

Computational Fabrication

Wojciech Matusik, Adriana Schulz

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s). SIGGRAPH '19 Courses, July 28 - August 01, 2019, Los Angeles, CA, USA ACM 978-1-4503-6307-5/19/07.
10.1145/3305366.3328064

Introduction

Manufacturing Today



Manufacturing in Near Future

THE FUTURE
OF EVERYTHING

WSJ

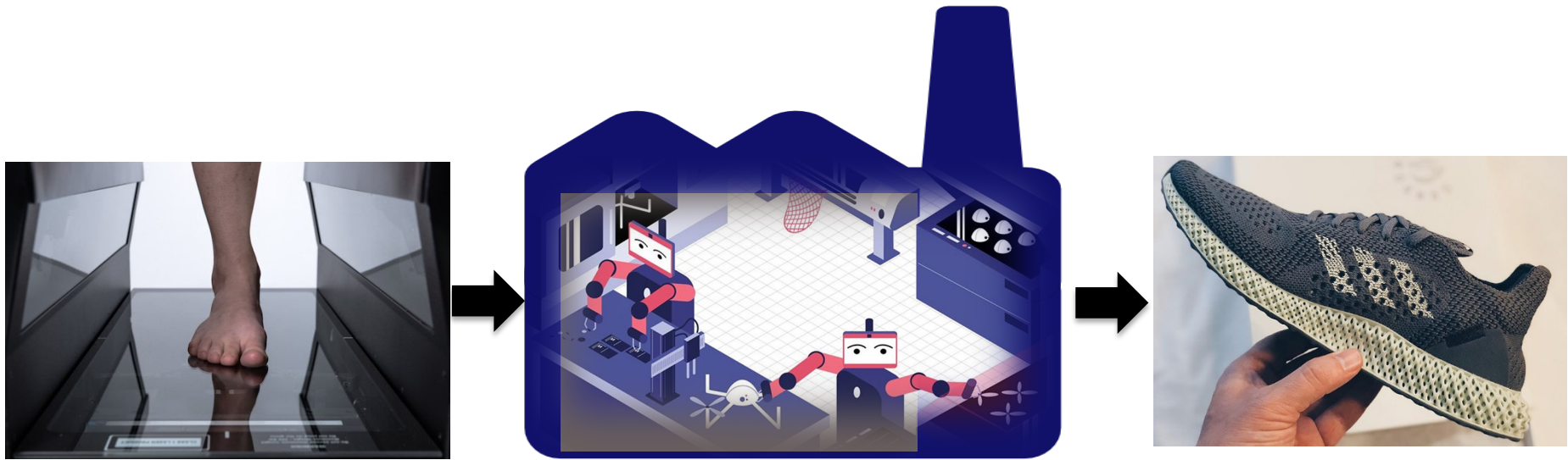
Carbon 3D - Adidas



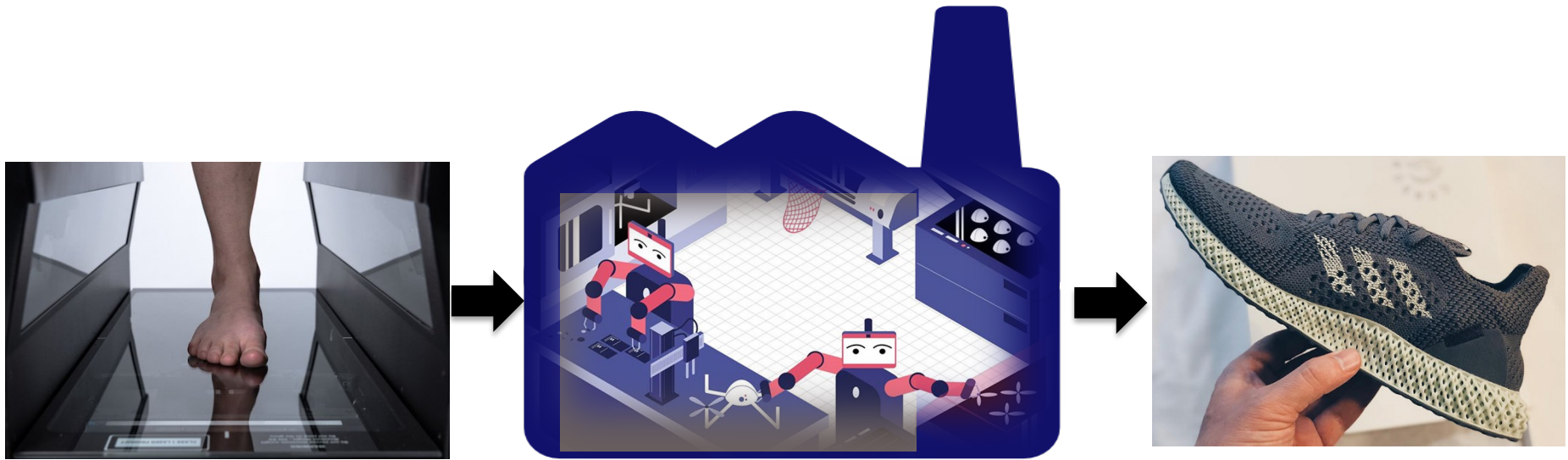
Manufacturing in Near Future



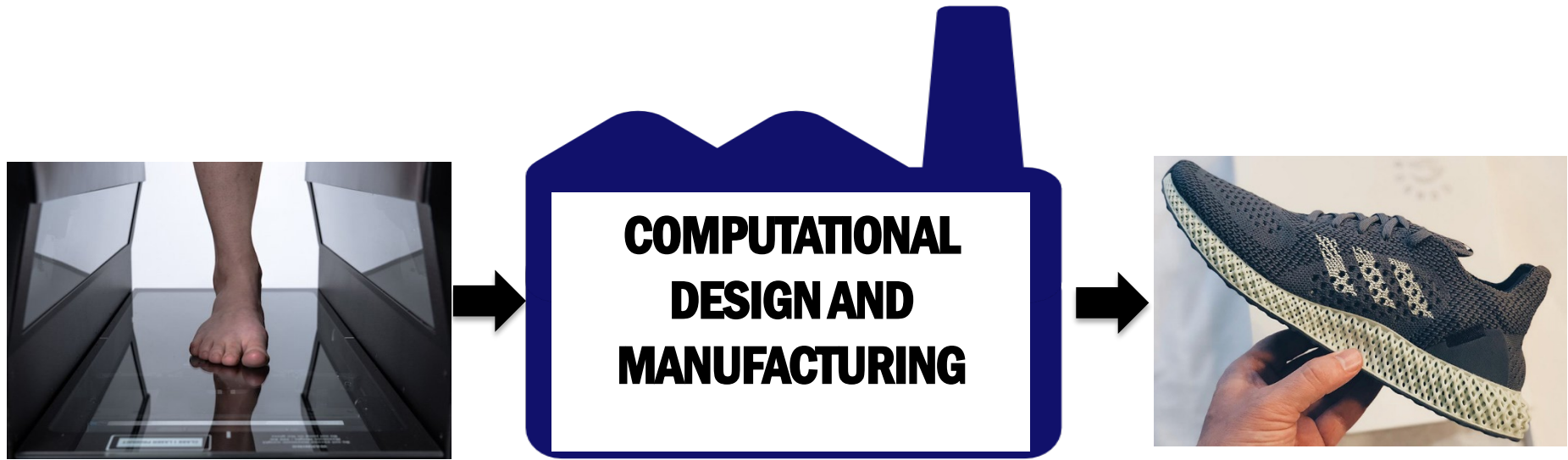
Manufacturing in Near Future



Why Computational Design and Manufacturing?



Why Computational Design and Manufacturing?



Hardware



Hardware

Hardware Abstraction & Machine Code

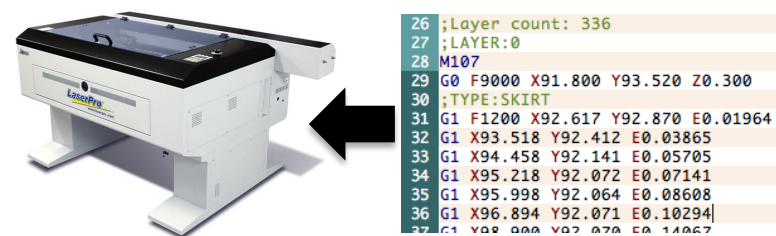


```
26 ;Layer count: 336
27 ;LAYER:0
28 M107
29 G0 F9000 X91.800 Y93.520 Z0.300
30 ;TYPE:SKIRT
31 G1 F1200 X92.617 Y92.870 E0.01964
32 G1 X93.518 Y92.412 E0.03865
33 G1 X94.458 Y92.141 E0.05705
34 G1 X95.218 Y92.072 E0.07141
35 G1 X95.998 Y92.064 E0.08608
36 G1 X96.894 Y92.071 E0.10294|
37 G1 X97.000 Y92.070 E0.11067
```

Hardware

Machine
Code

Hardware Abstraction & Machine Code



Hardware

Machine
Code

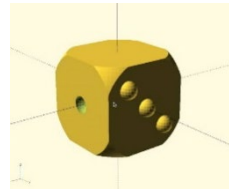
Design: Shape and Materials



Hardware

```
26 ;Layer count: 336
27 ;LAYER:0
28 M107
29 G0 F9000 X91.800 Y93.520 Z0.300
30 ;TYPE:SKIRT
31 G1 F1200 X92.617 Y92.870 E0.01964
32 G1 X93.518 Y92.412 E0.03865
33 G1 X94.458 Y92.141 E0.05705
34 G1 X95.218 Y92.072 E0.07141
35 G1 X95.998 Y92.064 E0.08608
36 G1 X96.894 Y92.071 E0.10294
37 G1 X97.800 Y92.070 E0.12067
```

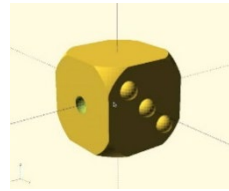
Machine
Code



```
1 difference(){
2   //cuerpo del dado
3   intersection(){
4     cube(20,center=true);
5     sphere(15,$fn=100);}
6   //cara del 1
7   translate([10,0,0])
8   sphere(2,$fn=20);
9 }
10 translate([0,10,0])
11 sphere(2,$fn=20);
12 translate([5,10,5])
13 sphere(2,$fn=20);
14 translate([-5,10,-5])
15 sphere(2,$fn=20);
```

Design

From Design to Machine Code



```
1 difference(){
2   //cuerpo del dado
3   intersection(){
4     cube(20,center=true);
5     sphere(15,$fn=100);}
6   //cara del 1
7   translate([10,0,0])
8   sphere(2,$fn=20);
9 }
10 translate([0,10,0])
11 sphere(2,$fn=20);
12 translate([5,10,5])
13 sphere(2,$fn=20);
14 translate([-5,10,-5])
15 sphere(2,$fn=20);
```



```
26 ;Layer count: 336
27 ;LAYER:0
28 M107
29 G0 F9000 X91.800 Y93.520 Z0.300
30 ;TYPE:SKIRT
31 G1 F1200 X92.617 Y92.870 E0.01964
32 G1 X93.518 Y92.412 E0.03865
33 G1 X94.458 Y92.141 E0.05705
34 G1 X95.218 Y92.072 E0.07141
35 G1 X95.998 Y92.064 E0.08608
36 G1 X96.894 Y92.071 E0.10294
37 G1 X97.800 Y92.070 E0.12067
```

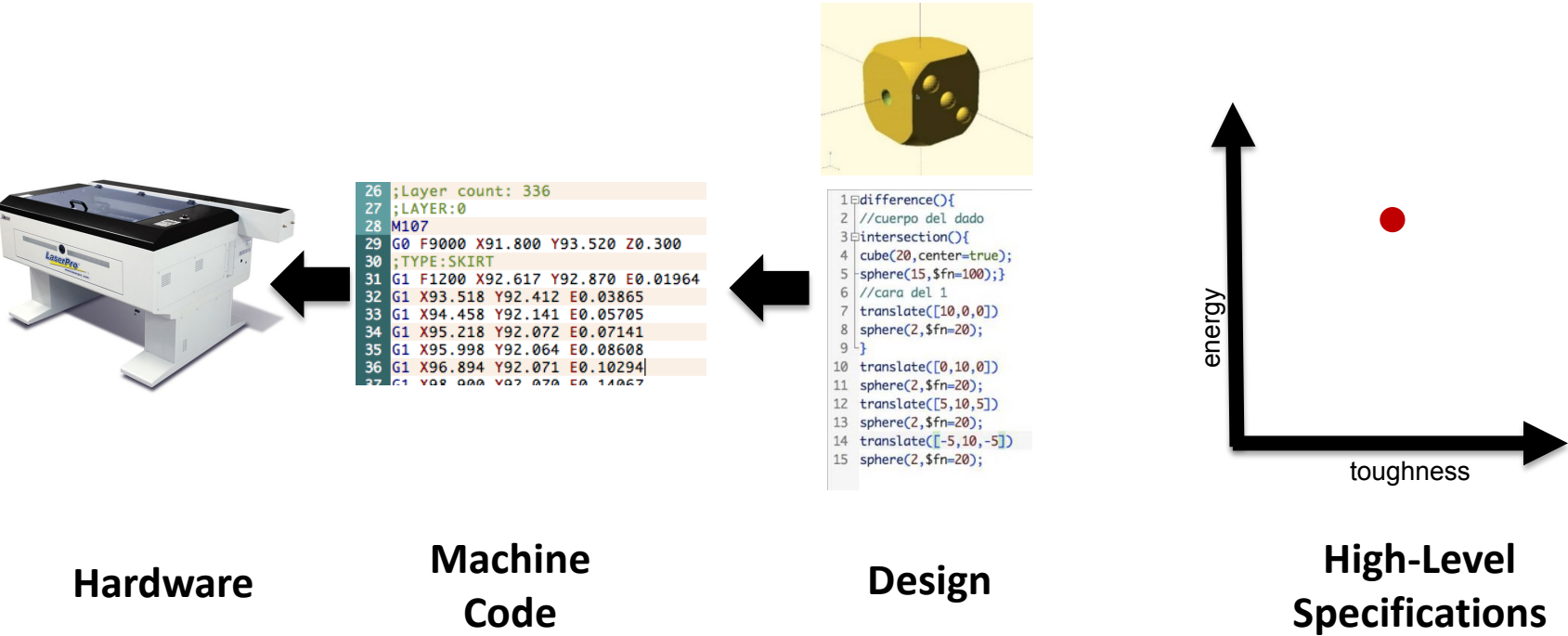


Hardware

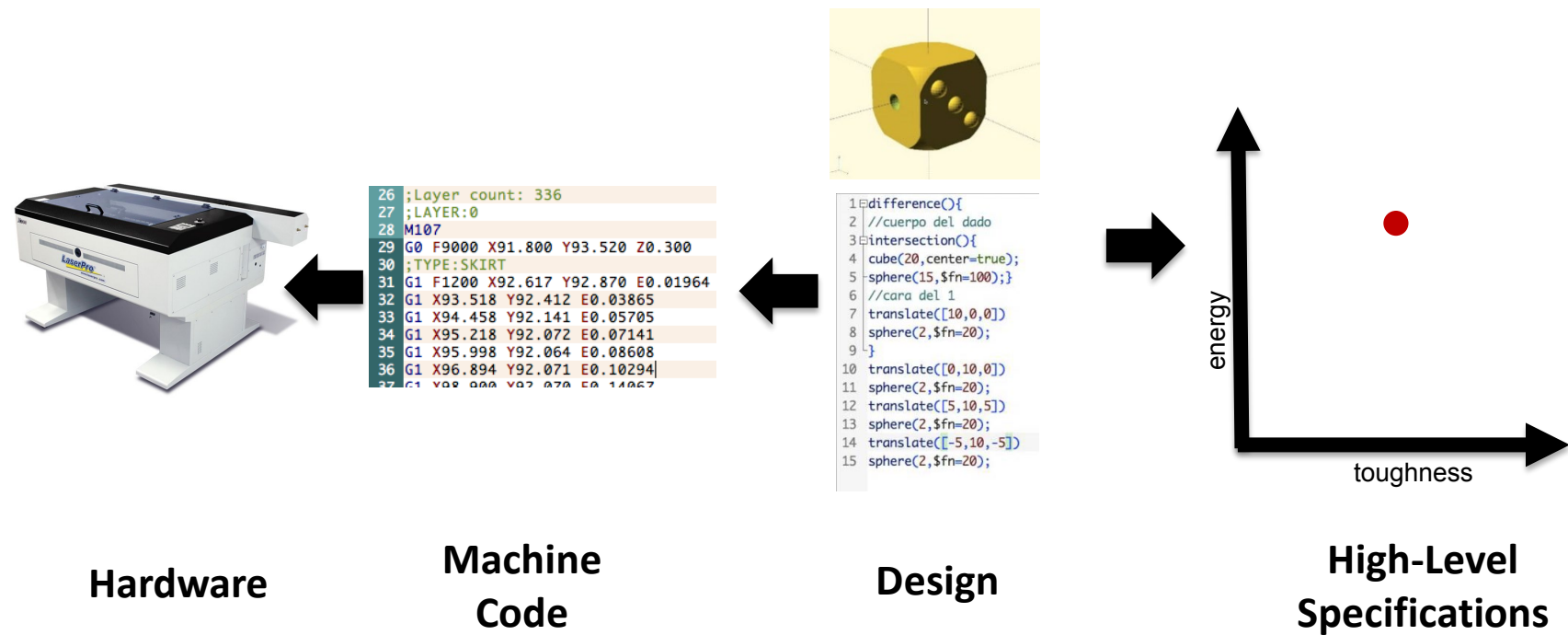
**Machine
Code**

Design

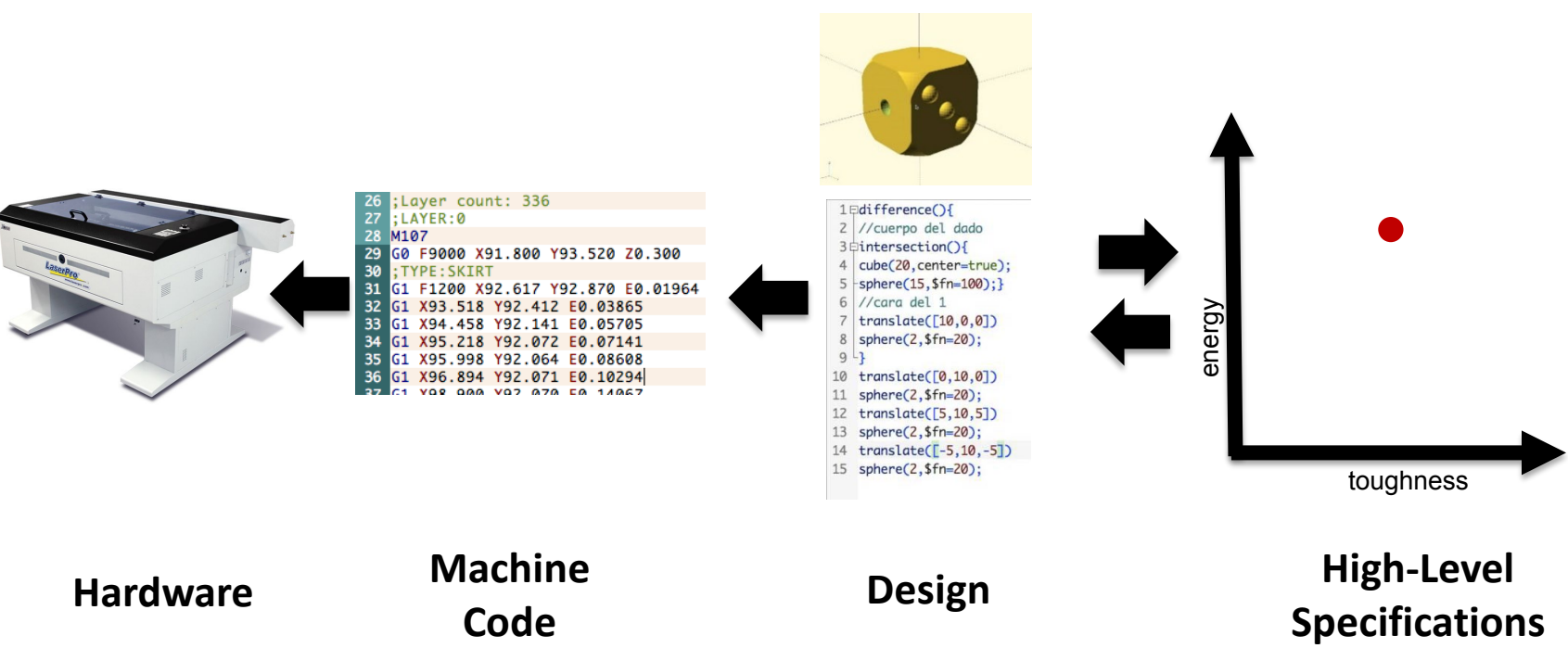
High-level Specifications/Performance



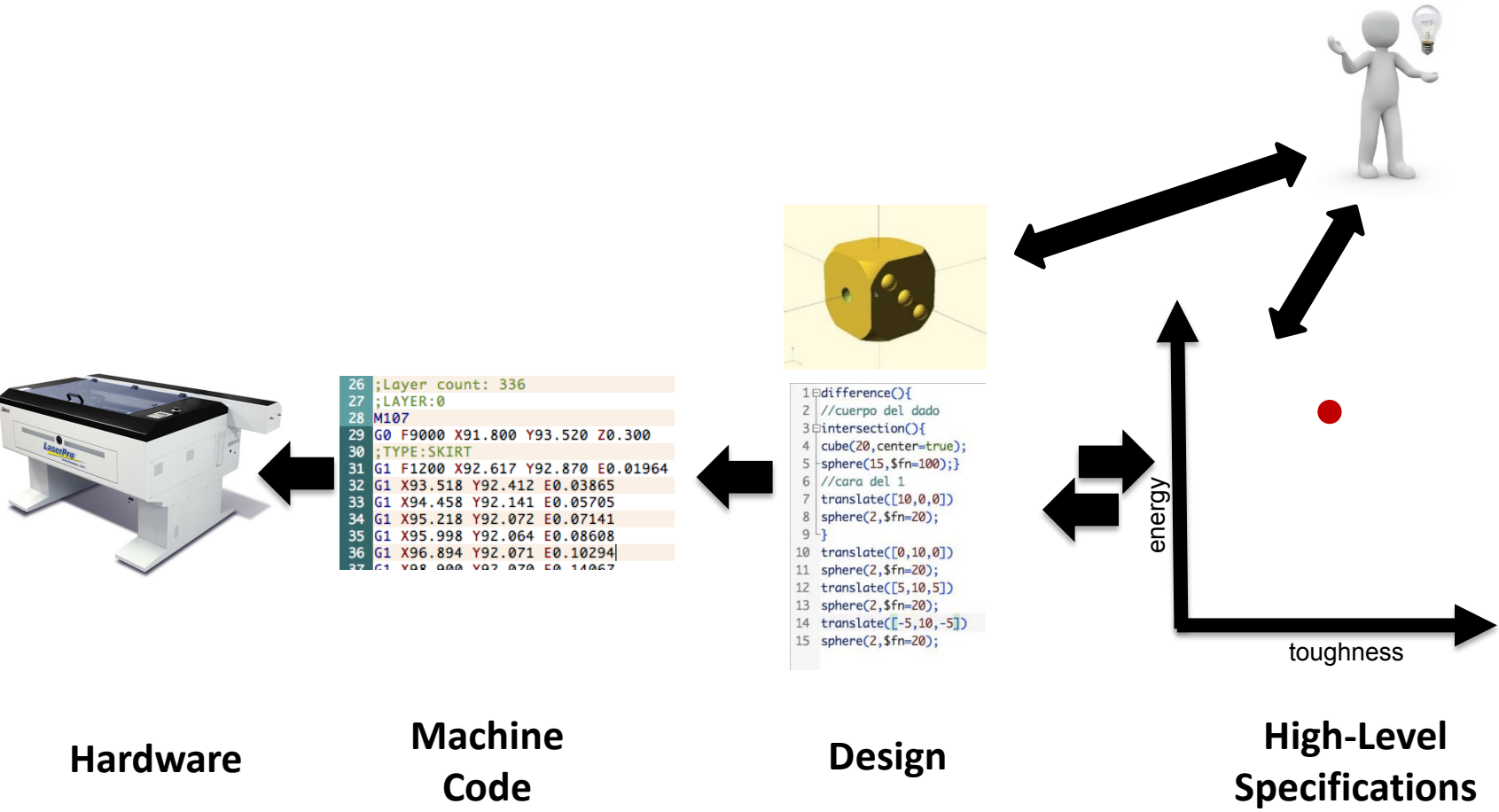
From Design to Performance



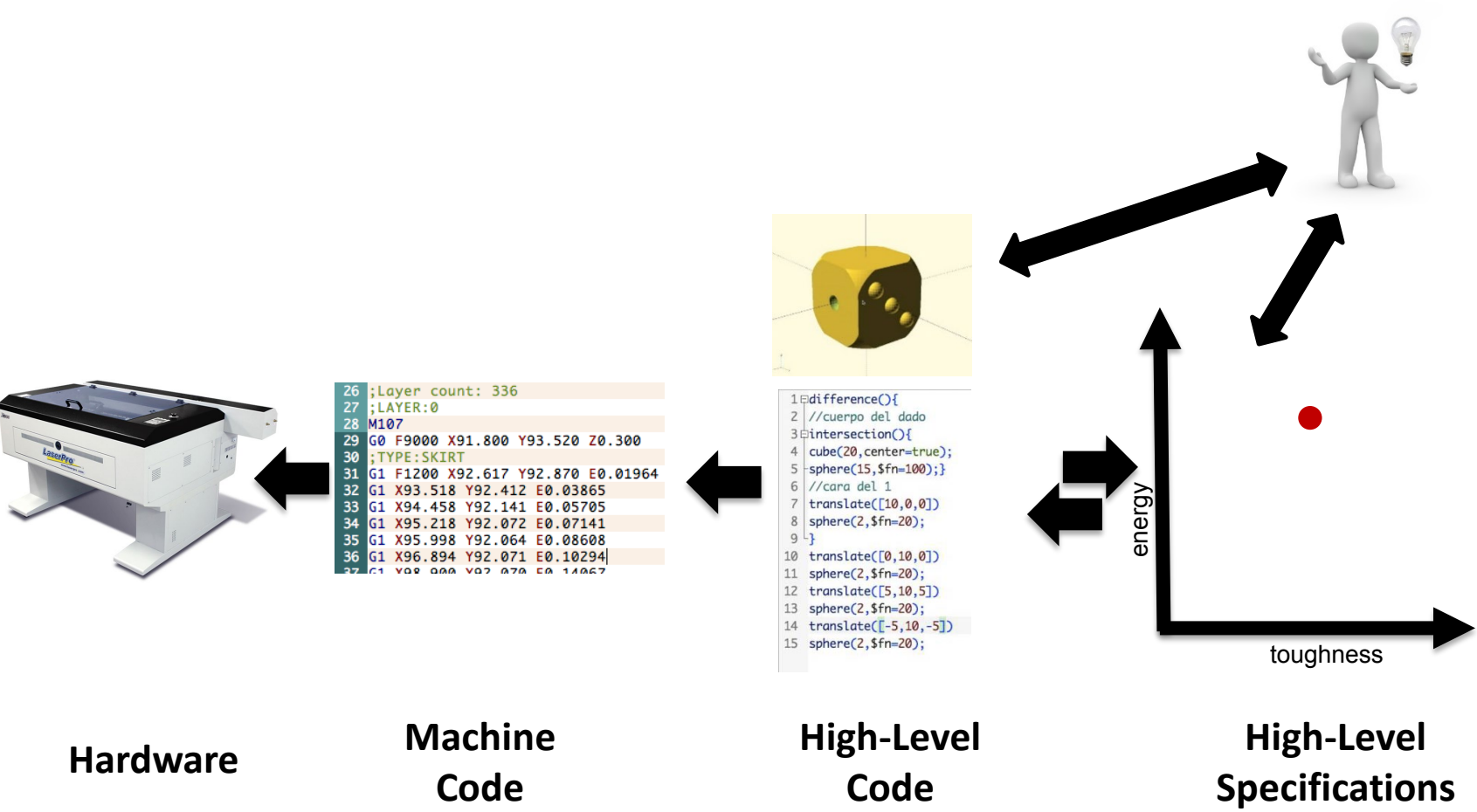
From Performance to Design



Human Computer Interaction (HCI)



Artificial Intelligence (AI) & Machine Learning



Operating Systems for Future Factory



```
26 ;Layer count: 336
27 ;LAYER:0
28 M107
29 G0 F9000 X91.800 Y93.520 Z0.300
30 ;TYPE:SKIRT
31 G1 F1200 X92.617 Y92.870 E0.01964
32 G1 X93.518 Y92.412 E0.03865
33 G1 X94.458 Y92.141 E0.05705
34 G1 X95.218 Y92.072 E0.07141
35 G1 X95.998 Y92.064 E0.08608
36 G1 X96.894 Y92.071 E0.10294
37 G1 X97.800 Y92.070 E0.11967
```

```
26 ;Layer count: 336
27 ;LAYER:0
28 M107
29 G0 F9000 X91.800 Y93.520 Z0.300
30 ;TYPE:SKIRT
31 G1 F1200 X92.617 Y92.870 E0.01964
32 G1 X93.518 Y92.412 E0.03865
33 G1 X94.458 Y92.141 E0.05705
34 G1 X95.218 Y92.072 E0.07141
35 G1 X95.998 Y92.064 E0.08608
36 G1 X96.894 Y92.071 E0.10294
37 G1 X97.800 Y92.070 E0.11967
```

- Multi-tasking
- Distributed systems

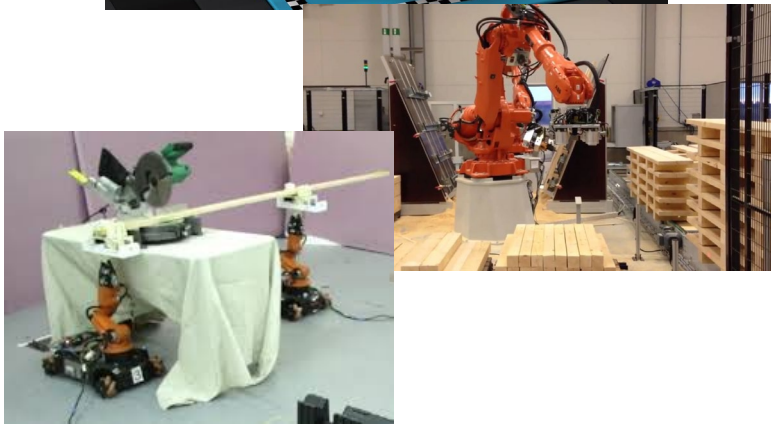
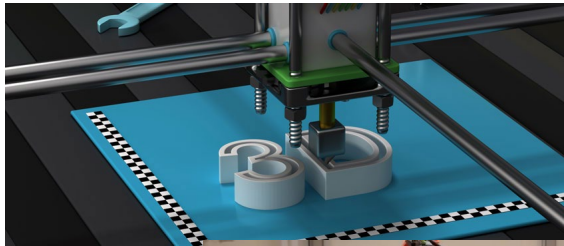


Who's working on this?

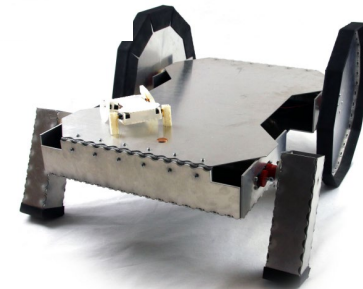


Highly interdisciplinary field!

Robotics



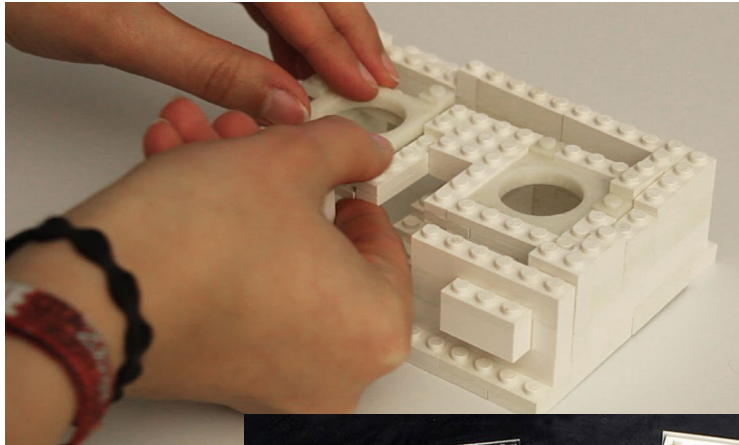
Fabrication Hardware



Design of Cyber-physical
Systems



Human-Computer Interaction

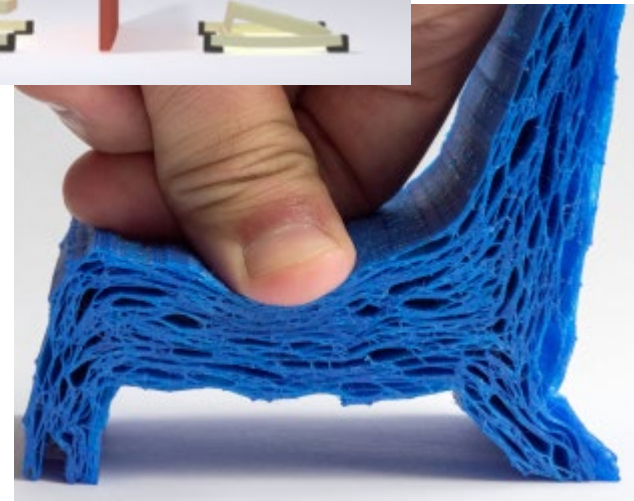


Hardware interfaces

Design Systems



Computer Graphics

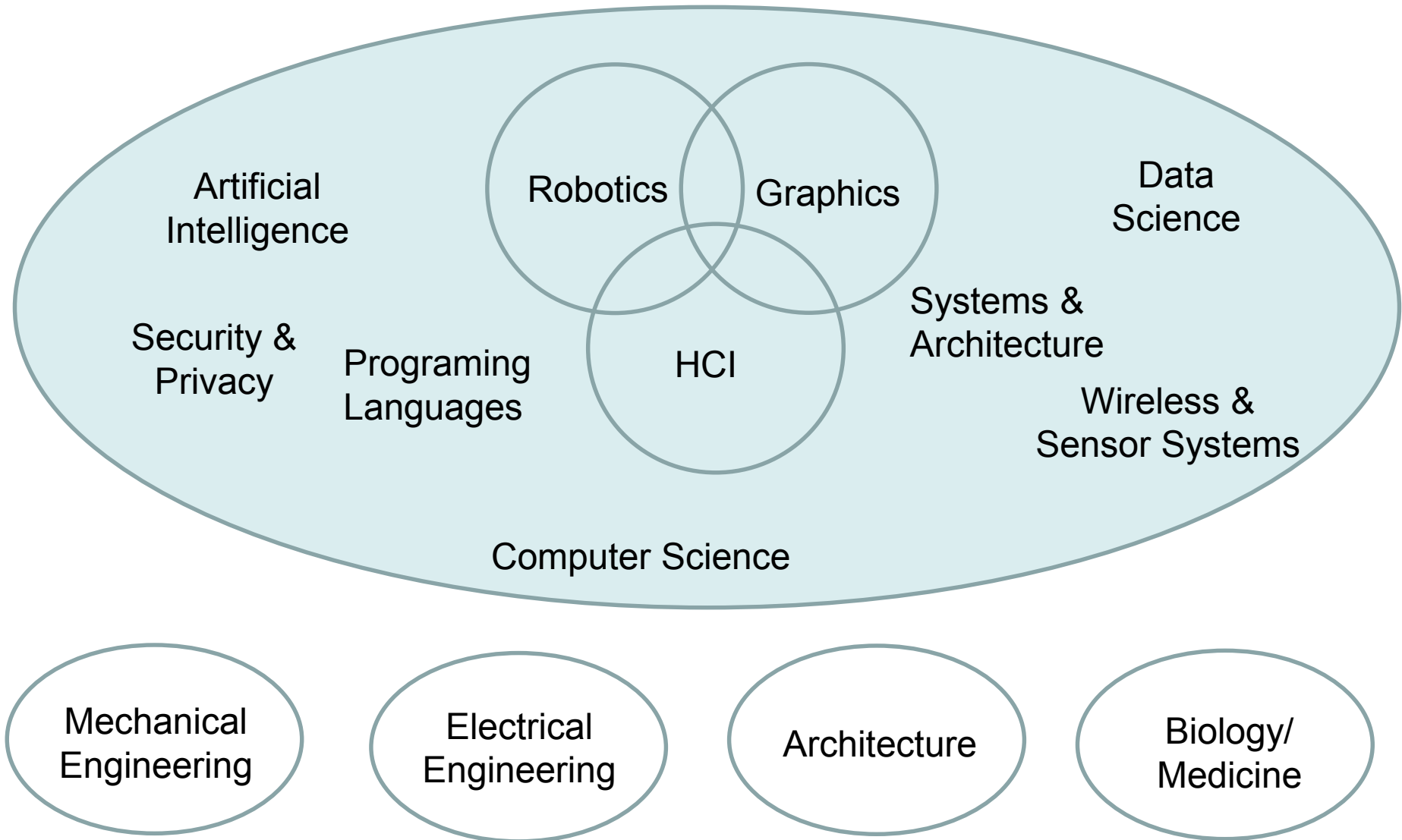


Geometry Processing

Performance-Driven Design

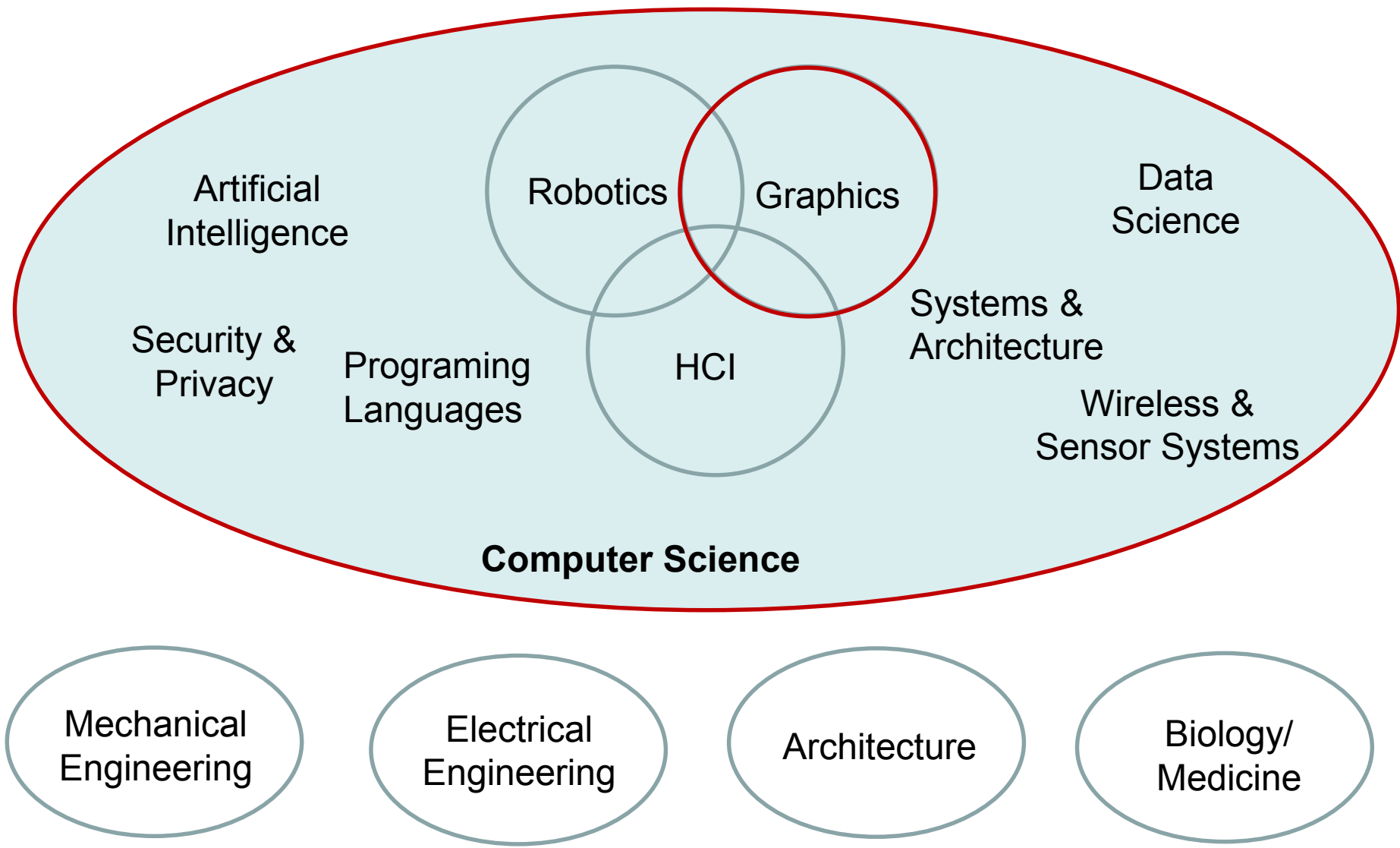


Active Research Areas





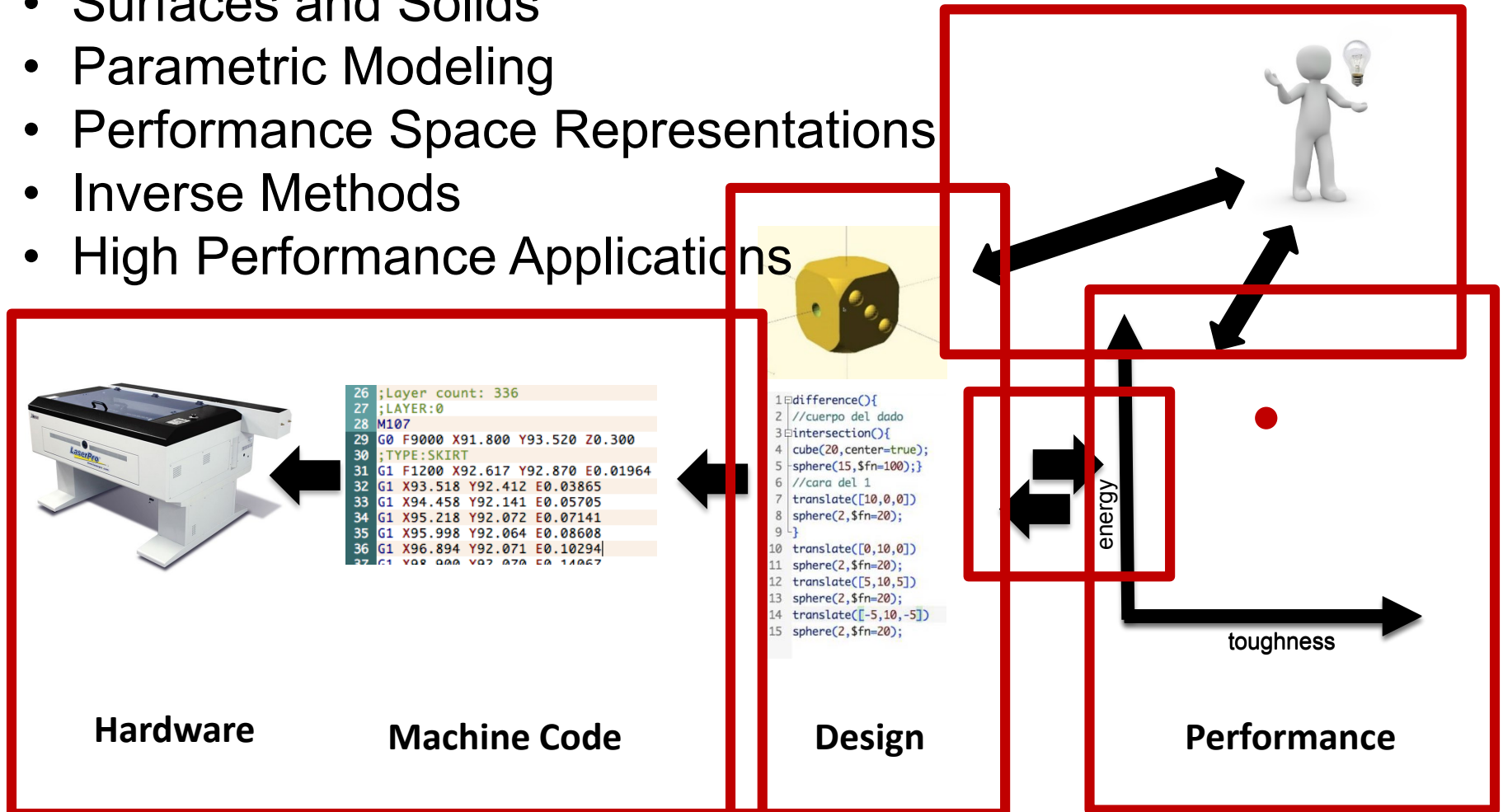
Our Perspective





What Will You Learn?

- Hardware and its abstractions
- Surfaces and Solids
- Parametric Modeling
- Performance Space Representations
- Inverse Methods
- High Performance Applications



The Team

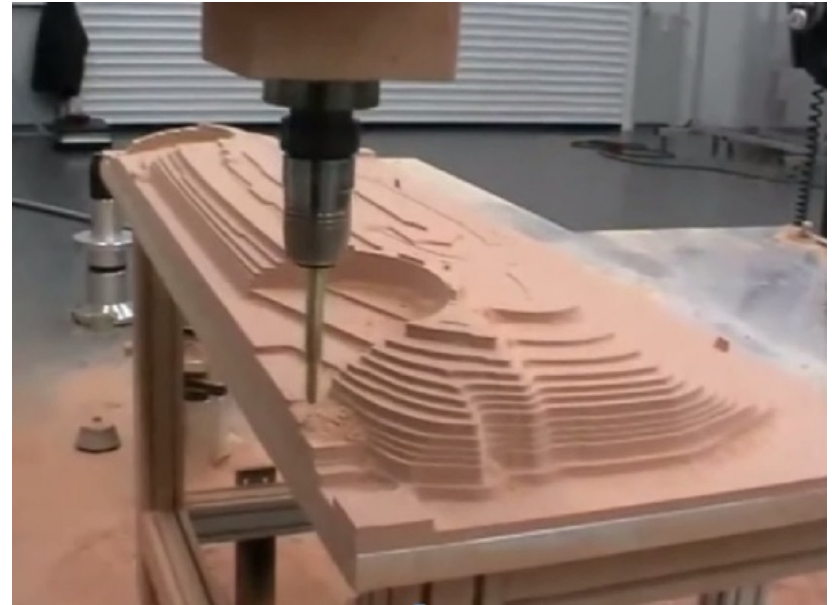
- Instructors
 - Wojciech Matusik
 - Adriana Schulz



Hardware Overview

Subtractive Manufacturing

- Start with a block of material
- Remove material to obtain a given shape

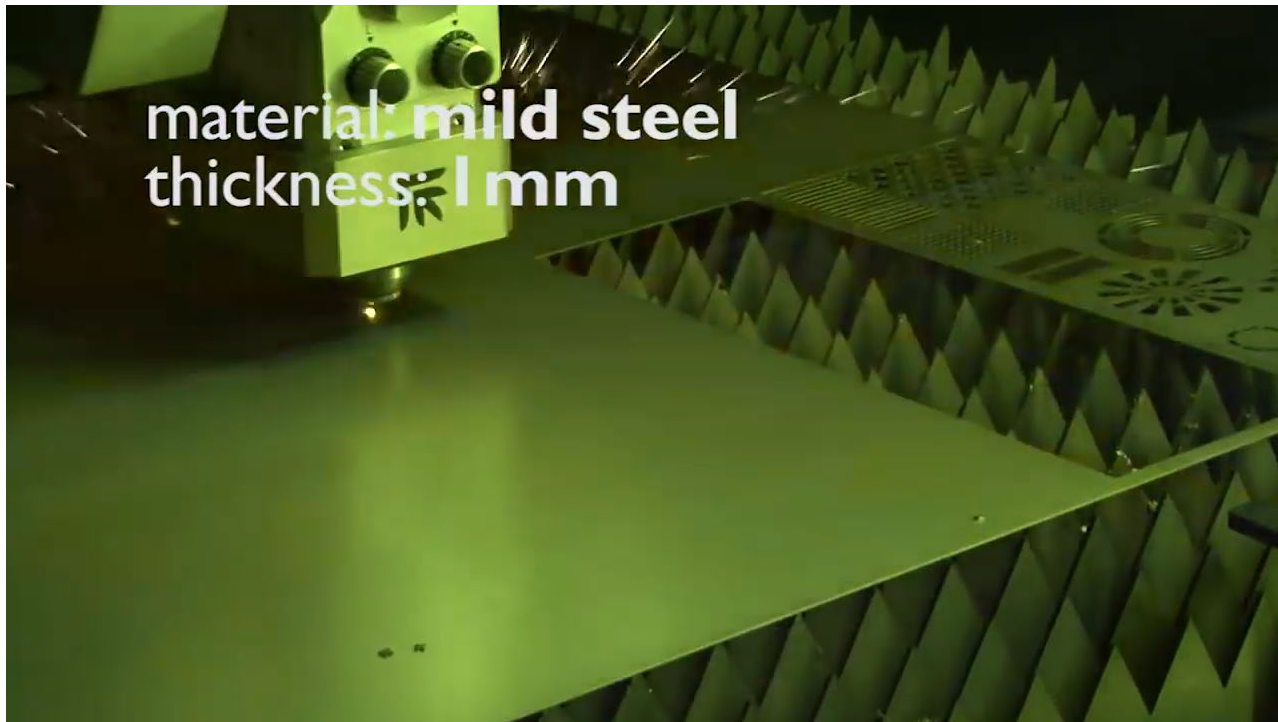


Source:

<https://commons.wikimedia.org>

Computer Numerical Control (CNC) Cutting

- Many different methods (e.g, laser cutting, water jetting)
- Input: 2D vector format
- Output: G-code



3-Axis CNC Milling/Engraving

- Input: 2.5D (heightfield)
- Output: G-code, multiple passes



5-Axis CNC Milling

- Input: 3D Mesh
- Output: G-code, multiple passes



Whole-garment Knitting



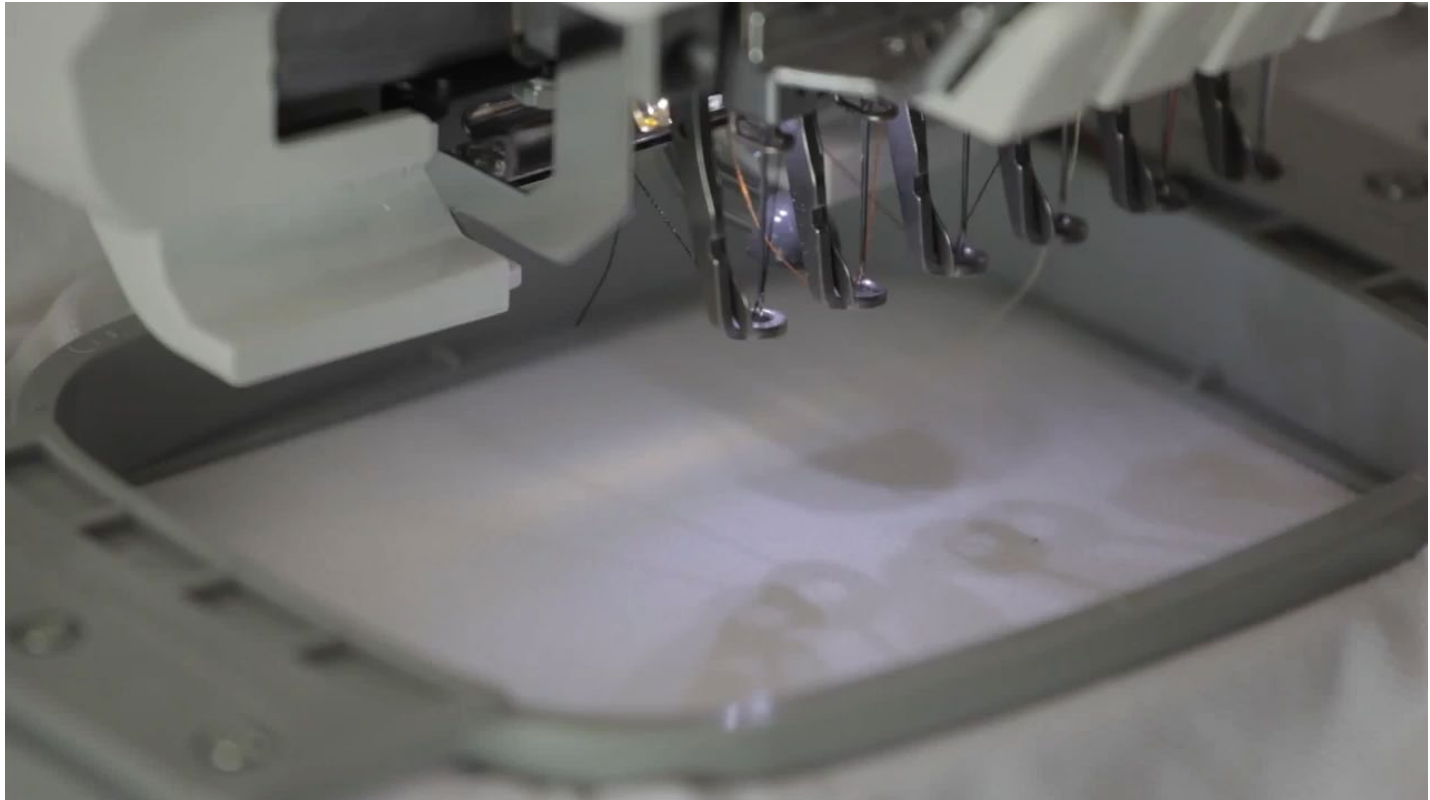
Shima/Ministry of Supply

Whole-garment Knitting



Computerized Embroidery

- Input: 2D vector graphics/image
- Output: proprietary format



3D Printing Basics

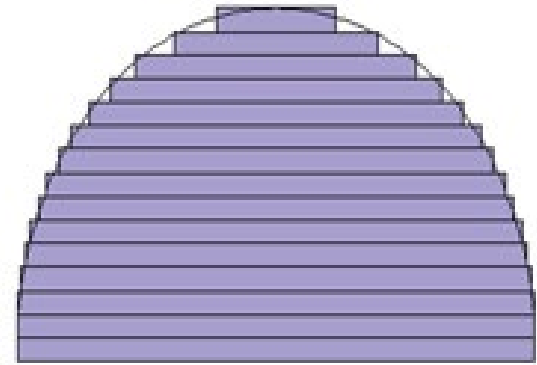
3D Printing = Additive Manufacturing



https://commons.wikimedia.org/wiki/File:3D_printing_on_replicator_2.webm

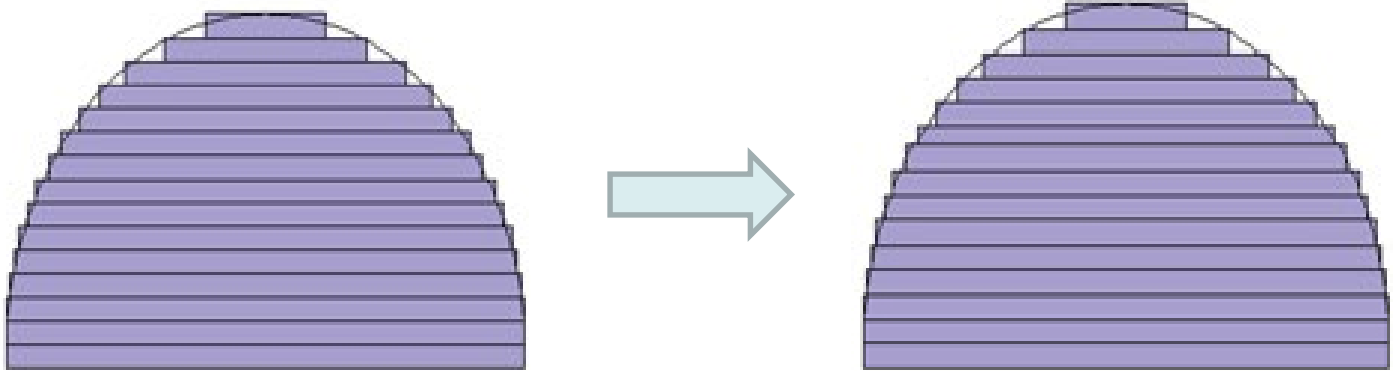
3D Printing Process

- Slice 3D model into layers



3D Printing Process

- Slice 3D model into layers
- Manufacture layers one by one (e.g., bottom-up)



Source:

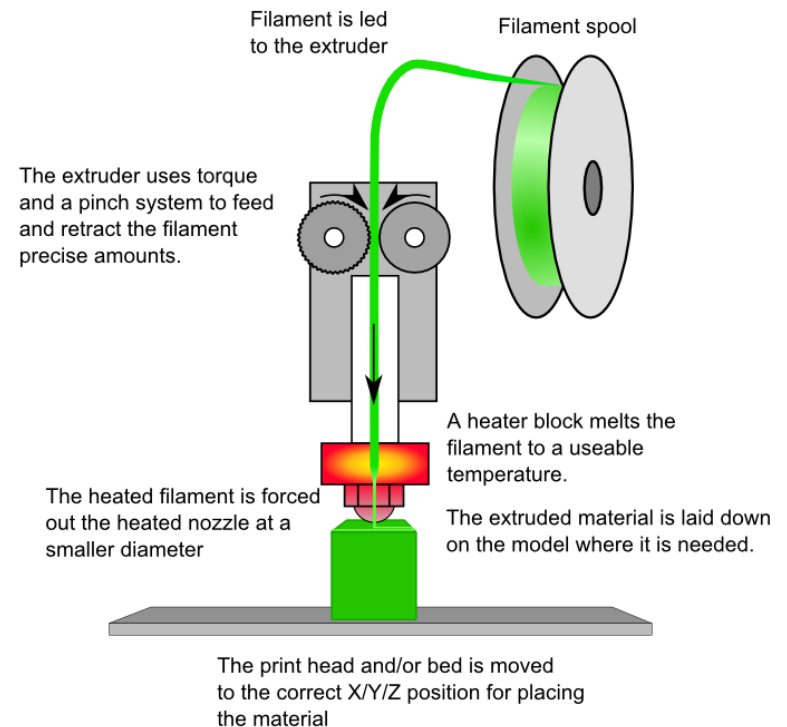
<https://commons.wikimedia.org>

Additive Manufacturing Processes

- Fused deposition modeling (FDM)
- Stereolithography (SLA)
- DLP 3D printing
- Selective laser sintering (SLS)
- Direct metal laser sintering (DMLS)
- Plaster-based 3D printing (PP)
- Photopolymer Phase Change Inkjets
- Thermal Phase Change Inkjets
- Laminated object manufacturing (LOM)

Fused Deposition Modeling (FDM)

- Fused Filament Fabrication (FFF)
- Filament is made of thermoplastic materials
 - e..g., ABS, polycarbonate, PLA
- Temporary support structure can be made from water-soluble material such as PVA
 - removed using heated sodium hydroxide solution



Source: <http://reprap.org>

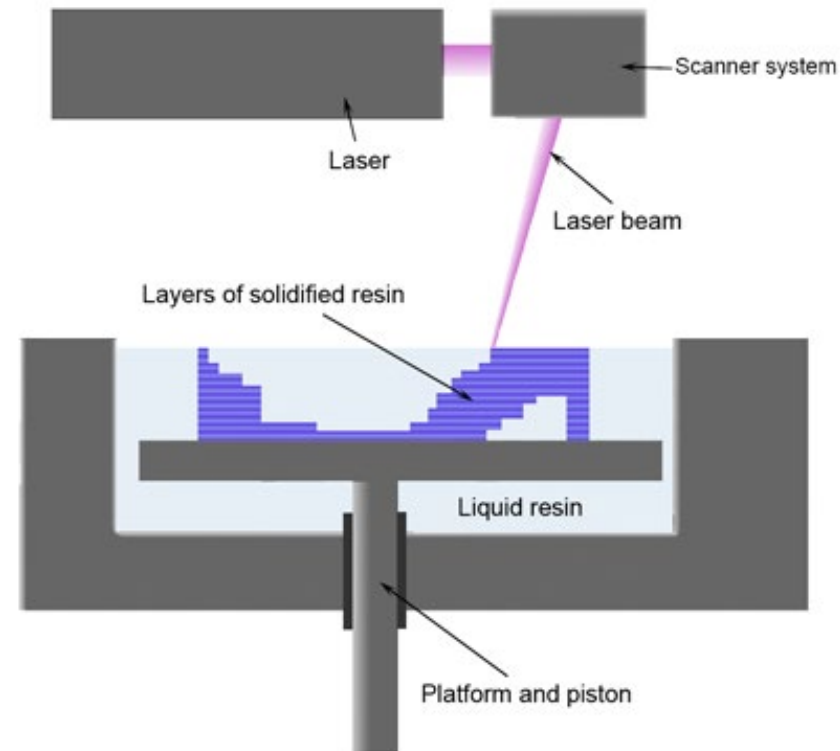
FDM with Carbon Fiber



Source: Markforged

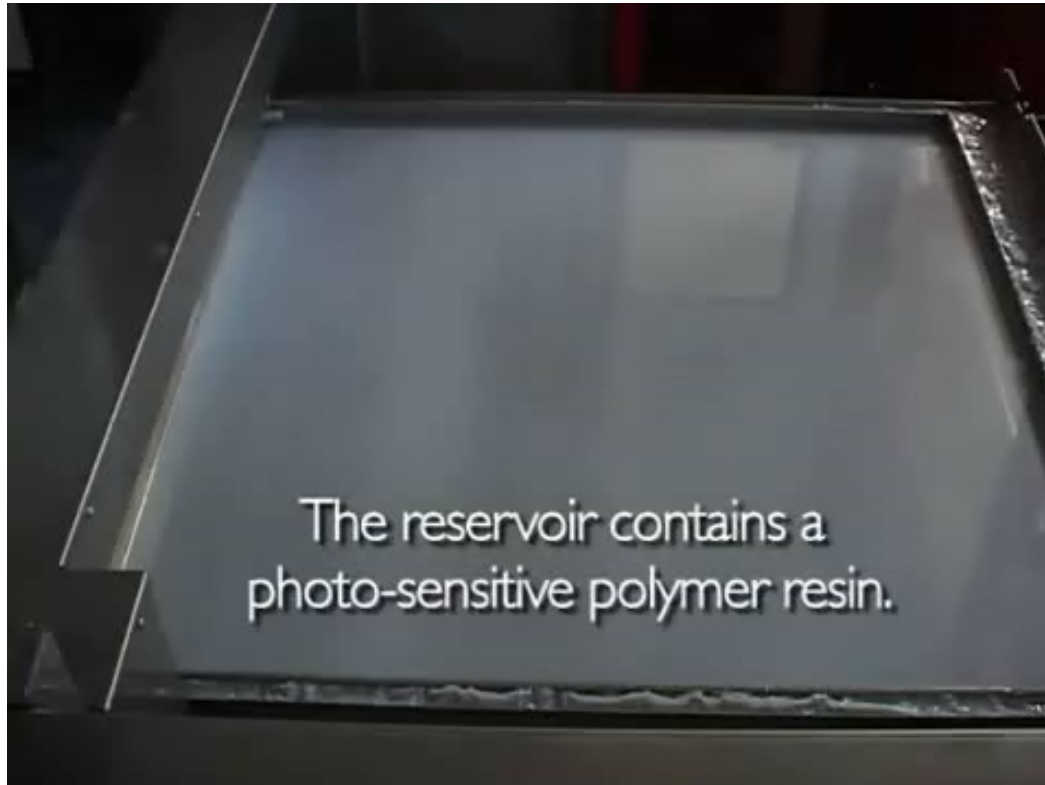
Stereolithography (SLA)

- SLA uses liquid ultraviolet curable photopolymer resin
- Laser beam traces one layer on the surface of the resin
- Laser light cures and solidifies the layer



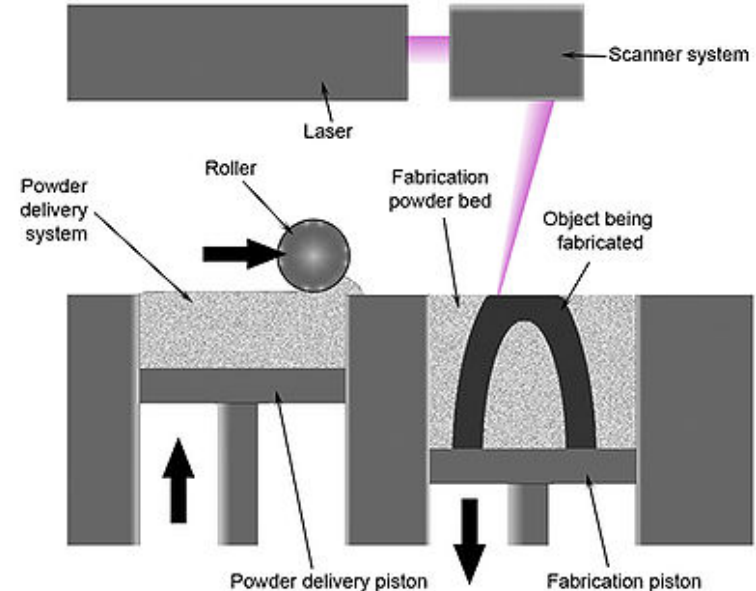
Source: <http://en.wikipedia.org/>

Stereolithography Process



Selective Laser Sintering (SLS) Direct Metal Laser Sintering (DMLS)

- SLS and DMLS use a bed of small particles (made of plastic, metal, ceramic, or glass)
- High-power laser traces one layer on the surface of the powder bed melting/fusing the particles



Source:

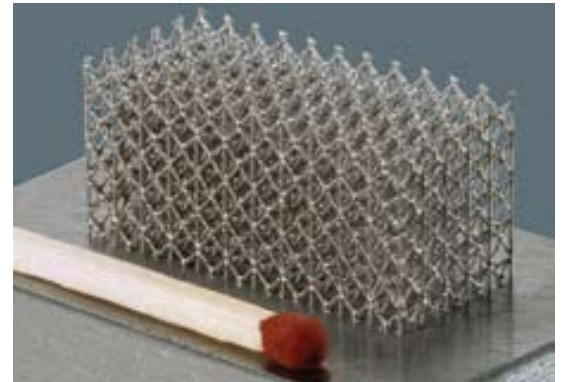
http://en.wikipedia.org/wiki/Selective_laser_sintering

SLS & DMLS Process



Source:
<https://www.youtube.com/watch?v=BZLGLzyMKn4>

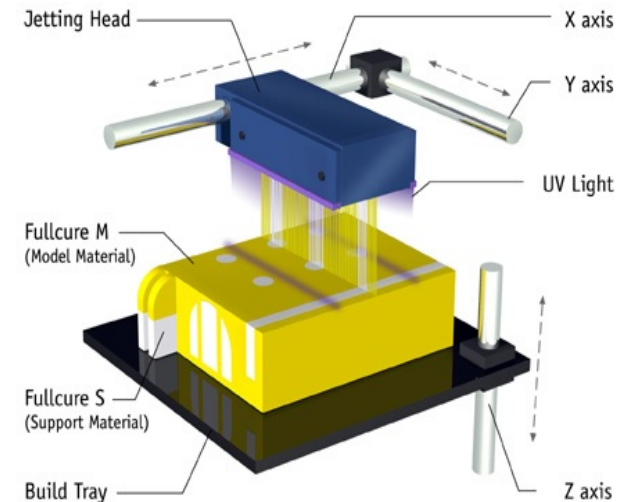
Sample Fabricated Parts



Sources: <http://www.bridgesmathart.org> ,
<http://www.freedomofcreation.com>

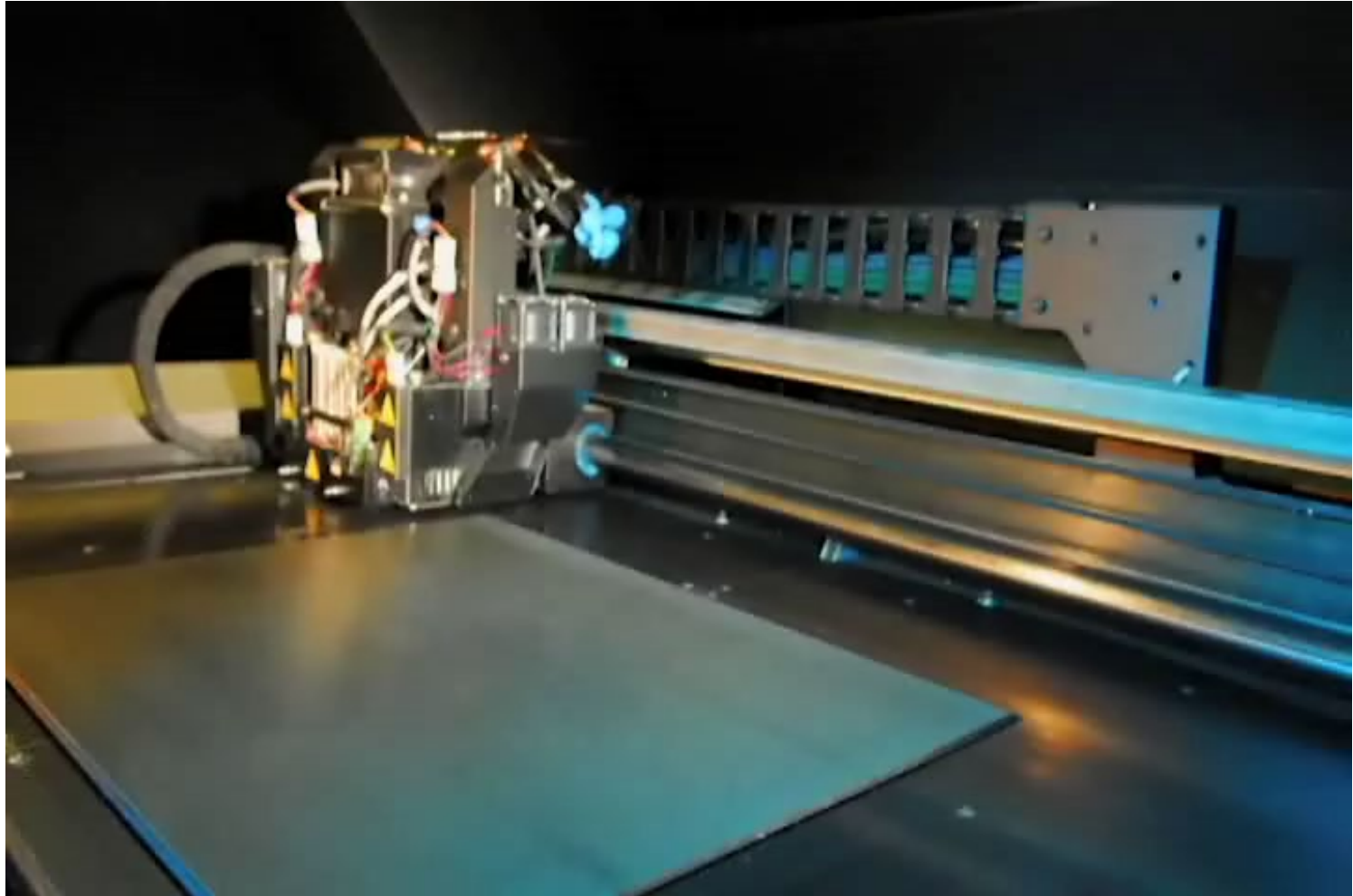
Photopolymer Phase Change Inkjets

- Inkjet printhead jets liquid photopolymer and support material
- UV light cures photopolymer and support material
- Excess material is removed using a roller
- The platform descends by one layer



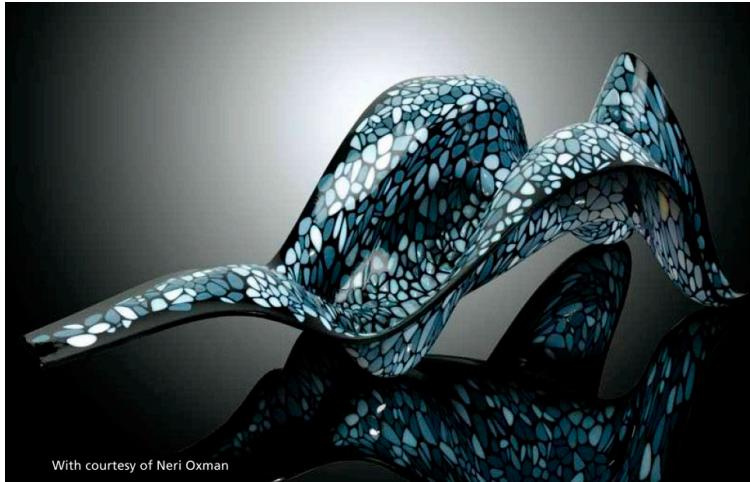
Source: <http://www.engatech.com/Objet-3D-Printing-Technology.asp>

Printing Process



<https://www.youtube.com/watch?v=XLLq9SwSTpM>

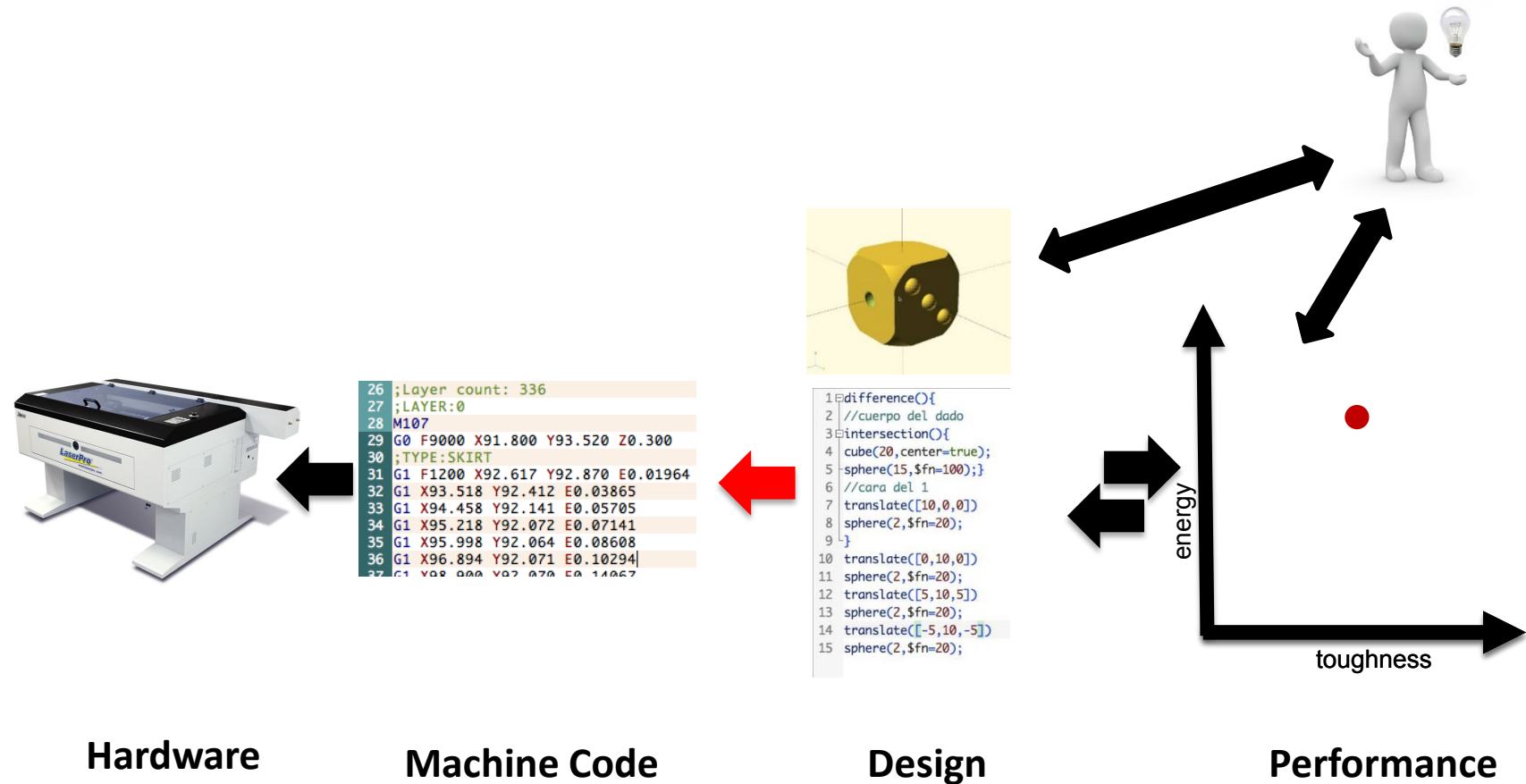
Sample Fabricated Objects



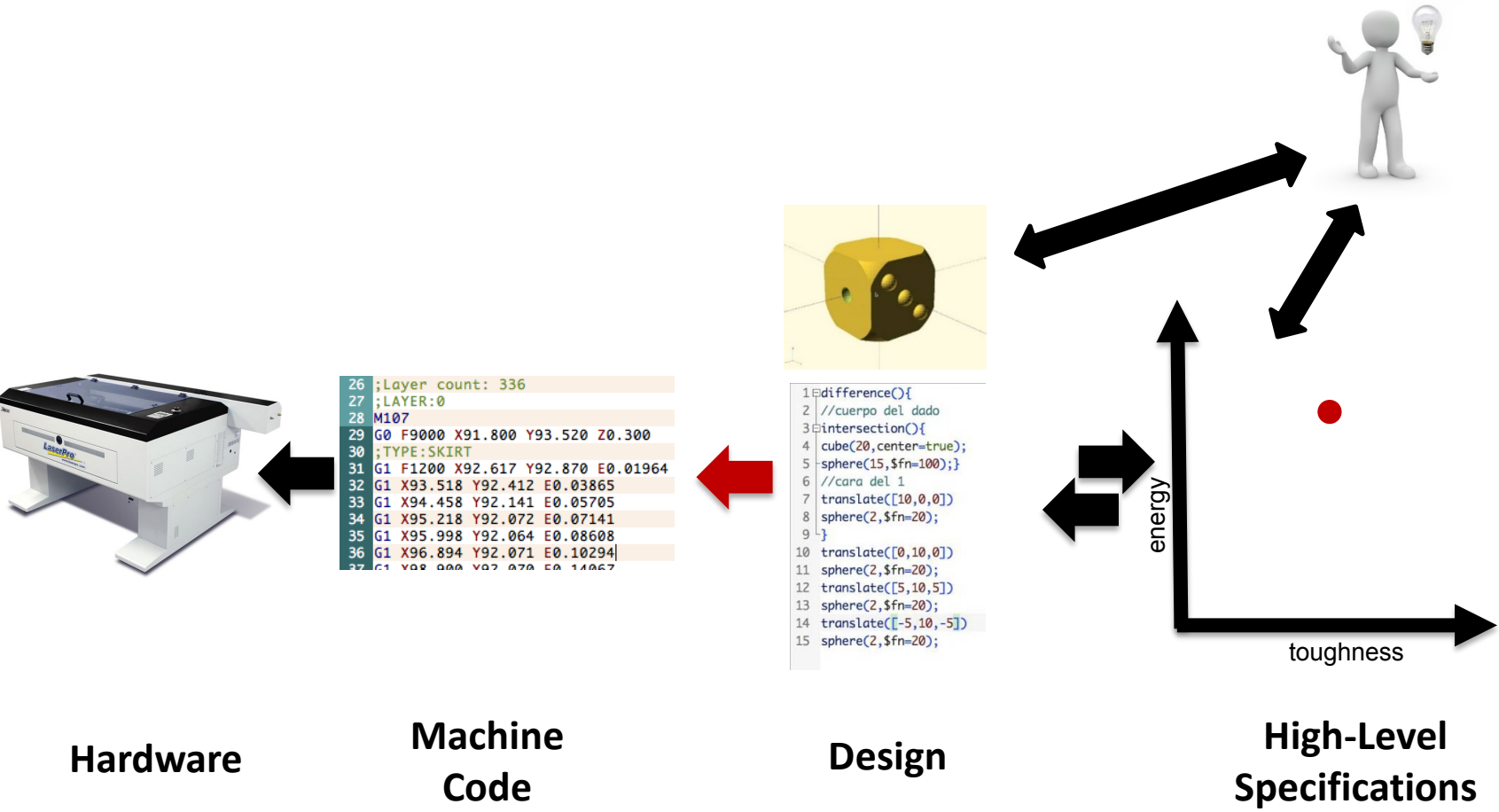
Source: Objet Geometries

From Design to Machine Code

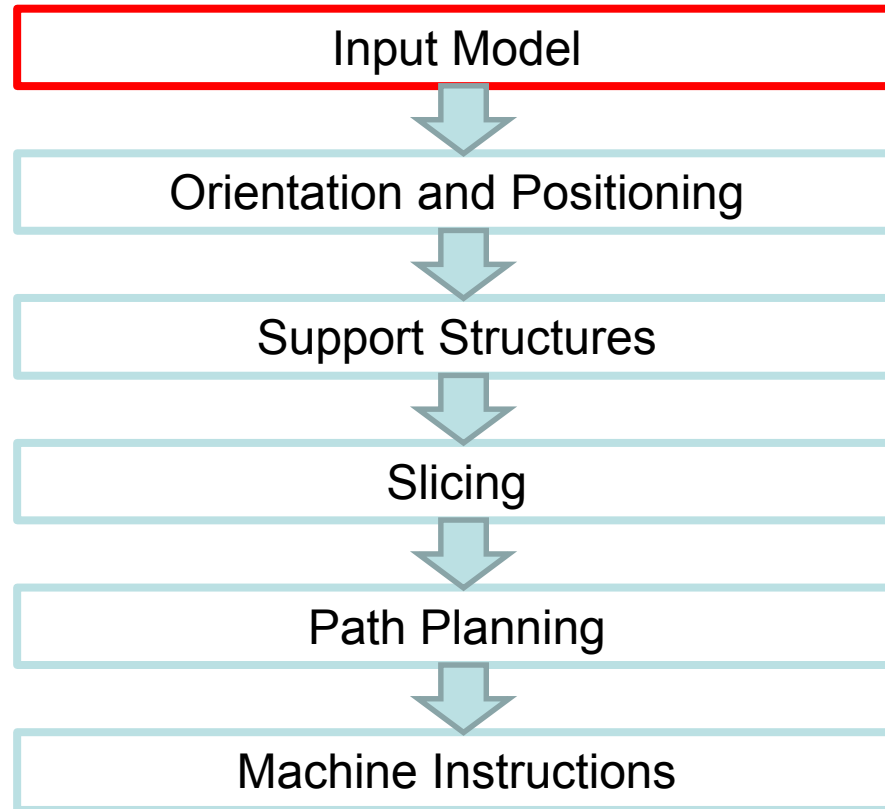
Computational Design and Fabrication Pipeline



Computational Design and Fabrication Pipeline

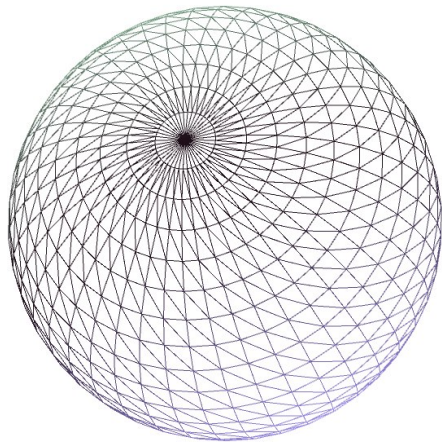


3D Printing Software Pipeline



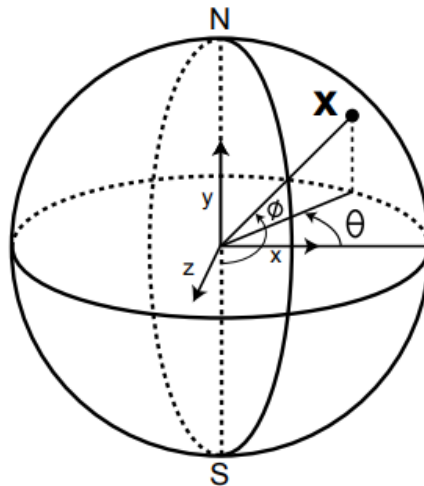
Surface Representations

Polygonal Mesh



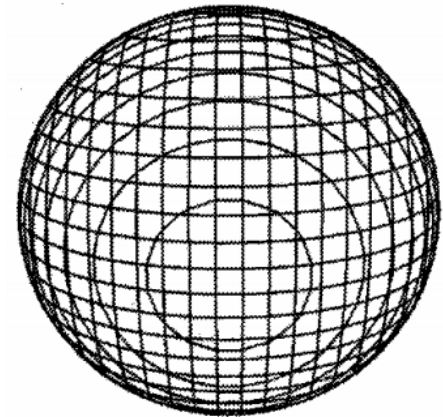
e.g. triangle soup

Parametric Surface



$$\begin{aligned}x &= r \sin \phi \cos \theta, \\y &= -r \cos \phi, \\z &= -r \sin \phi \sin \theta,\end{aligned}$$

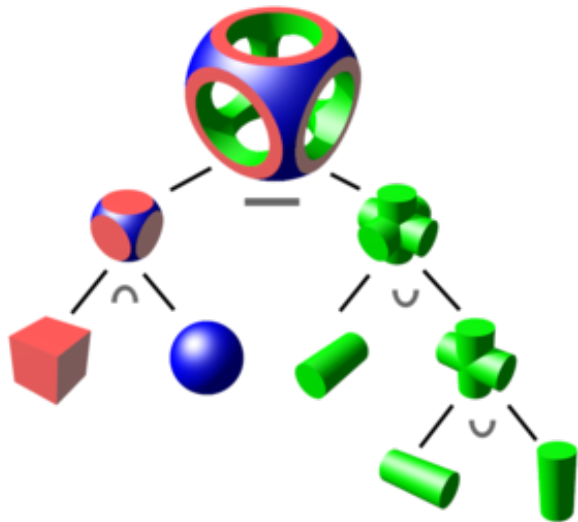
Implicit Surface



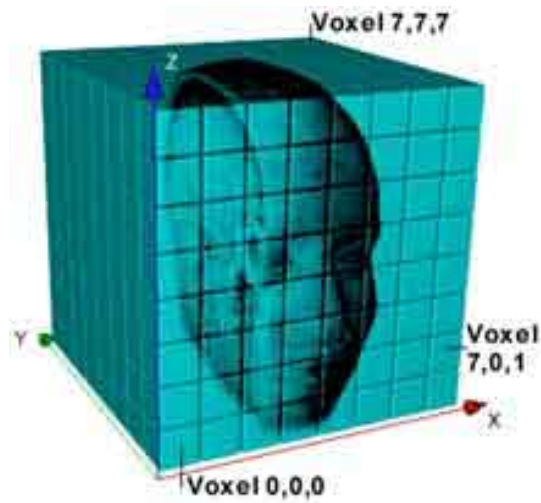
$$x^2 + y^2 + z^2 - r^2 = 0.$$

Volumetric Representations

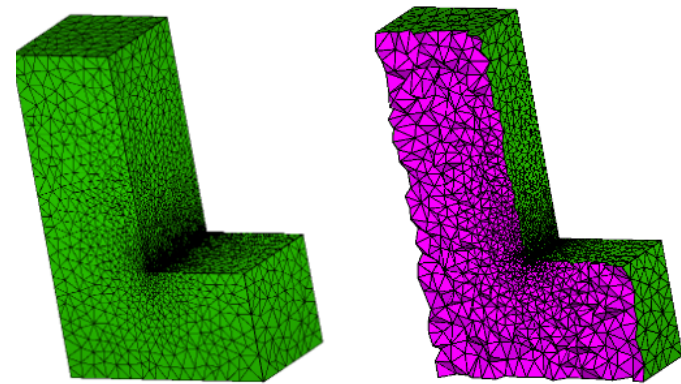
Constructive Solid Geometry



Voxel Grid

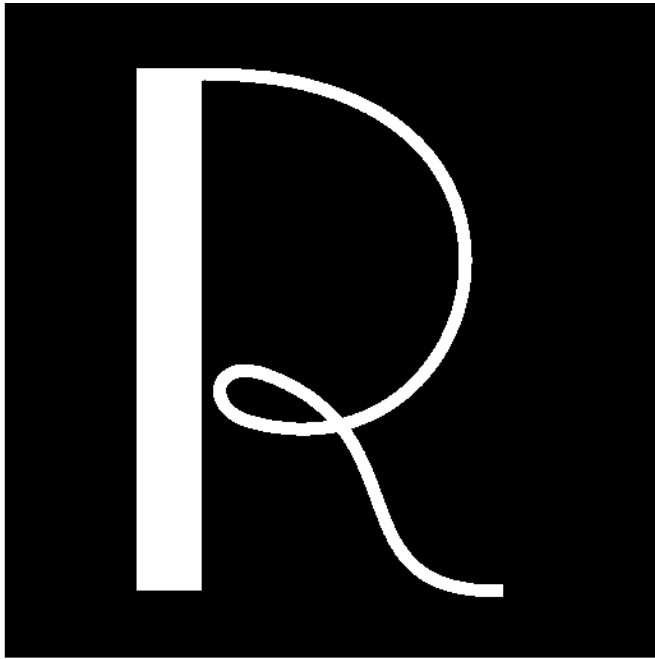


Tetrahedral Mesh

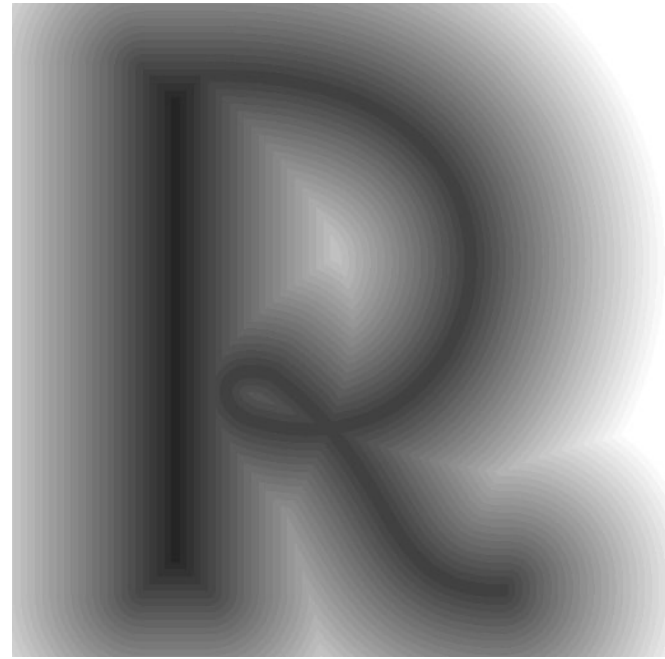


2D Euclidean Distance Field Example

- An object's distance field represents, for any point in space, the distance from that point to the object
 - The distance can be signed to distinguish between the inside and outside of the object

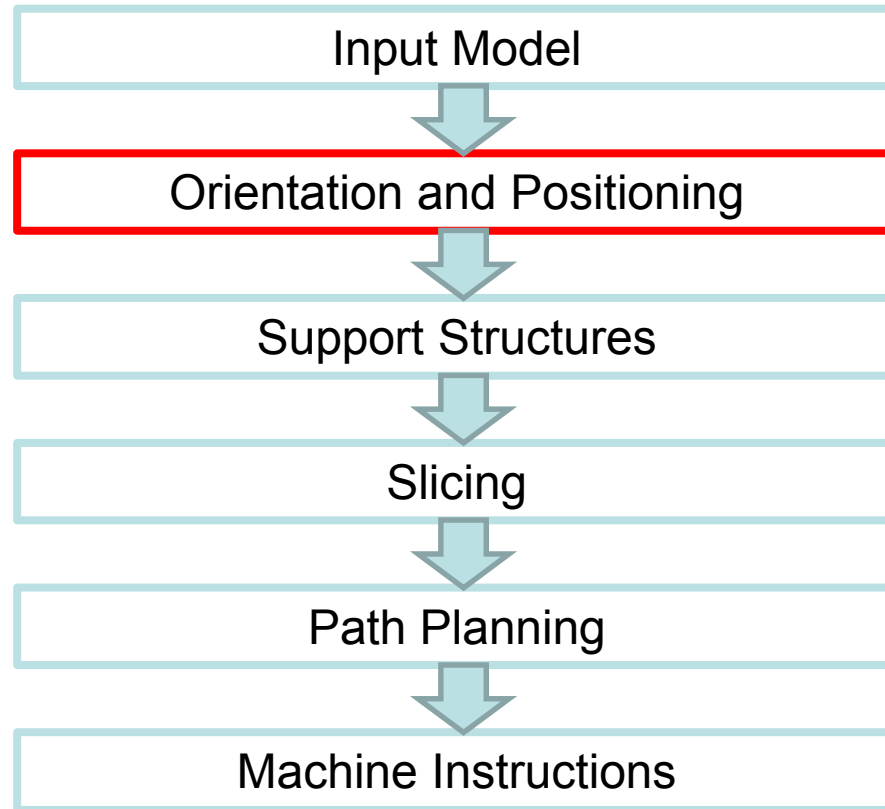


R shape



Distance field of R

3D Printing Software Pipeline



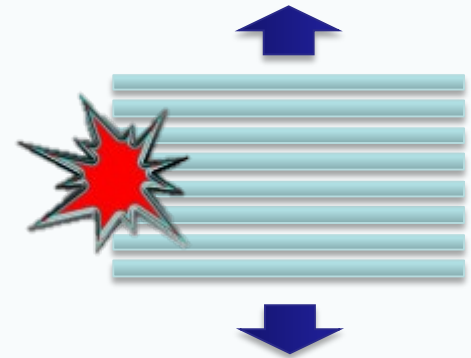
Model Orientation: Mechanical Properties



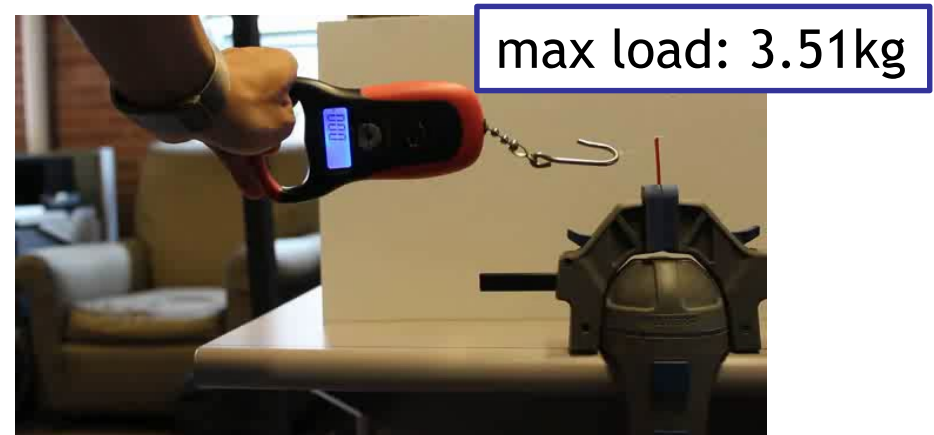
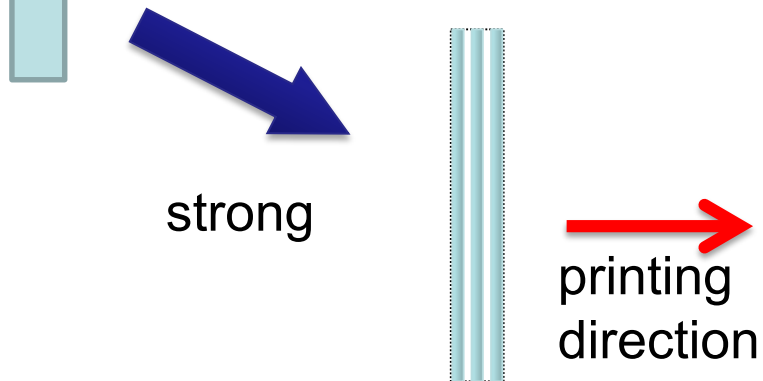
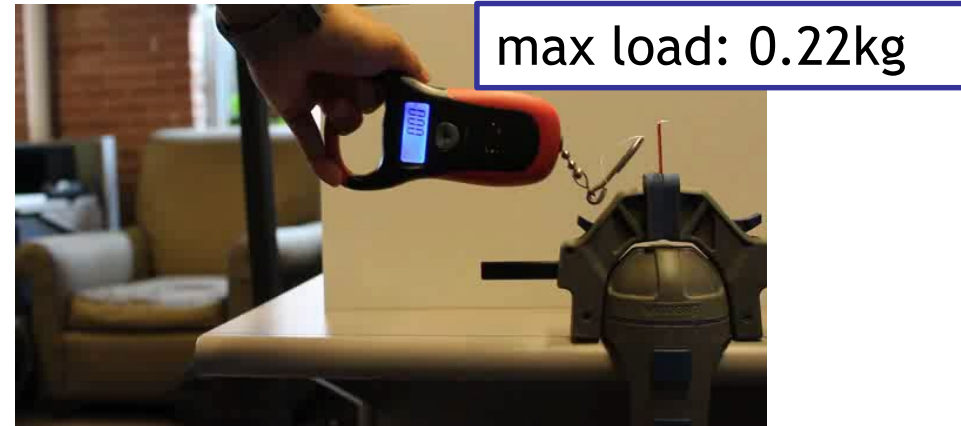
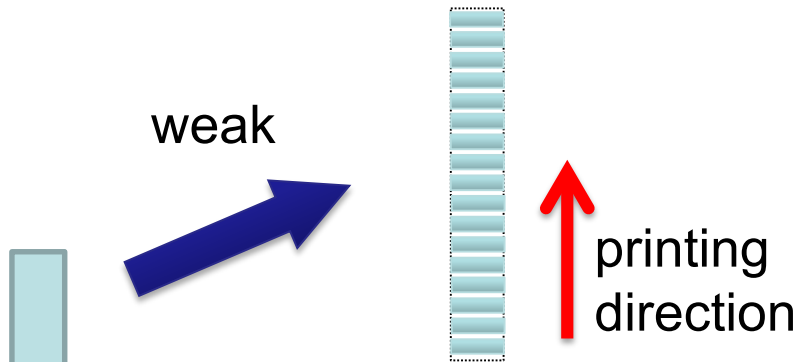
↑
printing
direction

strong

weak

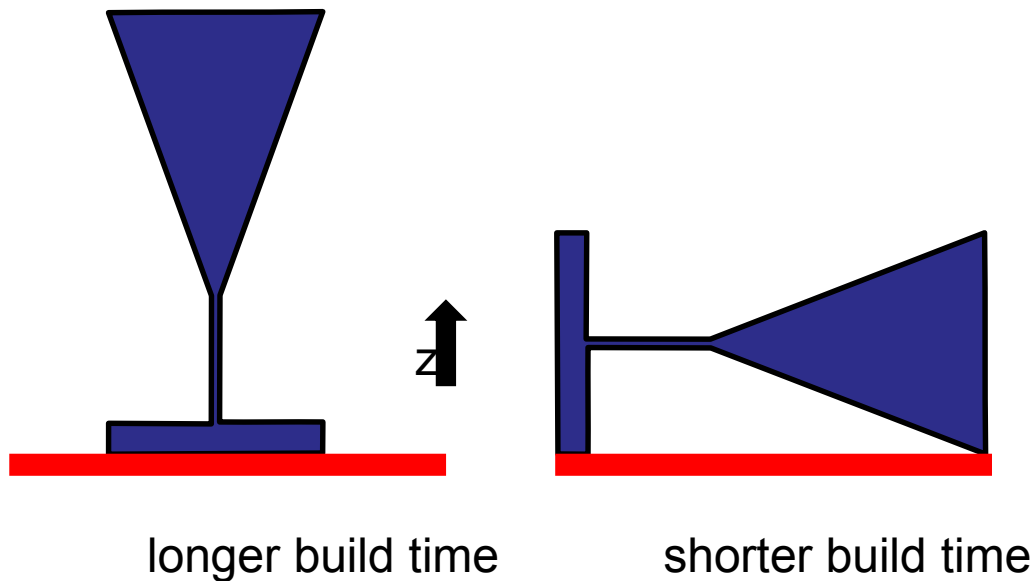


Model Orientation: Mechanical Properties

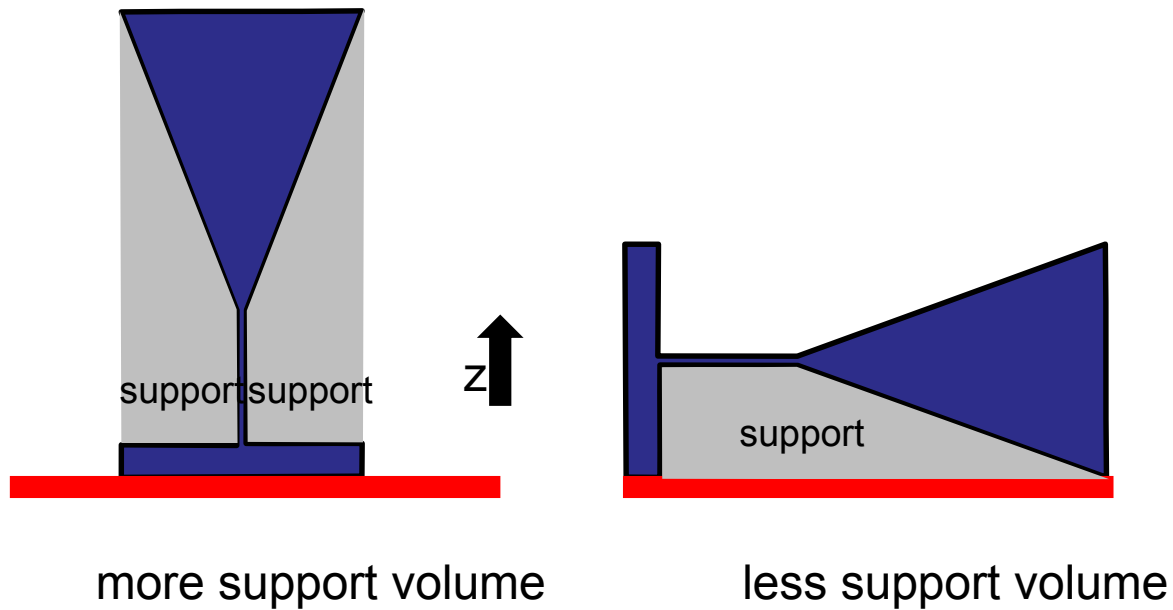


Model Orientation: Build Time

- Build speed is slower for the z direction compared to the xy direction



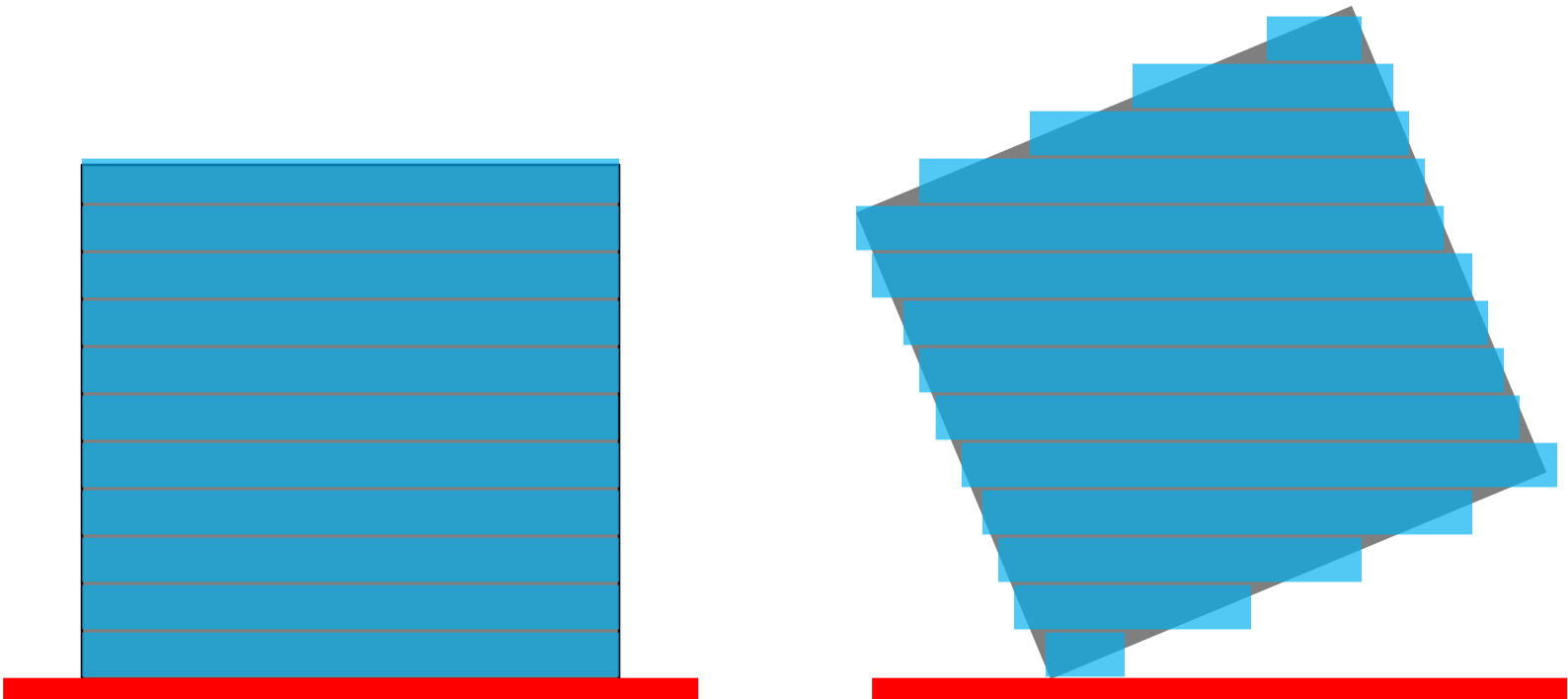
Model Orientation: Support Volume





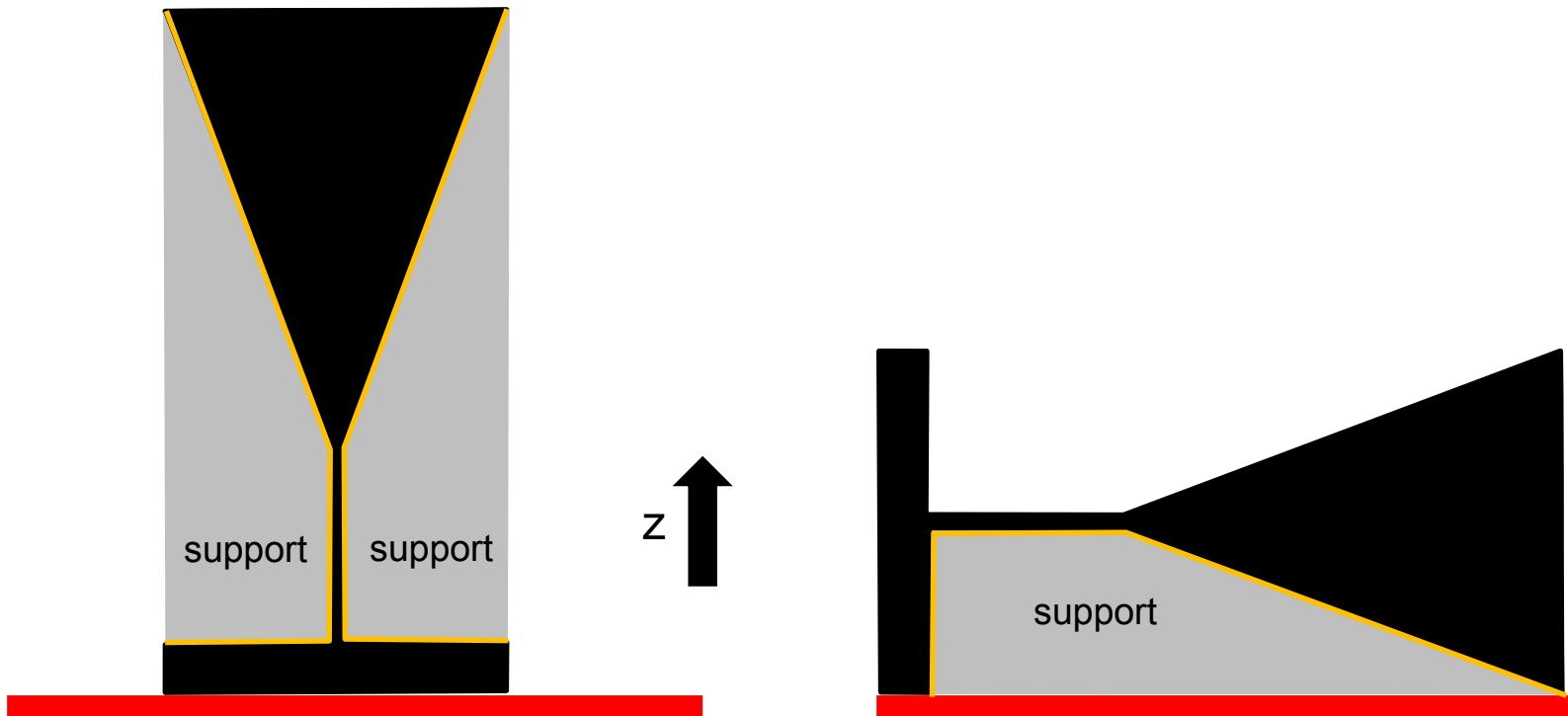
Model Orientation: Surface Accuracy

- Difference between the input shape and the printed shape
 - Difference in volume
 - Difference in the normal direction



Model Orientation: Support Contact Area

- **Contact area** between support material and object
 - should be minimized
 - can be geometrically computed

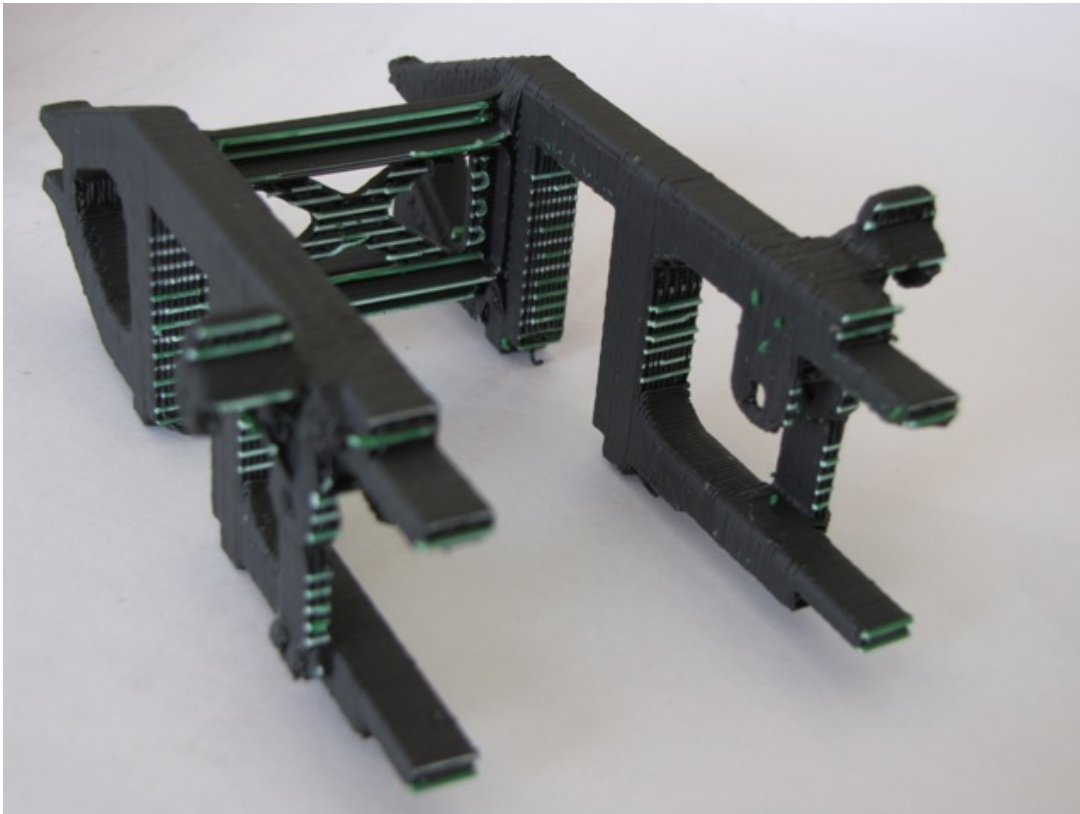


larger contact area

smaller contact area

Model Orientation: Support Contact Area

- After support removal



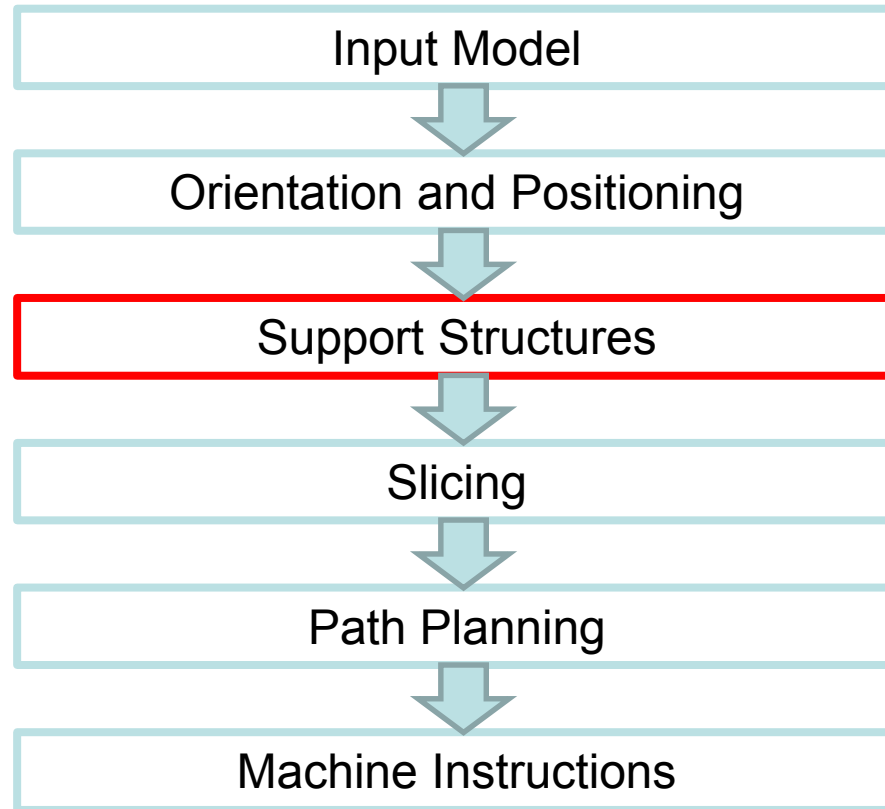
Algorithms for Specifying Model Orientation

- Manual placement
 - User is responsible for placing parts on the build tray
- Semi-automated placement
 - User places parts on the build tray
 - System provides feedback on build time, support volume, support contact area, mechanical properties
- Automated placement
 - orientation is computed using optimization according to one or more objectives (build time, support volume, support area, mechanical properties)

Simple Algorithm to Compute Orientation

- Exhaustive search
- Compute a uniform set of directions
- Compute a value of objective function for each direction
 - Build time
 - Support volume
 - Support contact area
 - Mechanical strength
- Pick orientation with the minimum value of the objective
 - Single objective (typical)
 - Multiple objectives need to be weighted (weights are difficult to set)

3D Printing Software Pipeline

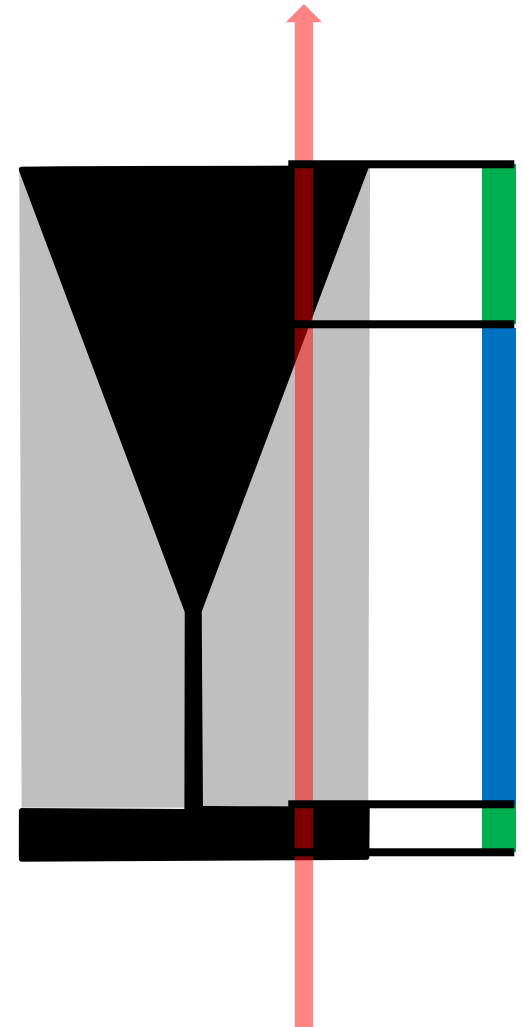


Why Support Structures?



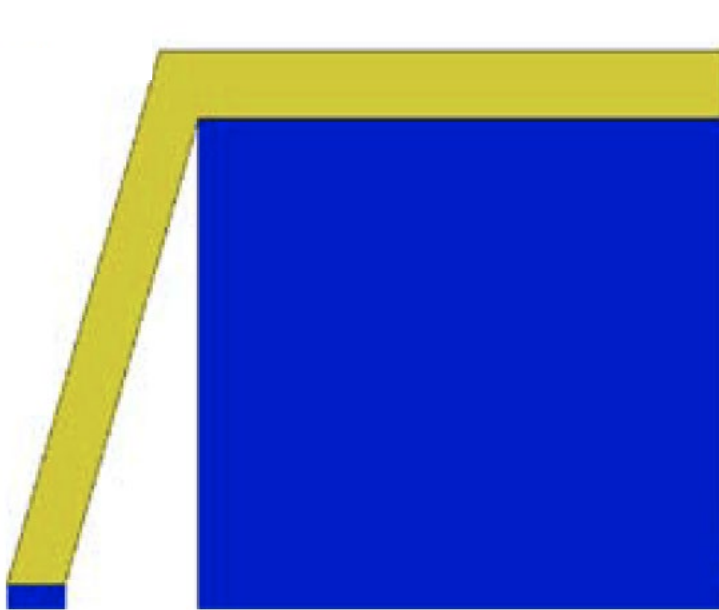
Simple Conservative Algorithm

- Use ray casting in the z direction to compute all intersections for a ray
- Sort intersections in the increasing z to determine intervals **inside**/**outside** of the object
- Any **outside** intervals before the last **inside** interval should contain support

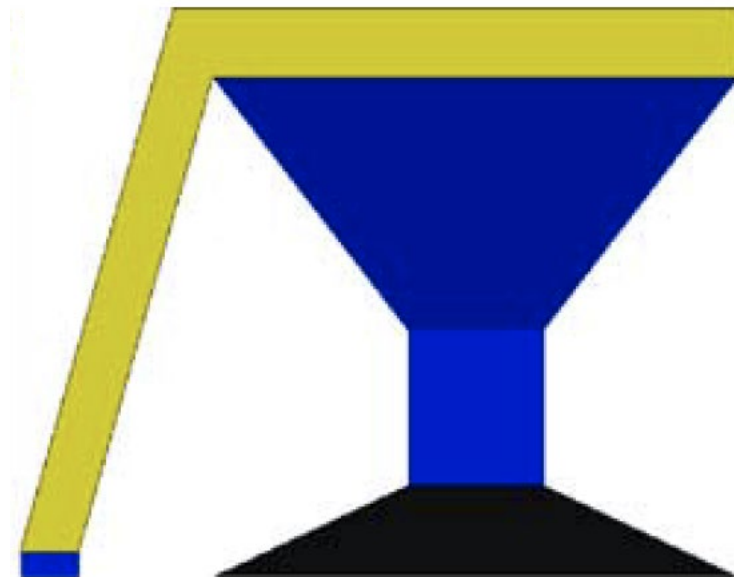


Support Generation: Draft Angle

- Many printers can print at some draft angle

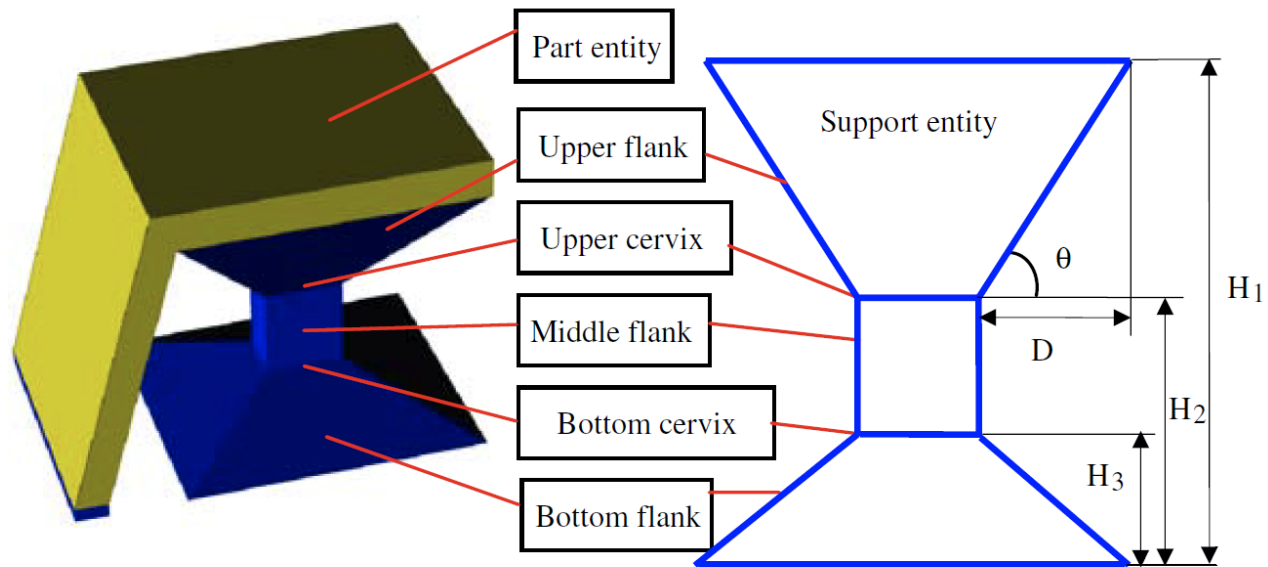


unoptimized support

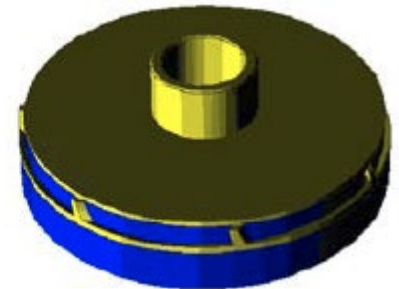
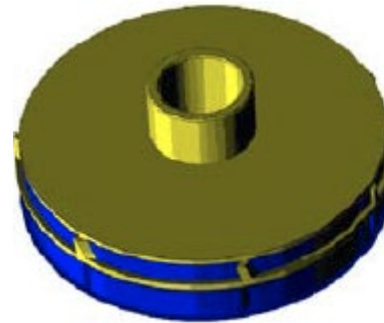
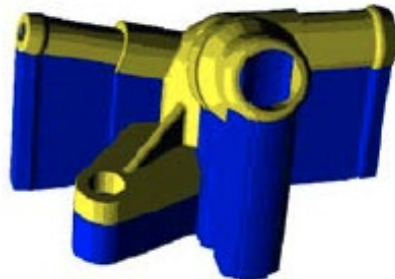
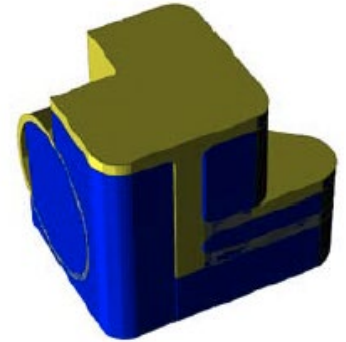
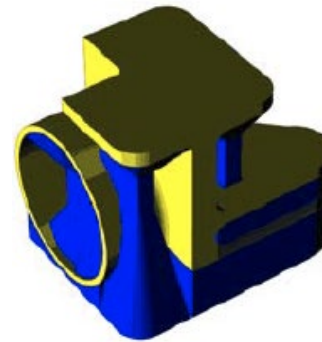
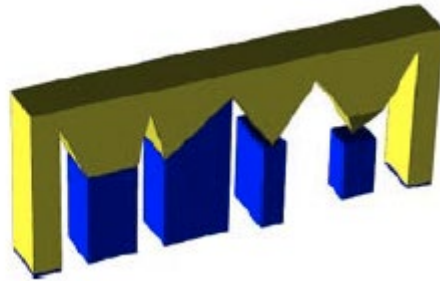
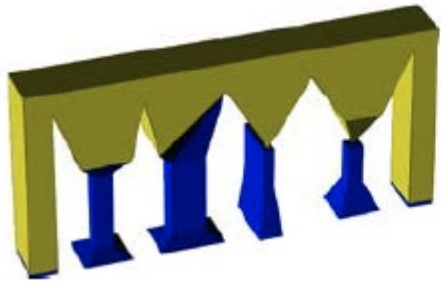


optimized support

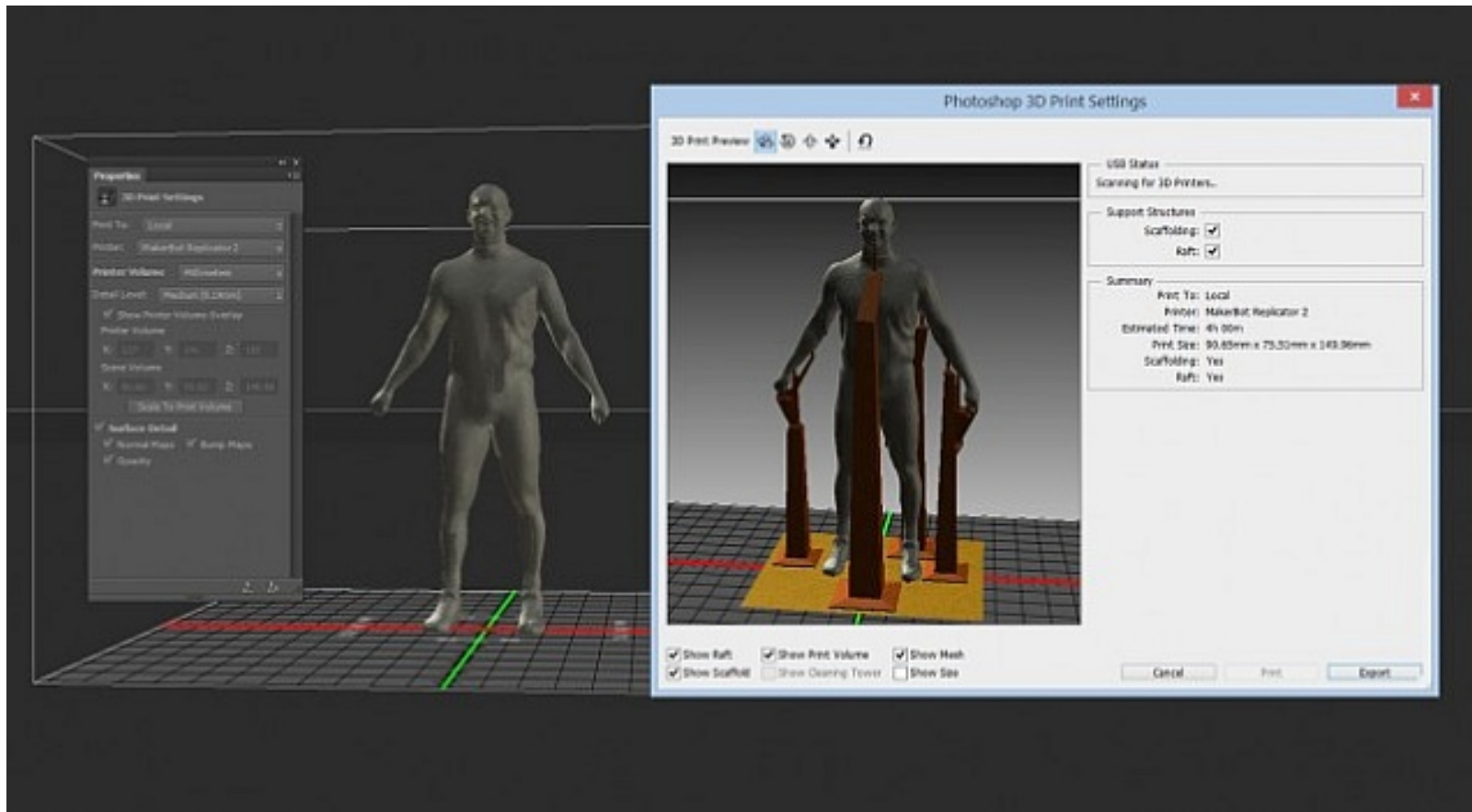
Optimized Support Structures



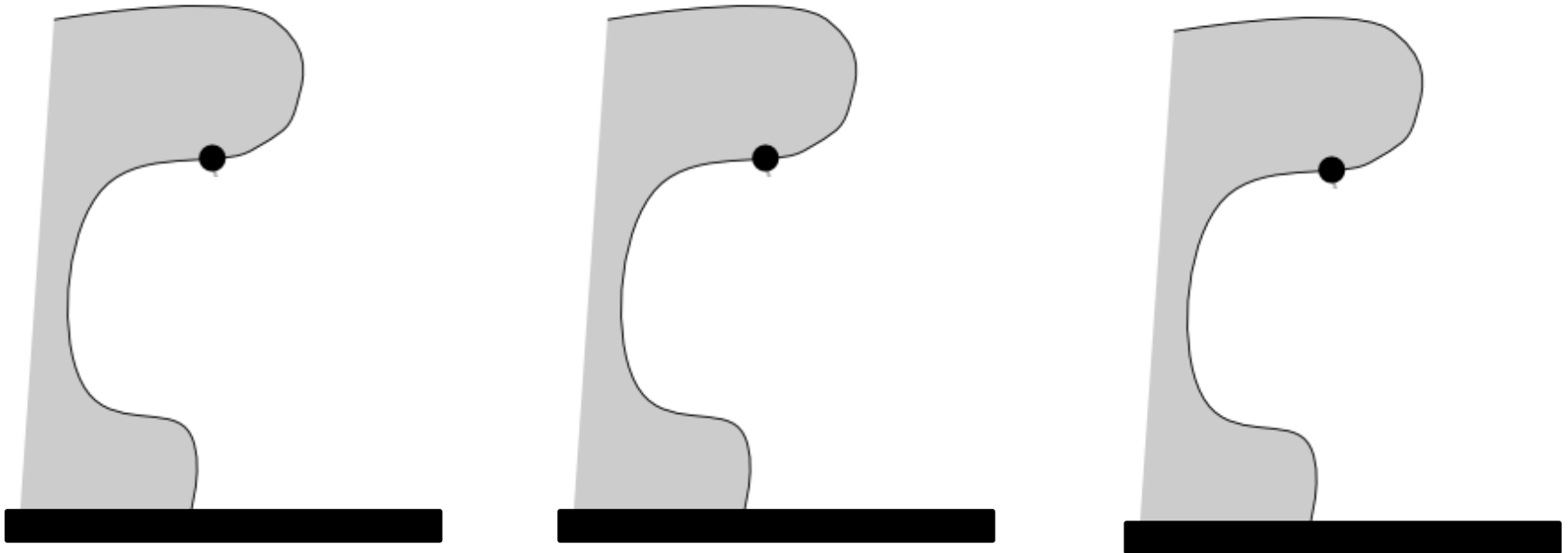
Unoptimized vs. Optimized Support Structure



Advanced Algorithms: Photoshop

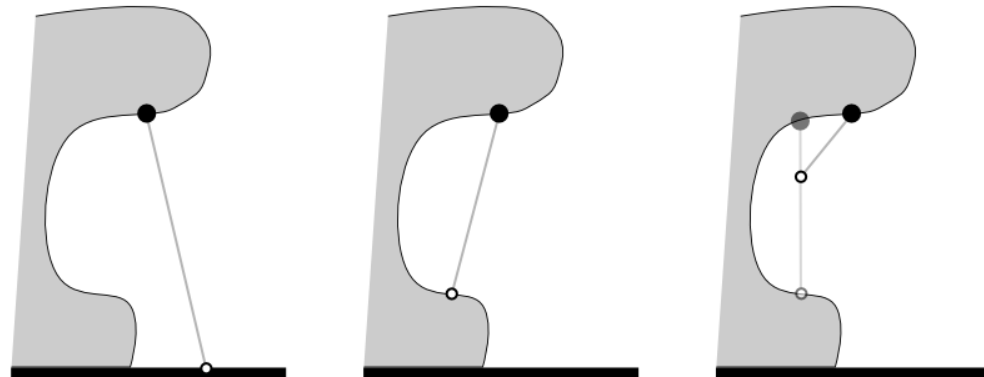


Advanced Algorithms: MeshMixer



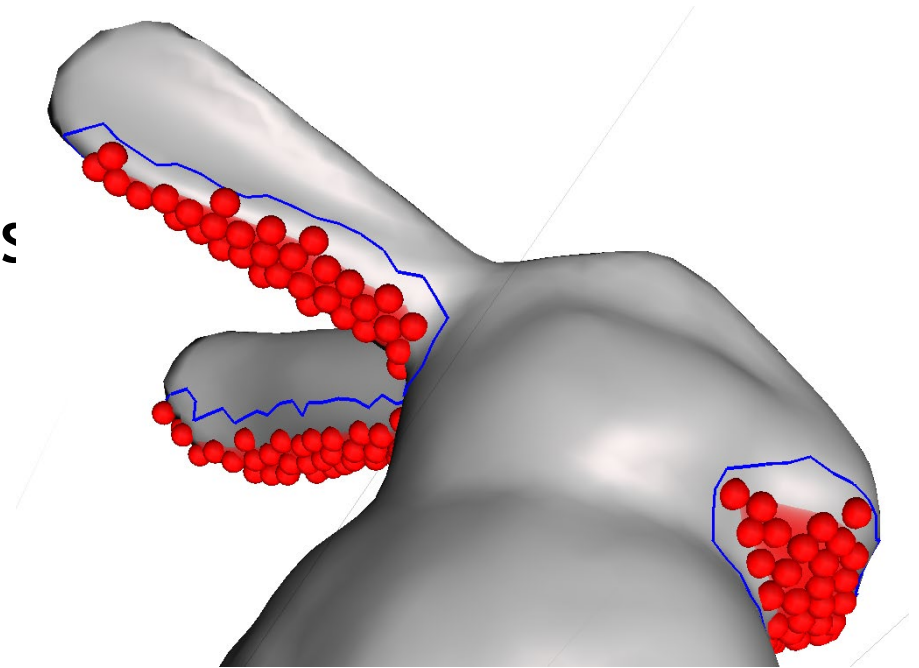
Advanced Algorithms: MeshMixer

- Assume we have a point that needs support
- Can connect point (via support post) to:
 - 1) Ground
 - 2) Object
 - 3) Another post
- Problems to solve:
 - 1) Support Points?
 - 2) Connection Strategy?



Advanced Algorithms: MeshMixer

- Compute point-sampling of overhang areas
- Densely cover overhangs from bottom to top



Advanced Algorithms: MeshMixer

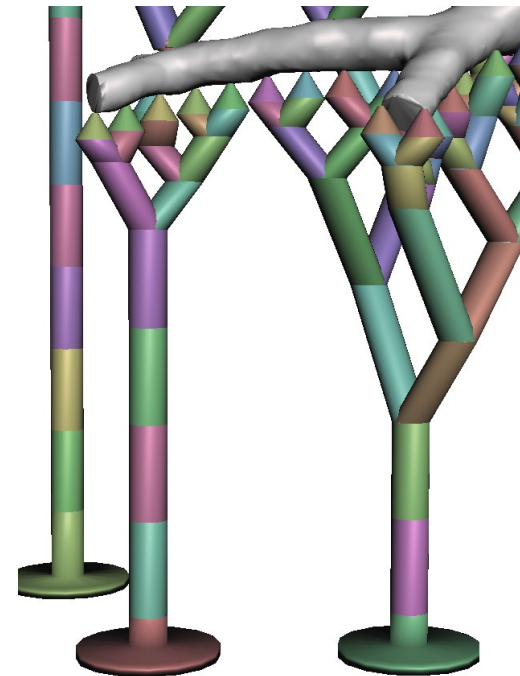
- Top-Down Growing Strategy:

- 1) Add support points to priority queue

- Sorted top-down

- 2) While queue is not empty

- Pop topmost point P
- Try to connect downwards to ground/object.
- Else try to pair with closest free point
- Else grow downwards a fixed step.
- Add new point to queue



Advanced Algorithms: MeshMixer



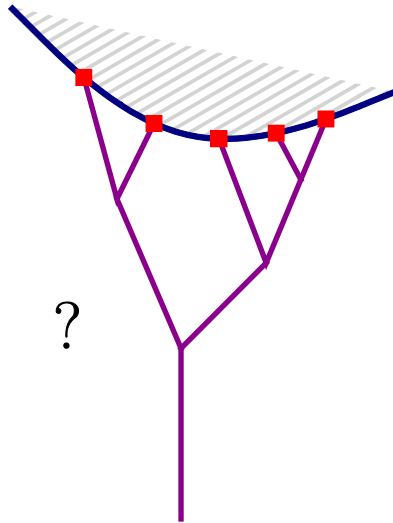
Source:

<http://www.youtube.com/watch?v=aFTyTV3wwsE>

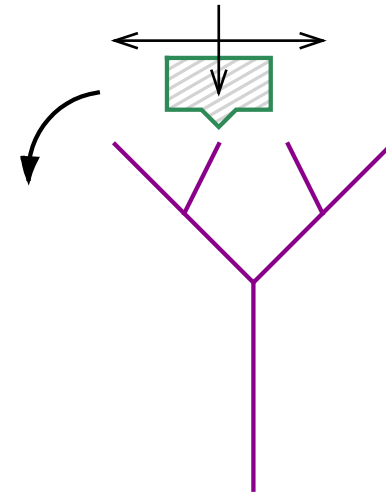


Advanced Algorithms: Bridging the Gap

Support/length ratio



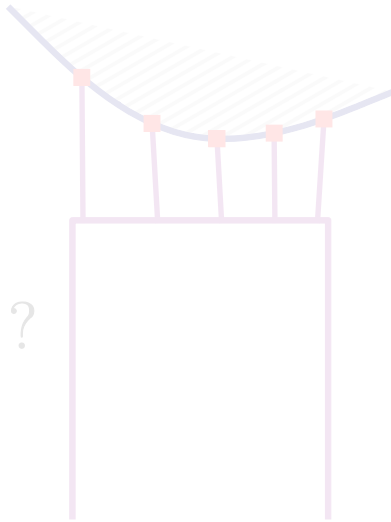
✗ Sensitive to **torque**



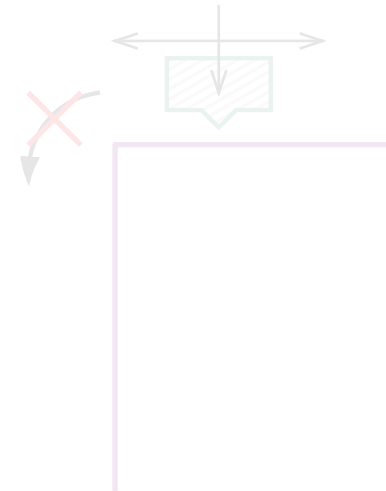
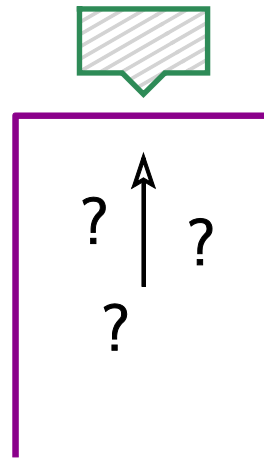
Source: Dumas, Siggraph 2014

Think With Bridges

Support/length ratio

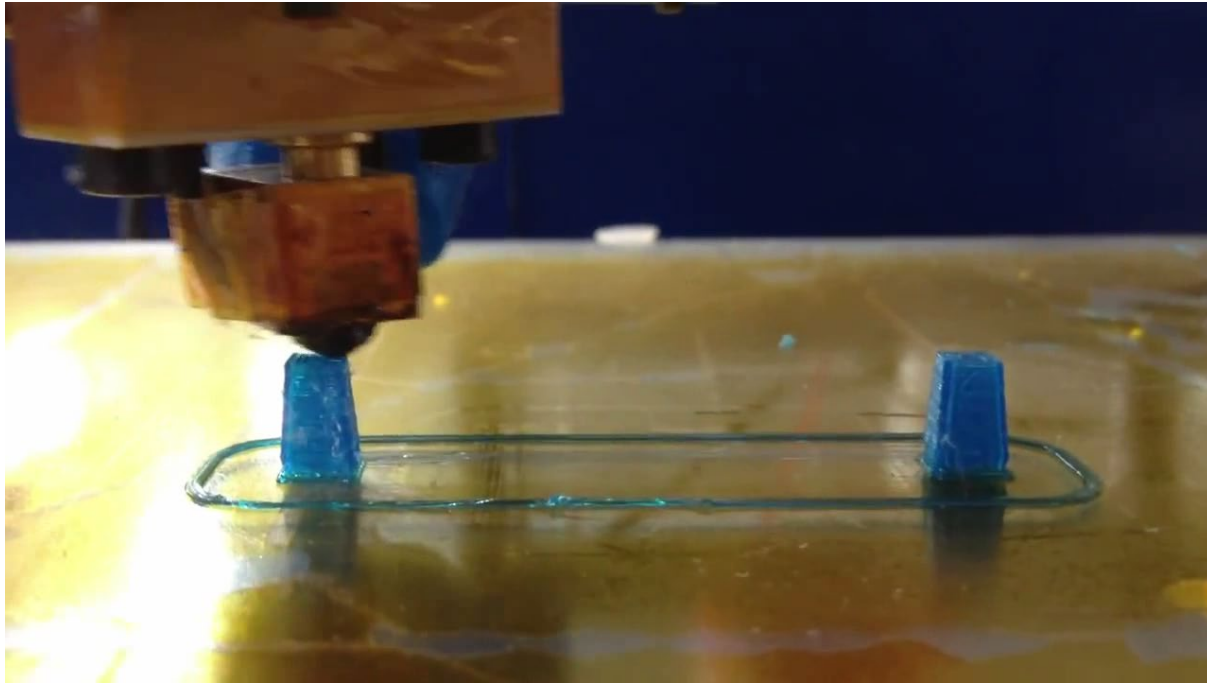


Resistant to torque



Source: Dumas, Siggraph 2014

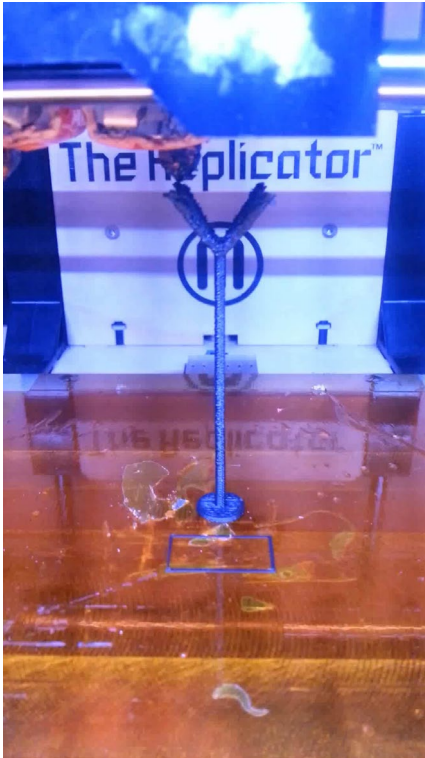
Printing Bridges



Source: <http://youtu.be/wK2APNwEoSk>

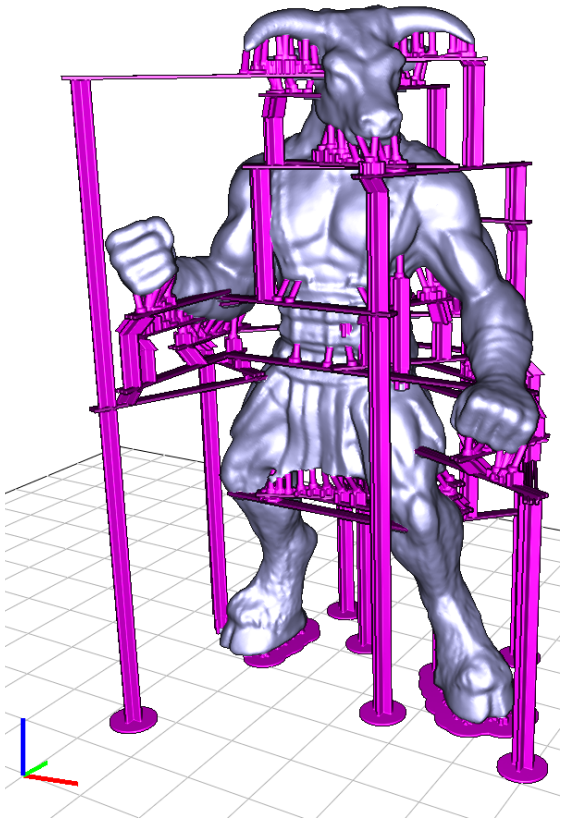
Source: Dumas, Siggraph 2014

Torque



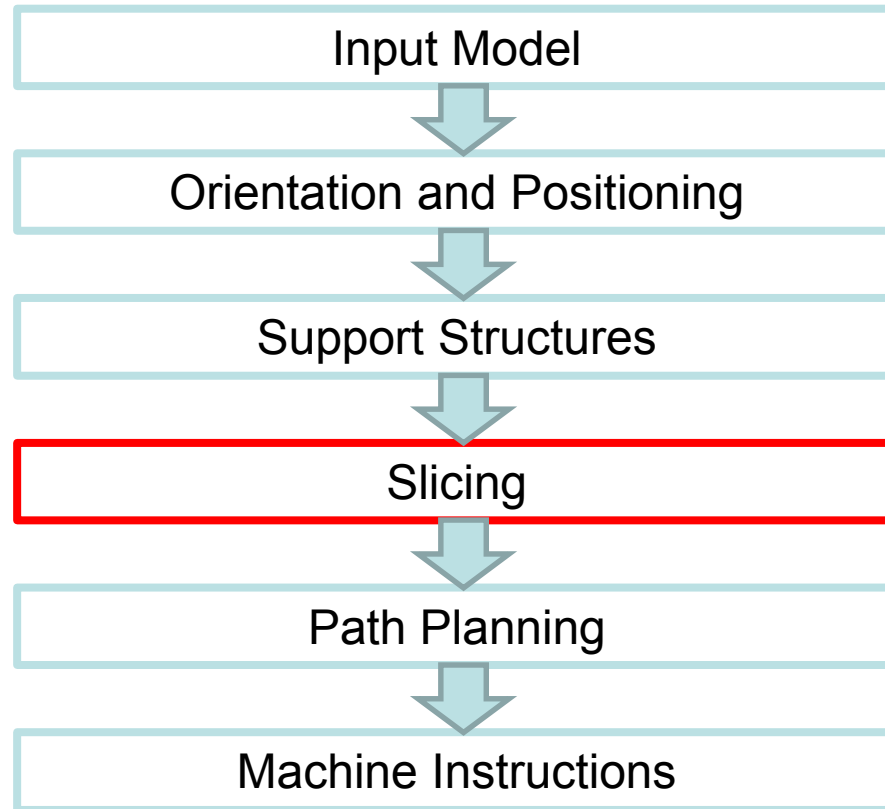
Source: Dumas, Siggraph 2014

Advanced Algorithms: Bridging the Gap



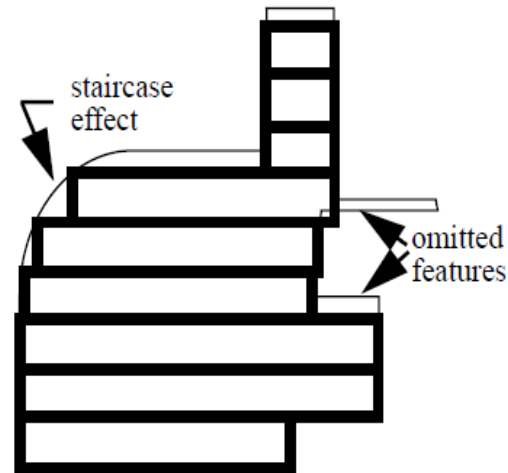
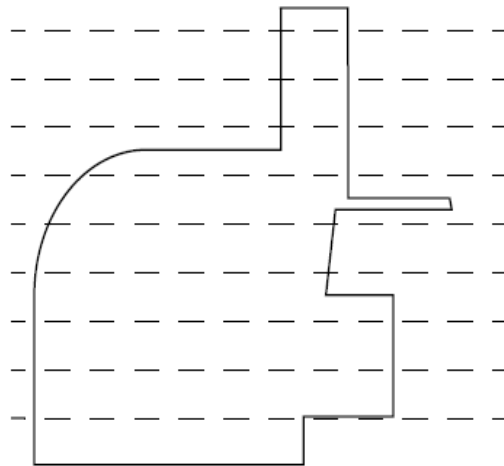
Source: Dumas, Siggraph 2014

3D Printing Software Pipeline



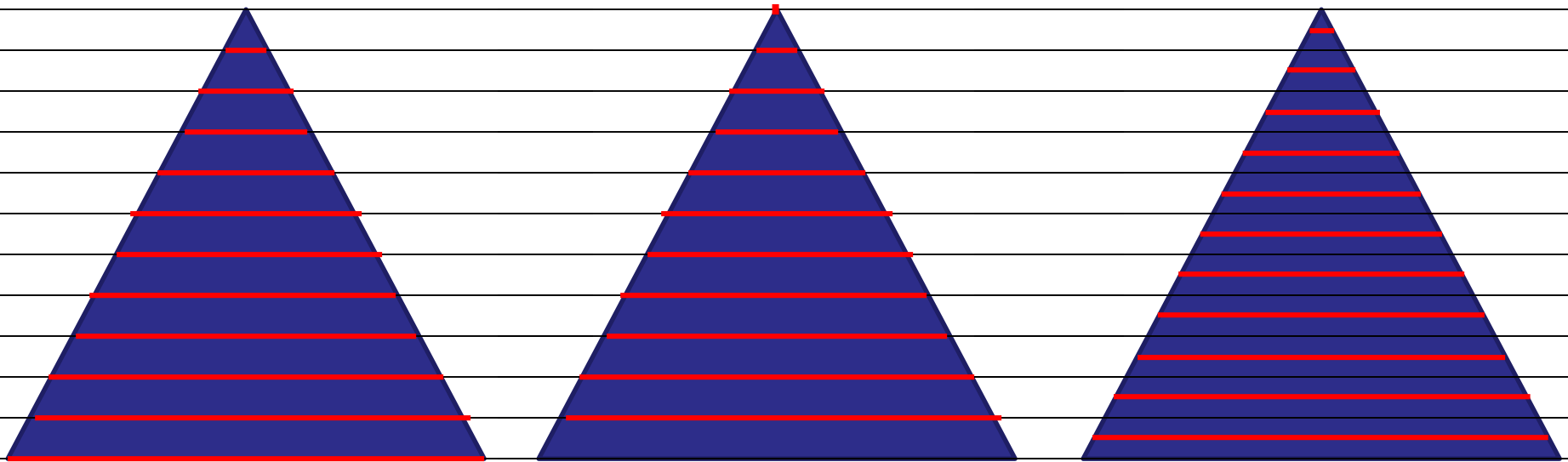
Slicing

- For a discrete z value, compute an intersection of a plane with a model



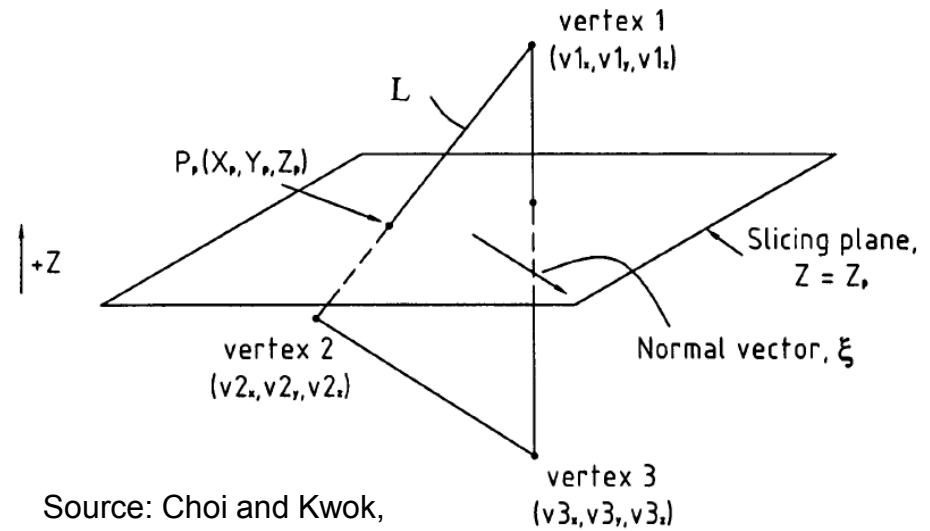
Slicing Issues

- Given that each layer has some finite thickness, which plane to choose
 - bottom
 - top
 - middle



Slicing Algorithms: Direct Plane-Triangle Intersection

- For each triangle
 - Intersect triangle with the z plane
 - If they intersect, store the line segment
- Connect line segments, store contours

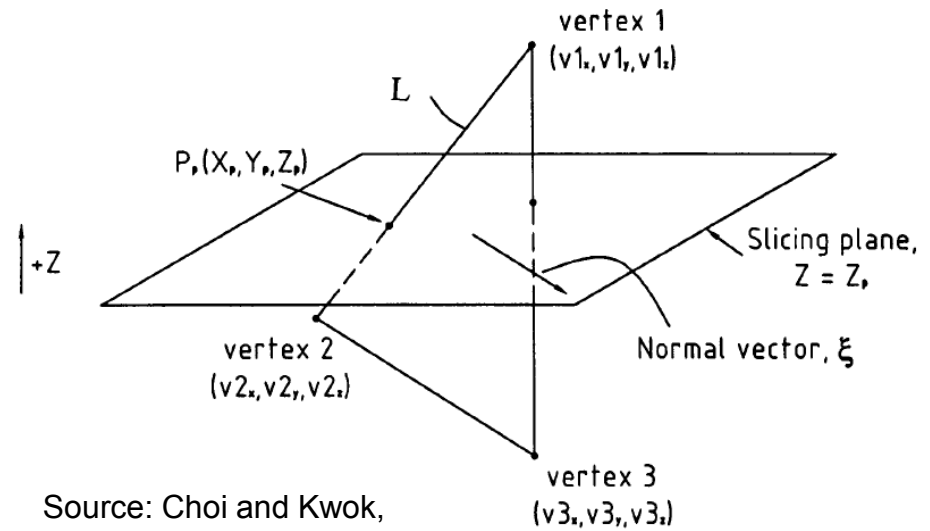


Source: Choi and Kwok,
2002

Slicing Algorithms: Direct Plane-Triangle Intersection

- For each triangle
 - **Intersect triangle with the z plane**
 - If they intersect, store the line segment
- Connect line segments, store contours

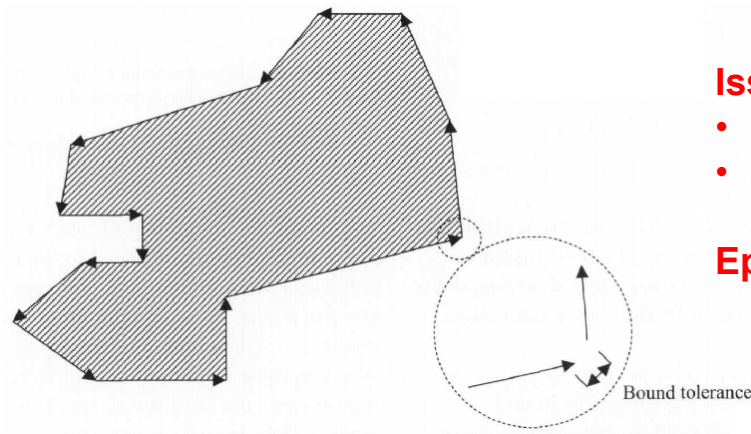
1. **Intersect each edge with the plane**
2. **If two intersection points, connect them to form a line segment**



Source: Choi and Kwok,
2002

Slicing Algorithms: Direct Plane-Triangle Intersection

- For each triangle
 - Intersect triangle with the z plane
 - If they intersect, store the line segment
- Connect line segments, store contours



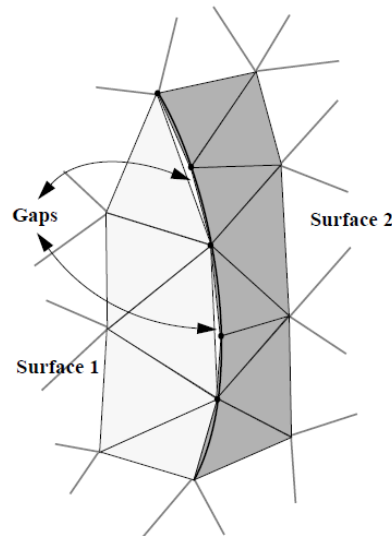
Issues

- spurious line segments
- missing line segments

Epsilons

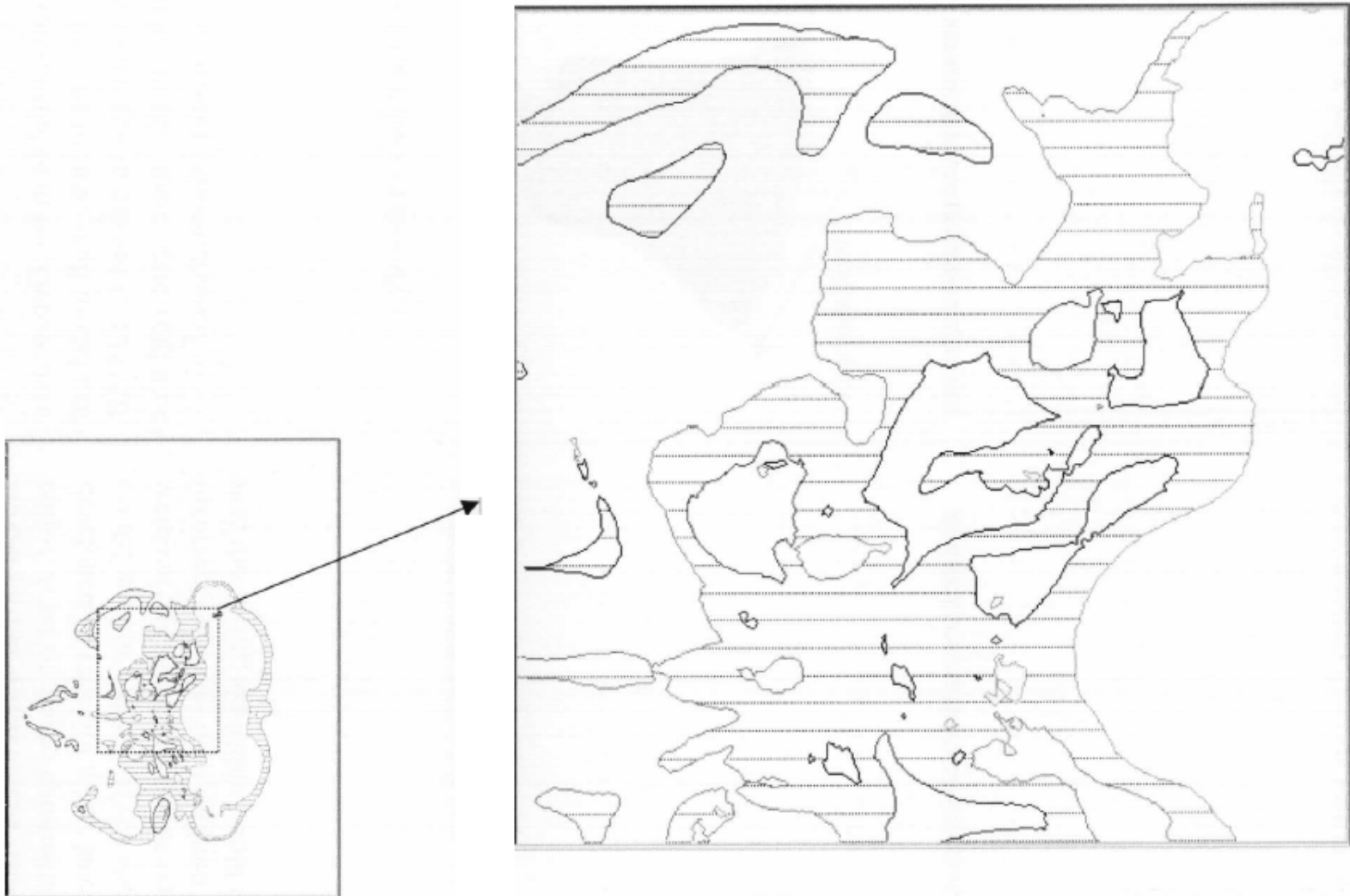
Slicing Algorithms: Direct Plane-Triangle Intersection

- STL models are not always watertight -> epsilons



Source: Marsan et al, 1998

Slicing Results with Direct Plane-Triangle Intersection



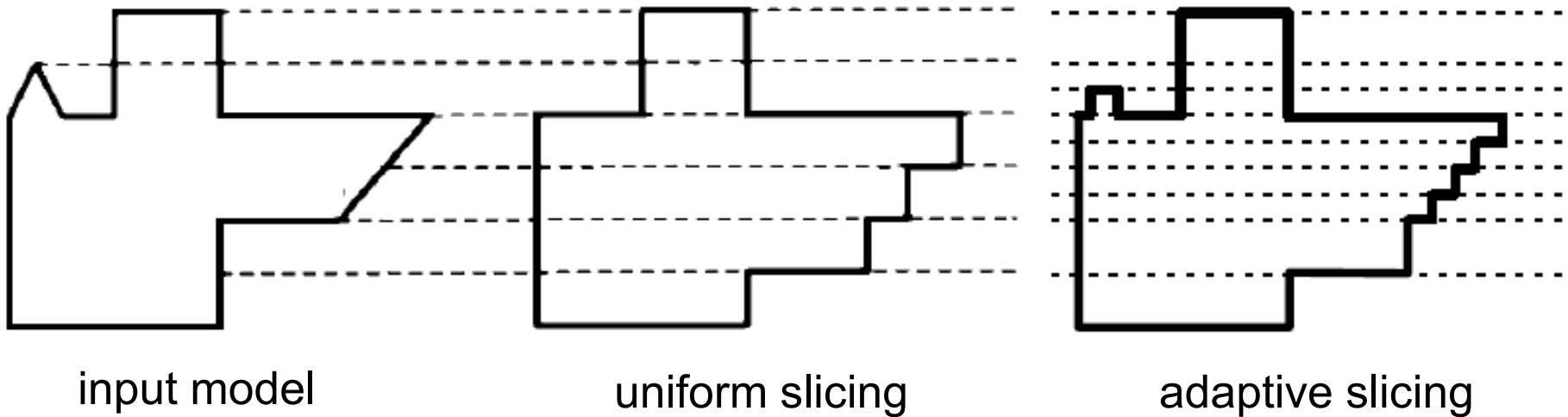
From Choi and Kwok, 2002

More Efficient Slicing

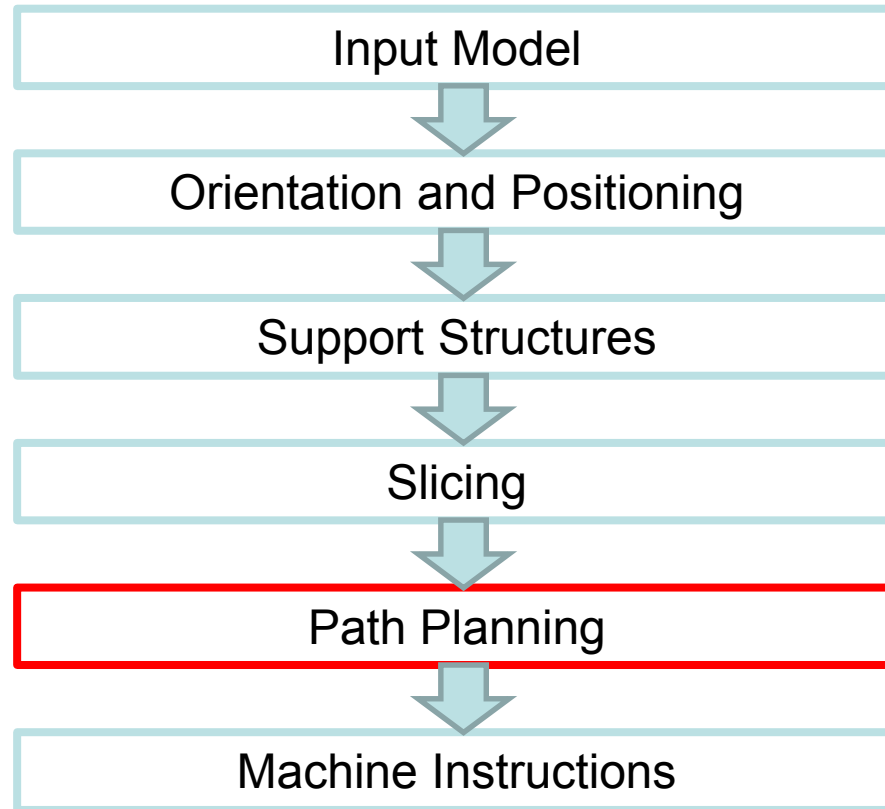
- Slicing might be a **bottleneck** when working with large models
- Lots of opportunities for very fast/robust algorithms
 - Using computer graphics acceleration data structures
 - Z-sorting
 - GPU-based methods
 - Efficient out-of-core methods
 - when model does not fit in the memory

Adaptive Slicing

- Slice height is adapted to the input geometry
- Adaptive slicing is rarely used



3D Printing Software Pipeline





What Does Path Planning Influence?

- Build time
 - repositioning the tool at the start of a new path
 - accelerating and decelerating for direction changes
- Surface accuracy
 - the filament size
- Distortion
 - materials with a high coefficient of thermal expansion
 - the top layer shrinks when it hardens and it distorts since it is tied to the bottom layer
- Stiffness and strength
 - fill pattern
 - the area and strength of bonds depends on spacing and the time interval between the tool traversal

Path Planning for Vector-based 3D Printing

- Contour
- Contour + solid interior
- Contour + interior fill pattern

Path Planning for Vector-based 3D Printing: Contour

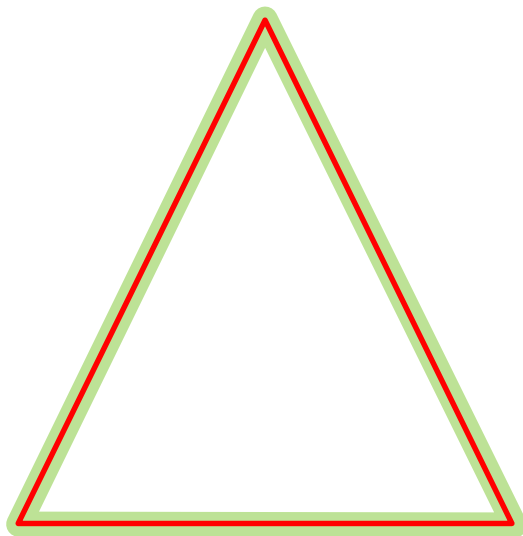
- Allows manufacturing hollow objects, some overhangs, some tilted surfaces
- Reduces frequency of tool repositioning
- Reduces support structures



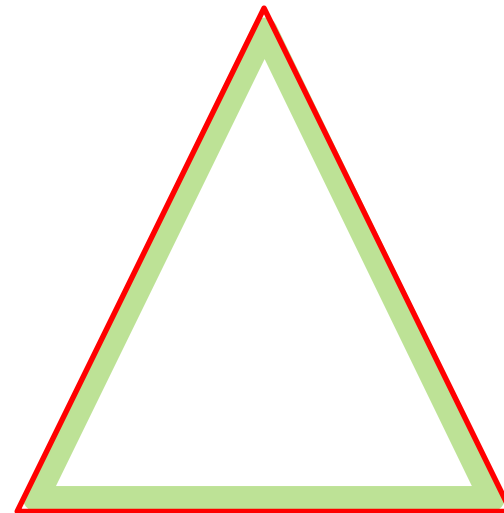
Source: <http://www.3ders.org>

Path Planning for Vector-based 3D Printing: Contour

- Offset inwards by distance equal to the filament radius
 - offset vertices
 - remove self-intersections and model intersections



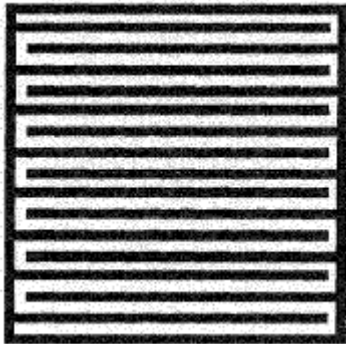
no offset



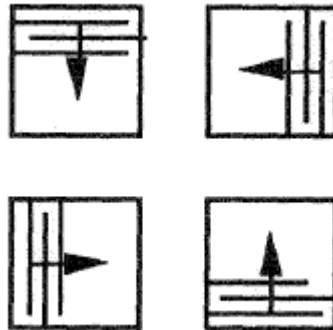
offset inwards

Path Planning for Vector-based 3D Printing: Interior

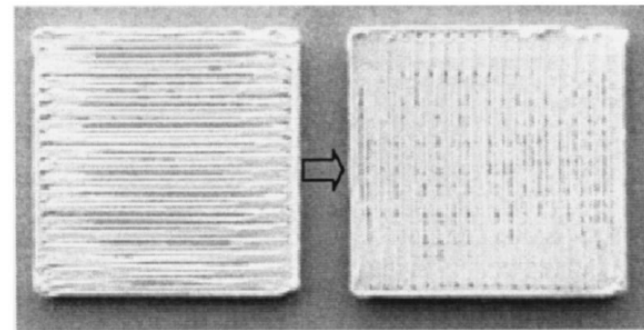
- Tracing contours is combined with filling the interior
- The interior can be completely filled



retracted hatch



alternate sequencing

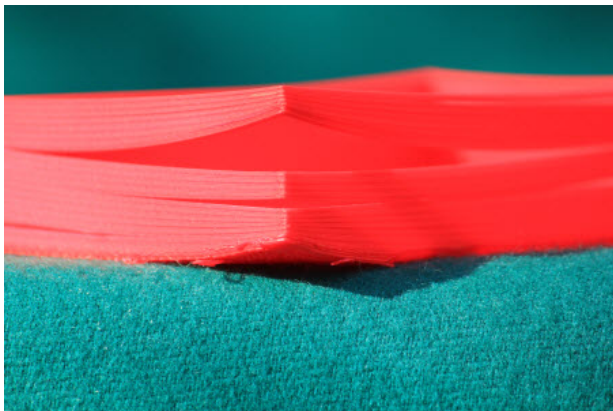
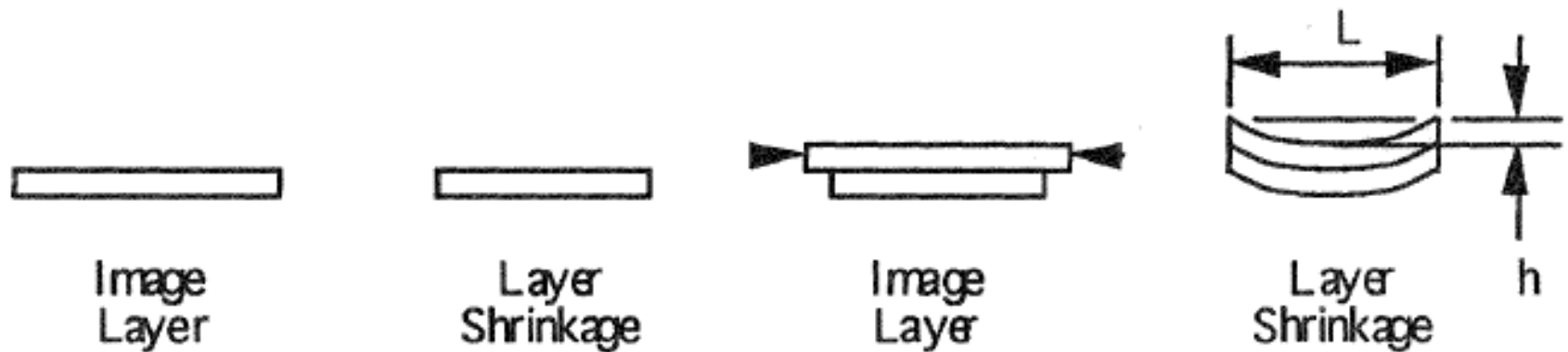


Horton et al 1993

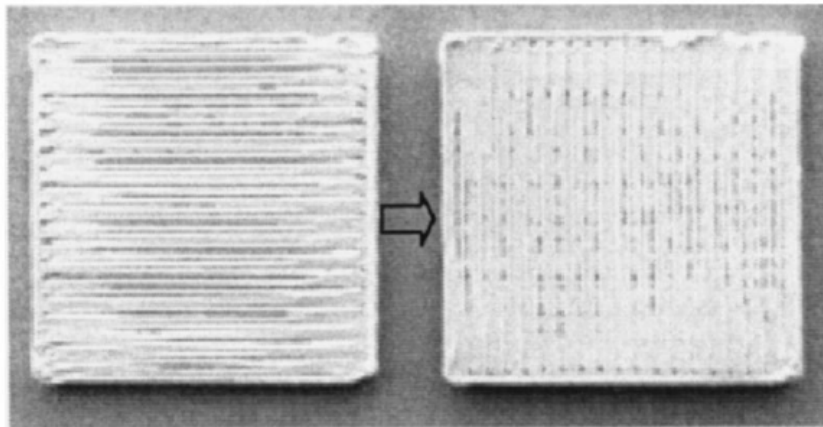
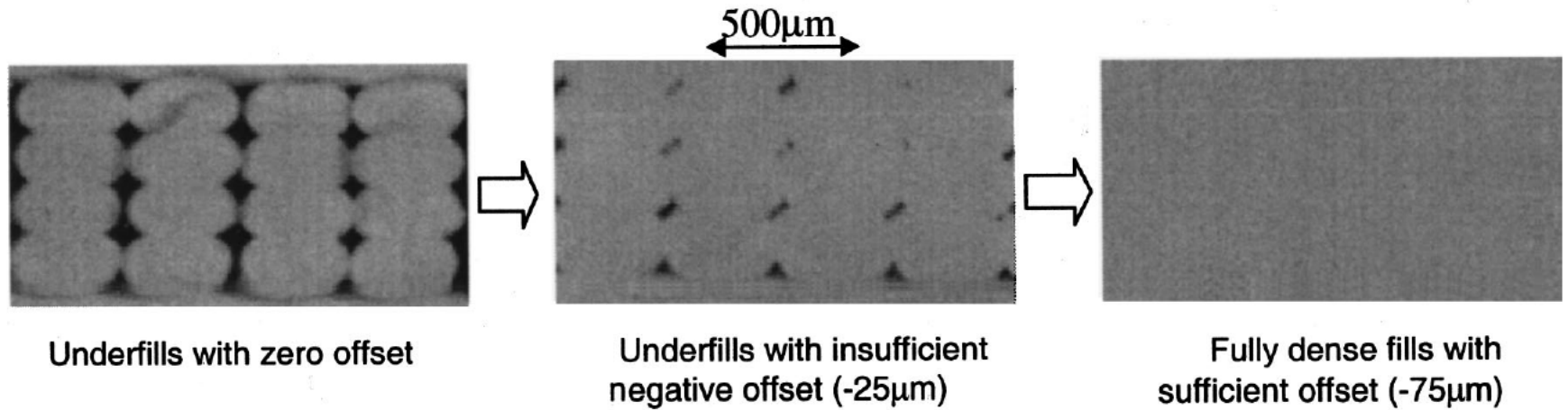
Han et al 2002

Material Shrinkage/Warping

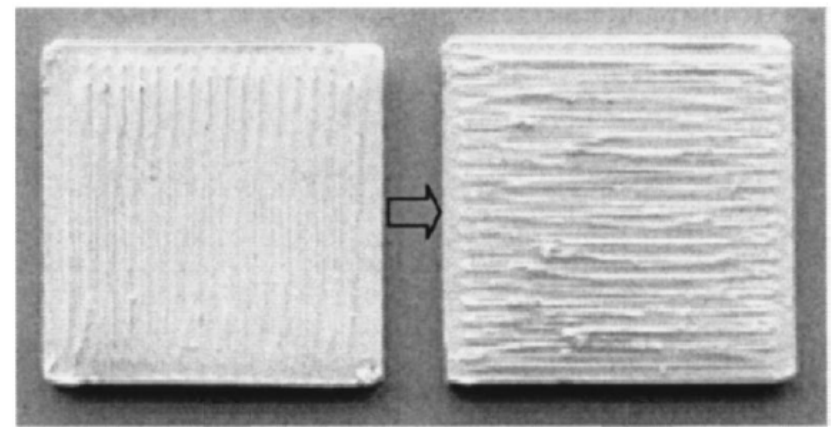
- Materials can shrink when cooling down/curing
- Path patterns can minimize this effect



Filling in the Whole Object



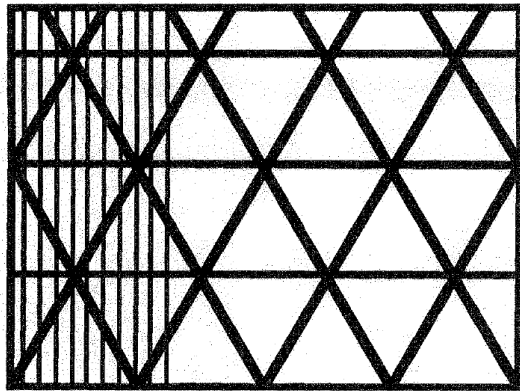
underfill



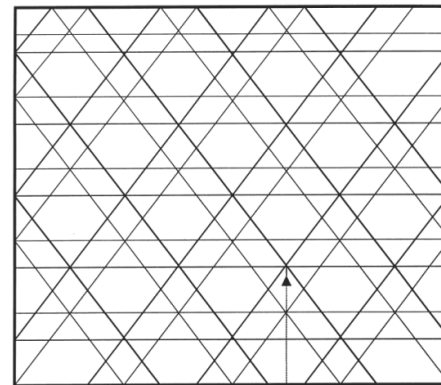
overflow

Path Planning: Interior

- Tracing contours is combined with filling the interior
- Many different fill patterns can be used



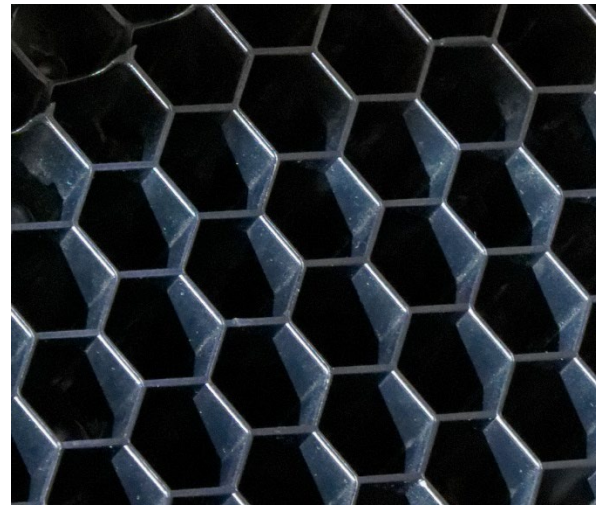
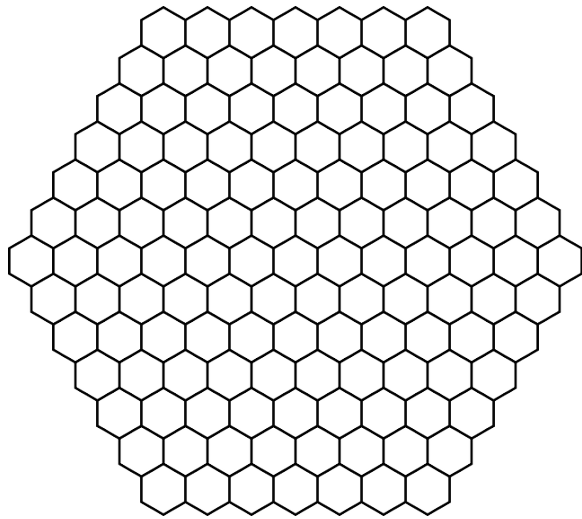
TriHatch



QuickCast

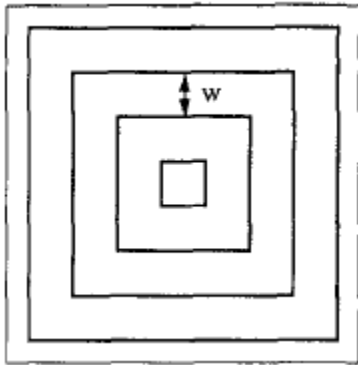
Path Planning for Vector-based 3D Printing: Interior

- A ***honeycomb-cell structure*** a good trade-off between overall weight and strength

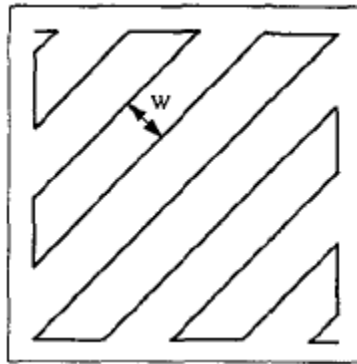


More Fill Patterns

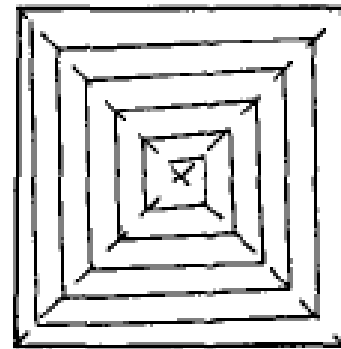
- In-plane
- Out-of-plane



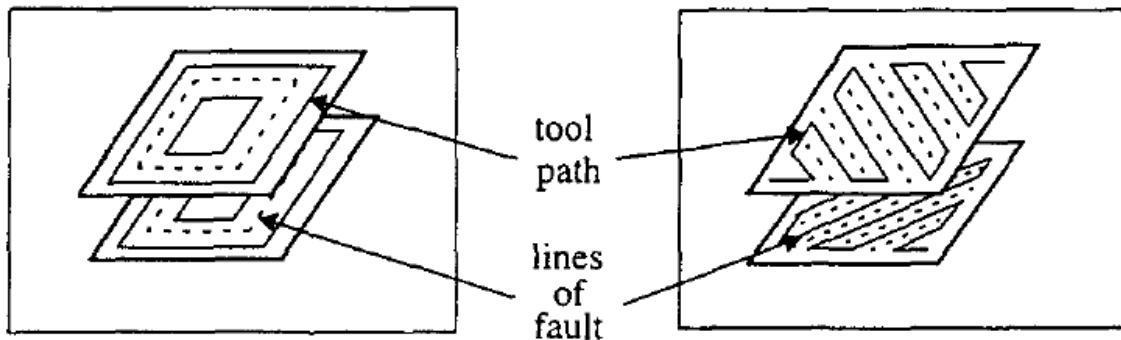
contour



raster

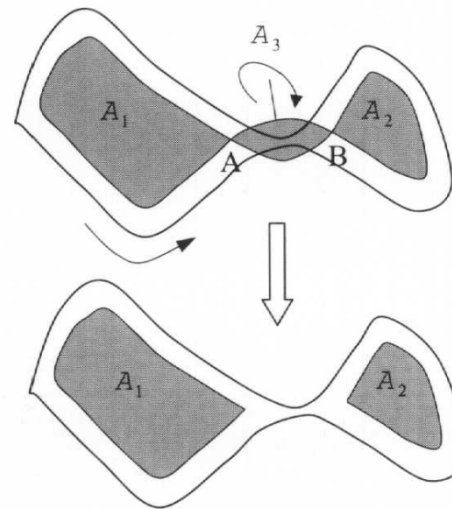
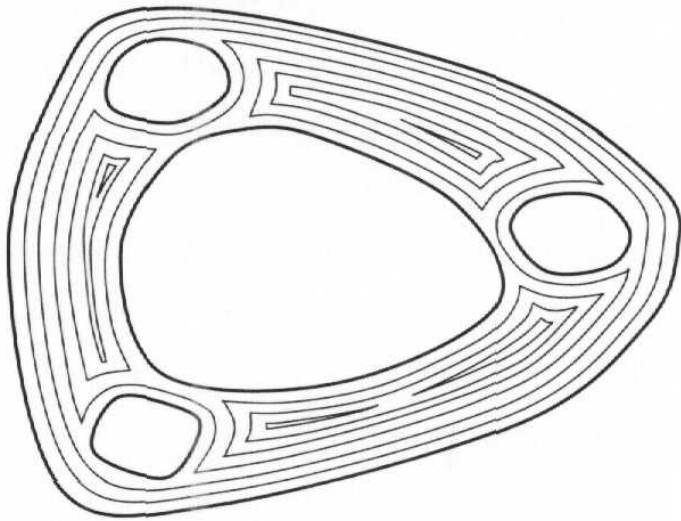


spiral

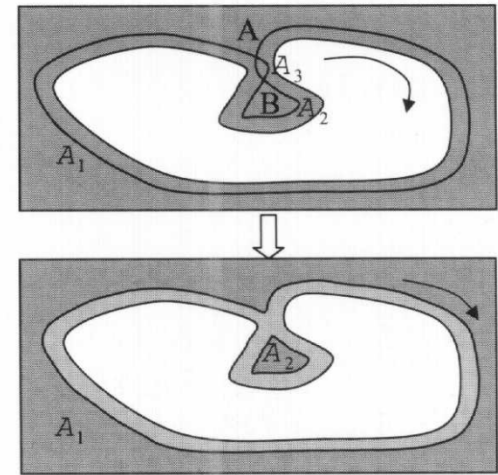


Equidistant Path Generation

- A set of paths are extracted based on distance to the surface

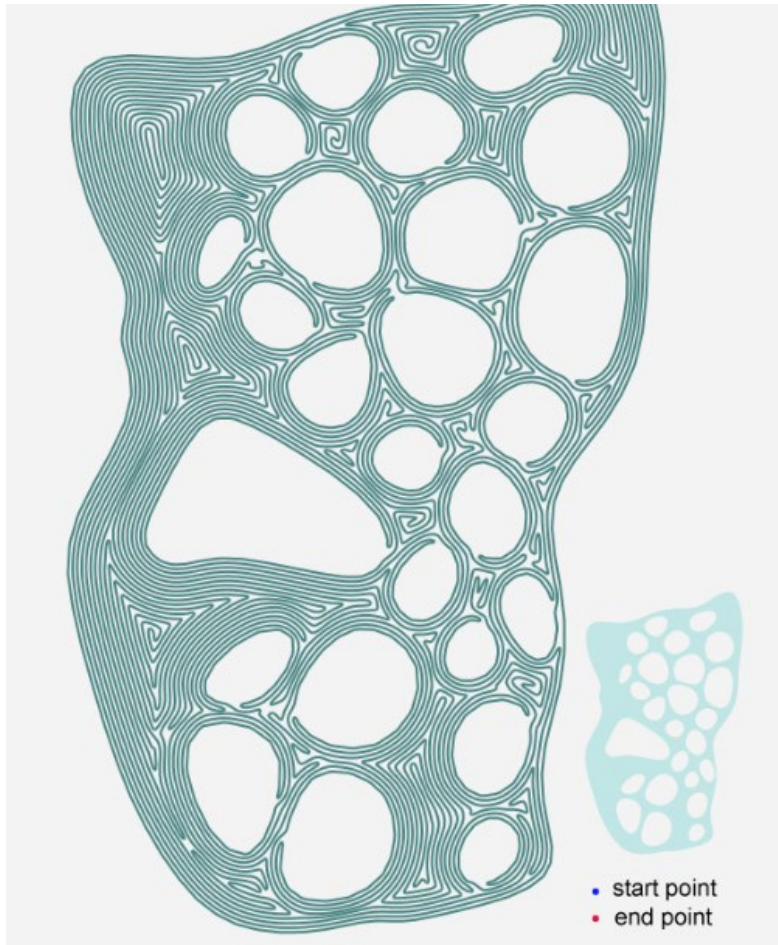


inward offset



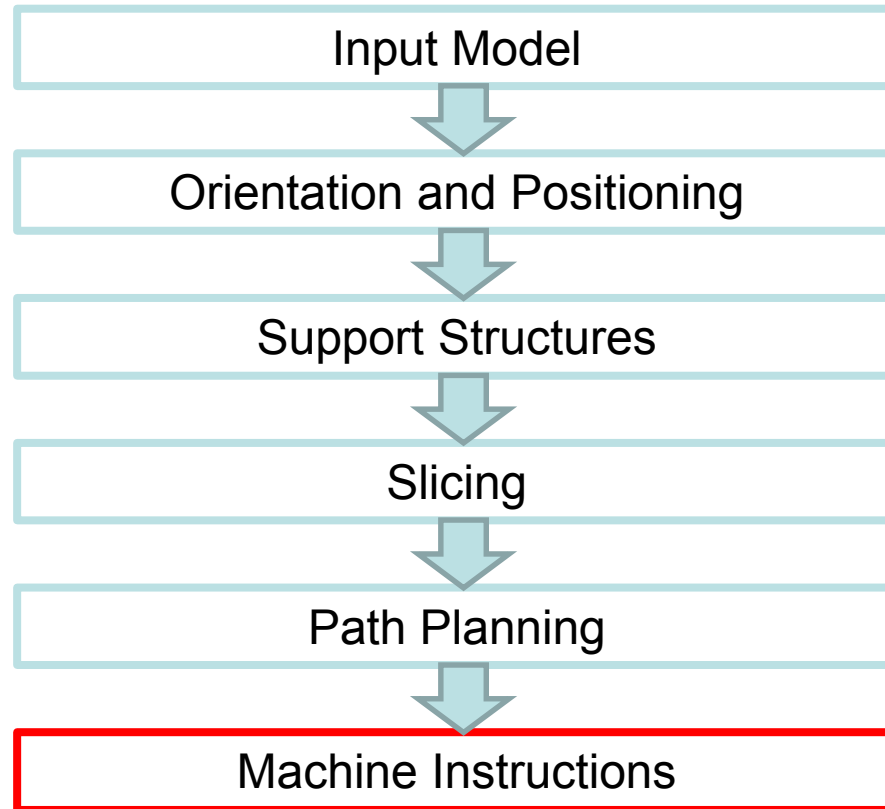
outward offset

Fermat Spirals

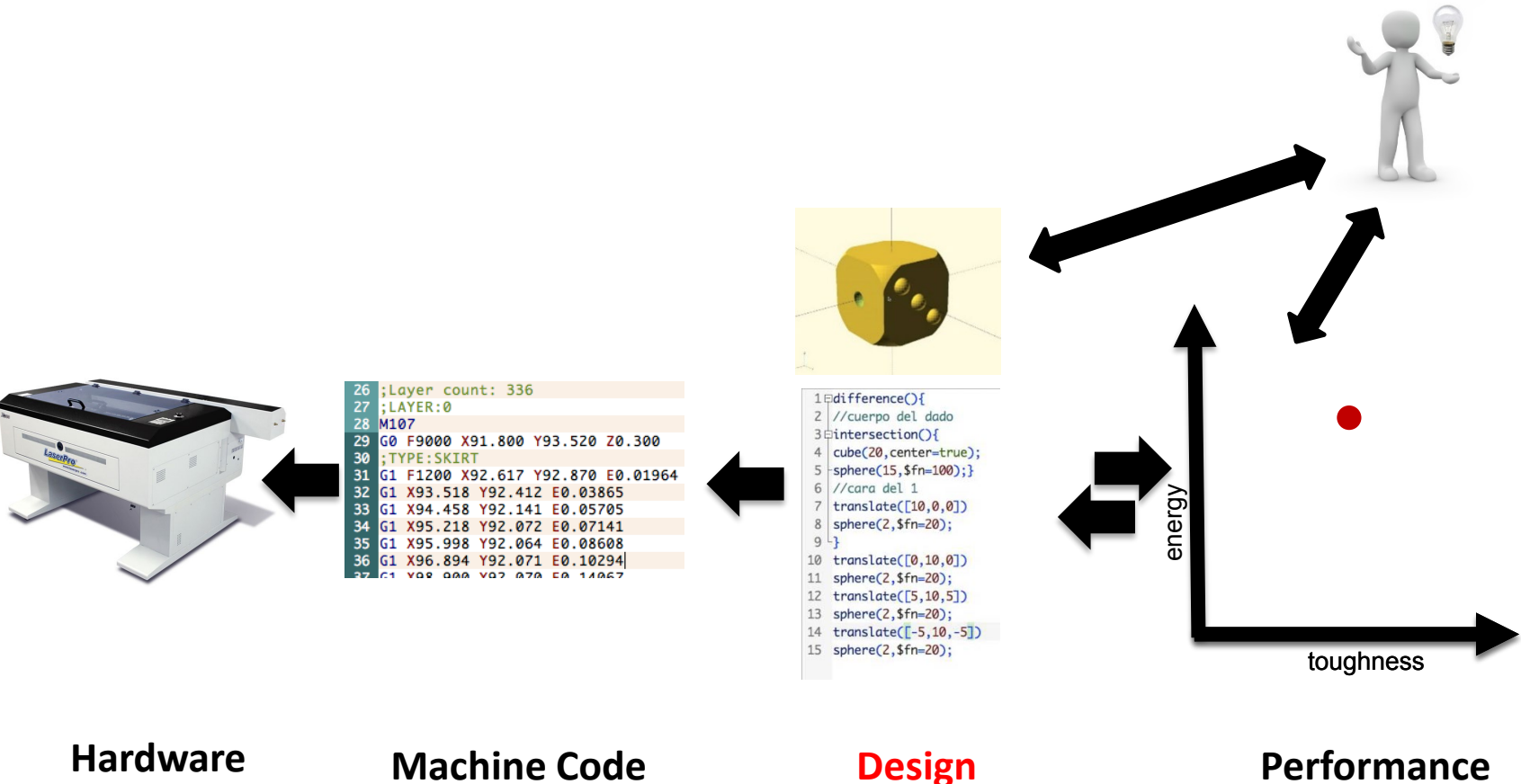


Zhao et al 2017

3D Printing Software Pipeline



Computational Design and Fabrication Pipeline

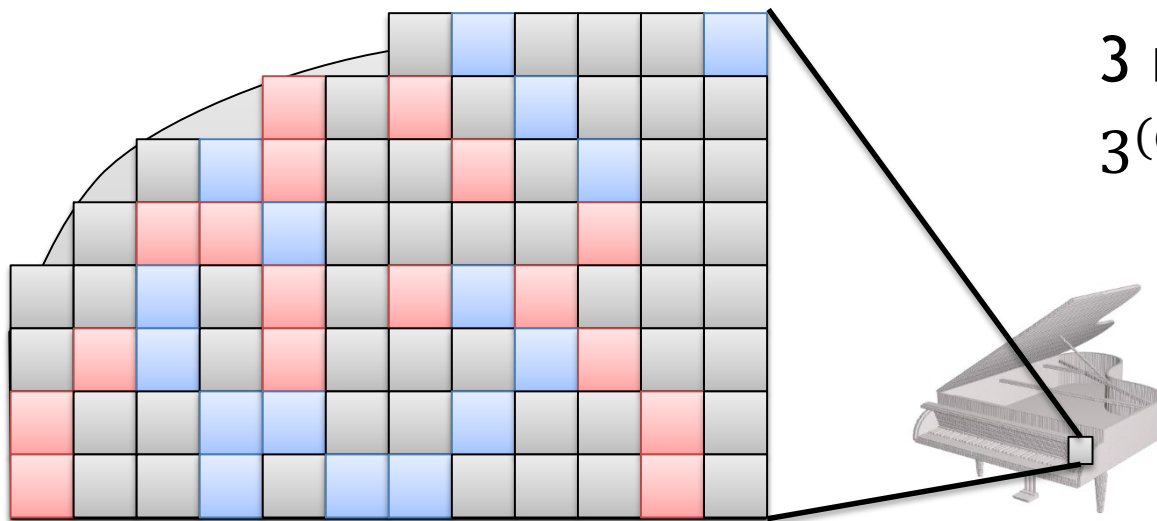


Design Space Representations



Brute Force Inverse is Hard

- Try all per-voxel combinations



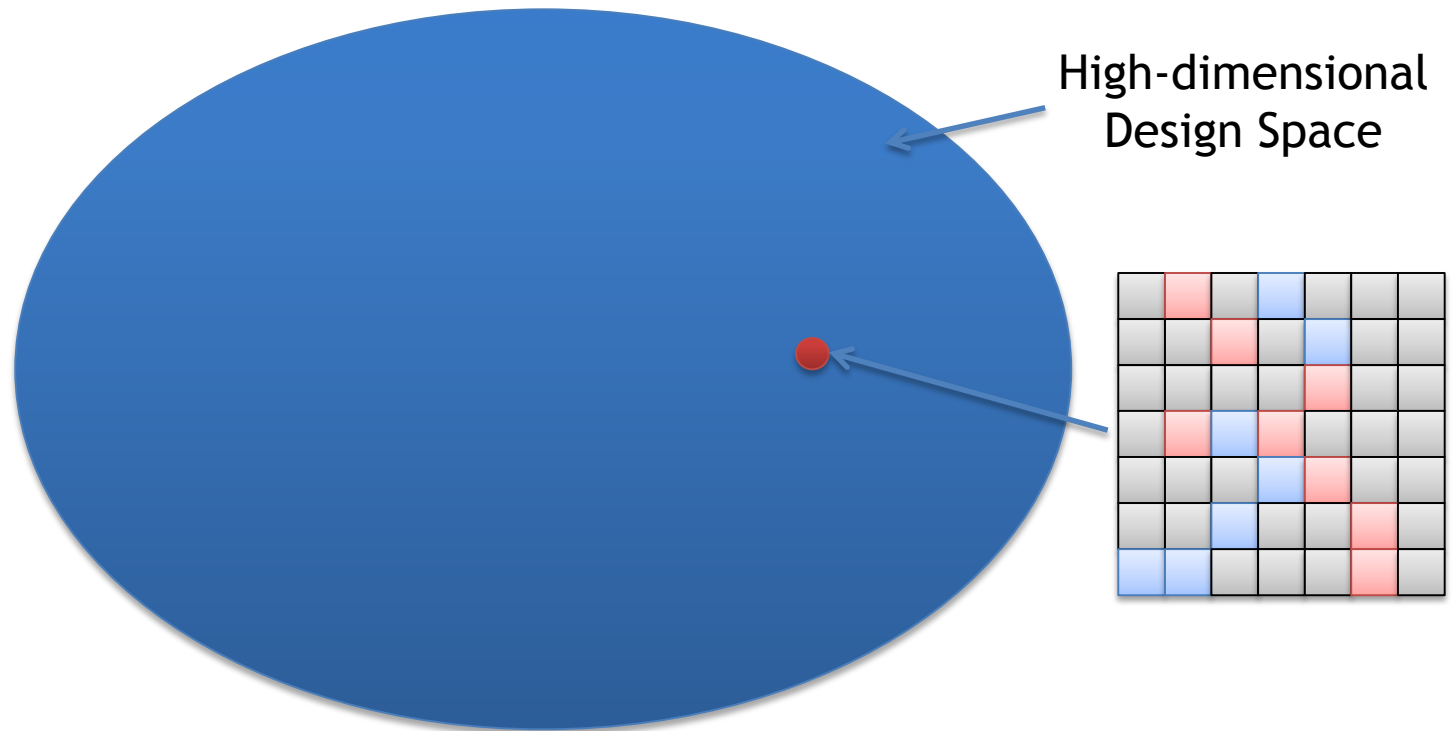
Number of combinations

1 inch cube at 600 dpi

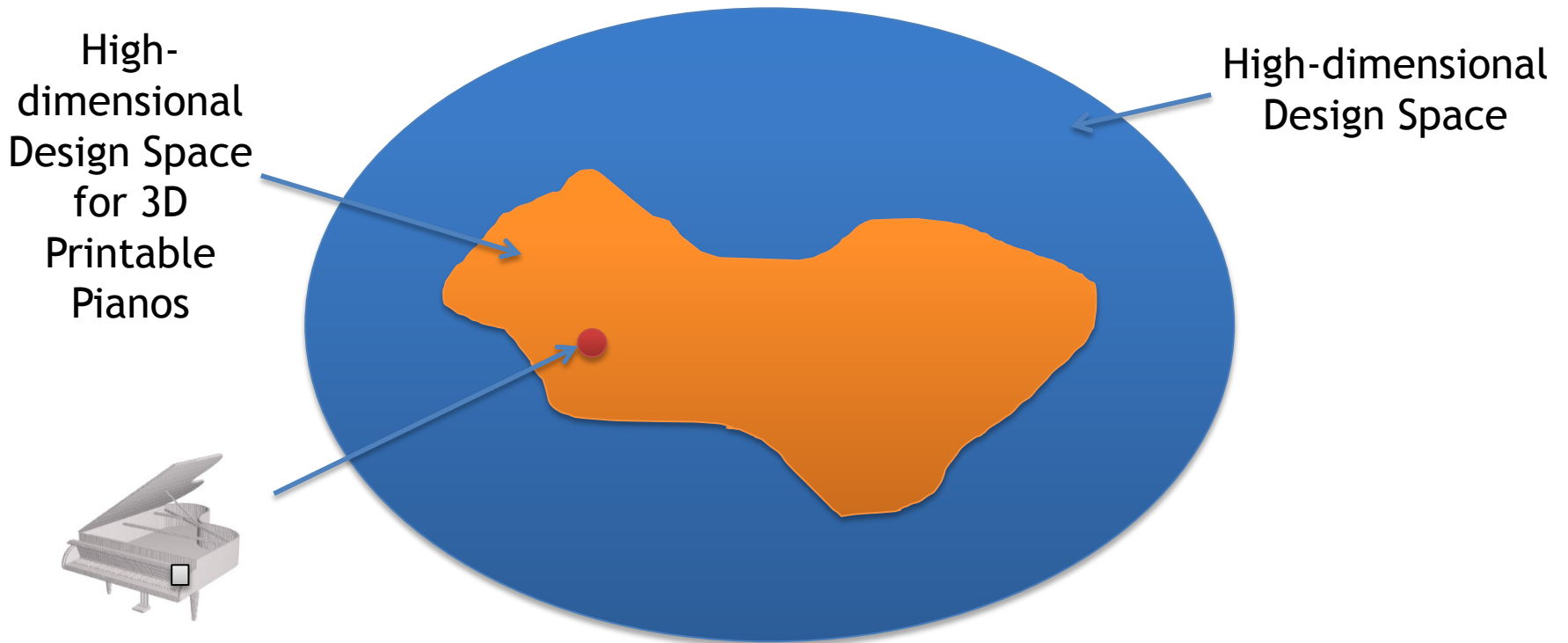
3 materials

$3^{(600^3)}$ combinations

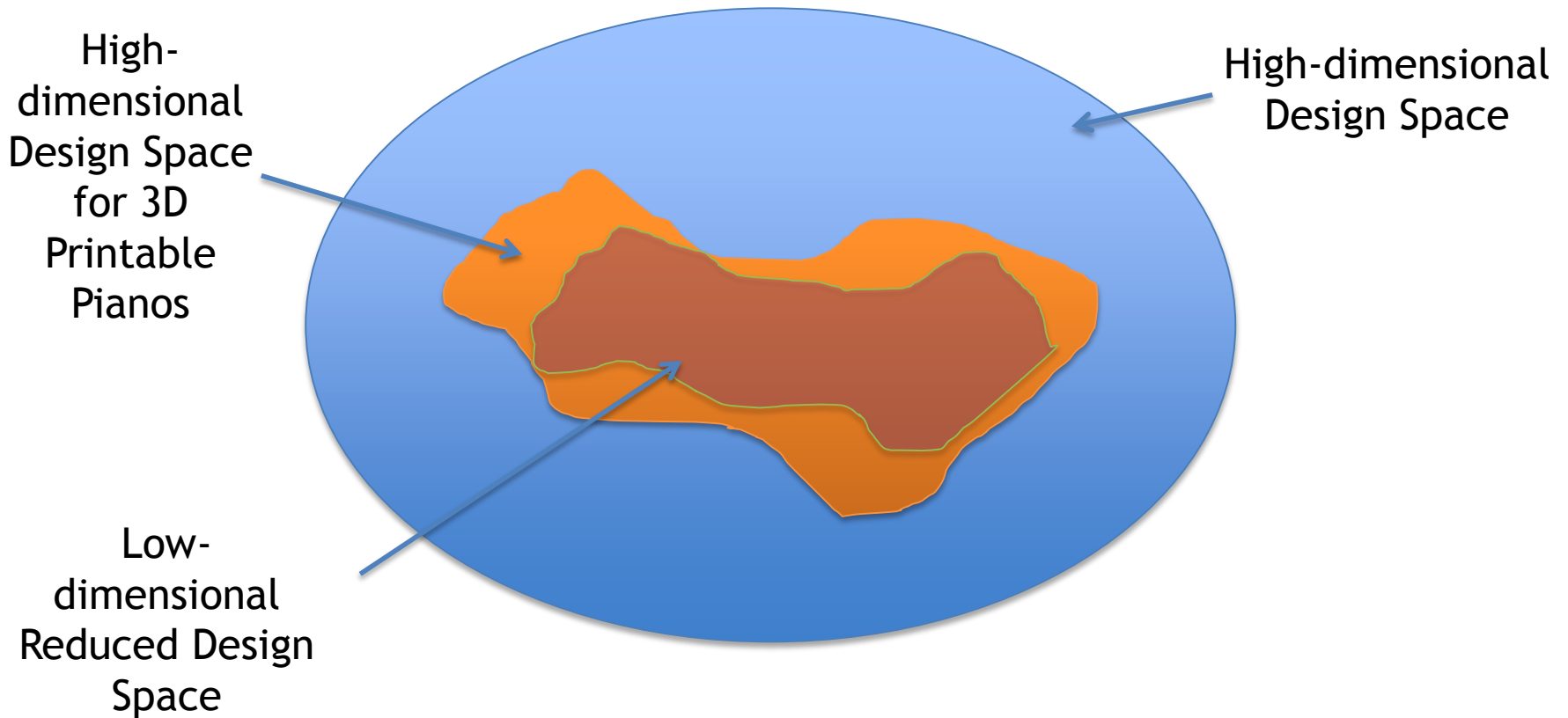
Reduced Parameters



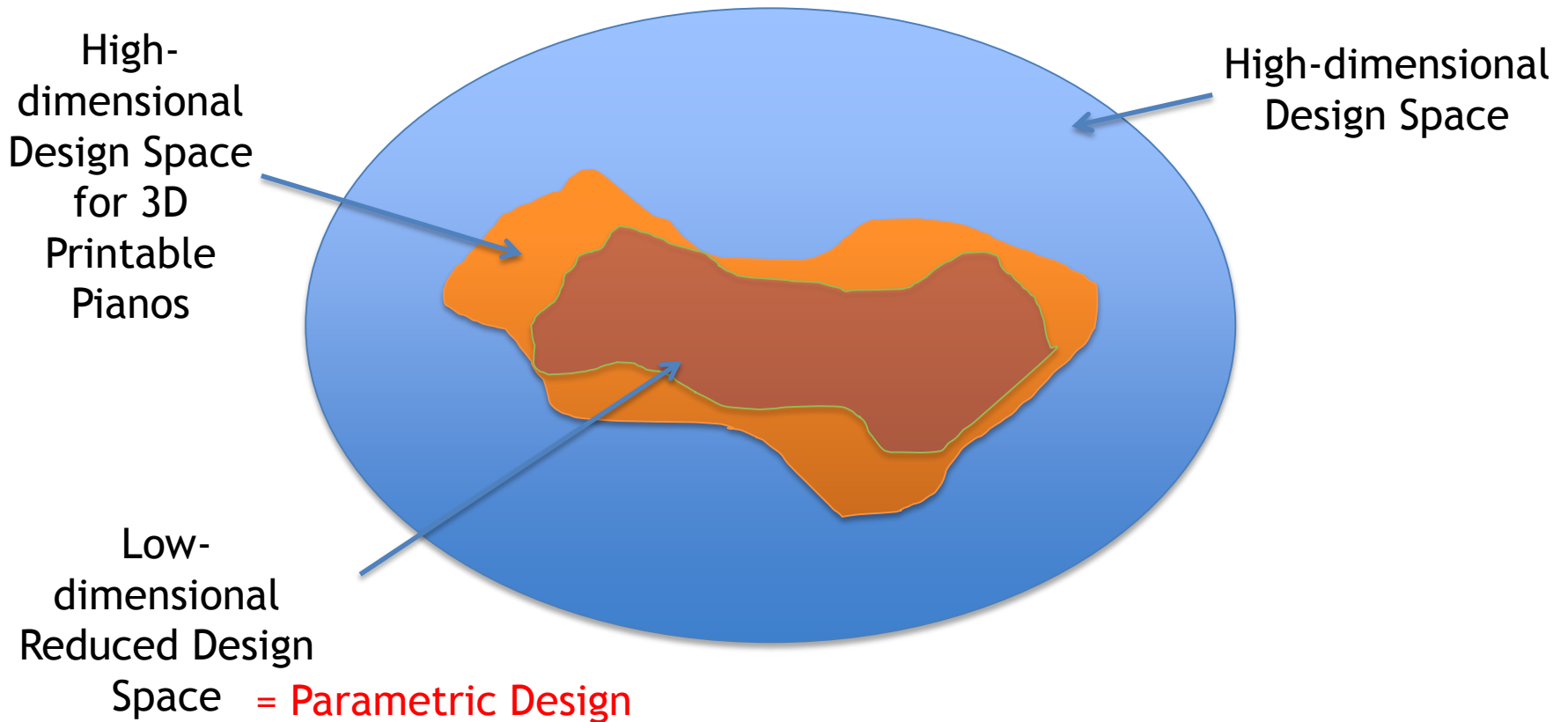
Reduced Parameters



Reduced Parameters



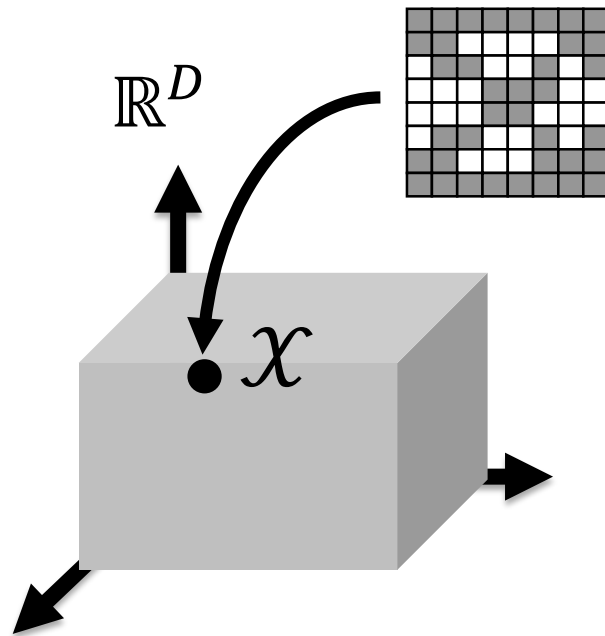
Reduced Parameters





Design Space

- Each design can be mathematically represented as a point in $\mathbb{R}^D$, where D = number of voxels in a build volume

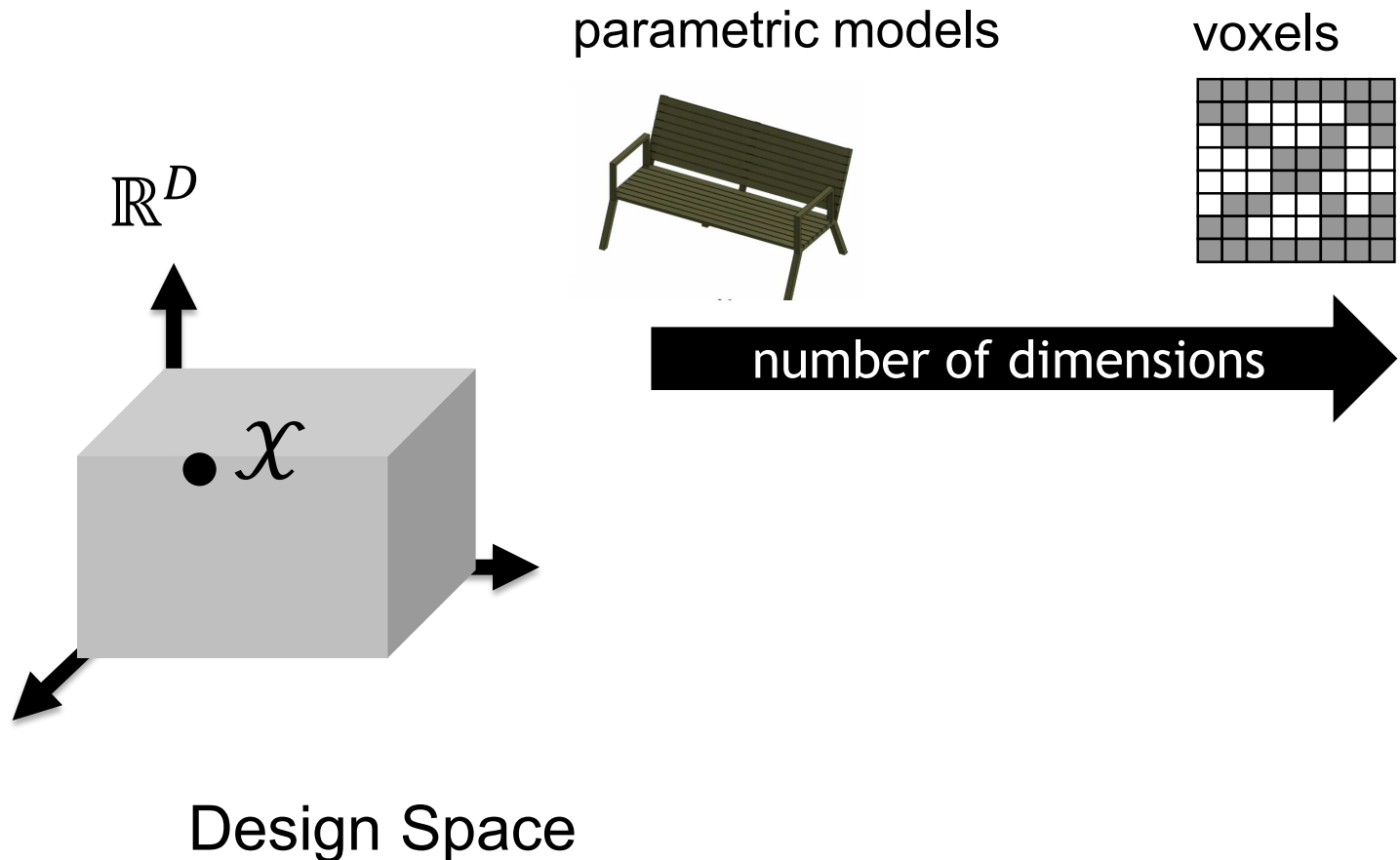


Design Space



Reducing Design Space

- Each design can be mathematically represented as a point in $\mathbb{R}^D$





Reduced Parameters

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

parameters

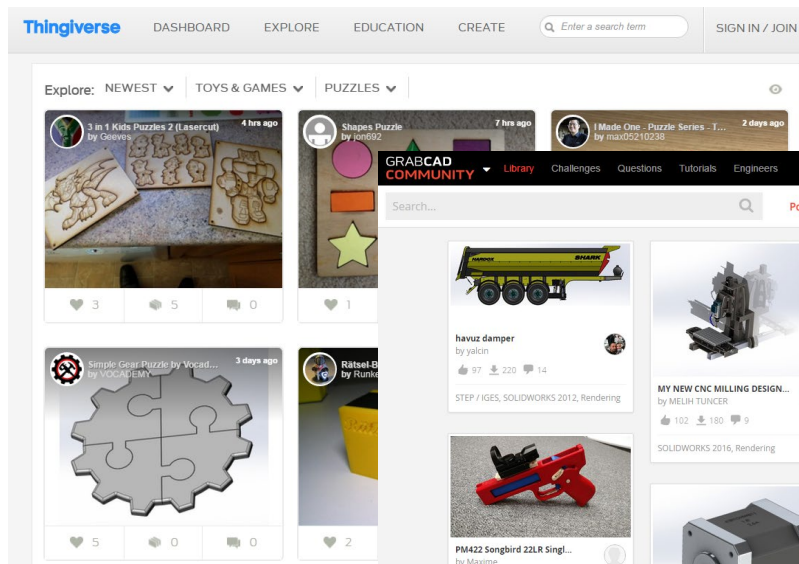


**Parametric
Design**

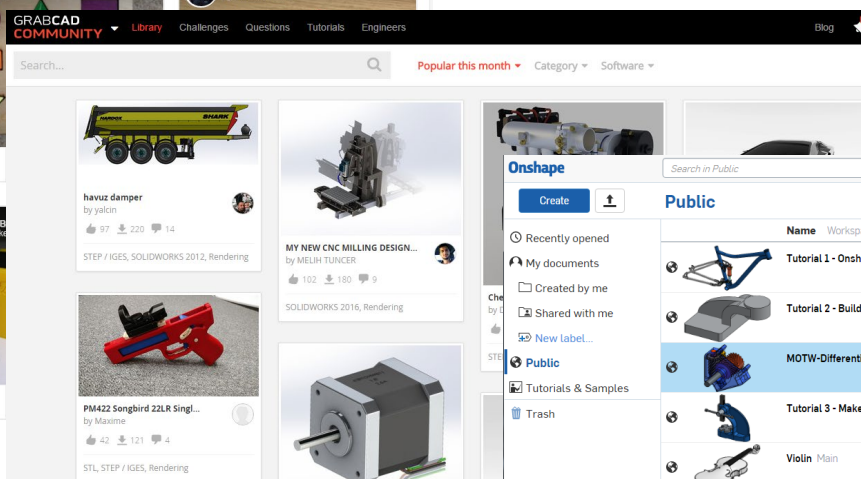




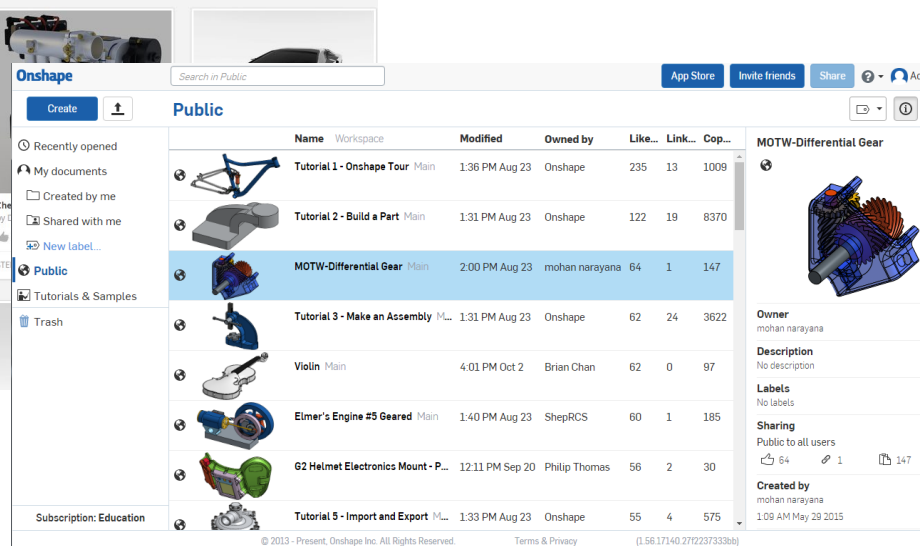
CAD Shape Collections



Thingiverse



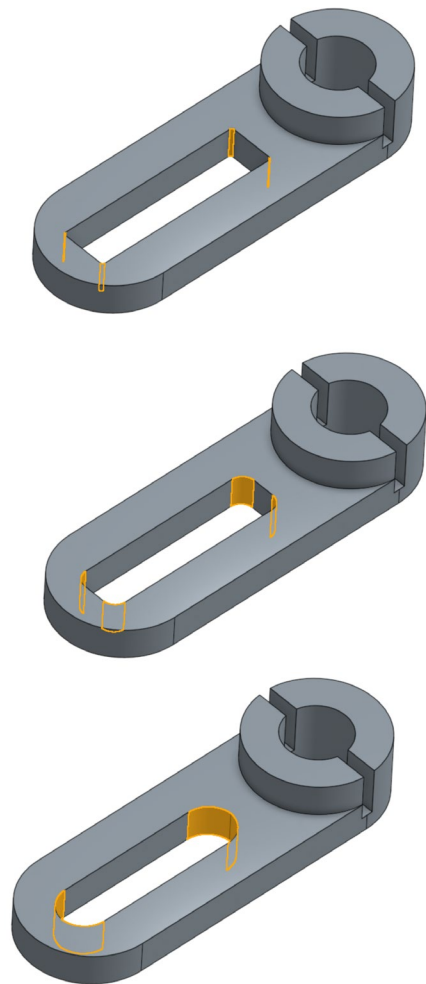
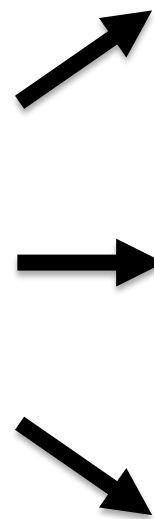
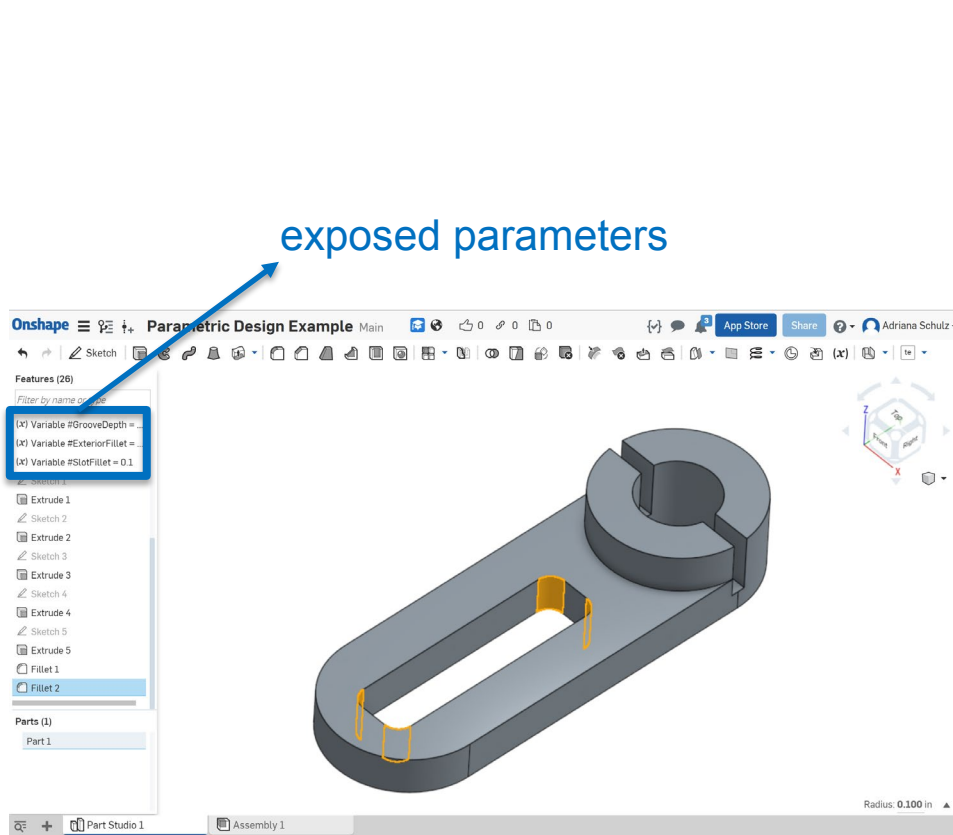
GrabCAD



Onshape's Public Repository

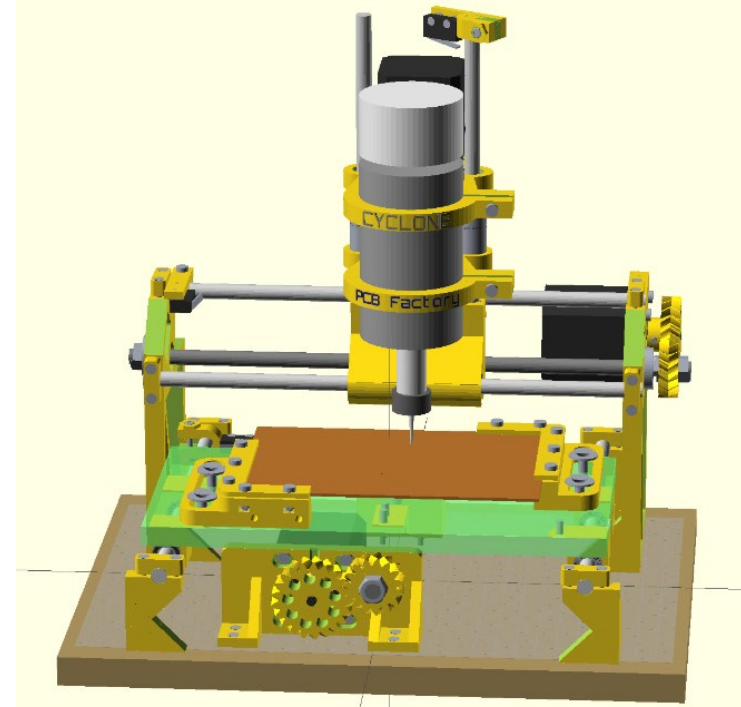


Expert Designed Parameters



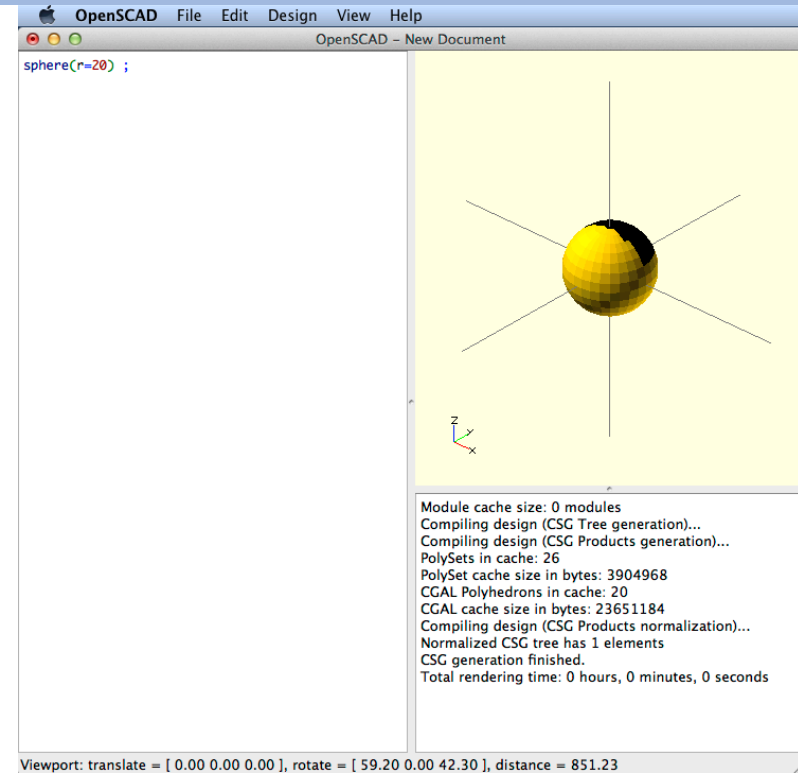
OpenSCAD

- Software for creating solid 3D CAD models
- Not an interactive modeler
 - Very basic UI
- A 3D-compiler
 - Geometry written as a script
 - Executed using CGAL/OpenCSG
 - Rendered with OpenGL
- Available for Linux/UNIX, Windows, Mac OS X
 - <http://www.openscad.org>



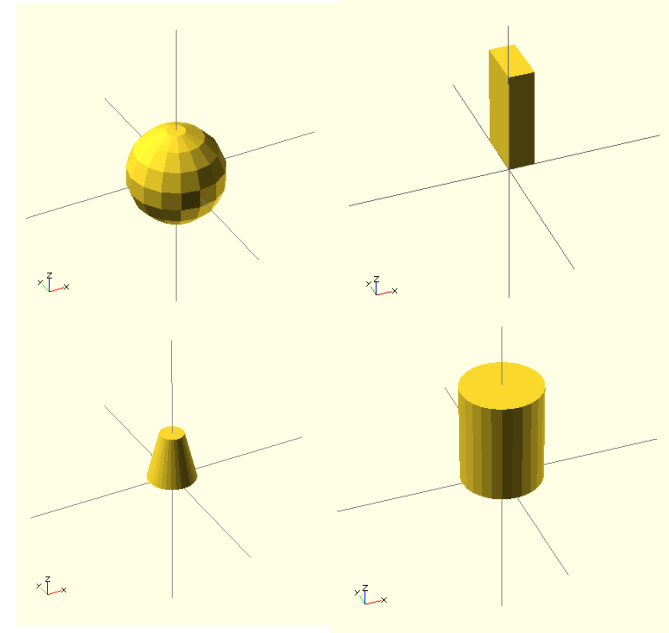
OpenSCAD

- Interface
 - 3 panels
 - Script
 - View
 - Info
- Compile (F5)
 - Design->Compile
- Show Axes (Ctrl+2)



3D Primitives

- Sphere
 - `sphere(5); sphere(r=5);`
- Cube
 - `cube(5);`
 - `cube([4,8,16]);`
- Cylinder
 - `cylinder(20,10,5);`
`cylinder(h = 20, r1 = 10, r2 = 5);`
 - `cylinder(h=20,r=10);`



Transformations

- Translate

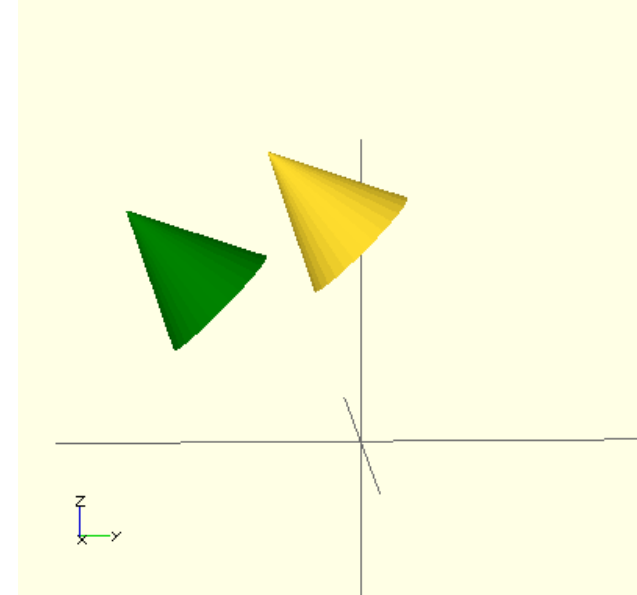
- e.g., `translate([10,0,0])`
`sphere(5); // translate along`
`x axis`

- Rotate

- Scale

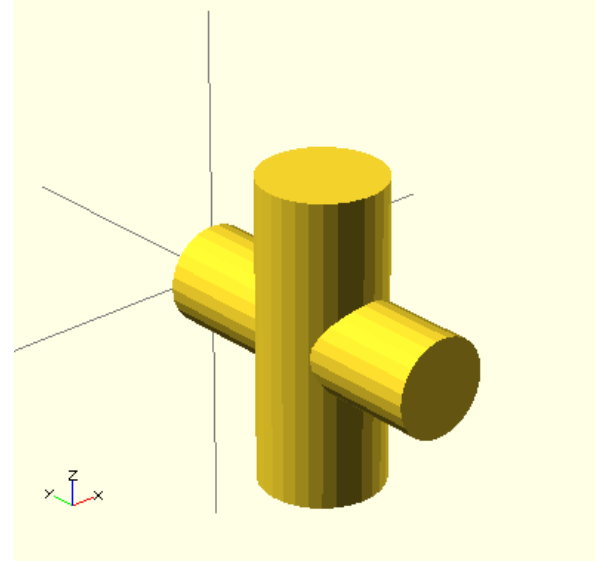
- Order dependent

- `translate([0,0,10])`
`rotate([45,0,0])`
`cylinder([20,10,0]);`
 - `Color("green")`
`rotate([45,0,0])`
`translate([0,0,10])`
`cylinder([20,10,0]);`



- Union
- Intersection
- Difference
- Example:

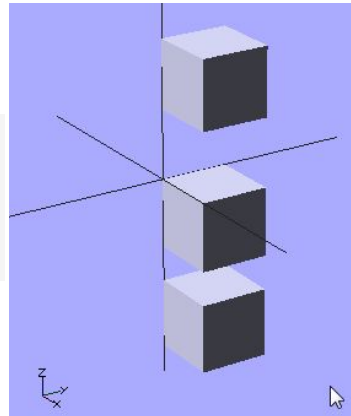
```
union()  
{  
  translate([0,-25,-25]) cylinder(50,10,10);  
  rotate([90,0,0]) cylinder(50,8,8);  
}
```



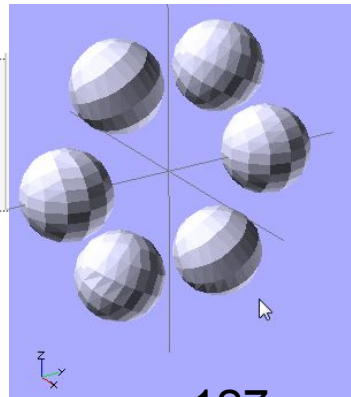
Loops

```
for (loop_variable_name = range or vector) {  
    ...  
}
```

```
for ( z = [-1, 1, -2.5]) {  
    translate( [0, 0, z] )  
    cube(size = 1, center = false);  
}
```



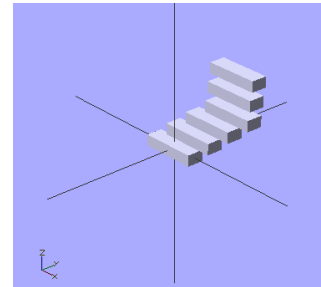
```
for ( i = [0:5] ) {  
    rotate( i*360/6, [1, 0, 0])  
    translate( [0, 10, 0] ) sphere(r = 1);  
}
```



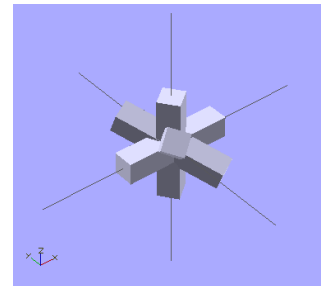


Loops

```
for(i = [ [ 0, 0, 0],  
          [10, 12, 10],  
          [20, 24, 20],  
          [30, 36, 30],  
          [20, 48, 40],  
          [10, 60, 50] ] )  
{  
    translate(i)  
    cube([50, 15, 10], center = true);  
}
```



```
for(i = [ [ 0, 0, 0],  
          [ 10, 20, 300],  
          [200, 40, 57],  
          [ 20, 88, 57] ] )  
{  
    rotate(i)  
    cube([100, 20, 20], center = true);  
}
```



- Procedures/Functions

```
module leaves() { cylinder(20,5,0); }
```

```
module box() { cube([5,10,15]); }
```

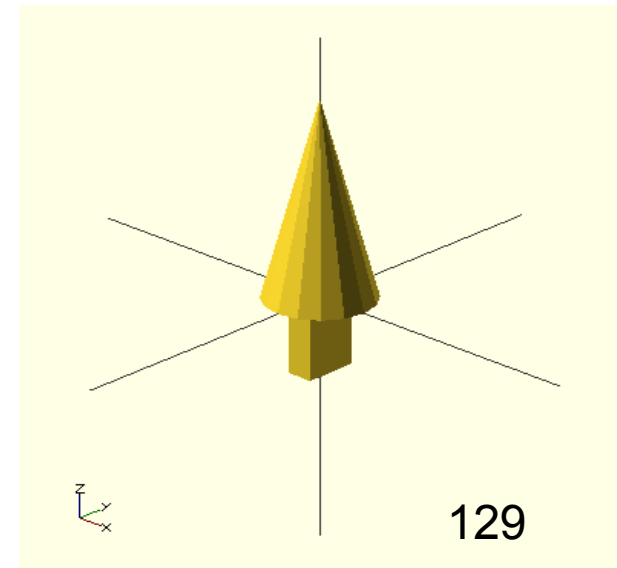
```
module tree() {
```

```
  leaves();
```

```
  scale([0.5,0.5,0.5]) translate([-2.5,-5,-15]) box();
```

```
}
```

```
tree();
```

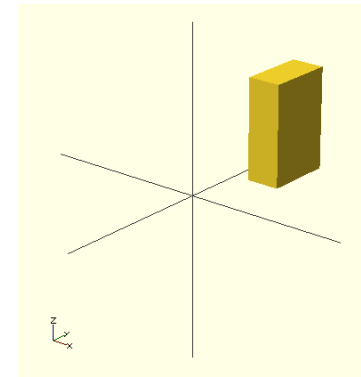


- Parameters

```
module box(w,l,h,tx,ty,tz){  
    translate([tx,ty,tz])  
    cube([w,l,h]);  
}  
box(5,10,15,10,0,5);
```

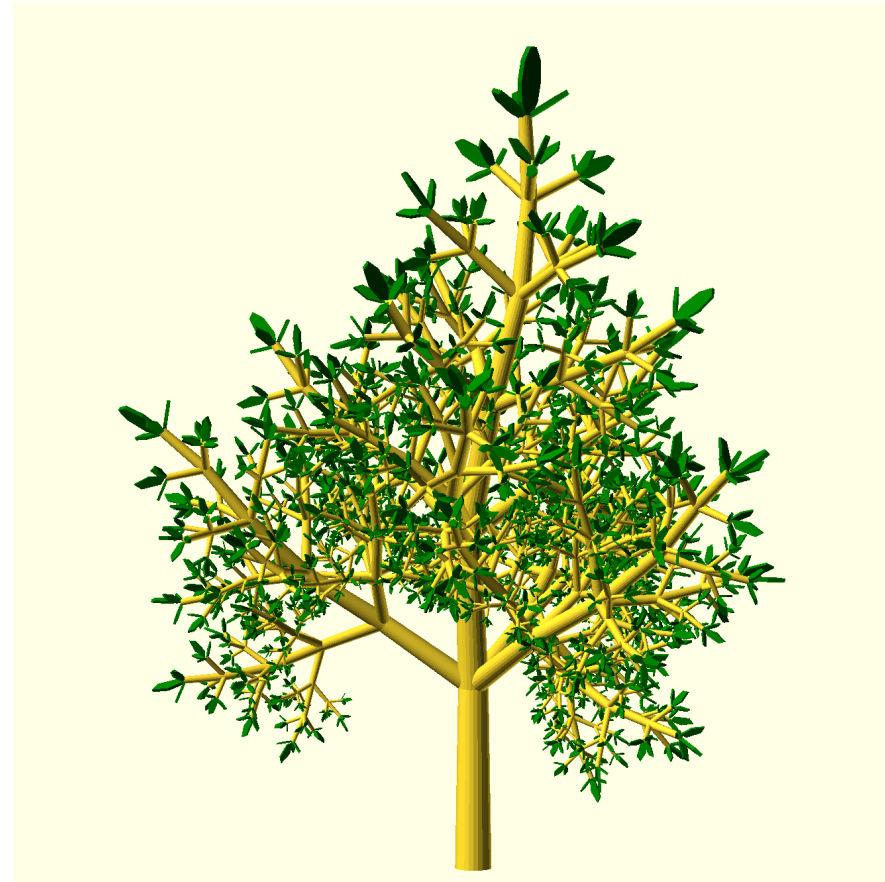
- Default values

```
module box2(w=5,l=10,h=20){  
    echo("w=", w, " l=", l, " h=", h);  
    cube([w,l,h]);  
}  
box2();
```



Parametric Model in OpenSCAD

```
module simple_tree(size, dna, n) {  
  if (n > 0) {  
    // trunk  
    cylinder(r1=size/10, r2=size/12, h=size, $fn=24);  
    // branches  
    translate([0,0,size])  
    for(bd = dna) {  
      angx = bd[0];  
      angz = bd[1];  
      scal = bd[2];  
      rotate([angx,0,angz])  
      simple_tree(scal*size, dna, n-1);  
    }  
  }  
  else // leaves  
    color("green")  
    scale([1,1,3])  
    translate([0,0,size/6])  
    rotate([90,0,0])  
    cylinder(r=size/6,h=size/10);  
}  
// dna is a list of branching data bd of the tree:  
//   bd[0] - inclination of the branch  
//   bd[1] - Z rotation angle of the branch  
//   bd[2] - relative scale of the branch  
dna = [ [12, 80, 0.85], [55, 0, 0.6],  
        [62, 125, 0.6], [57, -125, 0.6] ];  
simple_tree(50, dna, 6);
```



L-system

- nonterminals : 0, 1
- terminals : [,], +, -
- start : 0
- rules : $(1 \rightarrow 11), (0 \rightarrow 1[+0][-0])$
- Visual representation: turtle graphics
 - 0: draw a line segment ending in a leaf
 - 1: draw a line segment
 - - turn left 45 degrees
 - + turn right 45 degrees
 - [: push position and angle,
 -]: pop position and angle,

Iterations:

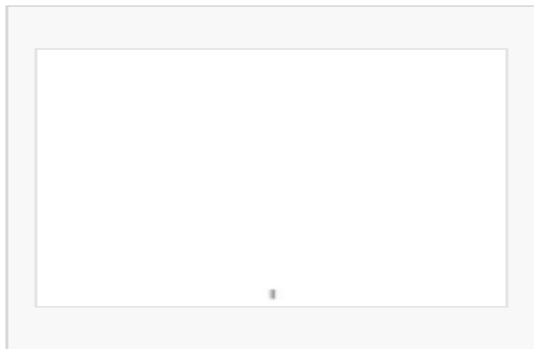
0

1[+0][-0]

11[+1[+0][-0]][-1[+0][-0]]

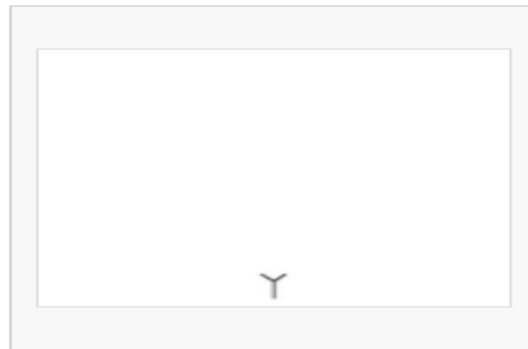
L-system

- nonterminals : 0, 1
- terminals : [,], +, -
- start : 0
- rules : $(1 \rightarrow 11), (0 \rightarrow 1[+0][-0])$
- Visual representation: turtle graphics
 - 0: draw a line segment ending in a leaf
 - 1: draw a line segment
 - - turn left 45 degrees
 - + turn right 45 degrees
 - [: push position and angle,
 -]: pop position and angle,



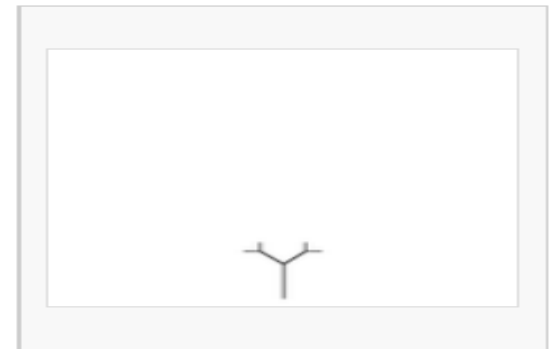
Axiom

$n = 0$



First recursion

$n = 1 : 1[+0][-0]$



Second recursion

$n = 2 : 11[+1[+0][-0]][-1[+0][-0]]$

L-systems



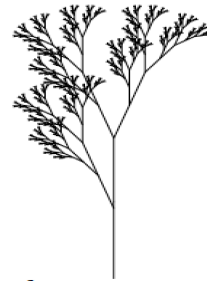
a
 $n=5, \delta=25.7^\circ$
 F
 $F \rightarrow F[+F]F[-F]F$



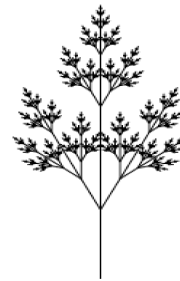
b
 $n=5, \delta=20^\circ$
 F
 $F \rightarrow F[+F]F[-F][F]$



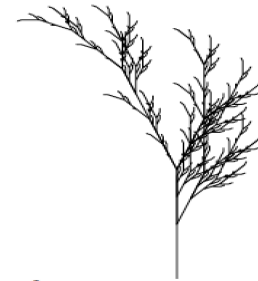
c
 $n=4, \delta=22.5^\circ$
 F
 $F \rightarrow FF[-F+F+F]+$
 $[+F-F-F]$



d
 $n=7, \delta=20^\circ$
 X
 $X \rightarrow F[+X]F[-X]+X$
 $F \rightarrow FF$

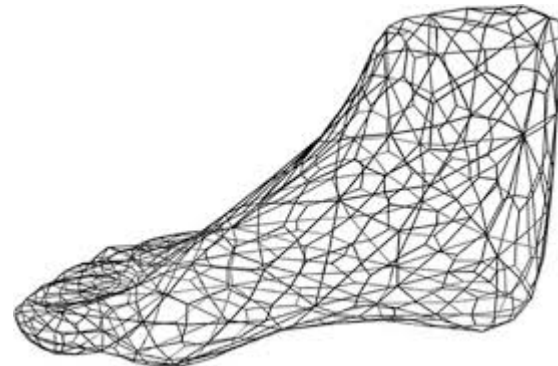
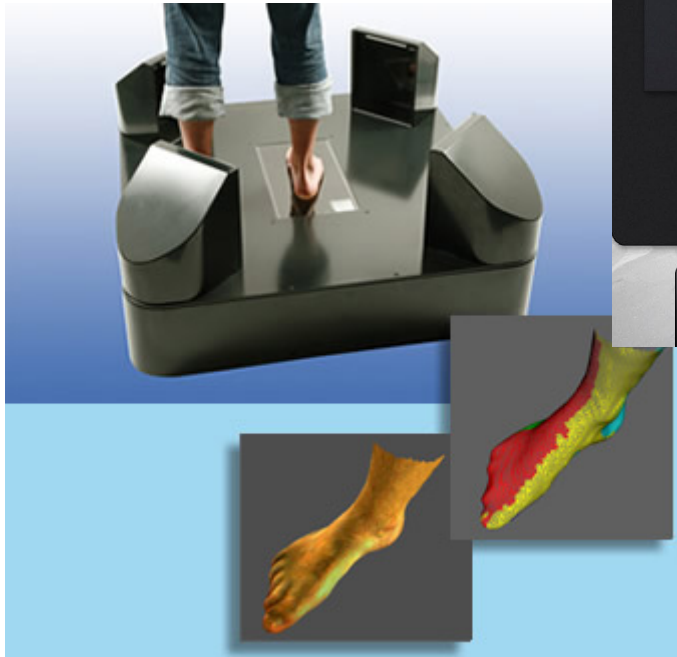


e
 $n=7, \delta=25.7^\circ$
 X
 $X \rightarrow F[+X][-X]FX$
 $F \rightarrow FF$



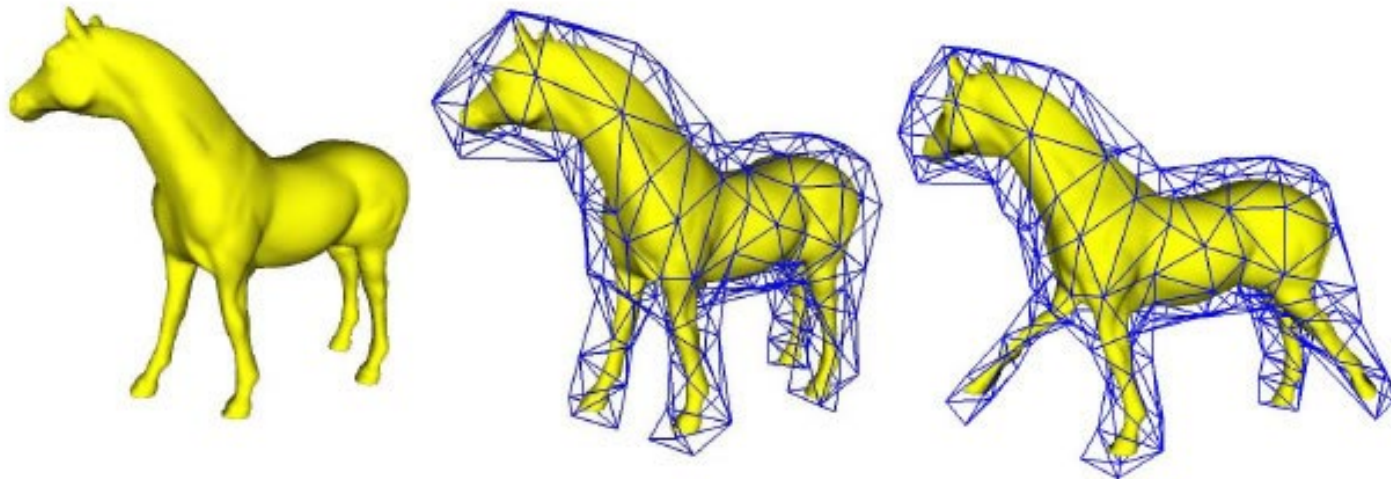
f
 $n=5, \delta=22.5^\circ$
 X
 $X \rightarrow F-[X]+X]+F[+FX]-X$
 $F \rightarrow FF$

CAD Does Not Work For Organic Shapes



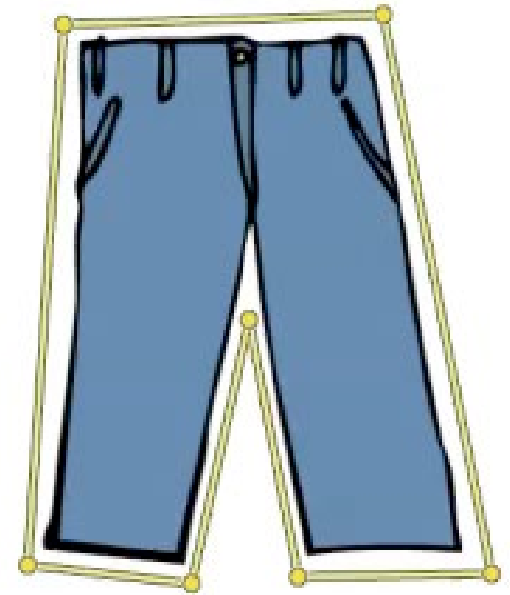
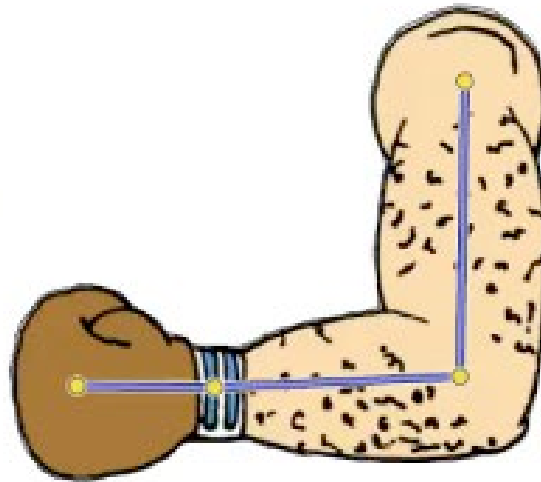
Reduced Parameters

- Mesh vertices
- Mesh deformation “knobs”
 - e.g., cages
- Mesh scaling

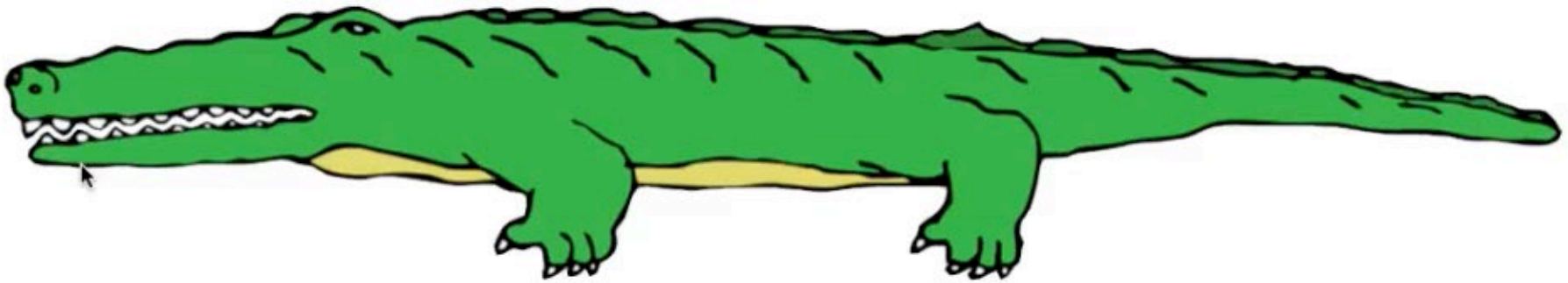




Deformations should be smooth and shape-aware

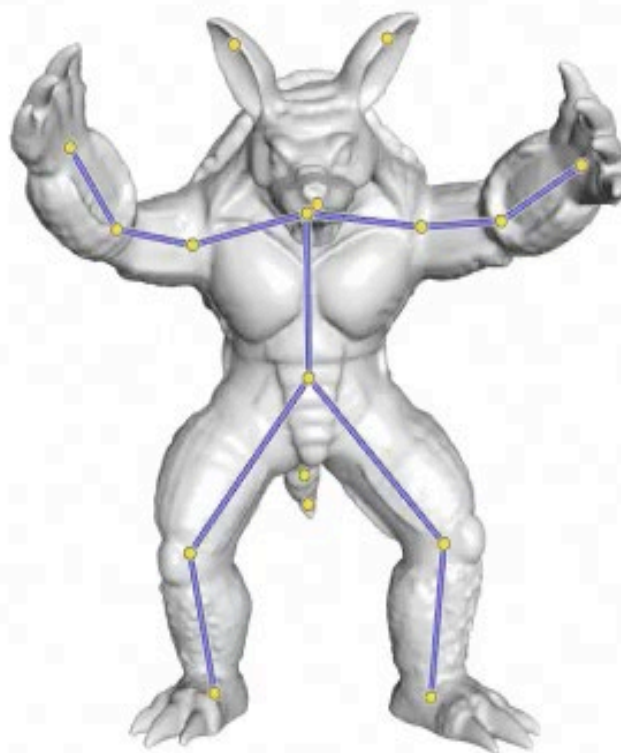


Deformation Methods



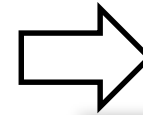
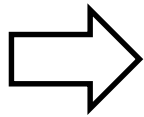


3D Characters



Advanced Applications

Parametrization to find design that are structurally sound

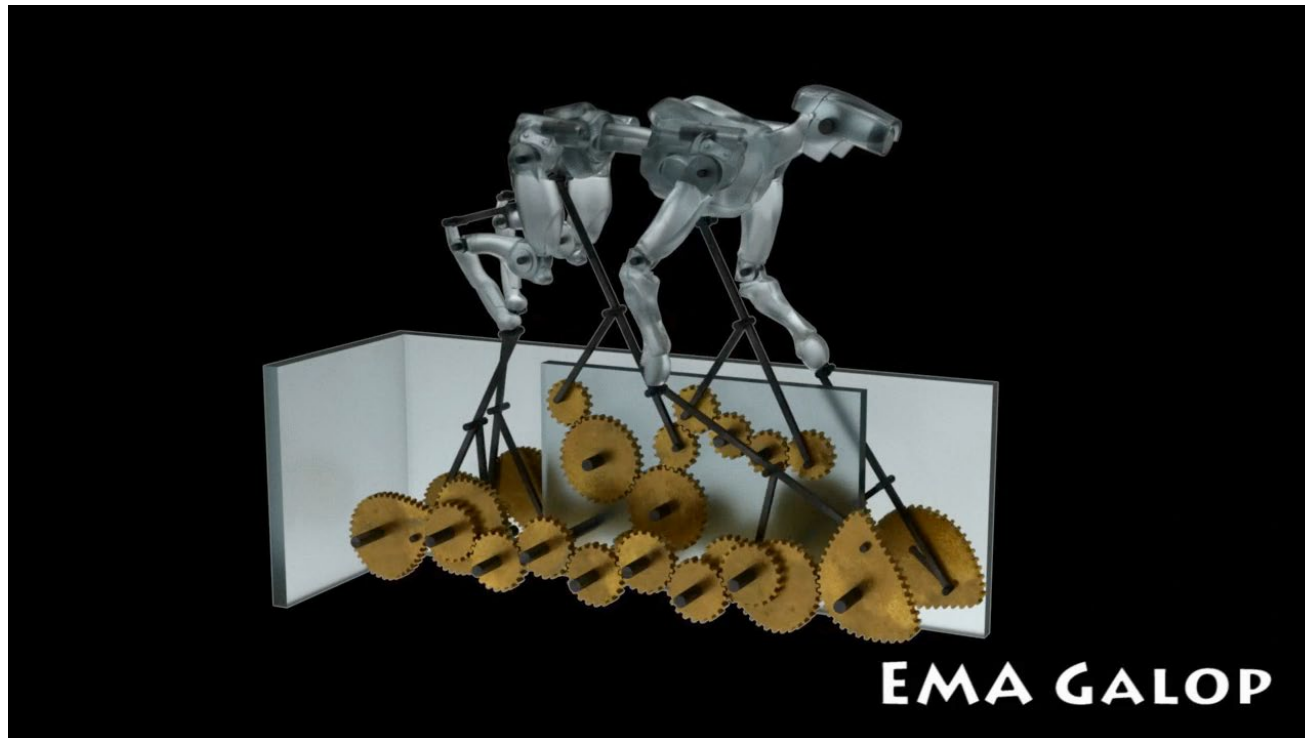


input model

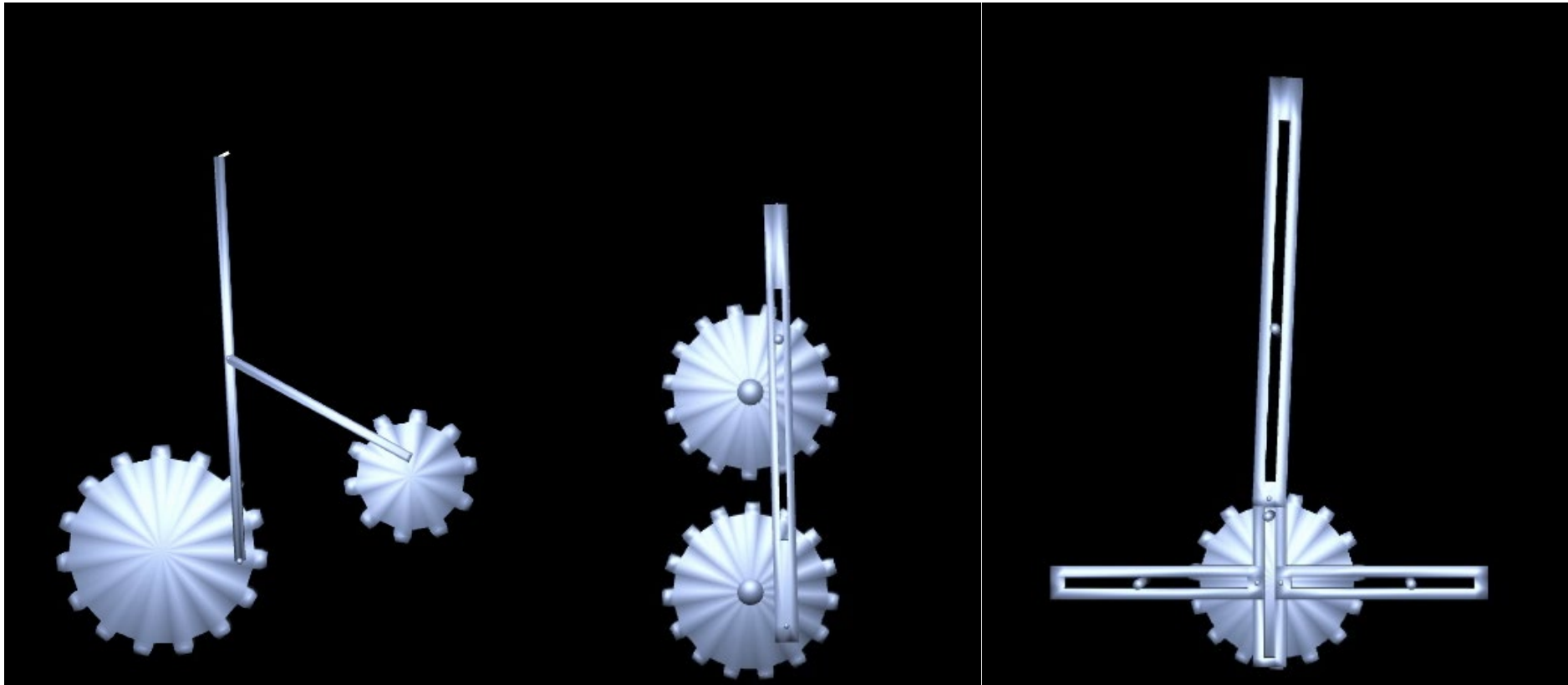
balanced
model

Beyond Geometry

What is the Design Space of Mechanisms?

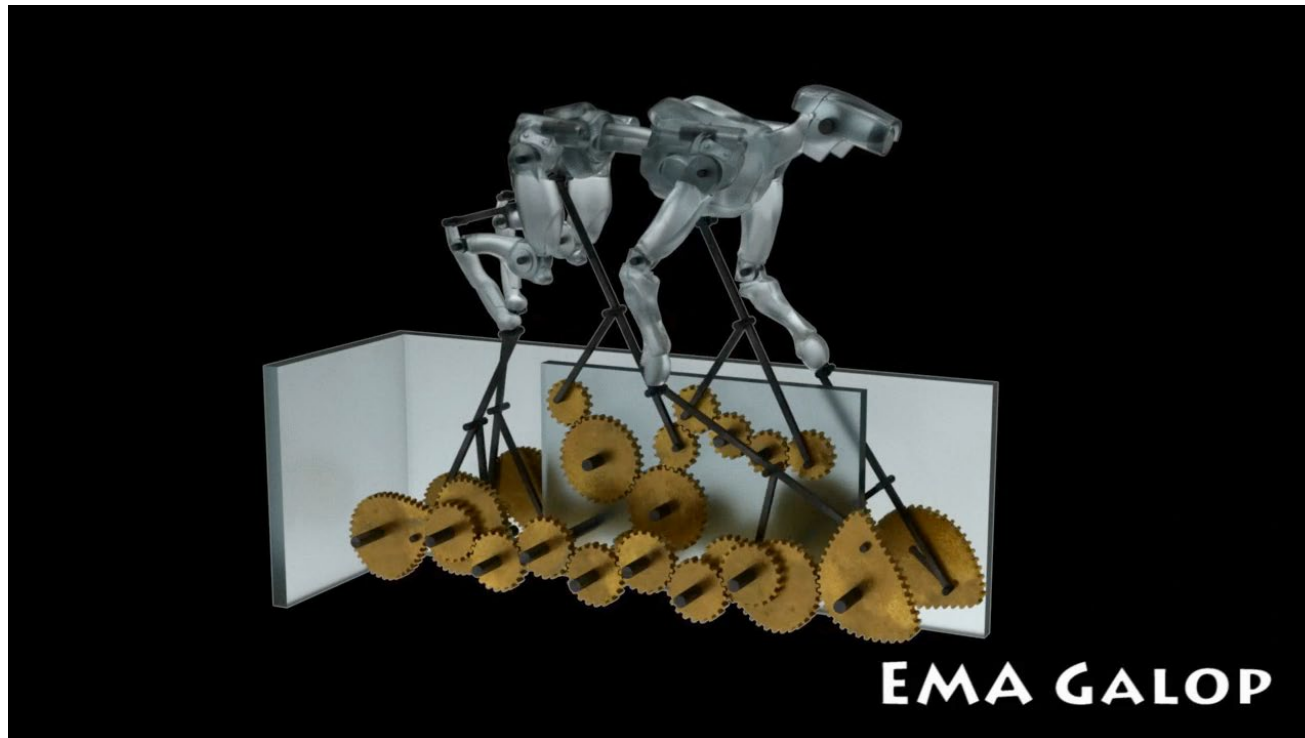


Library of Mechanisms

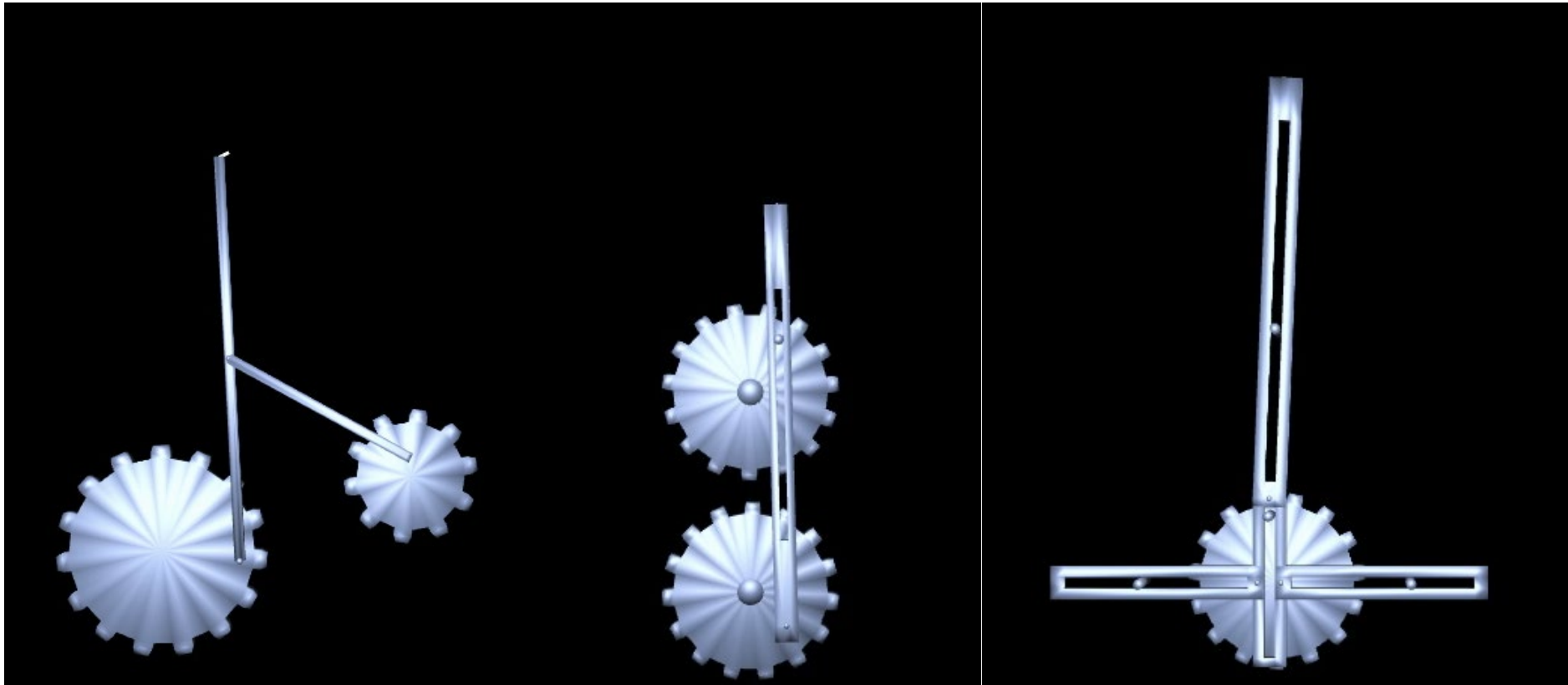


Beyond Geometry

What is the Design Space of Mechanisms?

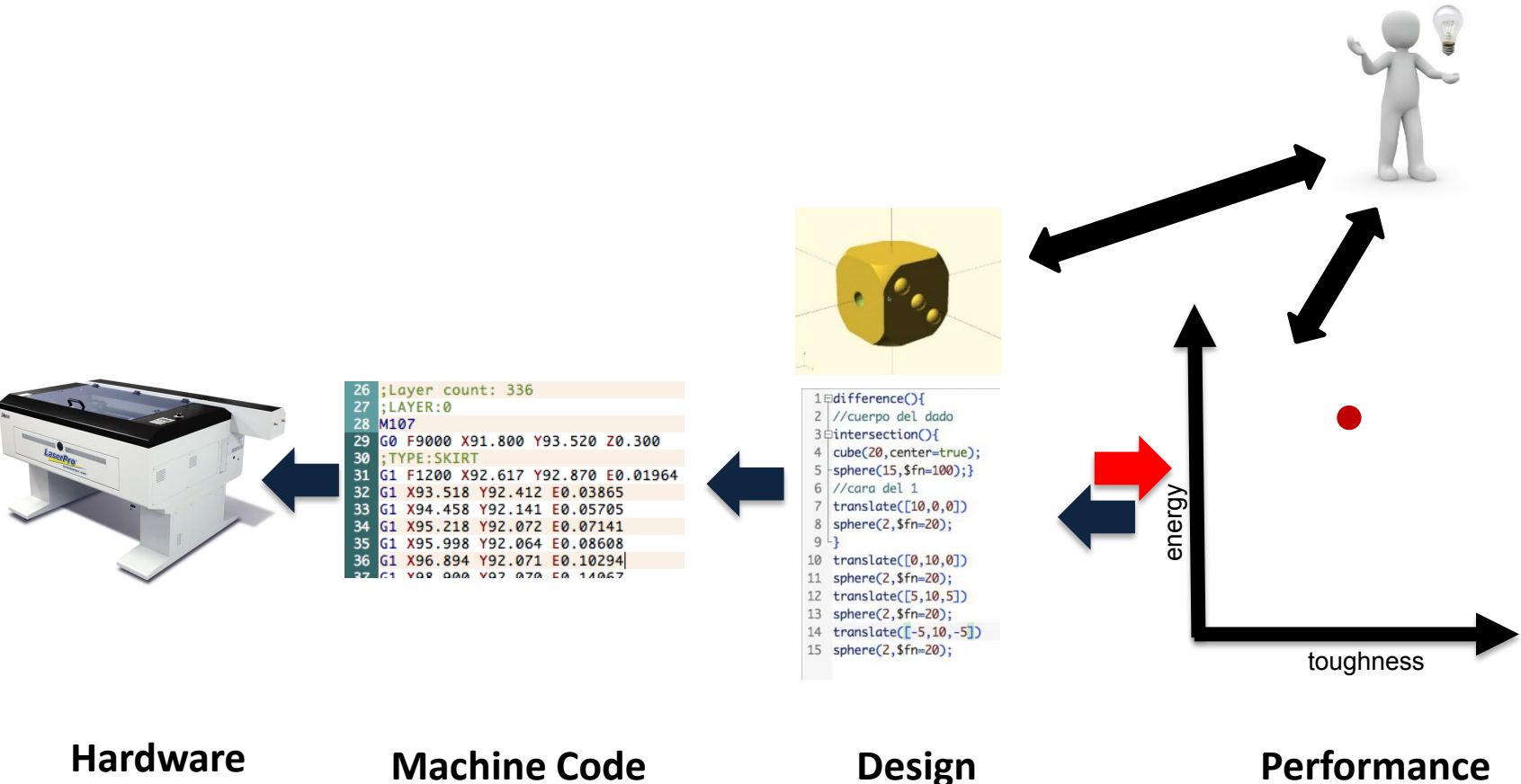


Library of Mechanisms

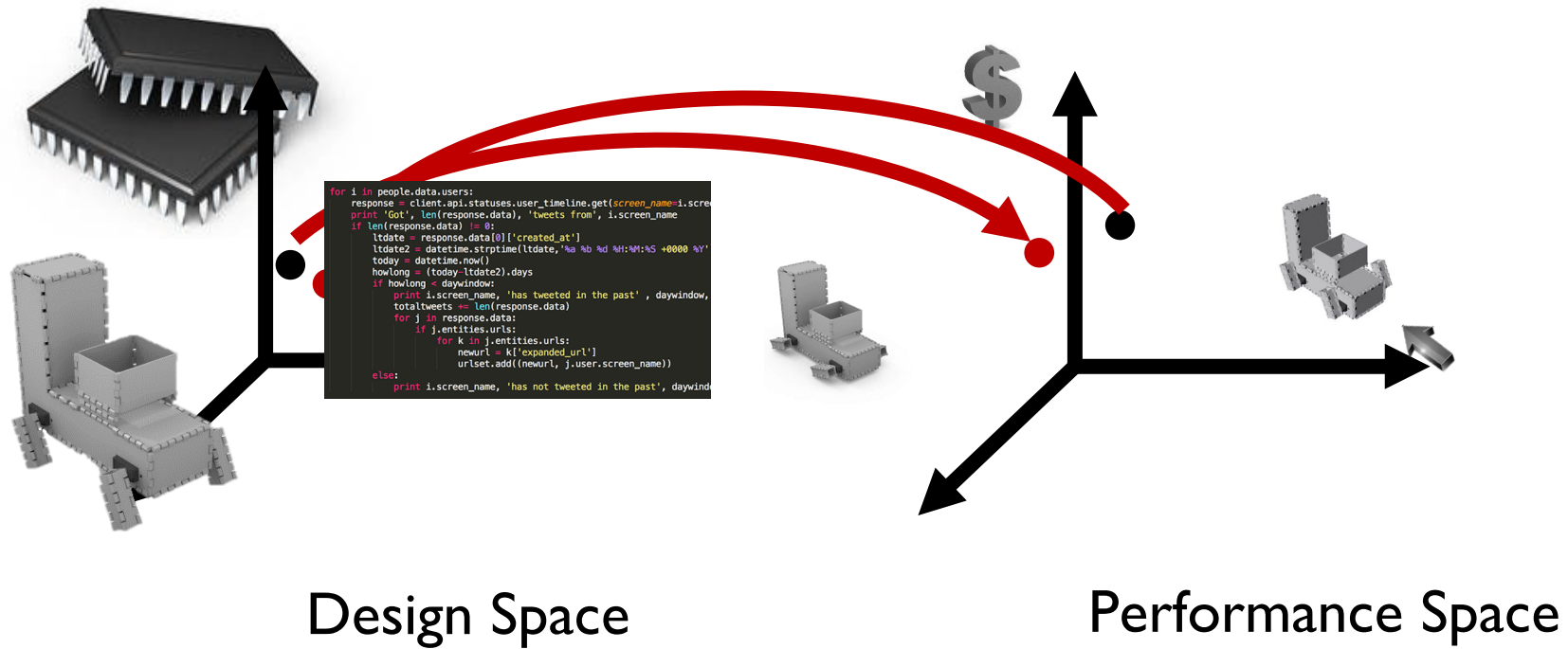


Simulation and Performance-Driven Design

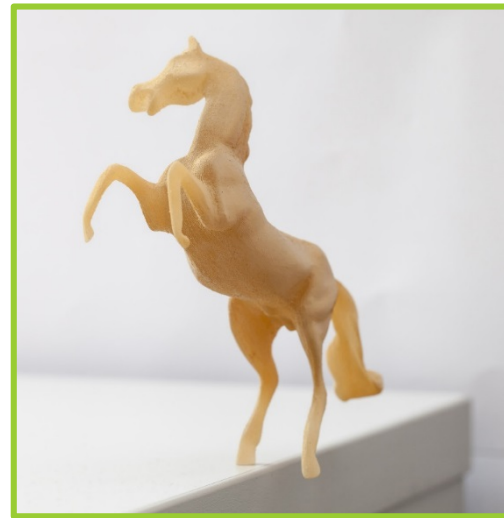
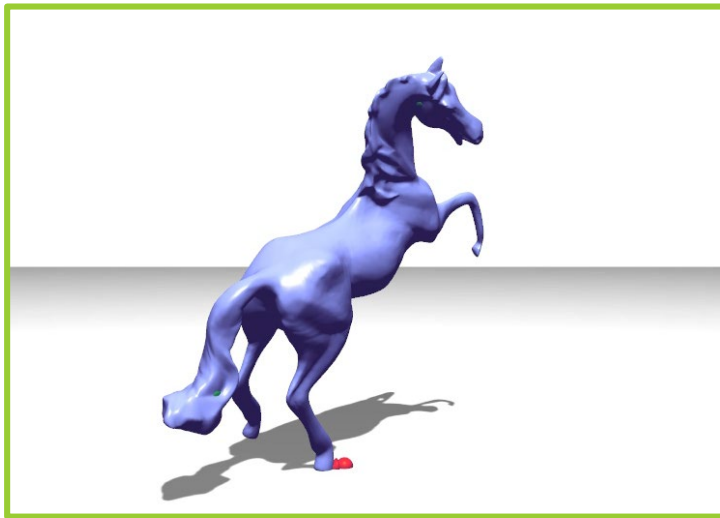
Computational Design and Fabrication Pipeline



Design Driven By Performance



Structural Stability



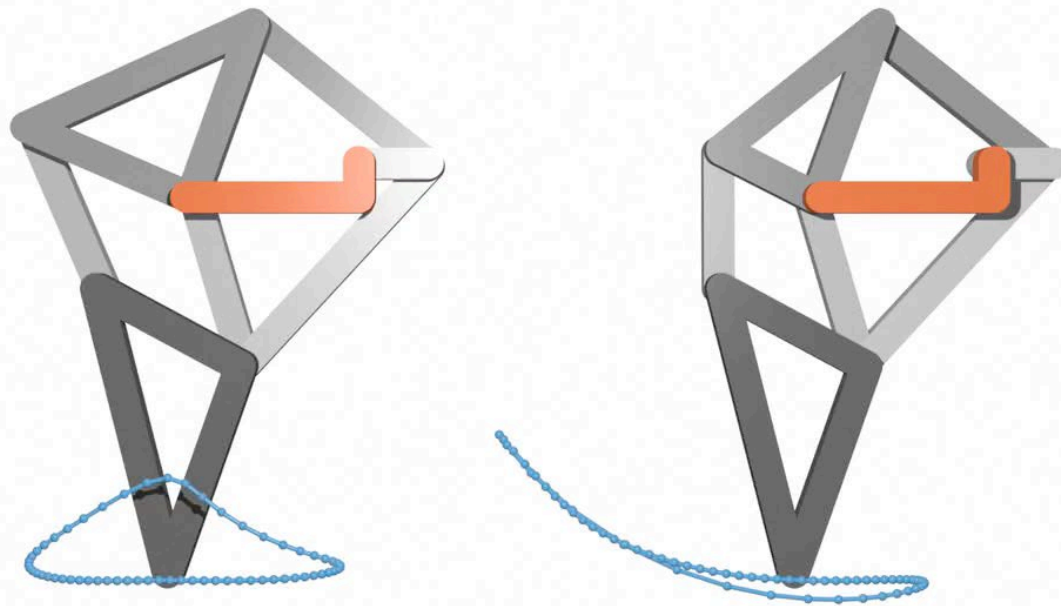
Prevost et al 2013

Worse-Case Structural Analysis



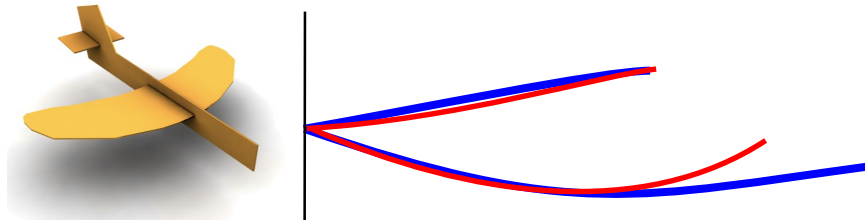
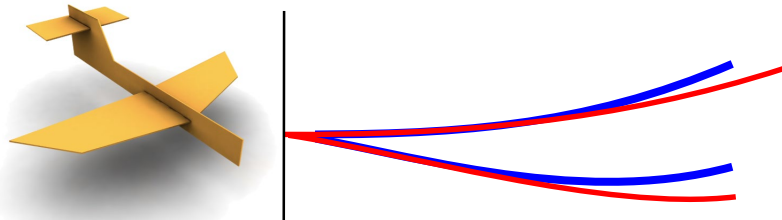
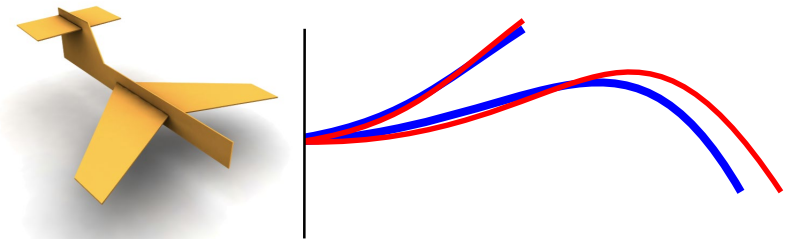
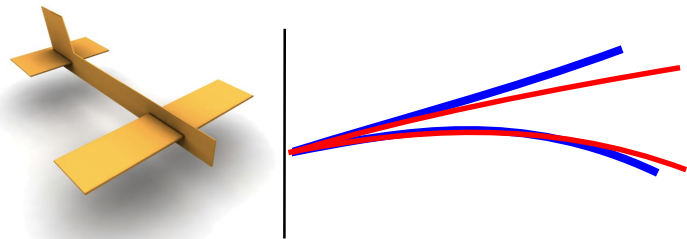
Zhou et al 2013

Simulating Kinematics



Bacher et al 2014

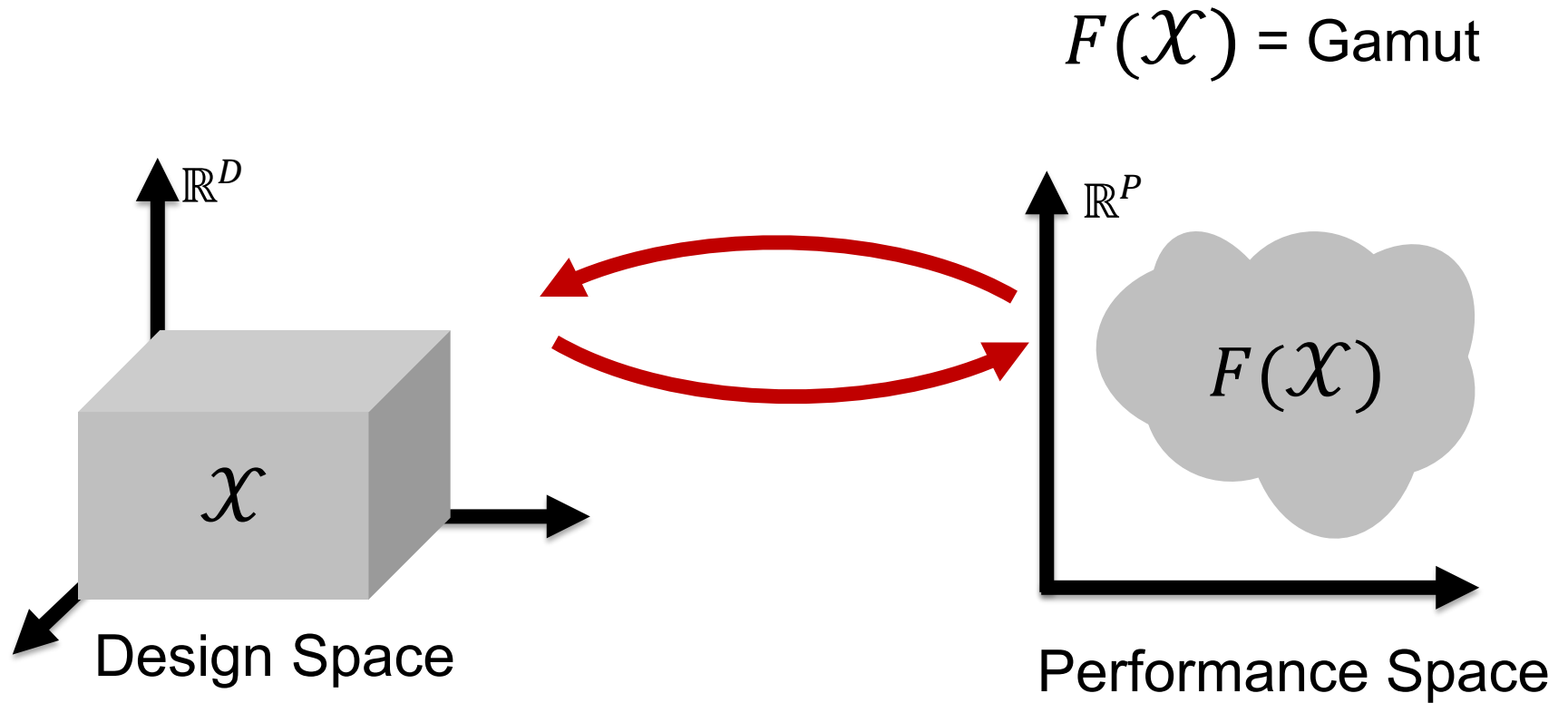
Simulating Aerodynamics



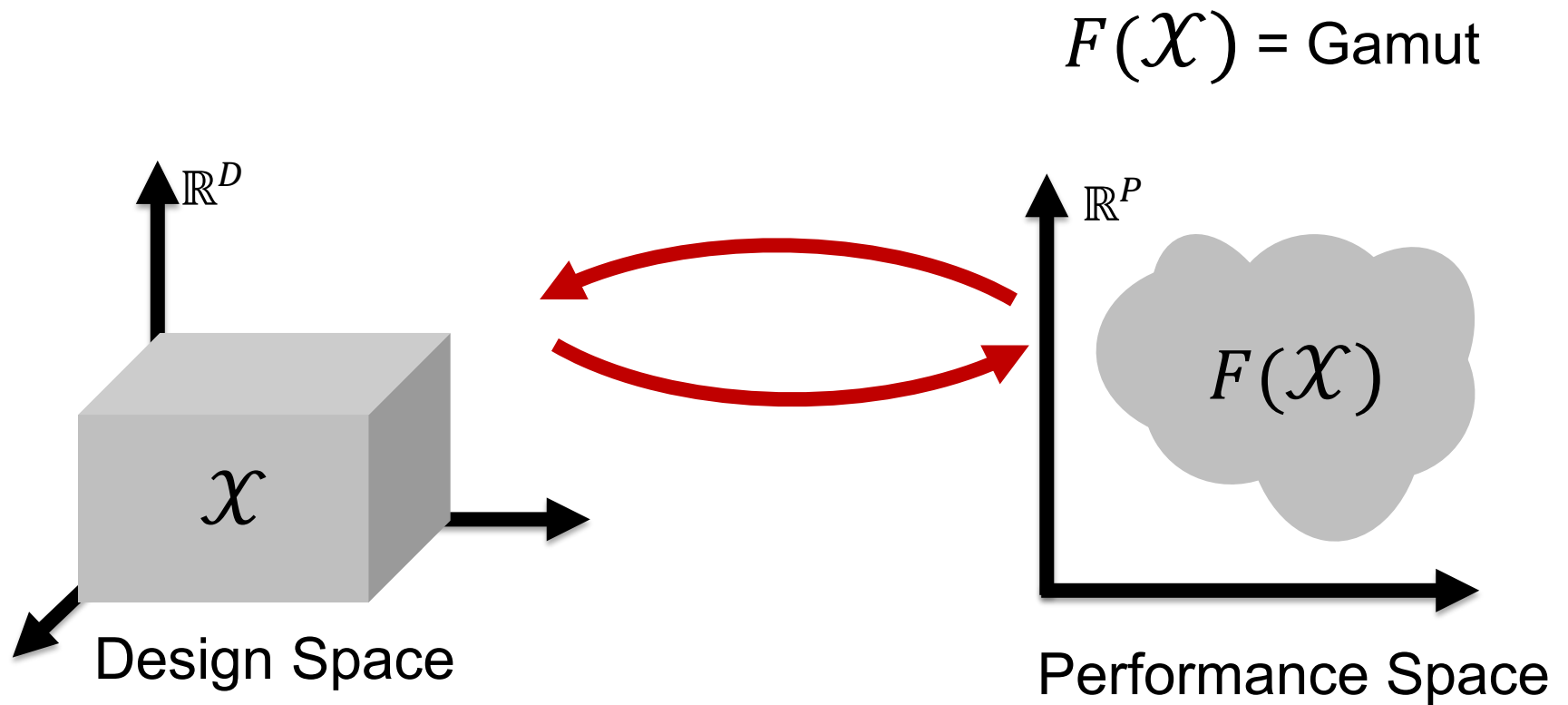
Umetani et al 2014

Performance Space Representations

This Lecture

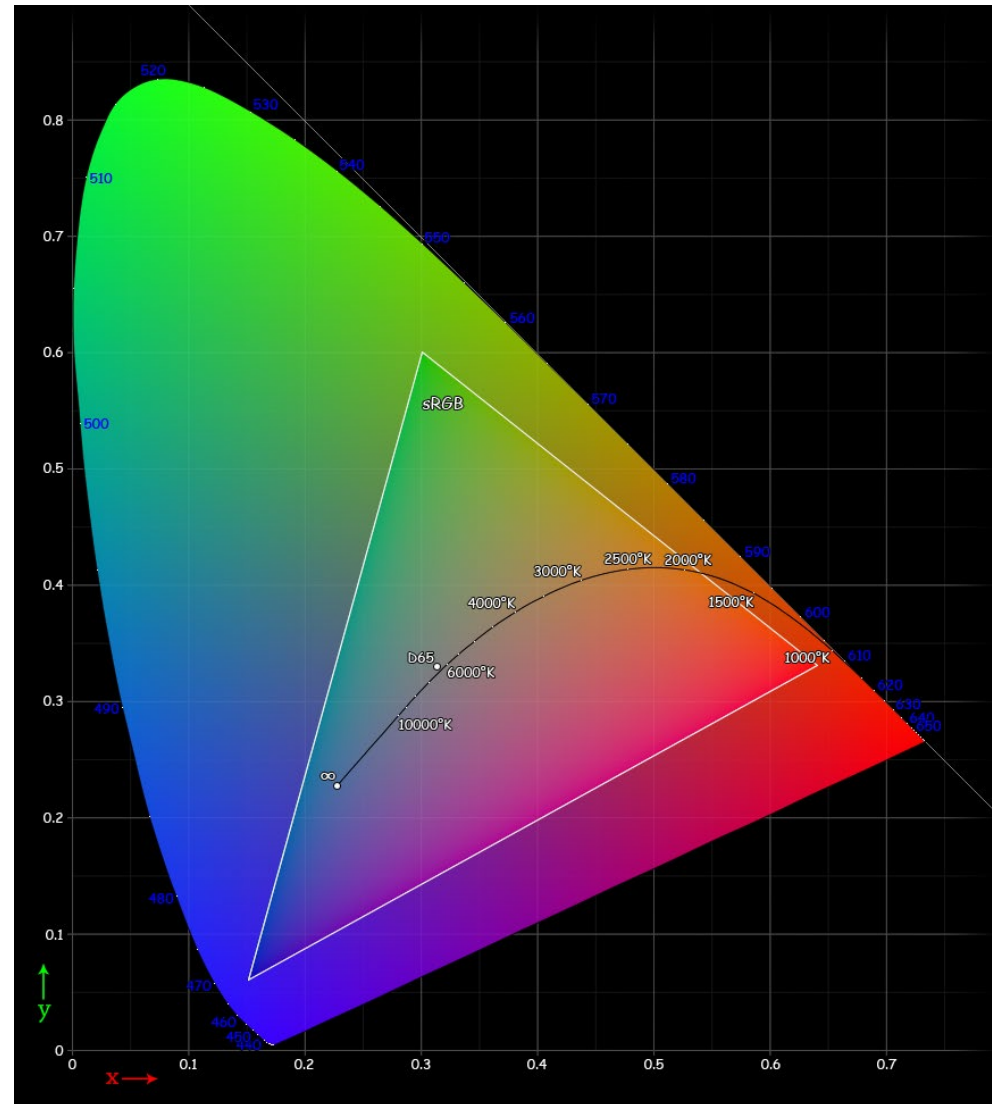


Why Gamut and not Pareto Front?



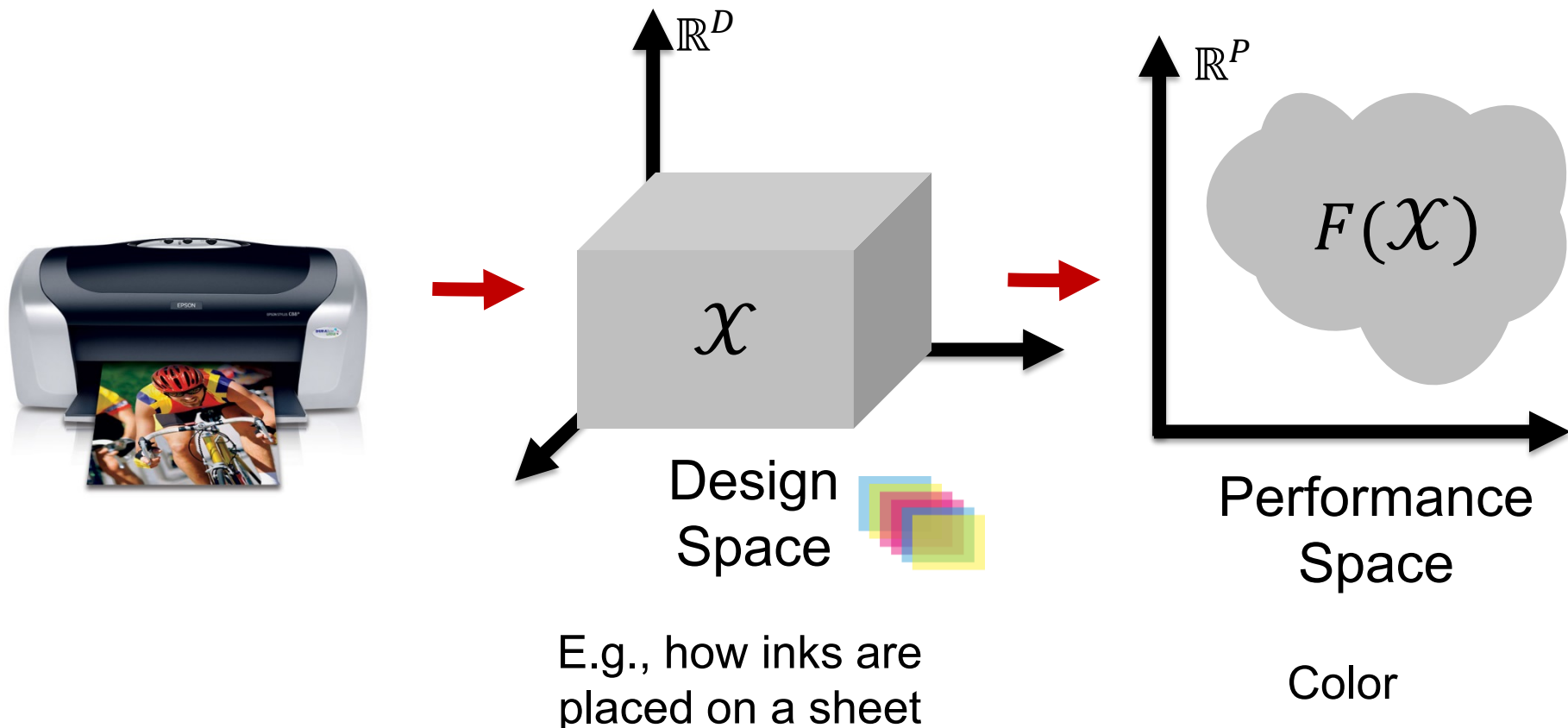
Example II: Color Gamut

- The subset of colors which can be accurately represented within a given color space or by a certain output device.



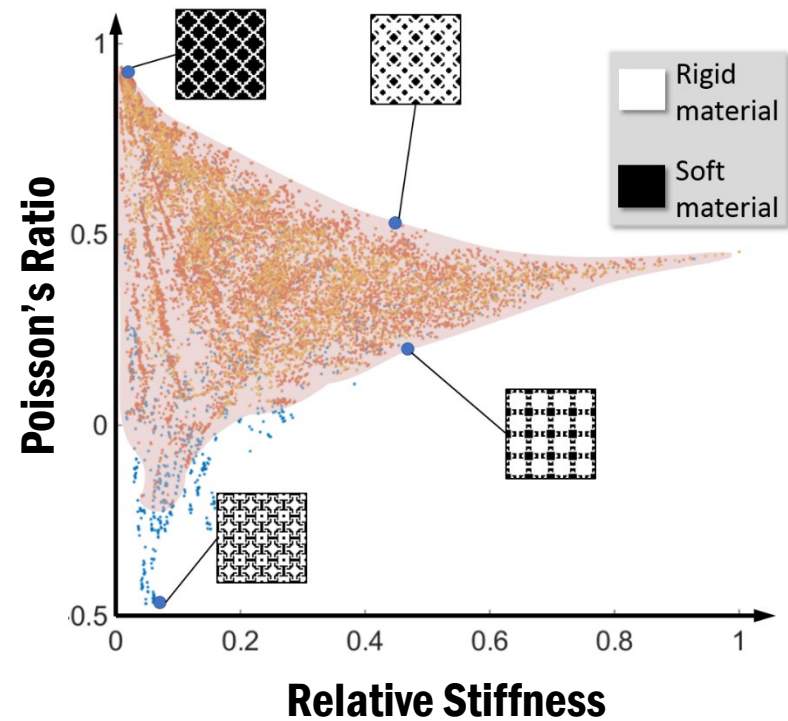
Hardware Capabilities Limit Gamut

- Gamut can be directly tied to capabilities of a given hardware

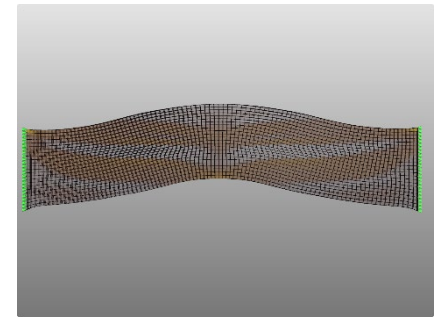
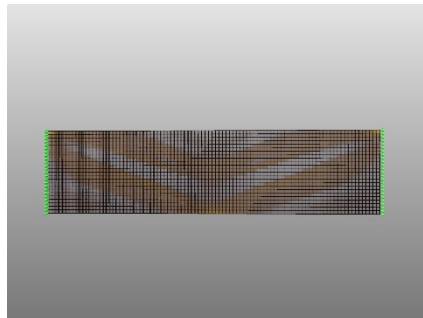
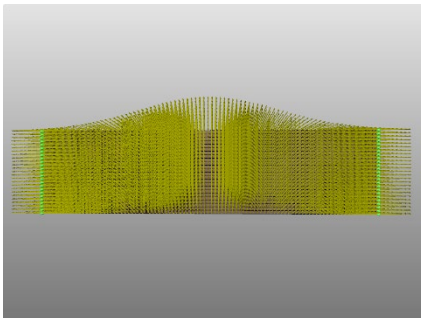
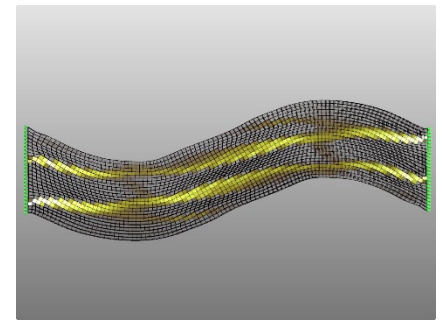
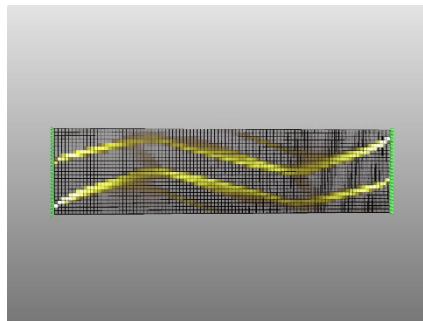
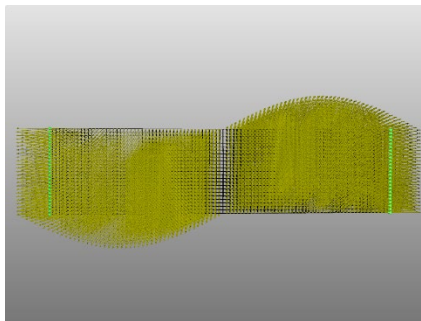
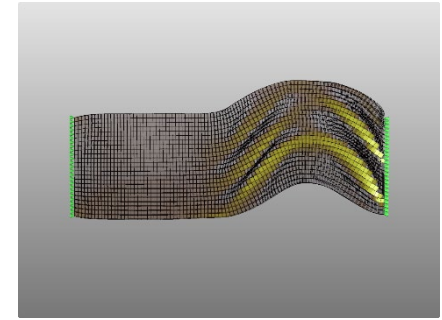
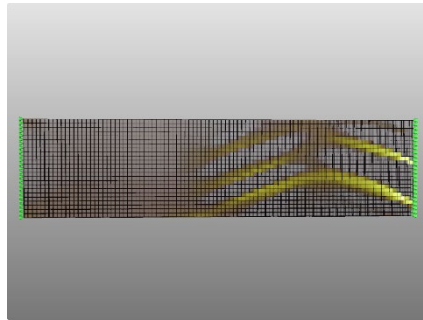
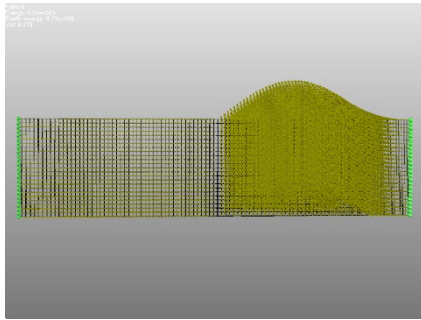


Example I: Mechanical Properties Gamut

- Space of bulk material properties that can be achieved with all material microstructures of a given size



Example I: Topology Optimization with Target Deformation

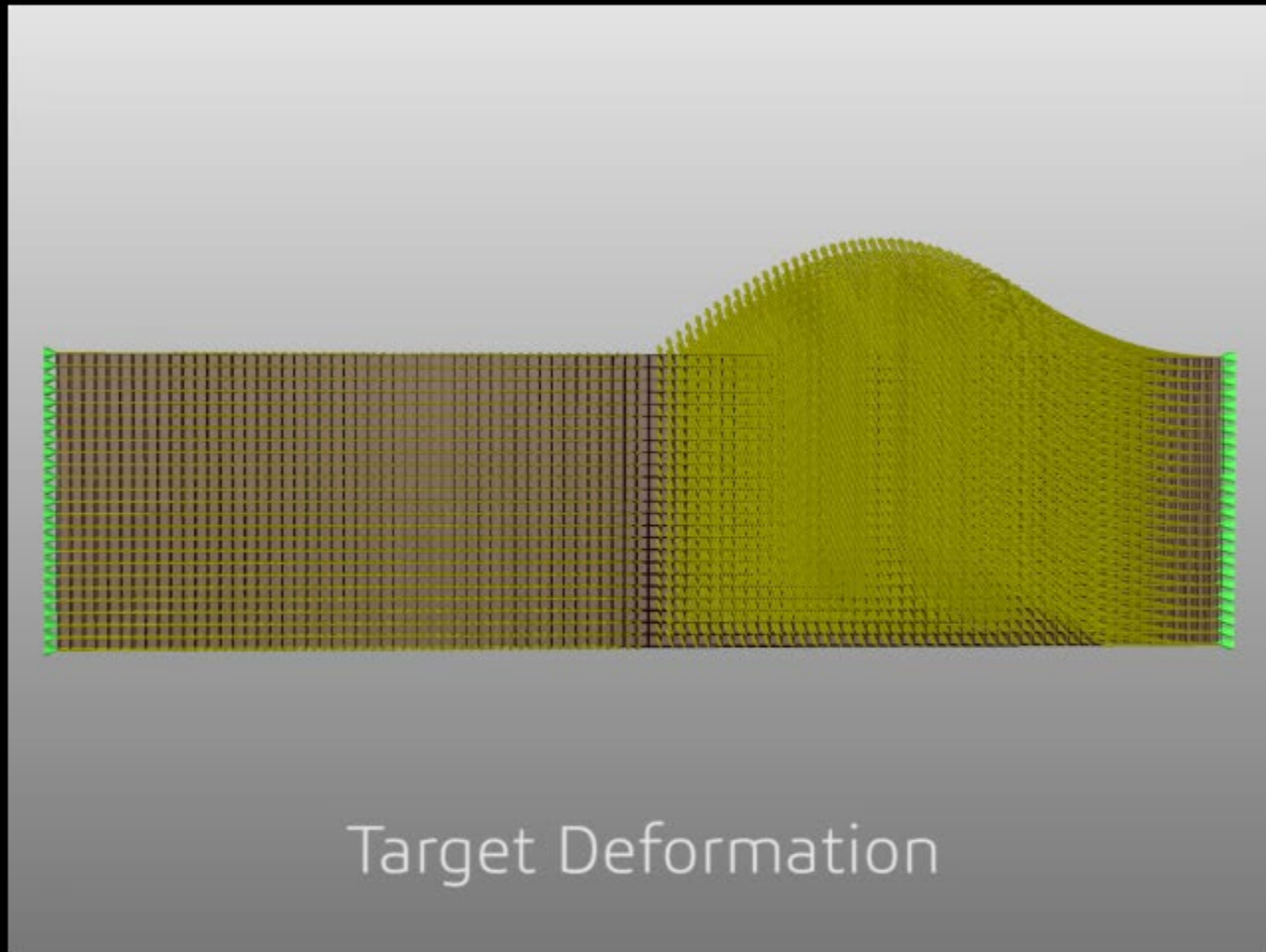


Target Deformation

Optimized Material Distribution

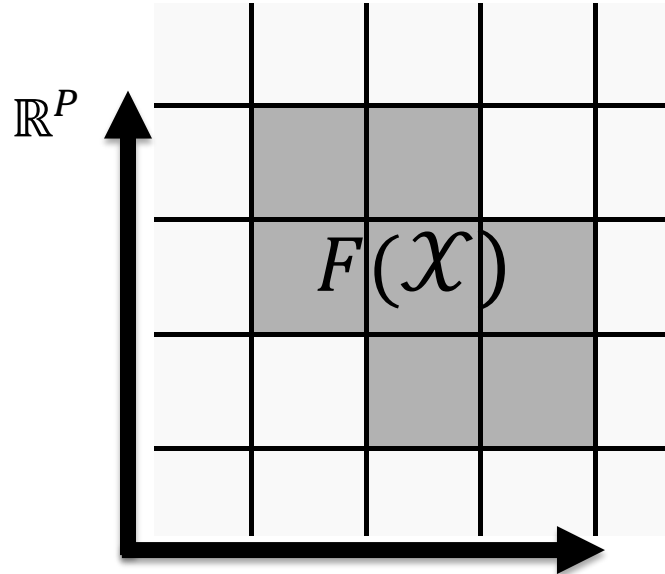
Simulation Results

Example I: Topology Optimization with Target Deformation

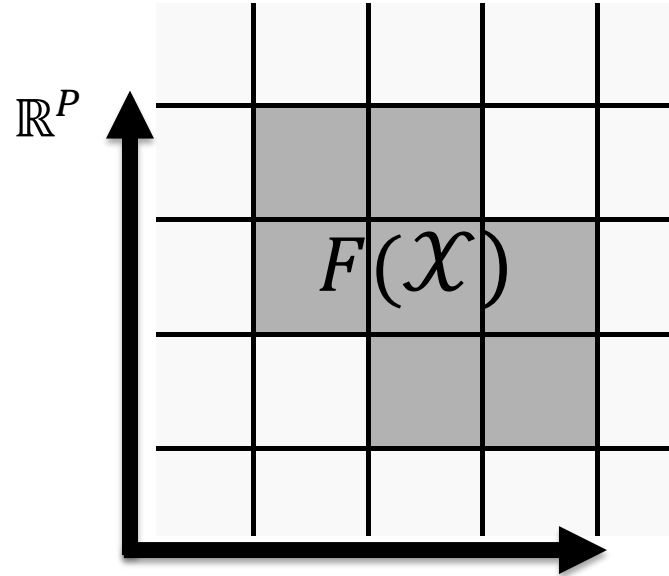


How to Represent Gamut?

- Boundary
 - Mesh/contour
- Volume
 - Grids (e.g., voxels), adaptive grids, points, distance fields

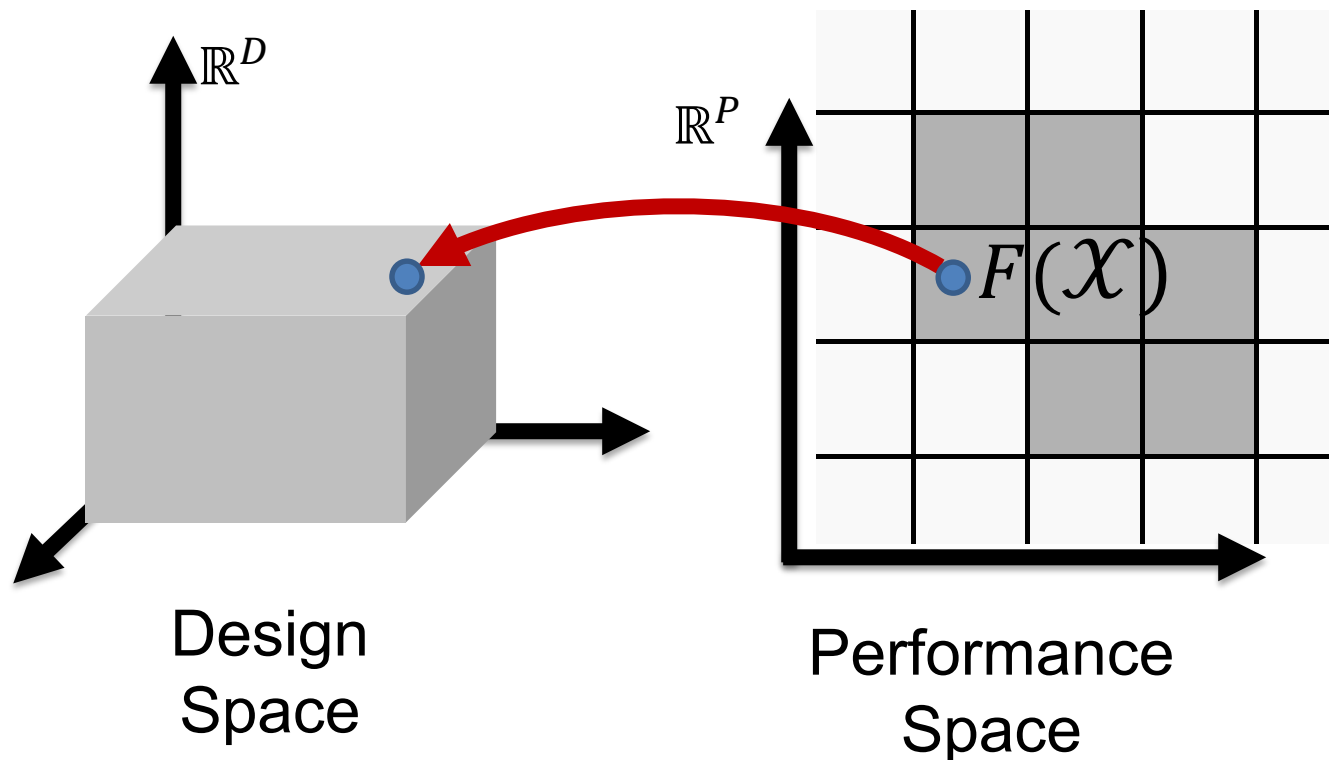


Why Volumetric Gamut Representations?

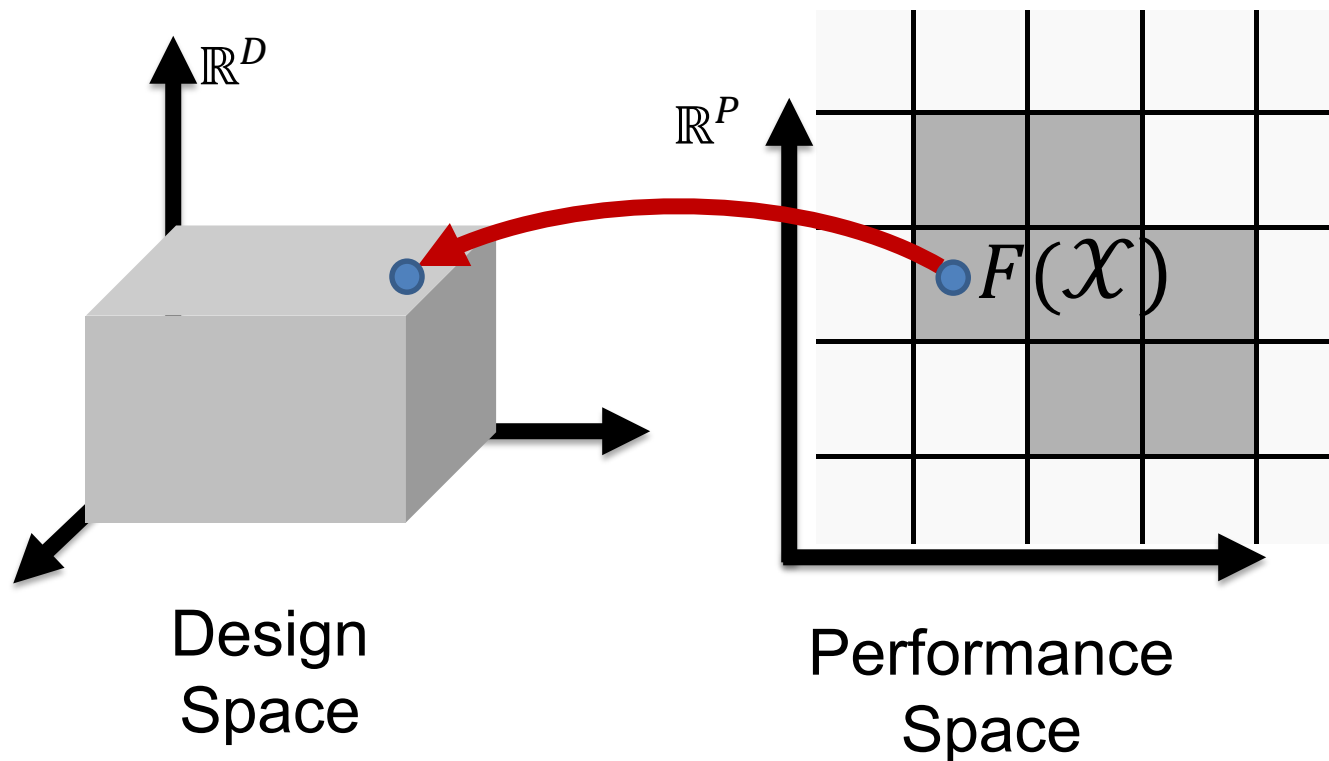


Why Volumetric Gamut Representations?

- Easy to check whether points are inside/outside
- Each cell can store points mapping back to the design space

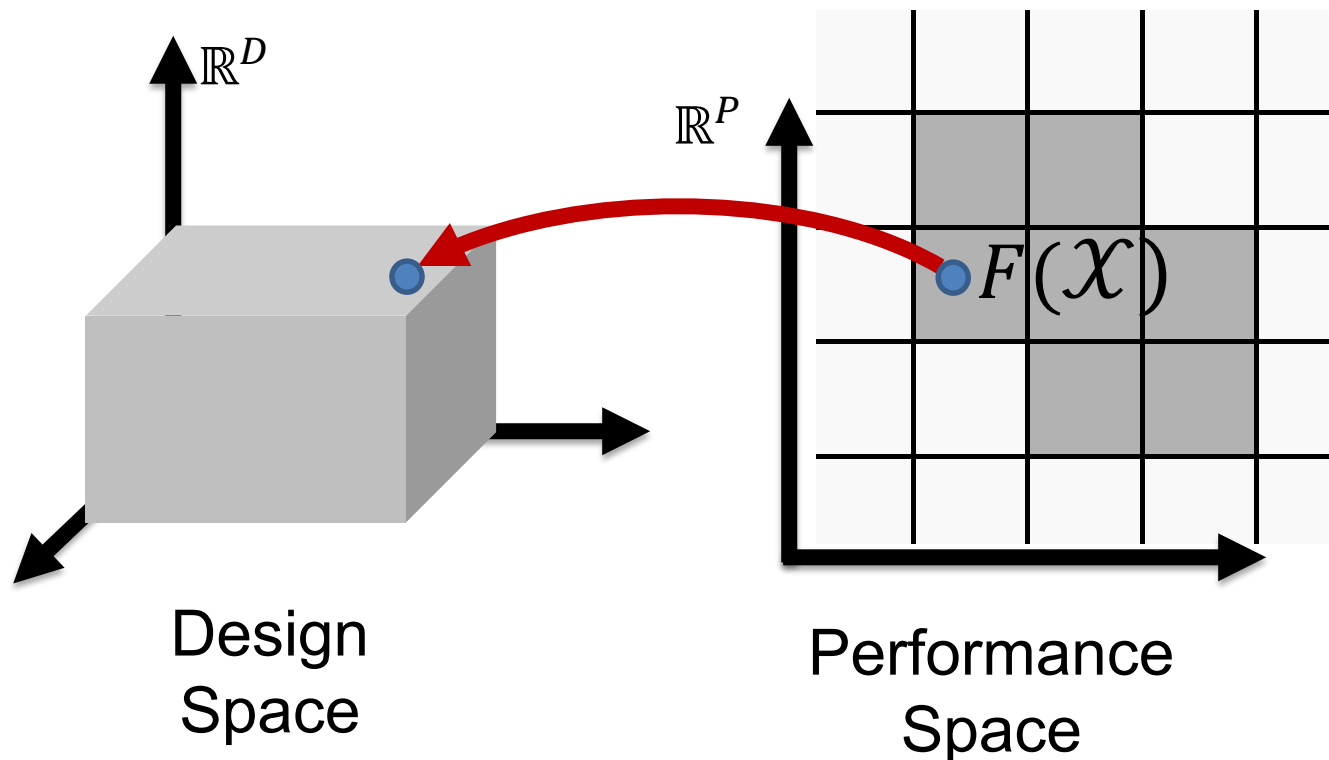


How to Represent Gamut in Higher Dimensions?



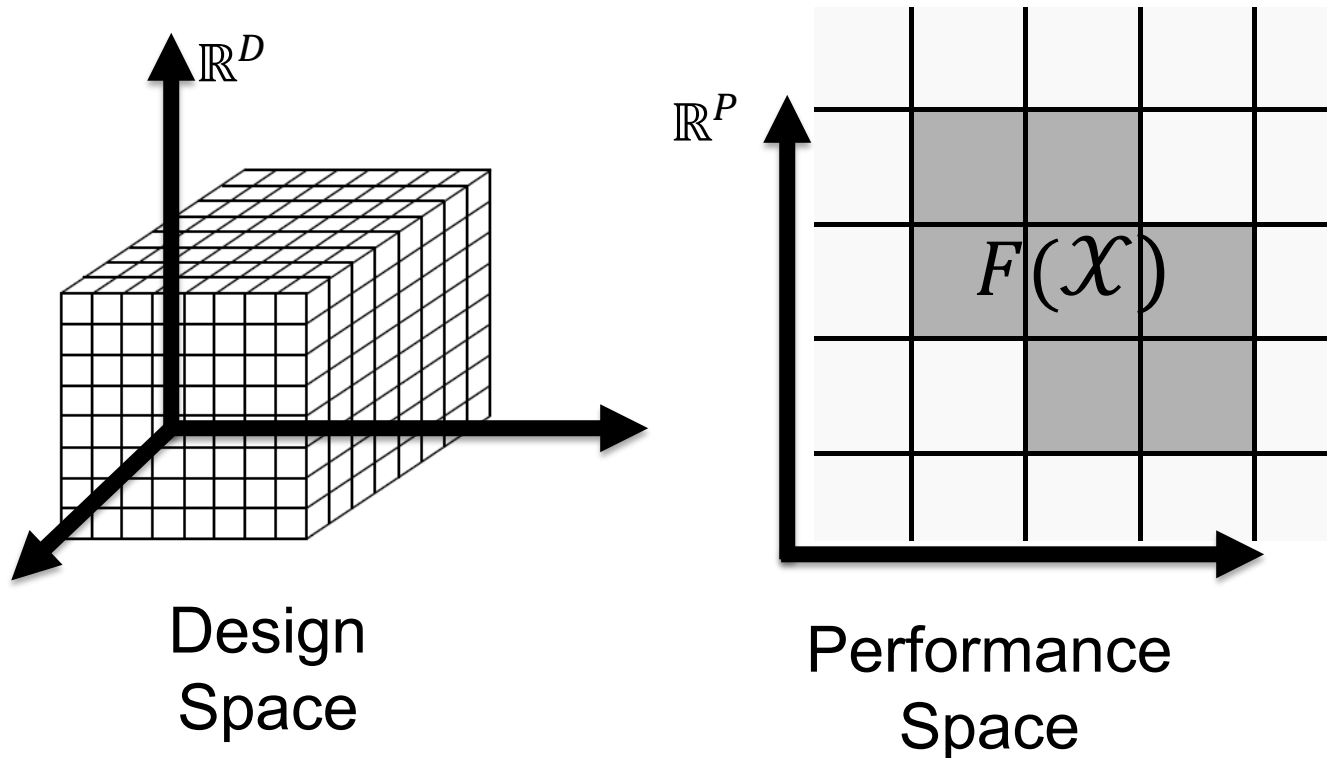
How to Represent Gamut in Higher Dimensions?

- These representations are useful but have not been explored much
- Possible representations: points, classifiers



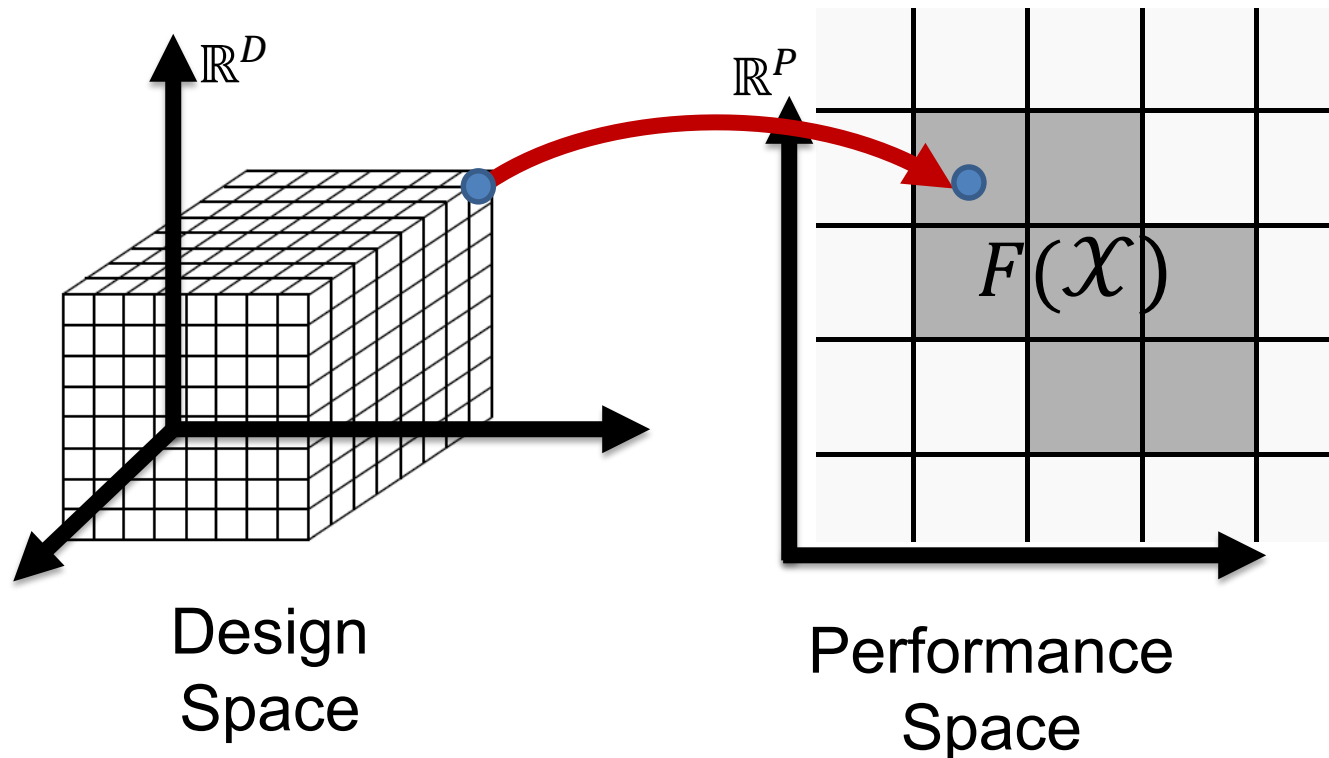
How to Compute Gamut?

- When design space is low-dimensional



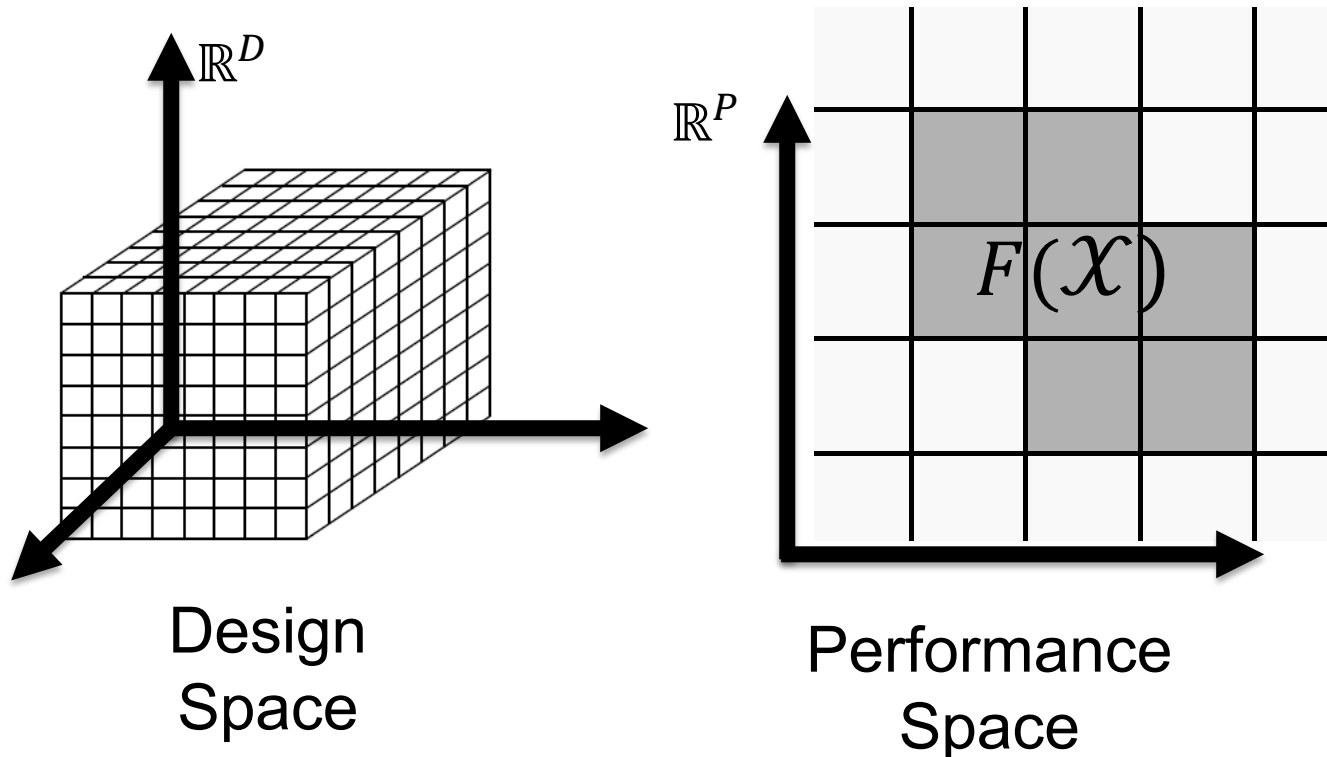
How to Compute Gamut?

- When design space is low-dimensional we can explicitly compute the mapping for all points in design space
- Example: 2D printers/color



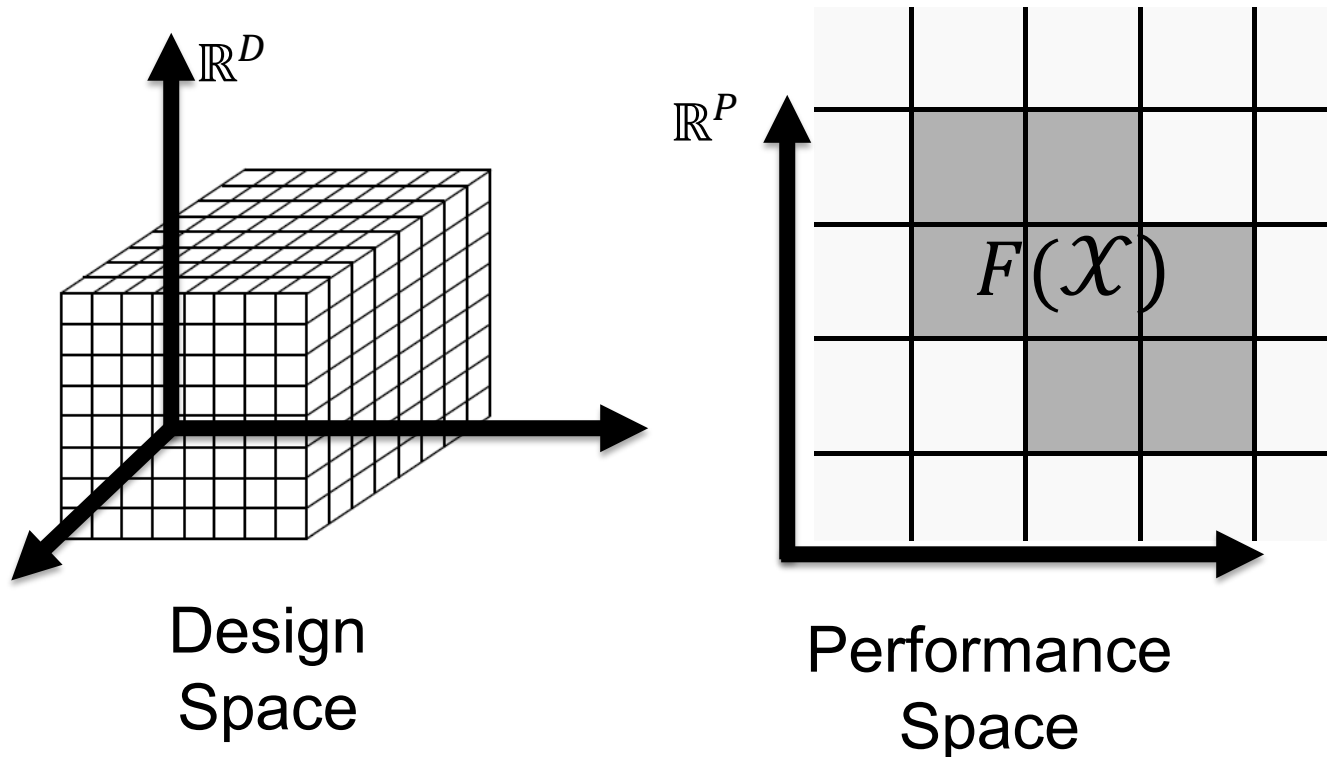
How to Compute Gamut?

- When design space is high dimensional



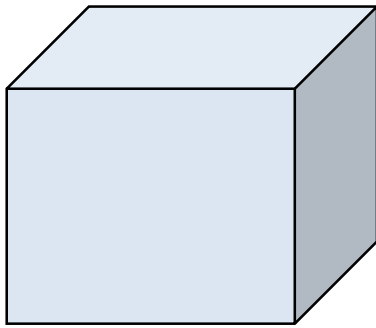
How to Compute Gamut?

- When design space is high dimensional we can use genetic algorithms to expand gamut in all directions

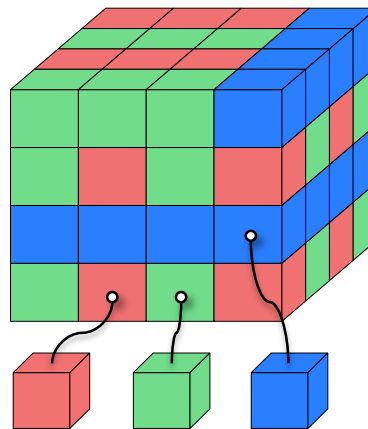


Example: Printing Microstructures

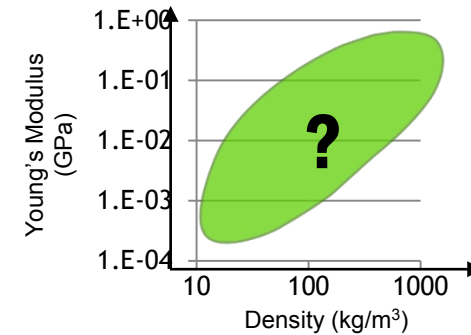
A unit microstructure cell



Heterogeneous material



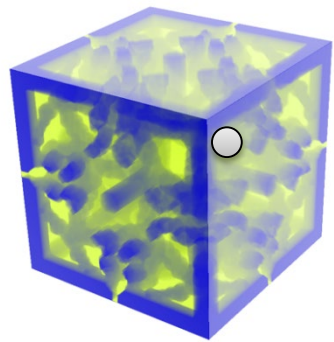
Base homogenous materials



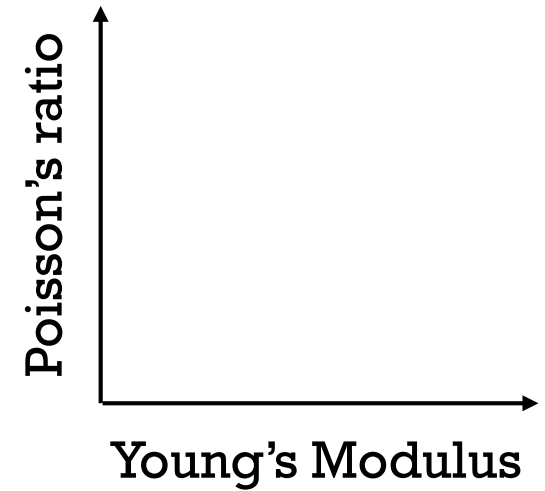
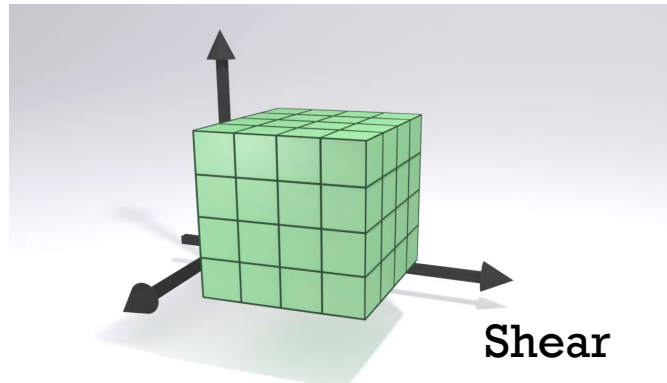
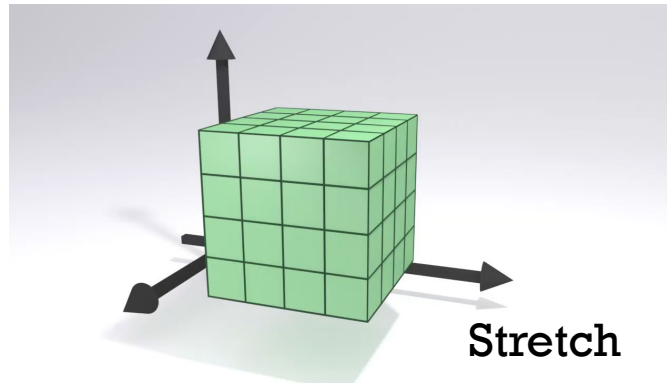
What physical properties can be achieved with microstructures?



Mapping Microstructures to Material Properties



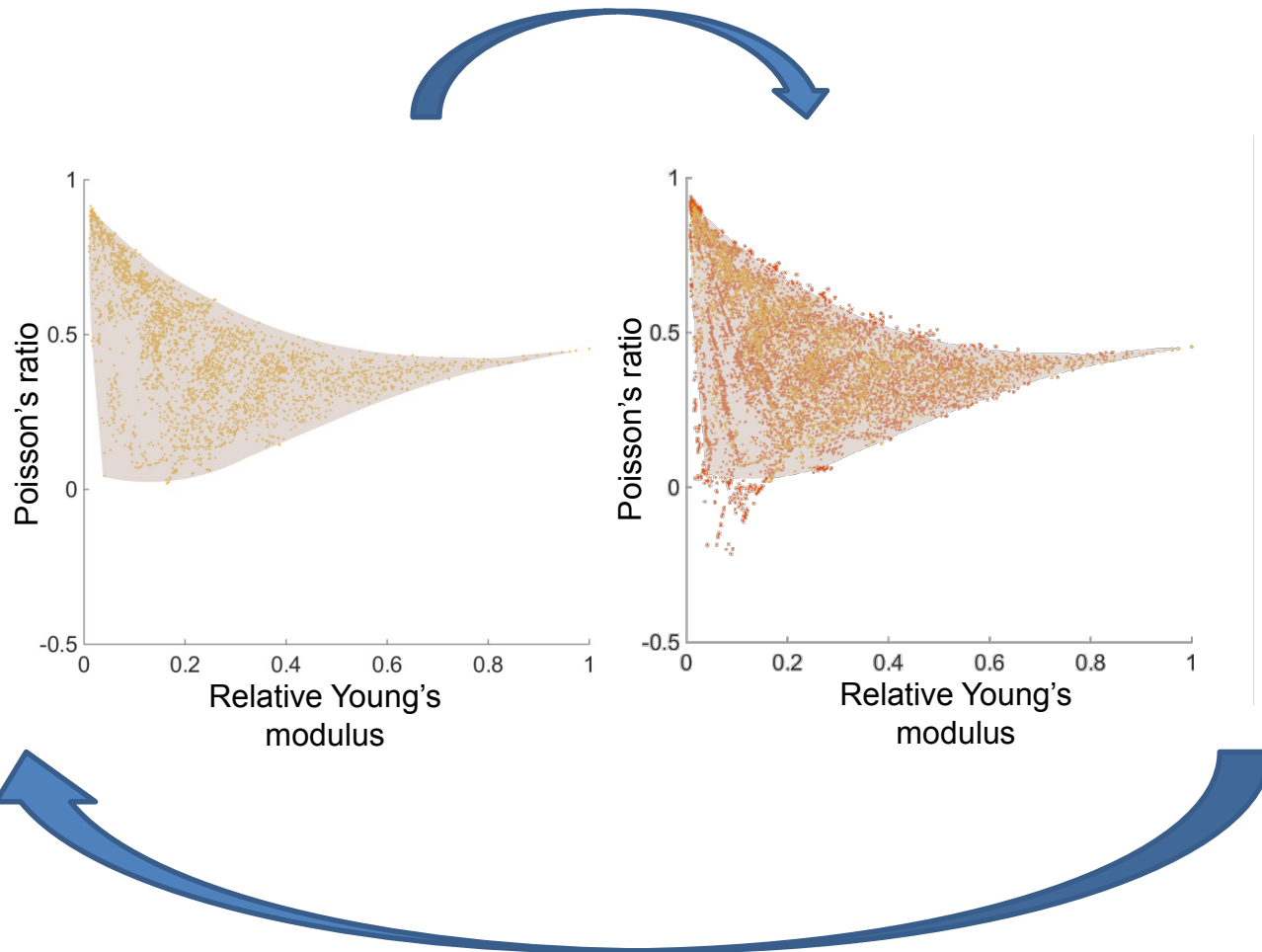
Microstructure





Discrete Sampling/Genetic Algorithm

Discrete Sampling



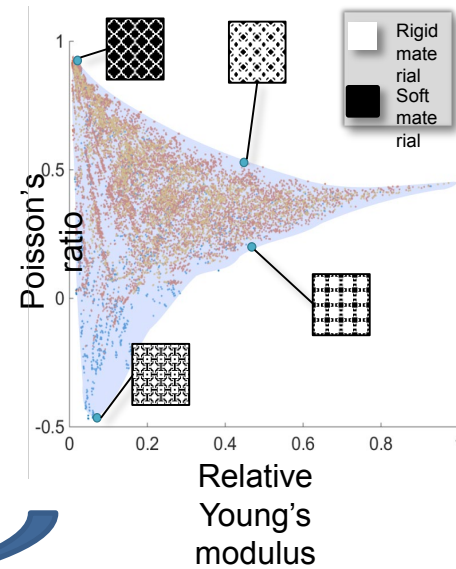
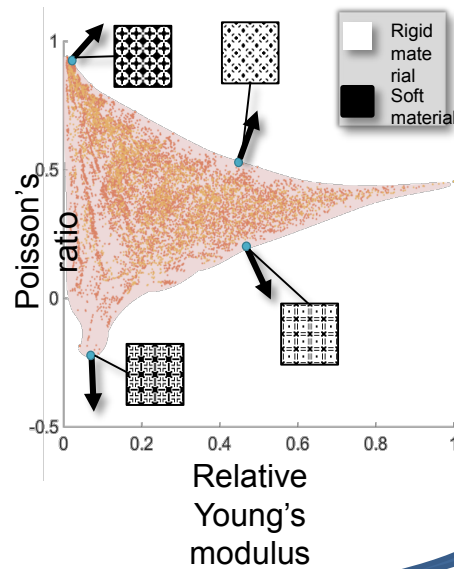
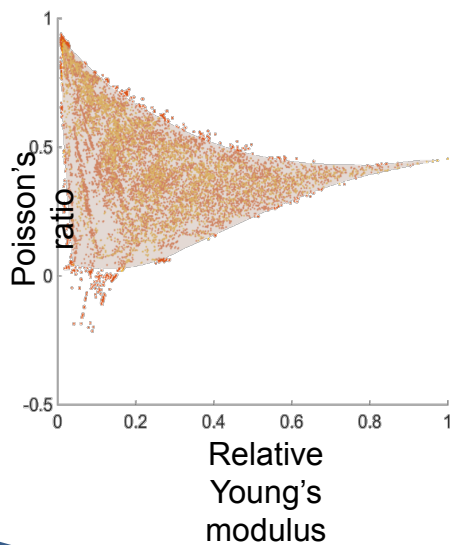
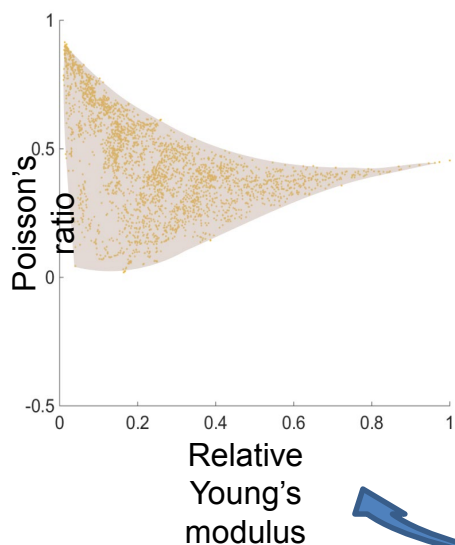


Discrete Sampling and Continuous Optimization

Discrete Sampling

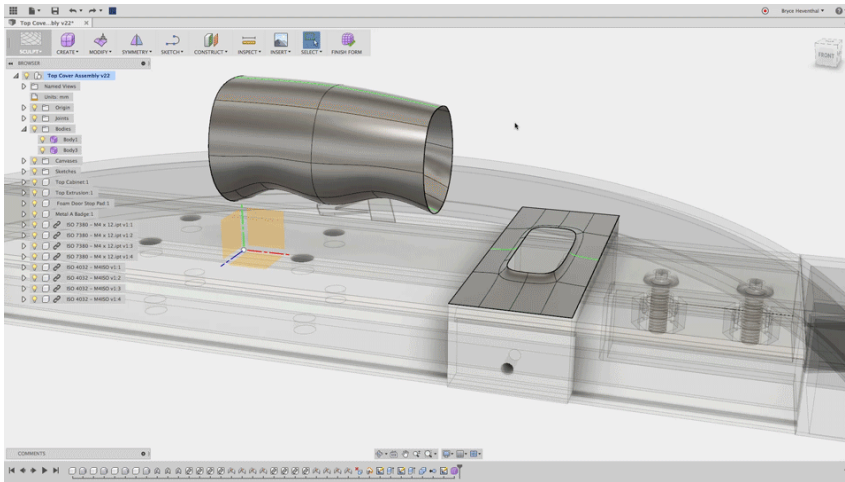


Continuous
Optimization

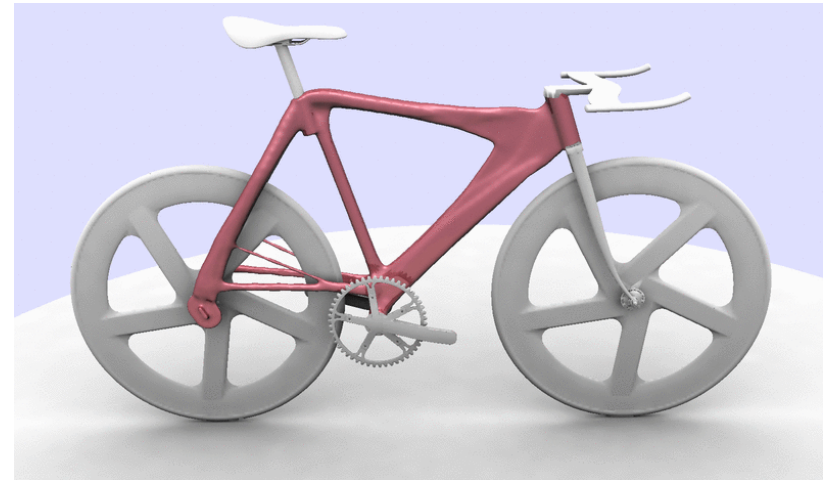


Inverse Methods

Direct Design vs. Generative Design



Direct Design

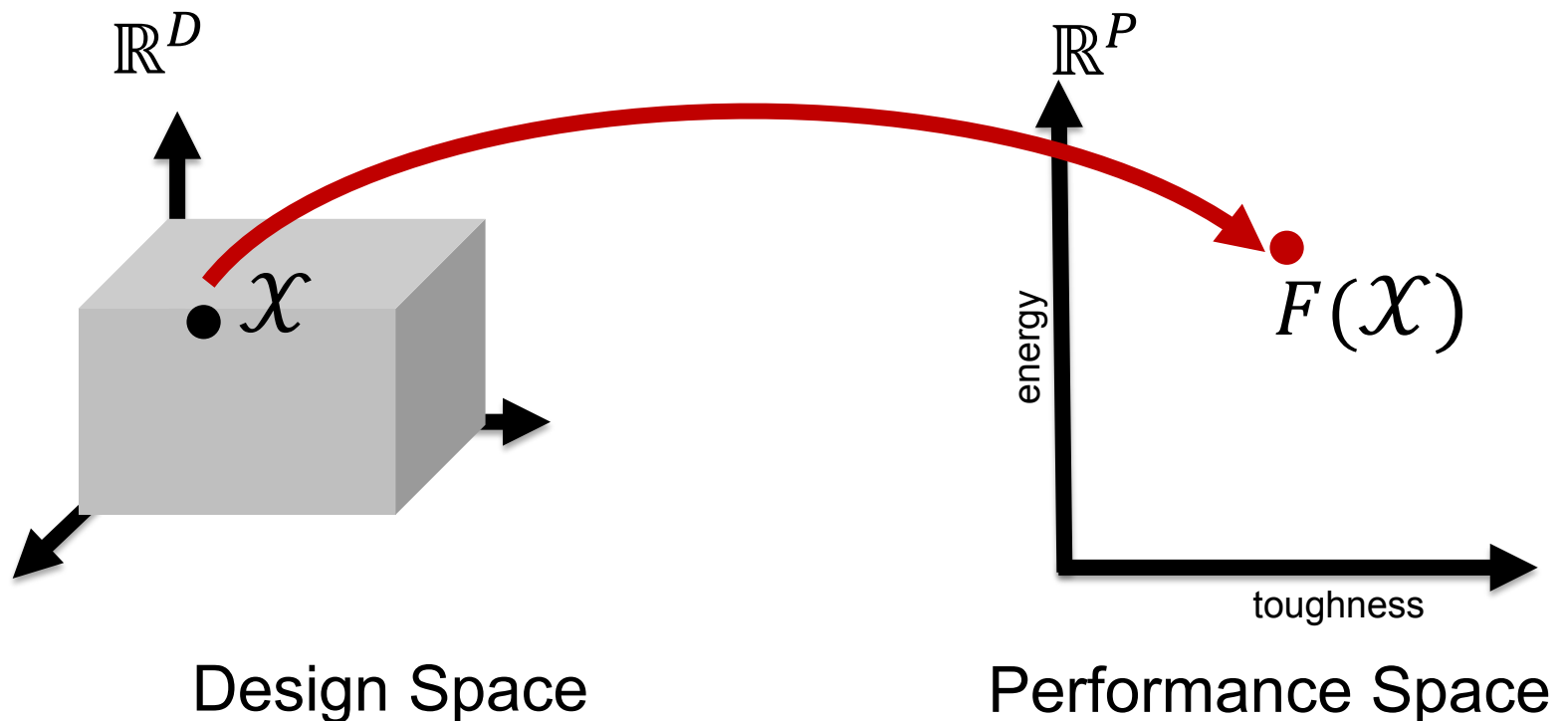


Generative Design



From Design Space to Performance Space

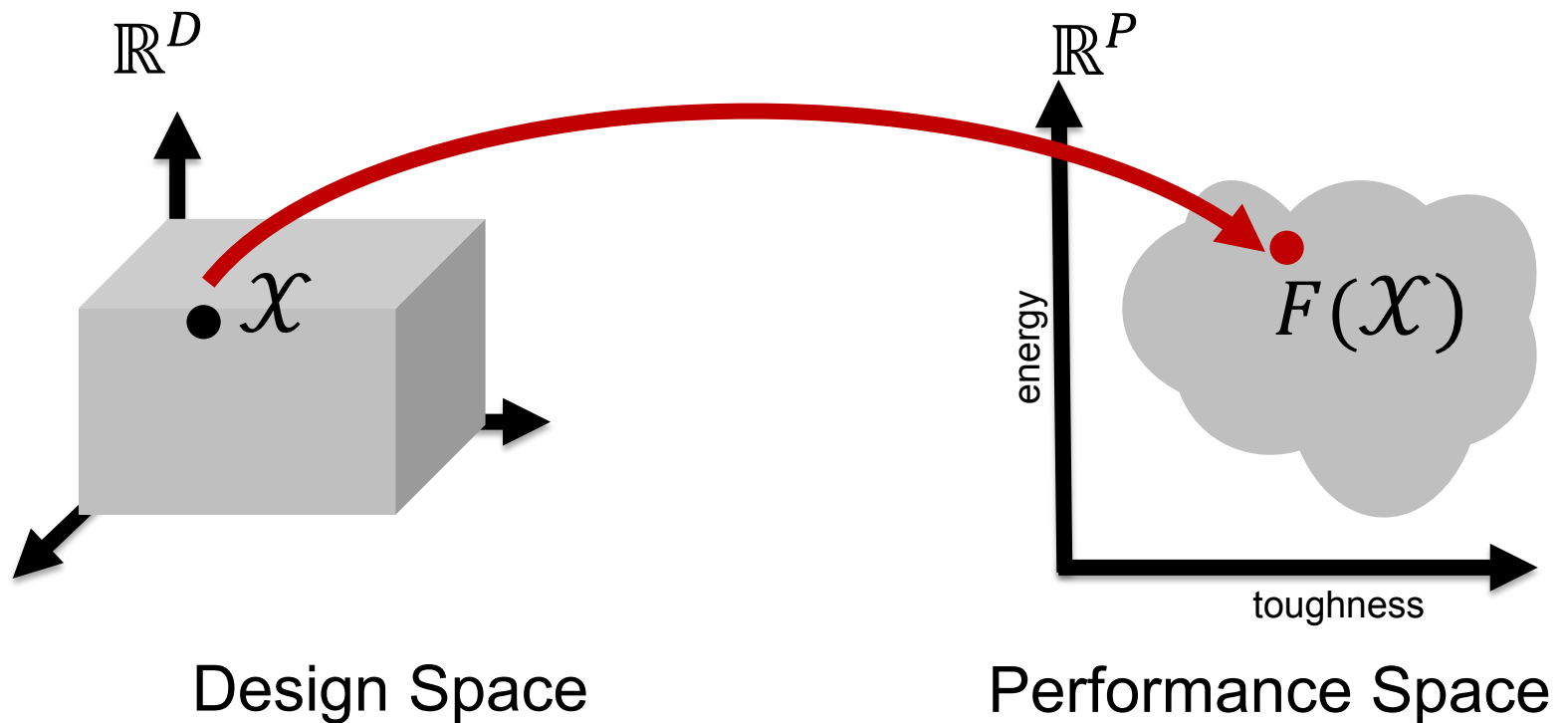
- Numerical simulation maps points from design space to performance space





Mapping the Whole Performance Space

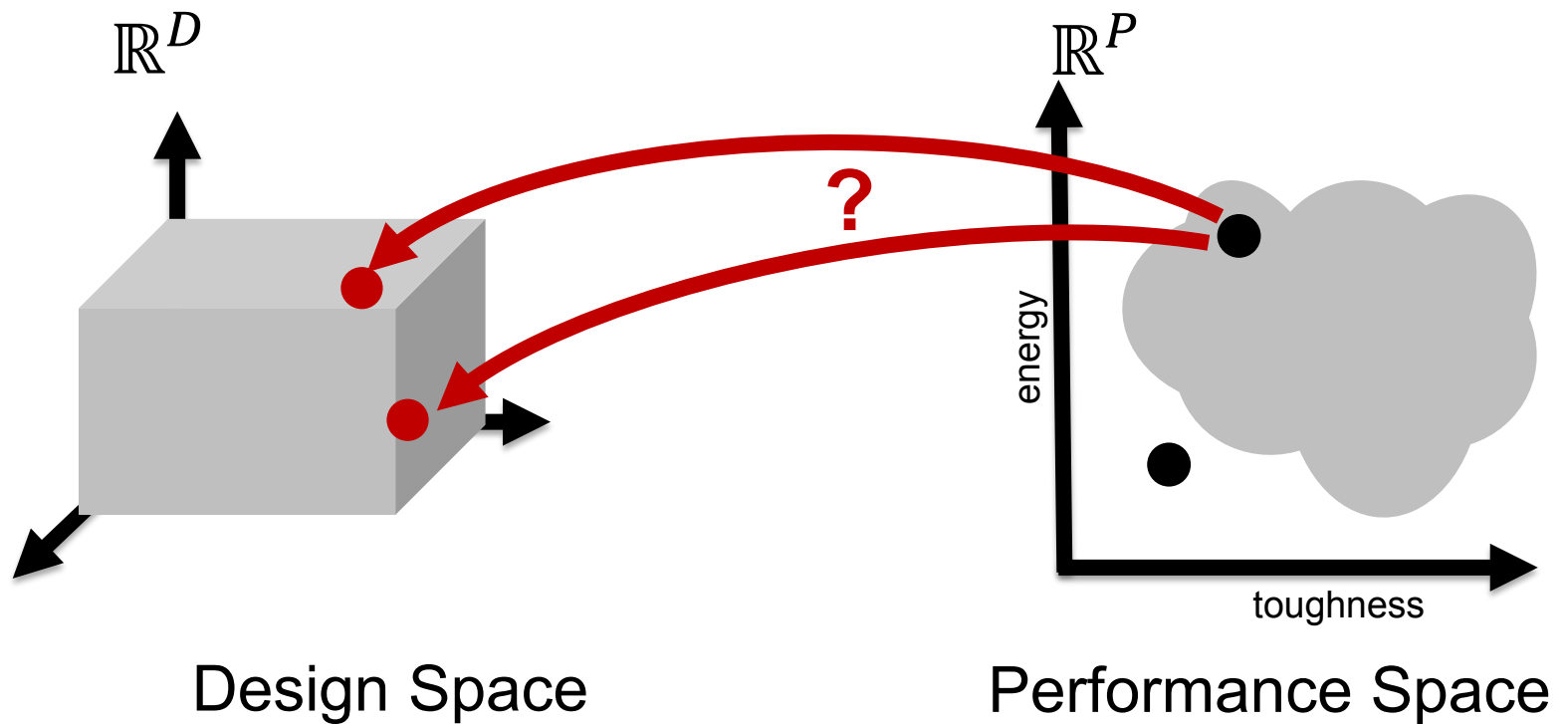
- Numerical simulation maps points from design space to performance space





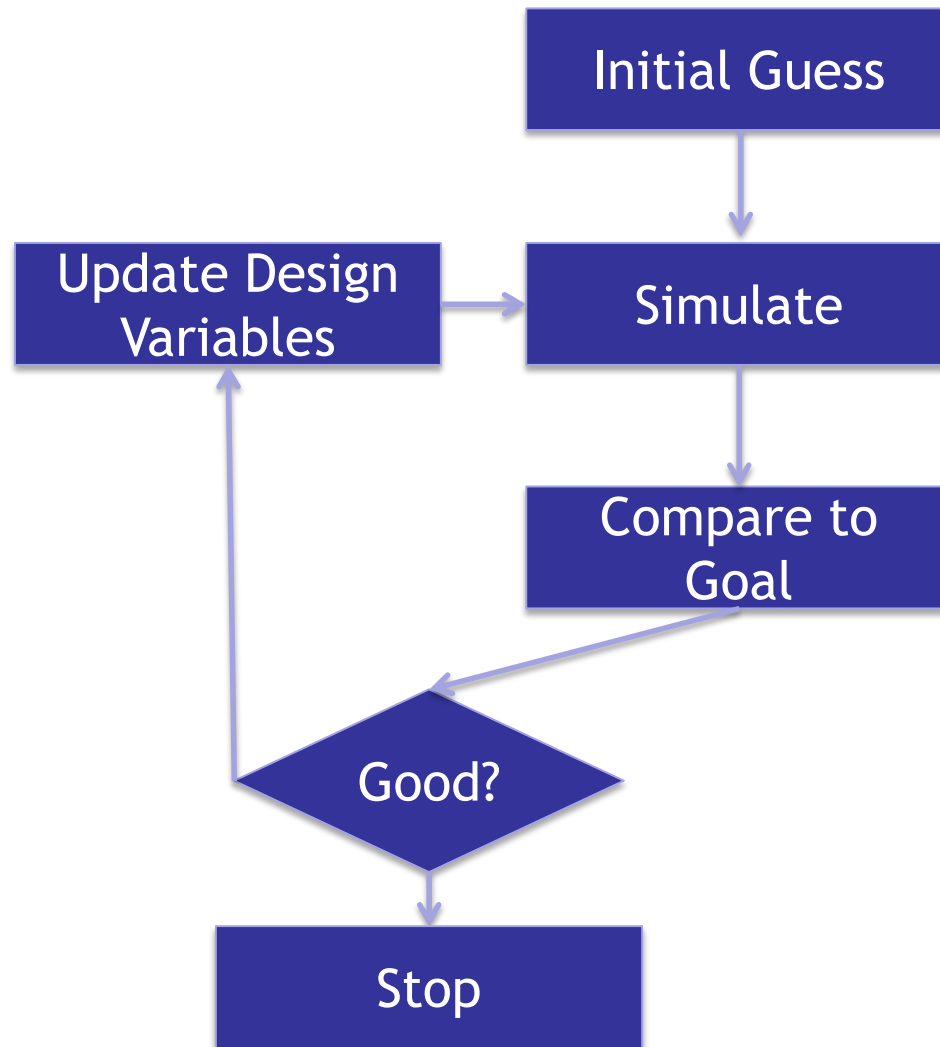
Inverse: From Performance Space to Design Space

- Inverse problem is much more difficult

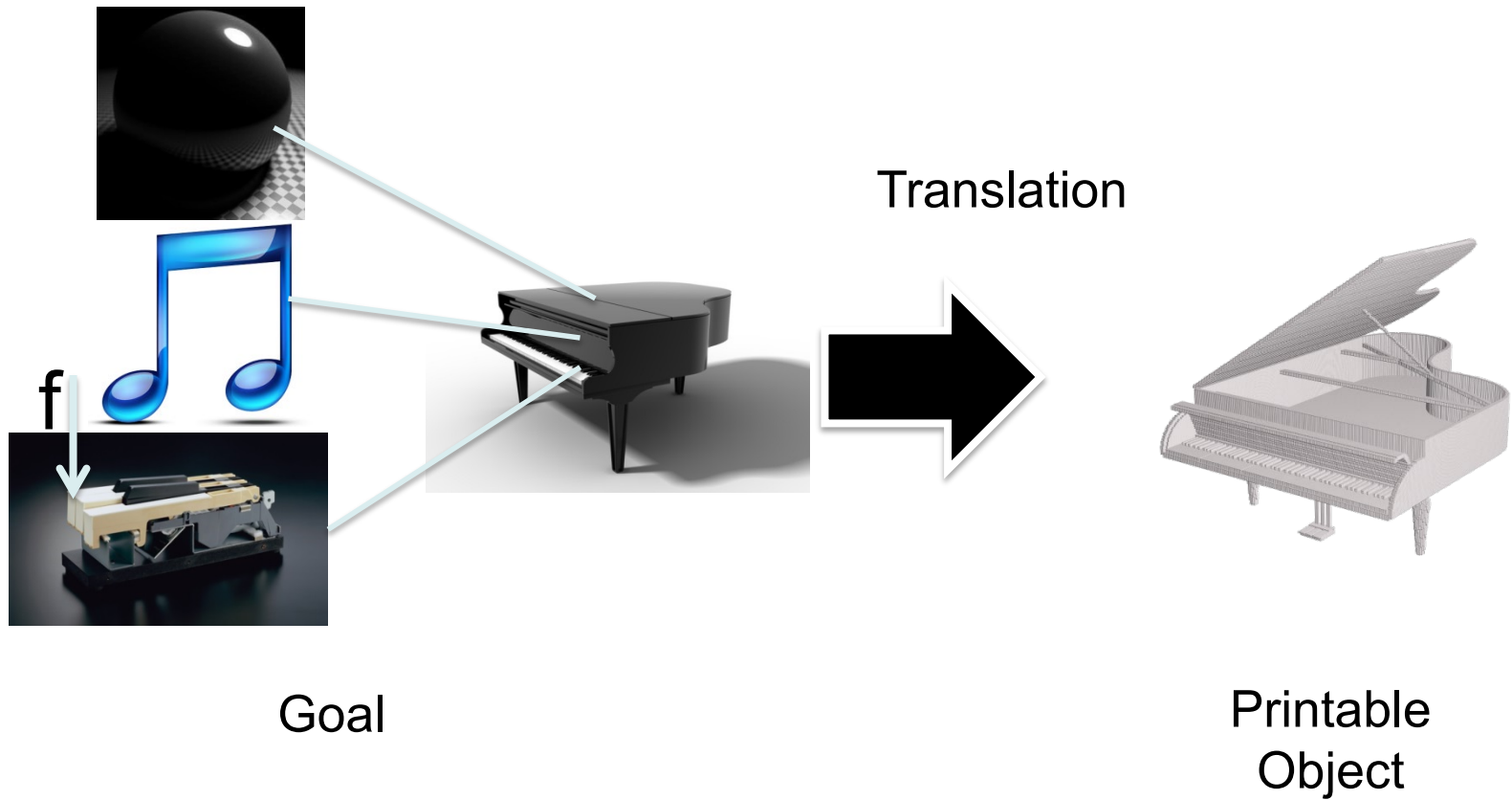




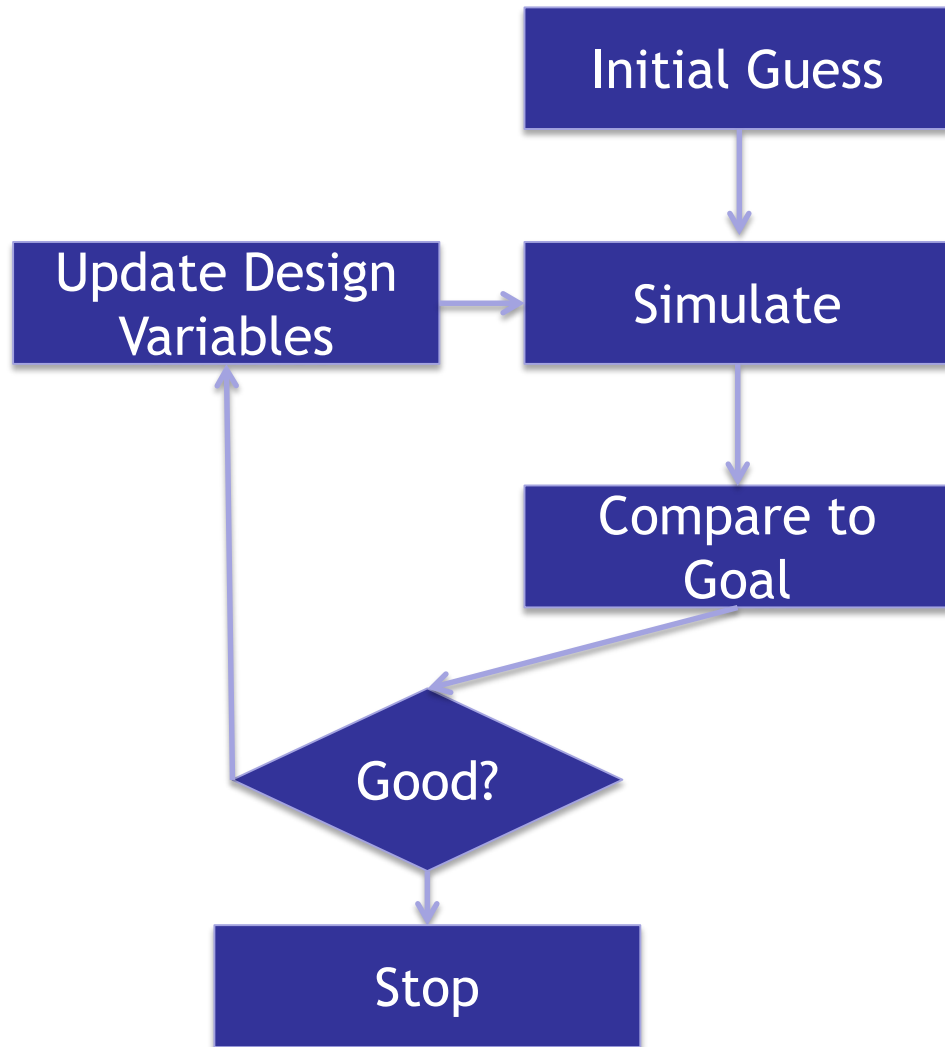
Inverse: From Performance Space to Design Space



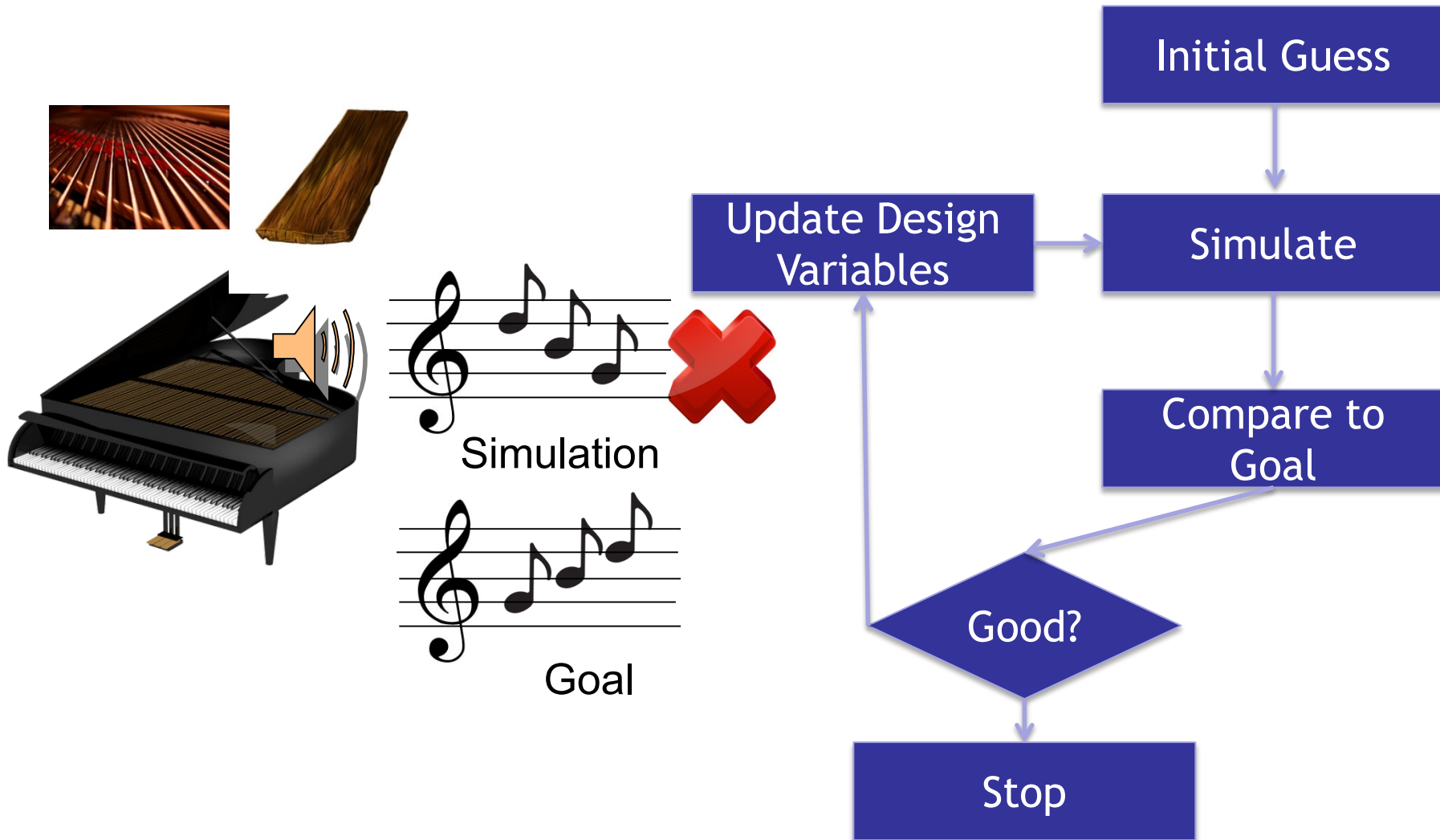
Inverse: From Performance Space to Design Space



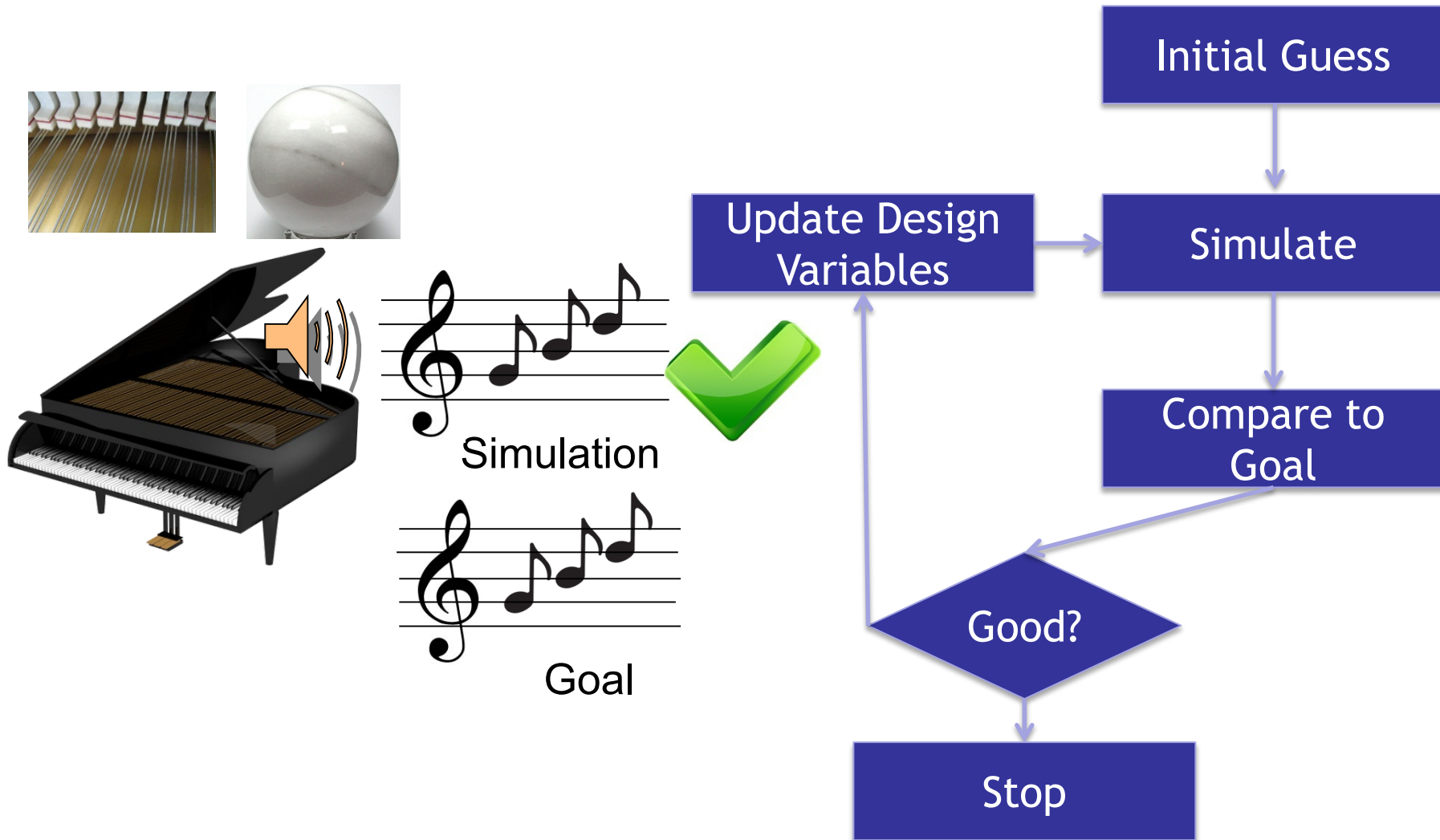
Inverse: From Performance Space to Design Space



Inverse: From Performance Space to Design Space



Inverse: From Performance Space to Design Space



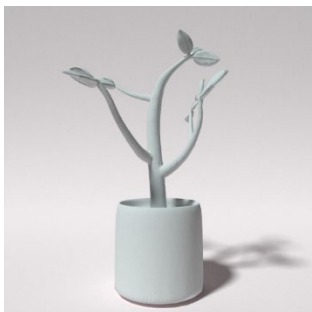


Example

Design target



fabrication



Rest shape

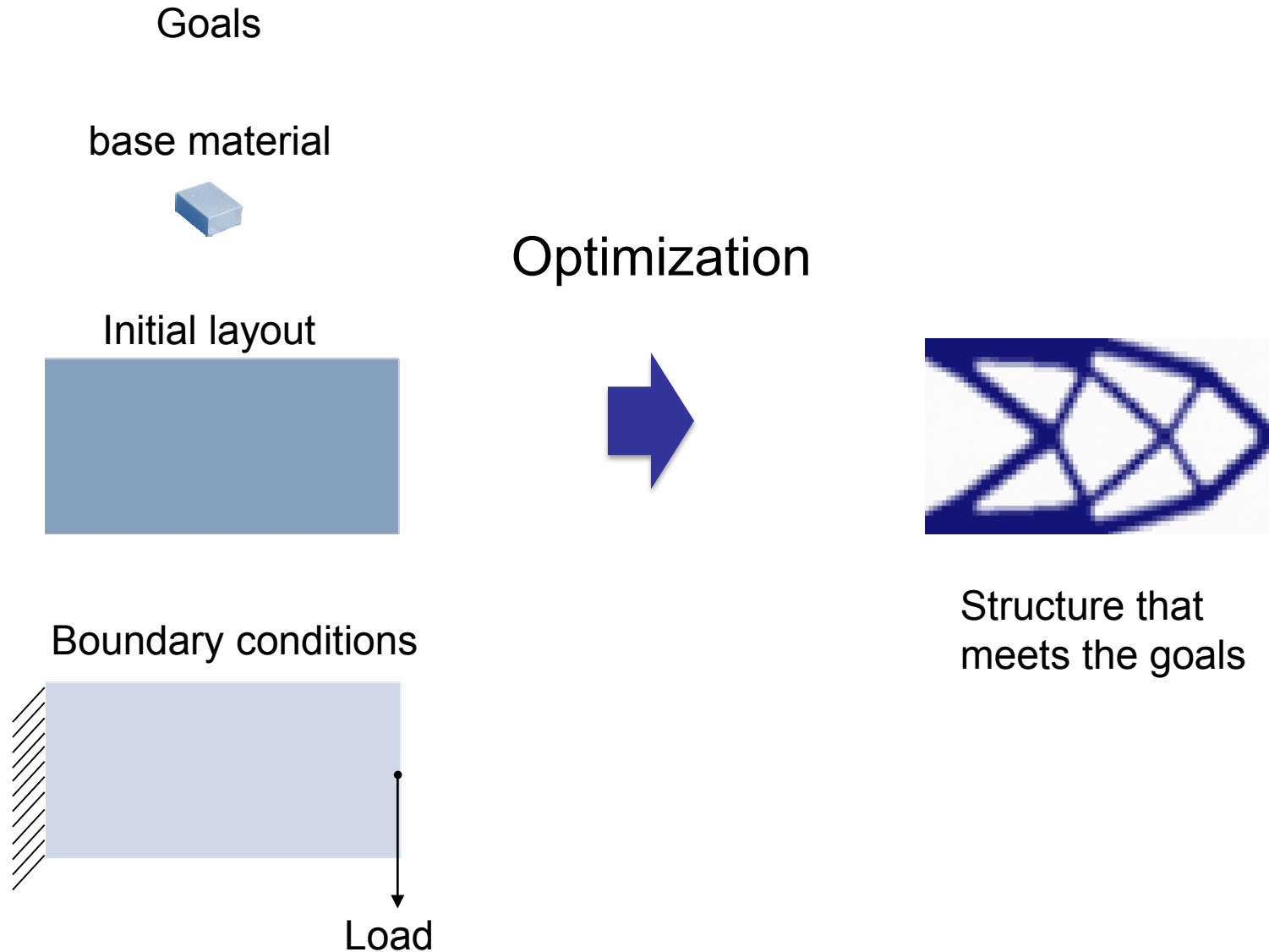
fabrication



Inverse Methods: Topology Optimization

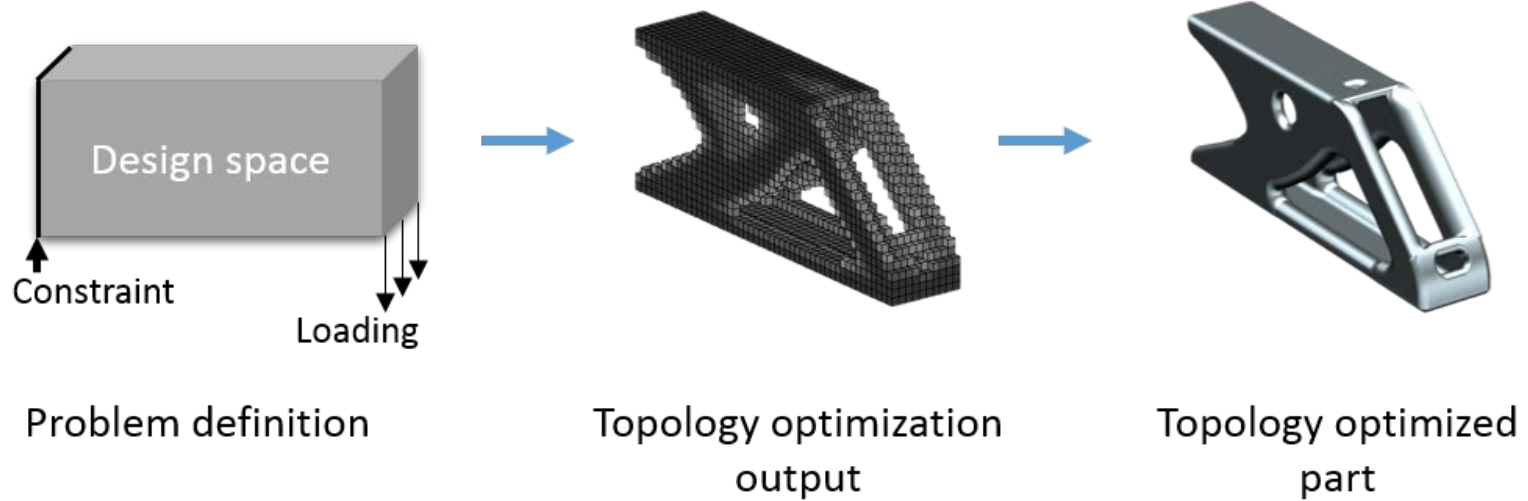


The Inverse Problem





Topology Optimization





Topology Optimization



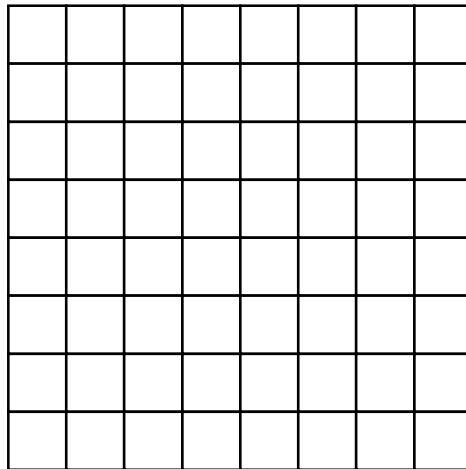


Topology Optimization

- We will look at the simplest algorithm
- Input:



1 Base Material with Elasticity tensor $\mathbf{C}_0$

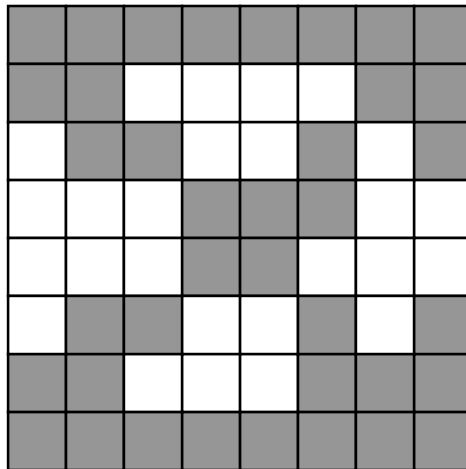


Initial layout



Topology Optimization

- Objectives:
 - Structure should be as stiff as possible (i.e. the compliance should be minimal) when a load is applied
 - The total amount of material should be equal to V_{max}
- Output:
 - Voxels with material assignment (no material, full)

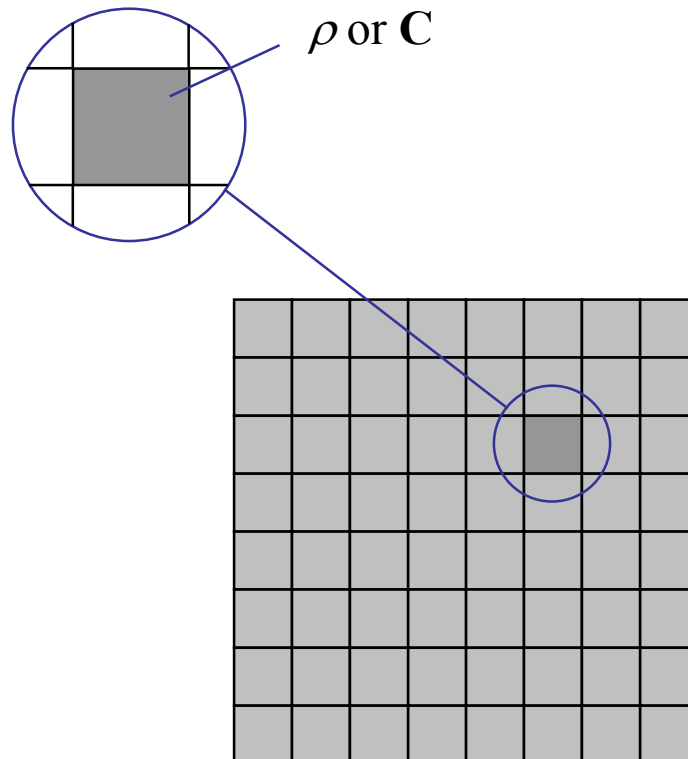




Topology Optimization

- Design variables

Material property $\mathbf{C} = \rho^p \mathbf{C}_0$

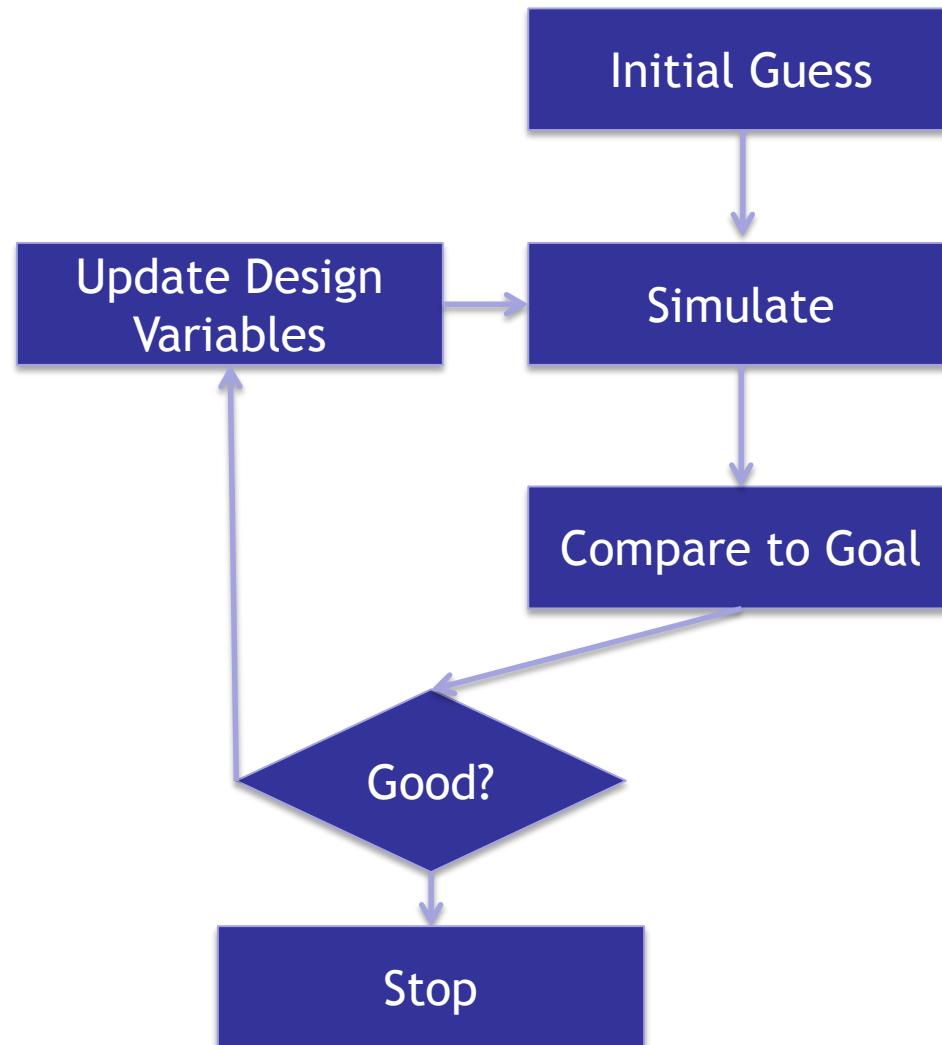


the method is called SIMP,
power-law or density
approach.

[Bendsøe, 1989]



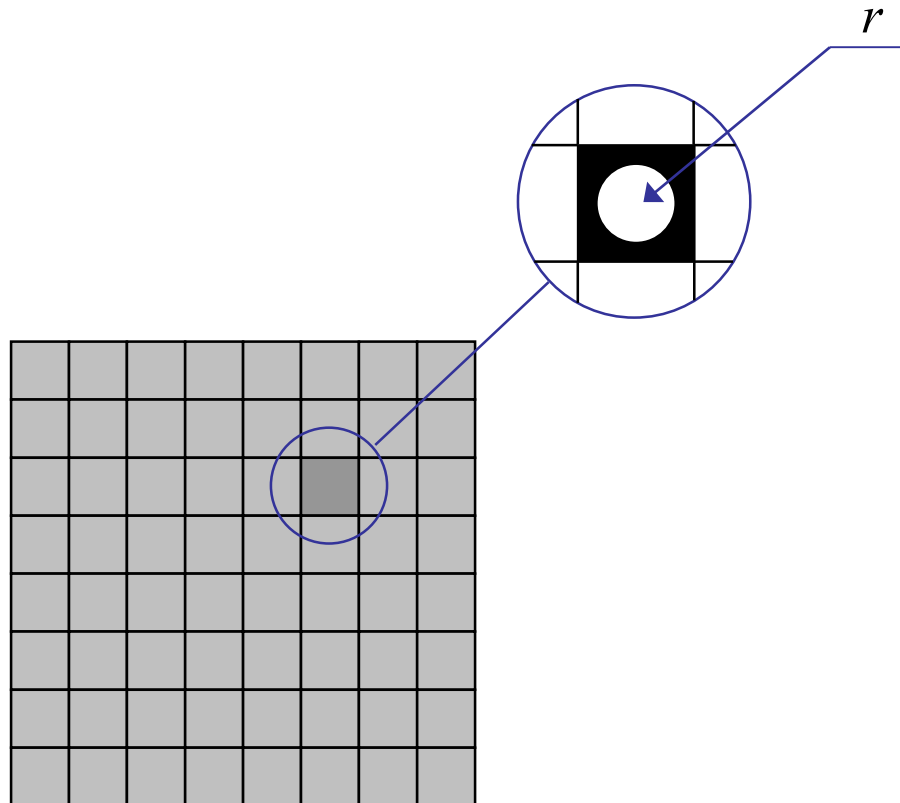
Topology Optimization Workflow





Other Topology Optimization Approaches

- Structural parameters as design variables





Implementation

- Compliance minimization of statically-loaded structures
- Based on the paper:
 - “A 99 line topology optimization code in Matlab” by Ole Sigmund, Structural and Multidisciplinary Optimization 21(2), 2001, pp. 120-127
- Code can be find here:
 - <http://www.topopt.mek.dtu.dk/apps-and-software>



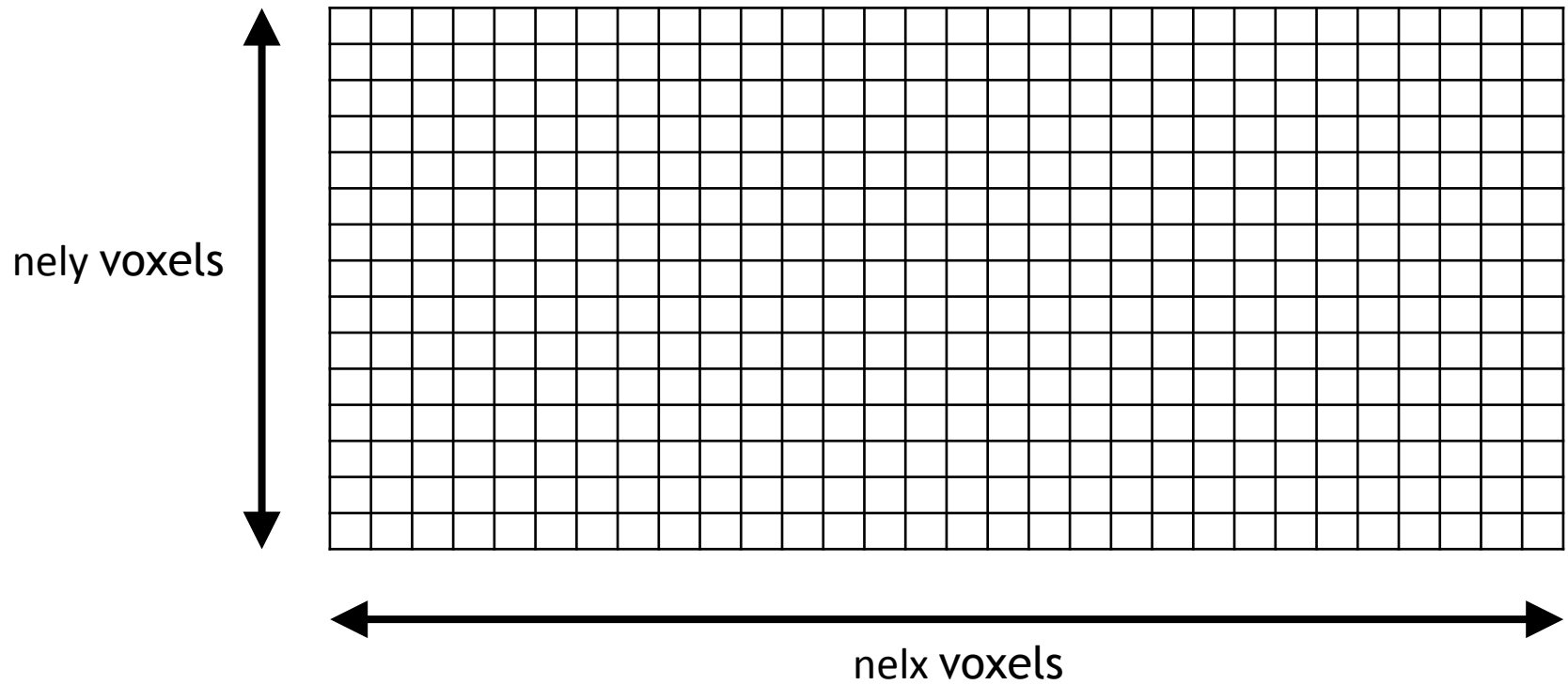
A 99 Line Topology Optimization Code

```
1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2  %%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
3  function top(nelx,nely,volfrac,penal,rmin)
4  % INITIALIZE
5  x(1:nely,1:nelx) = volfrac;
6  loop = 0;
7  change = 1.;
8  % START ITERATION
9  while change > 0.01
10     loop = loop + 1;
11     xold = x;
12     % FE-ANALYSIS
13     [U]=FE(nelx,nely,x,penal);
14     % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
15     [KE] = lk;
16     c = 0.;
17     for ely = 1:nely
18         for elx = 1:nelx
19             n1 = (nely+1)*(elx-1)+ely;
20             n2 = (nely+1)* elx +ely;
21             Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
22             c = c + x(ely,elx)^penal*Ue'*KE*Ue;
23             dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
24         end
25     end
26     % FILTERING OF SENSITIVITIES
27     [dc] = check(nelx,nely,rmin,x,dc);
28     % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
29     [x] = OC(nelx,nely,x,volfrac,dc);
30     % PRINT RESULTS
31     change = max(max(abs(x-xold)));
32     disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
33           ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
34           ' ch.: ' sprintf('%6.3f',change) ])
35     % PLOT DENSITIES
36     colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
37 end
38 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39 function [xnew]=OC(nelx,nely,x,volfrac,dc)
40 l1 = 0; l2 = 100000; move = 0.2;
41 while (l2-l1 > 1e-4)
42     lmid = 0.5*(l2+l1);
43     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid))));
44     if sum(sum(xnew)) - volfrac*nelx*nely > 0;
45         l1 = lmid;
46     else
47         l2 = lmid;
48     end
49 end
```

```
50 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51 function [dcn]=check(nelx,nely,rmin,x,dc)
52 dcn=zeros(nely,nelx);
53 for i = 1:nelx
54     for j = 1:nely
55         sum=0.0;
56         for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
57             for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
58                 fac = rmin-sqrt((i-k)^2+(j-l)^2);
59                 sum = sum+max(0,fac);
60                 dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
61             end
62         end
63         dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
64     end
65 end
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 function [U]=FE(nelx,nely,x,penal)
68 [KE] = lk;
69 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
70 F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
71 for elx = 1:nelx
72     for ely = 1:nely
73         n1 = (nely+1)*(elx-1)+ely;
74         n2 = (nely+1)* elx +ely;
75         edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
76         K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
77     end
78 end
79 % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
80 F(2,1) = -1;
81 fixeddofs = union([1:2*2*(nely+1)], [2*(nelx+1)*(nely+1)]);
82 alldofs = [1:2*(nely+1)*(nelx+1)];
83 freedofs = setdiff(alldofs, fixeddofs);
84 % SOLVING
85 U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
86 U(fixeddofs,:)= 0;
87 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
88 function [KE]=lk
89 E = 1.;
90 nu = 0.3;
91 k=[ 1/2-nu/6    1/8+nu/8   -1/4-nu/12   -1/8+3*nu/8 ...
92     -1/4+nu/12  -1/8-nu/8    nu/6        1/8-3*nu/8];
93 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
94                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
95                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
96                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
97                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
98                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
99                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
100                 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```



Topology Optimization: Initial Layout





Topology Optimization: Mathematical Formulation

- Minimum compliance problem

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$



Topology Optimization: Mathematical Formulation

- Minimum compliance problem

Densities $\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$

subject to $\frac{V(\mathbf{x})}{V_0} = f$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$



Topology Optimization: Mathematical Formulation

- Minimum compliance problem

Energy of the system

$$\min_{\mathbf{x}} c(\mathbf{x}) = \boxed{\mathbf{U}^T \mathbf{K} \mathbf{U}} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

subject to $\frac{V(\mathbf{x})}{V_0} = f$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$



Topology Optimization: Mathematical Formulation

- Minimum compliance problem

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

subject to $\frac{V(\mathbf{x})}{V_0} = f$ Desired volume fraction

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$



Topology Optimization: Mathematical Formulation

- Minimum compliance problem

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

Static equilibrium

$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$



Topology Optimization: Mathematical Formulation

- Minimum compliance problem

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

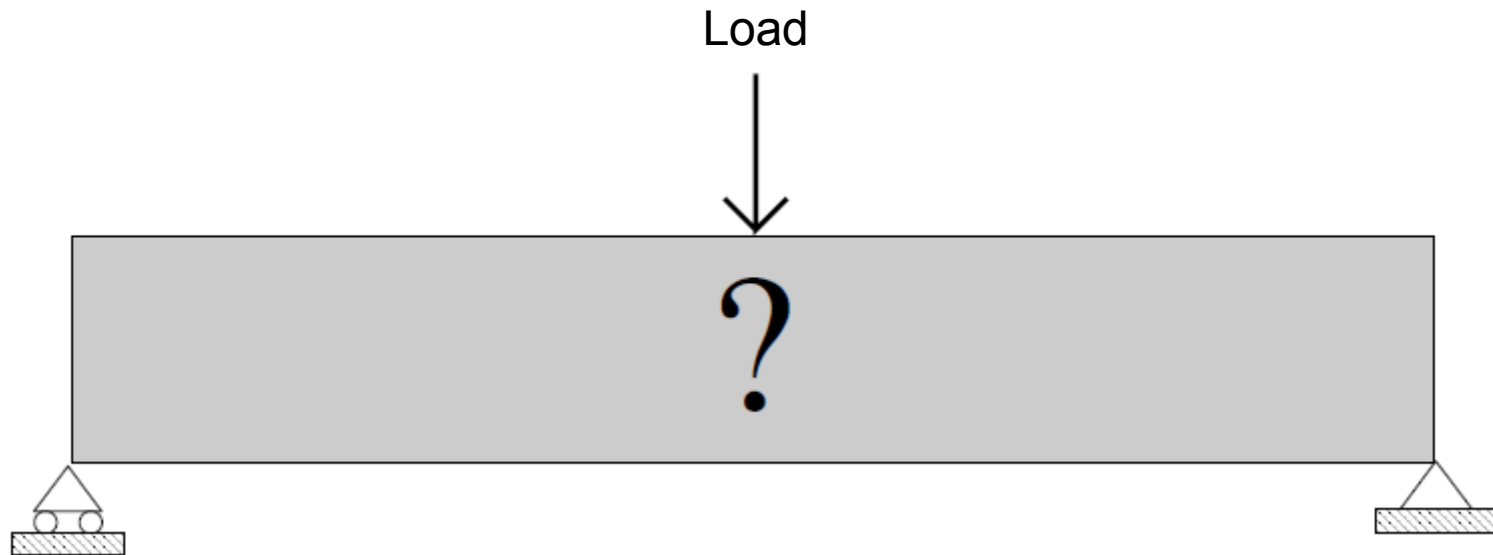
$$\mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1}$$

Valid range for densities



Topology Optimization: Boundary Conditions

- Default boundary conditions: MMB Beam

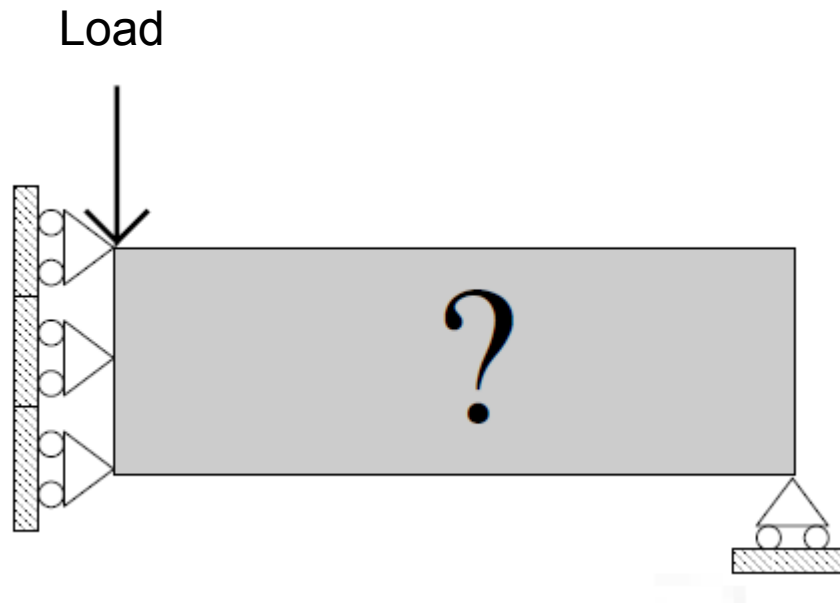


Full domain



Topology Optimization: Boundary Conditions

- Default boundary conditions: MMB Beam

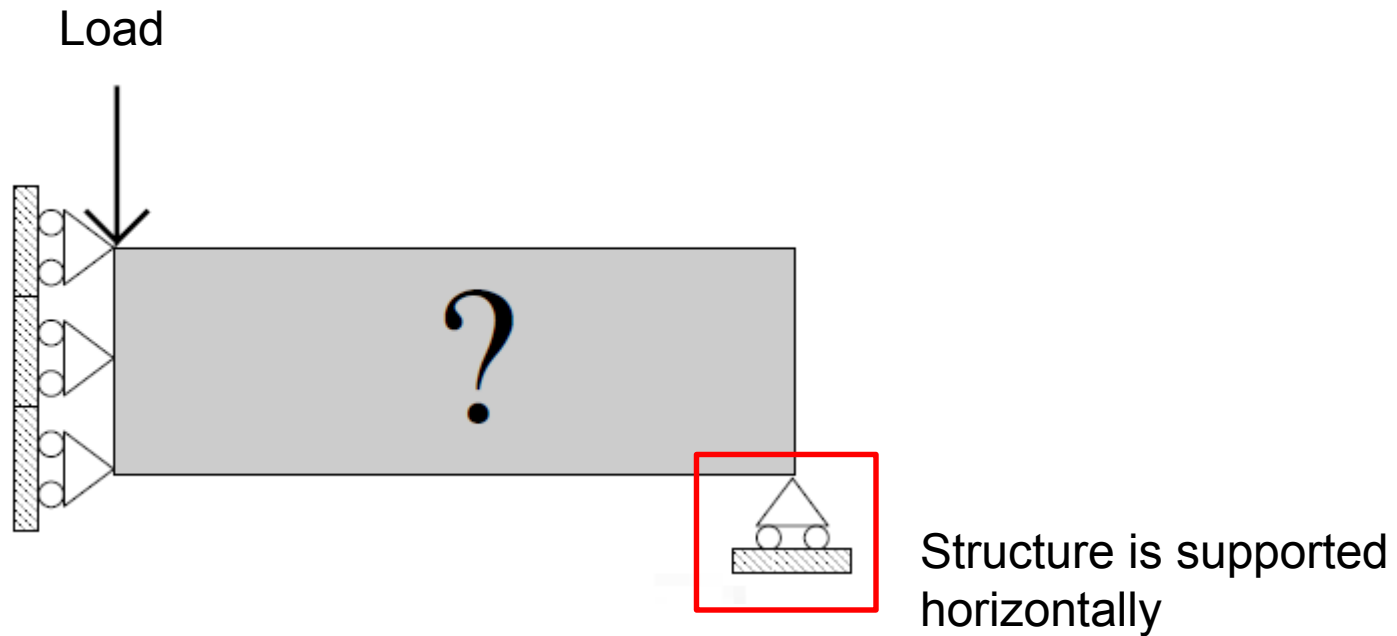


Half design domain



Topology Optimization: Boundary Conditions

- Default boundary conditions: MMB Beam

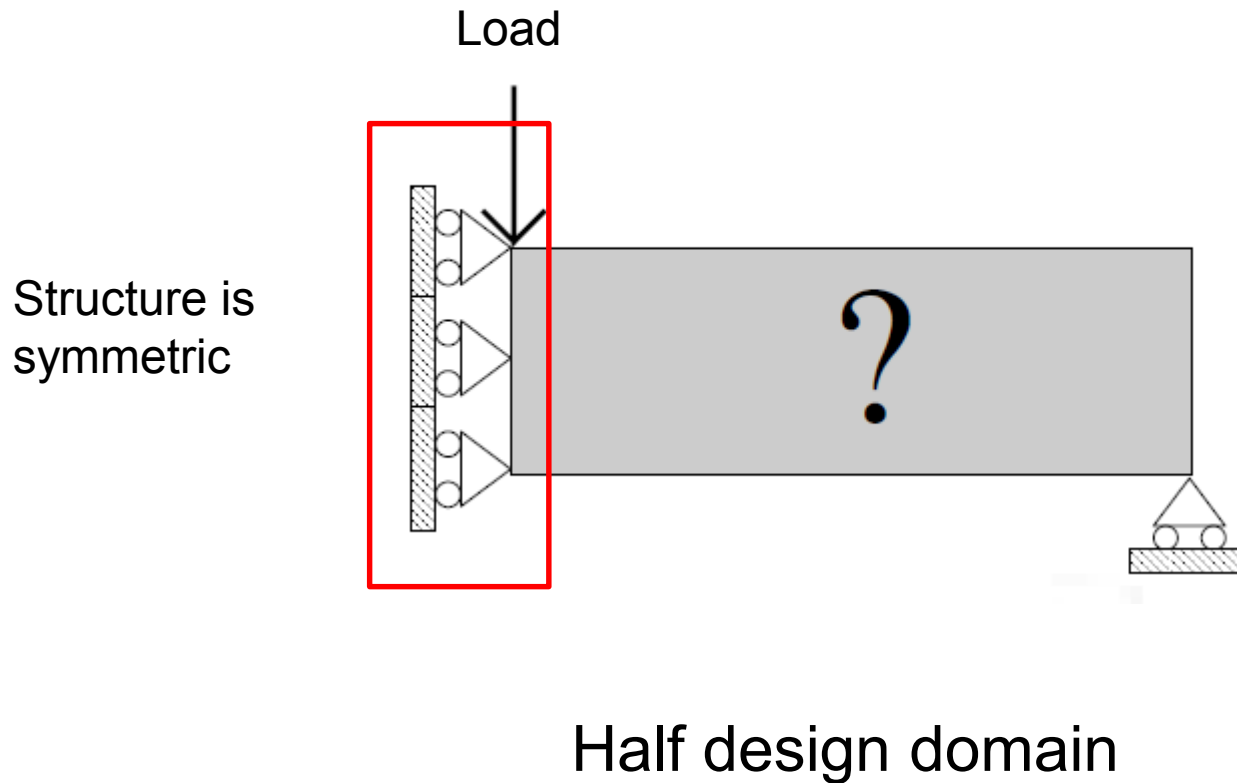


Half design domain



Topology Optimization: Boundary Conditions

- Default boundary conditions: MMB Beam





A 99 Line Topology Optimization Code

```
1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2  %%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
3  function top(nelx,nely,volfrac,penal,rmin)
4  % INITIALIZE
5  x(1:nely,1:nelx) = volfrac;
6  loop = 0;
7  change = 1.;
8  % START ITERATION
9  while change > 0.01
10     loop = loop + 1;
11     xold = x;
12     % FE-ANALYSIS
13     [U]=FE(nelx,nely,x,penal);
14     % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
15     [KE] = lk;
16     c = 0.;
17     for ely = 1:nely
18         for elx = 1:nelx
19             n1 = (nely+1)*(elx-1)+ely;
20             n2 = (nely+1)* elx +ely;
21             Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n1+1;2*n1+2],1);
22             c = c + x(ely,elx)^penal*Ue'*KE*Ue;
23             dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
24         end
25     end
26     % FILTERING OF SENSITIVITIES
27     [dc] = check(nelx,nely,rmin,x,dc);
28     % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
29     [x] = OC(nelx,nely,x,volfrac,dc);
30     % PRINT RESULTS
31     change = max(max(abs(x-xold)));
32     disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
33           ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
34           ' ch.: ' sprintf('%6.3f',change) ])
35     % PLOT DENSITIES
36     colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
37 end
38 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39 function [xnew]=OC(nelx,nely,x,volfrac,dc)
40 l1 = 0; l2 = 100000; move = 0.2;
41 while (l2-l1 > 1e-4)
42     lmid = 0.5*(l2+l1);
43     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid))));
44     if sum(sum(xnew)) - volfrac*nelx*nely > 0;
45         l1 = lmid;
46     else
47         l2 = lmid;
48     end
49 end
```

```
50 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51 function [dcn]=check(nelx,nely,rmin,x,dc)
52 dcn=zeros(nely,nelx);
53 for i = 1:nelx
54     for j = 1:nely
55         sum=0.0;
56         for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
57             for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
58                 fac = rmin-sqrt((i-k)^2+(j-l)^2);
59                 sum = sum+max(0,fac);
60                 dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
61             end
62         end
63         dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
64     end
65 end
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 function [U]=FE(nelx,nely,x,penal)
68 [KE] = lk;
69 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
70 F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
71 for elx = 1:nelx
72     for ely = 1:nely
73         n1 = (nely+1)*(elx-1)+ely;
74         n2 = (nely+1)* elx +ely;
75         edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n1+1; 2*n1+2];
76         K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
77     end
78 end
79 % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
80 F(2,1) = -1;
81 fixeddofs = union([1:2:2*(nely+1)], [2*(nelx+1)*(nely+1)]);
82 alldofs = [1:2*(nely+1)*(nelx+1)];
83 freedofs = setdiff(alldofs, fixeddofs);
84 % SOLVING
85 U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
86 U(fixeddofs,:) = 0;
87 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
88 function [KE]=lk
89 E = 1.;
90 nu = 0.3;
91 k=[ 1/2-nu/6    1/8+nu/8   -1/4-nu/12   -1/8+3*nu/8 ...
92    -1/4+nu/12   -1/8-nu/8    nu/6        1/8-3*nu/8];
93 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
94                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
95                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
96                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
97                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
98                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
99                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
100                 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```



A 99 Line Topology Optimization Code

- API

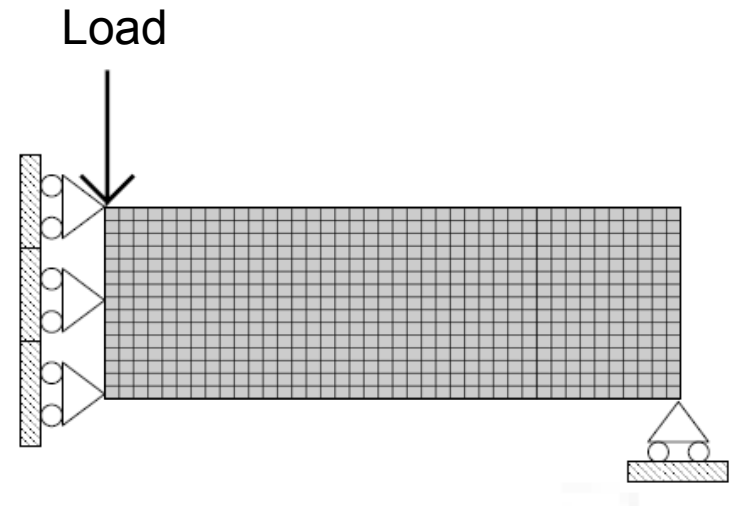
```
function top(nelx,nely,volfrac,penal,rmin)
```

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$0 < \mathbf{x}_{\min} \leq \mathbf{x} \leq 1$$





A 99 Line Topology Optimization Code

- API

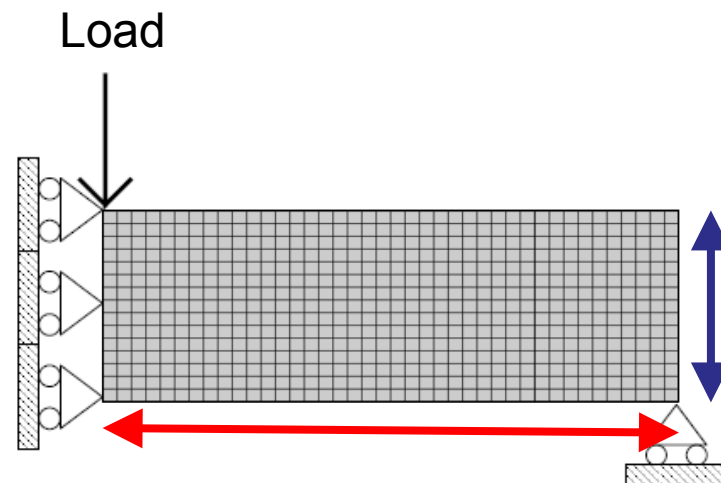
```
function top(nelx,nely,volfrac,penal,rmin)
```

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$0 < \mathbf{x}_{\min} \leq \mathbf{x} \leq 1$$





A 99 Line Topology Optimization Code

- API

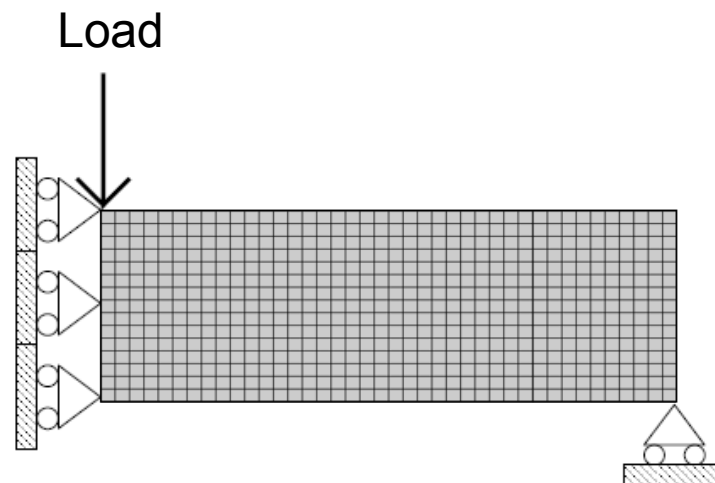
```
function top(nelx,nely,volfrac,penal,rmin)
```

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$0 < \mathbf{x}_{\min} \leq \mathbf{x} \leq 1$$





A 99 Line Topology Optimization Code

- API

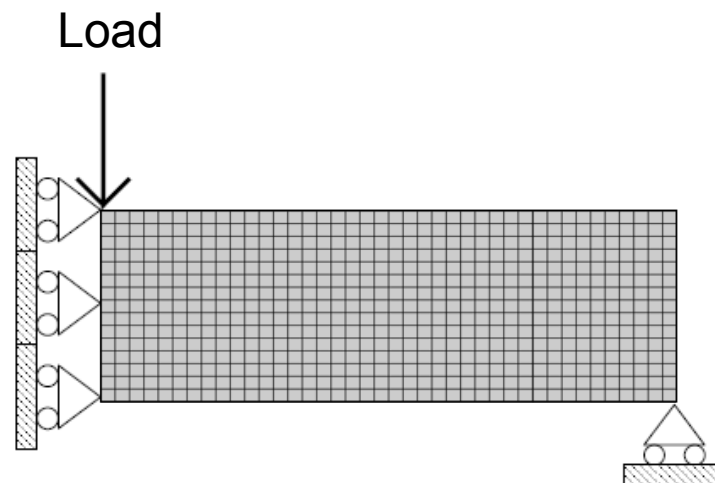
```
function top(nelx,nely,volfrac,penal,rmin)
```

$$\min_{\mathbf{x}} c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

$$\text{subject to } \frac{V(\mathbf{x})}{V_0} = f$$

$$\mathbf{K} \mathbf{U} = \mathbf{F}$$

$$0 < \mathbf{x}_{\min} \leq \mathbf{x} \leq 1$$





A 99 Line Topology Optimization Code

- API

```
function top(nelx,nely,volfrac,penal,rmin)
```

- An example call

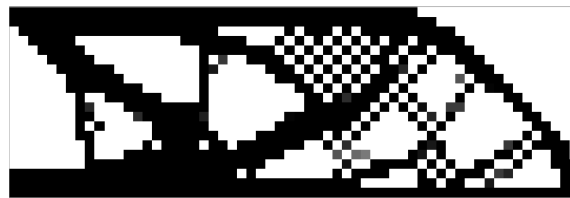
```
top(60,20,0.5,3.0,1.5);
```





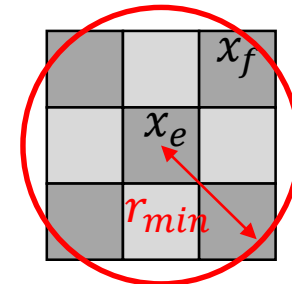
Filter

- With no filter, checkerboard patterns may appear in the solution



- Densities can be smoothed using a convolution operator

$$\bullet \text{API}_{x_e} = \frac{1}{\sum_f H_f} \sum_f H_f x_f \quad H_f = r_{\min} - \text{dist}(e, f)$$

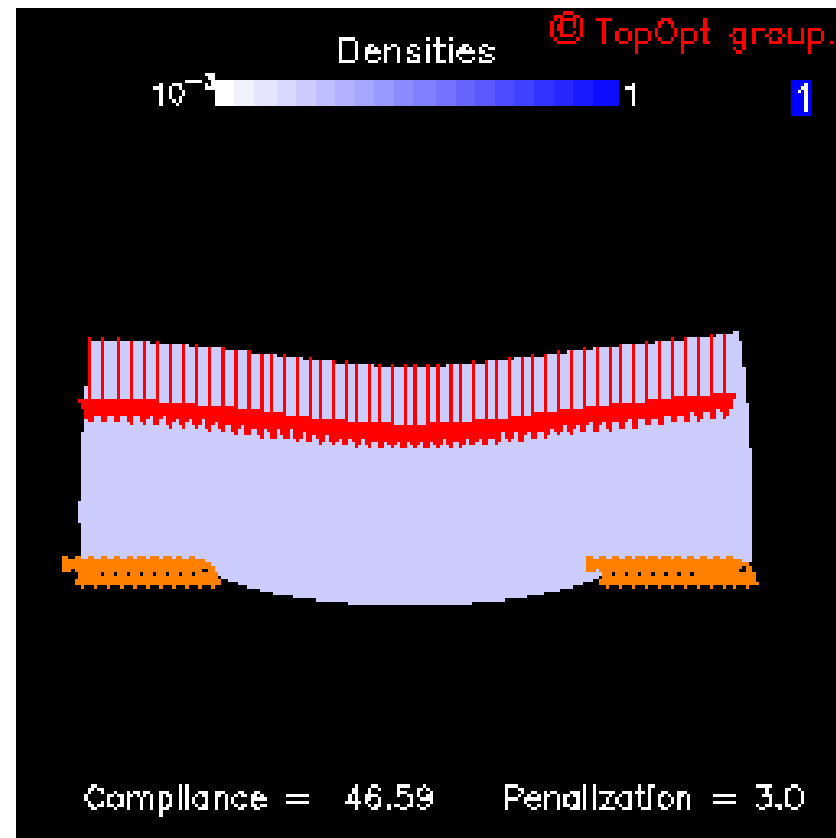


```
function top(nelx,nely,volfrac,penal,rmin)
```



Applications: Structural Optimization

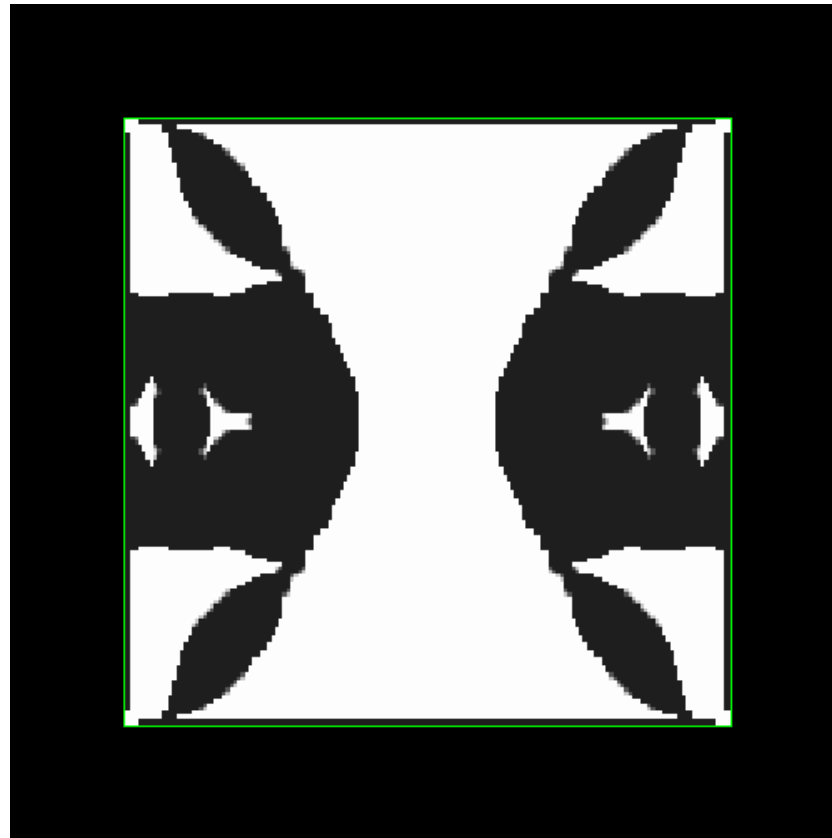
- Bridge Design





Applications: Structural Optimization

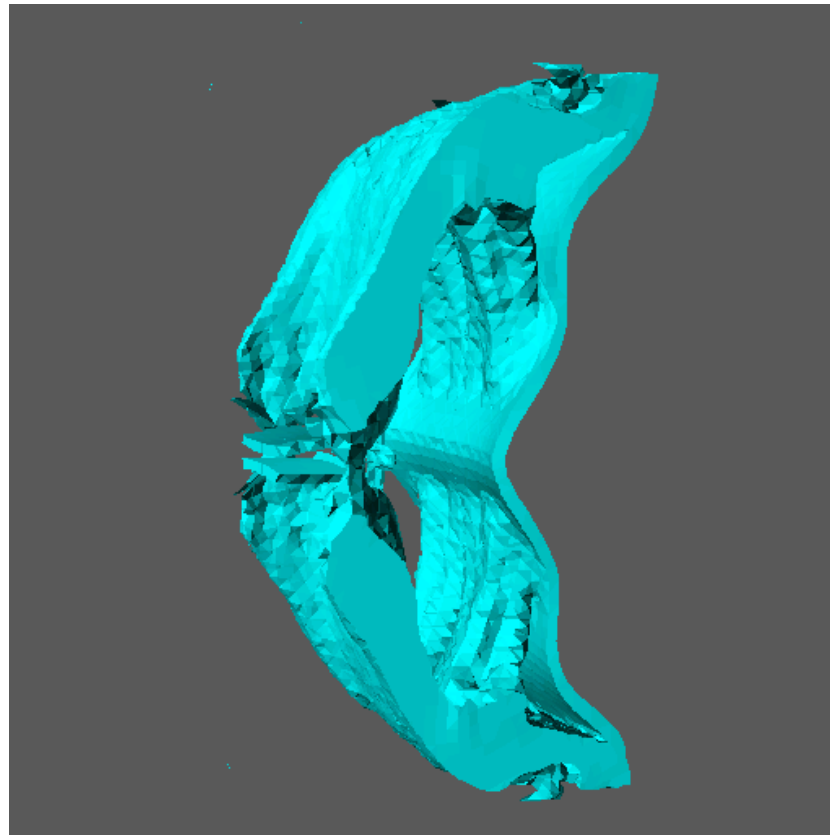
- Negative Poisson's ratio





Applications: Design of Compliant Mechanisms

- Gripper



Inverse Methods: Multi-Objective Optimization

Multiple Performance Objectives



Performance metric:
flexibility



Performance metric:
weight



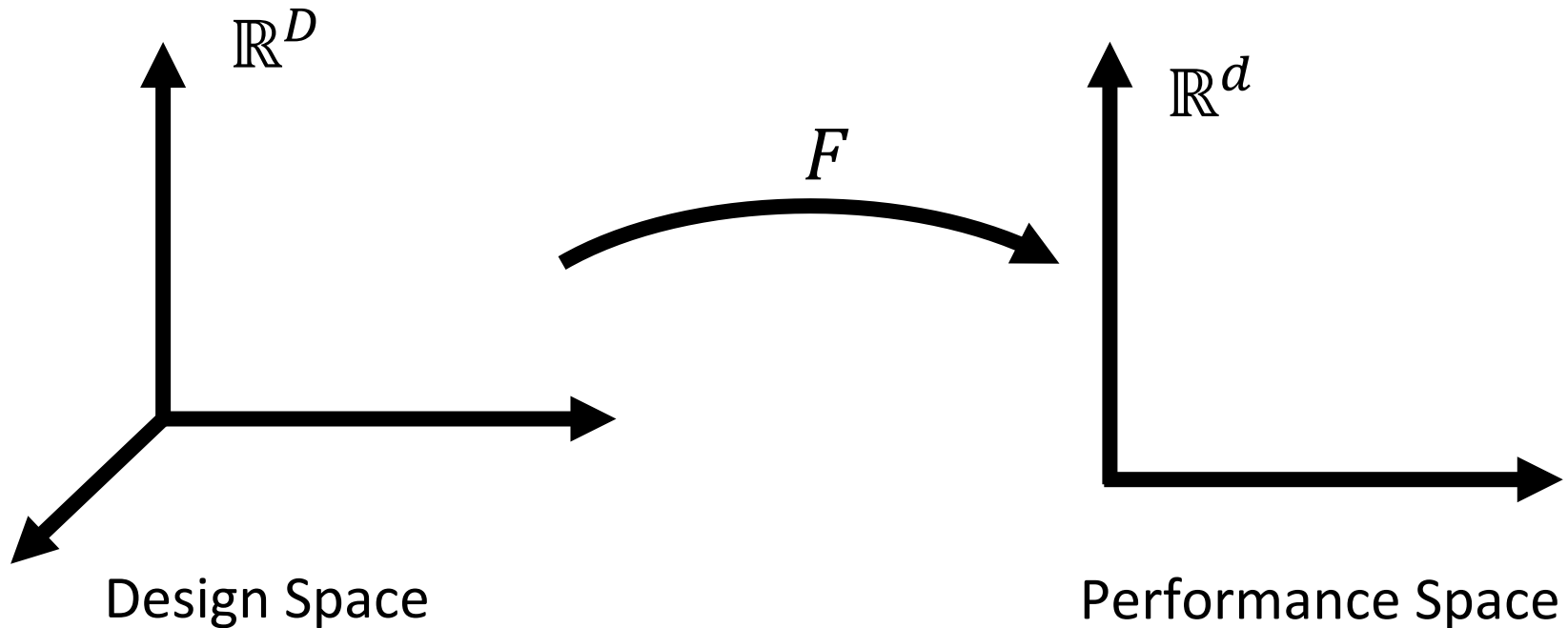
Performance metric:
stability

Multi-Objective Optimization

$$\min f_i(x), \quad i = 1, \dots, d \quad x \in \mathbb{R}^D$$

$$\text{Subject to} \quad g(x) \geq, \quad h(x) = 0$$

$$F(x) = [f_1(x), \dots, f_d(x)]$$



Multi-Objective Optimization

$$\min f_i(x), \quad i = 1, \dots, d$$

$$\text{Subject to } g(x) \geq, \quad h(x) = 0$$

$$F(x) = [f_1(x), \dots, f_d(x)]$$

We know how to do this:

$$\min f(x)$$

Multi-Objective Optimization

$$\min f_i(x), \quad i = 1, \dots, d$$

$$\text{Subject to } g(x) \geq, \quad h(x) = 0$$

$$F(x) = [f_1(x), \dots, f_d(x)]$$

We know how to do this:

$$\min f(x)$$

$$\text{Solution: } f(x) = \sum_i w_i f_i(x)$$

Multi-Objective Optimization

$$\min f_i(x), \quad i = 1, \dots, d$$

$$\text{Subject to } g(x) \geq, \quad h(x) = 0$$

$$F(x) = [f_1(x), \dots, f_d(x)]$$

We know how to do this:

$$\min f(x)$$

$$\text{Solution: } f(x) = \sum w_i f_i(x)$$

How do you pick the weights?

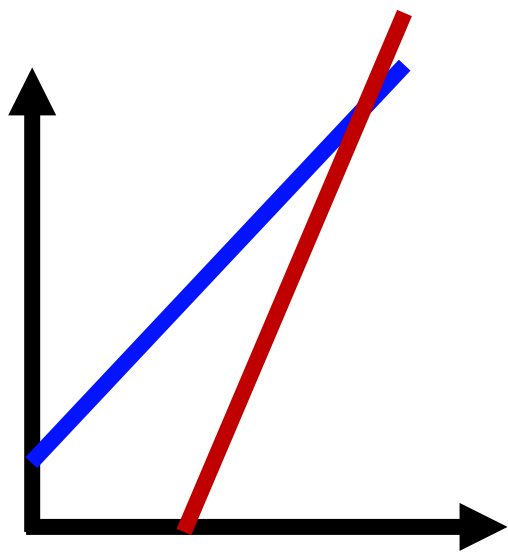
Do the Weights Matter?

Example:

$$f_1(x) = 2x - 5$$

$$f_2(x) = x + 3$$

$$0 \leq x \leq 1$$



$$f(x) = w_1 f_1(x) + w_2 f_2(x)$$

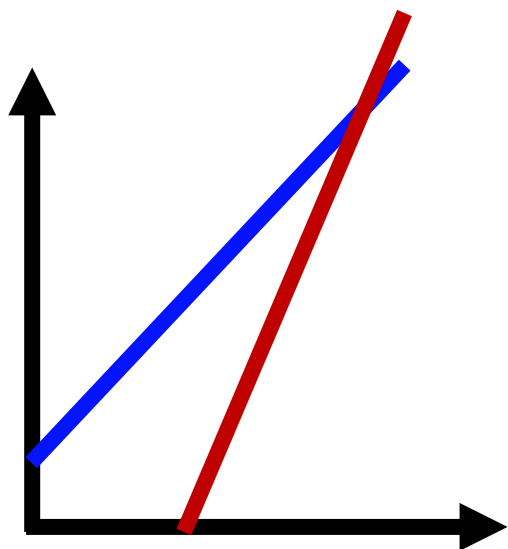
Do the Weights Matter?

Example:

$$f_1(x) = 2x - 5$$

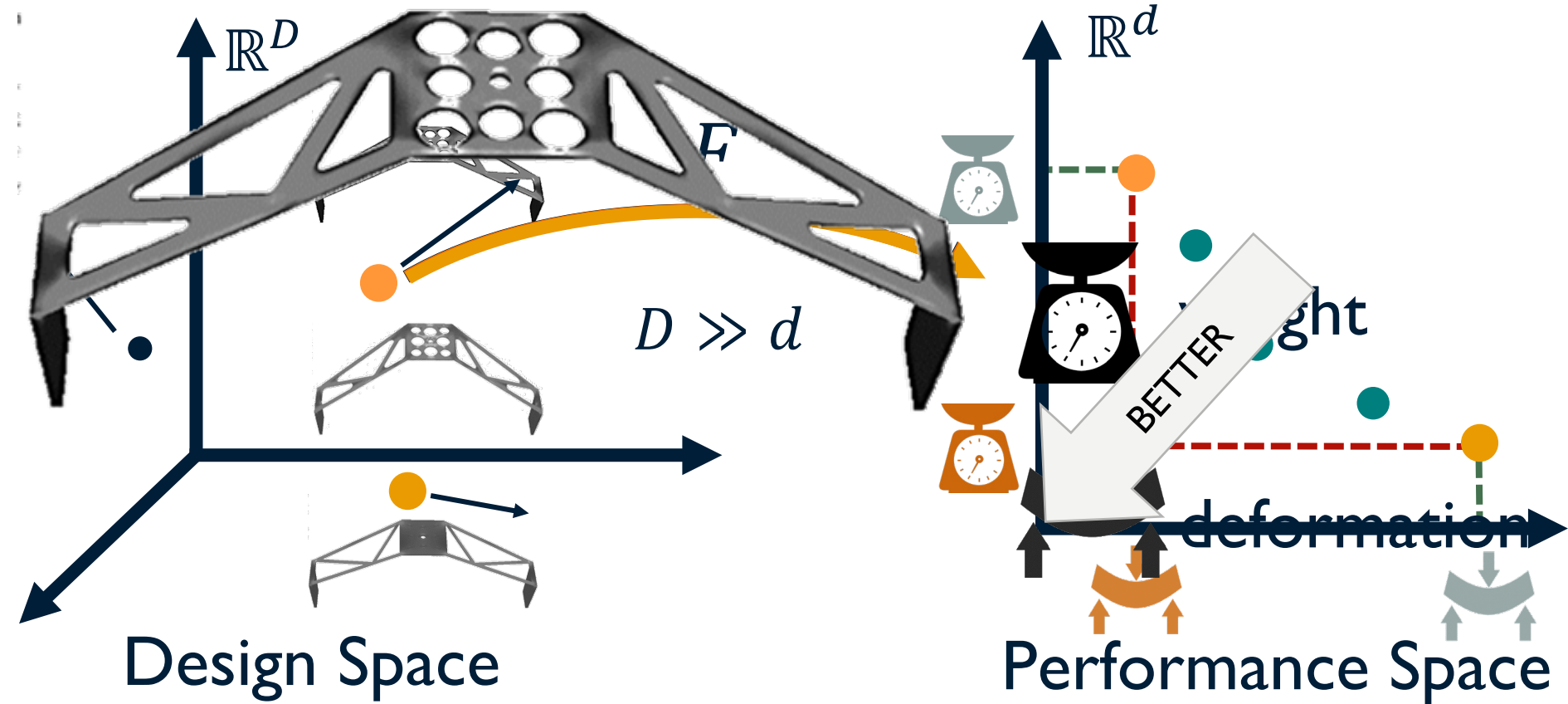
$$f_2(x) = x + 3$$

$$0 \leq x \leq 1$$



$$f(x) = w_1 f_1(x) + w_2 f_2(x)$$

No matter what weights you pick
 $\arg \min f(x) = 0$



Experiment

(A)



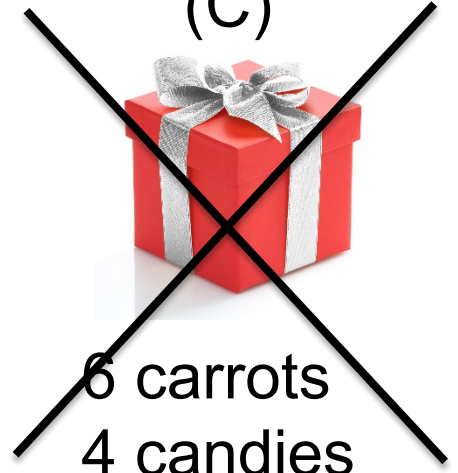
3 carrots
8 candies

(B)



6 carrots
6 candies

(C)



6 carrots
4 candies

Experiment

(A)



3 carrots
8 candies

(B)



6 carrots
6 candies

(C)



6 carrots
4 candies

(D)



7 carrots
3 candies

(E)



5 carrots
7 candies

(F)



4 carrots
9 candies

Definition: Dominance

(A)



3 carrots
8 candies

(B)



6 carrots
6 candies

(C)



6 carrots
4 candies

(D)



7 carrots
3 candies

(E)



5 carrots
7 candies

(F)



4 carrots
9 candies

Definition: Dominance

(A)



3 carrots
8 candies

(B)



6 carrots
6 candies

(C)



6 carrots
4 candies

(D)



7 carrots
3 candies

(E)



5 carrots
7 candies

(F)



4 carrots
9 candies

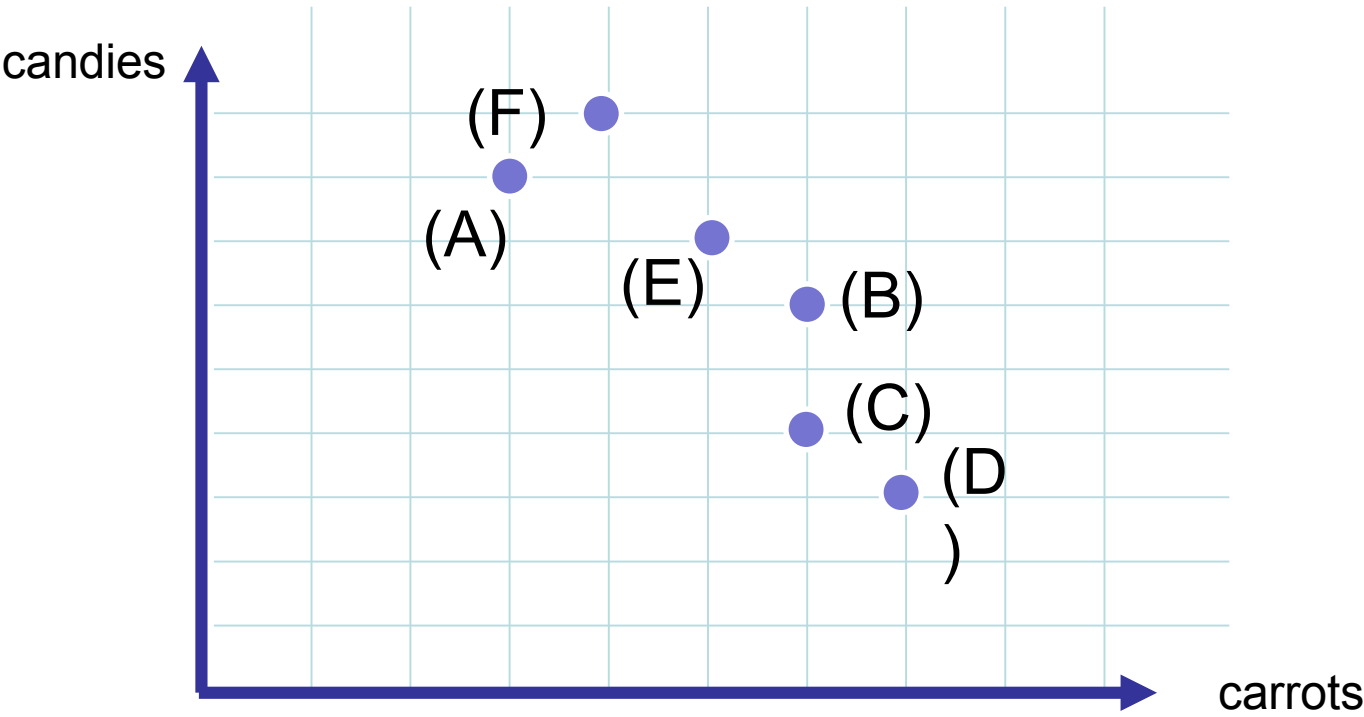
Pareto Optimality

A solution x_1 is said to dominate the other solution x_2 , if both the following conditions are true:

1. The solution x_1 is no worse than x_2 in all objectives.
2. The solution x_1 is strictly better than x_2 in at least one objective.

A point is **Pareto optimal** if it is not dominated by any point: called **non-dominated** point

Let's Plot this



- (A)



3 carrots
8 candies
- (B)



6 carrots
6 candies
- (C)



6 carrots
4 candies
- (D)



7 carrots
3 candies
- (E)

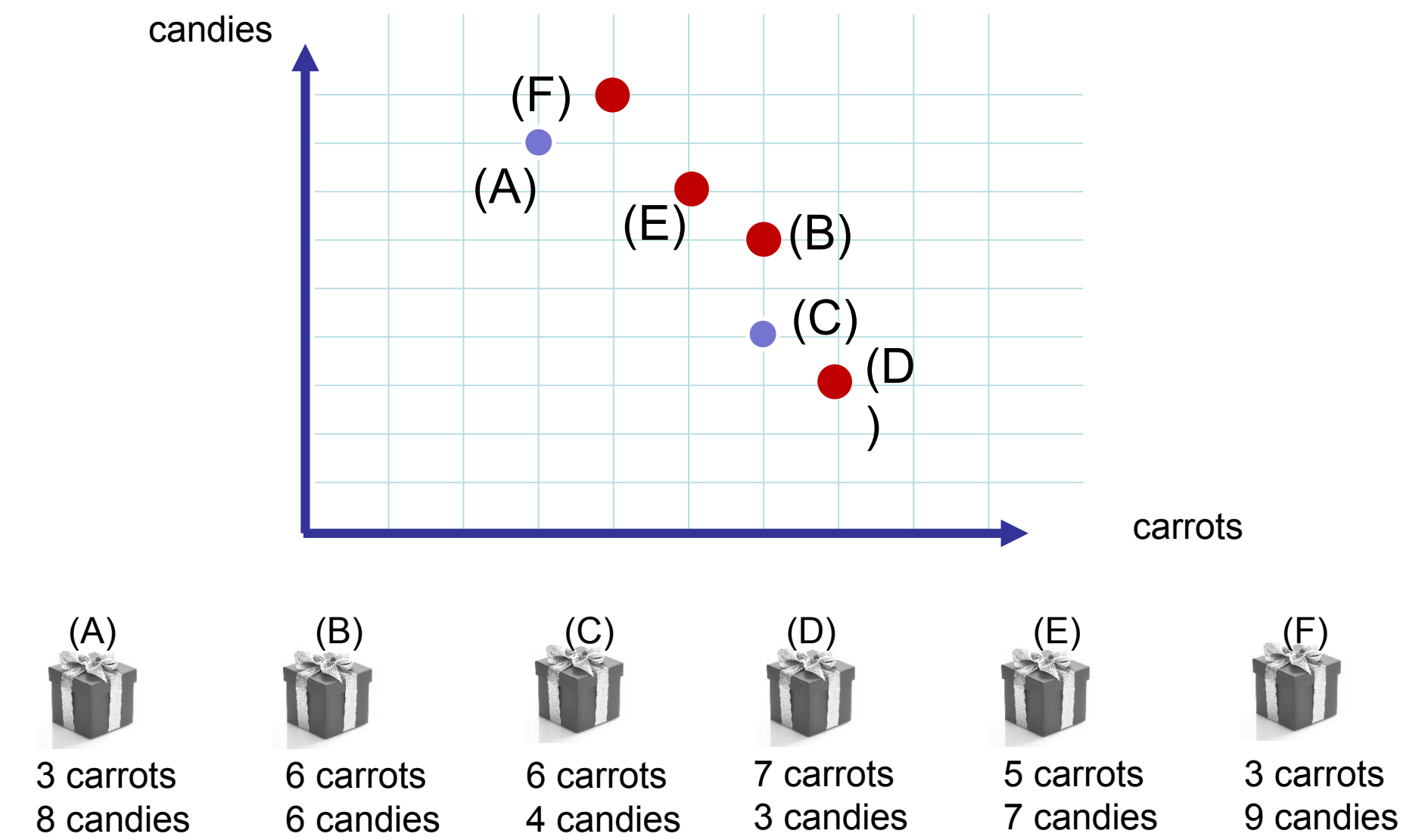


5 carrots
7 candies
- (F)

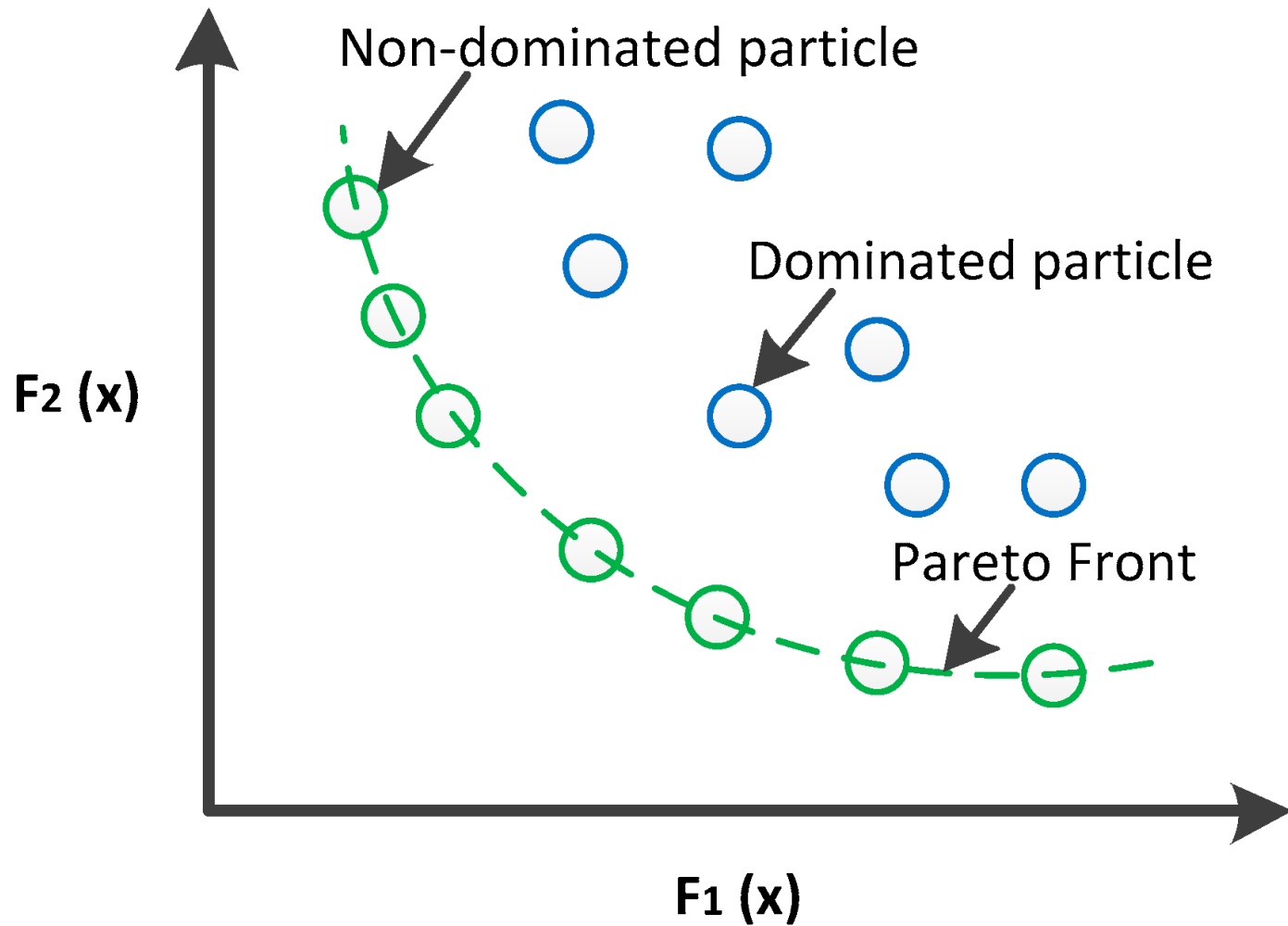


4 carrots
9 candies

Pareto Front

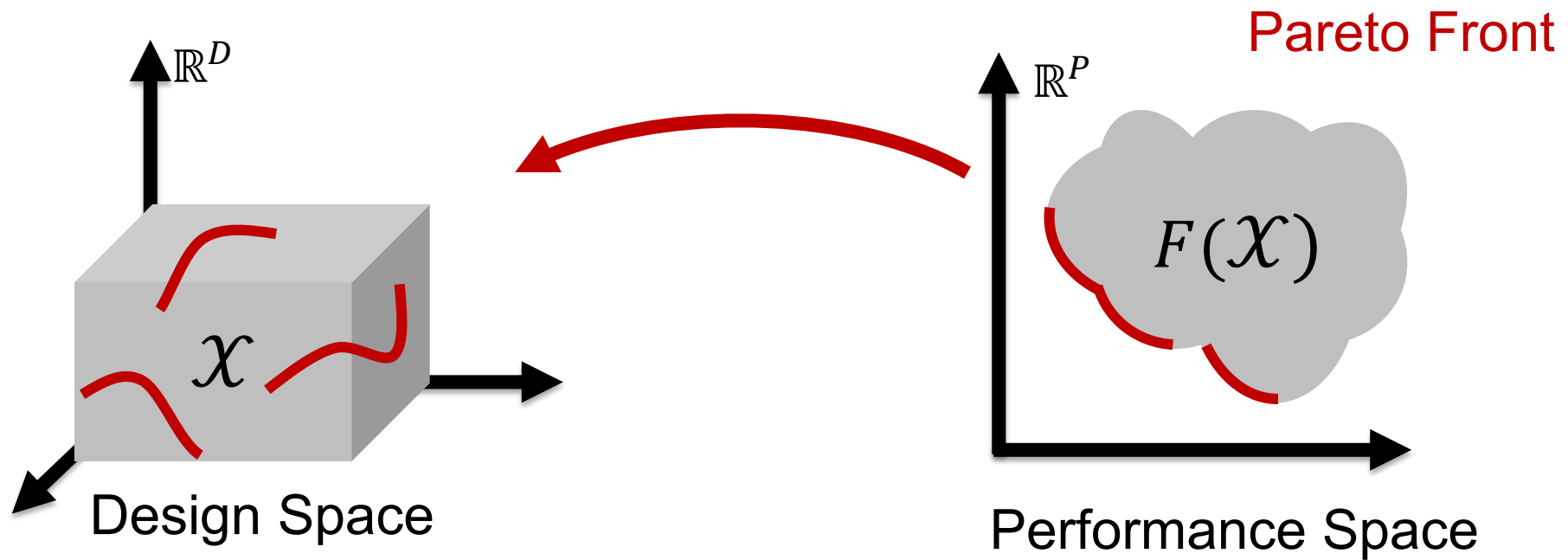


For Minimization



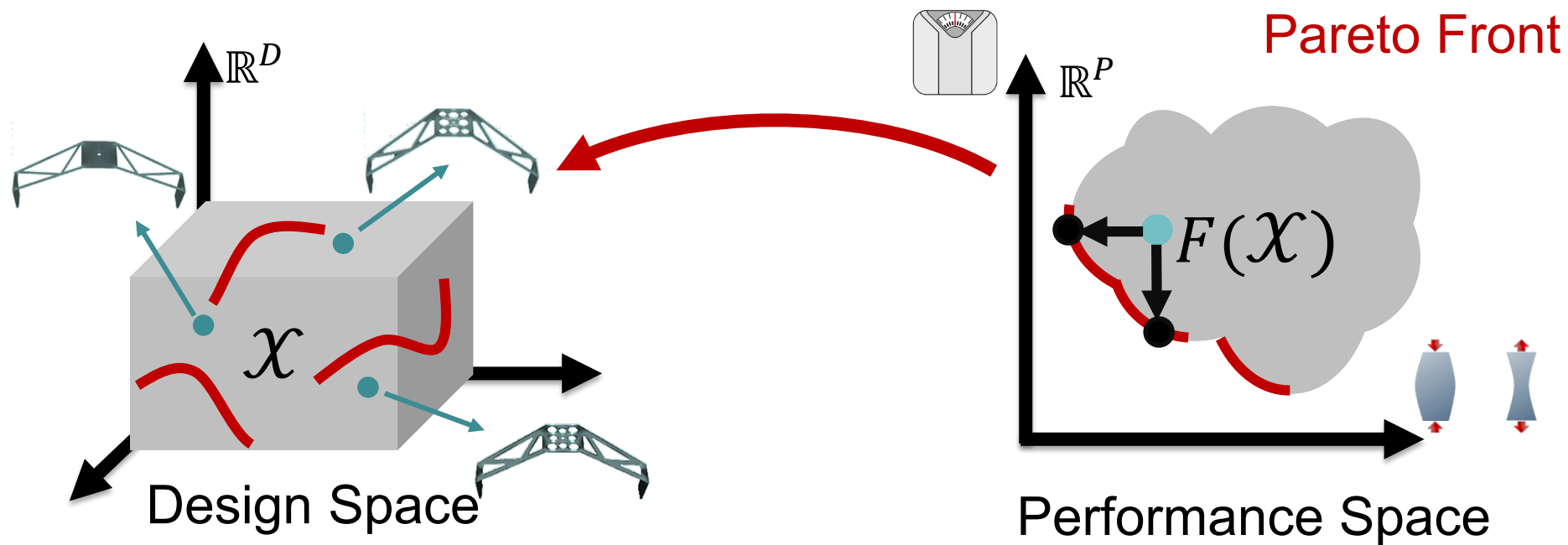


Space of Optimal Solutions



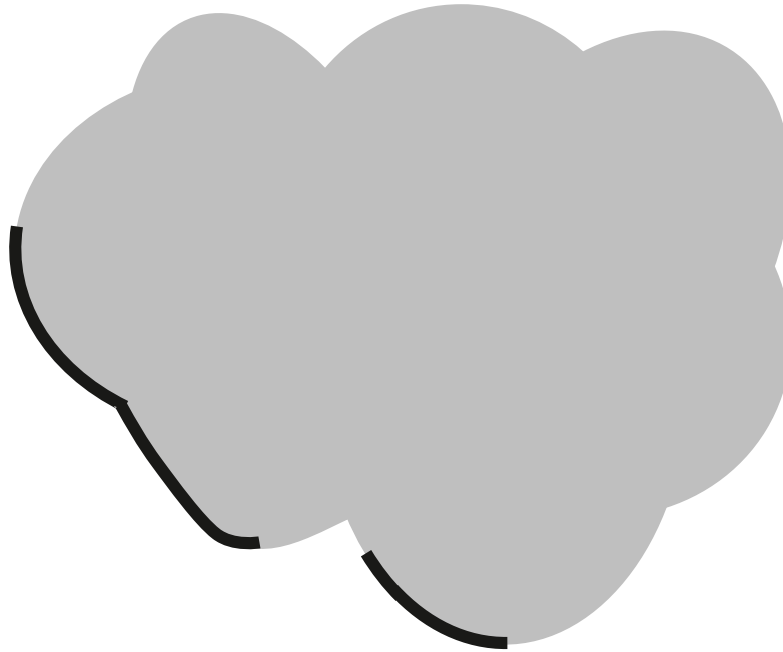


Space of Optimal Solutions





The Geometry of the Front

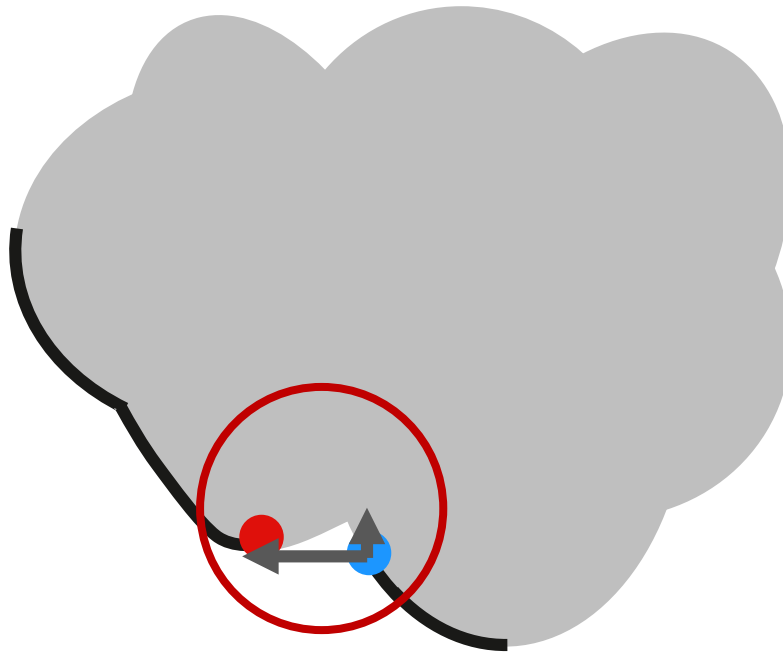


Not a straight line!

Solution: $f(x) = \sum_i w_i f_i(x)$ ☹

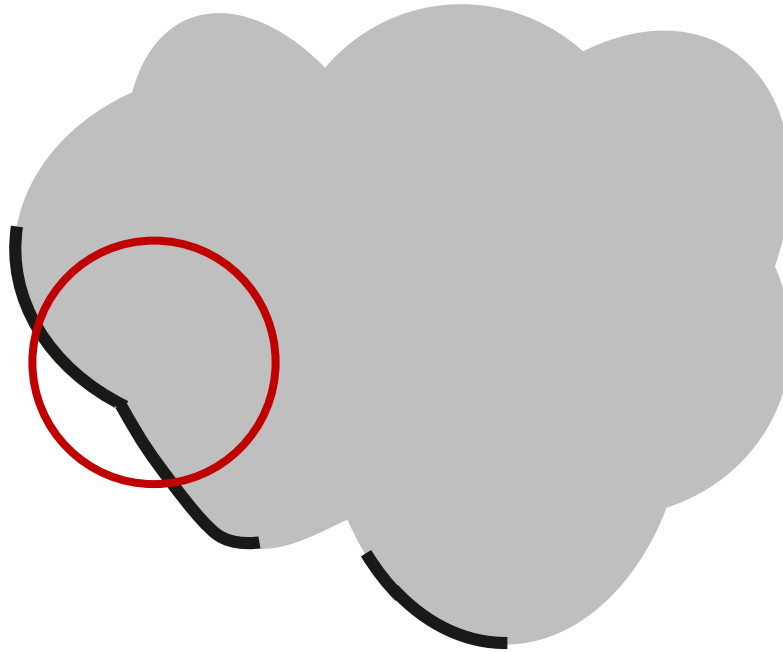


The Front Can Have Gaps





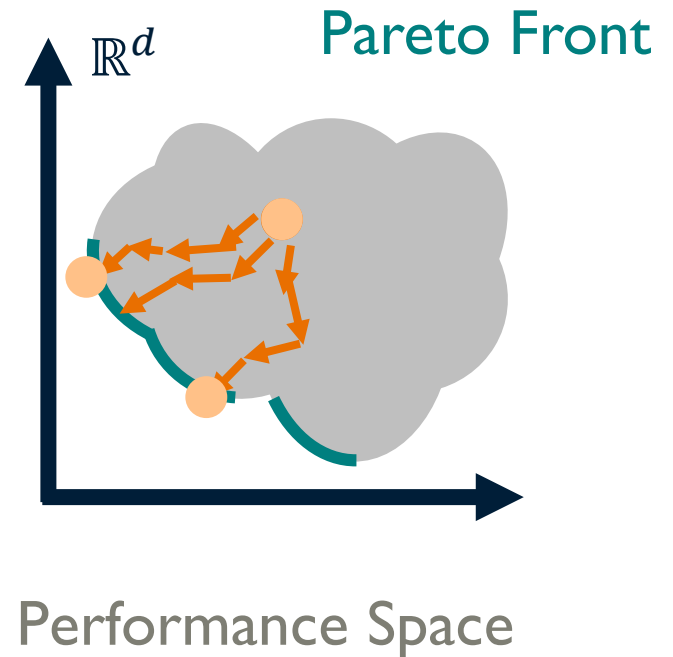
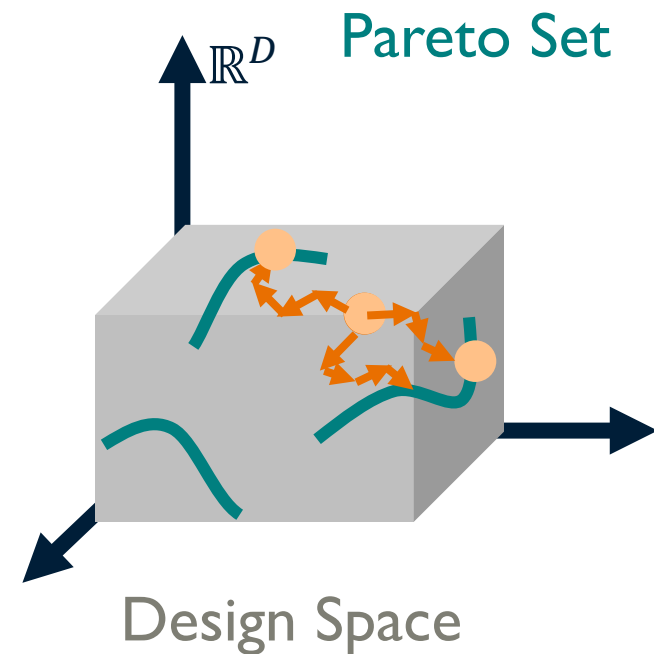
The Front Can Have Non-Convex Regions



Pareto Front Discovery

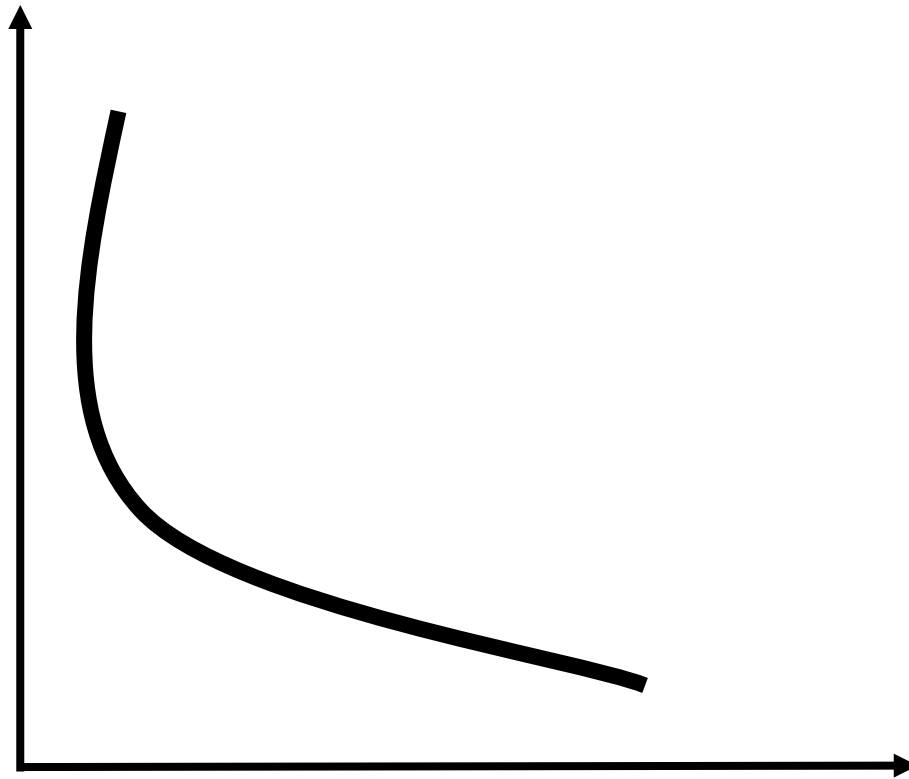
Main Challenge:

- Converge to optimal solutions
- Diverse set that describes the full front



Weighted Sum Method

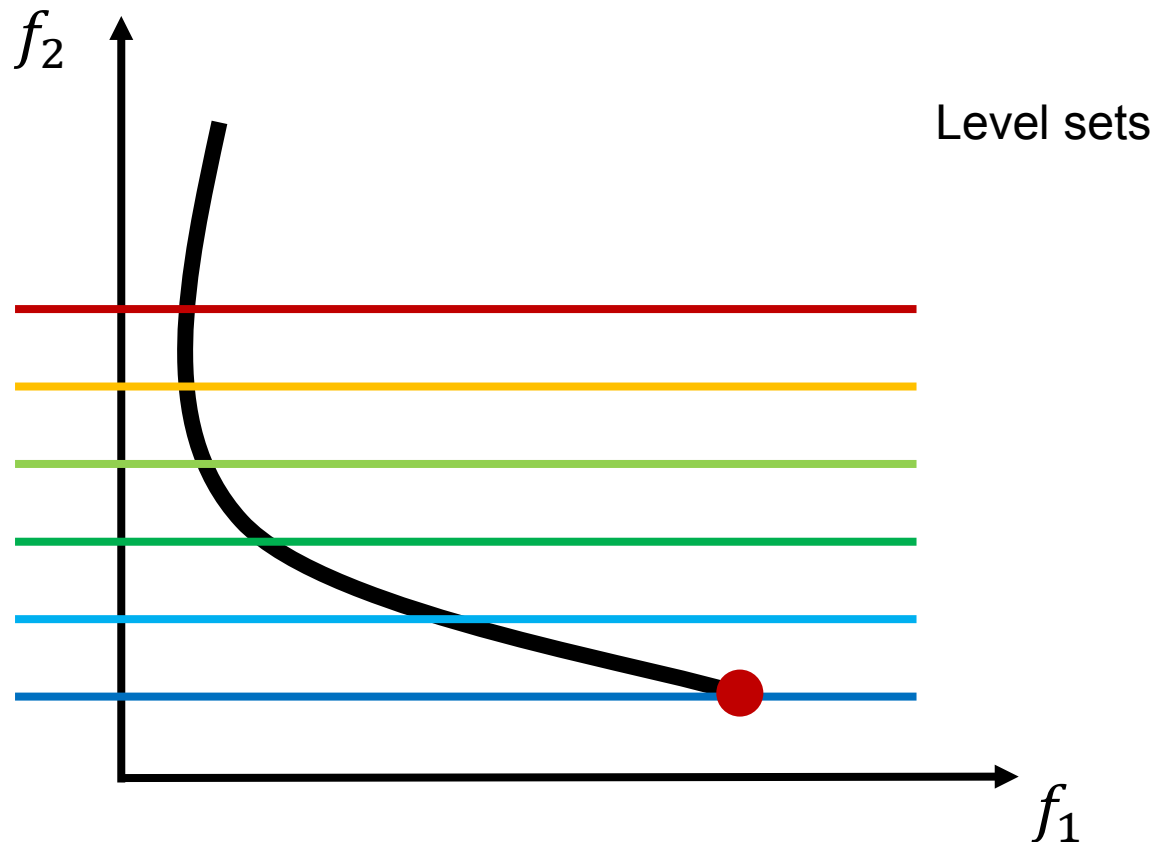
Vary w and solve $f(x) = wf_1 + (1 - w)f_2$



Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

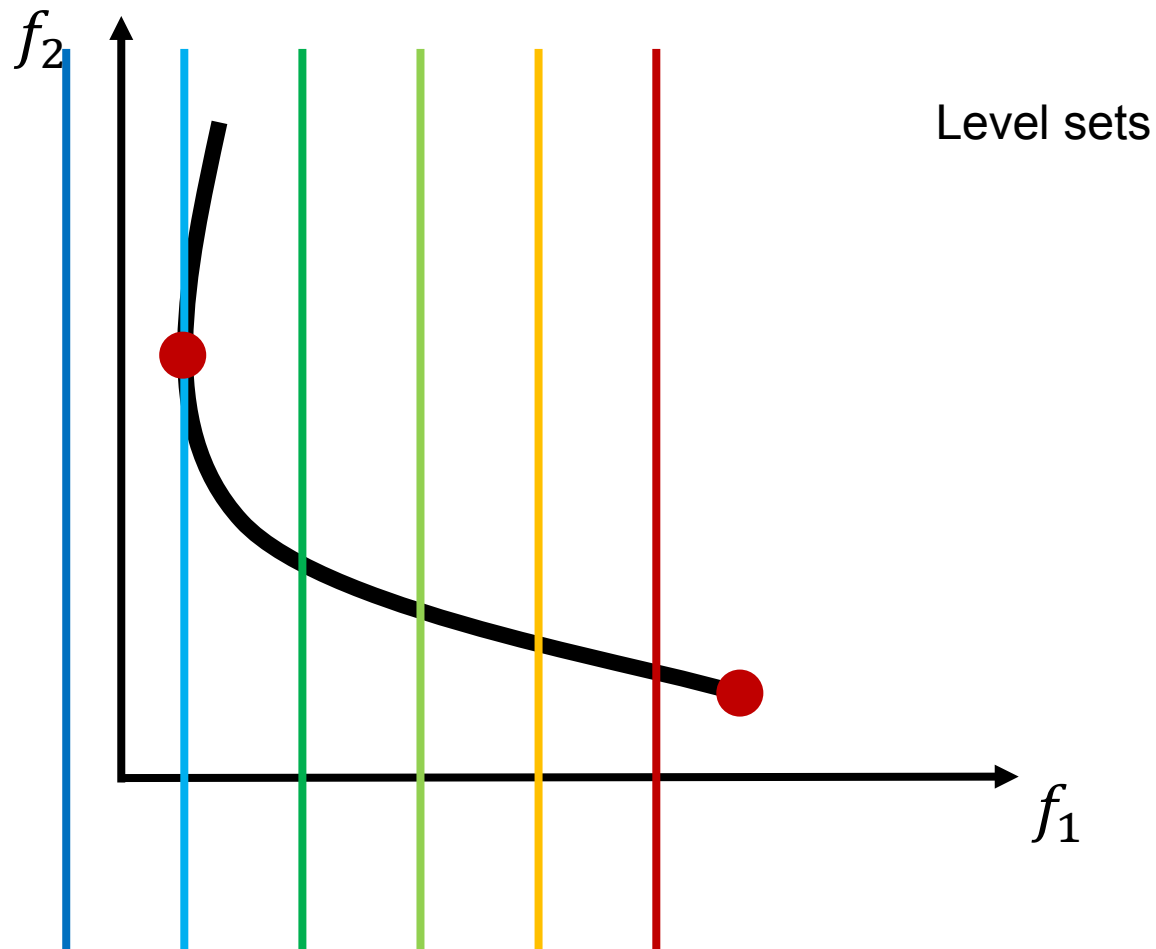
$w = 0$



Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

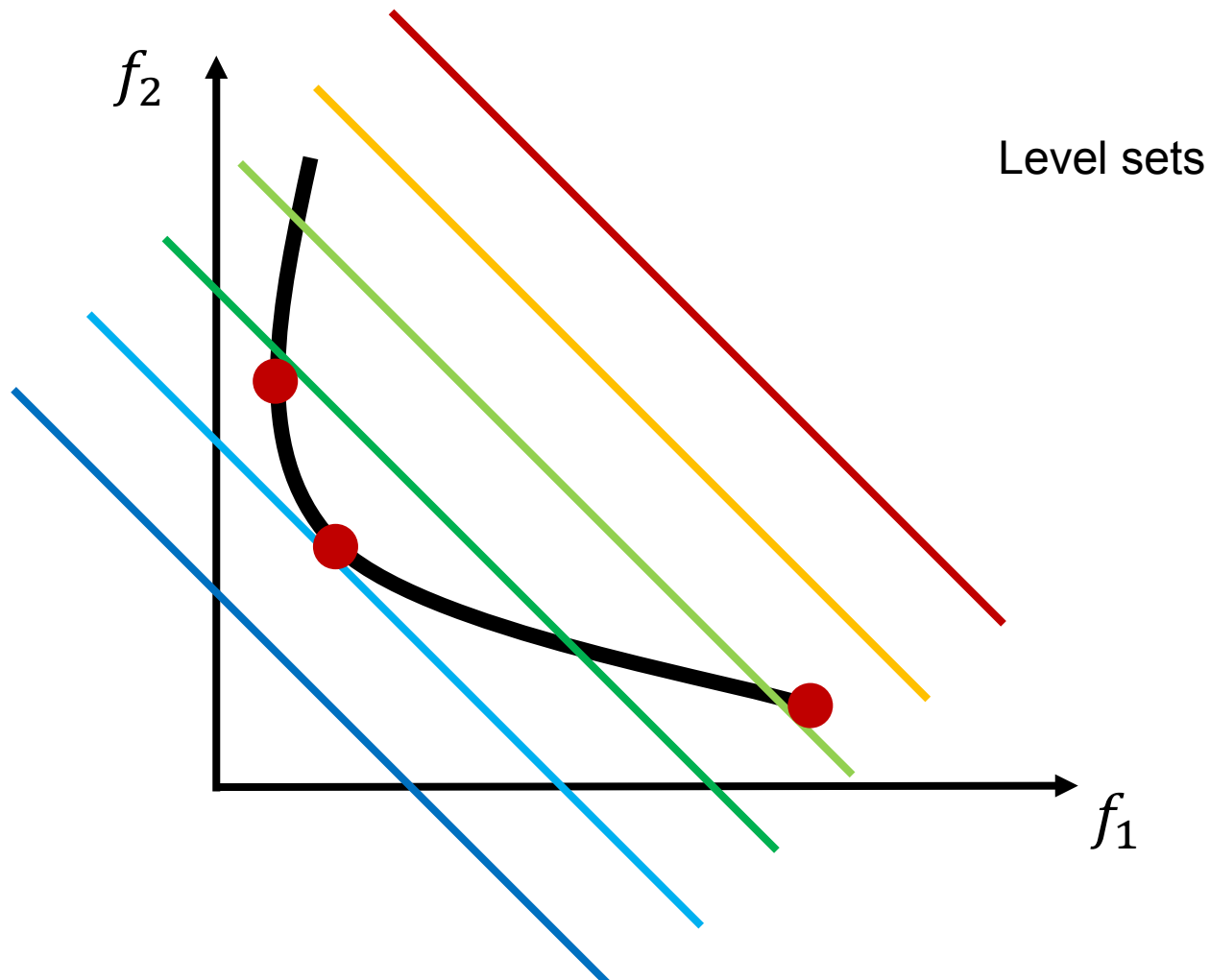
$w = 1$



Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

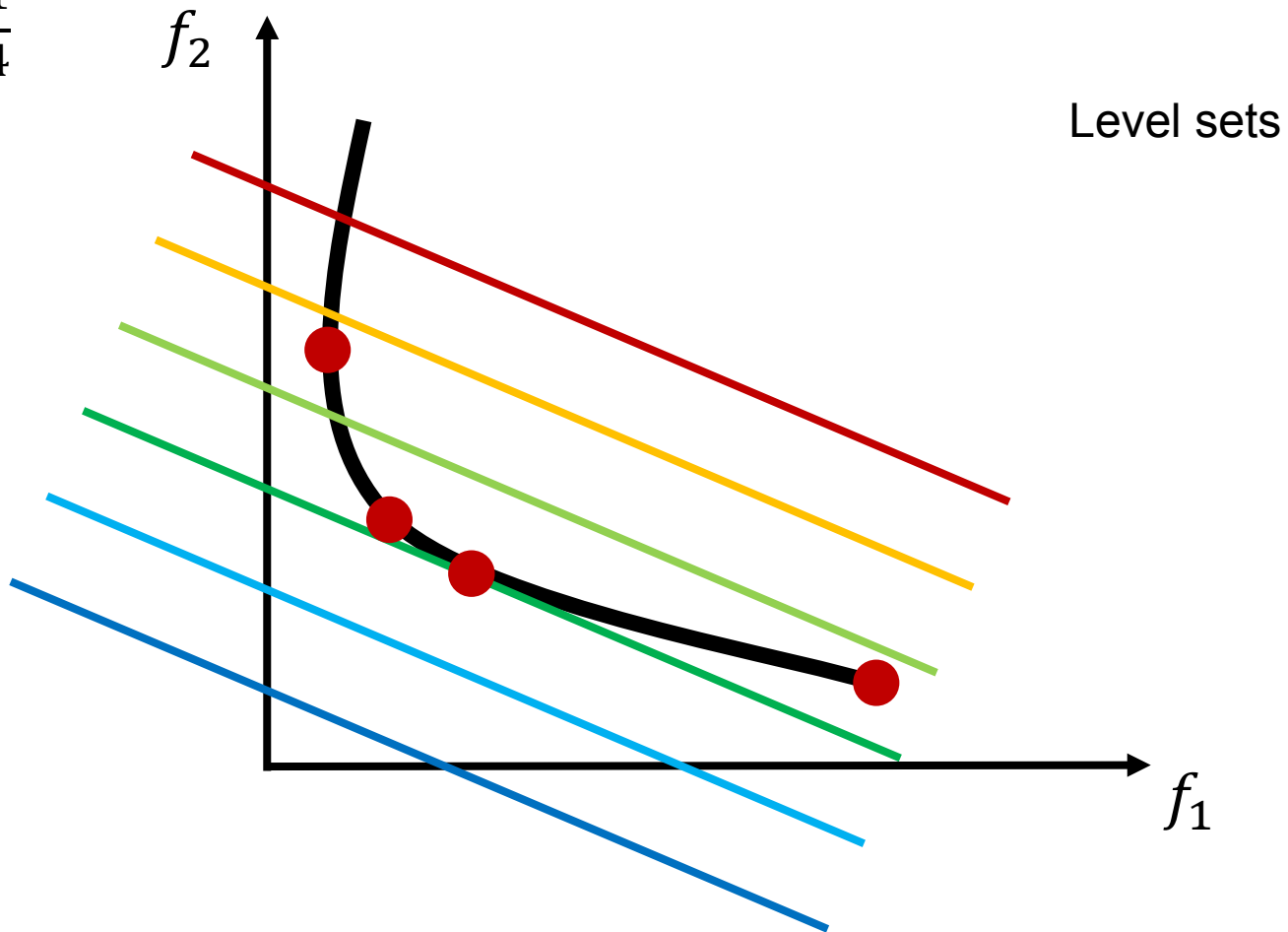
$$w = \frac{1}{2}$$



Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

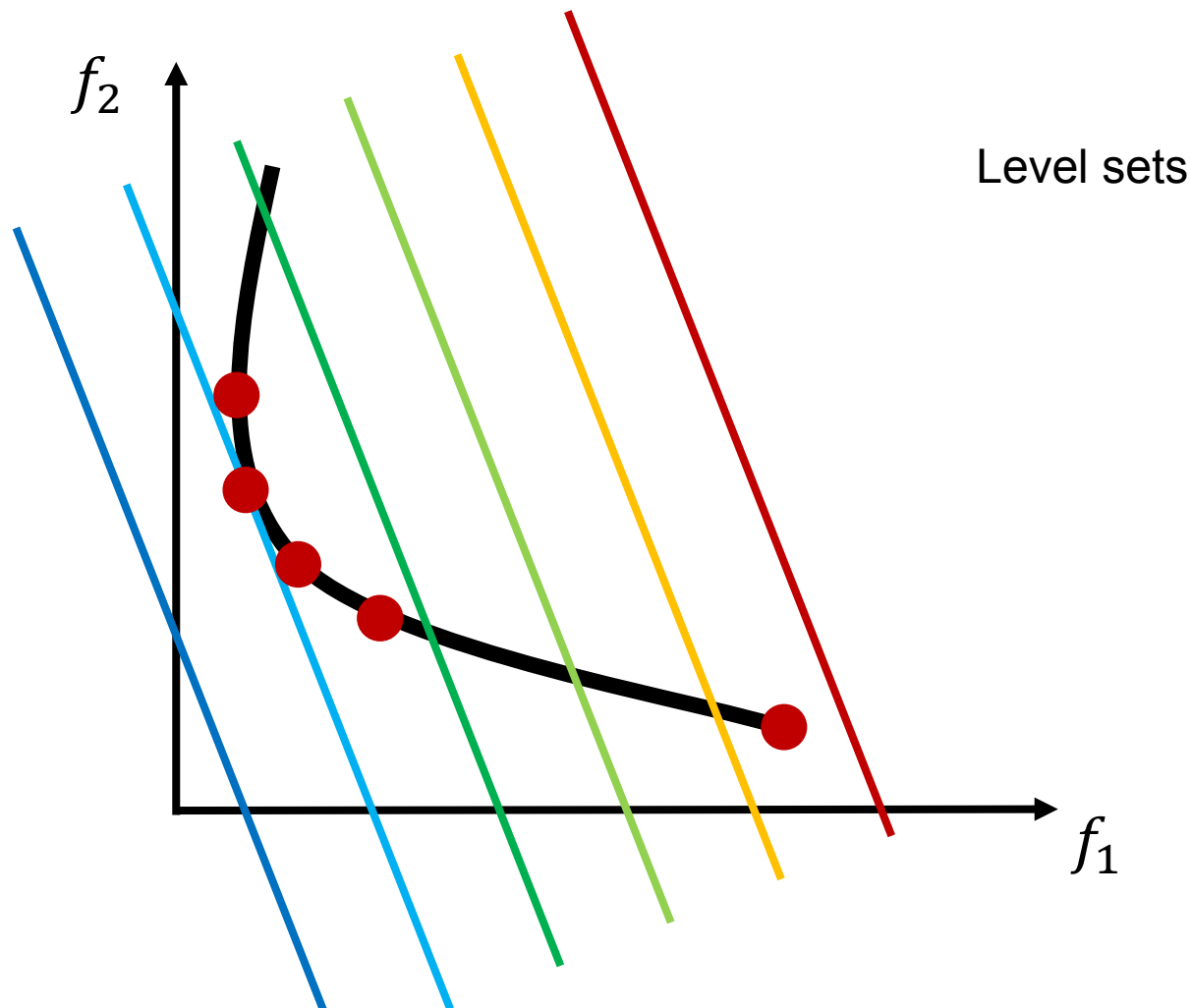
$$w = \frac{1}{4}$$



Weighted Sum Method

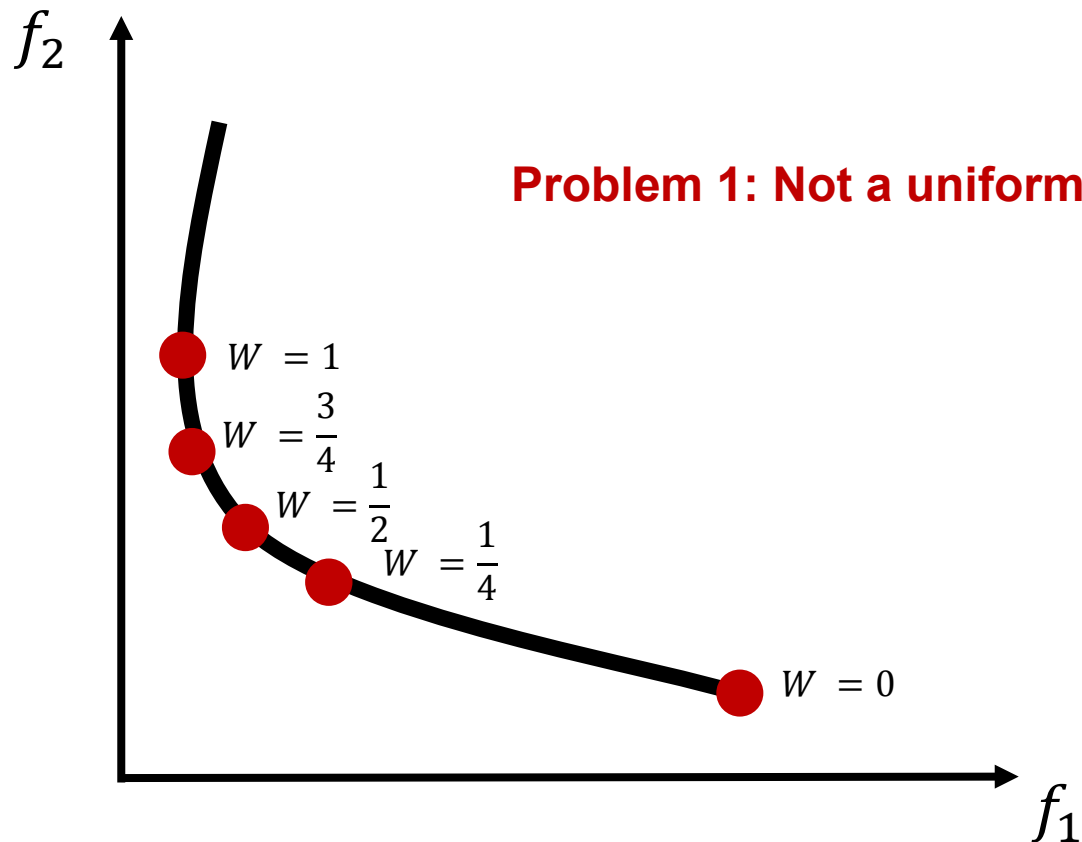
Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

$$w = \frac{3}{4}$$



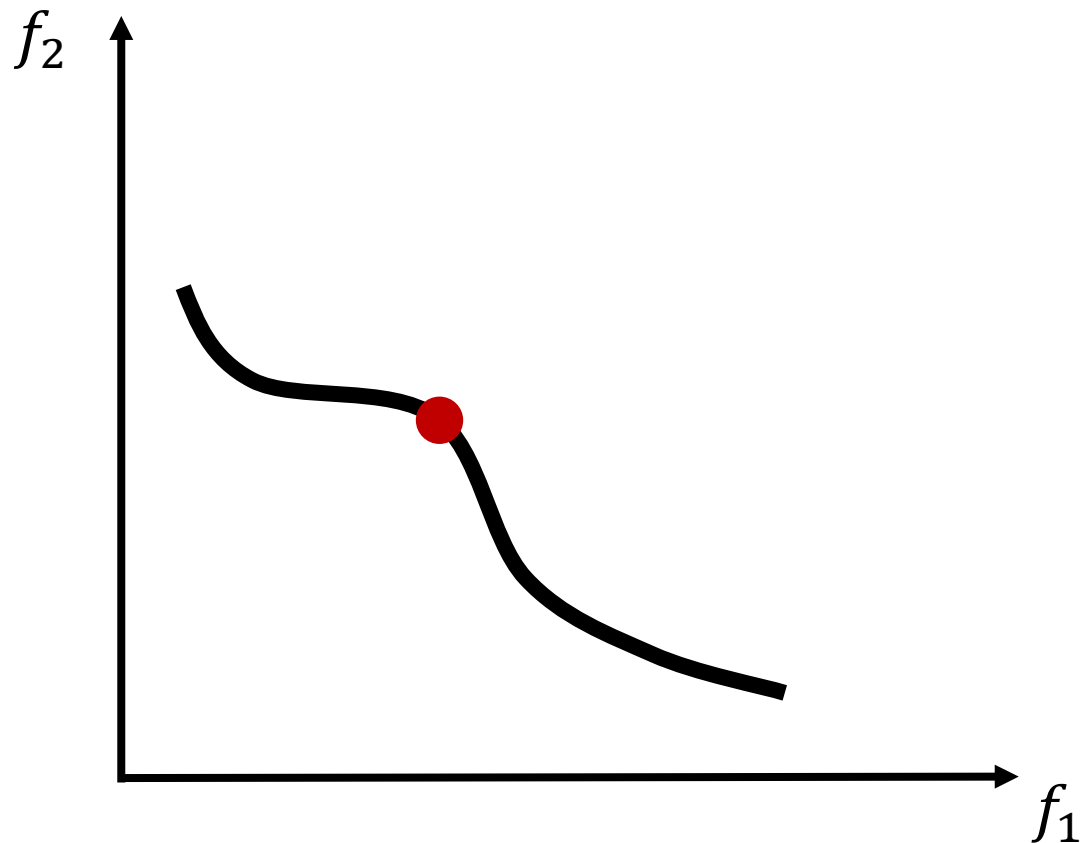
Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$



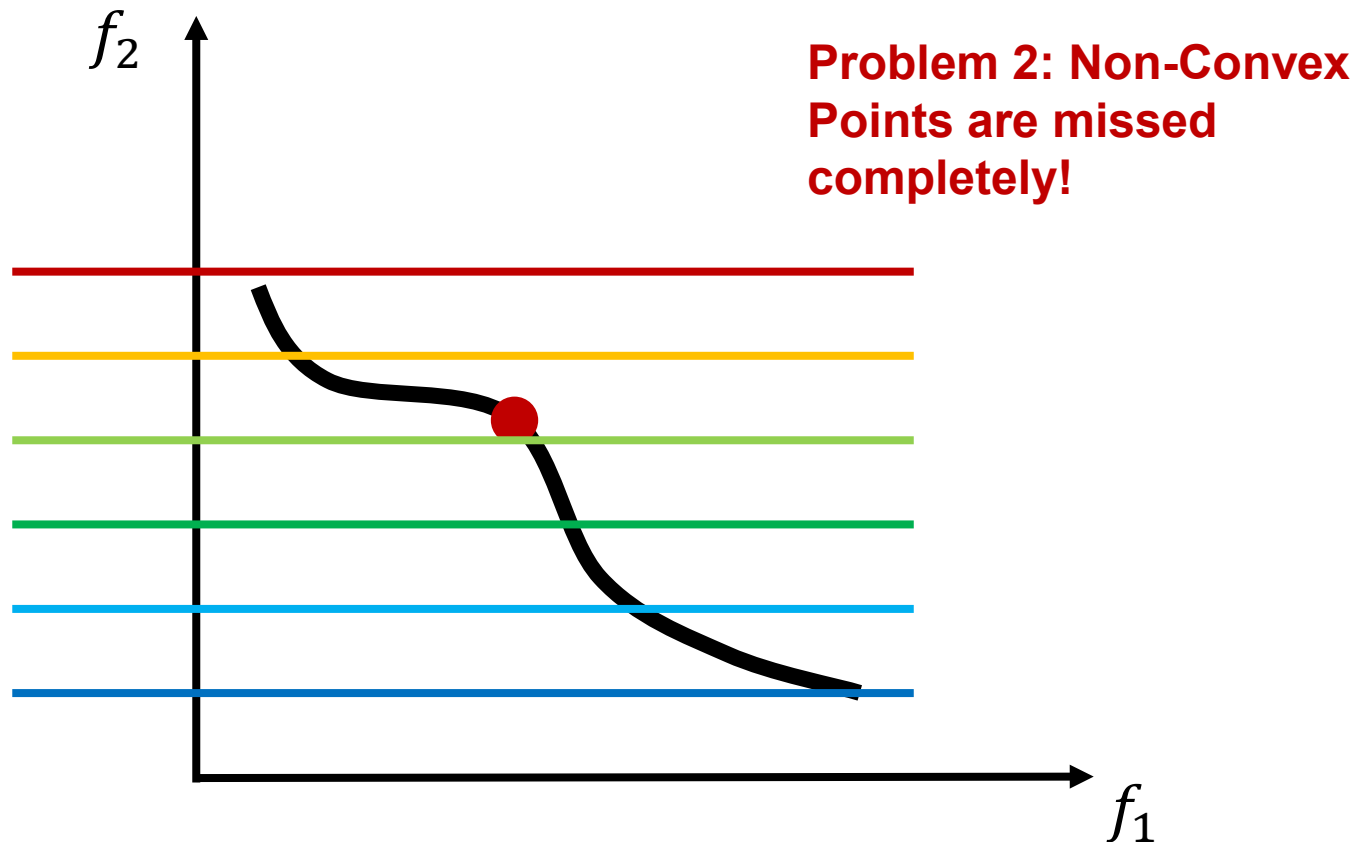
Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

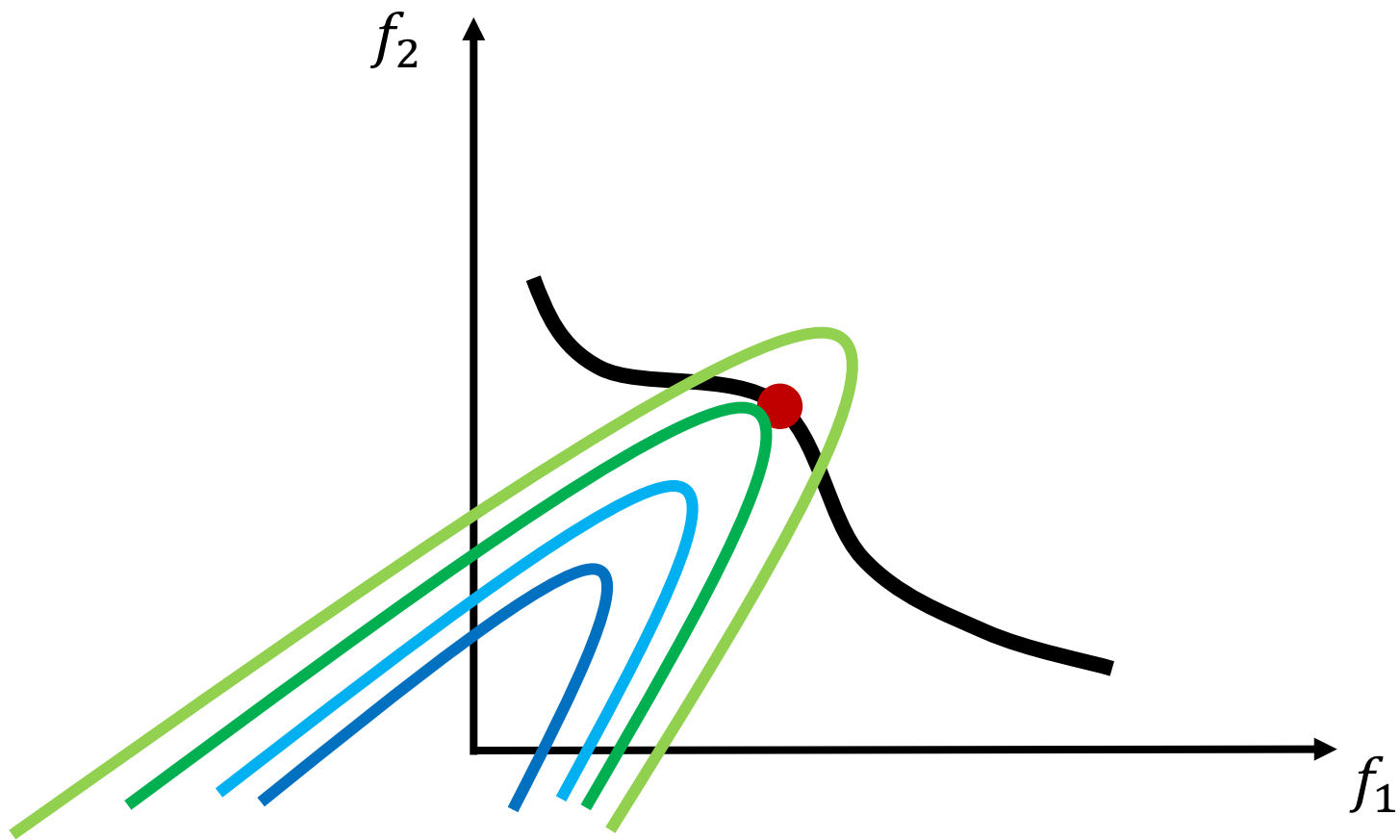


Weighted Sum Method

Vary w and solve $f(x) = wf_1 + (1 - w)f_2$

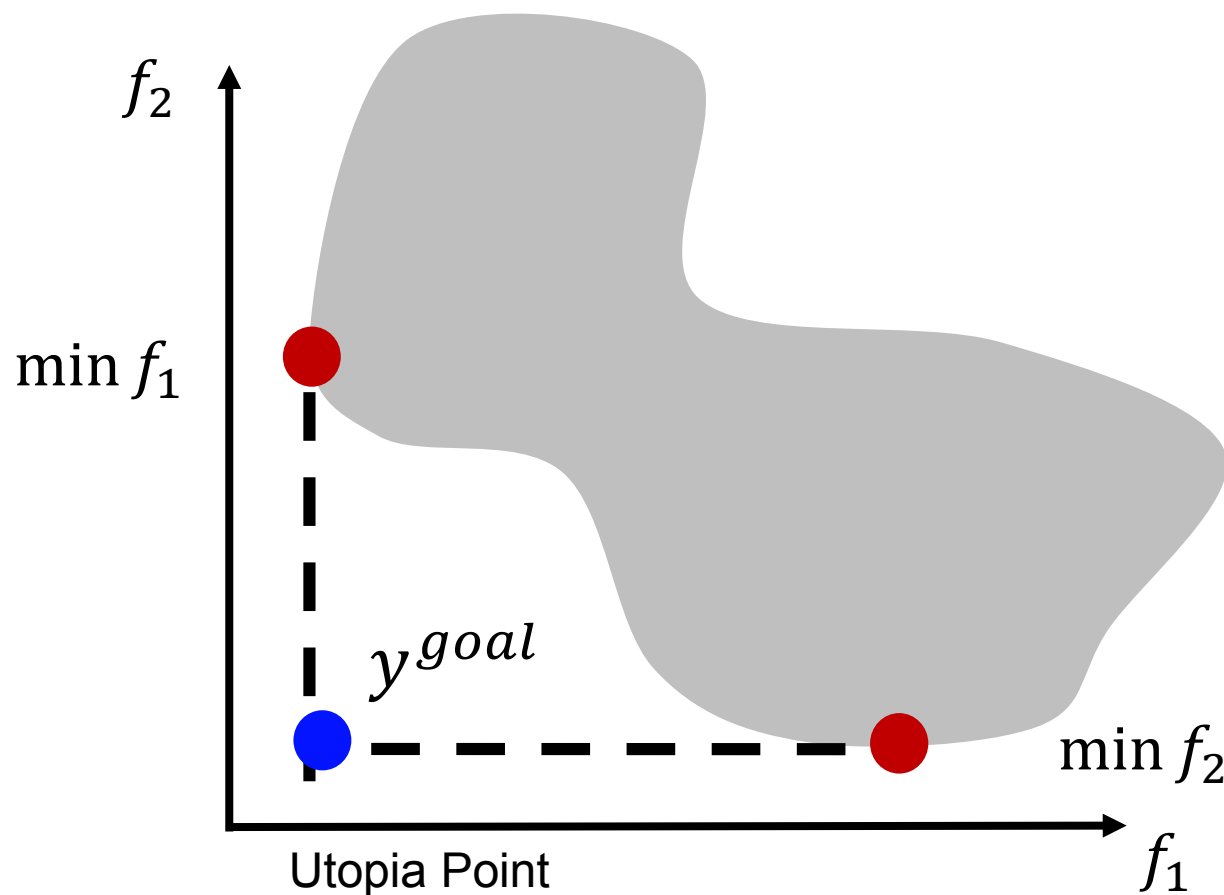


Solution: Non-linear Scalarization Functions



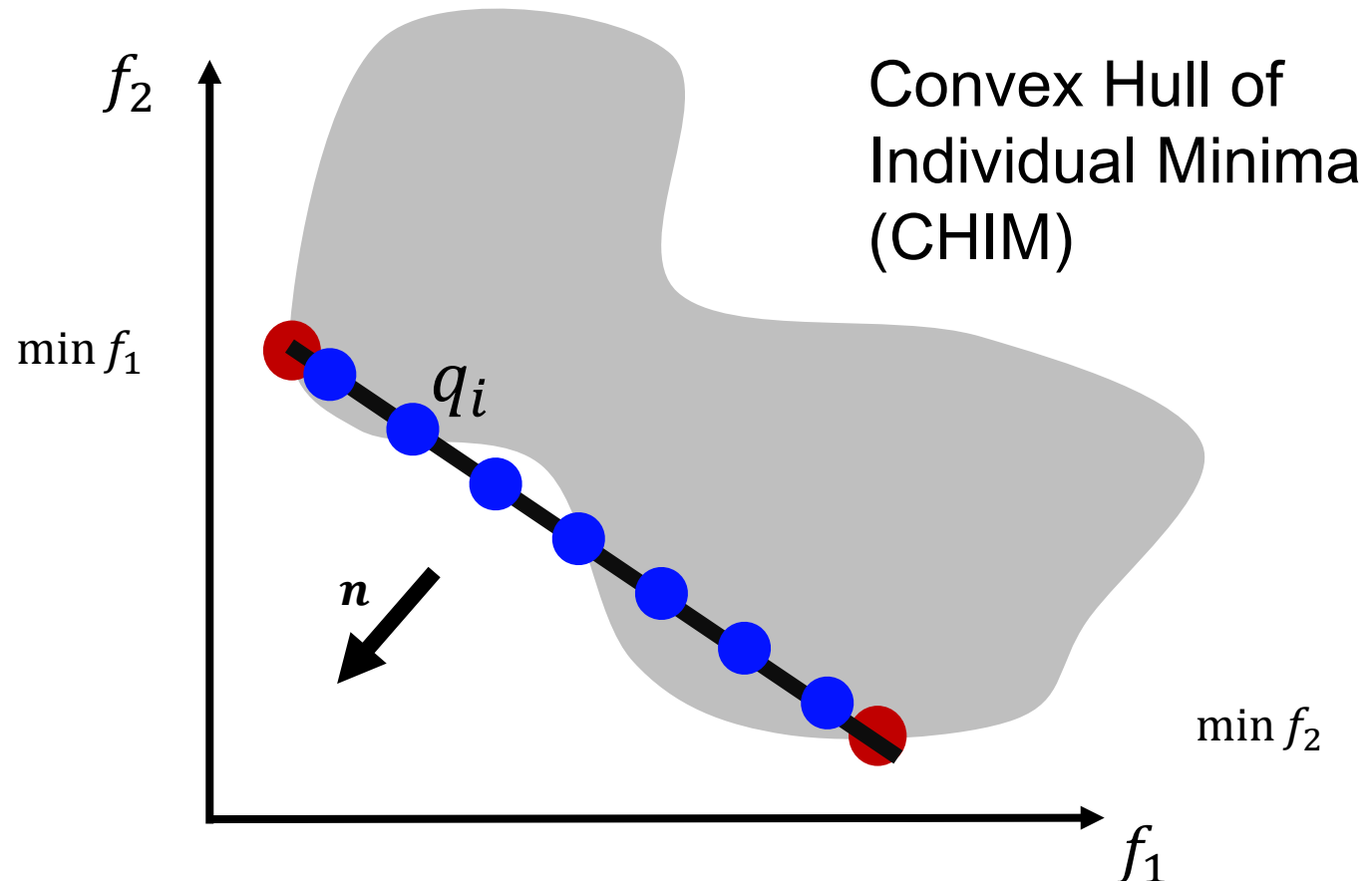
Weighted Exponential Method

$$f(x) = \sum w_i (f_i(x) - y_i^{goal})^p$$



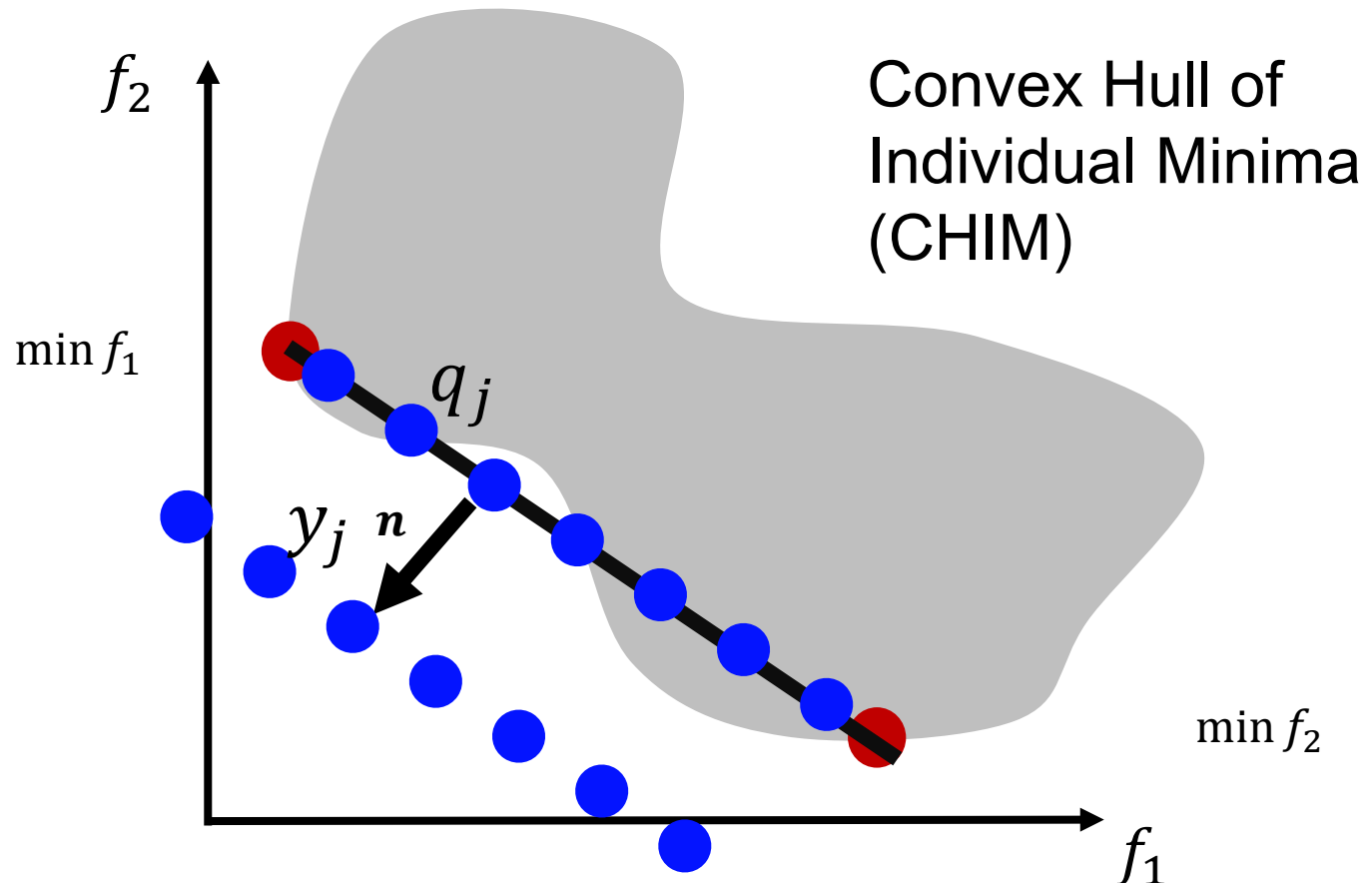
Normal Boundary Intersection

$$\max_{x,t} \quad \text{Additional Constraint : } q_i + t\mathbf{n} \geq F(x)$$

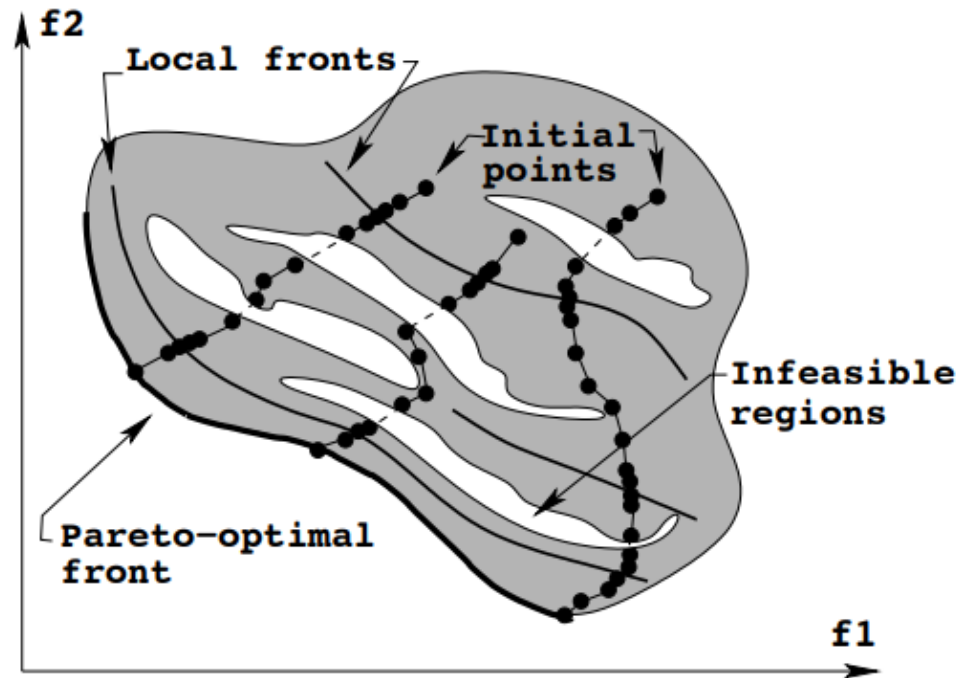


Normal Boundary Intersection (Variation)

Weighted Exponential Method $f(x) = \sum w_i (f_i(x) - y_j)^p$



Problem: Each Single Objective Optimization is not SIMPLE!



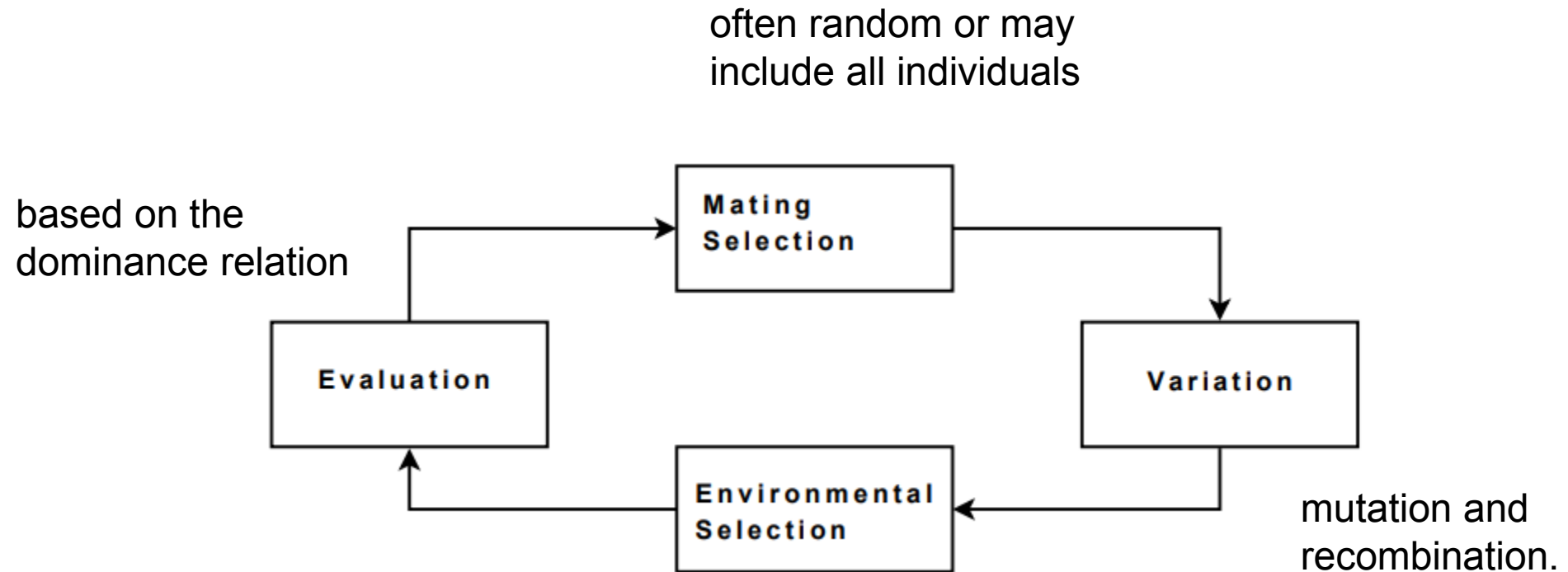
Local Minima: stochastic methods (e.g., MCMC)

Bottleneck: run the single objective optimization many times!

Population Methods

Move many points in parallel towards the front

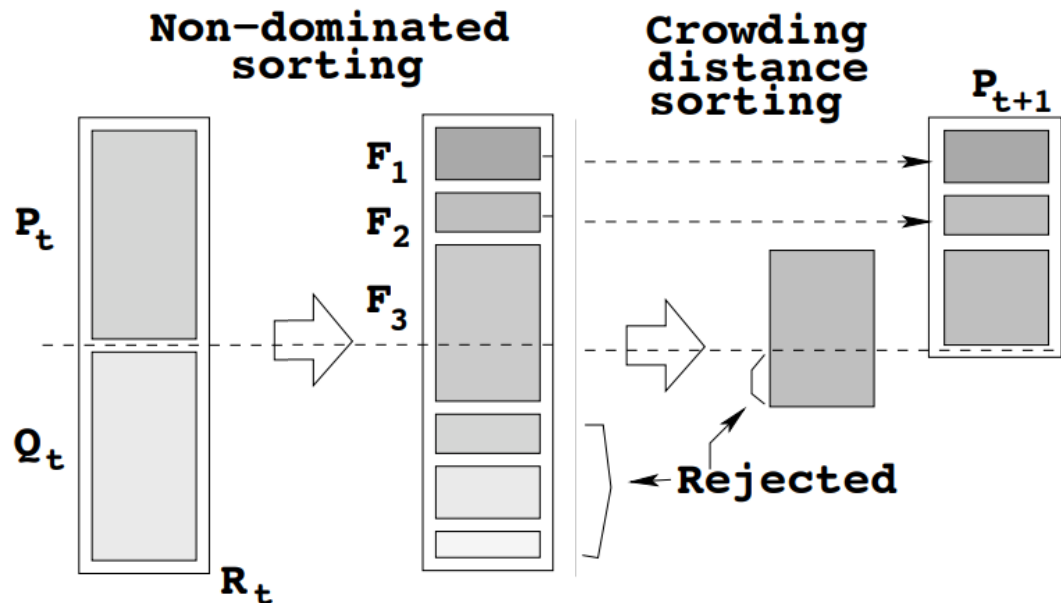
Evolutionary Algorithms





Elitist Non-dominated Sorting GA or NSGA-II

1. Combine parent and offspring populations and create $R_t = P_t \cup Q_t$.
2. Perform no-dominated sorting:
 - F1. Nondominated individuals in the population.
 - F2. Nondominated individuals except those in F1.
 - F3. Nondominated individuals except those in Levels F1 or F2.
3. Populate P_{t+1} : first all F_i 's then, the part of the last on based on diversity
4. Create Q_{i+1}



Creating Offspring's

Cross-over, Mutations and Selection

Every solution has two attributes:

1. a non-domination rank in the population
2. a local crowding distance d_i in the population (important for diversity)

Tournament Selection:

1. if solution i has a better rank
2. if they have the same rank but solution i has a better crowding distance

Elitist Non-dominated Sorting GA or NSGA-II

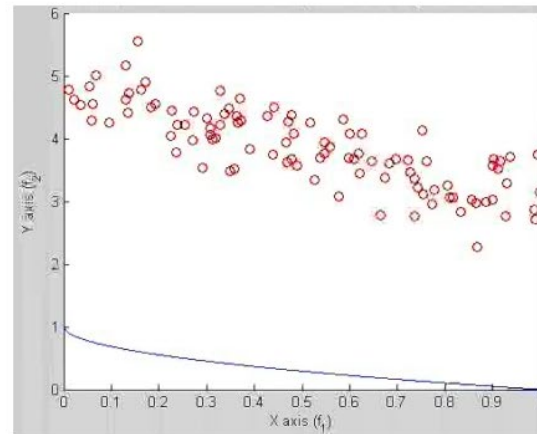


Figure 10: Initial population.

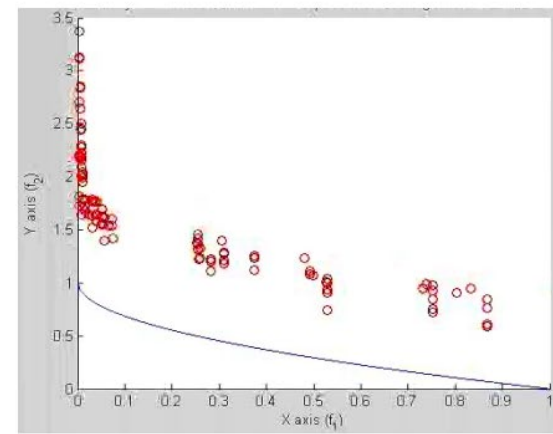


Figure 11: Population at generation 10.

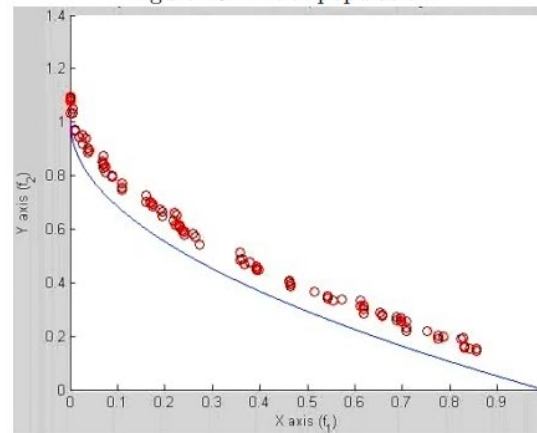


Figure 12: Population at generation 30.

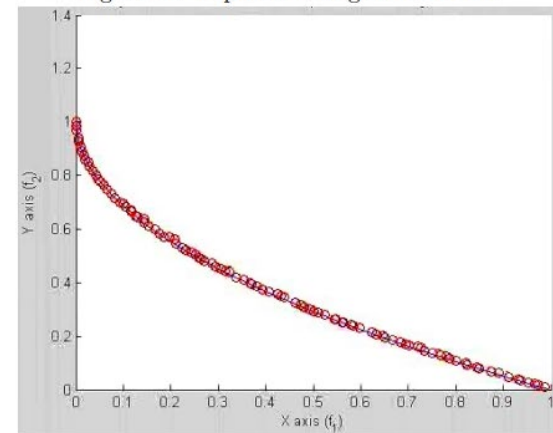


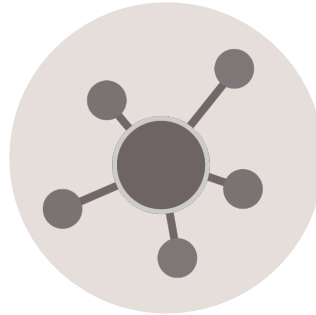
Figure 13: Population at generation 100.

Applications

Design Tools



interactive



multiple domains



intelligent

Today



interactive



multiple domains

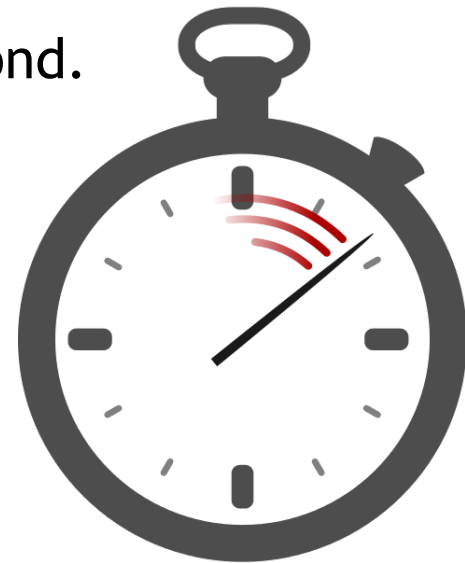


intelligent



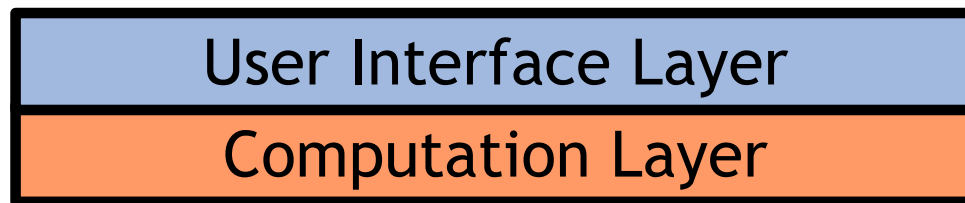
Interactive Applications

- What does it mean for the app to be interactive?
 - In general, the update should be at least a few frames per second.
- What does it mean for the app to be real-time?
 - The update should be at least 30 frames per second.





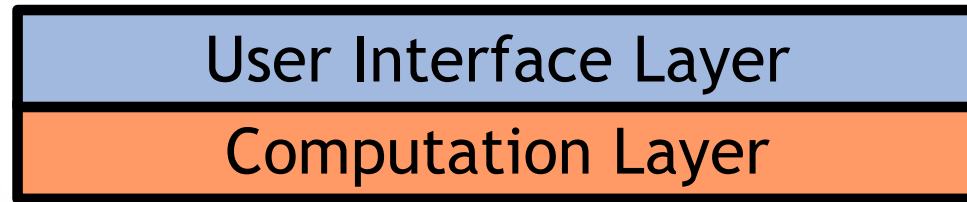
Building High-performance Interactive Apps



Multi-core processors
GPUs

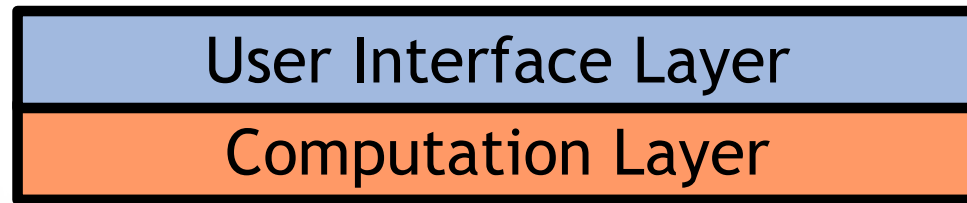


Building High-performance Interactive Apps



Smart Phone

Building High-performance Interactive Applications



Smart Phone

Problem: Limited
computation



Building High-performance Interactive Apps



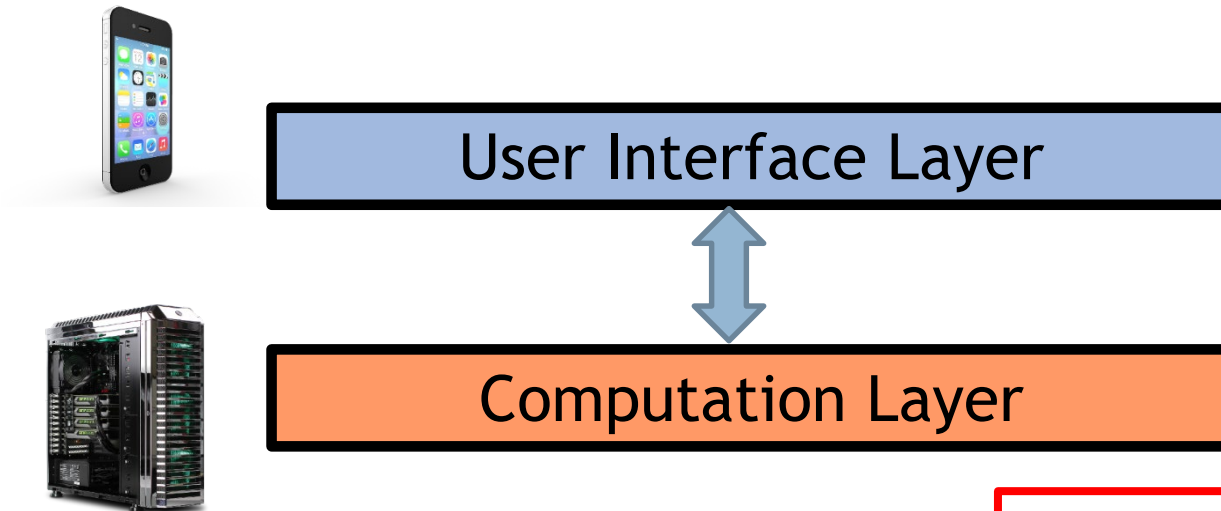
User Interface Layer



Computation Layer



Building High-performance Interactive Apps

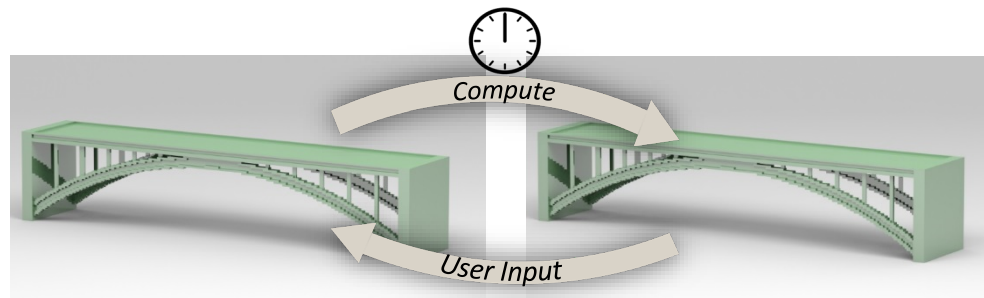
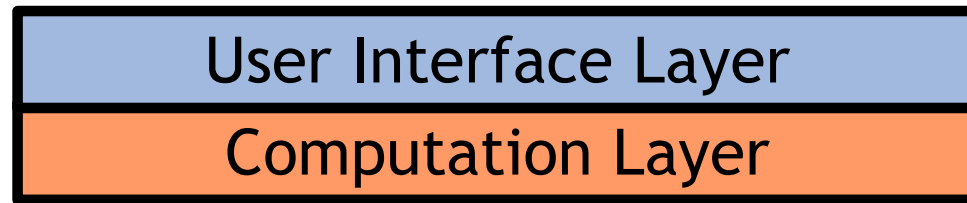


Issues:

- Communication delay
- Splitting computation



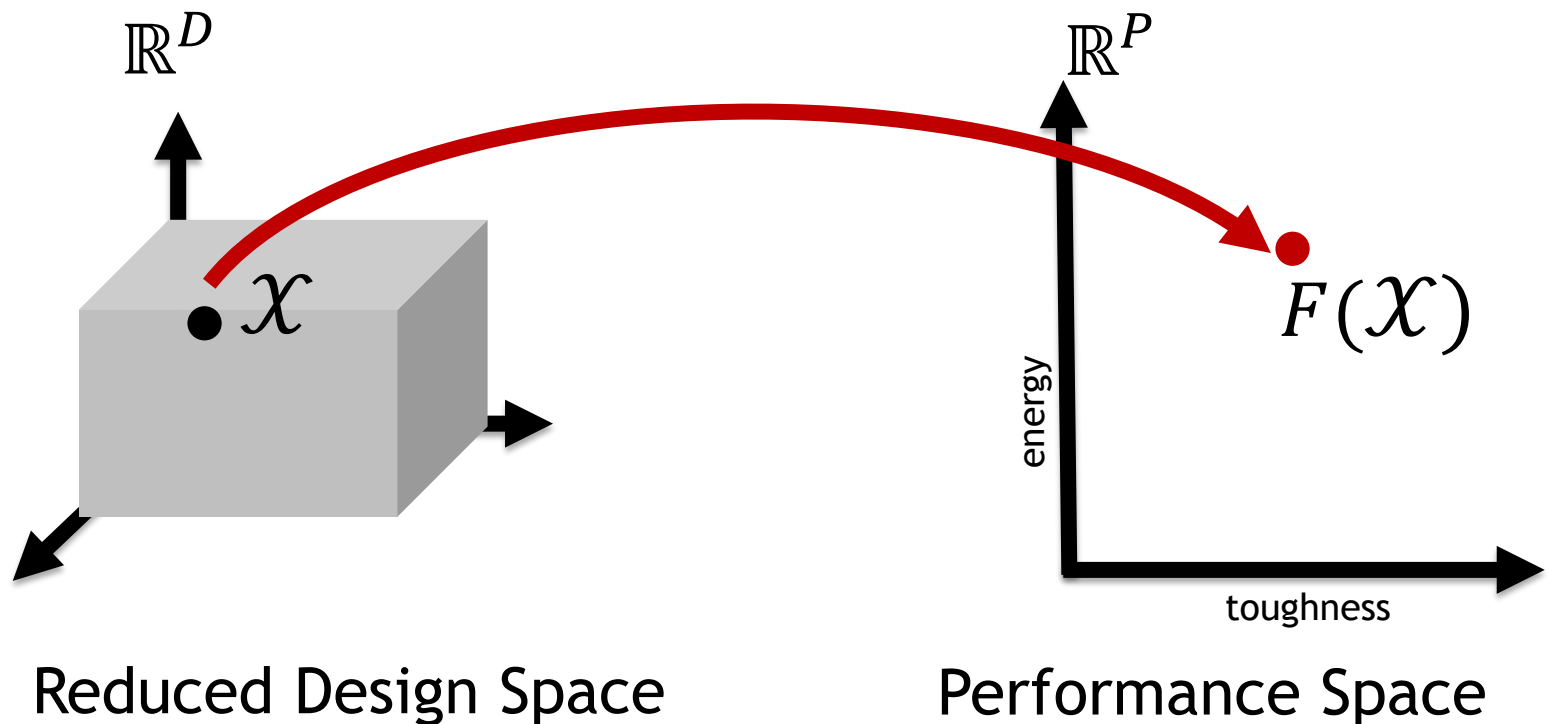
Some Problems Cannot be Computed Interactively





From Design Space to Performance Space

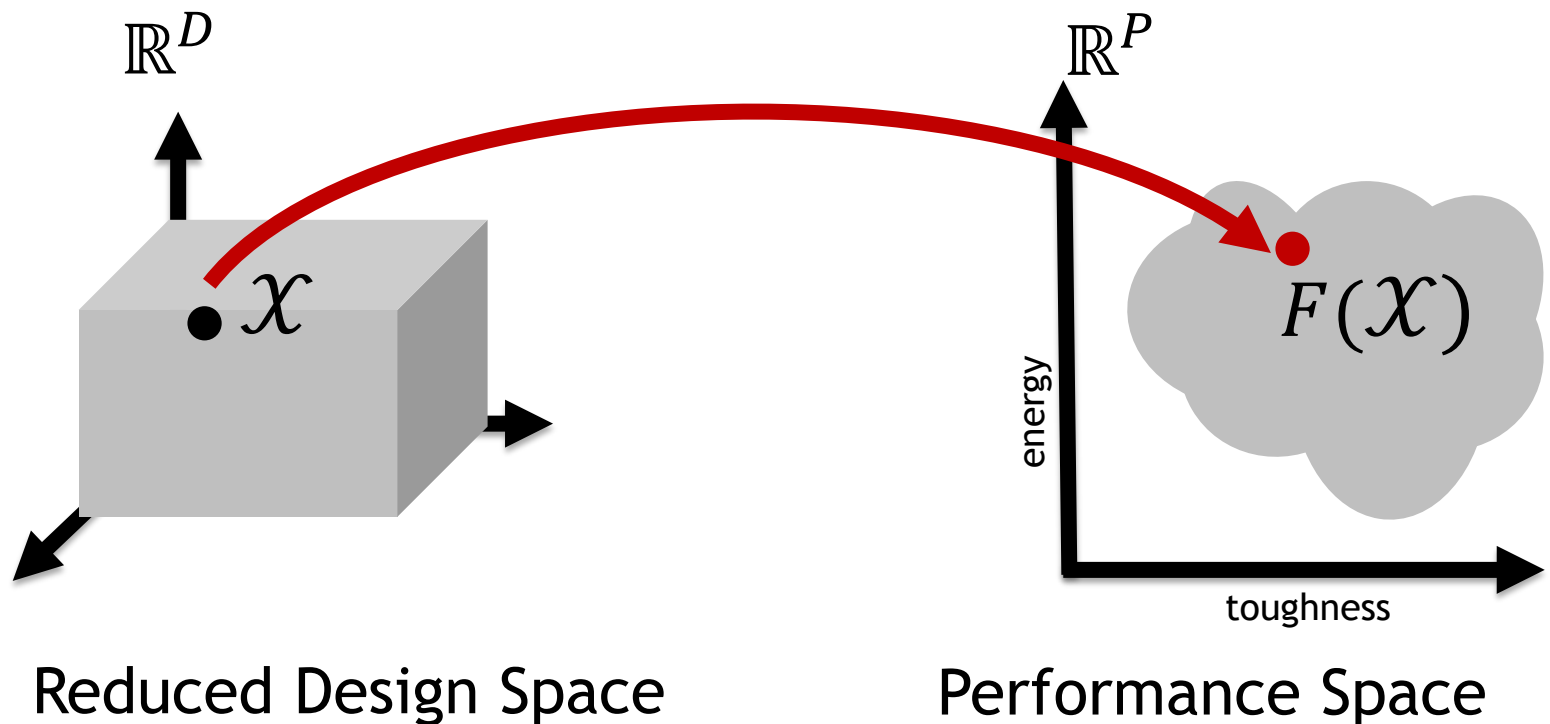
- Numerical simulation maps points from design space to performance space





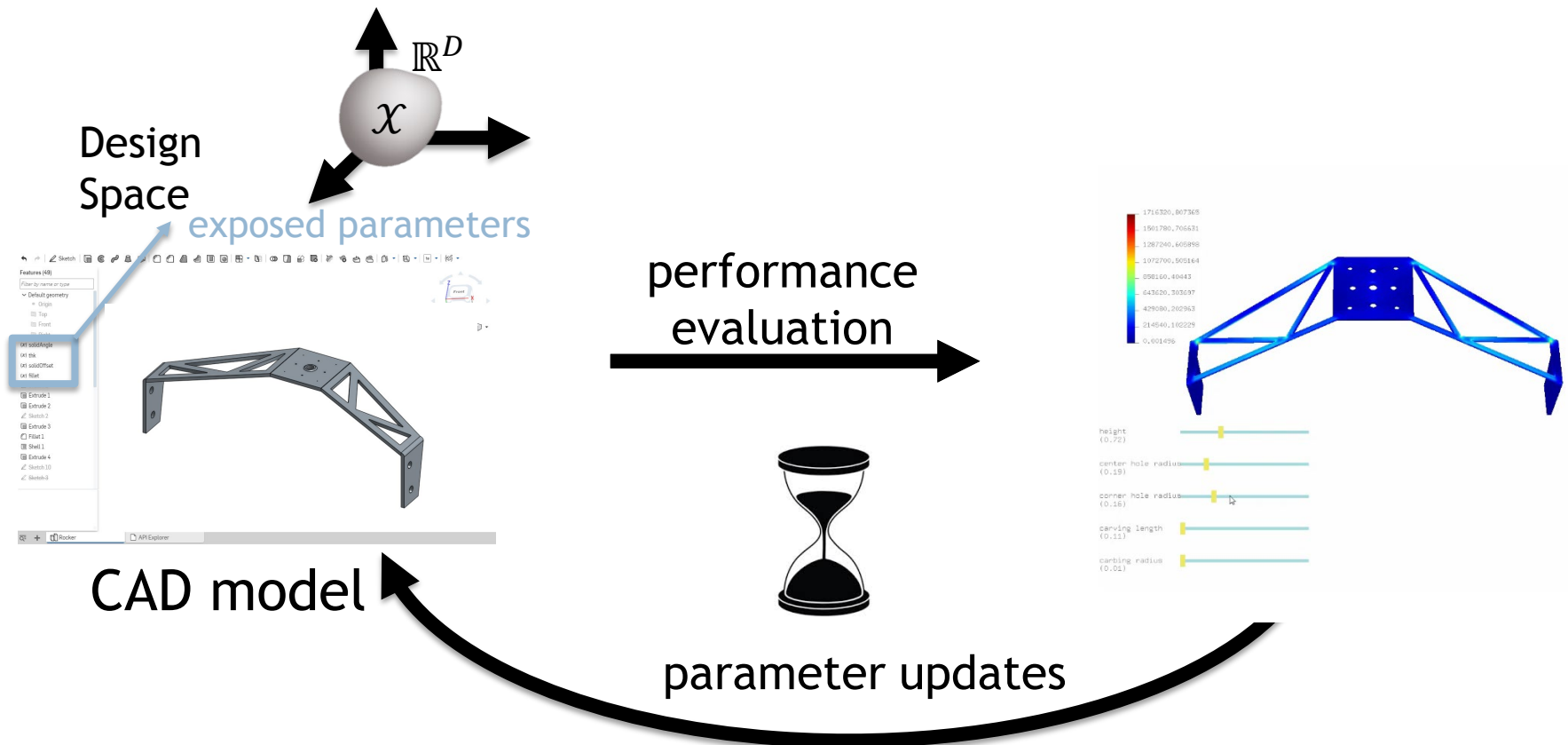
Mapping the Whole Performance Space

- Numerical simulation maps points from design space to performance space





Design Space Exploration for Engineers



Exploration in Comsol Multiphysics



How to Develop an Interactive Solution?

How to Develop an Interactive Solution?

Solution 1: Precomputation + Caching

Solution 2: Precomputation + Interpolation

Solution 3: Sensitivity Analysis

How to Develop an Interactive Solution?

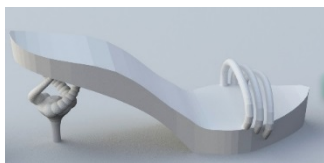
Solution 1: Precomputation + Caching

Solution 2: Precomputation + Interpolation

Solution 3: Sensitivity Analysis



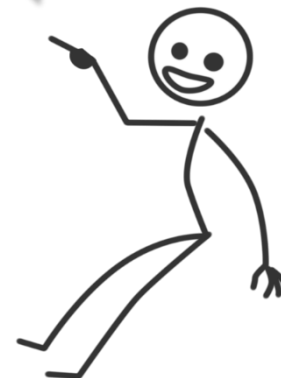
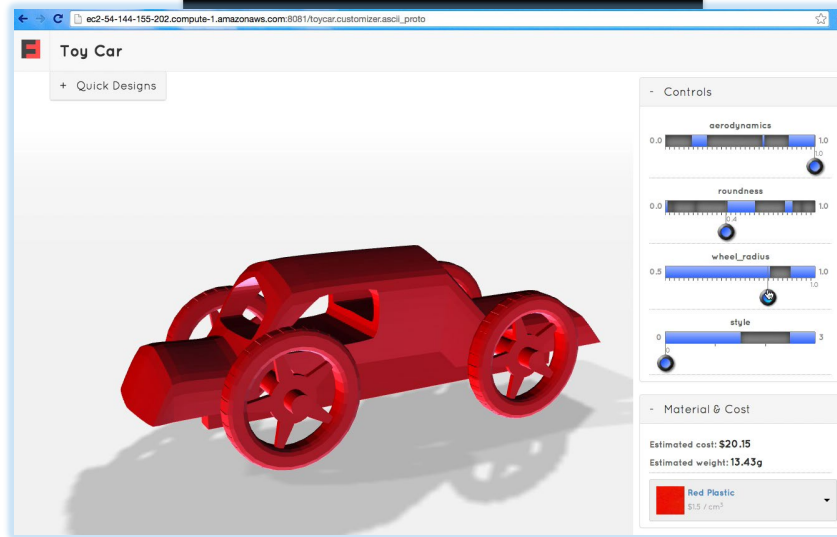
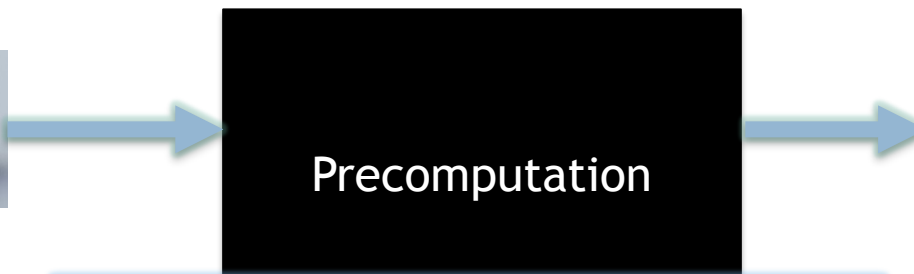
Overview



Parametric
Design



Designer



Novice

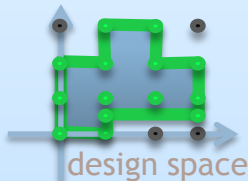


Fab Form Implementation

Fab Form Implementation



Parametric Design



Valid Regions of
the Design Space



Geometry Cache

Fab Form Requirements



1. Customizable



2. Valid



3. Interactive

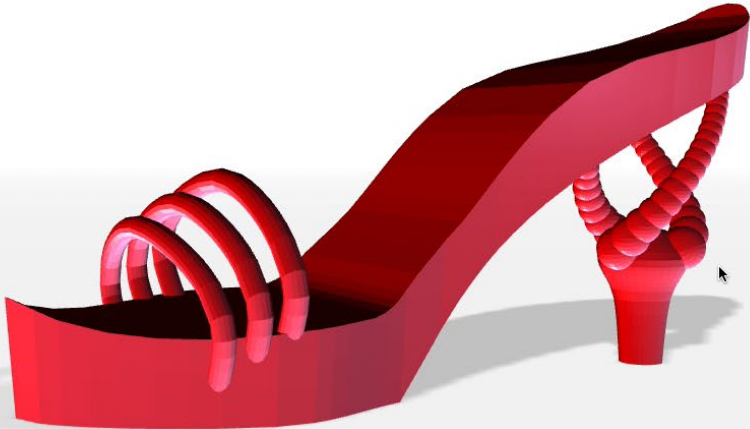


Recording of an Interactive Session

ec2-54-144-155-202.compute-1.amazonaws.com:8080/sandal.customizer.ascil_prot

Platform Sandal

+ Quick Designs



Controls

foot_length: 22.0 to 25.1 (current: 23.2)

heel_height: 2.0 to 10.0 (current: 6.4)

heel_width: 0.1 to 3.0 (current: 1.2)

Material & Cost

Estimated cost: **\$347.64**

Estimated weight: **231.76g**

Red Plastic \$1.5 / cm³

+ Save Model

How to Develop an Interactive Solution?

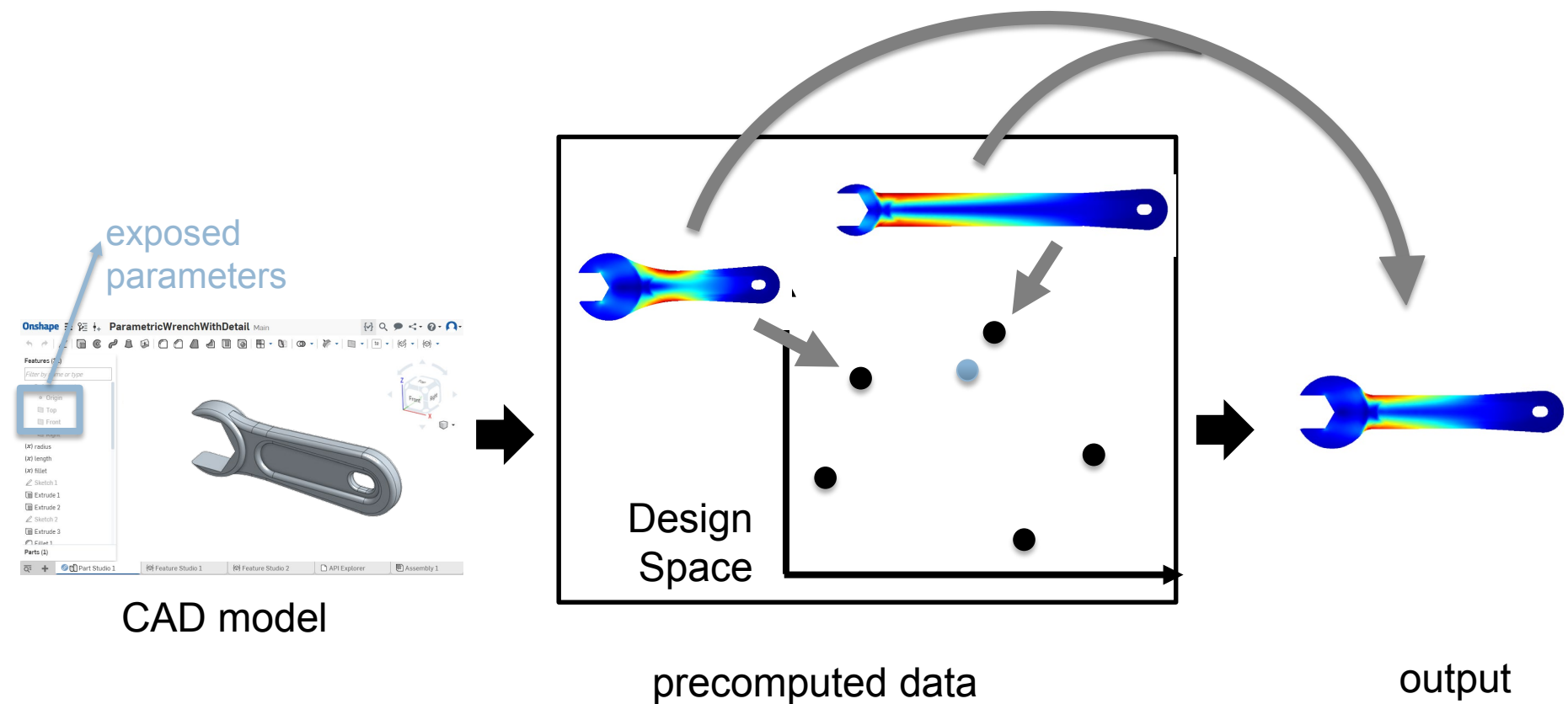
Solution 1: Precomputation + Caching

Solution 2: Precomputation + Interpolation

Solution 3: Sensitivity Analysis



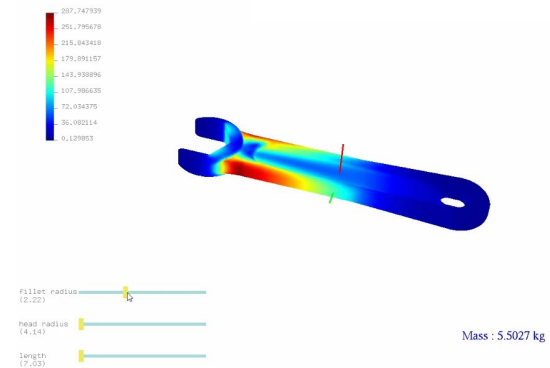
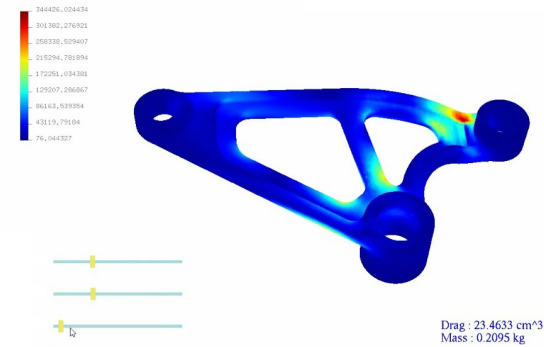
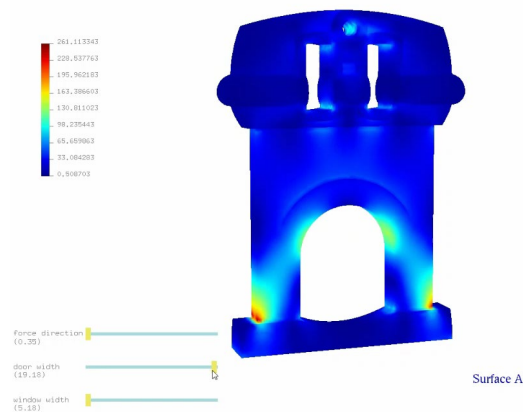
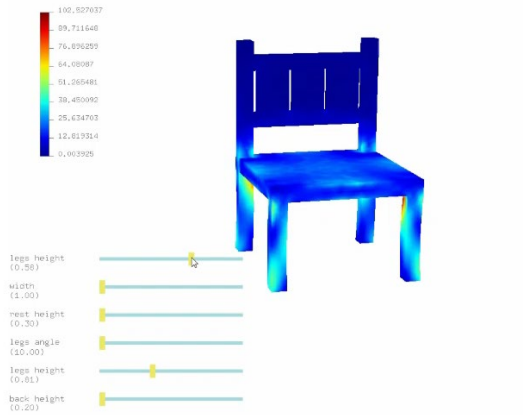
InstantCAD





Results

Stress Distribution



How to Develop an Interactive Solution?

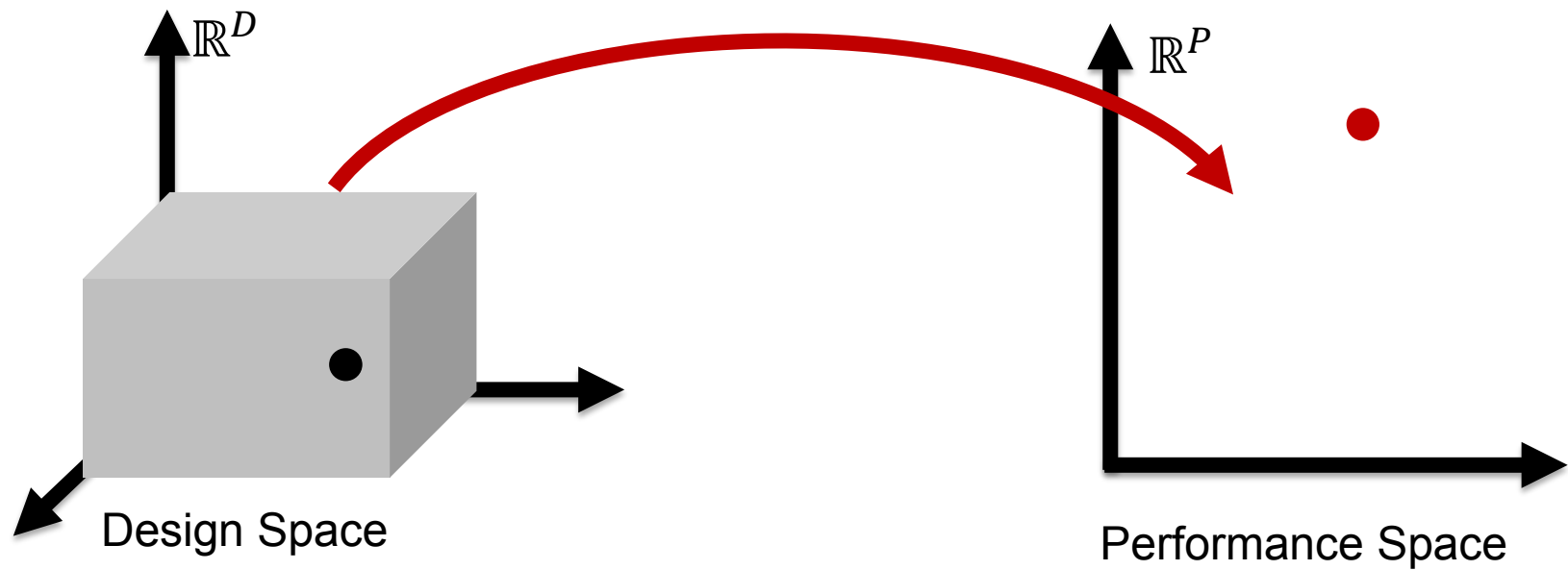
Solution 1: Precomputation + Caching

Solution 2: Precomputation + Interpolation

Solution 3: Sensitivity Analysis

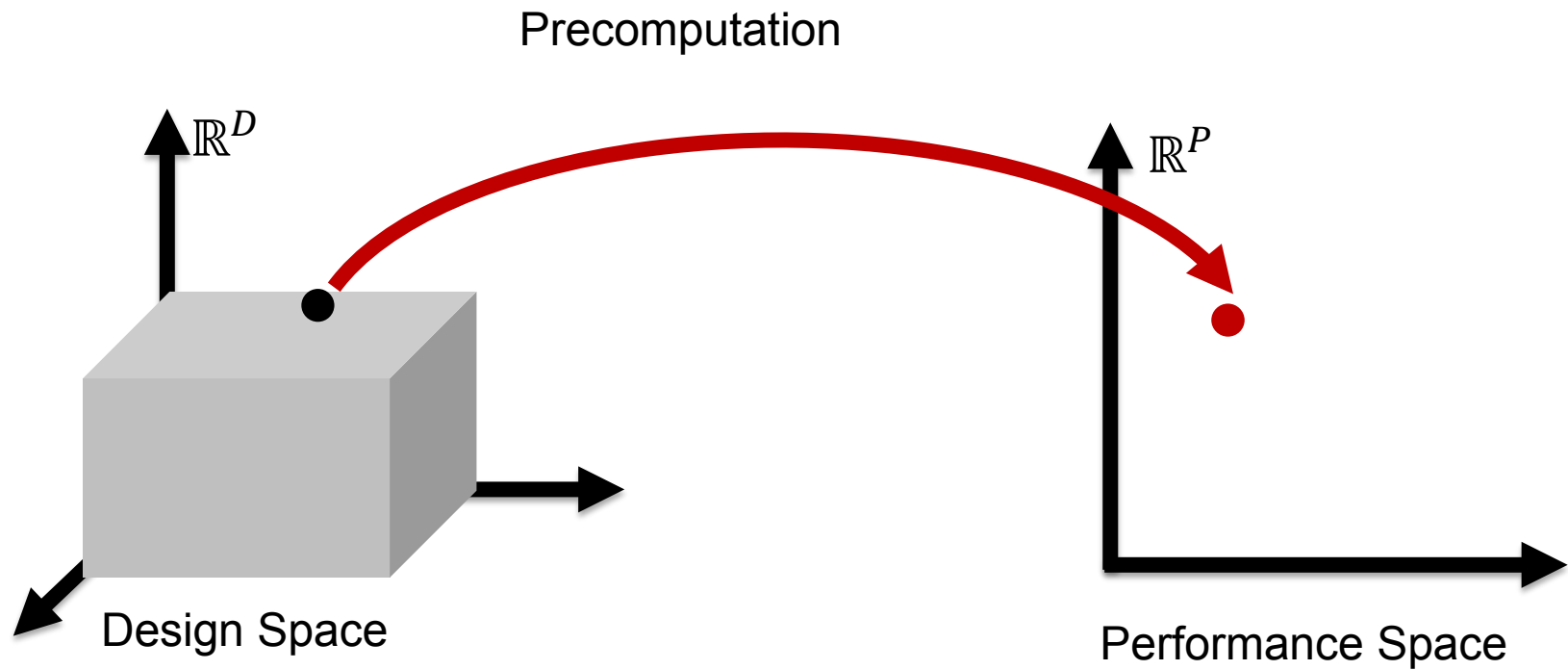


Design Space Exploration



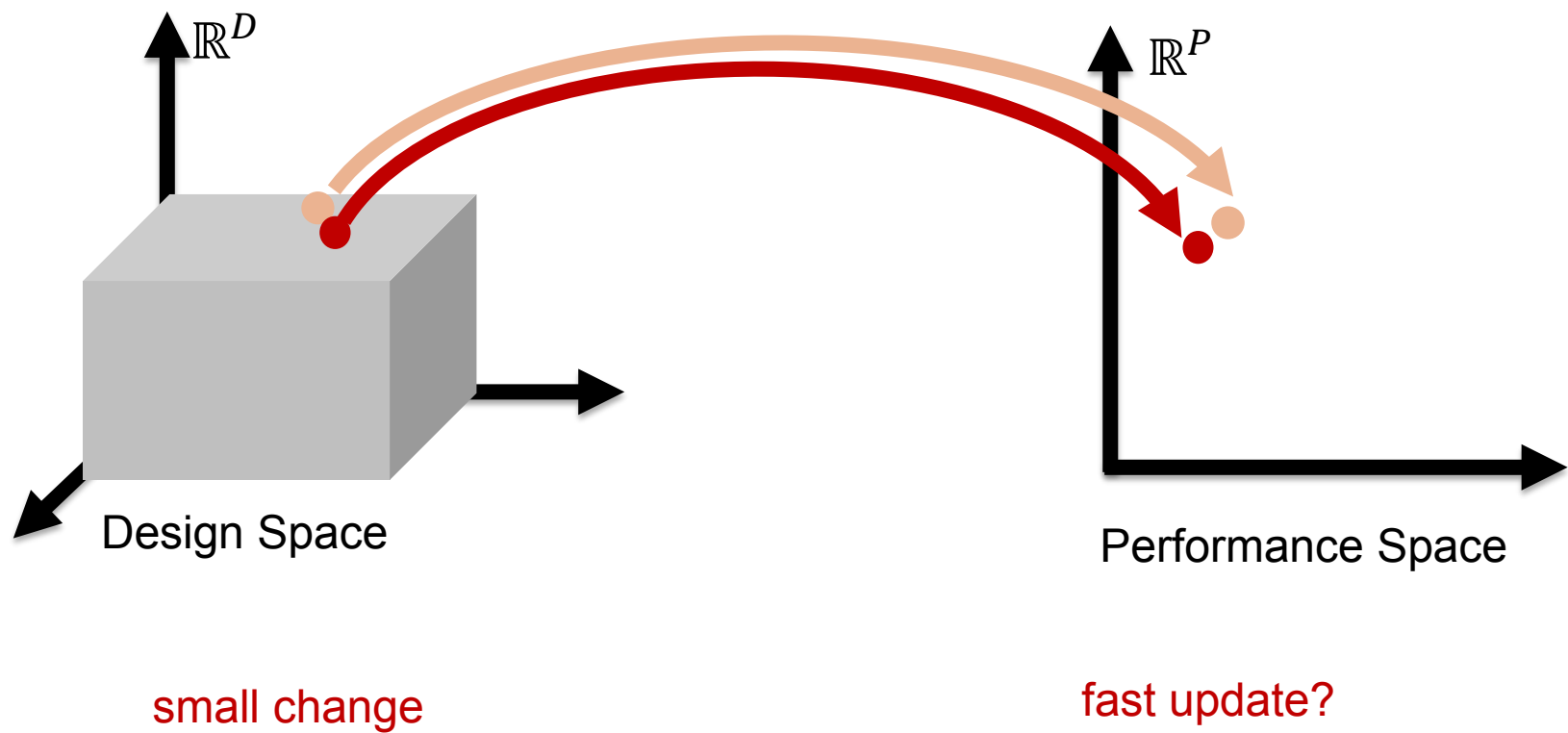


Design Space Exploration

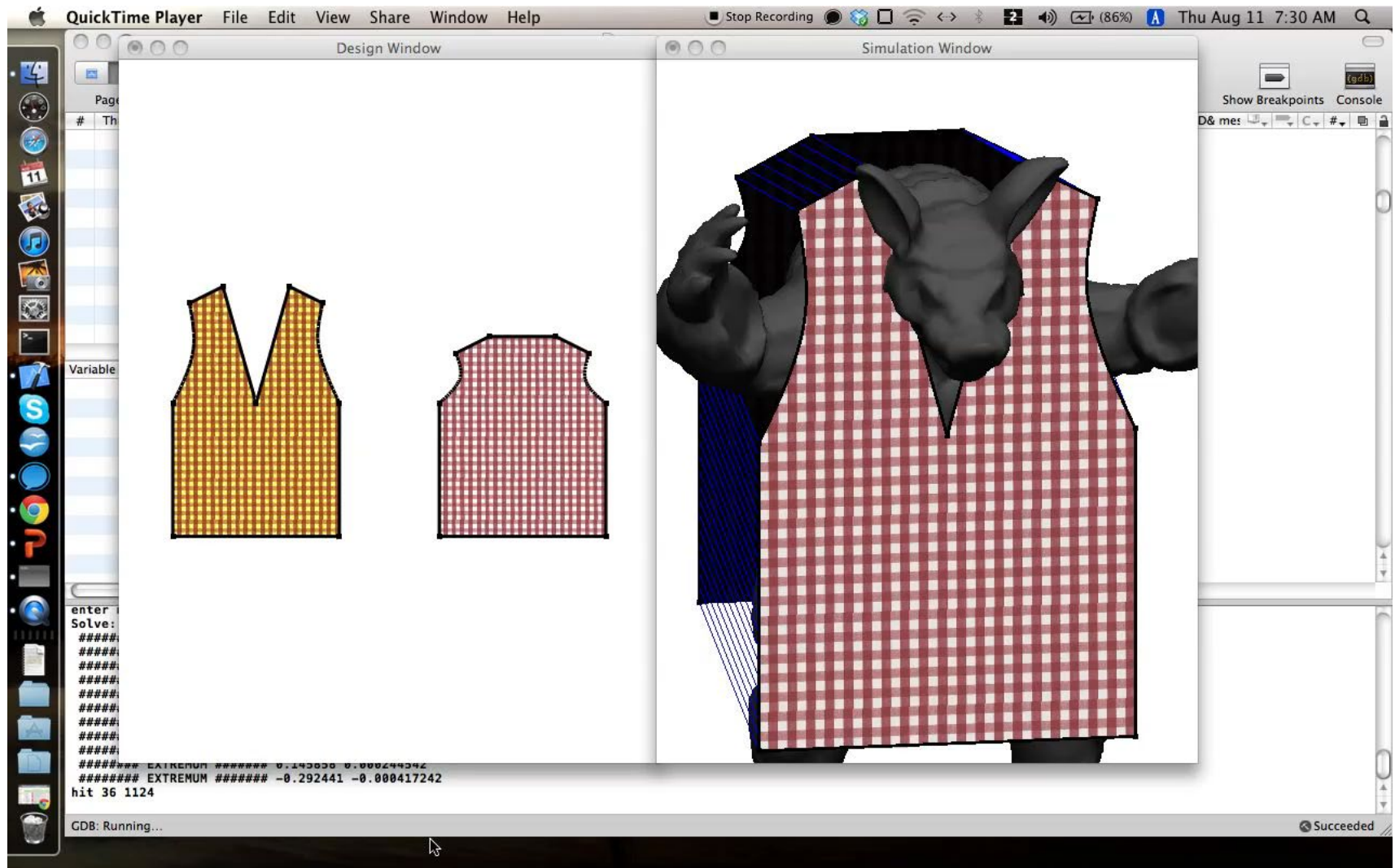




Design Space Exploration

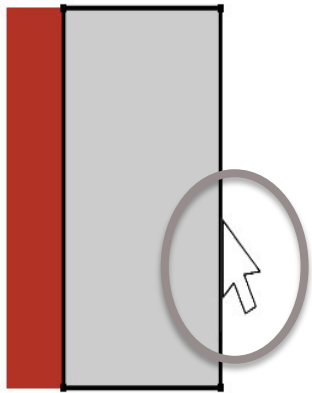


Example: Interactive Garment Design

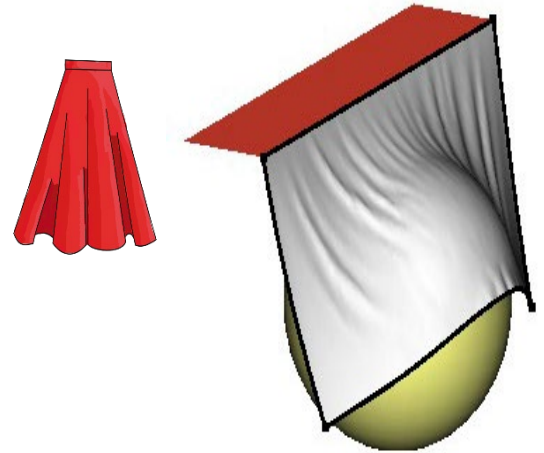




Sensitivity Analysis



Design Change

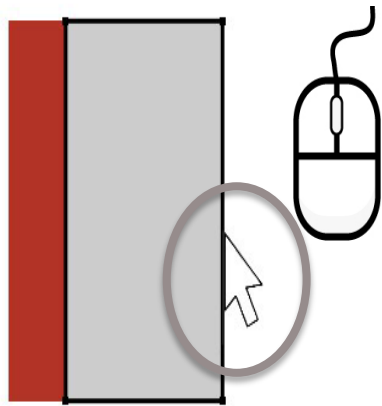


Static cloth shape

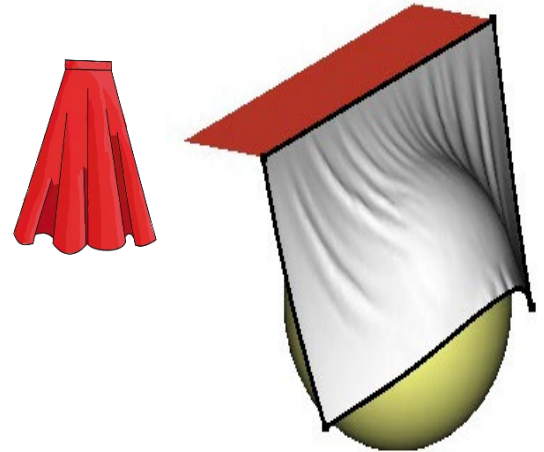
We need to relate changes in the design to changes in cloth shape



Sensitivity Analysis



Mouse position



Static cloth shape

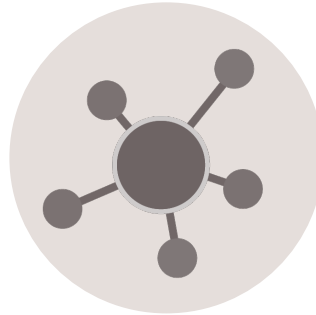
We need to relate changes in the design to changes in cloth shape

Changes in design = changes in mouse position = 2D Space ☺

Today



interactive



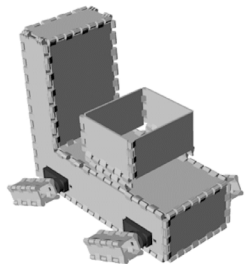
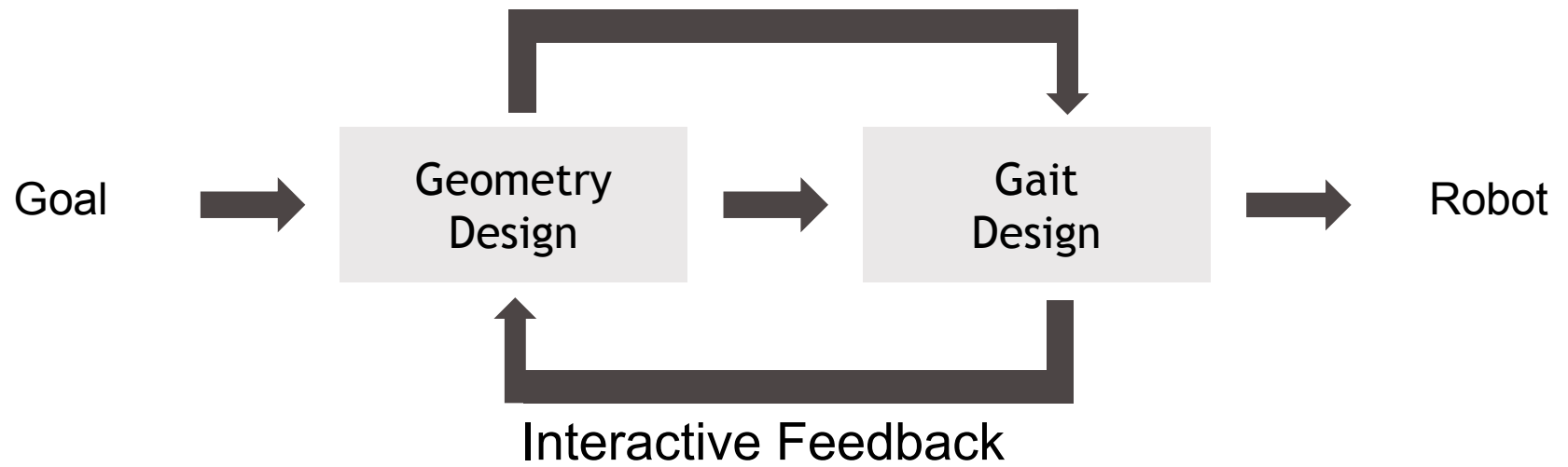
multiple domains



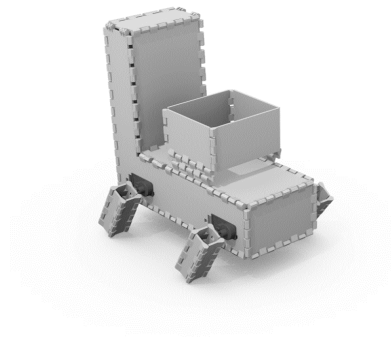
intelligent



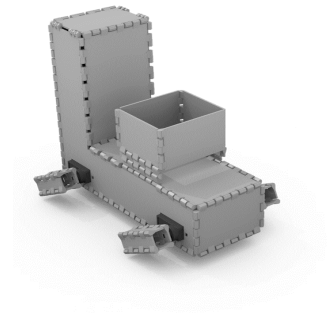
Typical Robot Design Process



Fast Gait

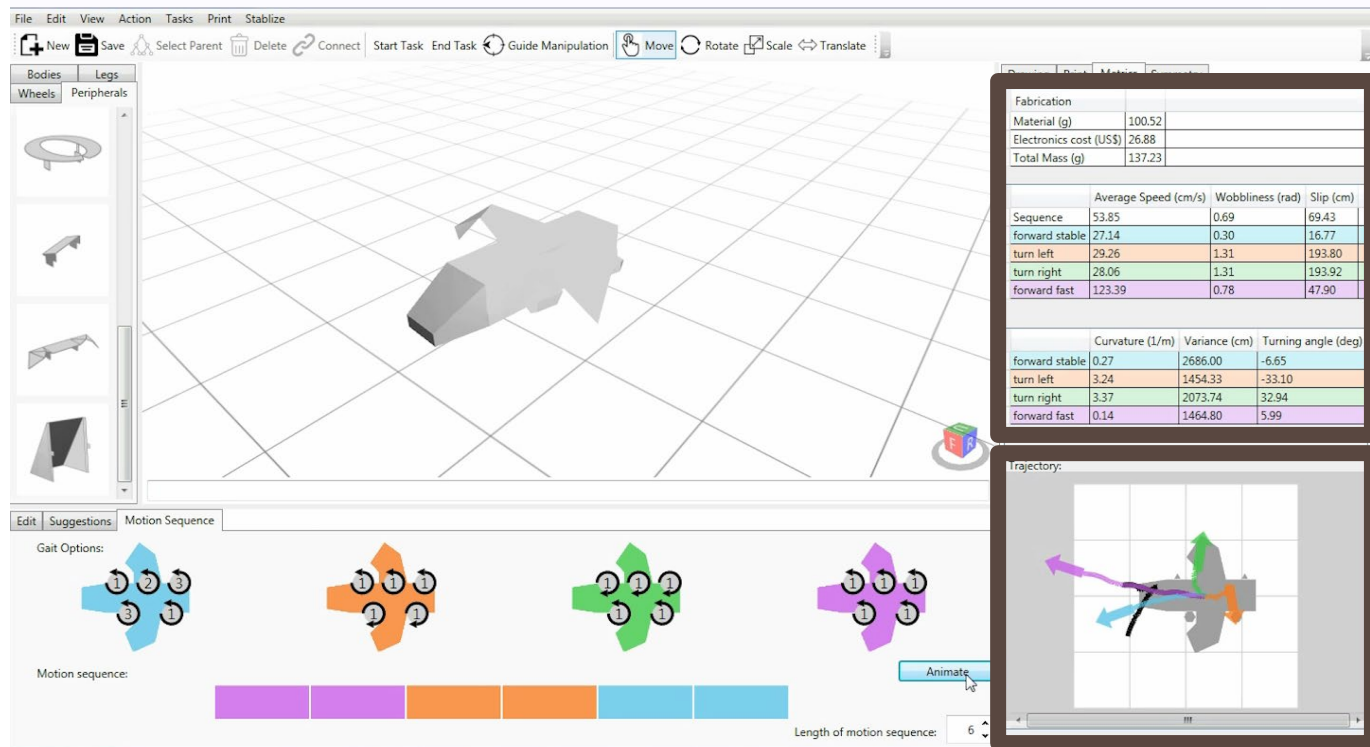


Slower Gait



Geometry Change

Simulation





Real Time Feedback

Fabrication

Material (g)	100.52
Electronics cost (US\$)	26.88
Total Mass (g)	137.23

	Average Speed (cm/s)	Wobbliness (rad)	Slip (cm)
Sequence	53.85	0.69	69.43
forward stable	27.14	0.30	16.77
turn left	29.26	1.31	193.80
turn right	28.06	1.31	193.92
forward fast	123.39	0.78	47.90

	Curvature (1/m)	Variance (cm)	Turning angle (deg)
forward stable	0.27	2686.00	-6.65
turn left	3.24	1454.33	-33.10
turn right	3.37	2073.74	32.94
forward fast	0.14	1464.80	5.99

Motion Sequence

Gait Options:

Motion sequence:

Length of motion sequence: 6

Trajectory:



Design Space Exploration

The screenshot displays a software interface for design space exploration, likely for a robot. The main window shows a 3D model of a robot on a grid. The interface includes a menu bar (File, Edit, View, Action, Tasks, Print, Stabilize), a toolbar with icons for New, Save, Select Parent, Delete, Connect, Start Task, End Task, Guide Manipulation, Move, Rotate, Scale, and Translate. A left sidebar shows a tree view with 'Bodies' and 'Legs' expanded, and a 'Wheels' tab selected. A bottom panel shows a 'Motion Sequence' editor with a 'turn left' task, a 'Desired Curvature' slider, and a sequence of four robot poses with theta values: -30, 60, 30, and 30. A right sidebar shows a 'Metrics' tab with a table of performance metrics.

Metrics Table:

	Average Speed (cm/s)	Wobbliness (rad)	Slip (cm)
Sequence	57.32	0.91	62.40
forward stable	16.55	0.37	23.53
turn left	87.71	1.86	120.21
turn right	2.89	1.66	192.38
forward fast	129.41	1.18	75.99

	Curvature (1/m)	Variance (cm)	Turning angle (deg)
forward stable	0.25	2628.04	3.80
turn left	0.11	4534.72	3.46
turn right	35.71	4581.85	36.07
forward fast	0.29	4064.51	13.09

Trajectory:

Optimization

The screenshot displays a software interface for robot optimization. The main workspace shows a 3D model of a robot on a grid. A sidebar on the left contains a 'Wheels' tab and a 'Peripherals' tab, with a 'Bodies' list showing various components. The top menu bar includes 'File', 'Edit', 'View', 'Action', 'Tasks', 'Print', 'Stabilize', and 'Optimization'. The 'Optimization' menu is open, showing options like 'New', 'Save', 'Select Parent', 'Delete', 'Connect', 'Start Task', 'End Task', 'Guide Manipulation', 'Move', 'Rotate', 'Scale', 'Translate', and 'Optimization'.

On the right, the 'Metrics' tab is active, displaying two tables of performance data:

	Average Speed (cm/s)	Wobbliness (rad)	Slip (cm)
Sequence	57.32	0.91	62.40
forward stable	16.55	0.37	23.53
turn left	87.71	1.86	120.21
turn right	2.89	1.66	192.38
forward fast	129.41	1.18	75.99

	Curvature (1/m)	Variance (cm)	Turning angle (deg)
forward stable	0.25	2628.04	3.80
turn left	0.11	4534.72	3.46
turn right	35.71	4581.84	36.07
forward fast	0.29	4064.51	13.09

Below the tables, the 'Trajectory' section shows a 2D plot of the robot's path. At the bottom, the 'Motion Sequence' tab is active, showing a sequence of robot poses and a 'Desired Curvature' of 0.00. The sequence is named 'forward fast' and includes an 'Update' button and an 'Animate' button. The sequence is visualized with a series of robot poses and a timeline with a 'theta = 60' slider.



Total time: 15:30 s

Total time: 8:40 s

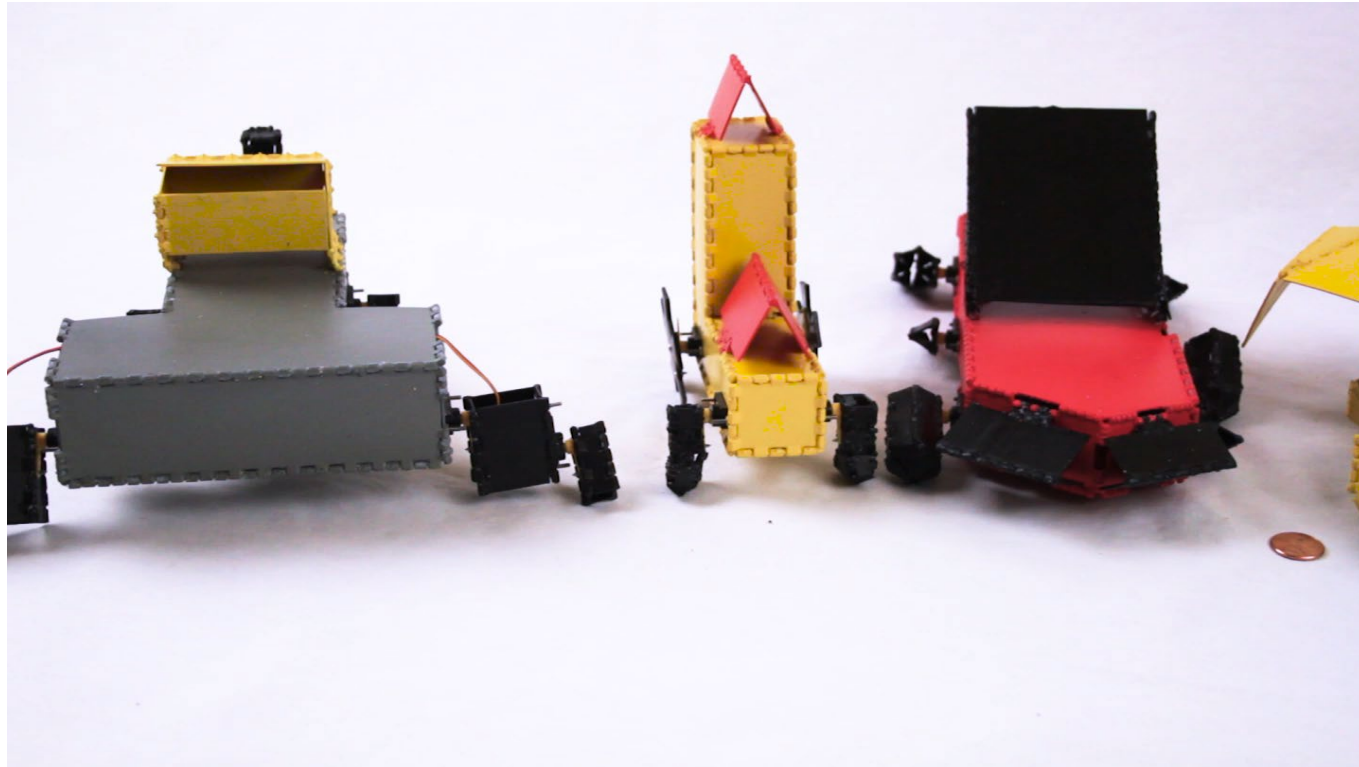


Physical Robots Created



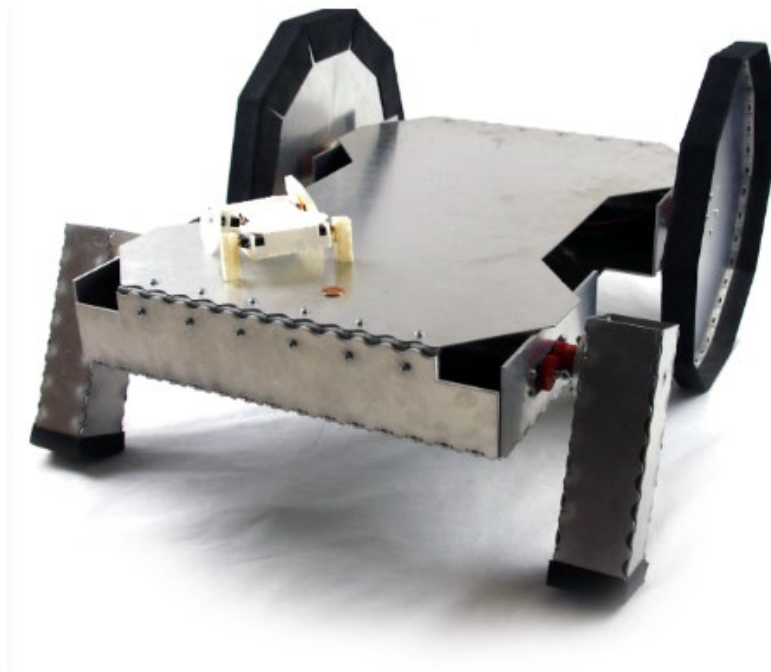


Physical Robots Created



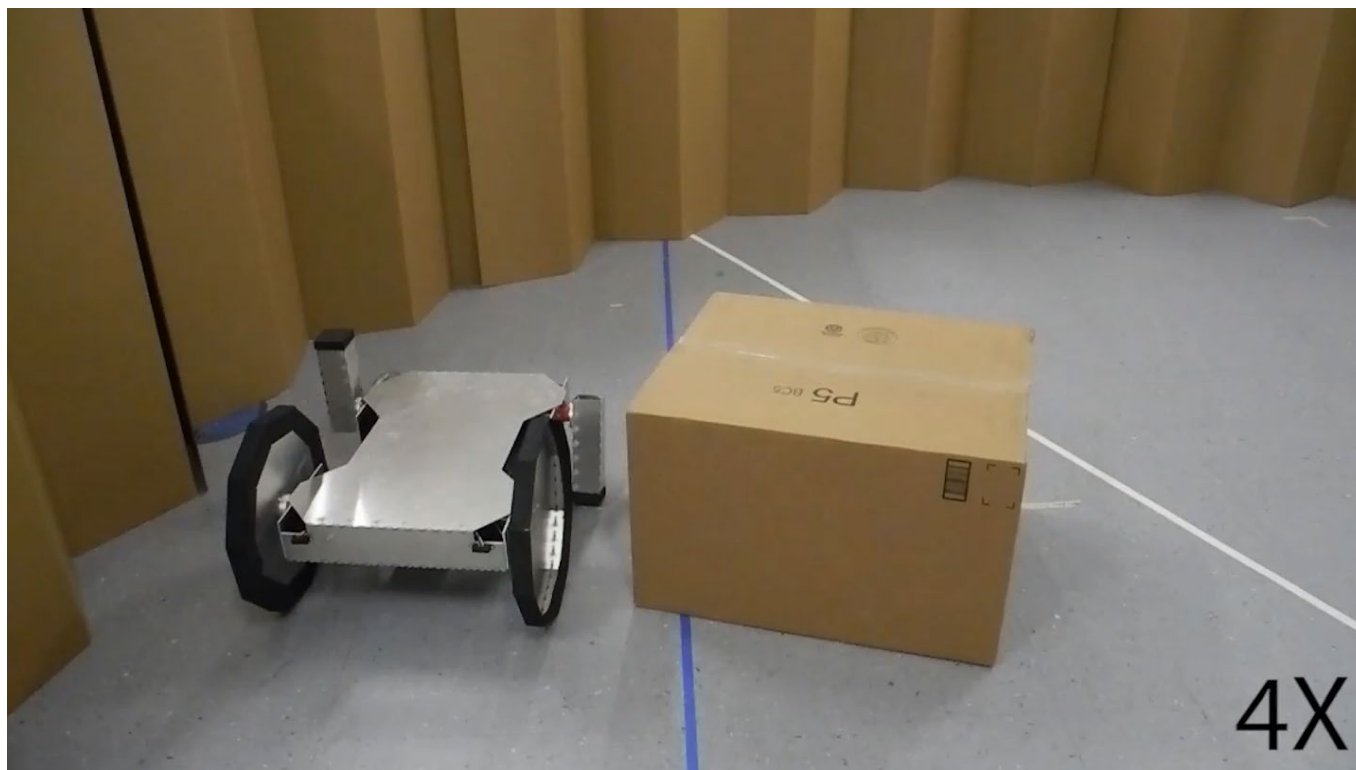


Larger Scale Result

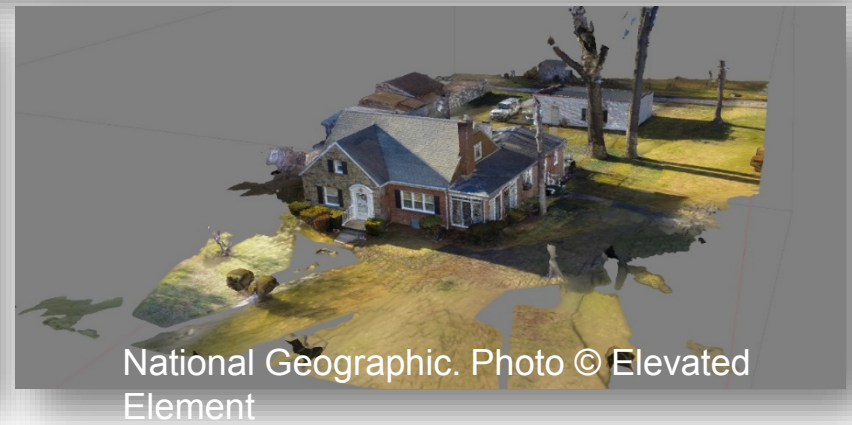




Larger Scale Result

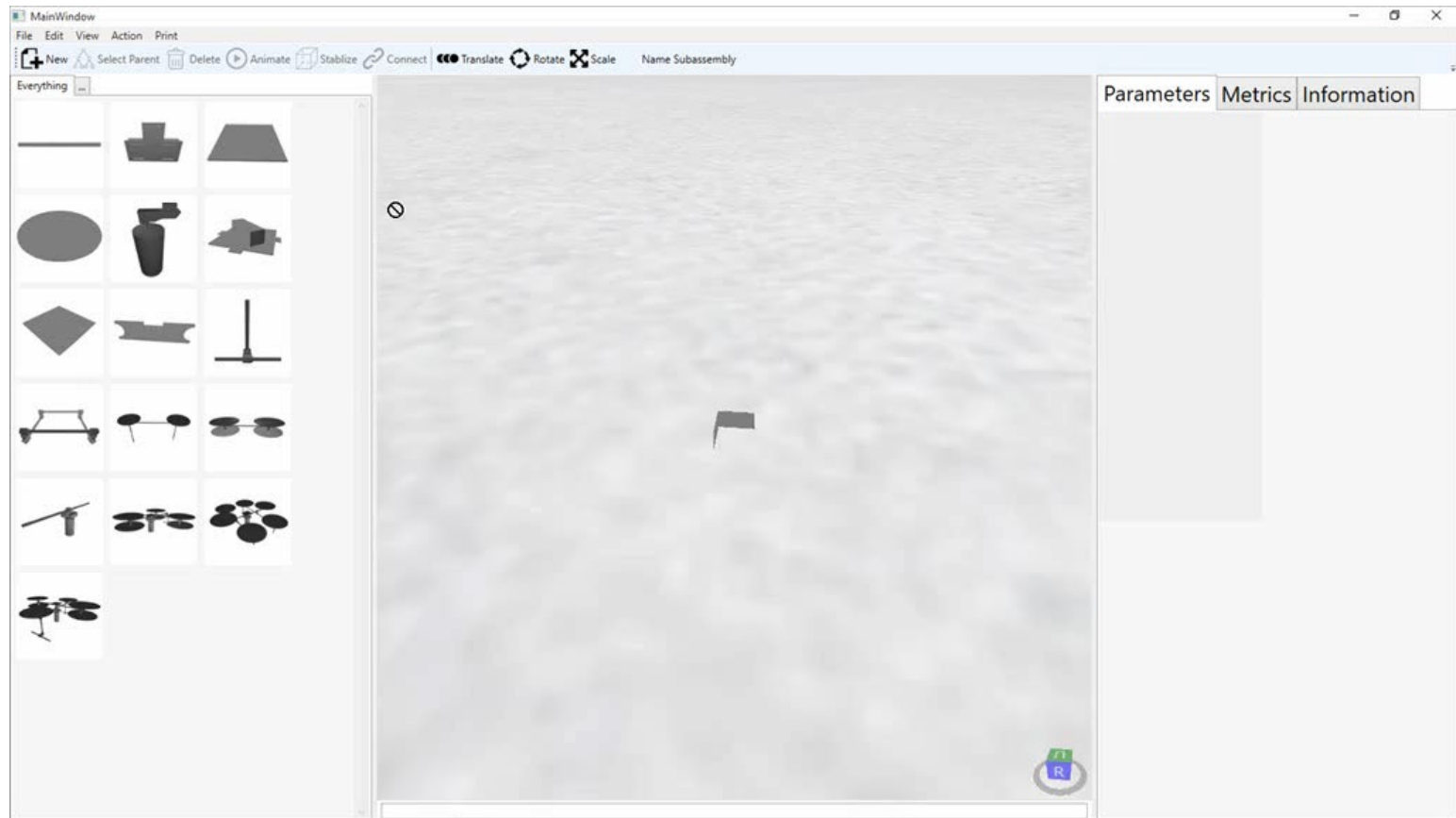


Motivation



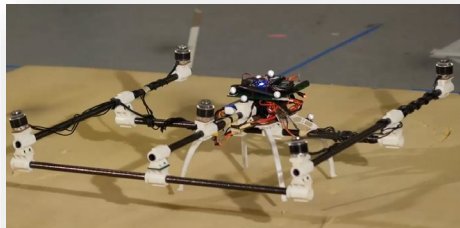


Multicopter Design Interface

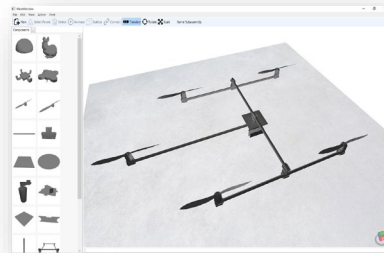




Controller Design



Geometry design



Dynamics

$$m\ddot{\vec{p}} = m\vec{g} + R \sum_i T_i \vec{d}_i$$

$$I\dot{\vec{\omega}} + \vec{\omega} \times I\vec{\omega} = \sum_i \lambda T_i \vec{d}_i + \vec{r}_i \times \vec{d}_i$$

Linear controller

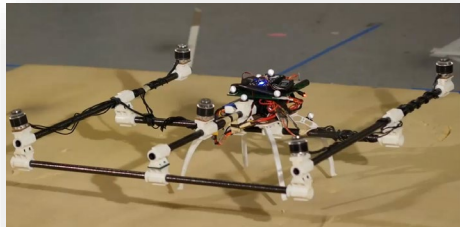
LQR

Linearization around a point $(\vec{x}^*, \vec{T}^*)$

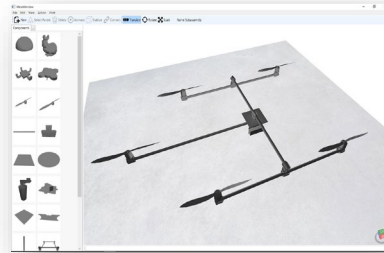
$$(\vec{x} - \vec{x}^*) \approx \frac{\partial h}{\partial \vec{x}} (\vec{x} - \vec{x}^*) + \frac{\partial h}{\partial \vec{u}} (\vec{T} - \vec{T}^*)$$



Controller Design



Geometry design



Dynamics

$$m\ddot{\vec{p}} = m\vec{g} + R \sum_i T_i \vec{d}_i$$
$$I\dot{\vec{\omega}} + \vec{\omega} \times I\vec{\omega} = \sum_i \lambda T_i \vec{d}_i + \vec{r}_i \times \vec{d}_i$$

Linear controller

LQR

Linearization around a point $(\vec{x}^*, \vec{T}^*)$?

$$(\vec{x} - \vec{x}^*) \approx \frac{\partial h}{\partial \vec{x}} (\vec{x} - \vec{x}^*) + \frac{\partial h}{\partial \vec{u}} (\vec{T} - \vec{T}^*)$$



Optimization

Controller Parameters: $(\vec{x}^*, \vec{T}^*)$

Geometry Parameters: $\vec{s}$

Typical metrics

$$E_{size} := \vec{a} \cdot \vec{s} + b$$

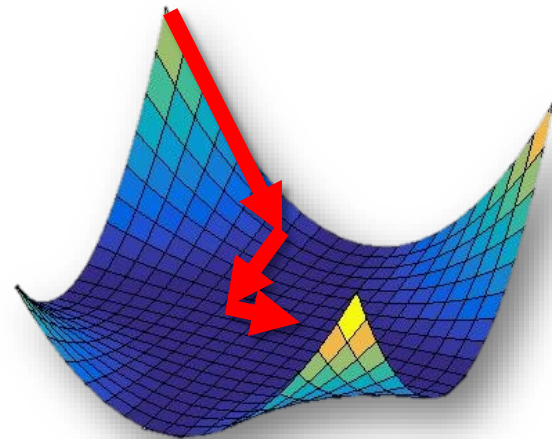
$$E_{mass} := \vec{a} \cdot \vec{s} + b$$

$$E_{amperage} := a(T_i^*)^{b_i}$$

$$E_{payload} := \max_i T_i^*$$



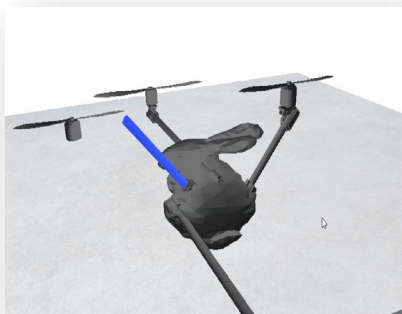
$$E_{metric} = (\text{bi})\text{convex}(\vec{s}, \vec{T}^*)$$



Alternating direction method



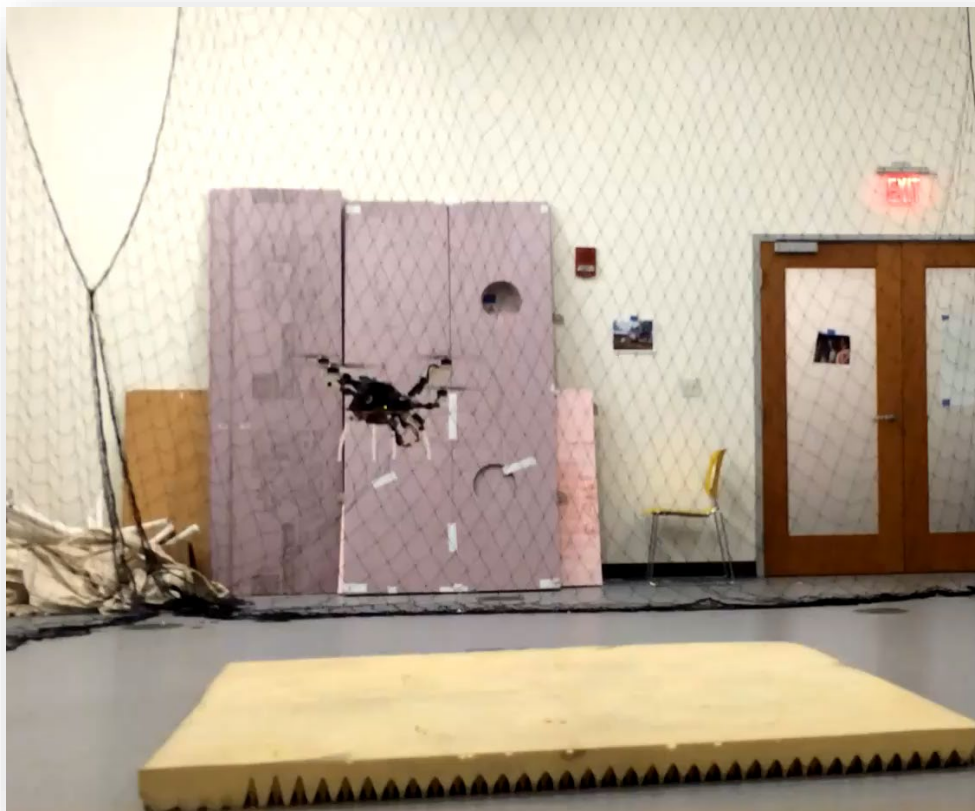
Bunny-Copter Example



Design

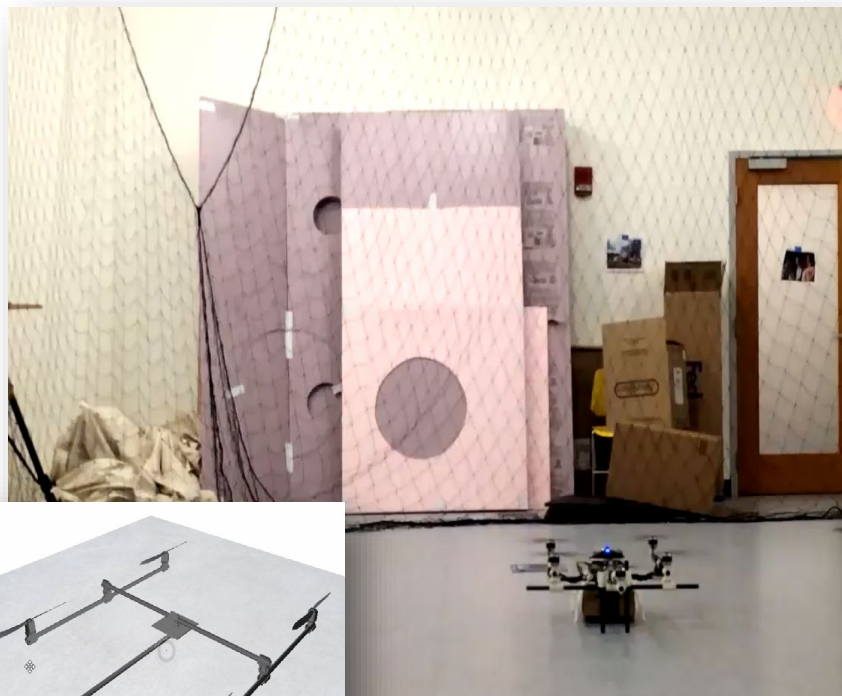


Fabrication

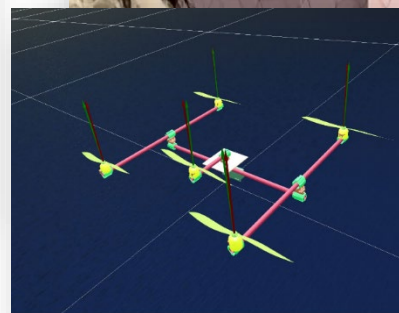
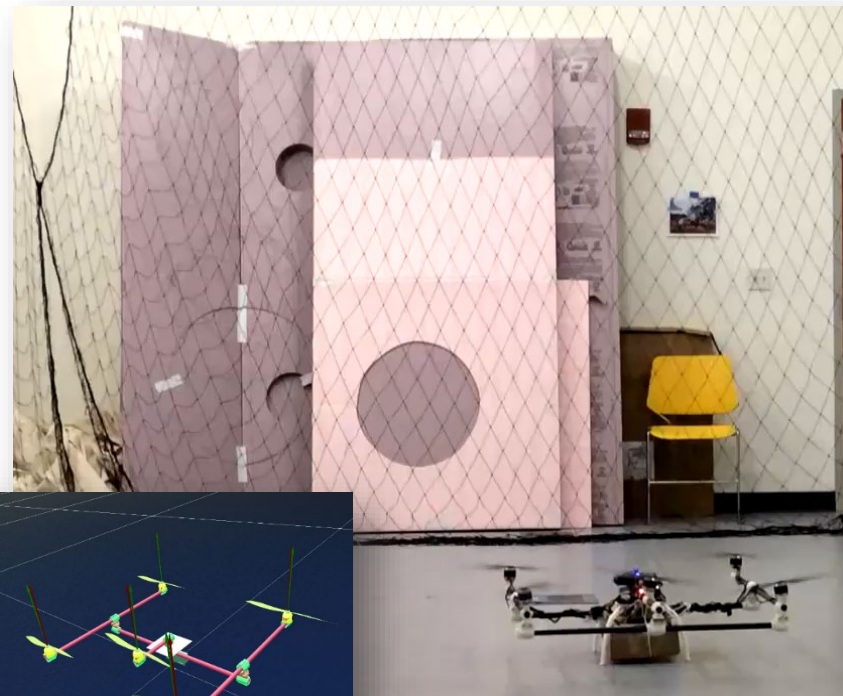




Optimized Pentacopter



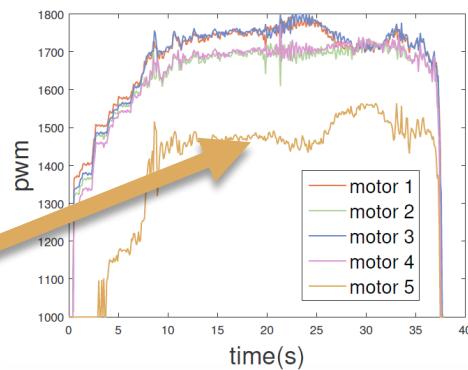
Original Design



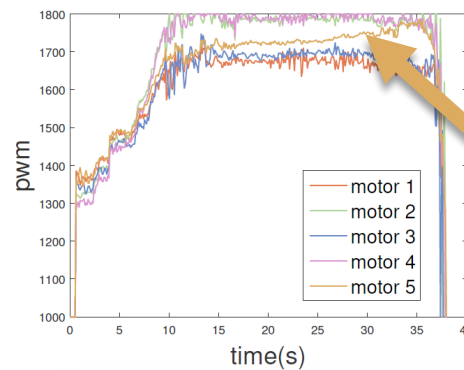
Optimized Result



Optimization Example

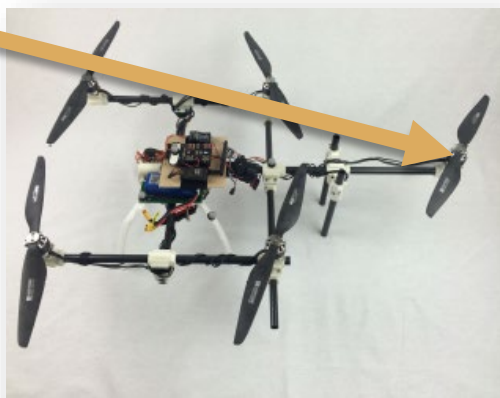


Motor 5



Motor 5

**Old
payload:
1047g**



**New payload:
1392g**

**33%
increase**