROXY (RUSPINI ET AL., 1997.)

- Ruspini et al., 1997.
- Based on god-object.
- Virtual proxy is a small sphere, instead of a point. Use configuration-space obstacles (C-obstacles), from robotics.
- More formal definition of constraint planes.
- Implementation of additional features, based on relocation of the virtual proxy.

VIRTUAL PROXY

- C-obstacles: for a spherical object, is reduced to computing offset surfaces at a distance equal to the radius of the sphere.
- Check the HIP against the offset surface.
- This is done to avoid problems with small gaps in the mesh.

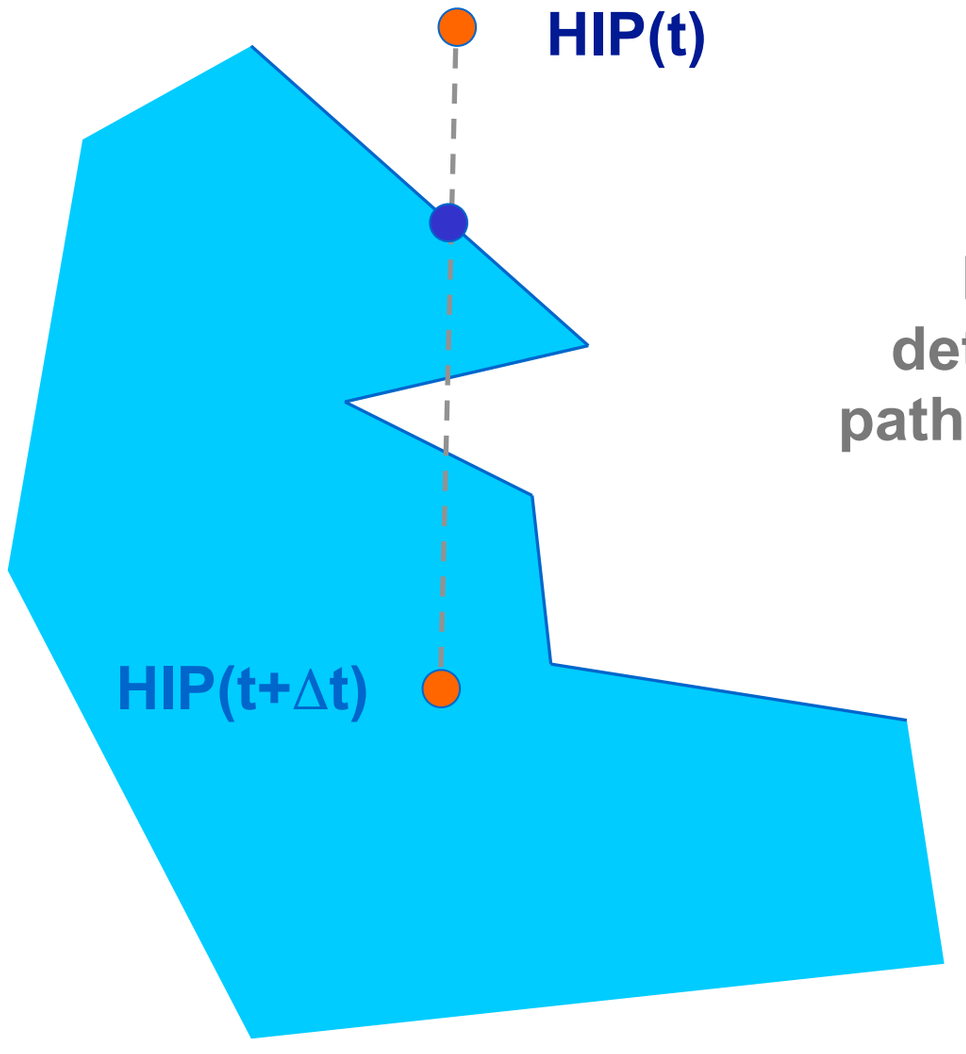




VIRTUAL PROXY

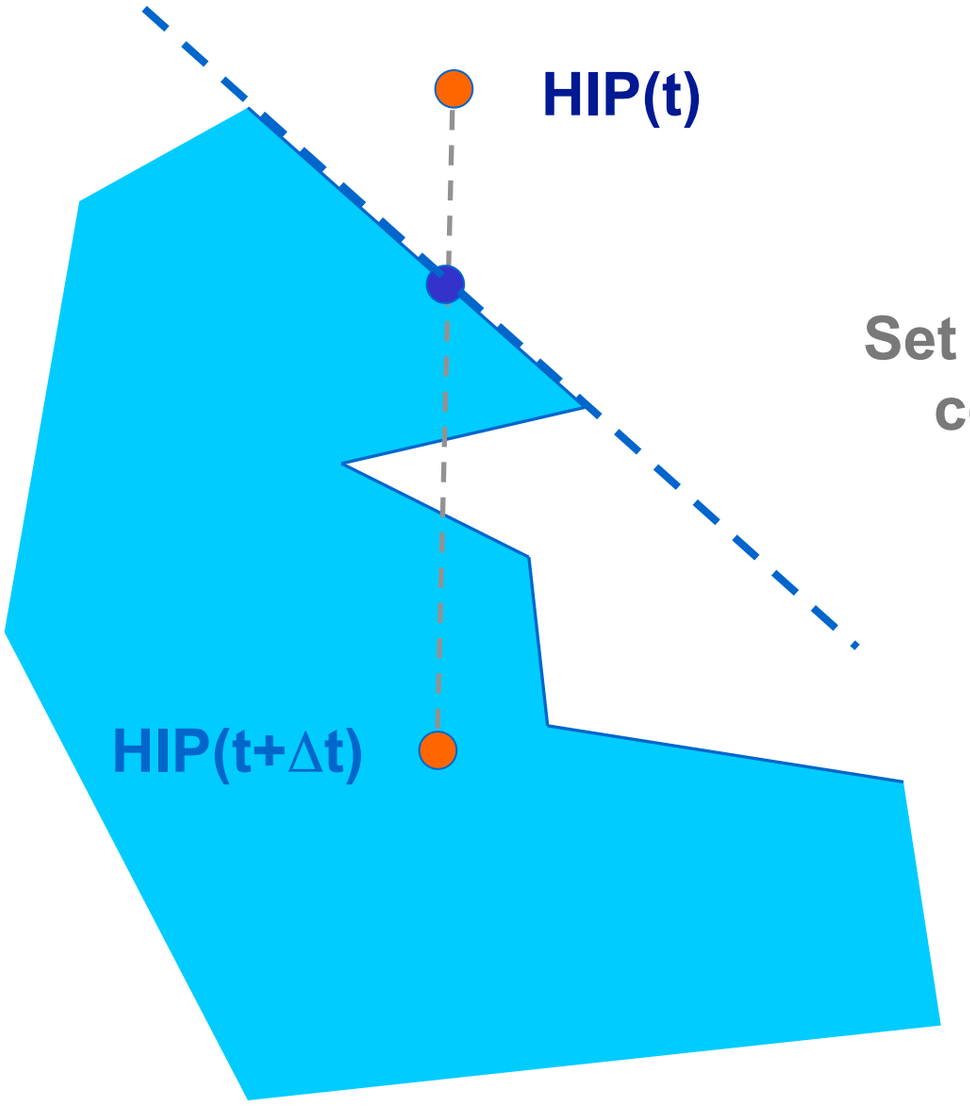
- Finding the virtual proxy is based on an iterative search.
- Find subgoals based on the same distance minimization as for the god-object.
- At each subgoal, all the planes that go through that point are potential constraints. The minimum set of active constraints is selected.
- If the subgoal is in free space, set as new subgoal the HIP. The path might intersect the C-obstacles. Add the first plane intersected as a constraint and the intersection point as the current subgoal.
- The process ends when the virtual proxy becomes stable.

VIRTUAL PROXY



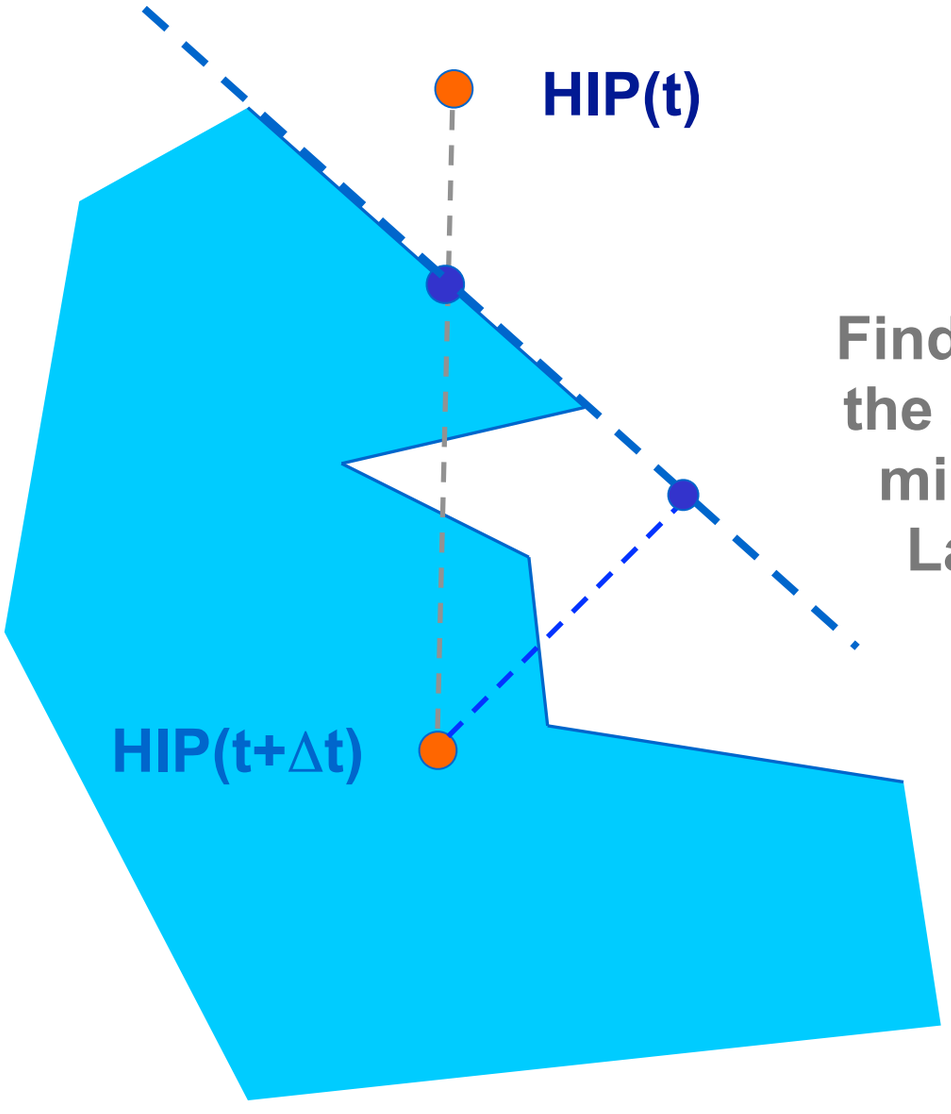
Perform collision detection between the path of the HIP and the C-obstacles

VIRTUAL PROXY



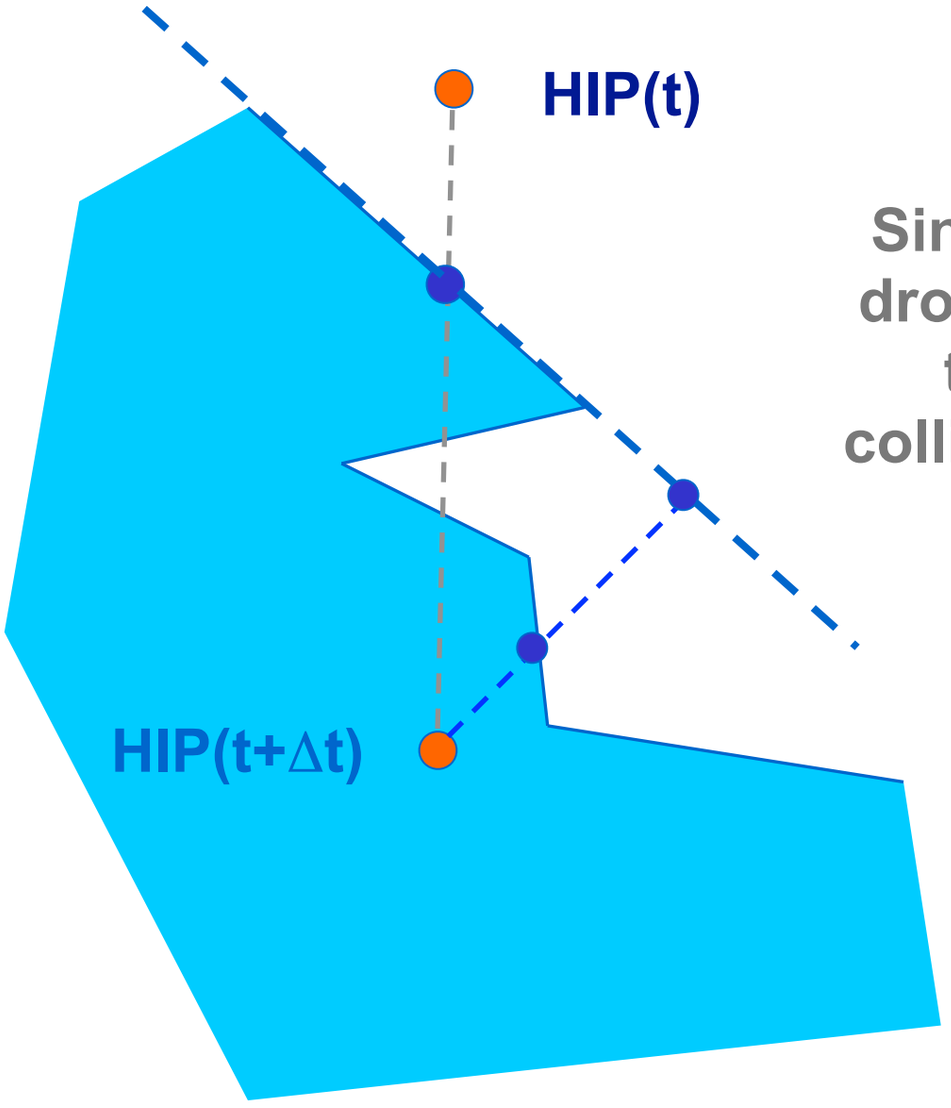
Set the subgoal and the constraint plane(s)

VIRTUAL PROXY



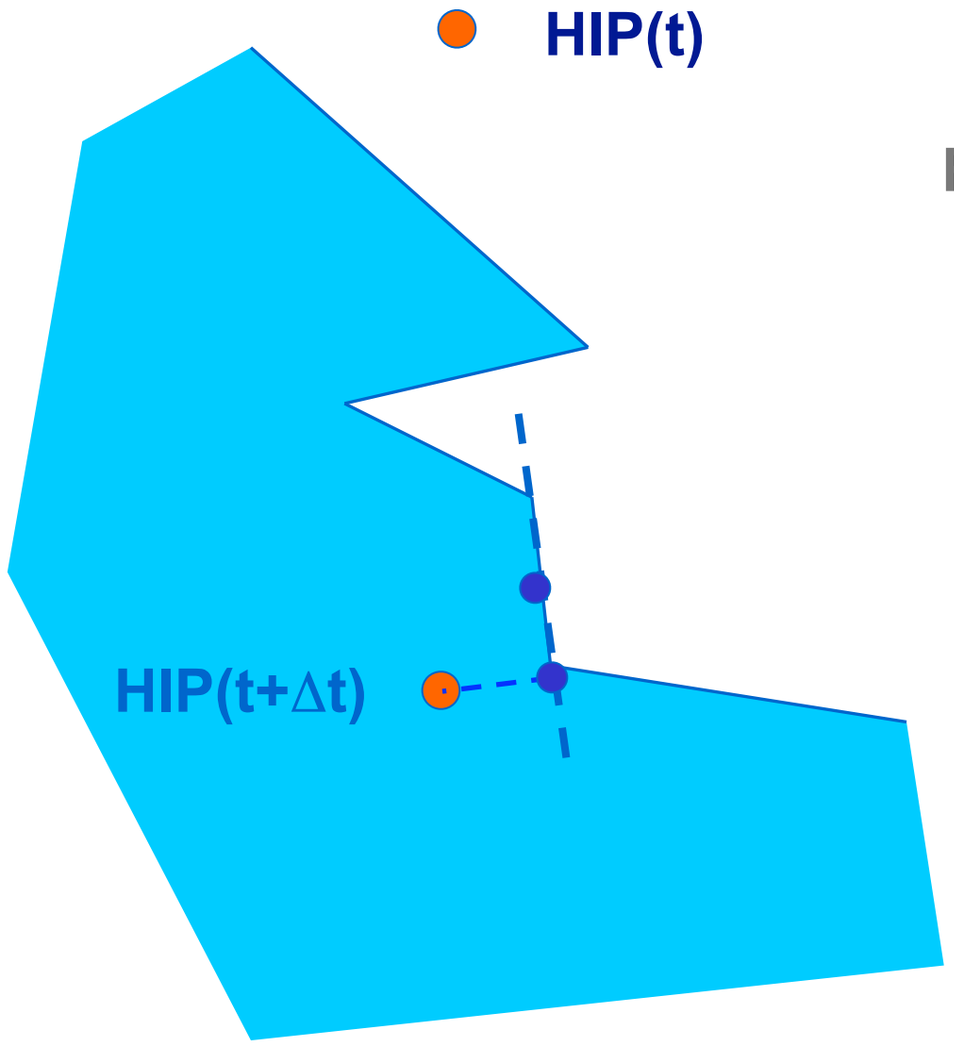
Find a new subgoal using the active planes and the minimization based on Lagrange multipliers

VIRTUAL PROXY



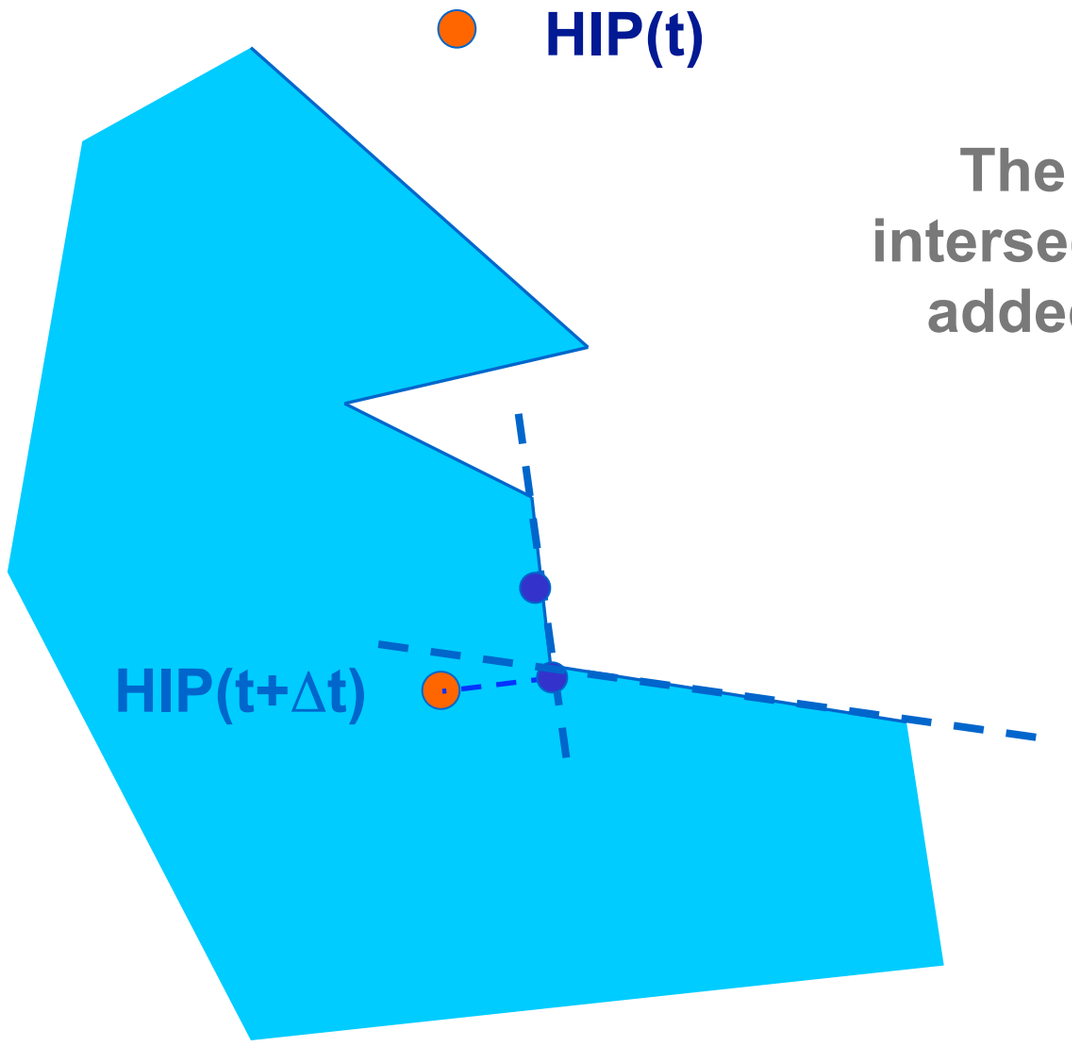
Since the subgoal is in free space, drop the constraints, set the HIP as the new subgoal and perform collision detection between the path and the C-obstacles

VIRTUAL PROXY



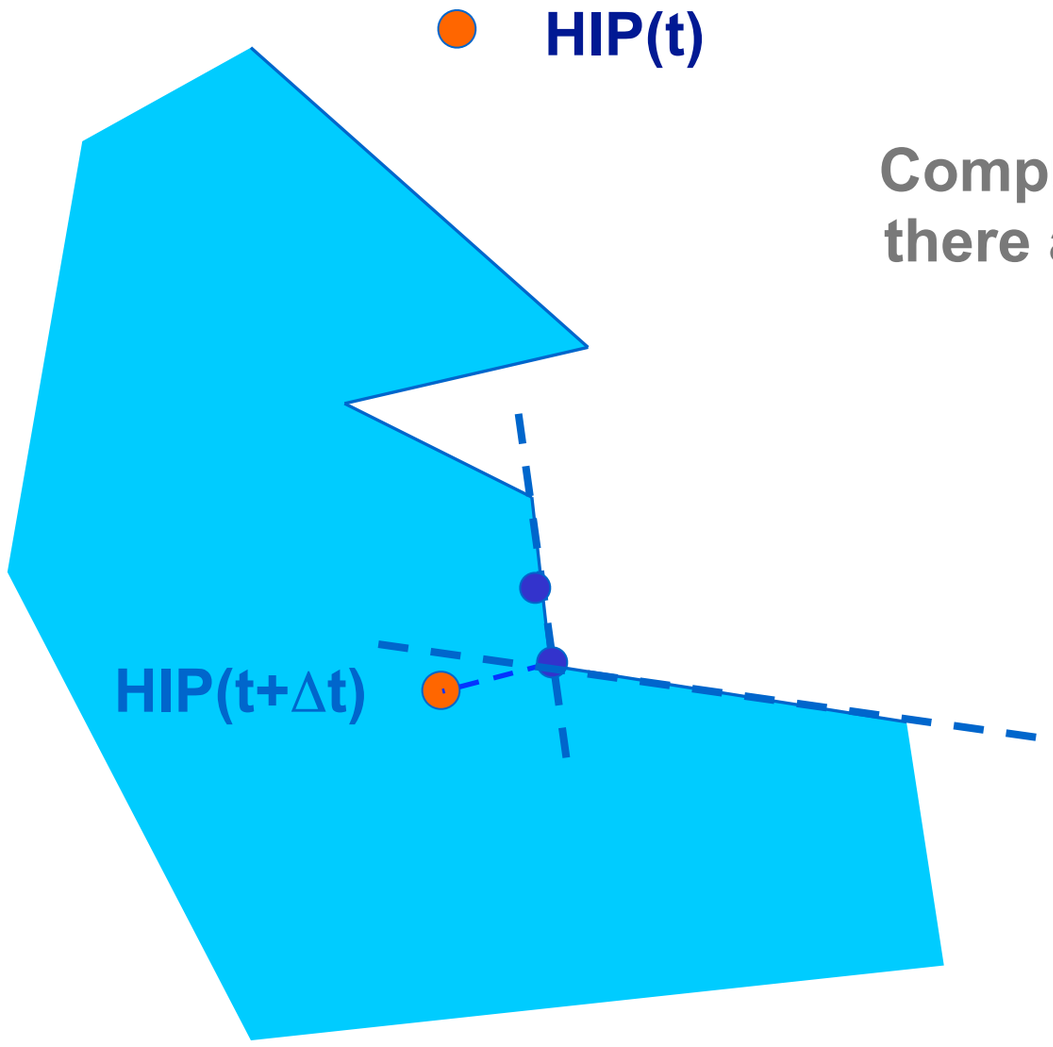
Recompute subgoal with new constraints

VIRTUAL PROXY



The path to the new subgoal intersects another plane, so this is added to the set of constraints

VIRTUAL PROXY



Compute active constraints (in 2D there are only 2) and find subgoal

For this example, this is the final position of the virtual proxy

VIRTUAL PROXY

- Quadratic programming approach:
  - The constraint planes define an open convex region (bounded by the plane at infinity).
  - The function to minimize is the distance from the haptic device (HIP) to the new subgoal (VPi+1):

$$C = \left\| VP_{i+1} - HIP(t + \Delta t) \right\|$$

Quadratic function

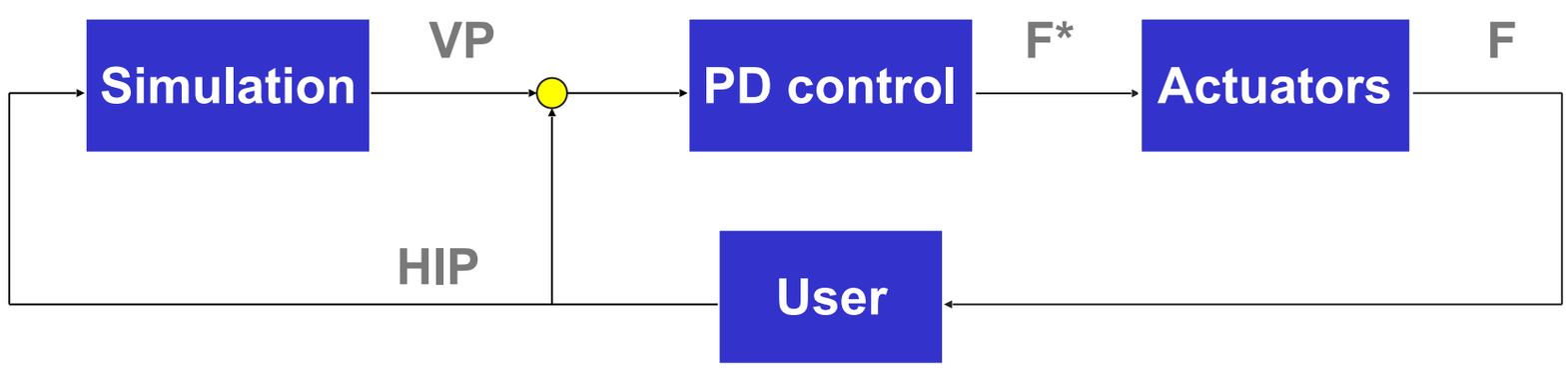
- The translation from the current location to the new subgoal cannot intersect the constraint planes. Define linear constraints based on the normals of the planes.

$$N_i \cdot \left(VP_{i+1} - VP_i \right) \geq 0$$

Linear constraints

VIRTUAL PROXY

- Force output: PD (proportional- derivative) control. Produces a force that will try to keep the VP and the HIP at the same position.

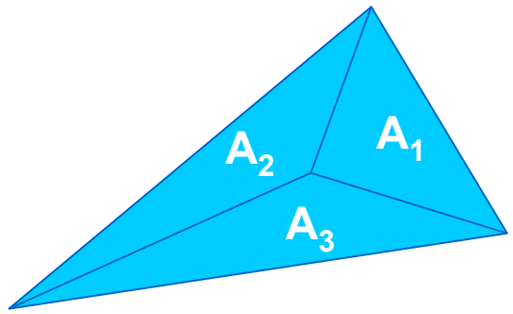


$$F^* = K_p \cdot (VP - HIP) + K_d \cdot \frac{d(VP - HIP)}{dt}$$

It's like a spring+damper, but the authors look at it from a control engineering approach

FORCE SHADING (BASDOGAN ET AL 97.)

- Interpolate per-vertex normals using baricentric coordinates (same as Gouraud shading)

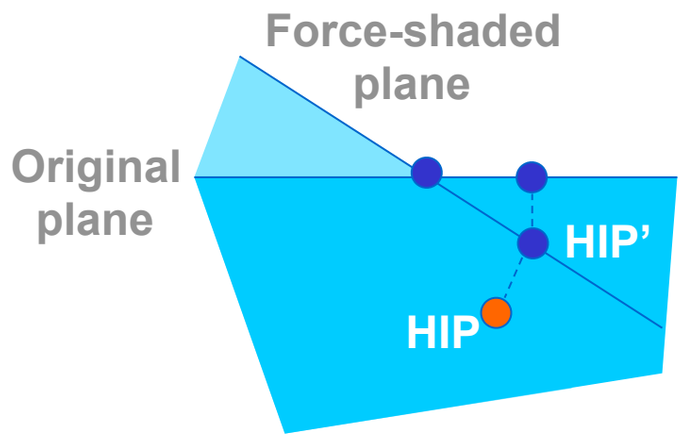


$$N' = \frac{\sum_{i=1}^{3} A_i \cdot N_i}{\sum_{i=1}^{3} A_i} \qquad N = \frac{N'}{\|N'\|}$$

- Effect: edges and vertices seem ‘rounded’

FORCE SHADING (RUSPINI ET AL., 1997)

- Modify the position of the VP.
- A subgoal (HIP') is computed changing the normal of the plane.
- This subgoal replaces the HIP, and the final VP is computed.

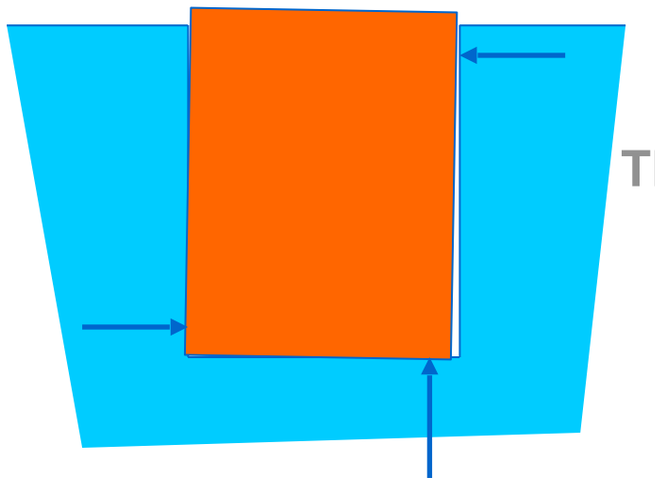


FORCE SHADING (RUSPINI ET AL., 1997)

- Other effects by Ruspini et al., 1997.
  - Friction.
  - Textures.
  - Deformation.
- All of them are achieved performing operations that modify the position of the VP.

6DOF RENDERING

- Output: 3D force + 3D torque
- For applications related to manipulation.
  - Assembly and maintenance oriented design. Removal of parts from complex structures.
- Typical problem: peg-in-the-hole.



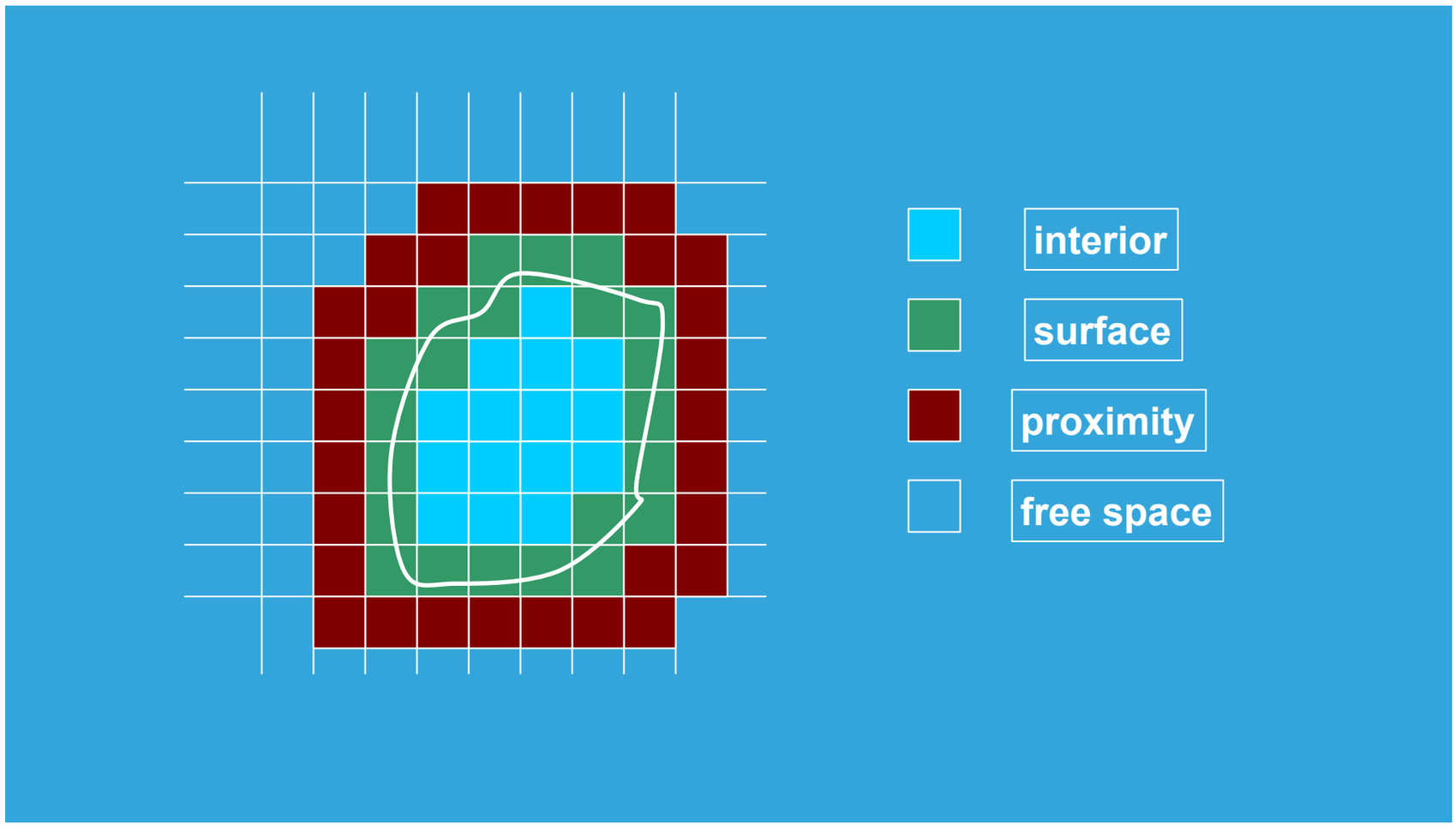
VOXEL SAMPLING

- McNeely et al., 1999.
- Voxelize all the models in the scene.
- For the haptic object, represent each voxel by a point -> Point Shell.
- Test points against voxels of the scene.
- Compute force at each point that penetrates an object.
- Add up all the force and torque, and apply them to the user through virtual coupling.

Original object **Voxels** **Pointshell**

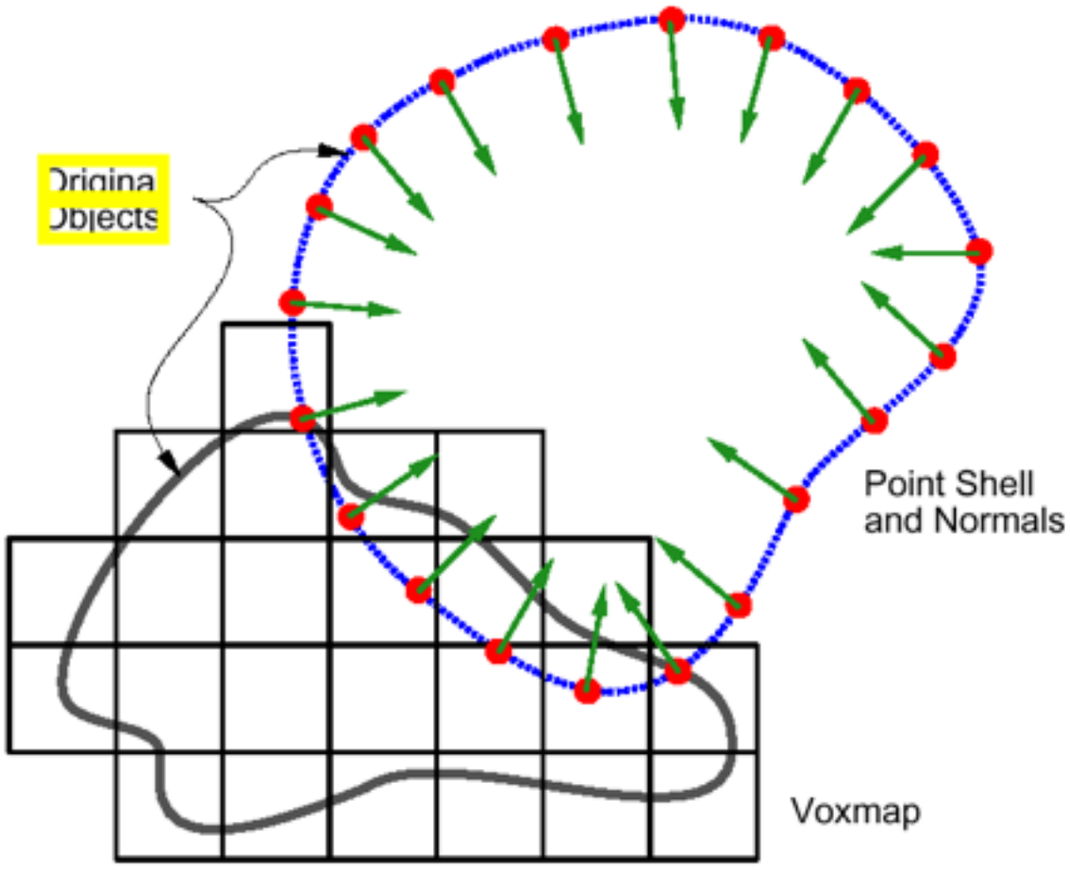
VOXEL SAMPLING

- All the voxels in the voxmap store info about their position in the object: interior, surface, proximity or free space.



VOXEL SAMPLING

- To start the collision query, all the pointshell has to be transformed to the local coordinates of the voxelized objects.
- Points that are in the surface or proximity voxels are considered as contacts.

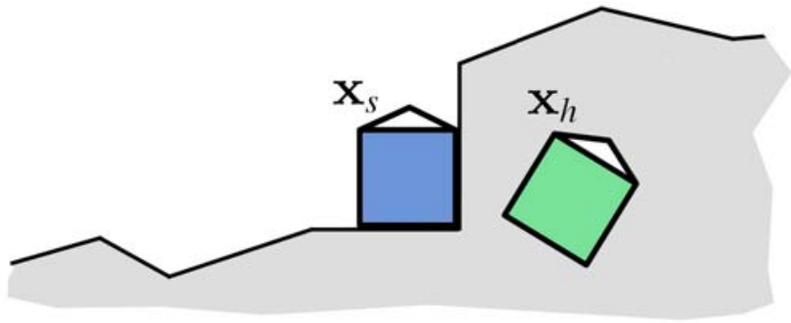


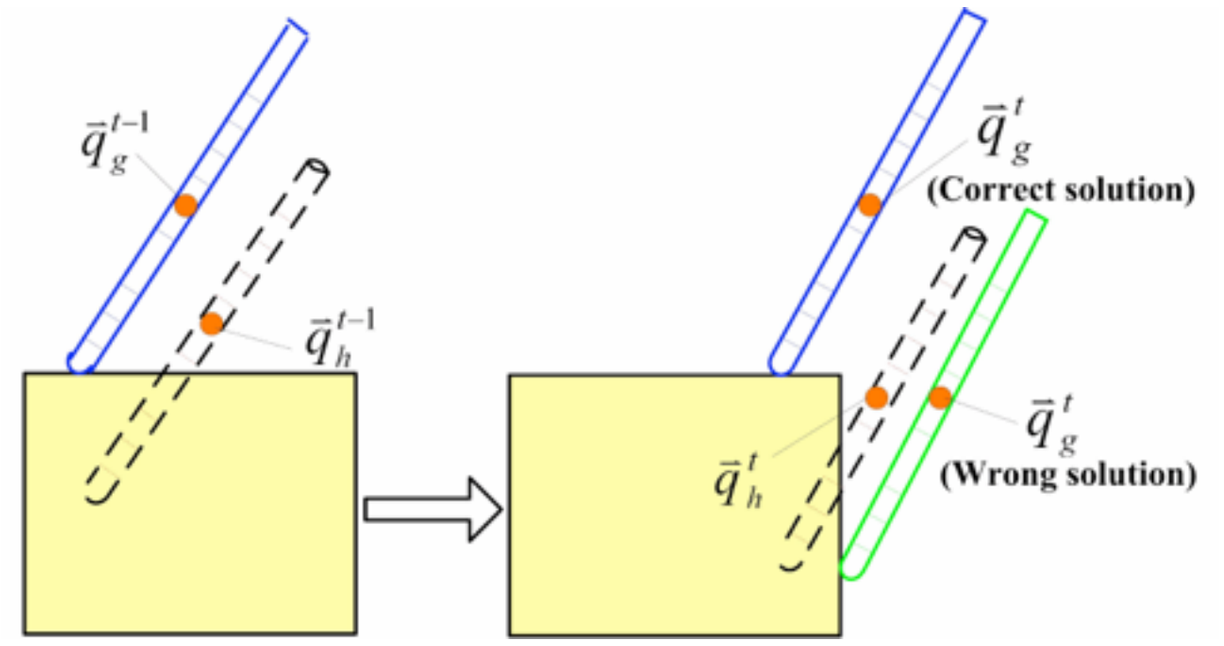
VOXEL SAMPLING

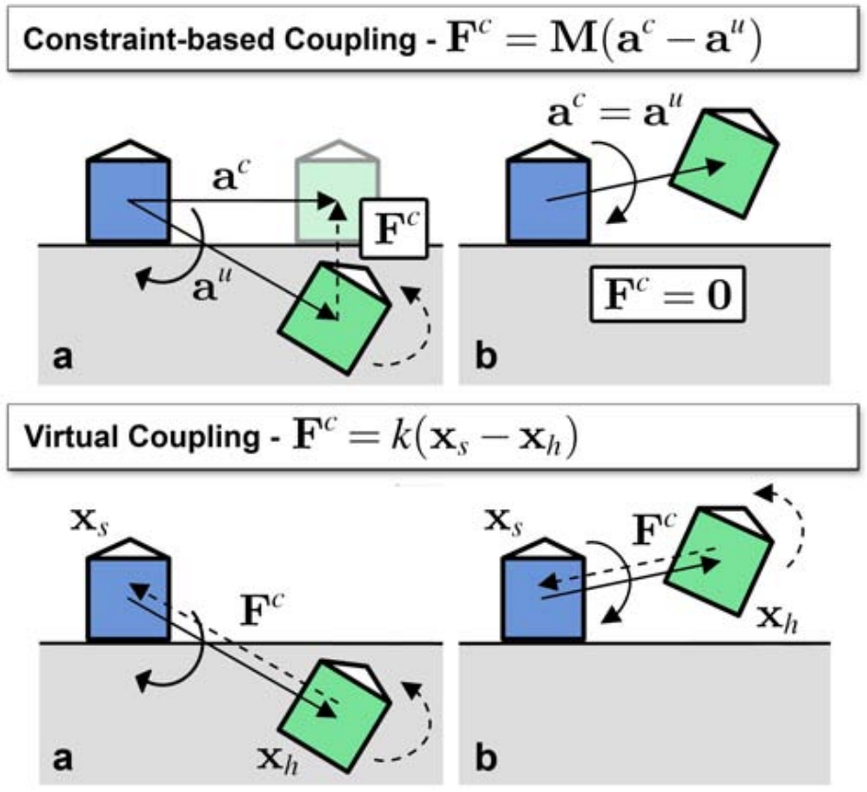
- Penalty based response (penetration)
- Irregular distribution of stiffness
- Pre-contact braking forces. Strong viscous force applied in the proximity layer.
- The viscosity is adjusted so that all the kinetic energy is dissipated in a distance equal to the width of the layer.

6DOF GOD-OBJECT (REDON 2003)

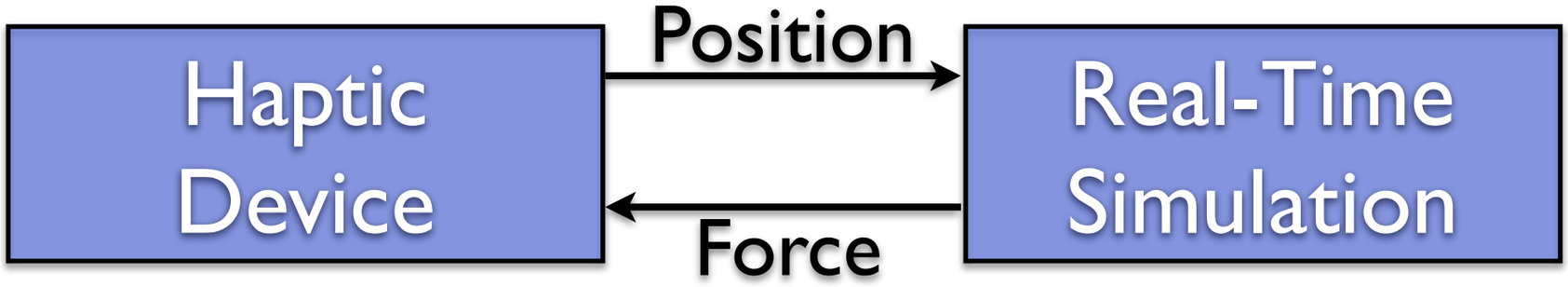
- More complex…





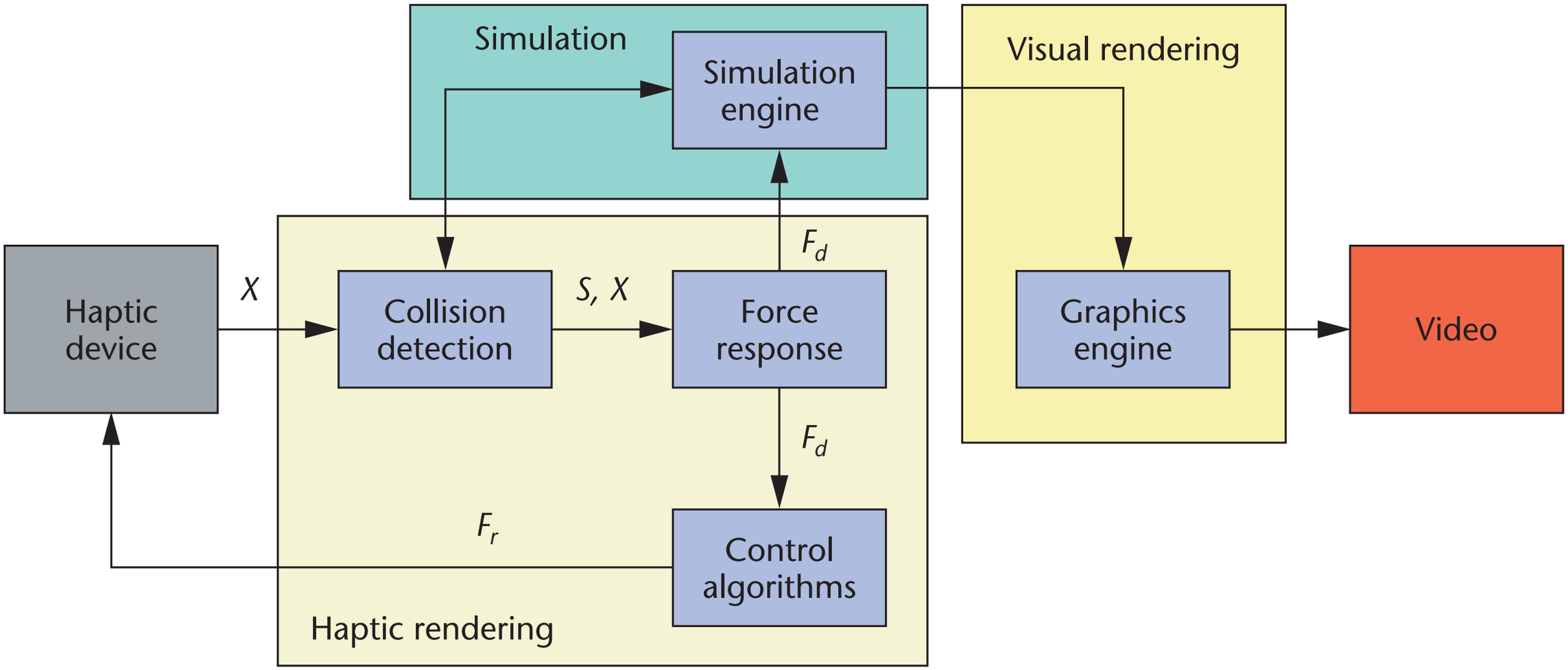


RENDERING BASED ON SIMULATION



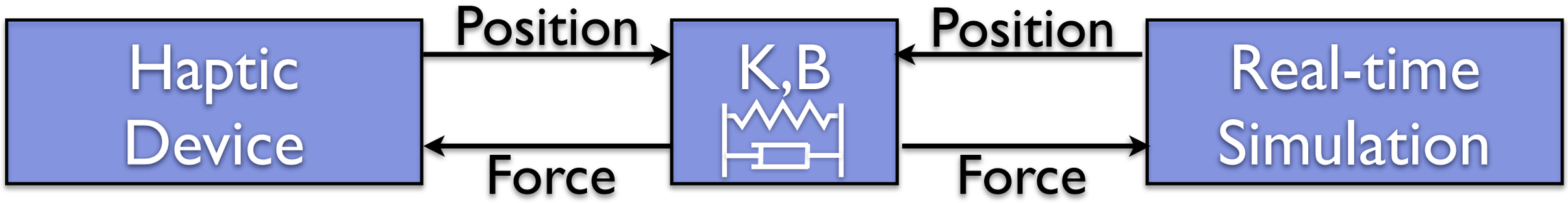
No !

HAPTIC RENDERING BASED ON SIMULATION



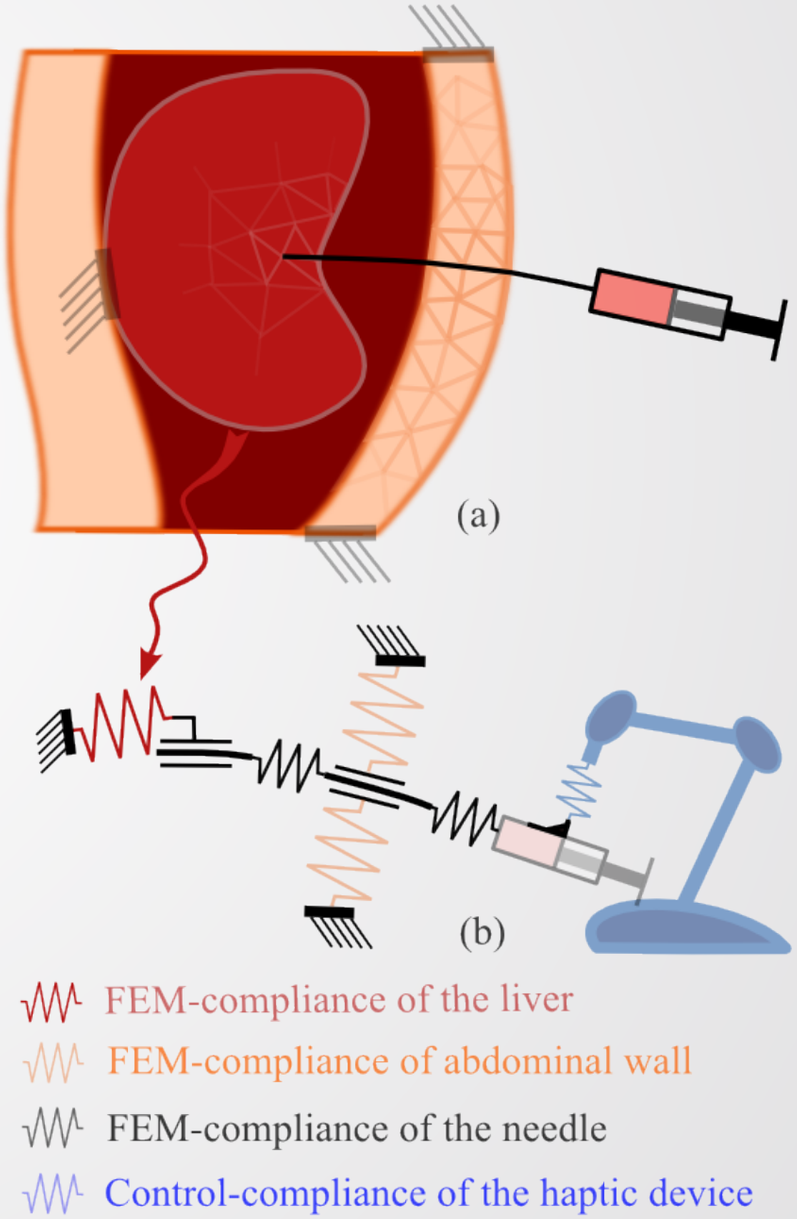
VIRTUAL COUPLING METHOD

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



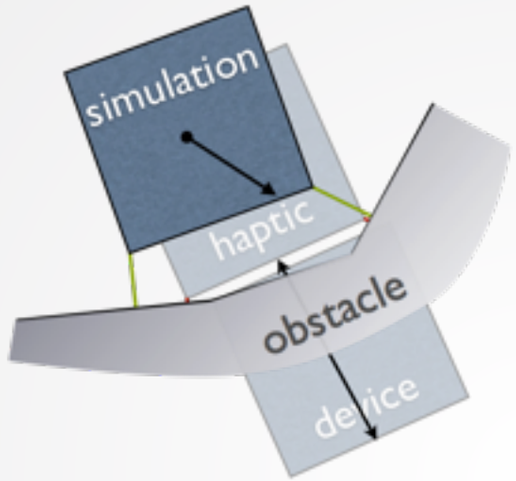
MULTIRATE COMPLIANT MECHANISMS

- 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.

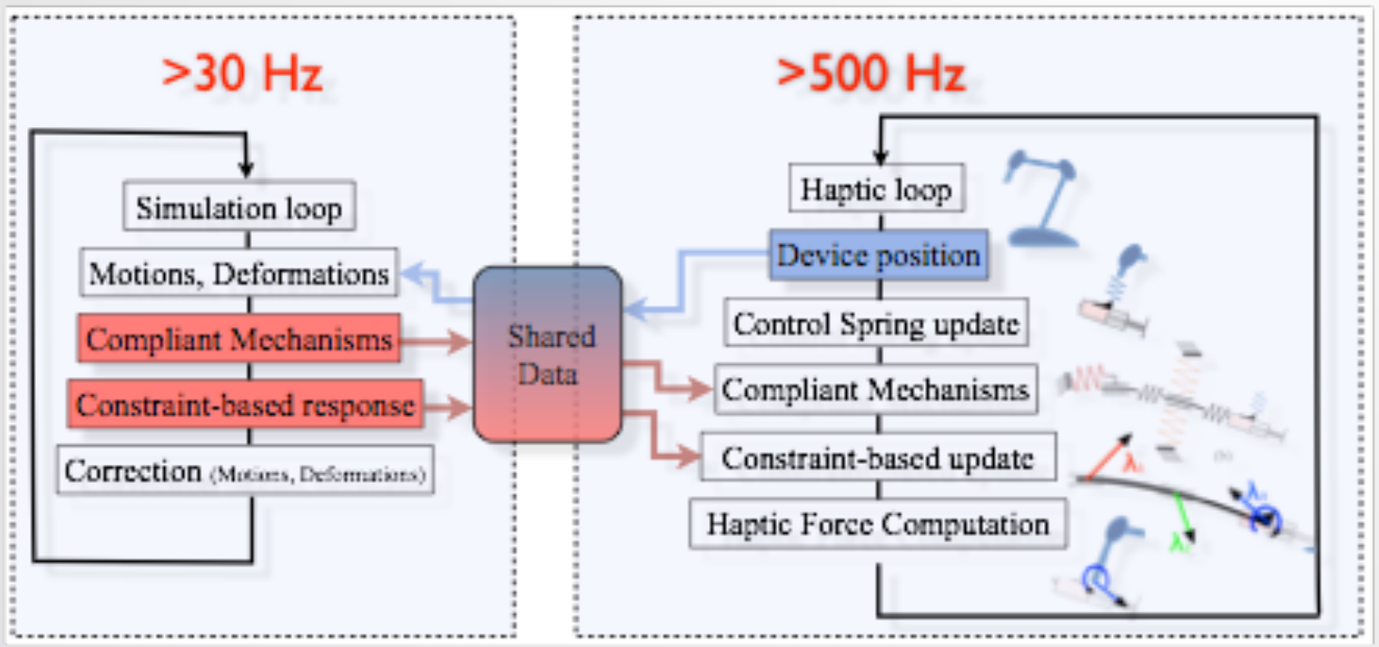


MULTIRATE COMPLIANT MECHANISMS

- 3 ≠ positions

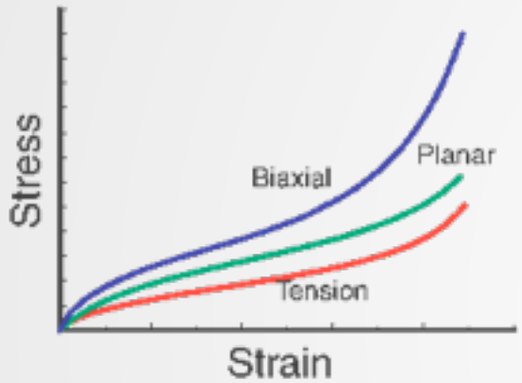


- Shared data to maintain coherency



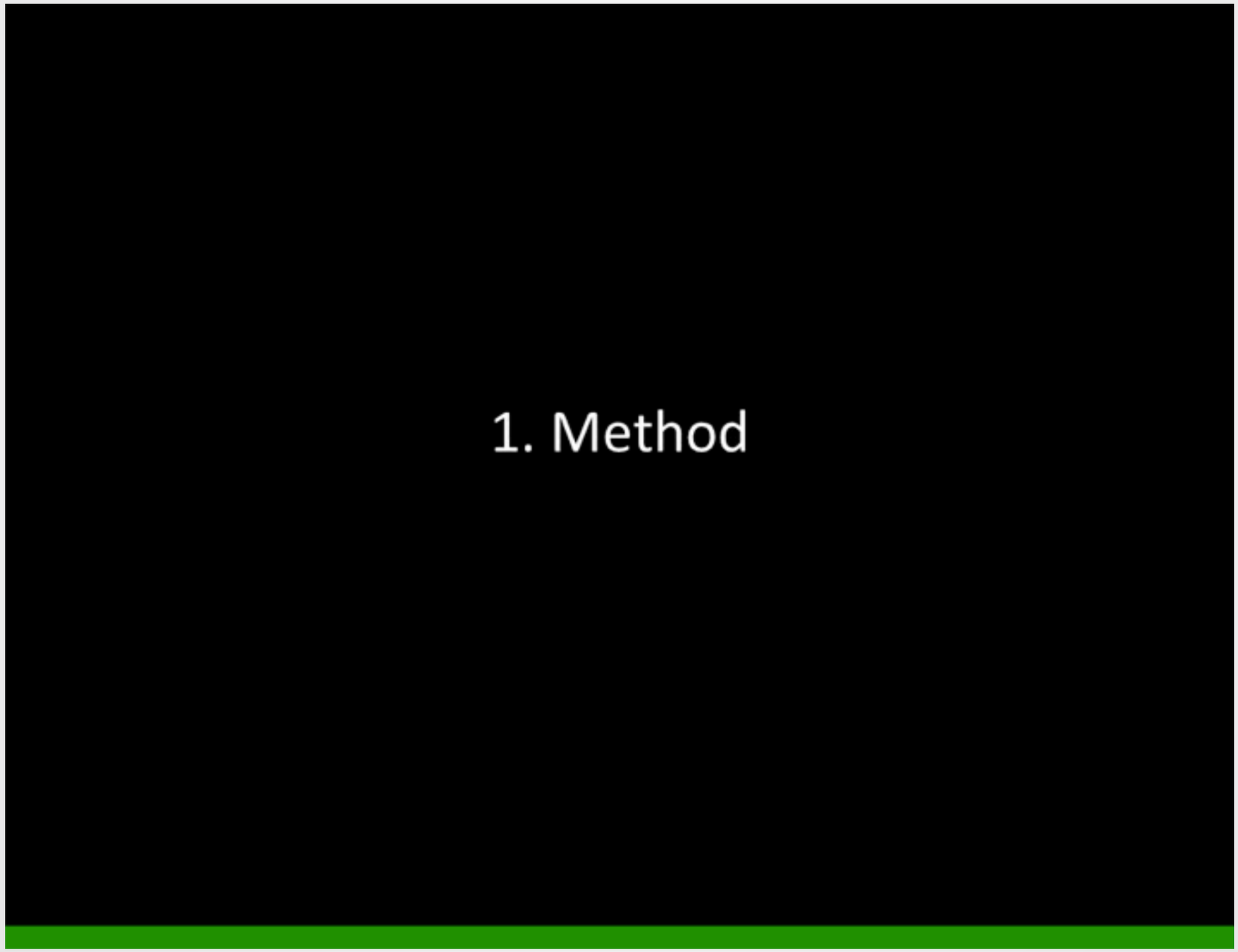
EXAMPLE: HYPERELASTIC MODELS WITH FRICTION

- What is « hyperelastic » ?



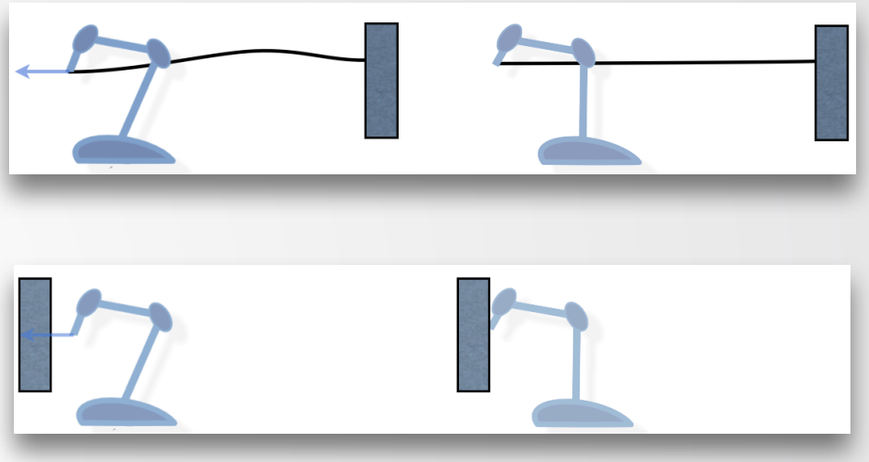
- 3 phenomenons at 3 different rates
  - Velocity of deformations (non-linearities): *low*
  - Velocity of user motions: *medium*
  - Velocity of sensitive (haptic) feedback: *high*

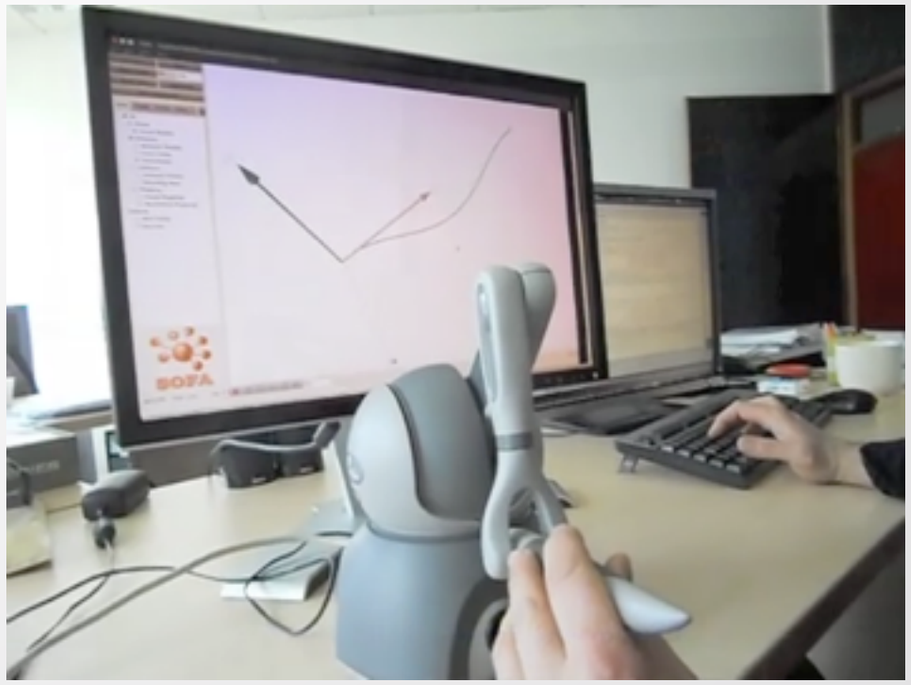
HAPTIC RENDERING OF HYPERELASTIC MODELS WITH FRICTION



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 ?)



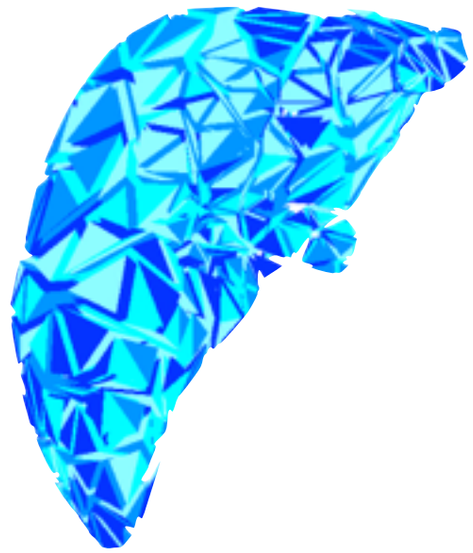


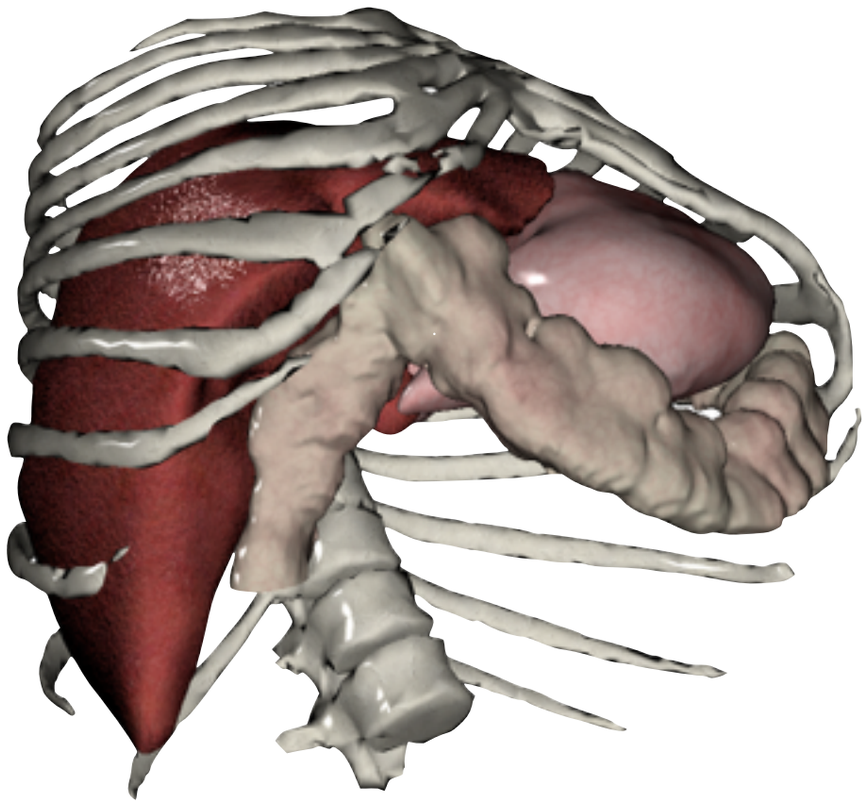
SOFTWARE ARCHITECTURE FOR REAL-TIME SIMULATION

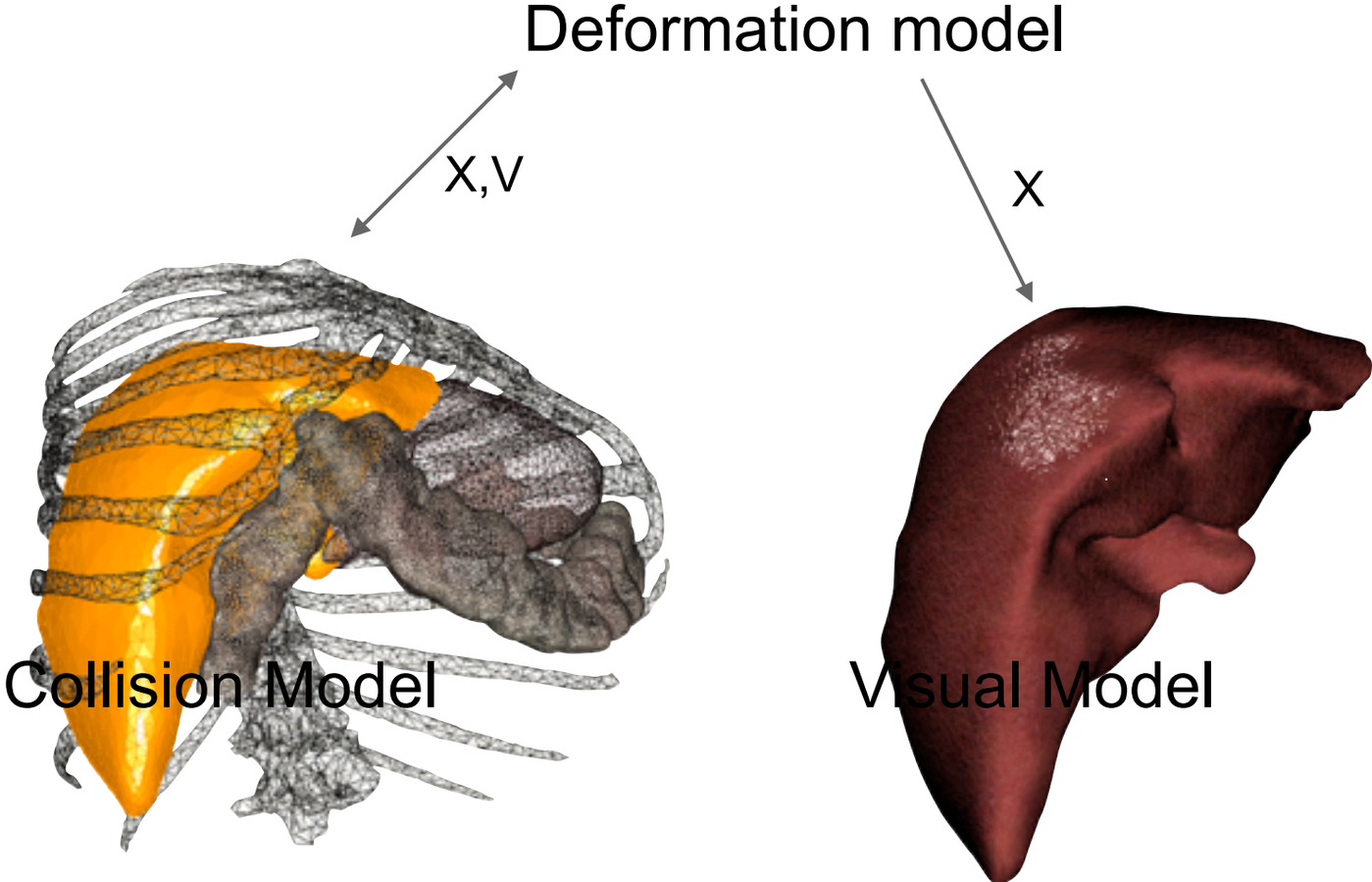
Frank Gehry

EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION

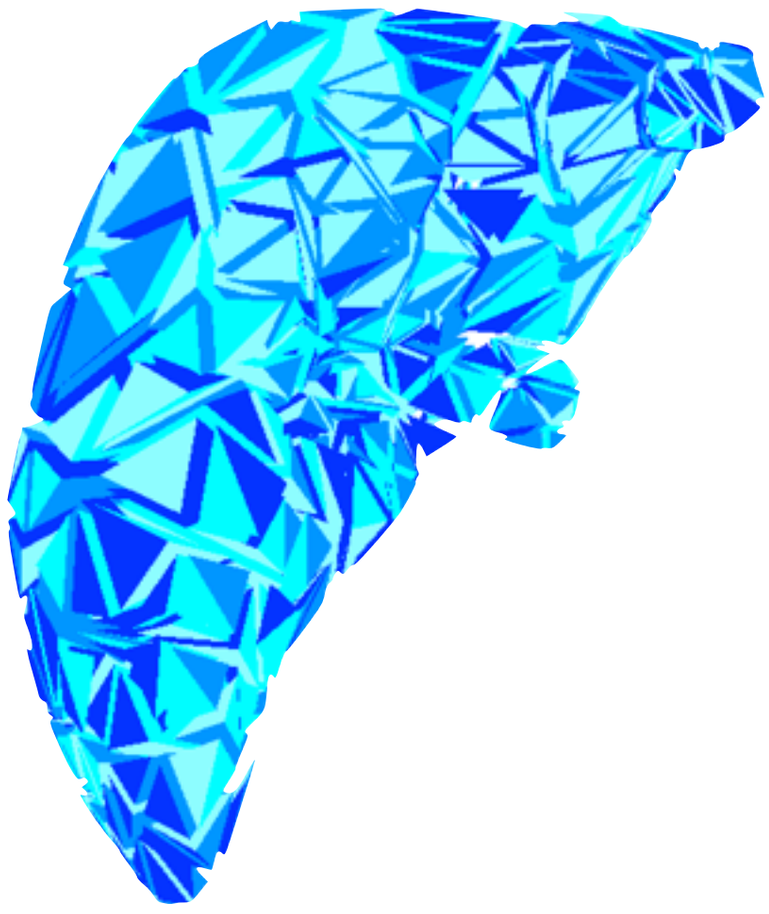
- Split computations into independent parts
  - Separate problems
  - Improve reusability







EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION



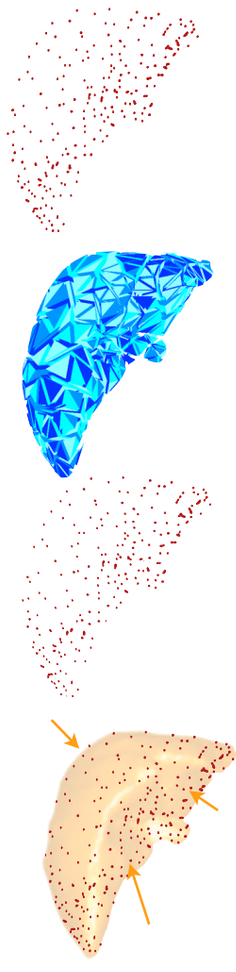
Deformation model

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

ODE Solver

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

Linear Solver



Degrees of Freedom

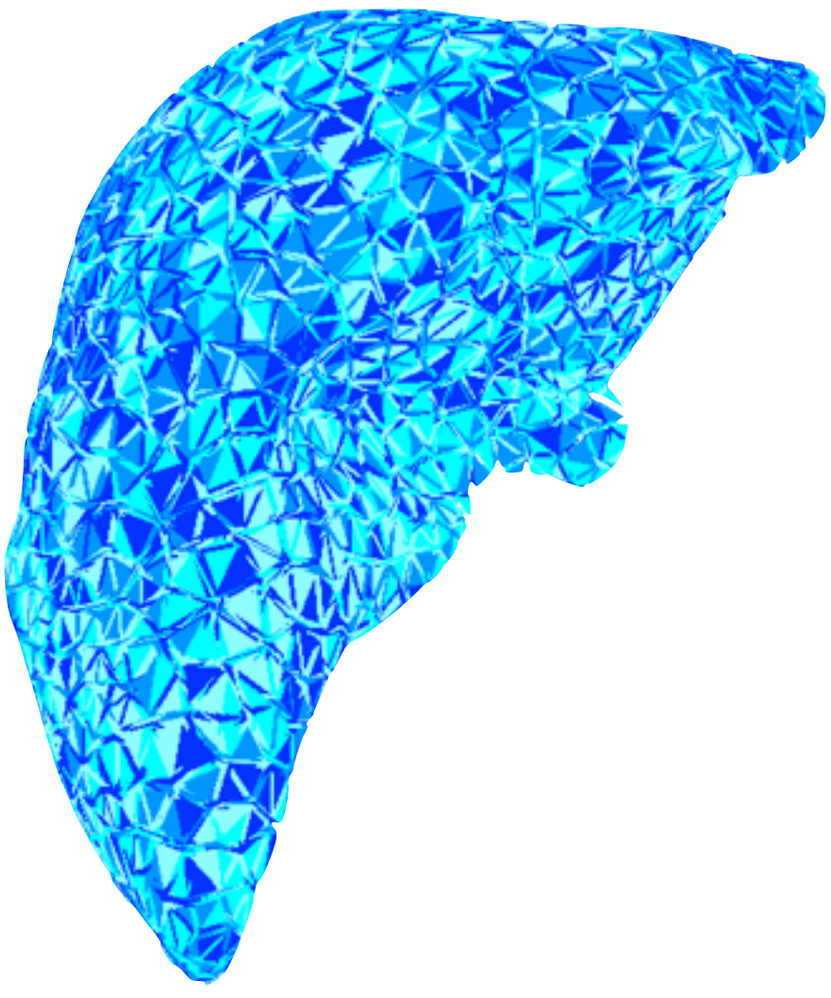
Finite Element Forces and Mass

External Loads

Mapped Forces

...

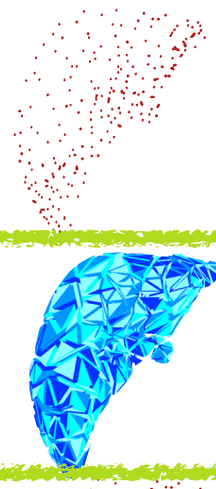
EXAMPLE WITH SOFA: MULTI-MODEL REPRESENTATION



computation on GPU

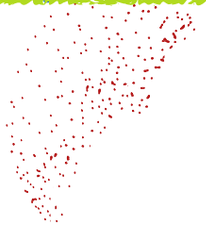
$\mathbb{M}\dot{\mathbf{v}} = \mathbf{p} - \mathbf{f}(\mathbf{v}, \mathbf{x}) + \mathbb{H}^T\lambda$ ODE Solver

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

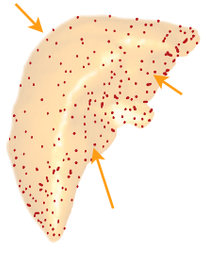


Degrees of Freedom

Finite Element Forces and Mass



External Loads

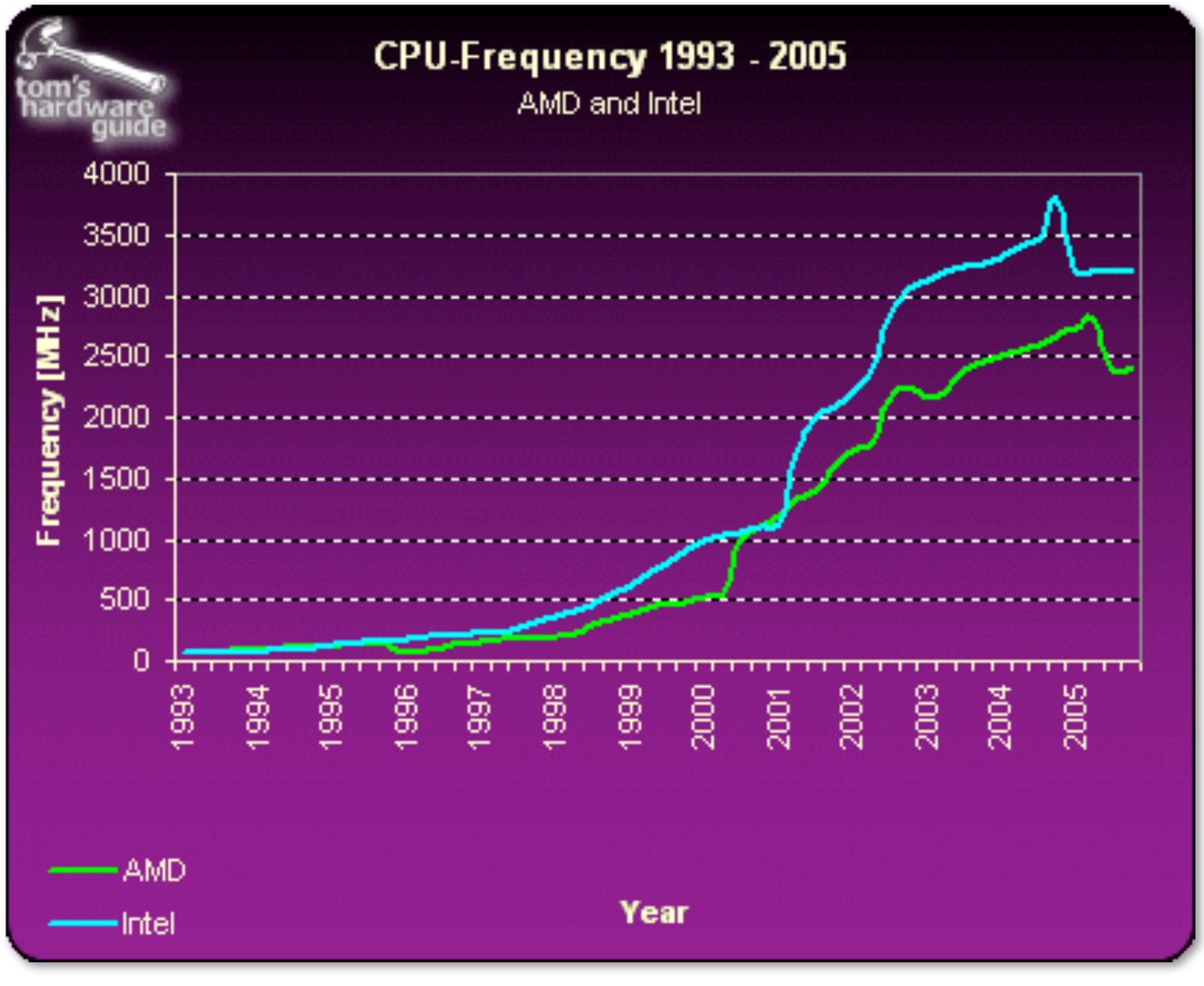


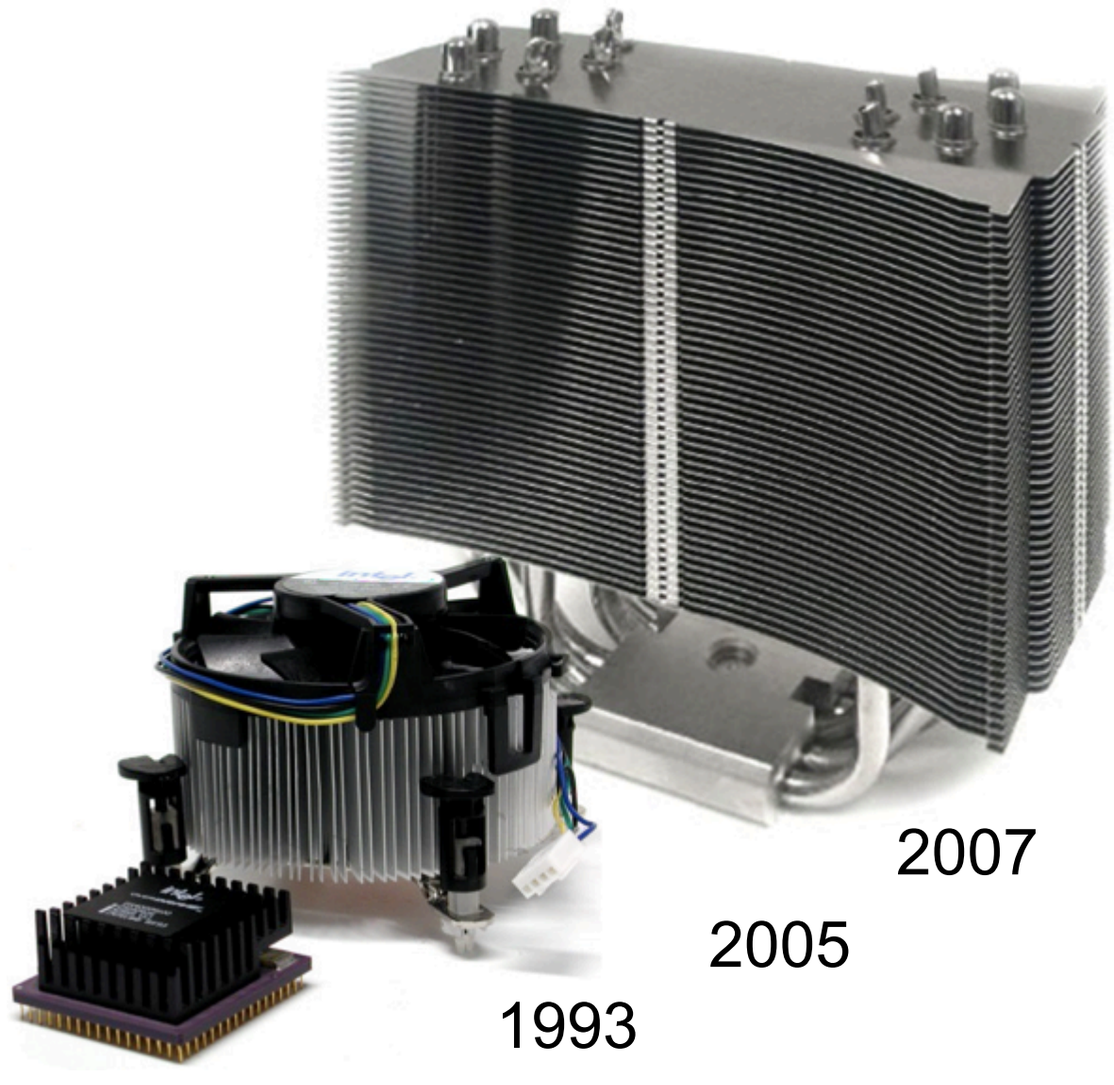
Mapped Forces

...

COMPUTATION TIME OPTIMISATION

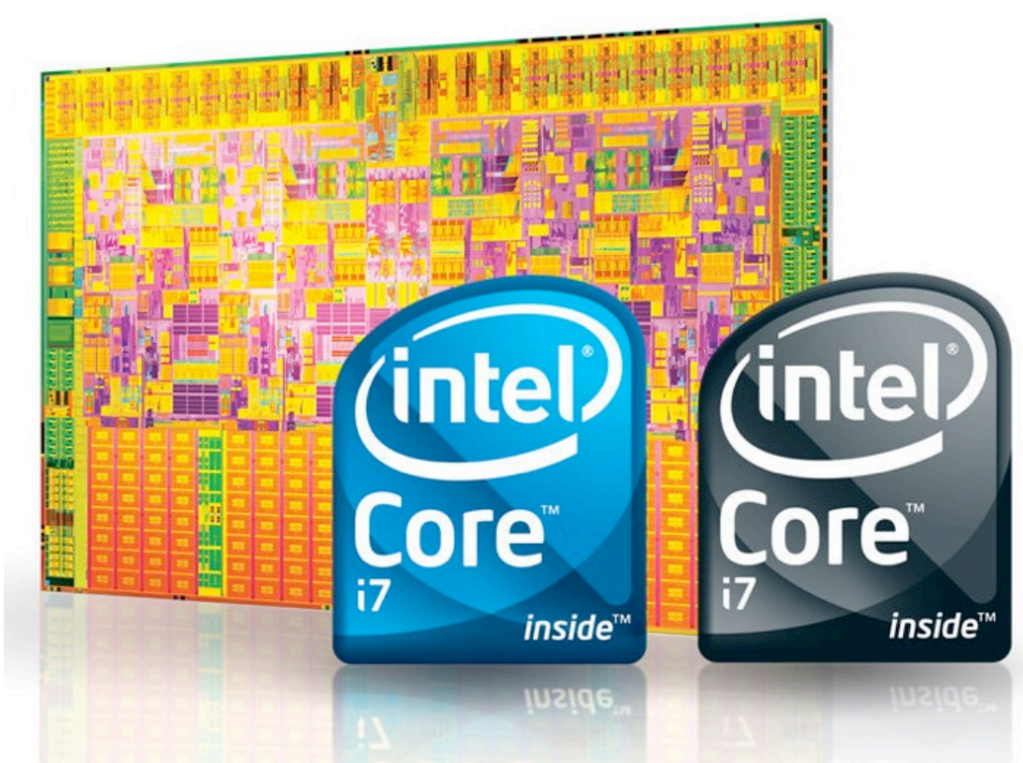
- Why parallel computing ?

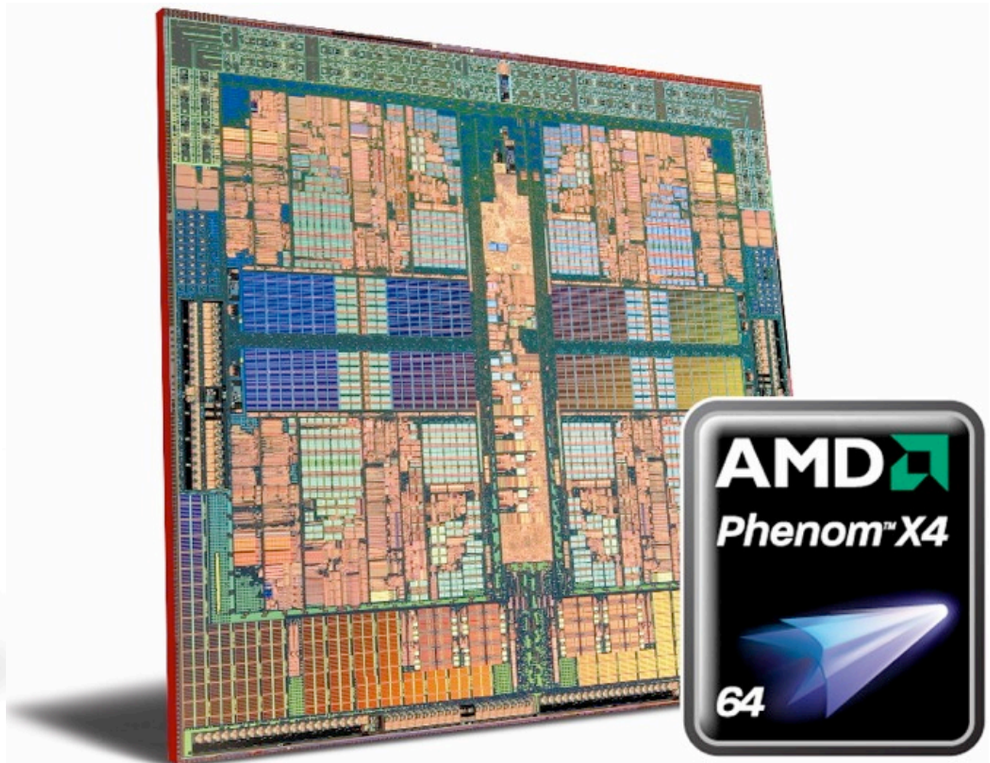




COMPUTATION TIME OPTIMISATION

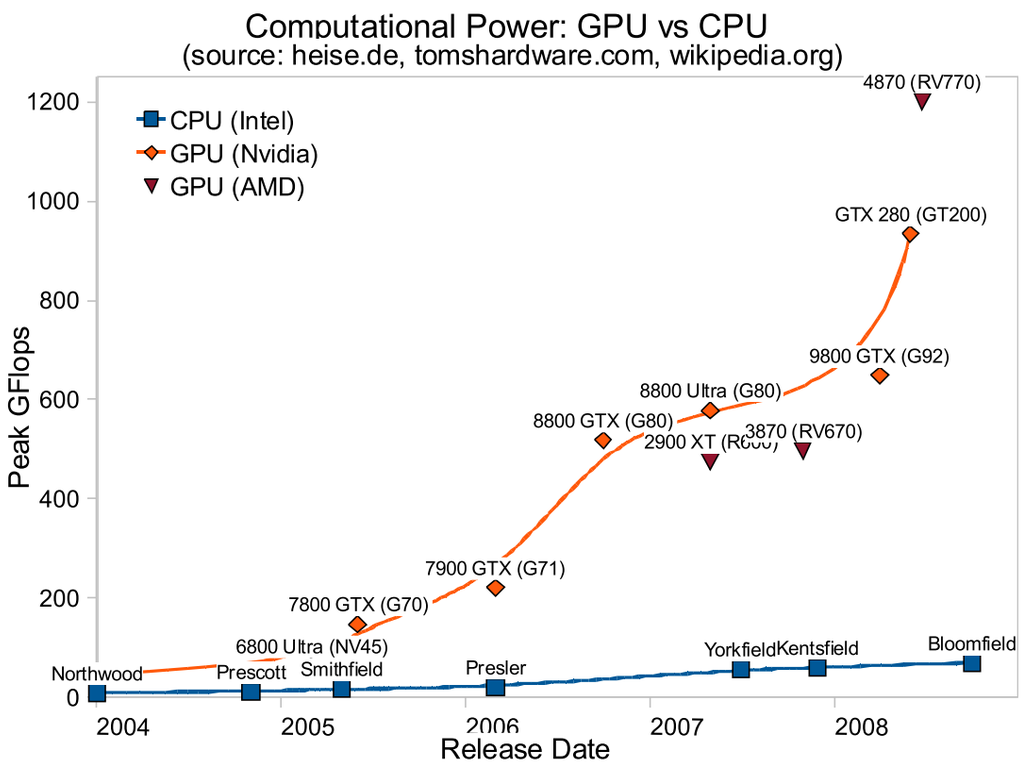
- Classical Parallel Computing
  - Solution : parallelize computations among CPU processing units (2 to 8 cores using 1 or 2 CPUs)
  - Issues : separate tasks, synchronizations, load balancing
    Improve performances, but not dramatically



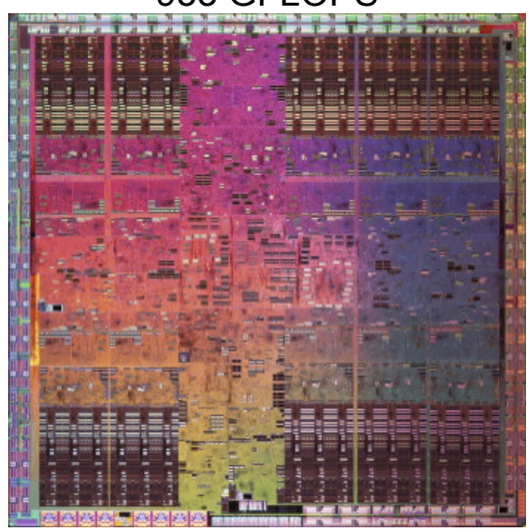


COMPUTATION TIME OPTIMISATION

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

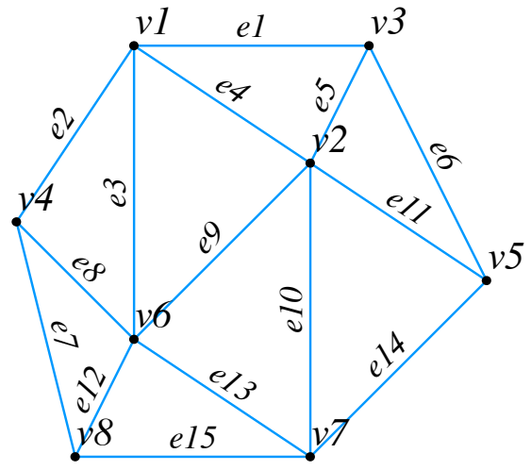


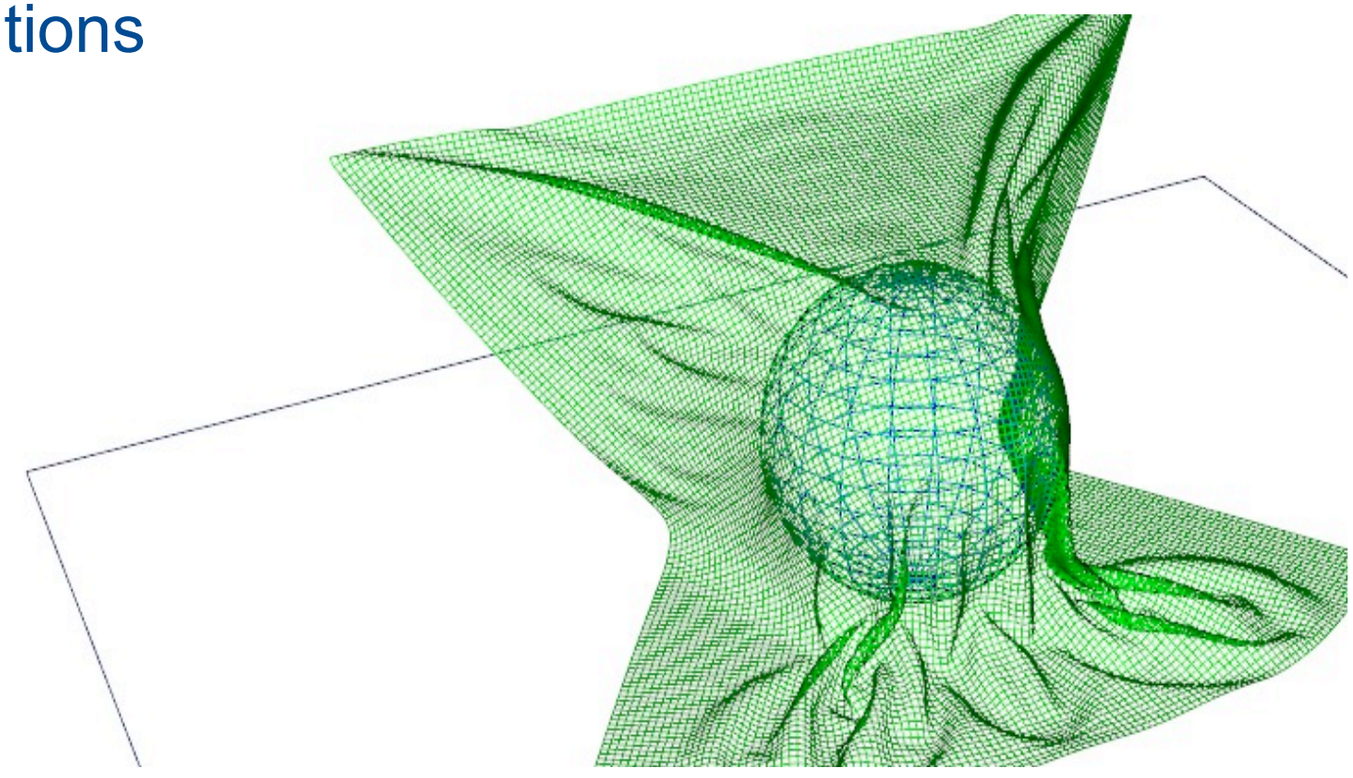
NVIDIA GT 200: 240 units,
933 GFLOPS



Mesh Example

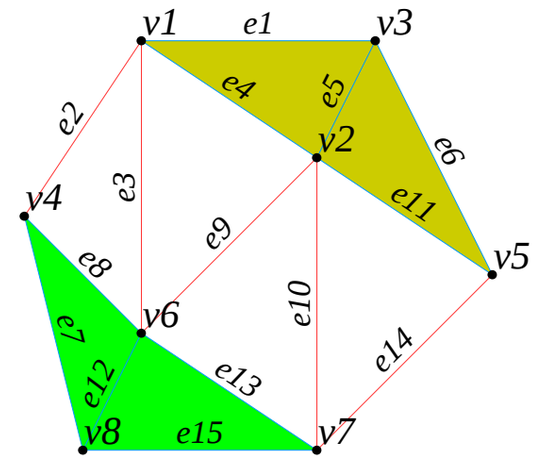
- Let's define a graph with *n* vertices and *m* edges
- We want to accumulate on vertices some values computed on each edge
  - Mass-Spring systems
  - Constraints impulses
  - …
- Problem : several edges connected to the same vertex
- Simple solution : use atomic additions
  - Not available on all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

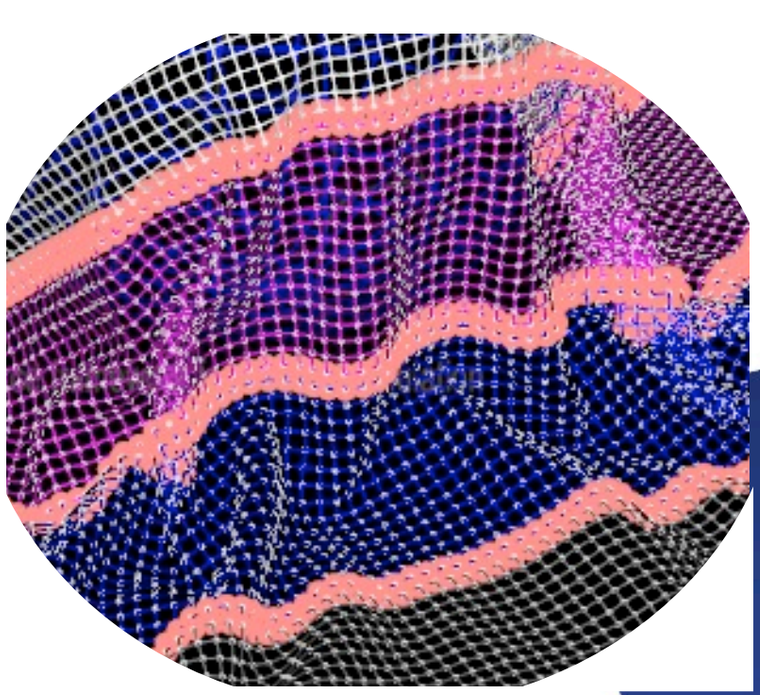


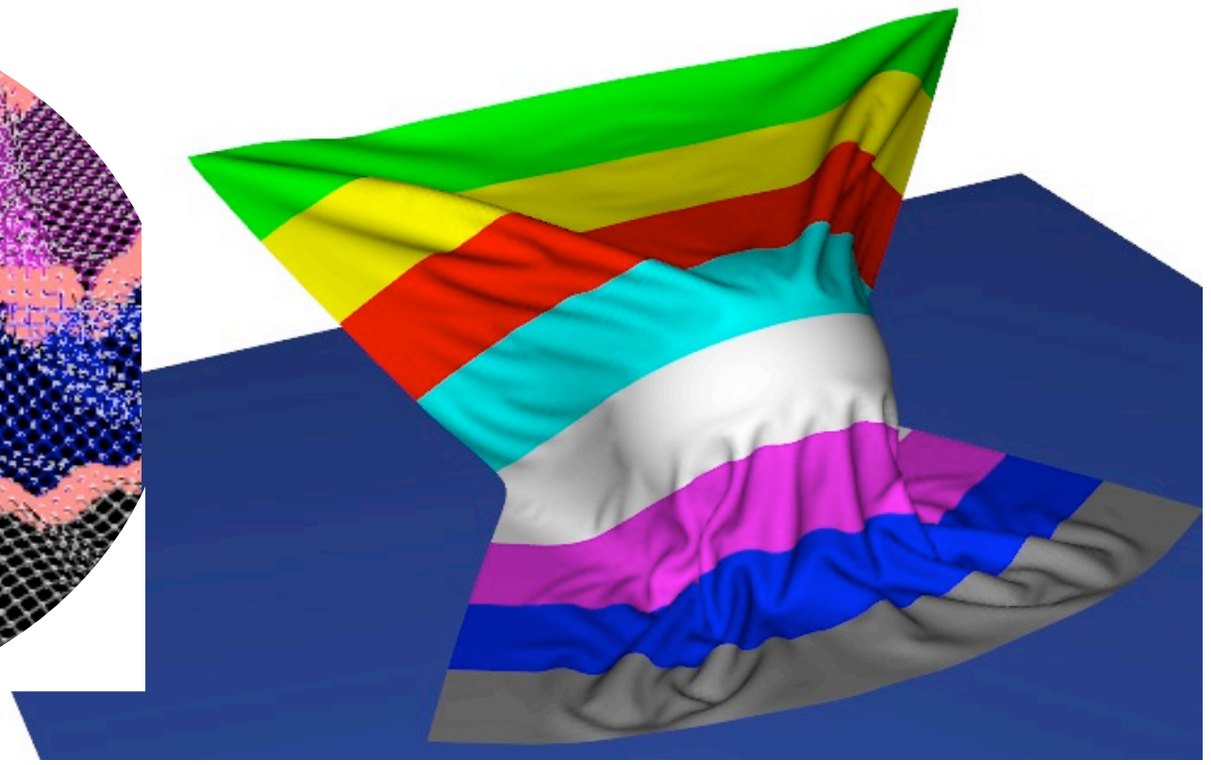


Graph Partitioning

- 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*

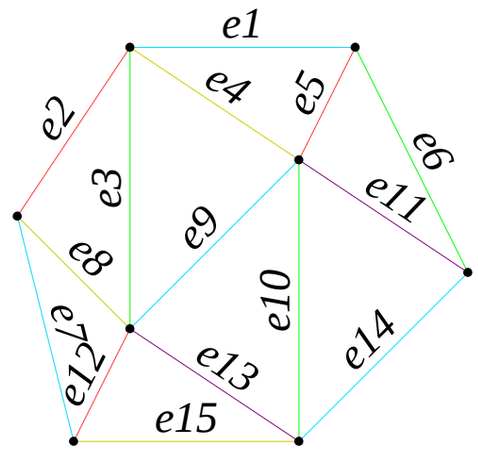


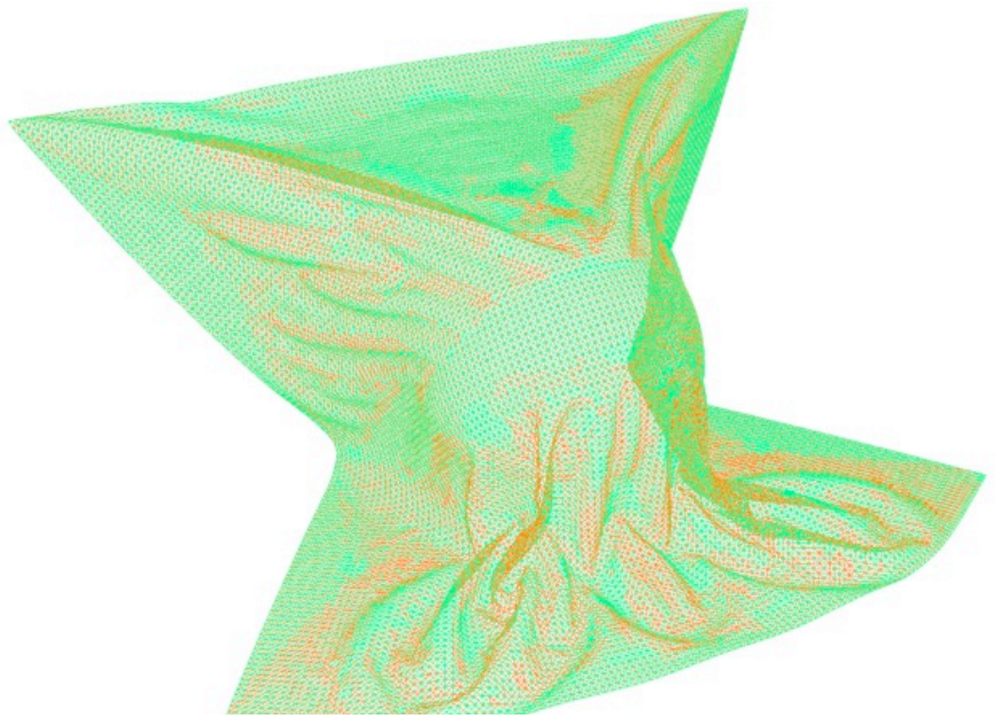




Graph Coloring

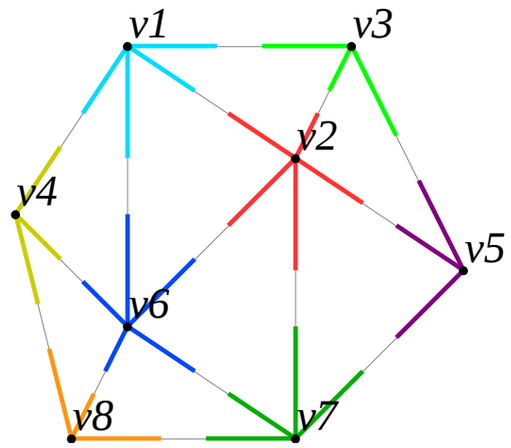
- Alternate parallelization : find *c* sets of unconnected edges
- Edges within a set can be processed in parallel
  - *m* / *c* average parallelism
  - requires *c-1* synchronization barriers
- Problem : c is at least equal to the max arity in the graph
  - 12 in typical mass-spring cloth meshes
- Needs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