Computational design for digital fabrication

Stelian Coros

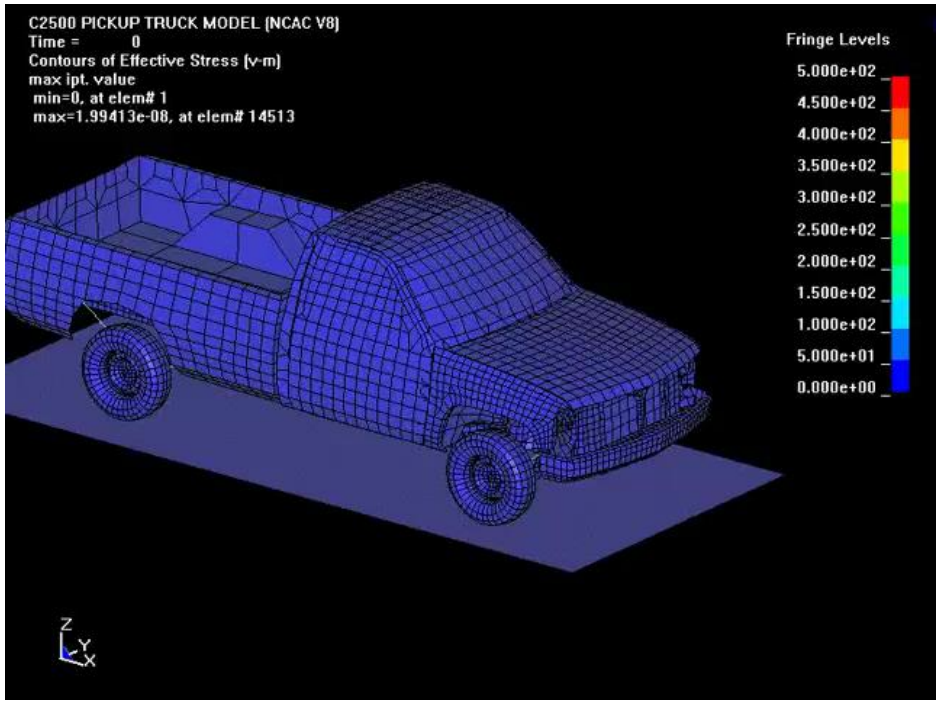
Animation

http://graphics.pixar.com/library/StableElasticity/index.html

Computer-aided design

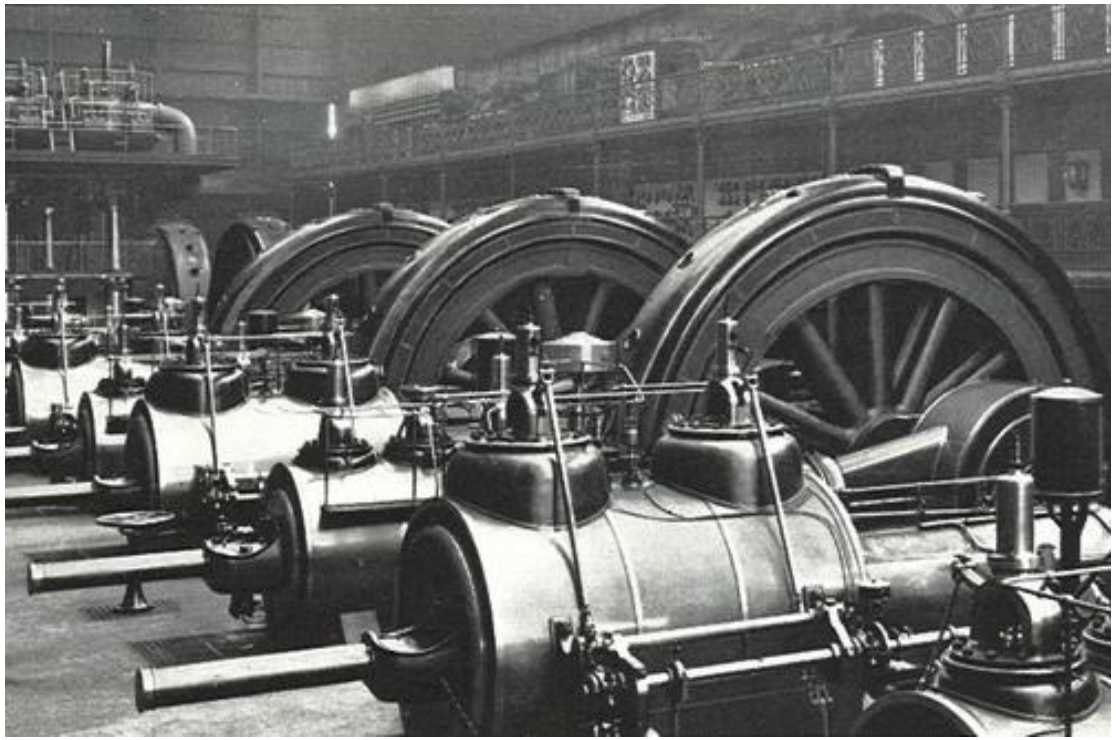
Digital Product Development Cycle

Computer-Aided Design



Simulation (FEM)

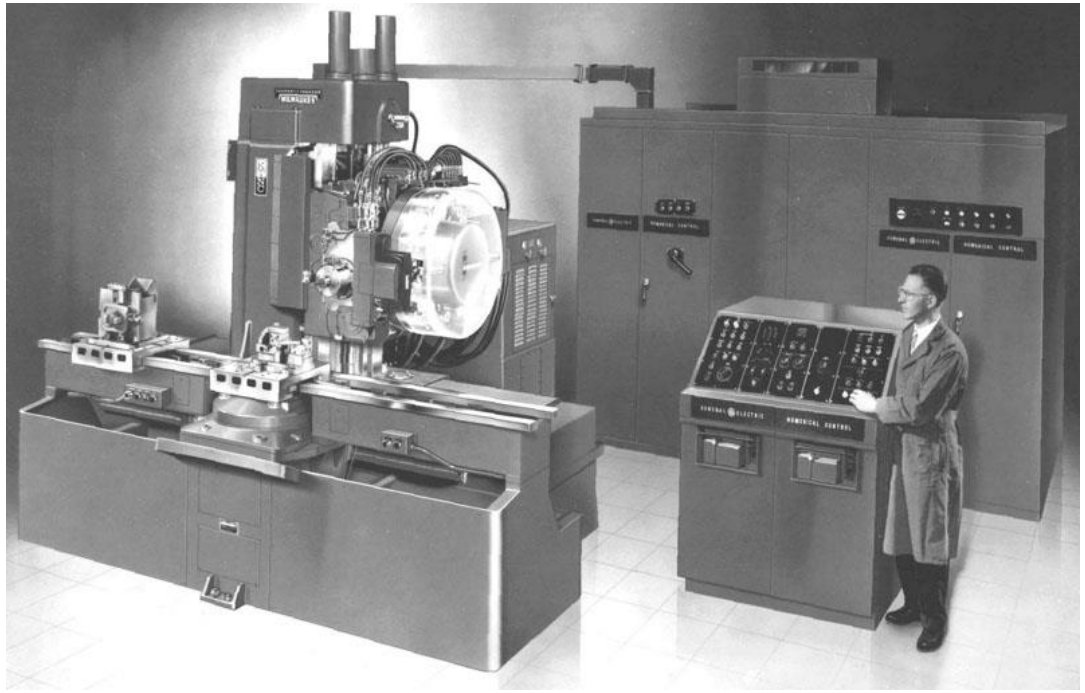
Manufacturing Automation



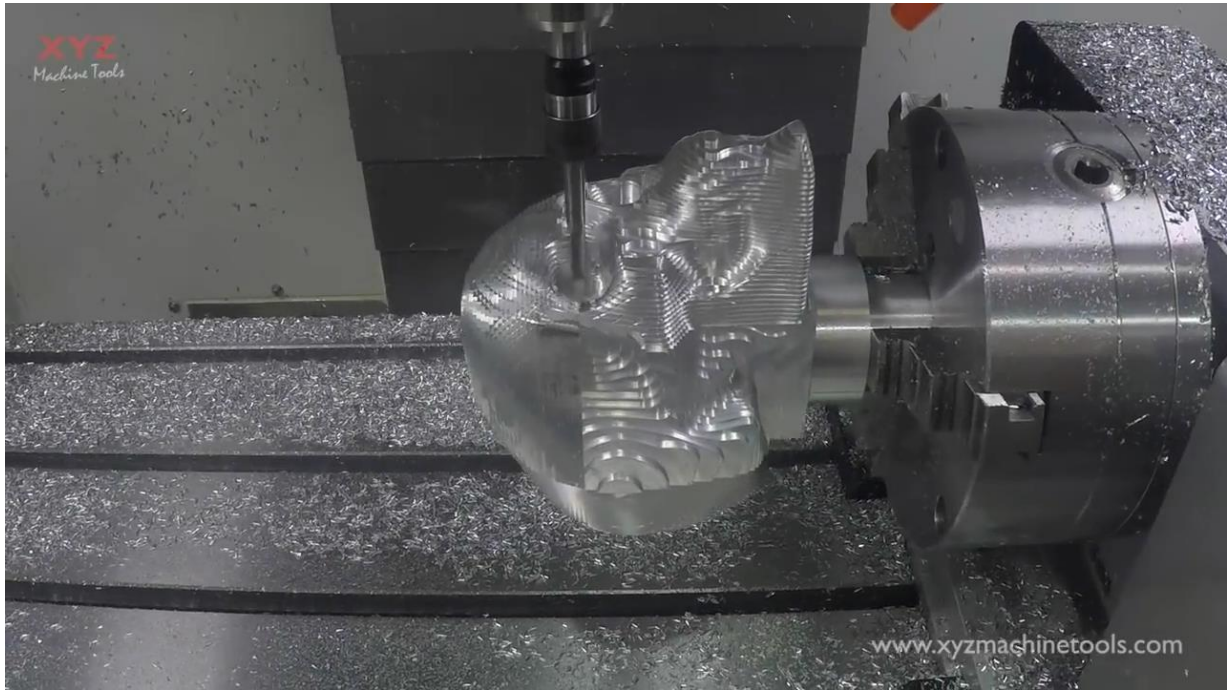
Sulzer refrigerating machines (steam engines), around 1880.

Model T car in Ford's assembly line, around 1913.

CNC – Computer Numerical Control



Milwaukee-Matic-II CNC Machine (1959)



XYZ 1060 VMC (5-axis milling machine)

Fabrication aware design!

3D Printing

Good morning America, 1989.

Additive Manufacturing

- Stereolithography
- Fused Deposition Modeling
- Selective Laser Sintering
- …

3D Printing Principle

3DPrintingIndustry.com

Opportunities: Geometric Complexity

Autodesk

Opportunities: Material Complexity

Applications

Current and future use cases for 3D printing in industry

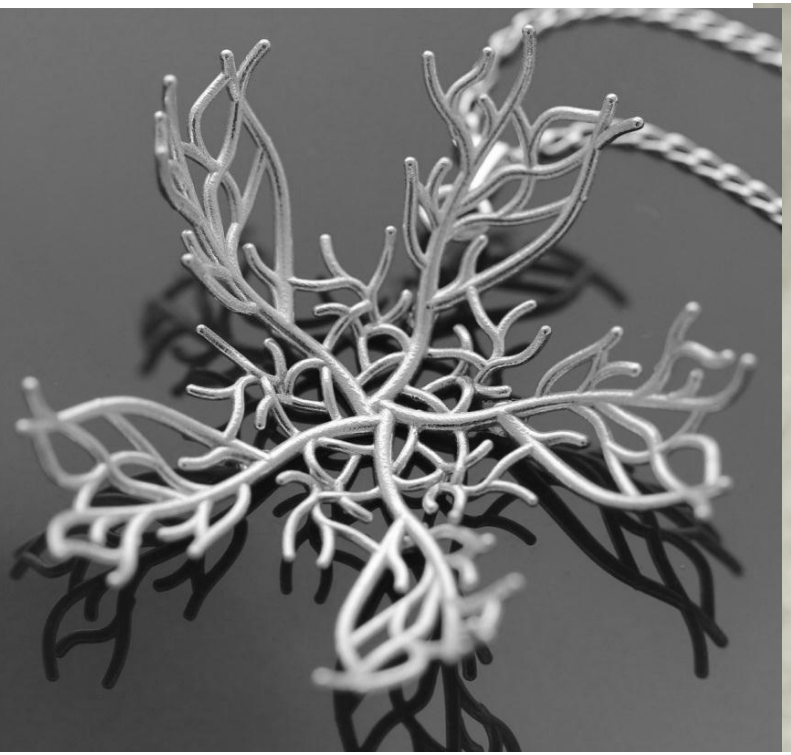
Aerospace & Automotive

Airbus wing bracket

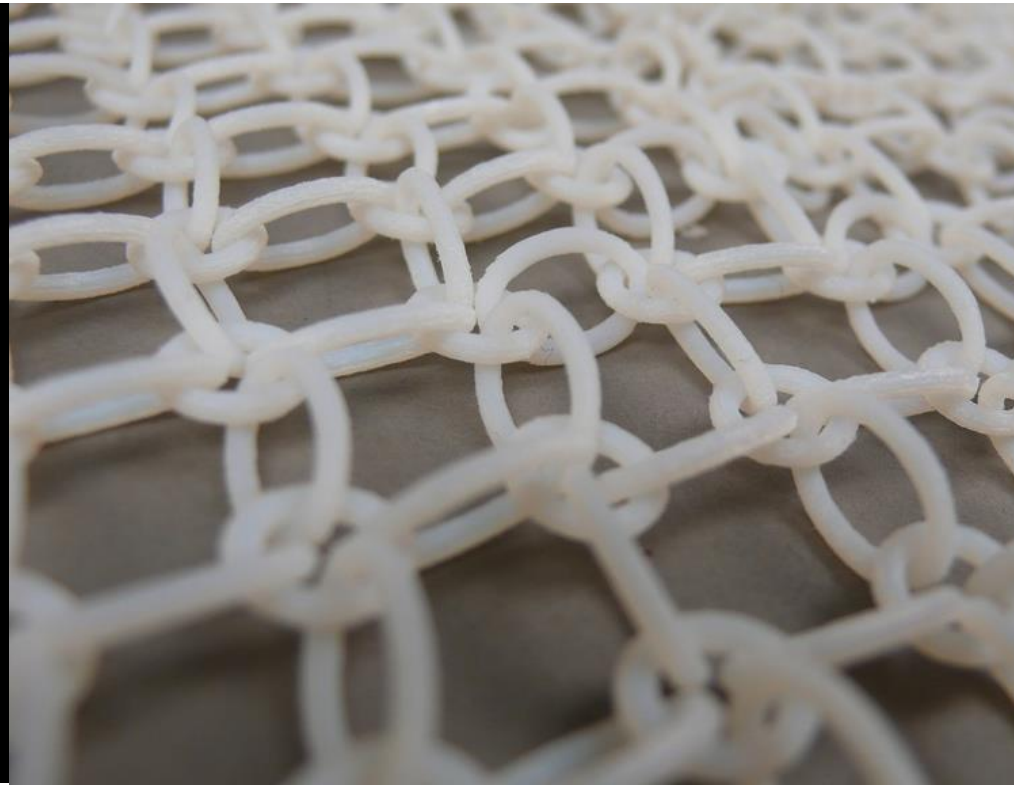
General Electrics nozzle

Medical

Personalized Consumer Goods

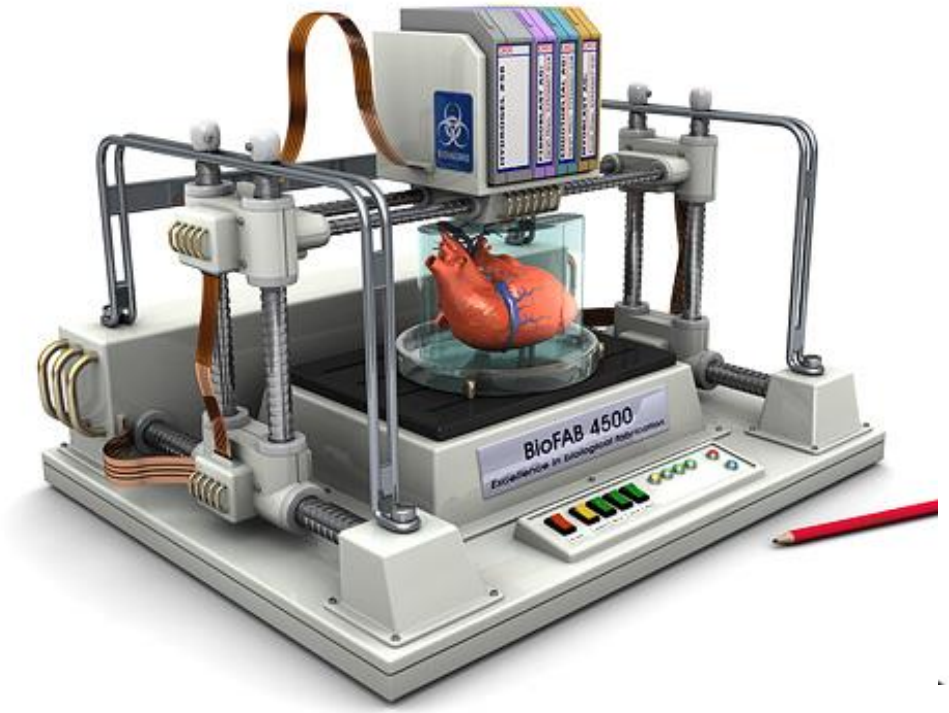


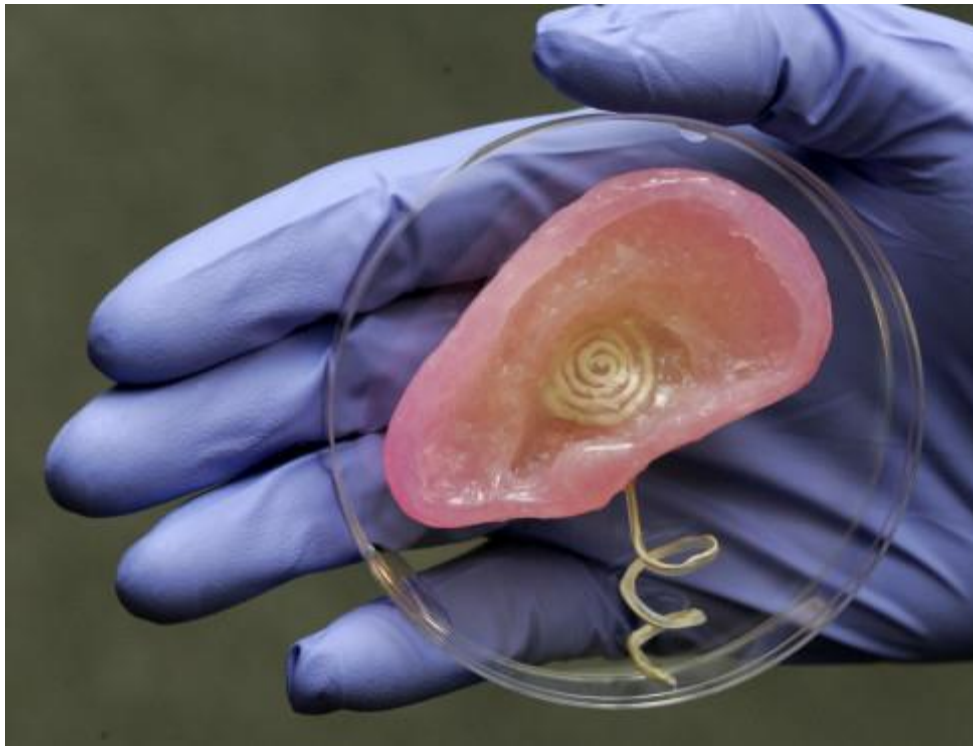
Fashion



Multifunctional Objects

Bioprinting

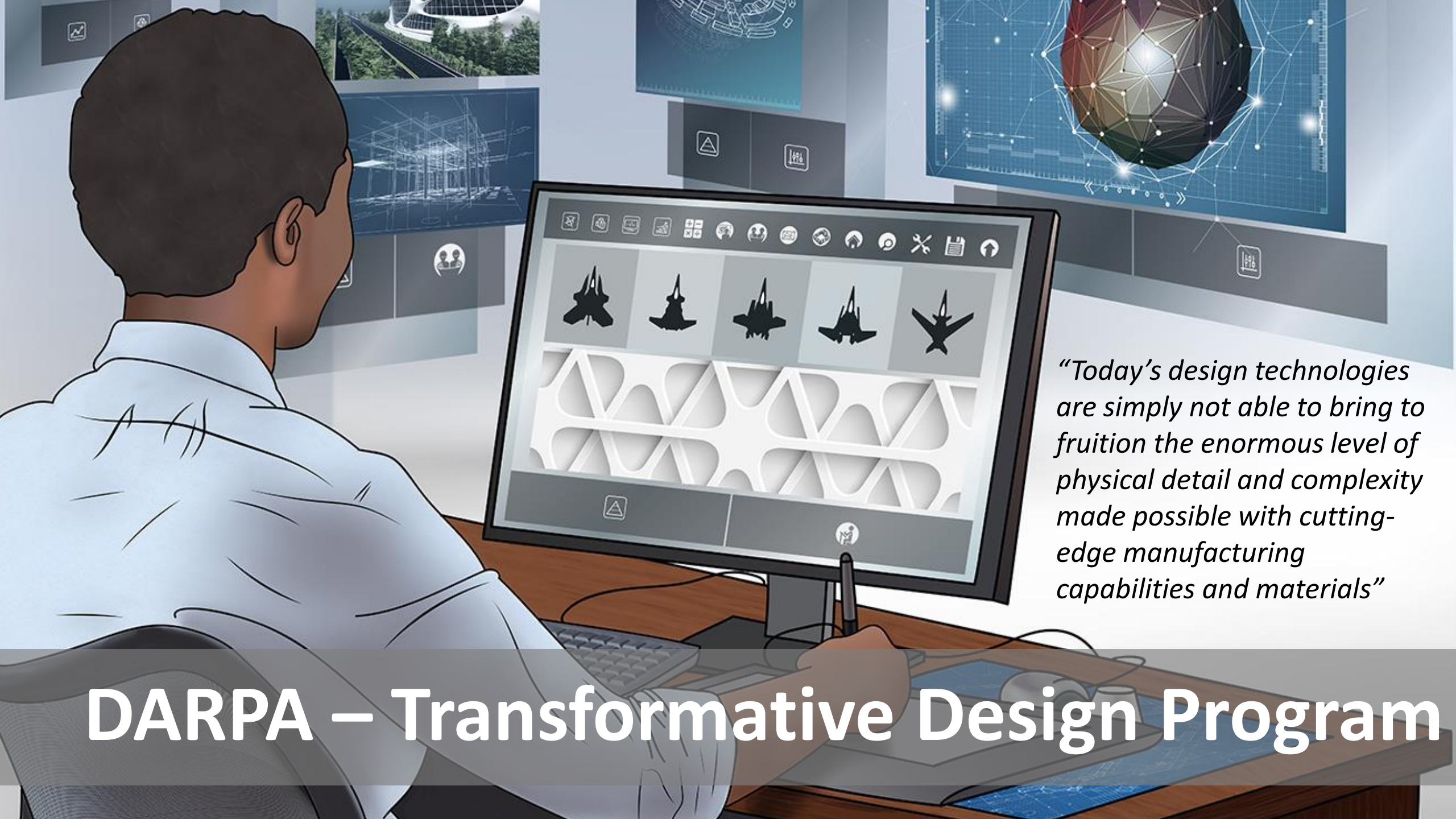




Architecture

Yesterday's Software Fails Today's Hardware

Carl Bass, CEO Autodesk

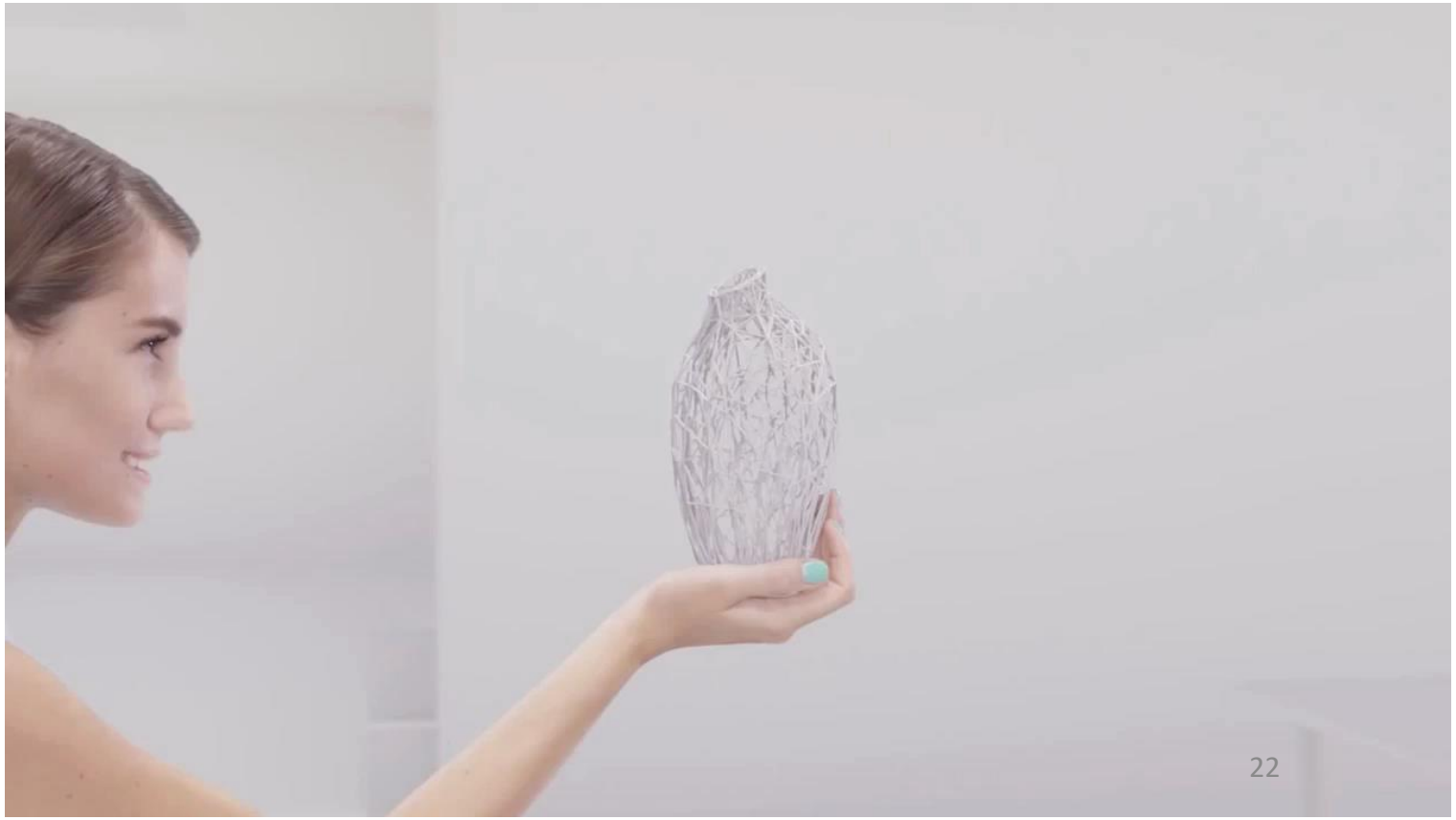


“Today’s design technologies are simply not able to bring to fruition the enormous level of physical detail and complexity made possible with cutting-edge manufacturing capabilities and materials”

DARPA – Transformative Design Program

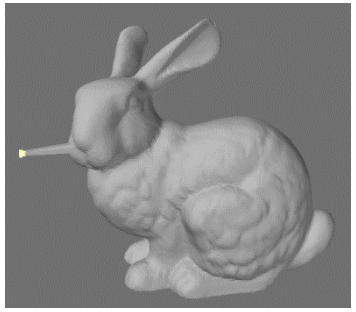
Computational Design

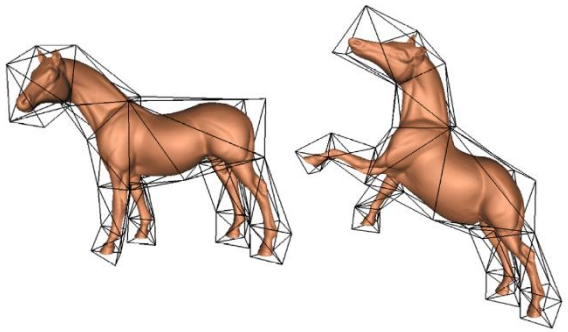
- Need a parameterized geometric model (examples?)
- Some way of measuring the “quality” of the design
- CAD systems typically expose design parameters to the user

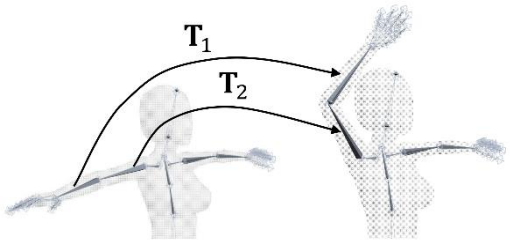


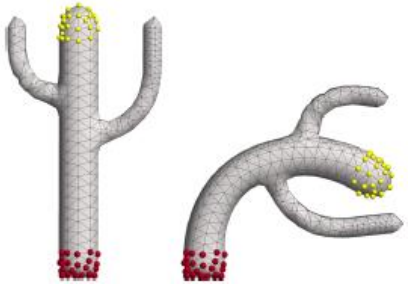
Computational Design

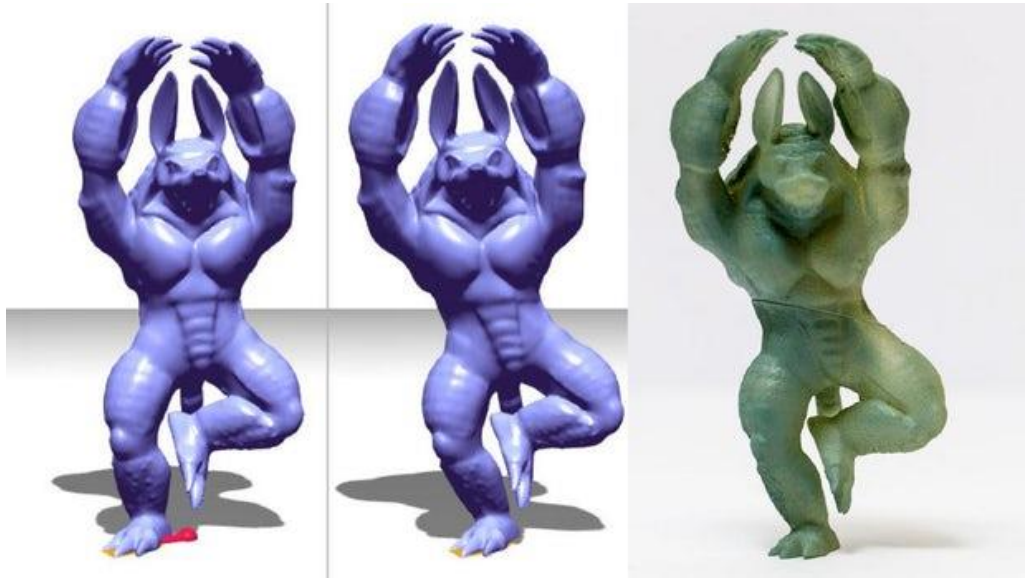
- Forward design: direct manipulation of design parameters
  - Level of abstraction matters











Make It Stand: Balancing Shapes for 3D Fabrication, Prevost et al. 2013

Forward design

- Finding the right level of abstraction is key
  - Restrict design space to some extent
  - Trade-off between flexibility and ease of use
- Design interfaces matter

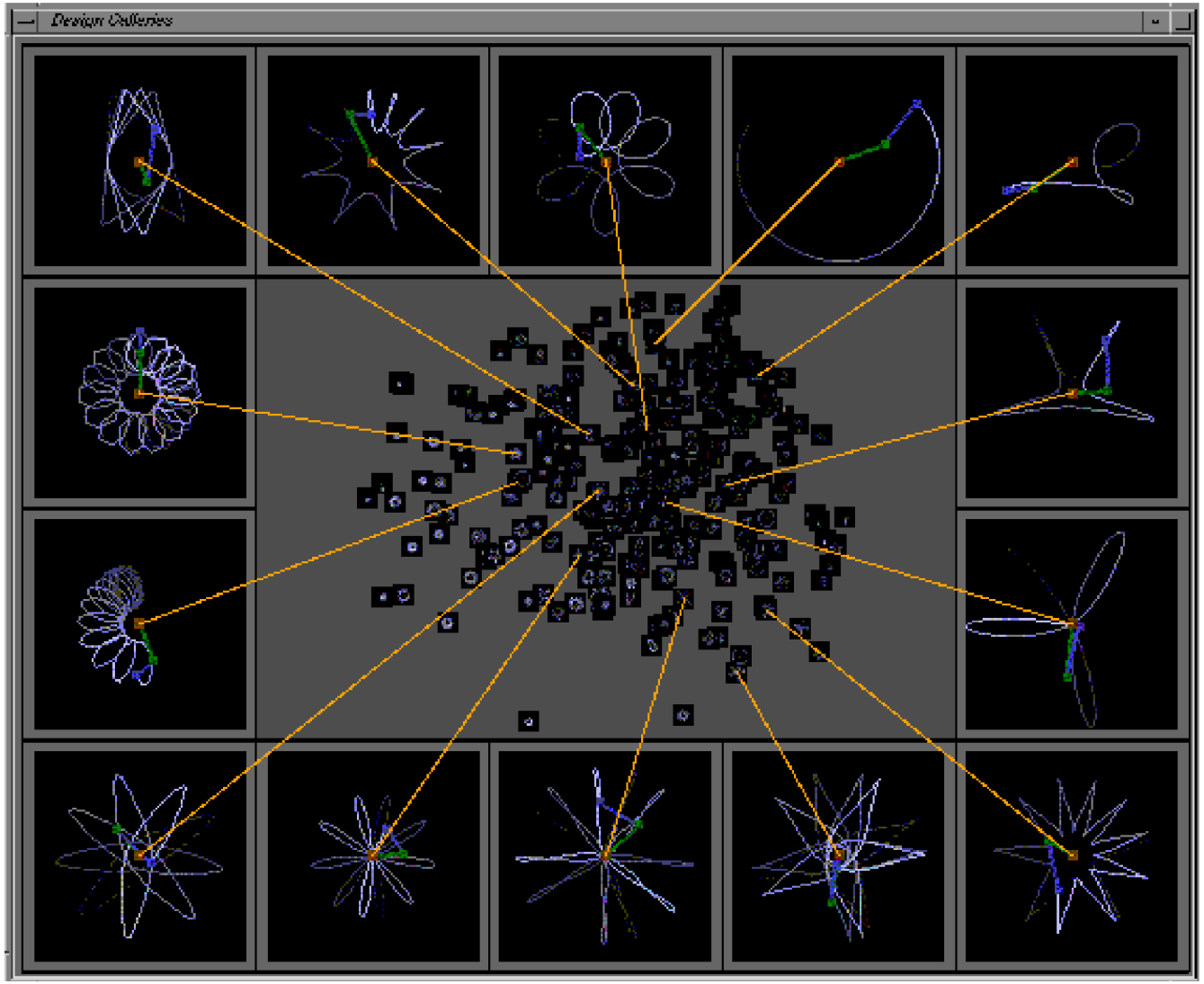
Design Interfaces

http://www.geocities.jp/igarashi_lab/plushie/index-e.html

Forward design

- Finding the right level of abstraction is key
  - Restrict design space to some extent
  - Trade-off between flexibility and ease of use
- Design interfaces matter
- Alternatives to hand-specified parameters?
  - System-guided design space exploration

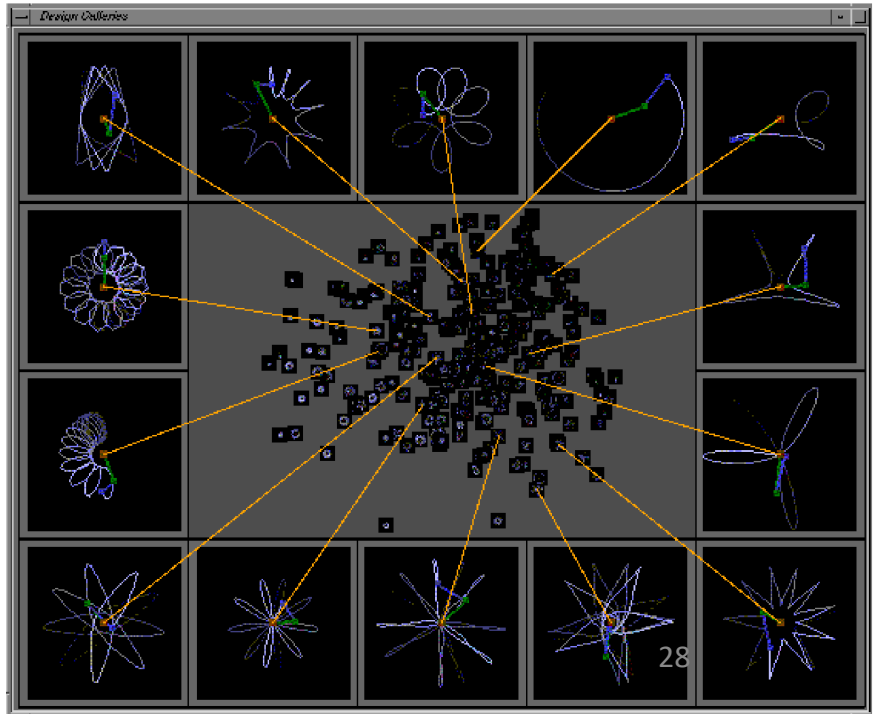
Design Space Exploration



Design galleries: a general approach to setting parameters for computer graphics and animation, Marks et al., 1997

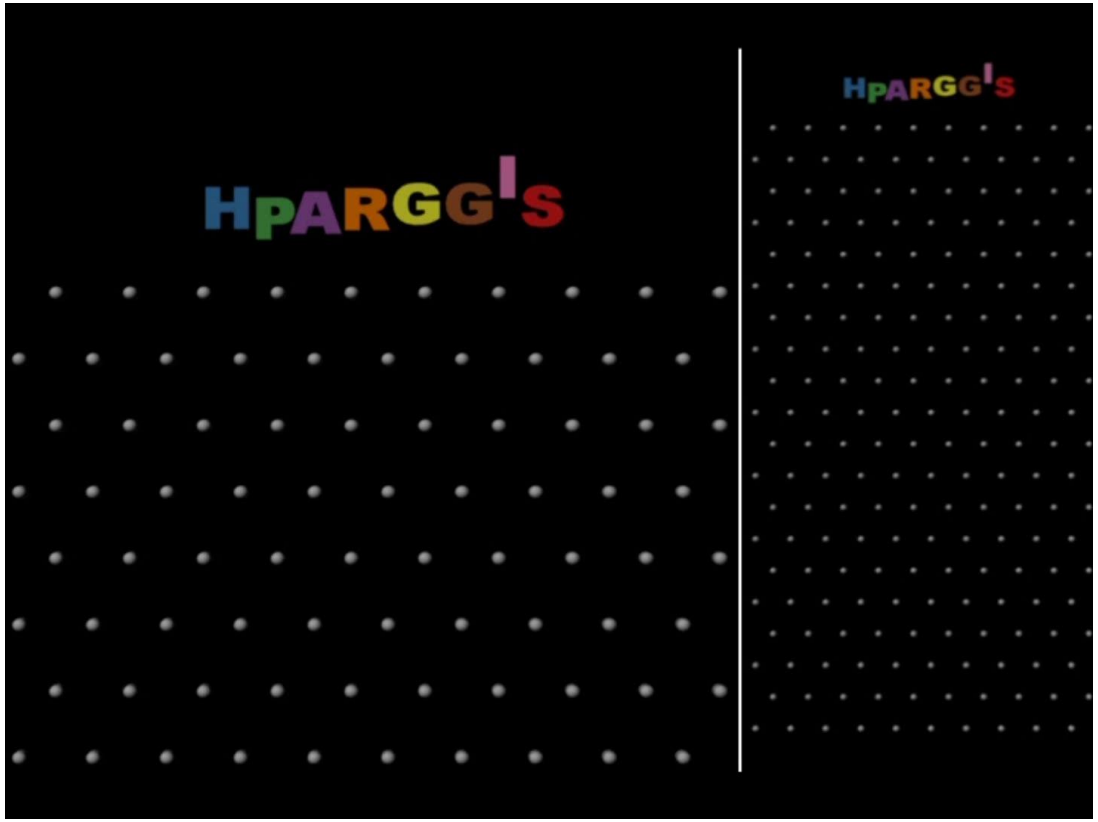
Design Space Exploration

- Sample parameter space
  - E.g. Poisson sampling
- Present designs in a manageable way
  - Cluster similar designs
  - Visualize designs exhibiting greatest variation
  - Hierarchical refinement



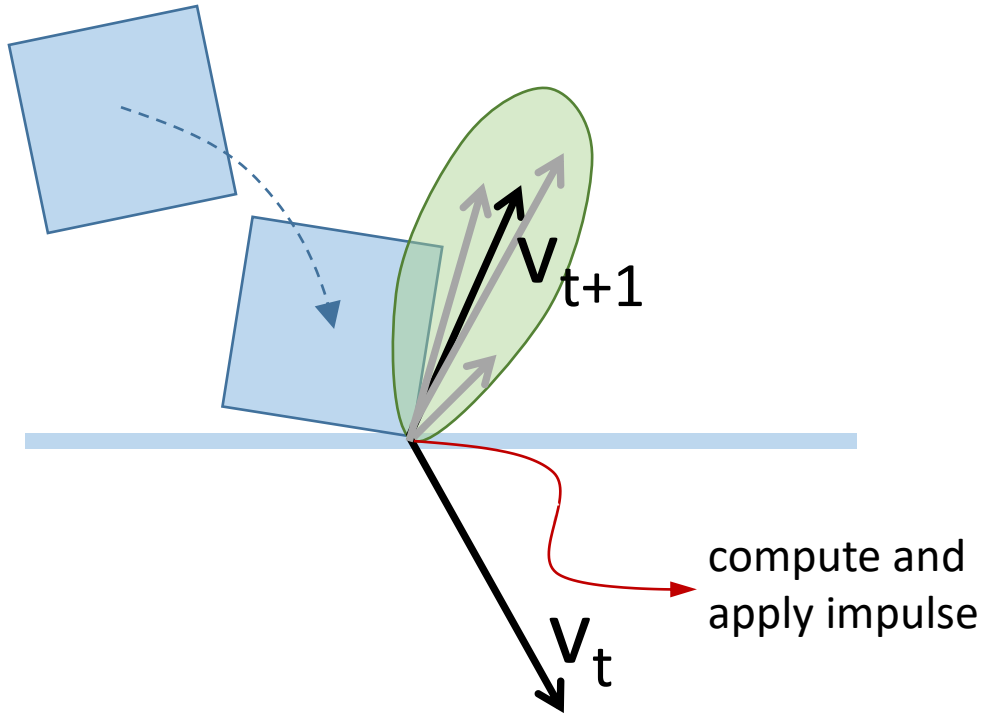
Design Space Exploration - Examples

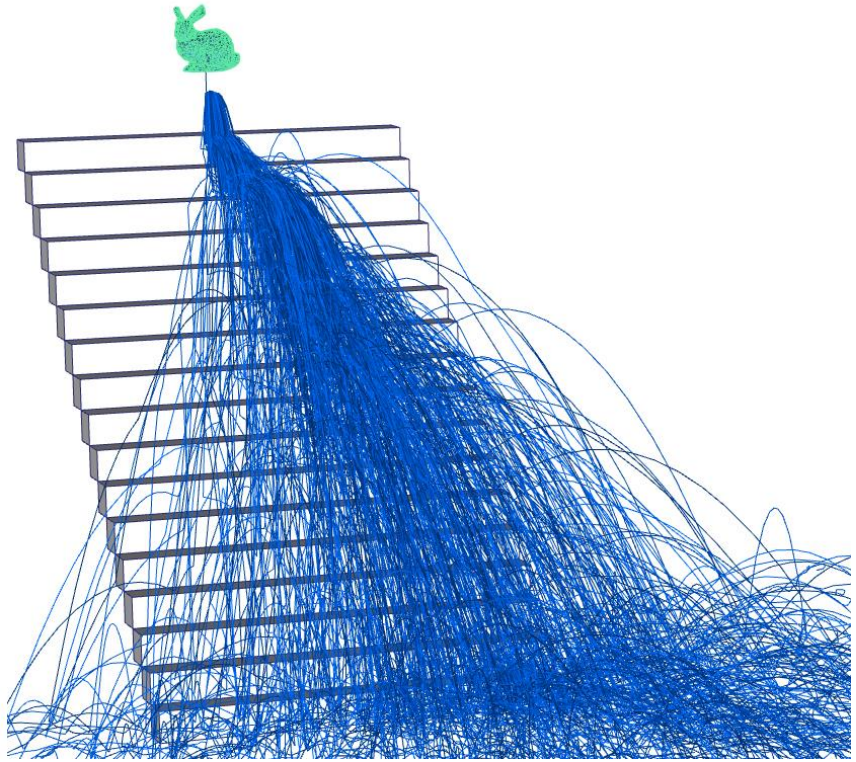
Many-Worlds Browsing for Control of Multibody Dynamics Twigg and James, SIGGRAPH 2007



Many Worlds Browsing

Sampling Plausible Worlds (parameter choices)

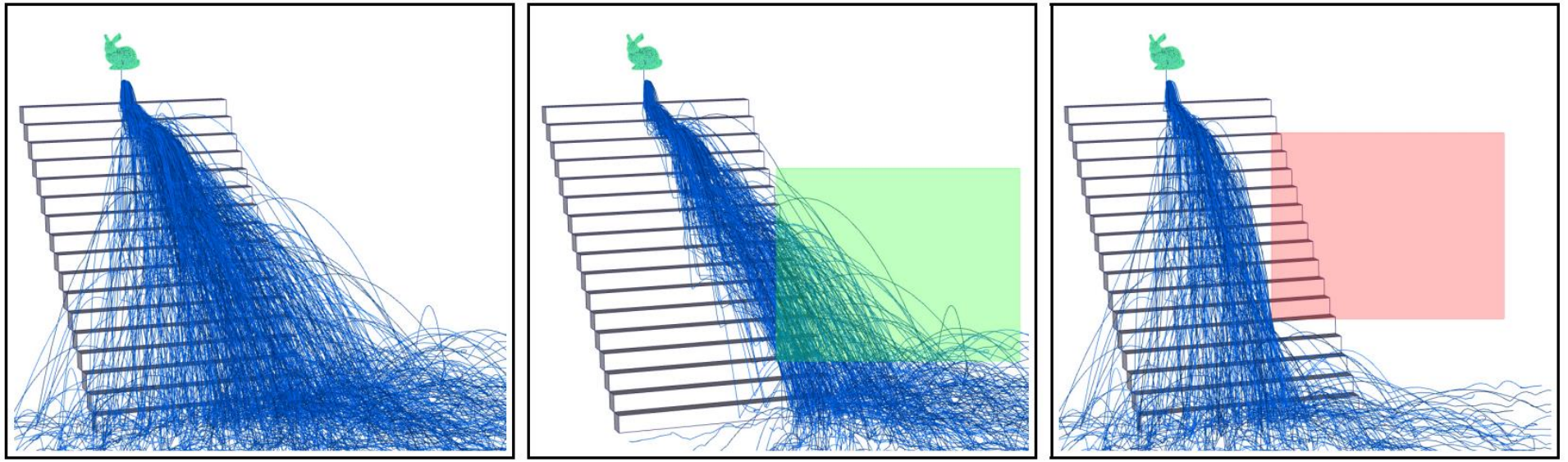




[O'Sullivan et al., 2003]

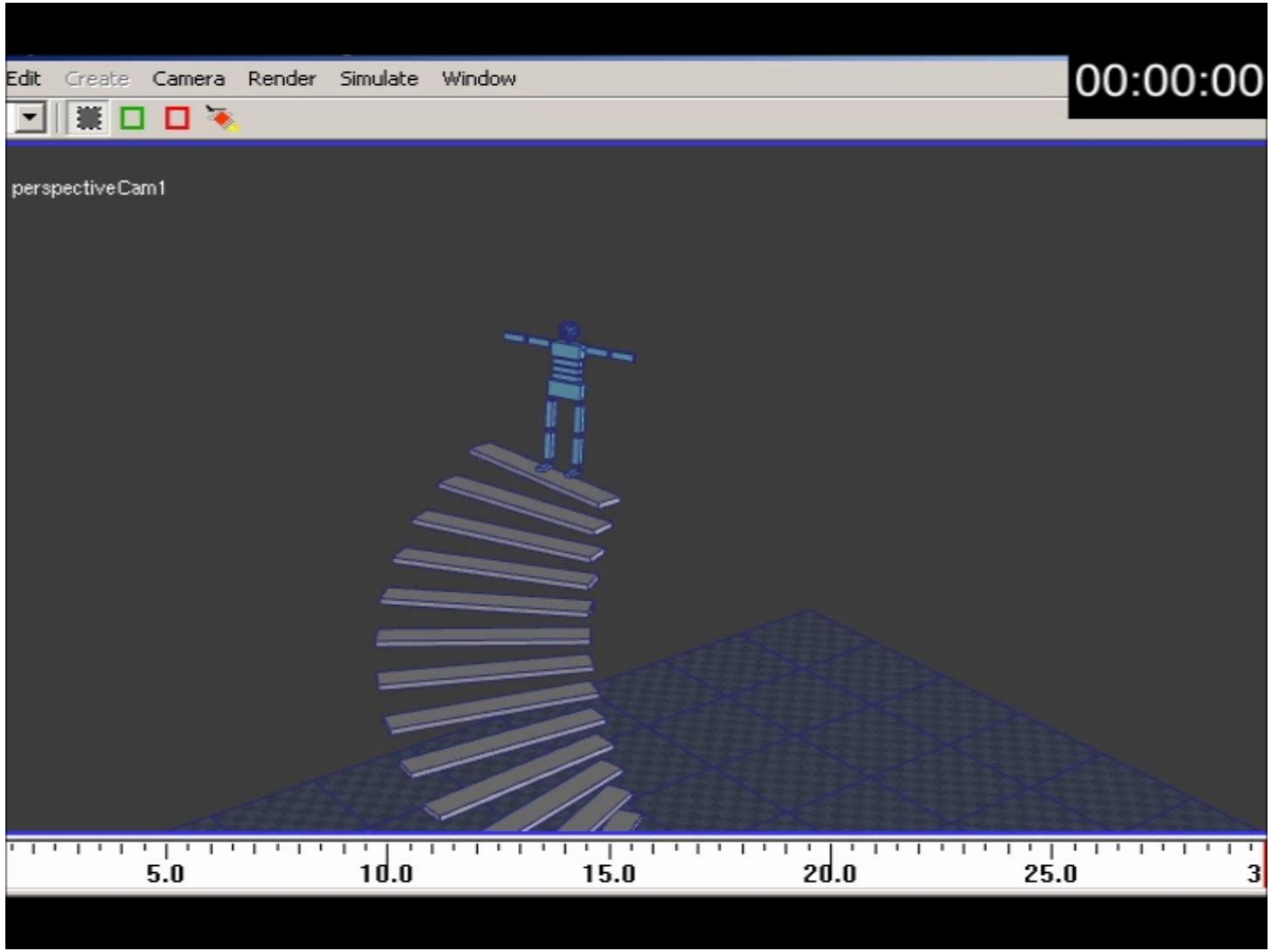
Many Worlds Browsing

Interactive Browsing – various criteria



Input Positive Negative

Many Worlds Browsing

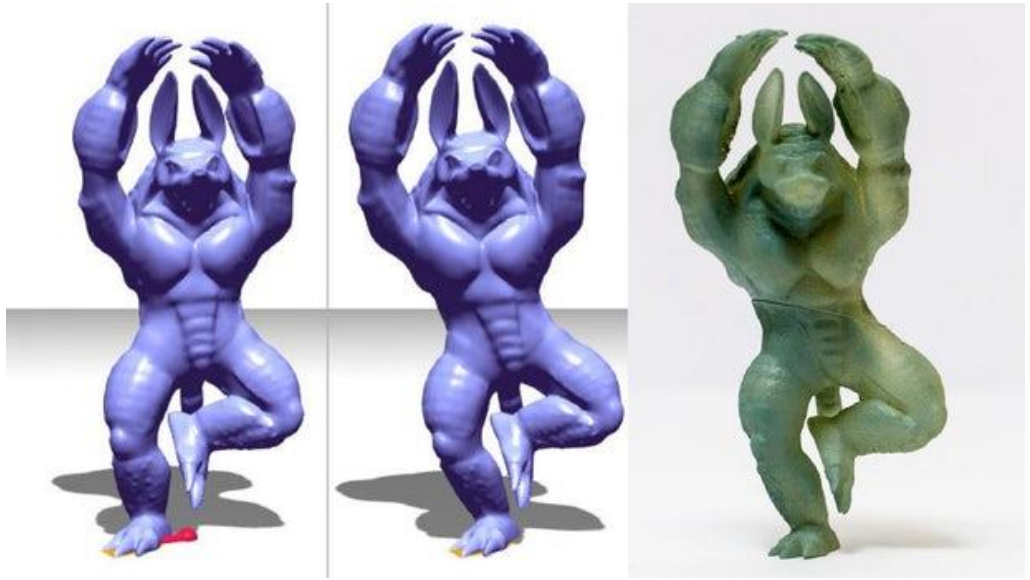


Computational Design

- Forward design: direct manipulation of design parameters
  - Level of abstraction matters

- Inverse design: automatically infer design parameters from functional specifications
  - Demands optimization-based solutions

    e.g. $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ s.t. $\boldsymbol{C}(\boldsymbol{x}) = 0, \boldsymbol{A}(\boldsymbol{x}) > 0$
  - What would this look like for "Make it stand?"

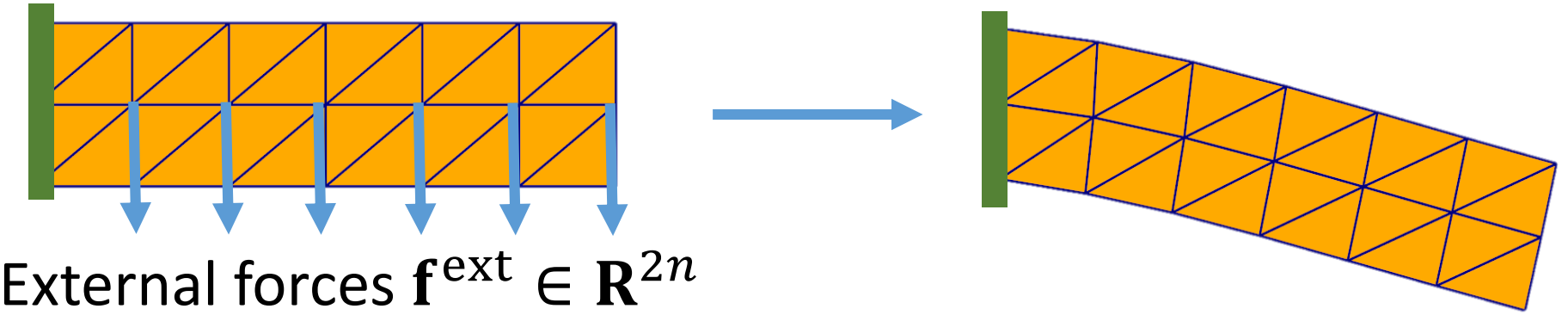


Make It Stand: Balancing Shapes for 3D Fabrication, Prevost et al. 2013

Another example - 2D Elastic Bar

Undeformed state $\overline{\boldsymbol{x}} \in \mathbf{R}^{2n}$

Deformed state $\mathbf{x} \in \mathbf{R}^{2n}$

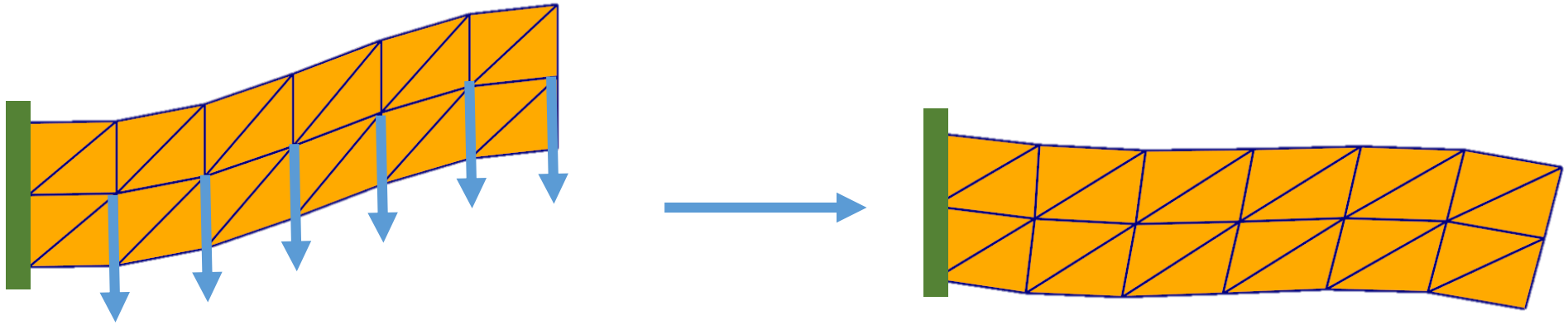


External forces $\mathbf{f}^{\text{ext}} \in \mathbf{R}^{2n}$

- *Forward problem*: given $\overline{\boldsymbol{x}}$ and $\mathbf{f}^{\text{ext}}$, compute equilibrium configuration $\mathbf{x}$ by solving

$$\mathbf{f}(\mathbf{x}, \overline{\boldsymbol{x}}) = \mathbf{f}^{\text{ext}} + \mathbf{f}^{\text{int}}(\mathbf{x}, \overline{\boldsymbol{x}}) = \mathbf{0}$$

Another example - 2D Elastic Bar



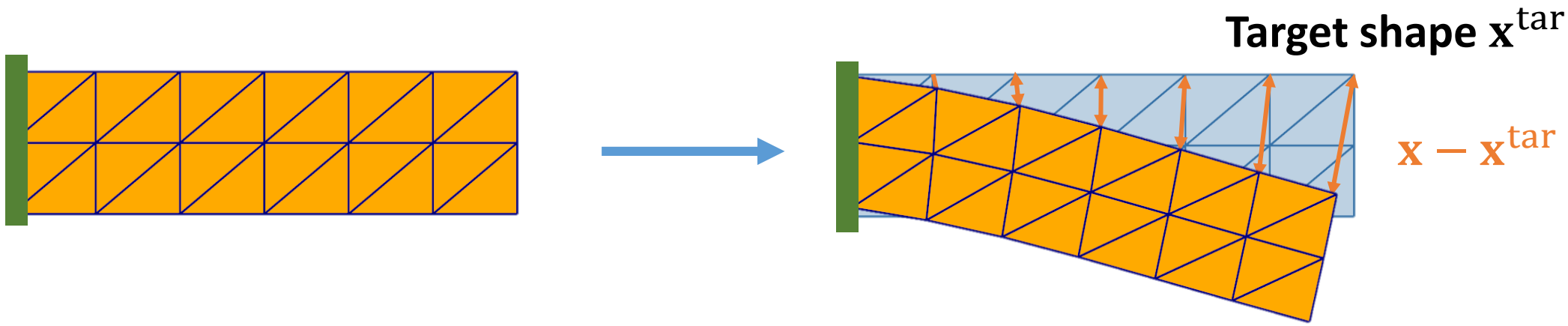
- *Observation*: changing the undeformed state $\overline{\boldsymbol{x}}$ changes the equilibrium state $\mathbf{x}$

  How can we determine $\overline{\boldsymbol{x}}$ that leads to a desired equilibrium state?

Objective

- Introduce *objective* that quantifies distance to target

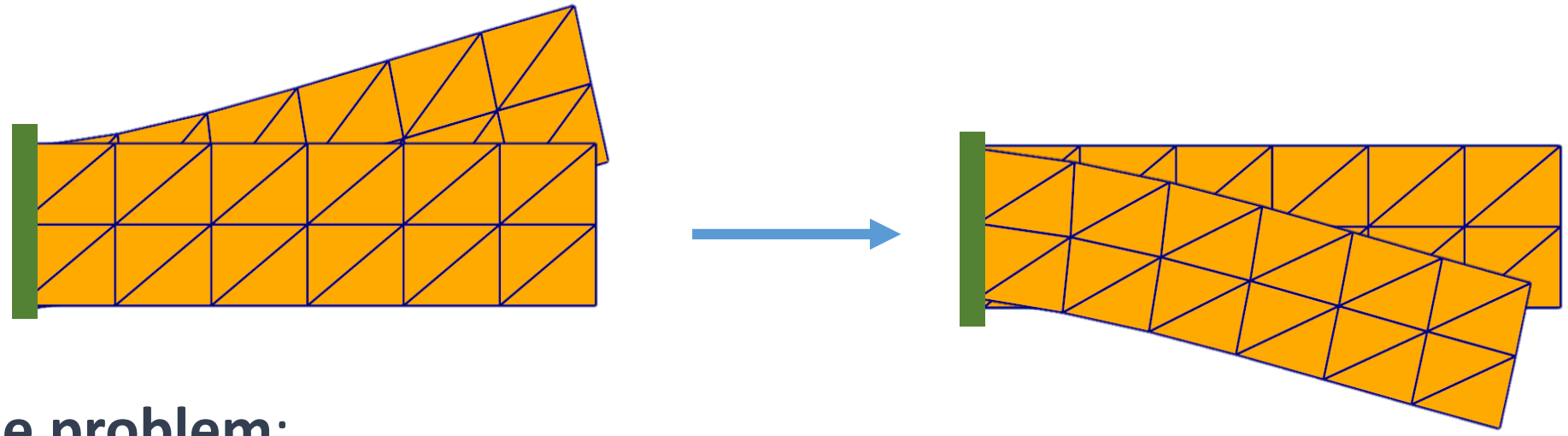
$$T(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{x}^{\text{tar}}\|^2$$



- **Goal**: find $\mathbf{x}$ and $\overline{\boldsymbol{x}}$ such that $\mathbf{x}$ minimizes distance to target
- **Constraint**: $\mathbf{x}$ has to be an equilibrium state for $\overline{\boldsymbol{x}}$, i.e., $\mathbf{f}(\mathbf{x}, \overline{\boldsymbol{x}}) = \mathbf{0}$

Inverse Problem

$$\min_{\bar{\boldsymbol{x}},\boldsymbol{x}} \; T(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{f}(\mathbf{x},\bar{\boldsymbol{x}}) = \mathbf{0}$$



Inverse problem:

from all possible equilibrium states $\mathbf{x}$,

i.e., those $\mathbf{x}$ for which there exists $\bar{\boldsymbol{x}}$ such that $\mathbf{f}(\mathbf{x},\bar{\boldsymbol{x}}) = \mathbf{0}$,

find the one that minimizes $T(\mathbf{x})$.

An Asymptotic Numerical Method for Inverse Elastic Shape Design, Chen et al. 2014

https://www.youtube.com/watch?v=PqvpNEi2Rfg

Generic optimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{C}(\boldsymbol{x}) = 0, \boldsymbol{A}(\boldsymbol{x}) > 0$$

- Objective function $f(\boldsymbol{x})$: $\mathbf{R}^n \to \mathbf{R}$
- Unknowns $\boldsymbol{x} \in \mathbf{R}^n$
- Constraints $\boldsymbol{C}(\boldsymbol{x})$: $\mathbf{R}^n \to \mathbf{R}^p$, $\boldsymbol{A}(\boldsymbol{x})$: $\mathbf{R}^n \to \mathbf{R}^q$

How can we solve such an optimization problem?

Optimization Techniques

- Unconstrained optimization with penalty/barrier functions
- Non-linear programming / (sequential) quadratic programming
- Randomized search
- Sensitivity Analysis
- …

Unconstrained optimization with penalty and/or barrier functions

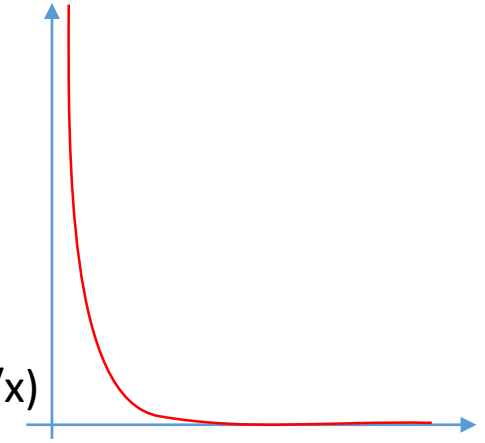
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{C}(\boldsymbol{x}) = 0, \boldsymbol{A}(\boldsymbol{x}) > 0$$

becomes

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda \boldsymbol{C}(\boldsymbol{x})^T \boldsymbol{C}(\boldsymbol{x}) + \Psi(\boldsymbol{A}(\boldsymbol{x}))$$

large constant

Barrier function (e.g. f(x)=1/x)



You should not use this approach with first-order methods!

Constrained optimization – First Naïve Approach

Generic optimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{C}(\boldsymbol{x}) = 0$$

- Objective should be at minimum $\rightarrow \nabla f(\boldsymbol{x}) = \boldsymbol{0}$
- Constraints should be satisfied $\rightarrow \boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{0}$
- Formulate as $\boldsymbol{g}(\boldsymbol{x}) = (\nabla f(\boldsymbol{x})^T, \boldsymbol{C}(\boldsymbol{x})^T)^T = \boldsymbol{0}$ and solve with Newton

- **Problem**: in general, there is no $\boldsymbol{x}$ such that $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$.

Counter Examples

Example 1: $\min_x x^2$ s.t. $x = 1$

- Solution is $x = 1$, but $\nabla f(1) = 2$

Rethinking the Problem

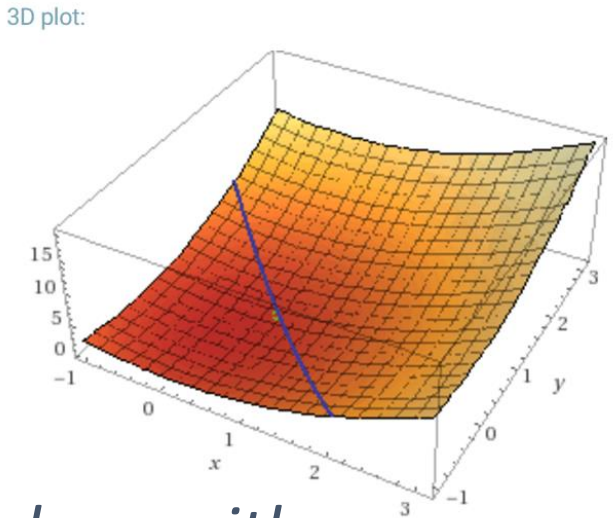
- Assume that for given $\boldsymbol{x}$ the constraints are satisfied, $\boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{0}$, but $\boldsymbol{x}$ is not the optimum.
- Then $\exists \boldsymbol{dx}$ such that $\boldsymbol{C}(\boldsymbol{x} + \boldsymbol{dx}) = \boldsymbol{0}$ and $f(\boldsymbol{x} + \boldsymbol{dx}) < f(\boldsymbol{x})$
- For linear constraints $\boldsymbol{C}(\boldsymbol{x} + \boldsymbol{dx}) = \boldsymbol{C}(\boldsymbol{x}) + \boldsymbol{\nabla \mathbf{C}(x) dx}$
- For small $|\boldsymbol{dx}|$, $f(\boldsymbol{x} + \boldsymbol{dx}) < f(\boldsymbol{x})$ implies that $\boldsymbol{\nabla f(x)^T dx < 0}$
- In summary, a valid search direction $\boldsymbol{dx}$ must satisfy

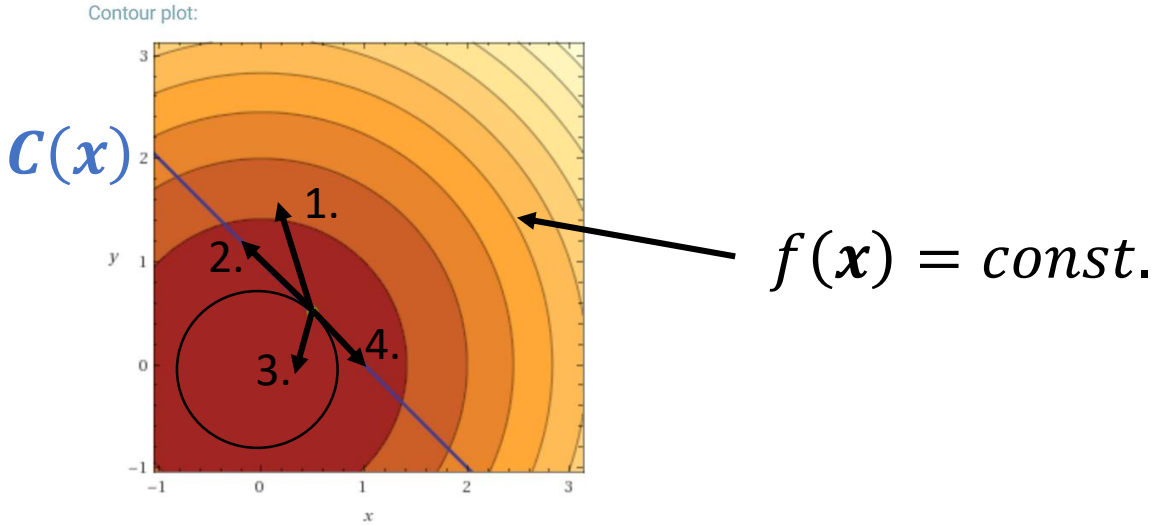
$$\boldsymbol{\nabla \mathbf{C}(x) dx = 0} \qquad \text{and} \qquad \boldsymbol{\nabla f(x)^T dx < 0}$$

Rethinking the Problem

$$\nabla \mathbf{C}(\boldsymbol{x})\boldsymbol{dx} = \mathbf{0} \qquad \text{and} \qquad \nabla f(\boldsymbol{x})^T \boldsymbol{dx} < \mathbf{0}$$

What does that mean?





For all dx, we have either

1. $f(\boldsymbol{x}+\boldsymbol{dx}) \geq f(\boldsymbol{x})$ and $\boldsymbol{C}(\boldsymbol{x}+\boldsymbol{dx}) \neq \mathbf{0}$
2. $\boldsymbol{C}(\boldsymbol{x}+\boldsymbol{dx}) = \mathbf{0}$ but $f(\boldsymbol{x}+\boldsymbol{dx}) \geq f(\boldsymbol{x})$
3. $f(\boldsymbol{x}+\boldsymbol{dx}) < f(\boldsymbol{x})$ but $\boldsymbol{C}(\boldsymbol{x}+\boldsymbol{dx}) \neq \mathbf{0}$
4. $f(\boldsymbol{x}+\boldsymbol{dx}) < f(\boldsymbol{x})$ but $\boldsymbol{C}(\boldsymbol{x}+\boldsymbol{dx}) = \mathbf{0}$

Optimality Conditions

- First Order Optimality Conditions

$$\nabla f(\mathbf{x}) = \nabla \mathbf{C}(\mathbf{x})^T \boldsymbol{\lambda} \text{ , and}$$
$$\mathbf{C}(\mathbf{x}) = \mathbf{0}$$

- Solving the constrained optimization problem: find $\mathbf{x}$ and $\boldsymbol{\lambda}$ such that the First Order Optimality Conditions are satisfied.
- **Note 1**: these conditions are also known as Karush-Kuhn-Tucker (KKT) conditions.
- **Note 2**: the KKT conditions are necessary, but not sufficient in general for $(\boldsymbol{x}, \boldsymbol{\lambda})$ to be a (strict local) solution to the optimization problem.

The Lagrangian – A Reformulation

- Define the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{C}(\boldsymbol{x})$$

- Consider the gradient of $L(\boldsymbol{x}, \boldsymbol{\lambda})$, i.e.,

$$\nabla_{\boldsymbol{x}} L = \nabla f + \nabla C^T \lambda \qquad \text{and} \qquad \nabla_{\lambda} L = \boldsymbol{C}(\boldsymbol{x})$$

- **Observations**:
  - the first-order optimality conditions correspond to $\nabla L = 0$
  - Solving the optimality conditions means we are looking for a stationary point of L
  - Since $\boldsymbol{C}$ is not bounded from below or above, $\nabla L = 0$ must be a saddle point
  - Aside: an optimization problem with quadratic f and linear $\boldsymbol{C}$ is called a **Quadratic Program** (QP).

Example

Optimization problem

$$\min x^2 \;\; s.t. \;\; x^2 - 1 = 0$$

Lagrangian

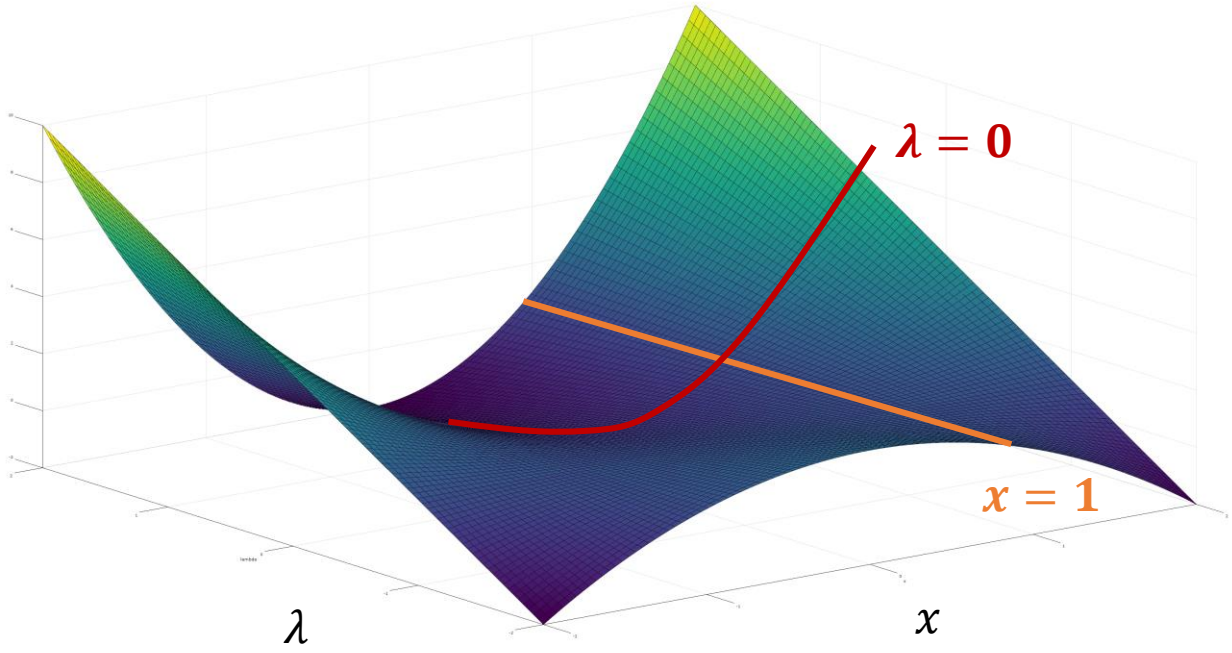
$$L(x, \lambda) = x^2 + \lambda(x^2 - 1)$$

KKT conditions

$$\nabla_x L = 2x + 2\lambda x = 0$$
$$\nabla_\lambda L = x^2 - 1 = 0$$

Solutions

$$(x_1^*, \lambda_1^*) = (1, -1)$$
$$(x_2^*, \lambda_2^*) = (-1, -1)$$



Quadratic Programming Assumptions

- The objective function $f(\boldsymbol{x})$ is quadratic, i.e.,

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + \boldsymbol{a}$$

- All m constraints are linear equality constraints, i.e.,

$$\boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

- Constraint gradients are linearly independent, i.e., $\text{rank}(\boldsymbol{A}) = m$
  - No redundant constraints → the optimal $\boldsymbol{\lambda}$ are unique
  - No inconsistent/conflicting constraints → problem is feasible

Solving a Quadratic Program*

- For a quadratic program, we have

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + \boldsymbol{a} \quad \text{and} \quad \boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

- Therefore

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + \boldsymbol{a} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

$$\nabla_{\boldsymbol{x}} L = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{A}^T\boldsymbol{\lambda} \qquad \text{and} \qquad \nabla_{\boldsymbol{\lambda}} L = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

- For first-order optimality, we need

$$\begin{bmatrix} \nabla_{\boldsymbol{x}} L \\ \nabla_{\boldsymbol{\lambda}} L \end{bmatrix} = \mathbf{0} \quad \Rightarrow \quad \begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix}$$

*with only equality constraints

Solving a Quadratic Program

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix}$$

- KKT-Matrix is
  - Symmetric
  - Indefinite (n positive eigenvalues, m negative eigenvalues) – Why?
  - Non-singular (since $\boldsymbol{A}$ is full-rank and $\boldsymbol{H}$ is non-singular)
- If $\boldsymbol{A}$ is full-rank and $\boldsymbol{H}$ is positive-definite, then the QP is termed **convex** and has a unique solution
- How can we compute this solution?

Solving the KKT System

- Direct indefinite solvers
  - Cannot use Cholesky since the KKT matrix is indefinite
  - Can use LU, but ignores symmetry
  - Pardiso (Parallel symmetric indefinite factorization, i.e., $PAP^T = LDL^T$)
- Iterative solvers such as the Uzawa algorithm
- Alternatively: use QP solver
  - Mosek
  - ..

Inequality Constraints

- Inequality constraints occur naturally in optimization problems
  - Positivity on variables, $x_i \geq 0$
  - Limited resources available, but not all have to be used, $\sum_i x_i \leq M$
- Let $\mathcal{E}$ denote the index set of all equality constraints. Then

$$c_i(\boldsymbol{x}) = 0 \;\; \forall i \in \mathcal{E}$$

- Let $\mathcal{I}$ denote the index set of all equality constraints. Then

$$c_i(\boldsymbol{x}) \geq 0 \;\; \forall i \in \mathcal{I}$$

- Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x)$$

Inequality Constrained Problems

- Lagrangian $$\mathcal{L}(x, \lambda) = f(x) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x)$$

- First-order optimality (KKT) conditions

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, \\
c_i(x^*) &= 0, \quad \text{for all } i \in \mathcal{E}, \\
c_i(x^*) &\geq 0, \quad \text{for all } i \in \mathcal{I}, \\
\lambda_i^* &\leq 0, \quad \text{for all } i \in \mathcal{I}, \\
\lambda_i^* c_i(x^*) &= 0, \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}.
\end{aligned}$$

Feasibility: Inequality constraints have to be satisfied

One-sidedness: Inequality constraints can only push, not pull

Complementarity: Either constraint is active, or its LM is zero

Active Set

- For a feasible point $\boldsymbol{x}$, the inequality constraint c_i is

  $$\textit{active} \text{ if } c_i(\boldsymbol{x}) = 0 \quad \text{and} \quad \textit{inactive} \text{ if } c_i(\boldsymbol{x}) > 0.$$

- For any feasible point $\boldsymbol{x}$, the active set $\mathcal{A}(\boldsymbol{x})$ is defined as

  $$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(x) = 0\}$$

- **Assumption**: we have a QP with inequality constraints (ICs)
- If we knew the active set, then we could just solve an equality-constrained QP with only the active IC present
- However, we generally do not know the active set in advance
- **Idea**: instead of explicitly enforcing complementarity conditions, build active set iteratively by *guessing* active constraints and solving QPs until optimality conditions are satisfied.

Nonlinear Programming

- What changes if f and $\boldsymbol{C}$ are no longer quadratic/linear?
  - KKT-conditions are still necessary
  - Optimization problem will generally have multiple local minima
- **Strategy**: solve non-linear KKT conditions to find local optimum.

Nonlinear Programming

$$\begin{bmatrix} \nabla_{xx} L(\boldsymbol{x}) & \nabla \boldsymbol{C}(\boldsymbol{x})^T \\ \nabla \boldsymbol{C}(\boldsymbol{x}) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \nabla_x L \\ \nabla_\lambda L \end{bmatrix}$$

- Since the KKT system is based on a first-order approximation, $\nabla_s L(\boldsymbol{s} + \Delta \boldsymbol{s}) \neq \mathbf{0}$ in general.
- Idea: *iterate!*

Sequential Quadratic Programming (SQP)

Until convergence

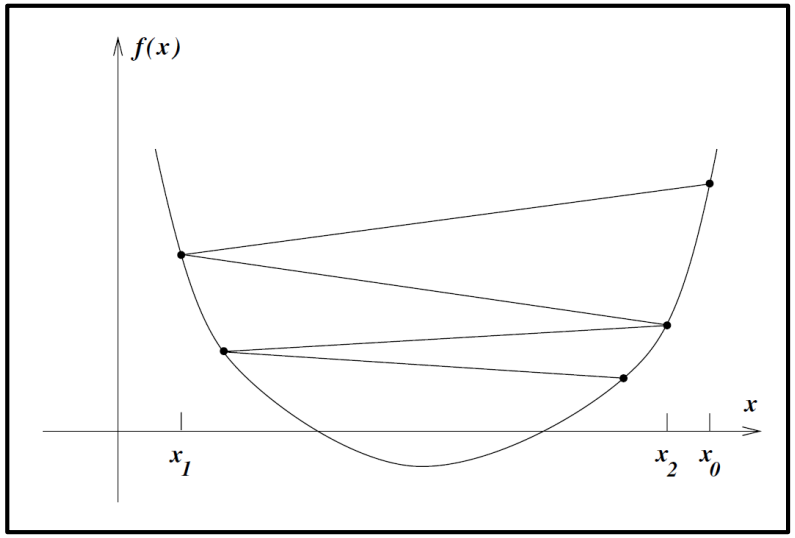
solve	$\nabla_{\mathbf{ss}} L \cdot \Delta \mathbf{s} = -\nabla_{\mathbf{s}} L$
line search	$\alpha = \text{line\_search}(\mathbf{s}, \Delta \mathbf{s})$
update	$\mathbf{s} = \mathbf{s} + \alpha \Delta \mathbf{s}$

end

Line Search – Unconstrained Case

- **Idea**: ensure progress by decreasing the objective in each iteration
- Simplest version: given search direction, compute step length such that

$$f(x_k + \alpha p_k) \leq f(x_k)$$



Line Search – Constrained Case

- Similar to (unconstrained) nonlinear minimization, we need a *step length control mechanism* to ensure convergence for nonlinear constrained optimization
- Can we use the objective function to measure progress?
  - If method generates only feasible configurations (and search directions), yes.
  - *Most algorithms start from (and generate intermediate) infeasible configurations*
- Therefore, we need ways to balance progress between objective and constraints → *merit functions* and *filters*
- Why not use the Lagrangian?
  - We do not know the function value at the optimum (could be smaller or larger)

Merit Functions

- **Idea**: combine objective and constraint values into one merit function that measures progress
- l_1 merit function

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

- l_2 merit function

$$\phi_2(x;\mu) = f(x) + \mu \|c(x)\|_2$$

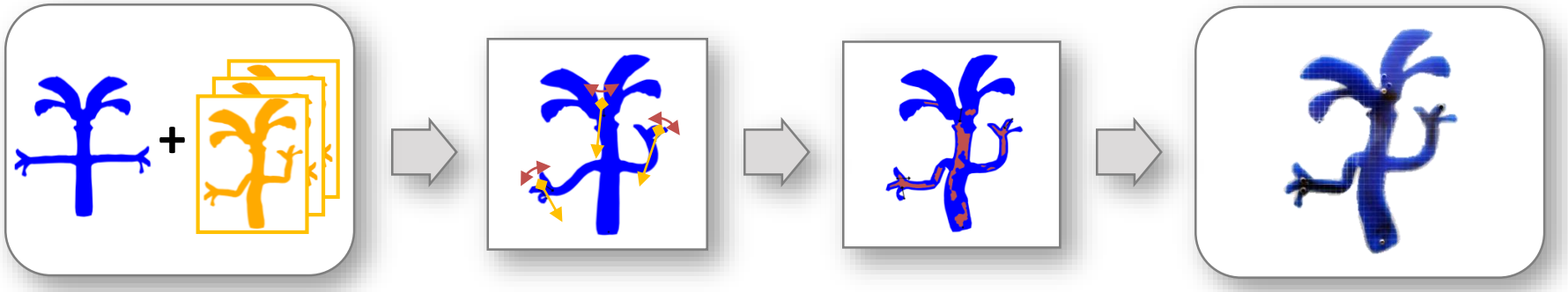
Inverse Design Example

- Design a physical object that can deform like this:

- Need model that predicts the way in which objects deform
  - Design parameters: material parameters, points of application for forces or constraints

“Computational Design of Actuated Deformable Characters”, Skouras et al, 2013

Pipeline

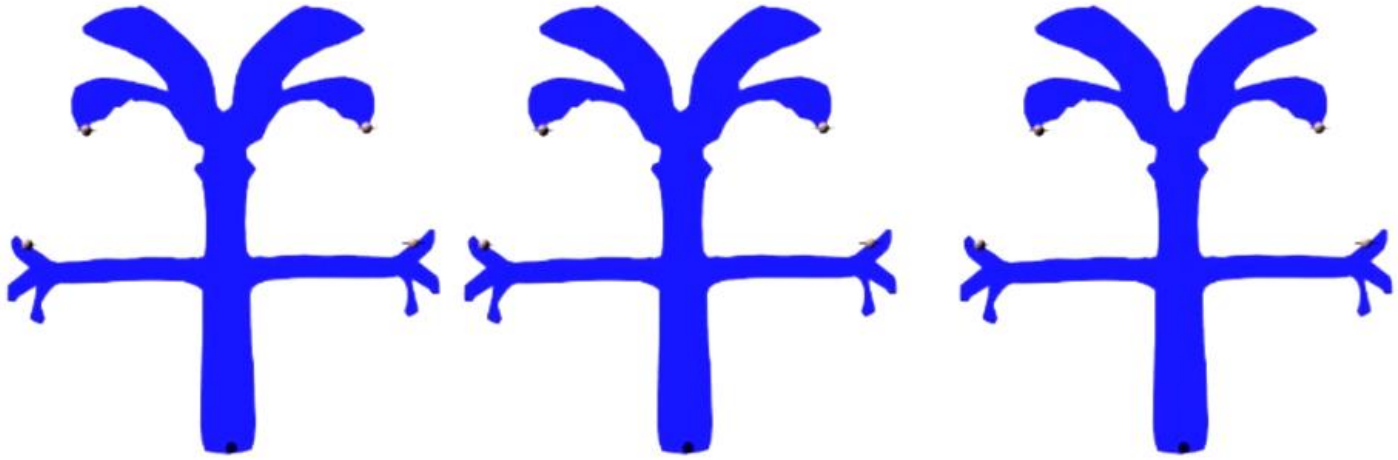

Input and Target Shapes
Actuator Location Optimization
Material Optimization
Fabricated Deformable Model

Mathematical Formulation

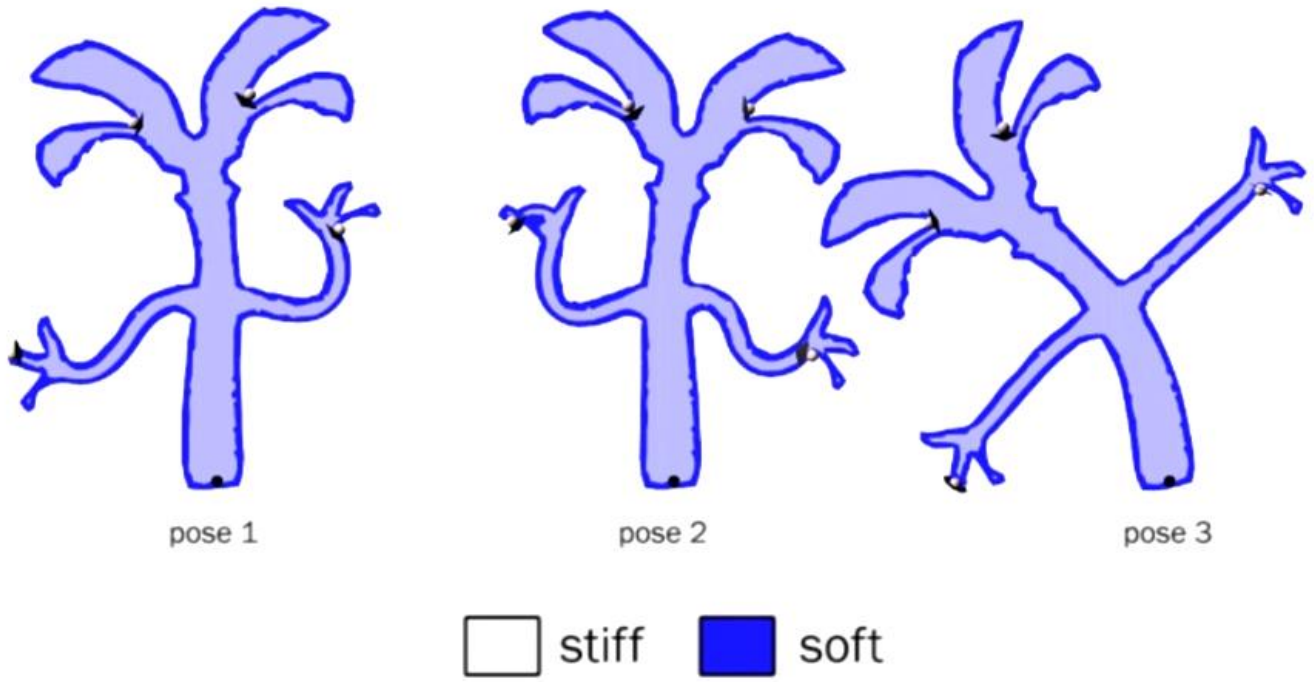
Design Parameters

$$E(\mathbf{x}, \mathbf{p}) = E_{match}(\mathbf{x}, \mathbf{x}_{target}) \quad \text{subject to} \quad \mathbf{f}^{total}(\mathbf{x}, \mathbf{p}) = 0$$

Actuator Location Optimization



Material Distribution Optimization



Further Points

- Line search vs. Trust Region methods
- Merit functions vs. filters
- Active set vs. interior point methods
- SQP vs. nonlinear interior point methods
- …

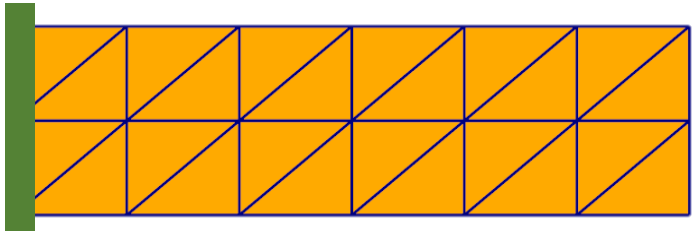
SQP Discussion

- SQP is a powerful optimization method
- Hessian of Lagrangian involves higher-order derivatives
  - can be difficult to derive and expensive to compute
  - can introduce indefiniteness
- Finding a good merit function and/or coefficient update scheme can be difficult
- SQP leads to large number of variables
  - One variable per degree of freedom
  - One variable per parameter
  - One Lagrange multiplier per constraint

Problem Size Inflation

- The SQP formulation leads to a large number of variables
- Example: material design for 2D bar

$$\min_{\boldsymbol{x},k} T(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{f}(\boldsymbol{x},k) = \boldsymbol{0}$$



  - 24 nodes
  - 48 variables for deformed positions
  - 24 variables for material stiffness k (one per element)
  - 48 variables for constraints (Lagrange multipliers)
- Although the number of design parameters is relatively small, SQP leads to an increase in problem size with every constraint
- *Is there a way to only use the real variables of the design problem?*

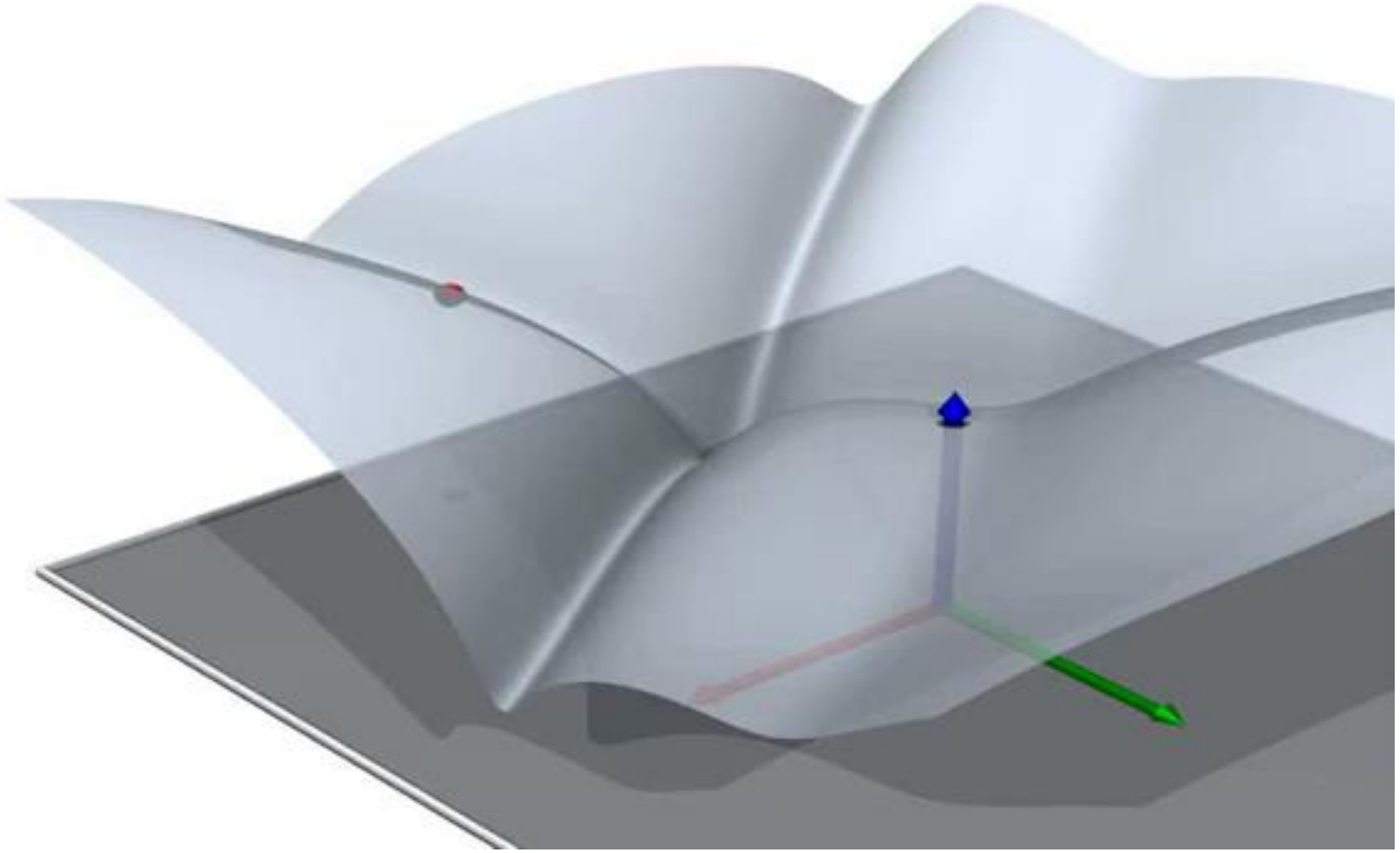
Randomized Search - Simulated Annealing

- Has four ingredients
  - Cost function
  - Configuration (e.g. parameter values)
  - Neighbor generator
  - Annealing schedule

Randomized Search - Simulated Annealing

- Basic concept taken from cooling of materials in metallurgy
  - At high “heat” atoms undergo rigorous motion, as they are cooled they move less
- Used to explore trade-off between exploration and exploitation

Randomized Search - Simulated Annealing



Examples from Graphics

Modeling continuous Relation between Parameters and State

- **Observation**: when we set parameters $\boldsymbol{p}$, we observe the state $\boldsymbol{x}$ as the result of simulation.
- Although $\boldsymbol{x}$ are problem variables, they are not real DOFs – they are functions of the parameters, i.e.,

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{p})$$

- Map from parameters to state is

$$\boldsymbol{x} = \text{simulate}(\boldsymbol{p})$$

- For design, we need derivatives of $\boldsymbol{x}(\boldsymbol{p})$,

$$\frac{\partial T}{\partial \boldsymbol{p}} = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}\right)^t \frac{\partial T}{\partial \boldsymbol{x}}$$

- But how to compute these derivatives,

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}} = \frac{\partial \text{simulate}}{\partial \boldsymbol{p}}?$$

Differentiating the Map

- Although we can evaluate the map $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$, this map is not available in closed-form (i.e., *analytically*)
- $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$ requires minimizing a function, i.e., solving a system of nonlinear equations.
- In general, it is impractical to compute derivatives of the minimization process.
- But even though $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$ is not given *explicitly*, the gradient of the objective

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{p}) = \boldsymbol{0}$$

provides this map *implicitly*.

Differentiating the Map

- Suppose that $(\boldsymbol{x},\boldsymbol{p})$ is a feasible pair, i.e., $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{p}) = \boldsymbol{0}$. In other words, $\boldsymbol{x}$ is an equilibrium configuration for $\boldsymbol{p}$.
- If we apply a parameter perturbation $\Delta\boldsymbol{p}$, the system will undergo displacements $\Delta\boldsymbol{x}$ such that it is again in equilibrium,

$$\boldsymbol{f}(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{p}+\Delta\boldsymbol{p}) = \boldsymbol{0}\,.$$

- Since this has to hold for arbitrary parameter variations, we have

$$\frac{d\boldsymbol{f}}{d\boldsymbol{p}} = \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}\frac{\partial\boldsymbol{x}}{\partial\boldsymbol{p}} + \frac{\partial\boldsymbol{f}}{\partial\boldsymbol{p}} = \boldsymbol{0}\,.$$

- If the Jacobian $\nabla_{\boldsymbol{x}}\boldsymbol{f}$ is square and non-singular, we have

$$\frac{\partial\boldsymbol{x}}{\partial\boldsymbol{p}} = -\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{x}}^{-1}\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{p}}\,.$$

Implicit Function Theorem

- **Implicit function theorem (IFT)**: let $f: \mathbf{R}^{n+p} \to \mathbf{R}^n$ be a continuously differentiable function. If for given $\boldsymbol{x}_0 \in \mathbf{R}^n$ and $\boldsymbol{p}_0 \in \mathbf{R}^p$ we have $\boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{p}_0) = \mathbf{0}$ and if $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{\boldsymbol{x_0}}$ is invertible, then there exists a unique, continuously differentiable function $\boldsymbol{y}$ such that $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{p}), \boldsymbol{p}) = \mathbf{0}$ for all $\boldsymbol{p}$ in a Neighborhood $\mathcal{N}$ around $\boldsymbol{p}$.

- The IFT is applicable to equilibrium-constrained optimization problems
- The IFT asserts the existence and (local) uniqueness of the map $\boldsymbol{x} = \boldsymbol{y}(\boldsymbol{p})$ between parameters and state as well as its derivative(s).

Sensitivity Analysis

- Used in many applications to quantify the sensitivity of a solution with respect to parameters ($\boldsymbol{S} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}$ is also called the sensitivity matrix)
- Example: Robust Optimization
  - Optimize mechanical structure such that it supports given loads with least material
  - Fabrication and assembly will lead to inaccuracies
    → parameter uncertainties (nodes will not be exactly at their assumed position)
    → structure might collapse
  - $\boldsymbol{S}$ will reveal such sensitivities by showing large $\Delta \boldsymbol{x}$ for certain $\Delta \boldsymbol{p}$
  - Avoid such theoretically-optimal but fragile solutions in favor of robust solutions
- SA is also used widely for shape optimization problems and other equilibrium-constrained problems.

Sensitivity Analysis for Design

- We can compute the Jacobian $\boldsymbol{S} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}$. How can we use this for solving design problems?
- We can compute the gradient of the objective wrt. the parameters,

$$\frac{\partial T}{\partial \boldsymbol{p}} = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}\right)^T \frac{\partial T}{\partial \boldsymbol{x}} \quad \Rightarrow$$

Sensitivity Analysis Steepest Descent - SASD

Until convergence

$$\mathbf{S} = -\nabla_{\mathbf{x}}\mathbf{f}^{-1}\nabla_{\mathbf{p}}\mathbf{f}$$

$$\Delta\mathbf{p} = -\mathbf{S}^T\nabla_{\mathbf{x}}T$$

$$\alpha = \text{line\_search}(\Delta\mathbf{p})$$

$$\mathbf{p} = \mathbf{p} + \alpha\Delta\mathbf{p};$$

$$\mathbf{x} = \text{simulate}(\mathbf{x}, \mathbf{p})$$

end

SASD – Updating the State

- For a given parameter update $\Delta \boldsymbol{p}$, we need to compute corresponding state update $\Delta \boldsymbol{x}$.

Can we simply use $\Delta \boldsymbol{x} = \boldsymbol{S} \Delta \boldsymbol{p}$ *?*

- Since the forces (*constraints*) are nonlinear, $\mathbf{f}(\boldsymbol{x} + \boldsymbol{S}\Delta \boldsymbol{p}, \boldsymbol{p} + \Delta \boldsymbol{p}) \neq \mathbf{0}$ in general.
- However, when $\mathbf{f}(\boldsymbol{x}, \boldsymbol{p}) \neq \mathbf{0}$, the IFT does not hold and we cannot compute $\boldsymbol{S}$ anymore.
- Need to ensure $\mathbf{f}(\boldsymbol{x}, \boldsymbol{p}) = \mathbf{0}$ always, so solve for $\Delta \boldsymbol{x}$ using Newton's method

Adjoint Sensitivity Analysis

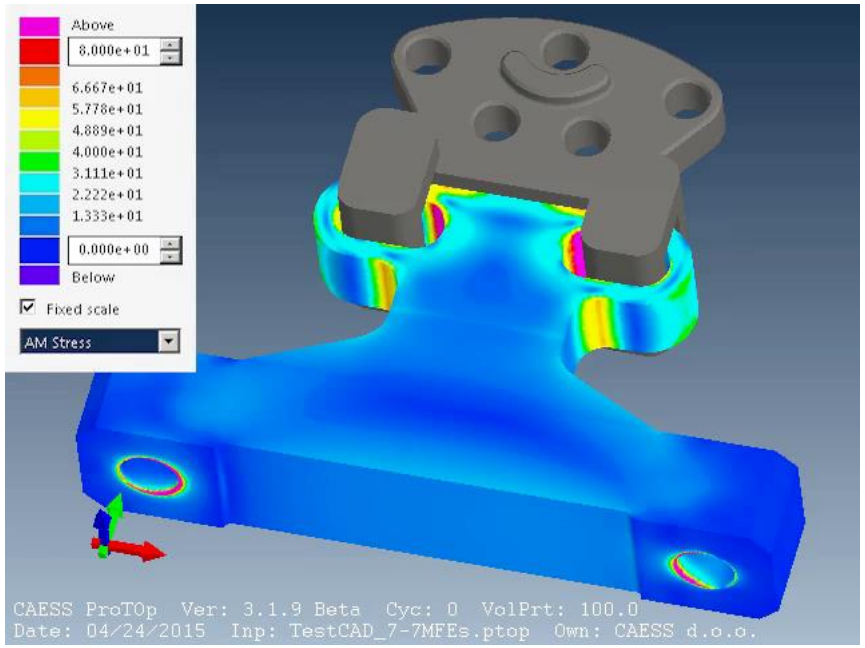
- Computing $\boldsymbol{S}$ is expensive for large number of design parameters.
- But do we need the entire matrix?
- Objective gradient $\frac{\partial T}{\partial \mathbf{p}} = \frac{\partial \mathbf{x}}{\partial \mathbf{p}}^T \frac{\partial T}{\partial \mathbf{x}}$

$$\boxed{\frac{\partial \mathbf{x}}{\partial \mathbf{p}} = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{p}}}$$

- Only need to compute action of $\boldsymbol{S}$ on $\nabla_x T$ $\rightarrow$ solve $\underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}}}_{\mathbf{g}} \mathbf{g} = \frac{\partial T}{\partial \mathbf{x}}$
- One linear solve instead of p $\rightarrow$ large savings for large number of design parameters

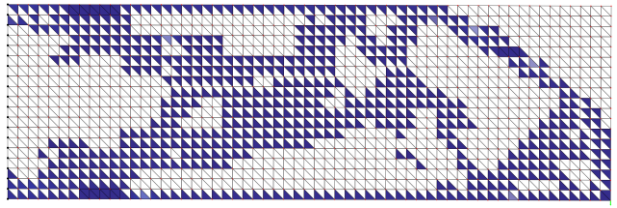
An example: topology optimization via Adjoint Sensitivity Analysis

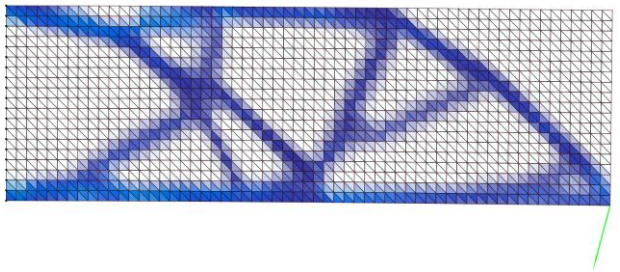
Generate™



Topology optimization

- Objective
  - Minimize compliance (e.g. deformations under load)
  - Encourage smooth solutions (e.g. eliminate checkerboarding/microstructure effects)

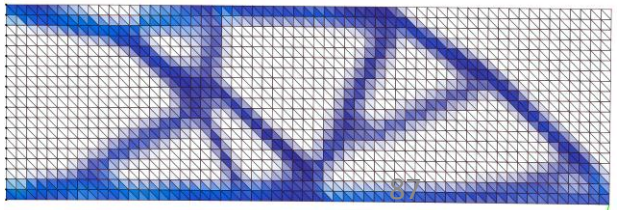




- Constraints
  - Upper limit on total amount of material used
  - Bounds on per-element density parameters
- How do we promote a binary density pattern?

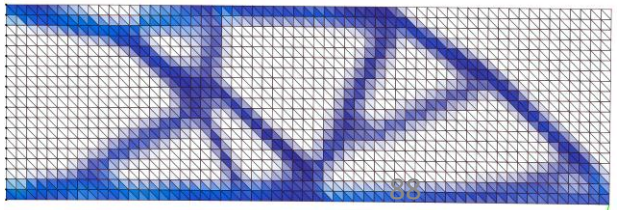
Topology optimization

- Design parameters $\boldsymbol{p}$: vector of scalars defining per-element material density: More specifically, define energy of element i as: $E_i = \left(p_{min} + \boldsymbol{p}_i^{\delta}\right)\bar{E}_i$, where $\bar{E}_i$ is the deformation energy associated with original material density, $0 \leq \boldsymbol{p}_i \leq 1$, δ is a constant scalar typically between 1 and 3, and p_{min} is a tiny number.
- Configuration of simulation mesh, deformed using current $\boldsymbol{p}$ values is $\boldsymbol{x}$
  - e.g. $\boldsymbol{x}(\boldsymbol{p}) = \text{argmin}_{\boldsymbol{x}^*}\, W(\boldsymbol{x}^*)$, where $W = \sum_i E_i$
- Objective: $\text{f}(\boldsymbol{p}) = E_{def}\left(\boldsymbol{x}(\boldsymbol{p})\right) + E_{smooth}(\boldsymbol{p})$
  - Compliance term: $E_{def}\left(\boldsymbol{x}(\boldsymbol{p})\right) = \sum_{\boldsymbol{i}} E_i$ (how deformed is the structure when density params are applied)?
  - Smoothness term: $E_{smooth}(\boldsymbol{p}) = \sum_{\boldsymbol{i}} \left(\boldsymbol{p_i} - \frac{\boldsymbol{1}}{\boldsymbol{n_i}} \sum_{\boldsymbol{k} \in \boldsymbol{N_i}} \boldsymbol{p_k}\right)$, where $\boldsymbol{N_i}$ is the set of elements neighboring element i



Topology optimization

- Constraints:
  - Density bounds $0 \leq \boldsymbol{p}_i \leq 1$
  - Upper limit on total mass: $\sum_i m_i \boldsymbol{p}_i \leq l$
- All derivatives are easy to compute analytically, but we need sensitivity analysis for the term $\frac{dE_{def}}{d\boldsymbol{p}}$:
  - Step 1: $\frac{dE_{def}}{d\boldsymbol{p}} = \frac{\partial E_{def}}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{d\boldsymbol{p}} + \frac{\partial E_{def}}{\partial \boldsymbol{p}}$
  - Step 2: $\frac{d\boldsymbol{x}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{G}^{-1}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{p}}$, where $\boldsymbol{G} = \frac{\partial W}{\partial \boldsymbol{x}}$
  - Step 3: Apply adjoint method to decrease computational cost of computing gradients



Topology optimization