

The Semantics of Shape

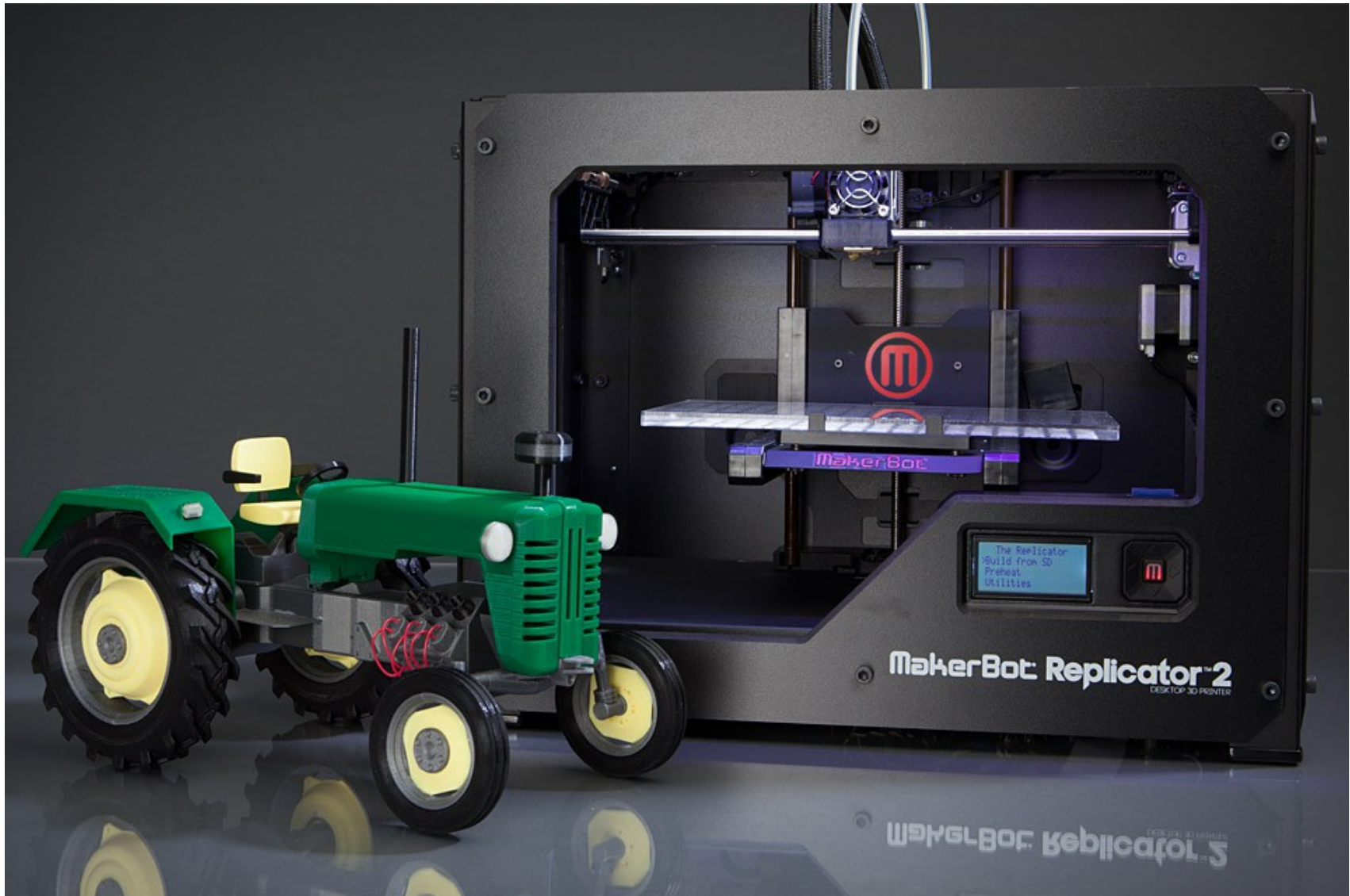
Computational Methods for
High-Level 3D Shape Analysis

Siddhartha Chaudhuri
IIT Bombay

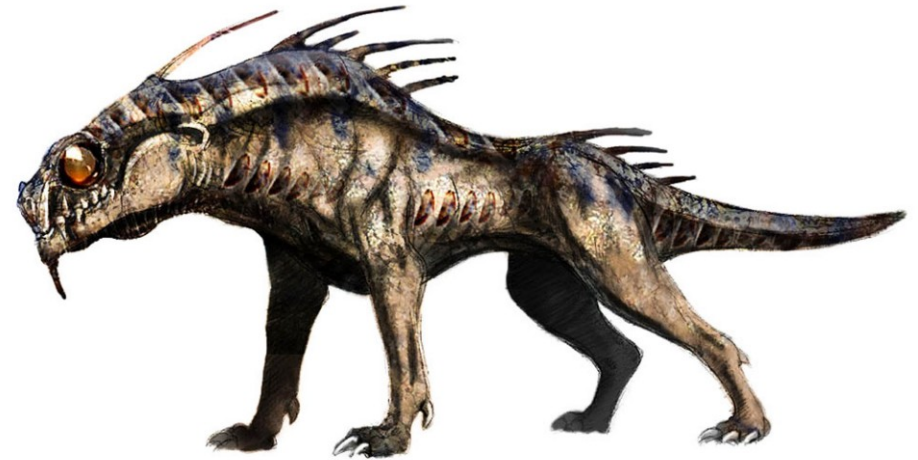
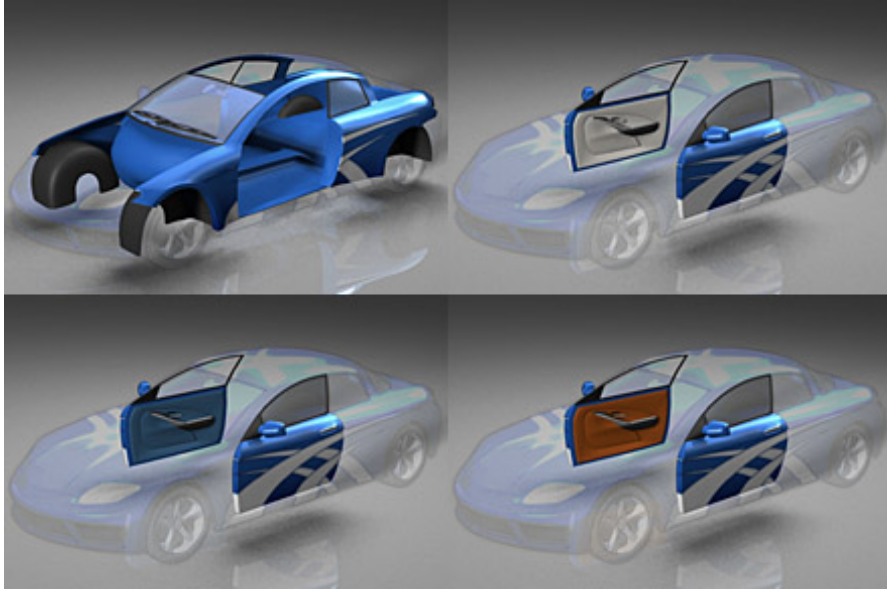




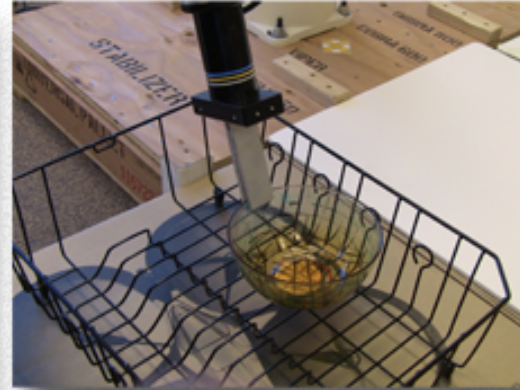
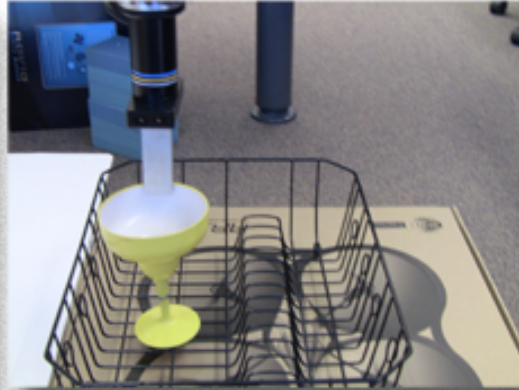
Shapes are everywhere!



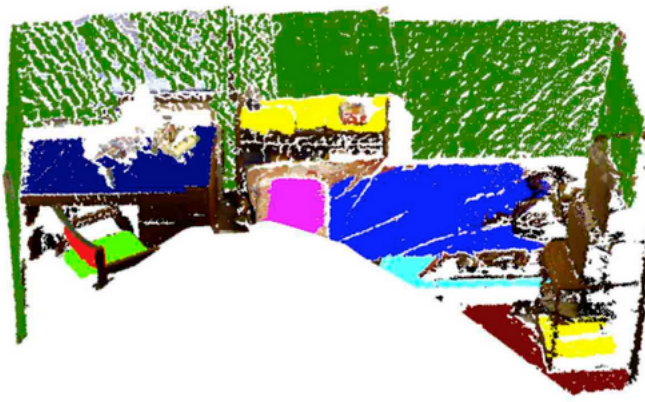
Shapes are everywhere!



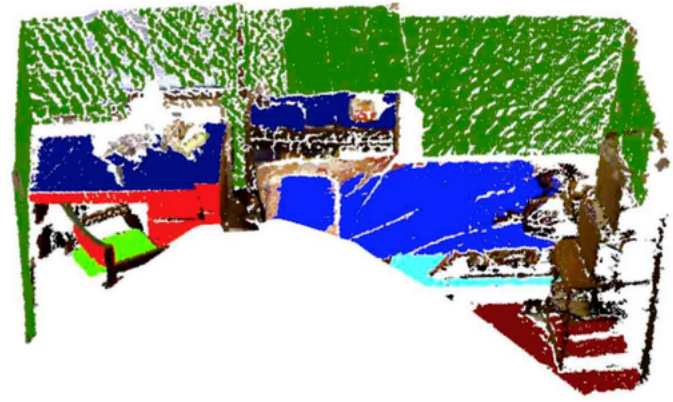
Shapes are everywhere!



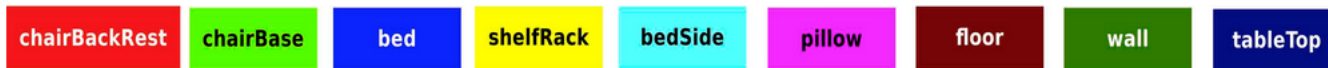
Original Scene



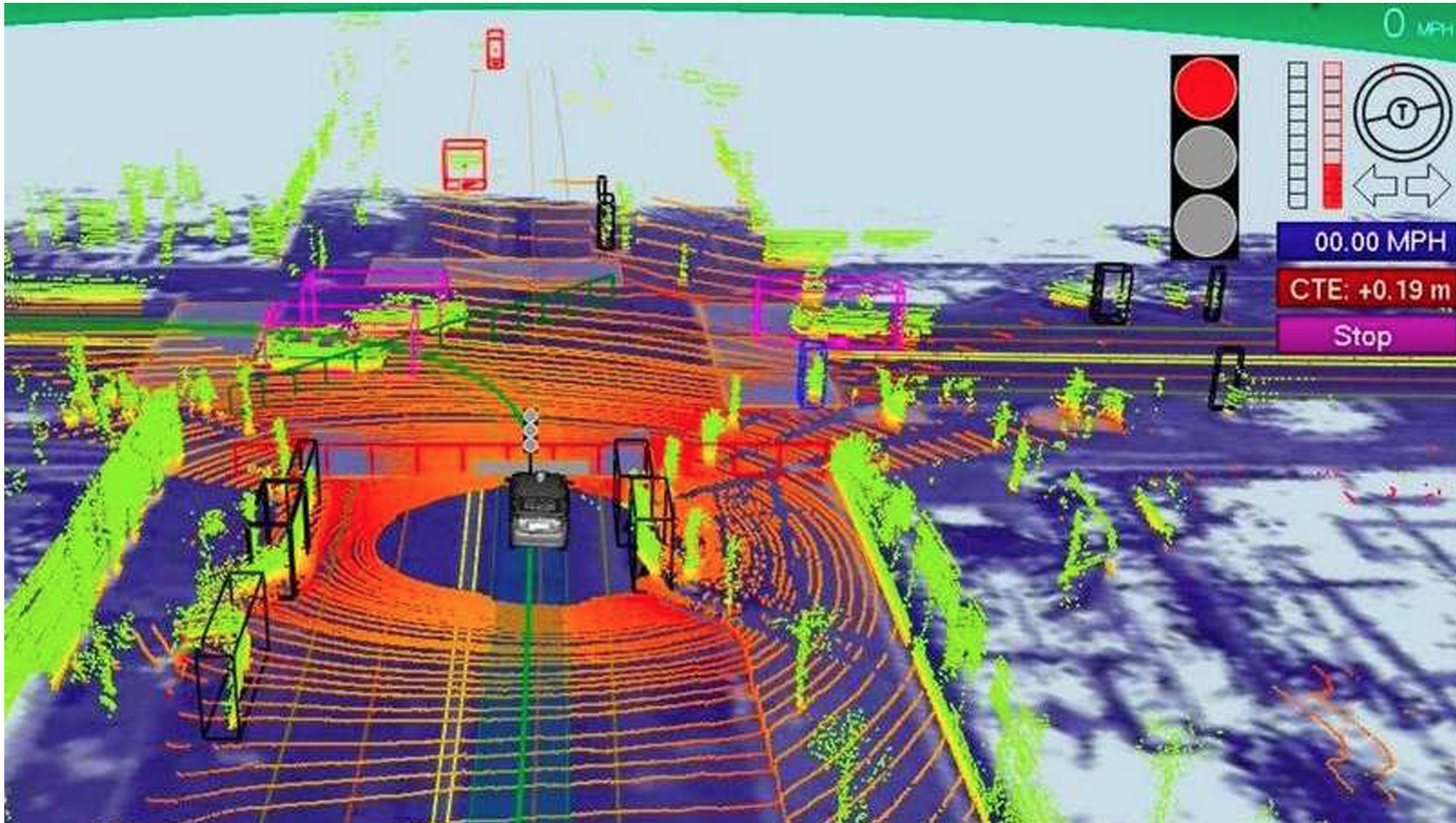
Ground Truth Labels



Predicted Labels



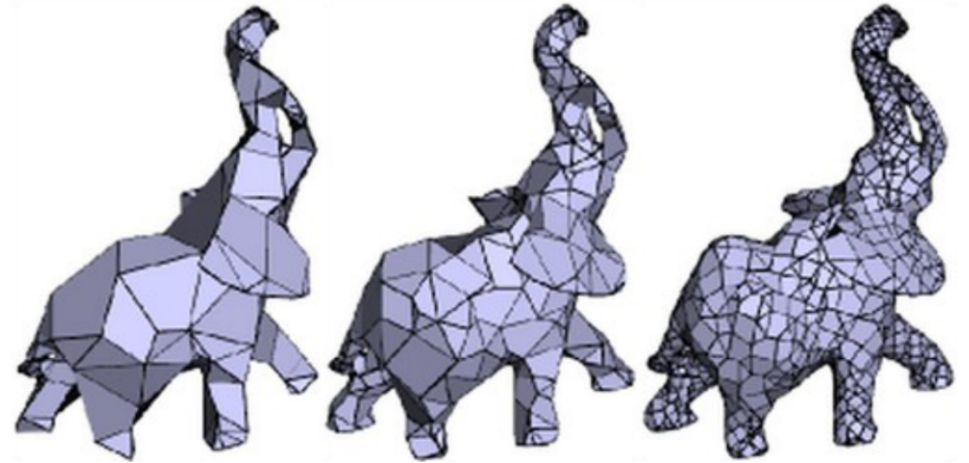
Shapes are everywhere!



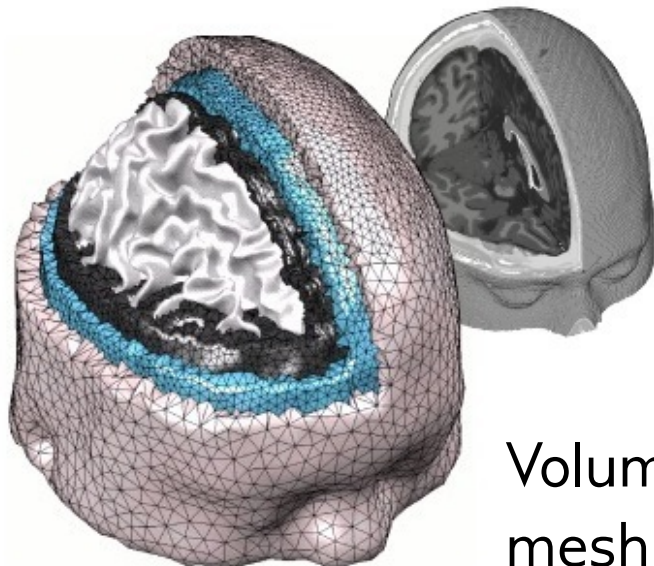
Shape Representations



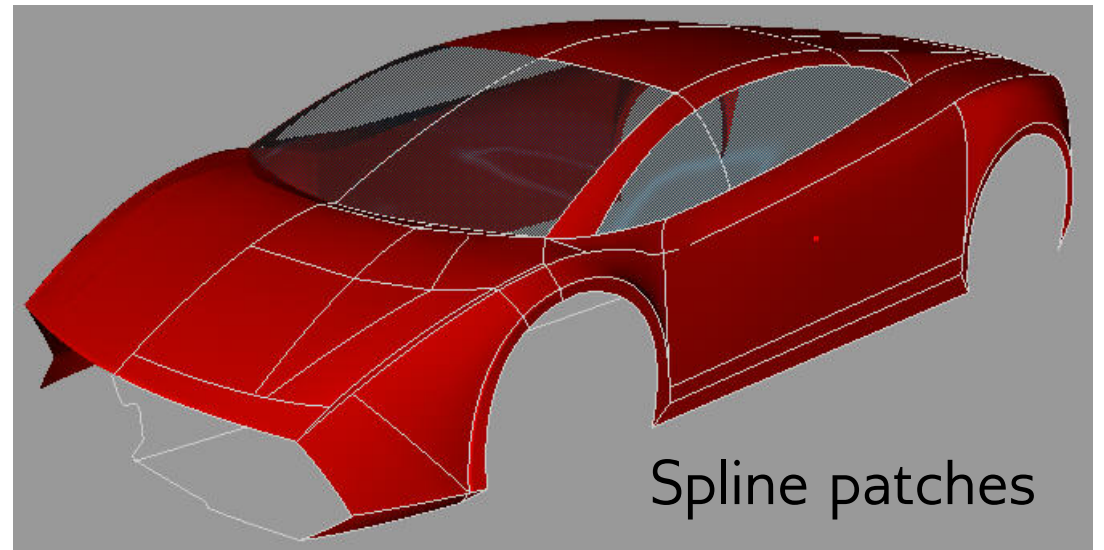
Point cloud



Polygon mesh

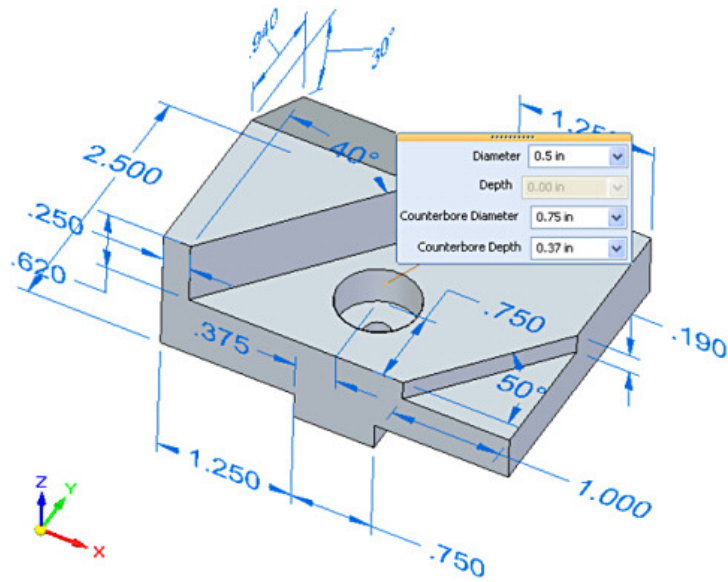


Volumetric
mesh

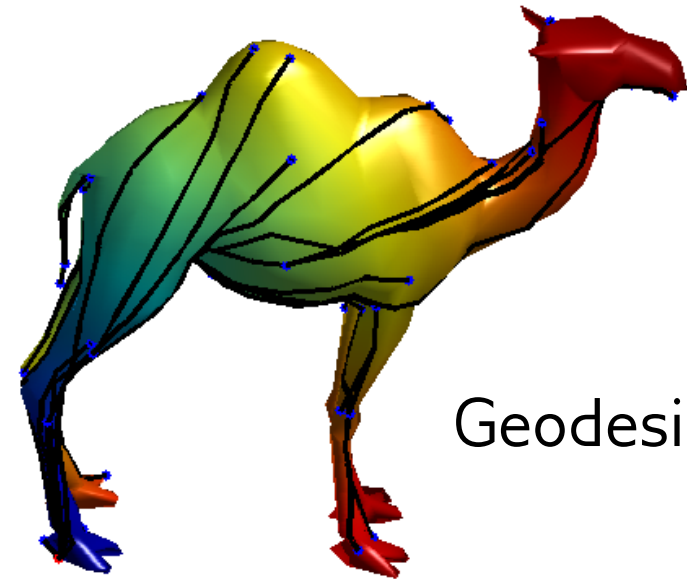


Spline patches

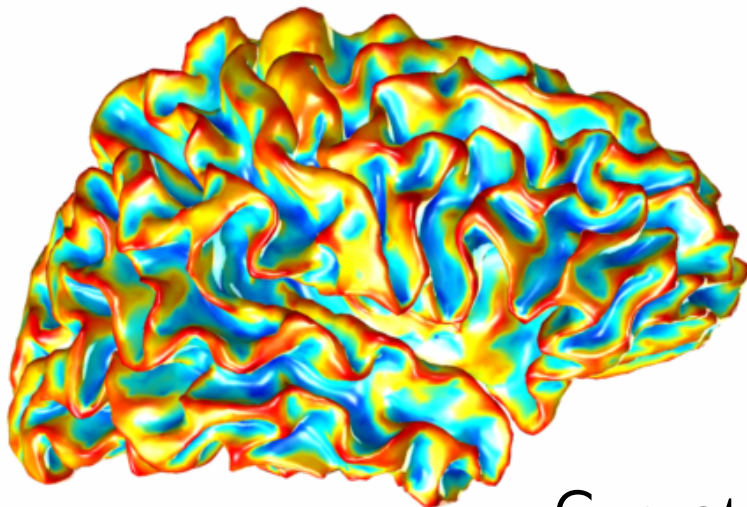
“Low-Level” Geometric Analysis



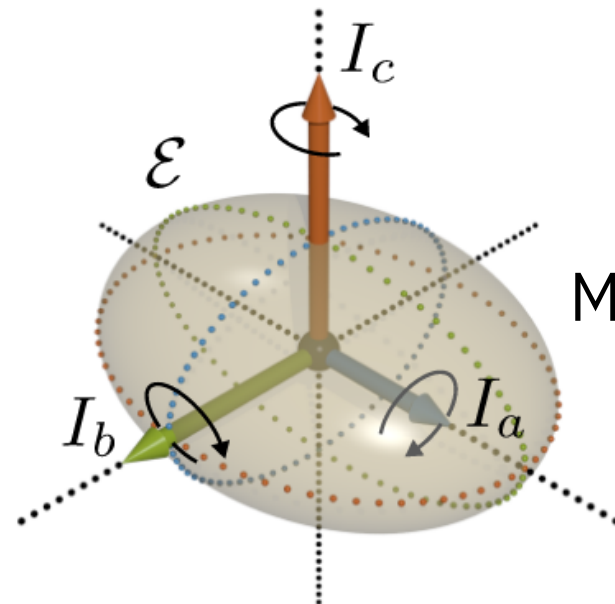
Dimensions



Geodesics

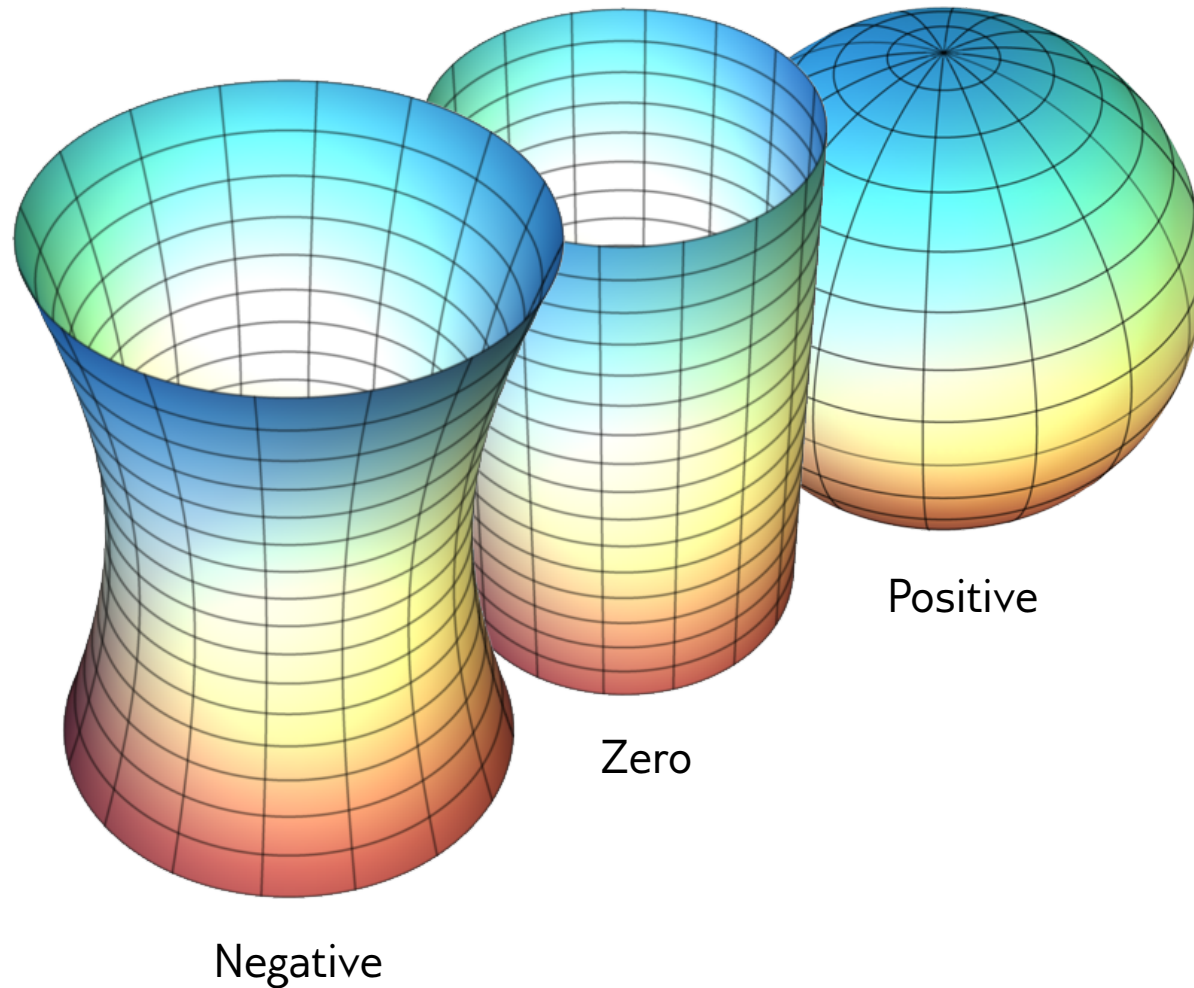


Curvature

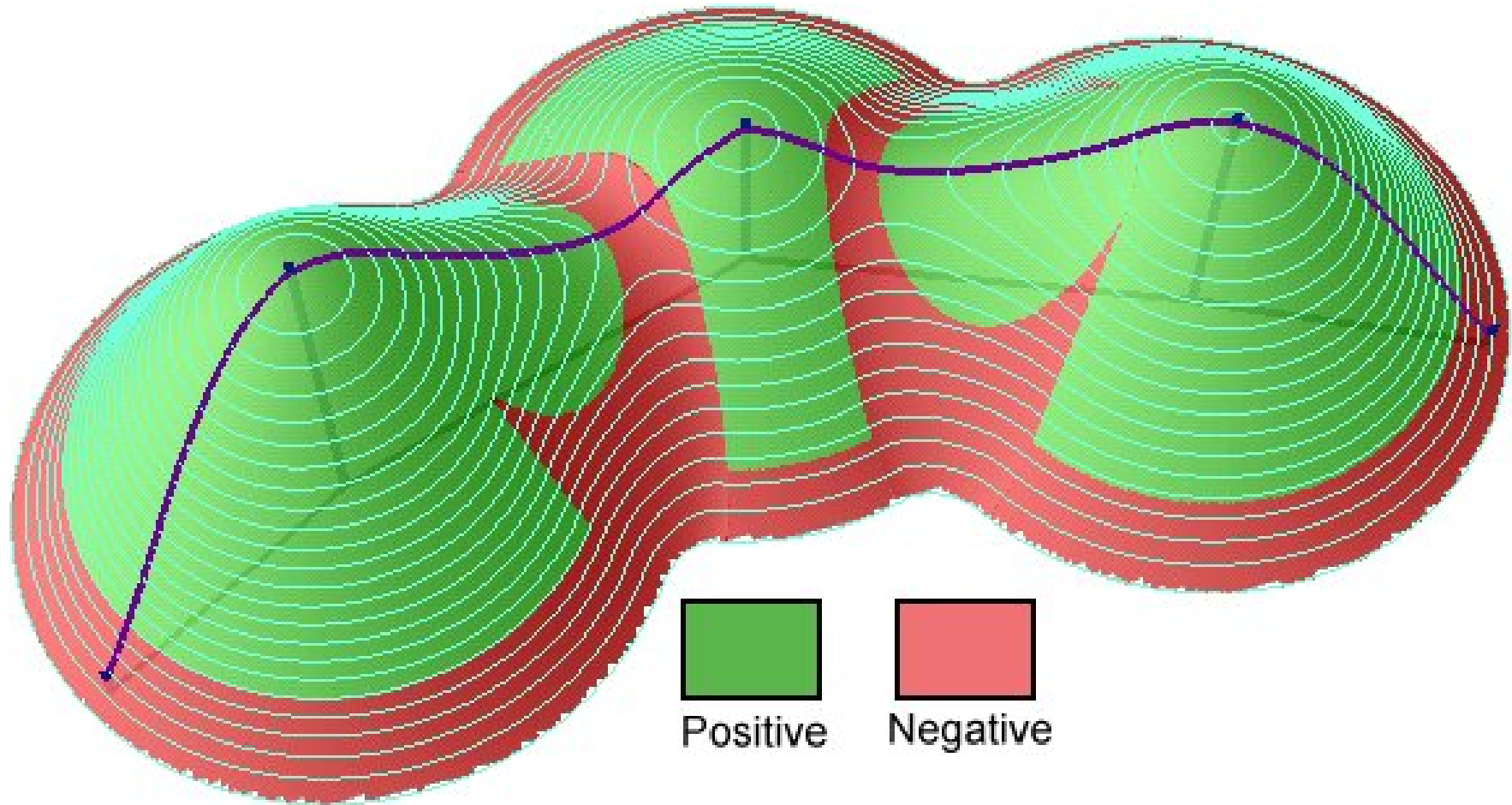


Moments

Gaussian Curvature



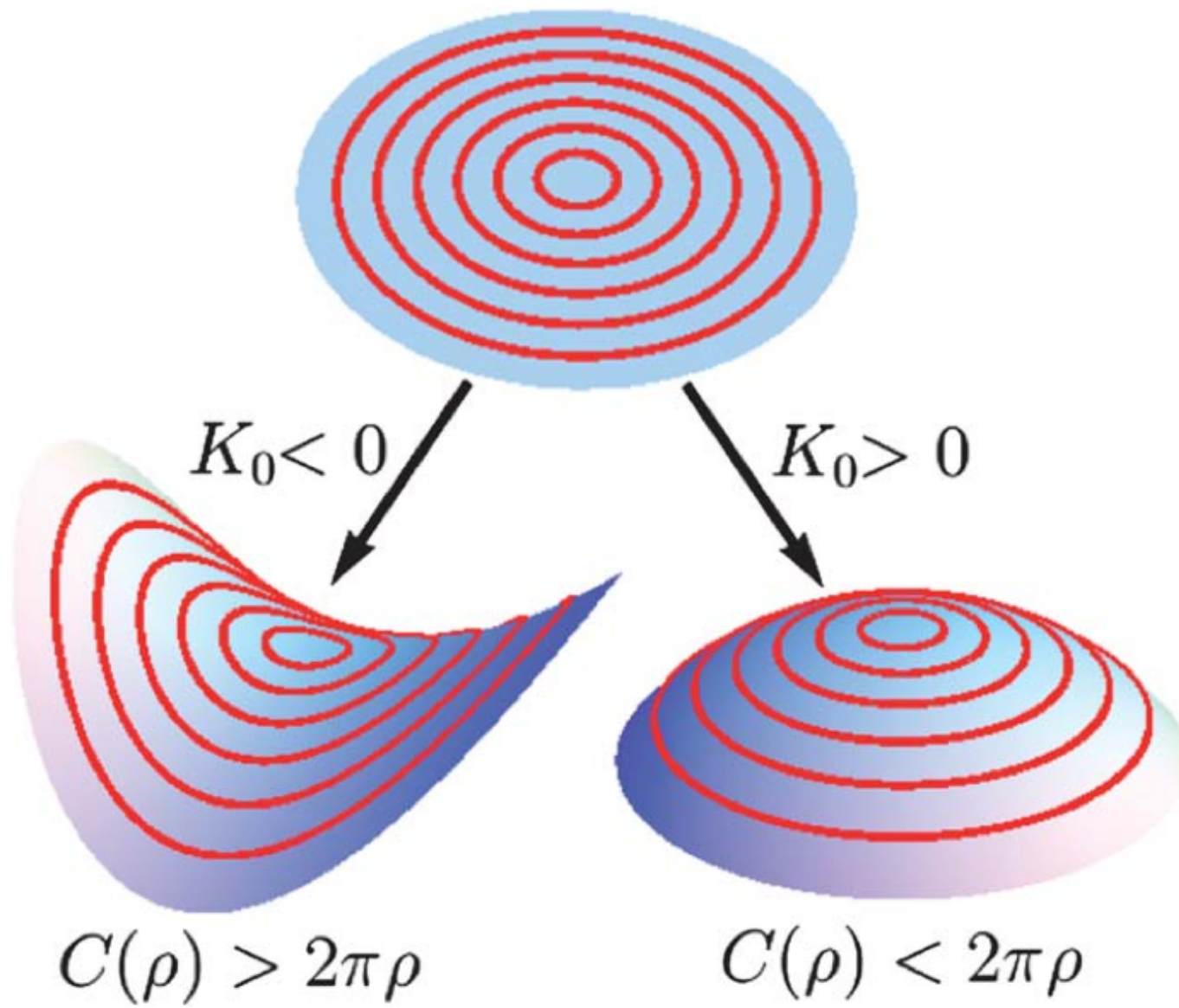
Gaussian Curvature



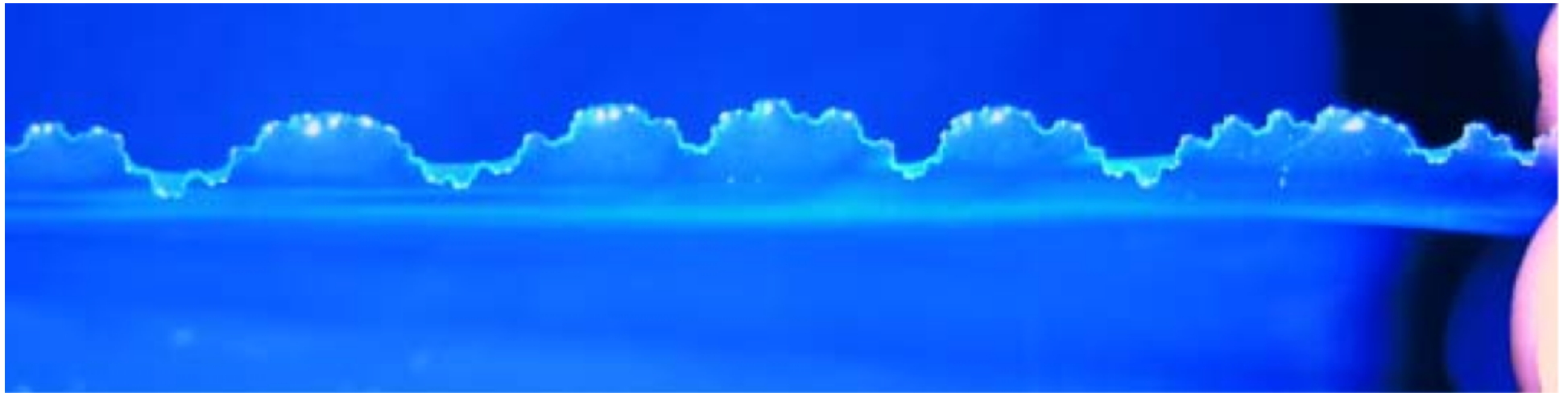
Can a 2D ant on a 2D surface tell if it lives in a space of positive, negative or zero curvature?

Can a person, in 3D?

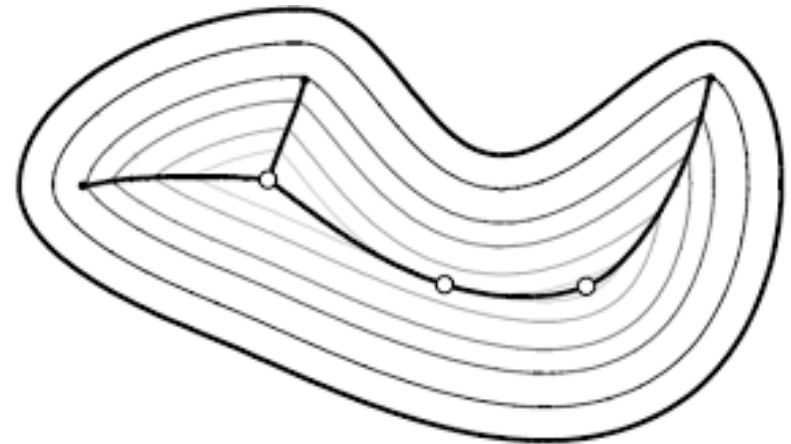
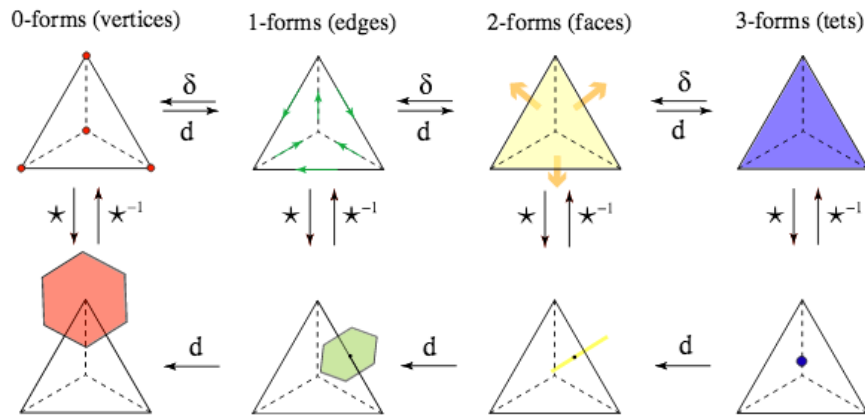
Yes, by measuring distances!







“Low-Level” Geometric Analysis



Discrete Differential Geometry

Medial Axis Transform



Spectral Decomposition

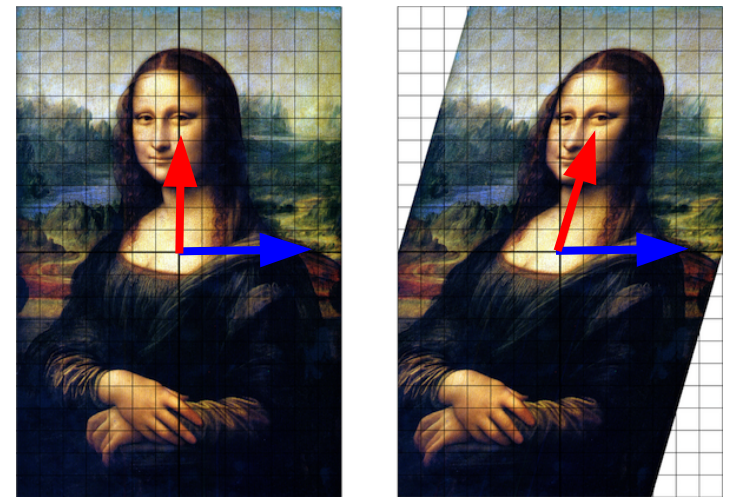
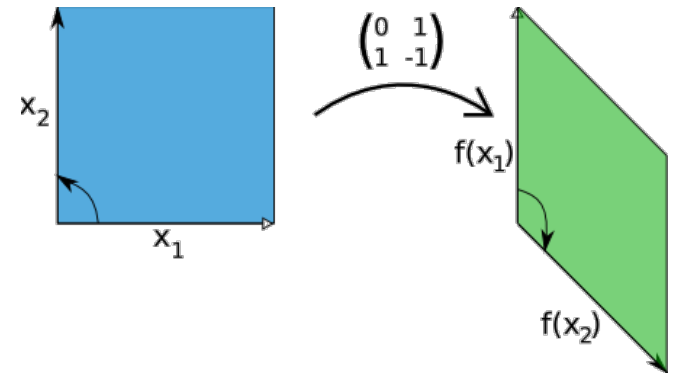
Matrices as transformations

- Let A be an $n \times n$ matrix
 - It can be thought of as a function that maps a vector $\mathbf{x} \in \mathbb{R}^n$ to a vector $A\mathbf{x} \in \mathbb{R}^n$
- A is a **linear transformation**
 - f is linear if $f(a + b) = f(a) + f(b)$
- An **eigenvalue** of A is a scalar λ such that

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

where \mathbf{x} is some n -D vector

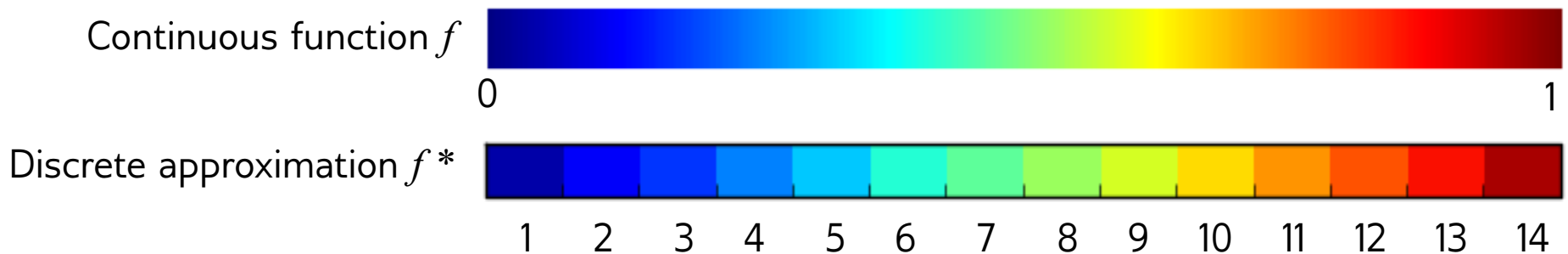
- \mathbf{x} is the corresponding **eigenvector**



Blue arrow is eigenvector of shear transform, red is not

Functions as vectors

- Functions from A to B form a vector space: we can think of functions as “vectors”
 - E.g. we can commutatively add two functions:
 $f + g = g + f$
 - Or distribute multiplication with a scalar: $s(f + g) = sf + sg$
- A function f can be **discretized** to an n -D vector of sampled values: $[f(x_1), f(x_2), \dots, f(x_n)]$



Linear operators

- An **operator** T is a mapping from a vector space U to another vector space V
 - T is a **linear operator** if $T(a + b) = T(a) + T(b)$
- The set of functions F from domain A to codomain B is a vector space
 - So we can have operators T that map from one function space F to another function space G
 - Note that T maps functions to functions!
- The differentials $\frac{d}{dx}$, $\frac{d^2}{dx^2}$, $\frac{d^3}{dx^3}$ etc are linear operators
 - They map functions to their derivatives

Eigenfunctions of operators

- An **eigenvalue** of a linear operator T that maps a vector space to itself is a scalar λ s.t.

$$T(\mathbf{x}) = \lambda \mathbf{x}$$

and \mathbf{x} is the corresponding **eigenvector**

- If T maps functions to functions, then we call \mathbf{x} an **eigenfunction**: $T(f) = \lambda f$

Discrete Linear Operators

- **Theorem:** Any linear operator between finite-dimensional vector spaces can be represented by a matrix
 - Let's say we have a set of functions F from A to B
 - The discrete versions of the functions form a finite-dimensional vector space F^* equivalent to \mathbb{R}^n
 - Each function is sampled at the same finite set of points
 - Let T be a linear operator from F to itself
 - ... and T^* be a “discrete version” of T acting on F^*
 - Then T^* can be represented by a $n \times n$ matrix (cf. theorem)

Example: Discrete Derivative

Continuous

- Function: f
- Operator: $\frac{d}{dx}$
- Applying operator:

$$\frac{df}{dx} = f'$$

Discrete

- Vector: $\mathbf{f} = [f(x_1), f(x_2) \dots f(x_n)]$
- Matrix:

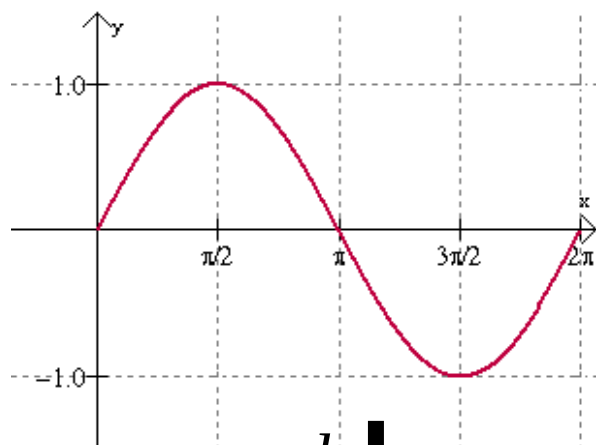
$$A = \frac{1}{h} \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ & 0 & -1 & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & -1 & 1 \\ 0 & 0 & \dots & & 0 & -1 \end{bmatrix}$$

- Applying matrix:

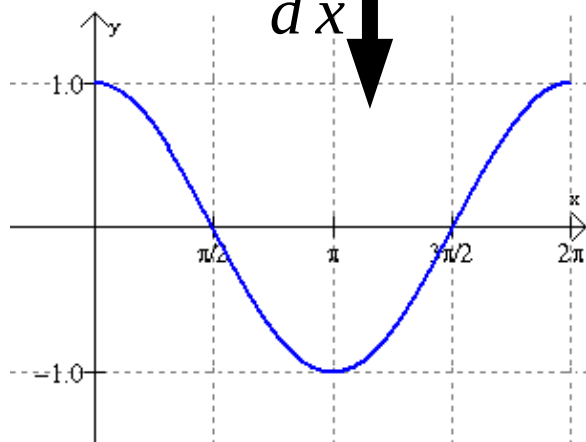
$$A\mathbf{f} = \mathbf{f}'$$

Example: Discrete Derivative

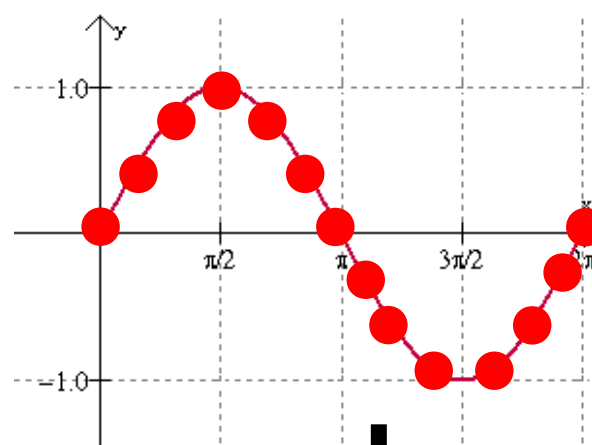
Continuous



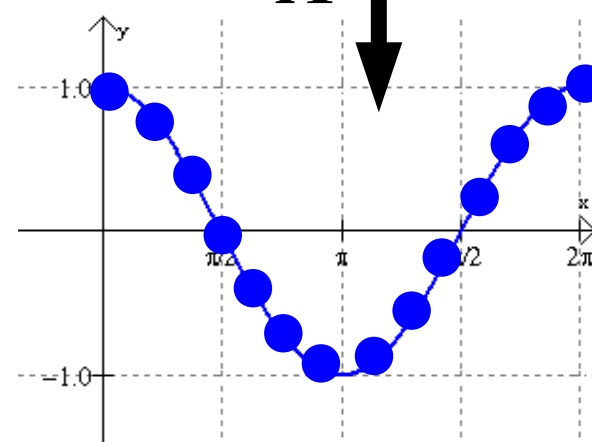
$\frac{d}{dx}$



Discrete



A



Example: Discrete 2nd Derivative

Continuous

- Function: f
- Operator: $\frac{d^2}{dx^2}$
- Applying operator:

$$\frac{d^2 f}{dx^2} = f''$$

Discrete

- Vector: $\mathbf{f} = [f(x_1), f(x_2) \dots f(x_n)]$
- Matrix:

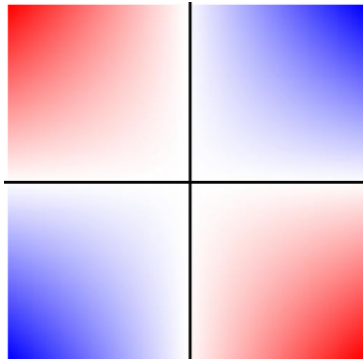
$$L = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 \\ & 1 & -2 & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & -2 & 1 \\ 0 & 0 & \dots & & 1 & -2 \end{bmatrix}$$

- Applying matrix:

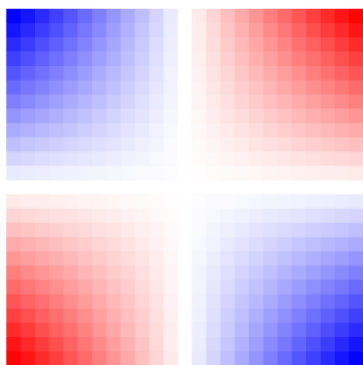
$$L\mathbf{f} = \mathbf{f}''$$

Operators in higher dimensions

- The underlying function space can have a higher-dimensional domain



Continuous function



Discrete approximation

$$\begin{bmatrix} -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & -4 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & -4 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & -4 & 1 & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & 1 & -4 & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & \cdot & 1 & -4 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & -4 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & -4 \end{bmatrix}$$

2D discrete Laplace operator

Interpreting eigenfunctions

- Eigenvalues of a linear operator form its **spectrum**
- The eigenfunctions are unchanged (except for scaling) when transformed by the operator
 - Think of them as standing waves on the domain
- E.g.

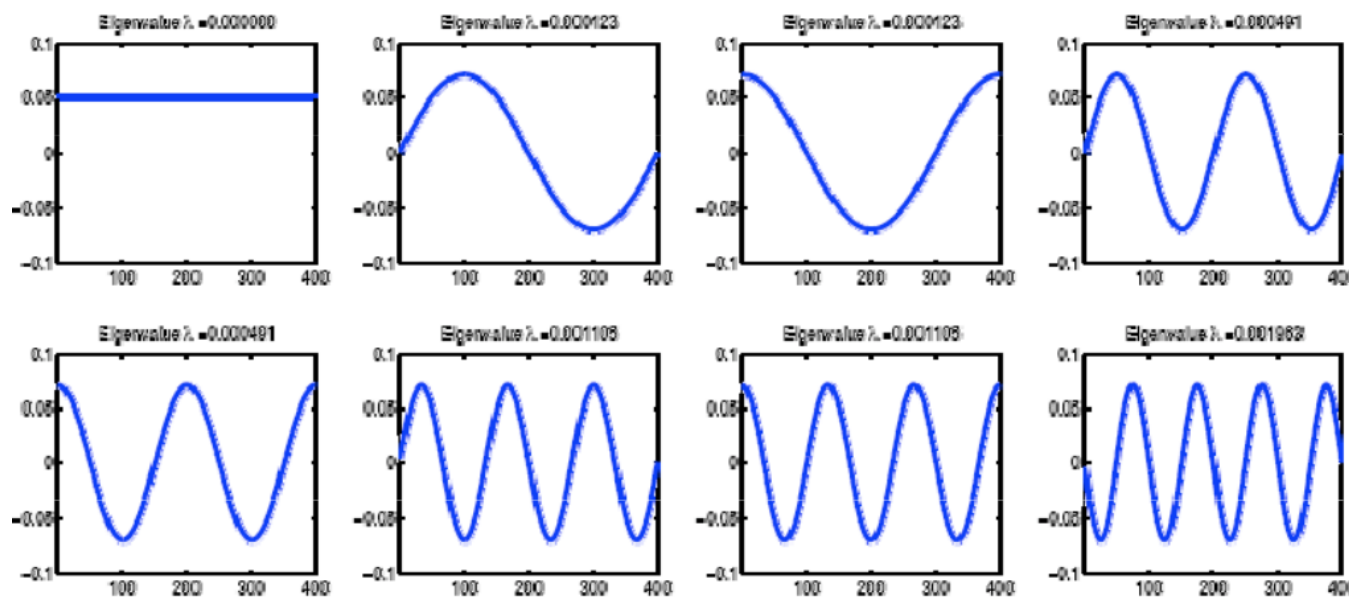
$$\frac{d^2 \sin(n x)}{d x^2} = -n^2 \times \sin(n x)$$

$$\frac{d^2 \cos(n x)}{d x^2} = -n^2 \times \cos(n x)$$

$$\frac{d^2 e^{\lambda x}}{d x^2} = \lambda^2 \times e^{\lambda x}$$

Interpreting eigenfunctions

- The eigenfunctions of the operator form a basis for the function space
 - E.g. the sinusoidal eigenfunctions of $\frac{d^2}{dx^2}$ form the Fourier basis



The first 8 sinusoidal eigenfunctions of the second derivative operator.
The eigenvalues are the negative squared frequencies.

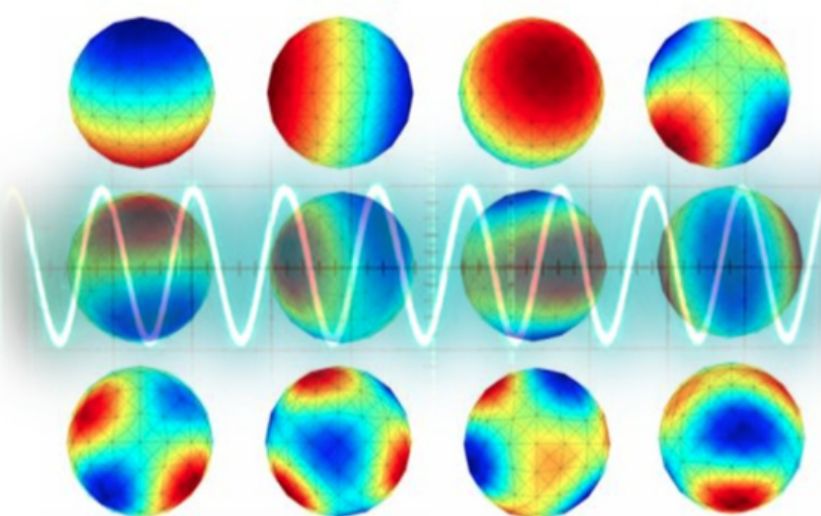
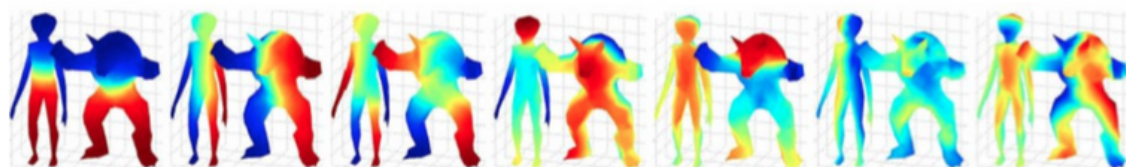
Operators on manifolds

- We can define a function on a manifold curve/surface!
 - E.g. the coordinate function: gives the (X, Y, Z) position of a point on the surface
- A common operator is the Laplace-Beltrami operator
 - Its eigenfunctions define a basis for functions over the surface



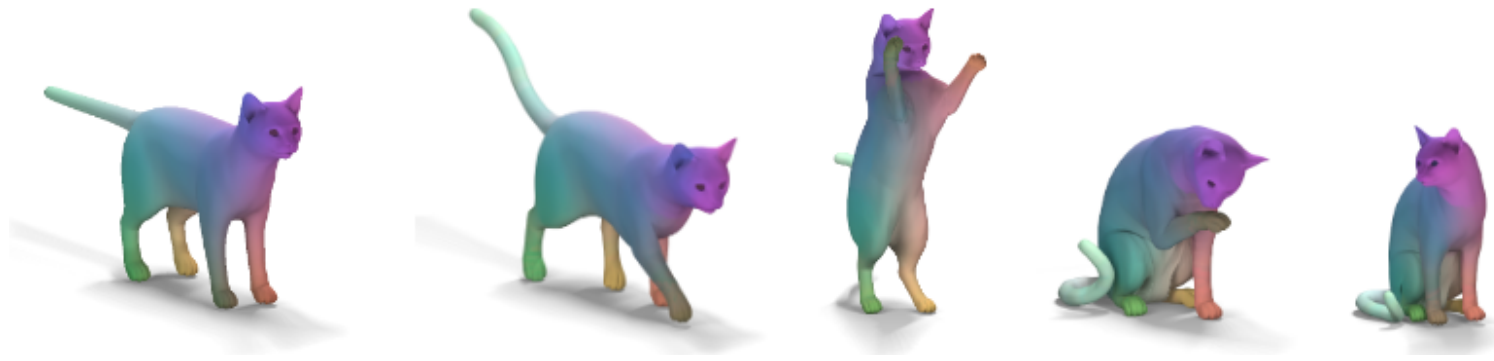
Eigenfunctions of Laplace-Beltrami

- *Intrinsic* basis for functions over surface
 - Doesn't change under isometry
- We can discretize it as usual: the function is defined at a fixed set of sample points on the shape



Eigenfunctions of Laplace-Beltrami

- The spectrum of the L-B operator characterizes the intrinsic geometry of the shape
- Two shapes related by isometry have the same Laplace-Beltrami spectrum



Expressing a function with eigenfunctions

- **Continuous:**

$$f(p) = w_1 \varphi_1(p) + w_2 \varphi_2(p) + \dots + w_n \varphi_n(p)$$

- **Discrete:**

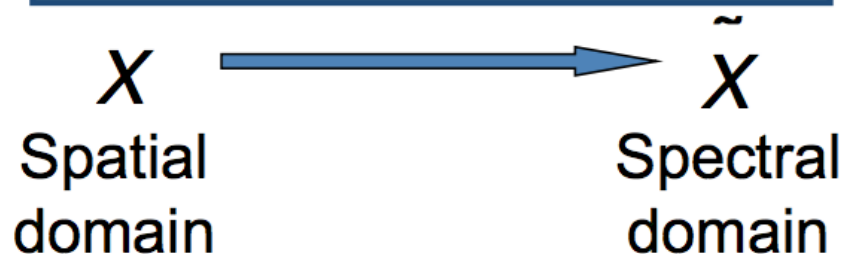
$$X = \sum_{i=1}^n \mathbf{e}_i \tilde{x}_i = \begin{bmatrix} E_{11} \\ E_{21} \\ \vdots \\ E_{n1} \end{bmatrix} \tilde{x}_1 + \dots + \begin{bmatrix} E_{1n} \\ E_{2n} \\ \vdots \\ E_{nn} \end{bmatrix} \tilde{x}_n = \begin{bmatrix} E_{11} & \dots & E_{1n} \\ E_{21} & \dots & E_{2n} \\ \vdots & \vdots & \vdots \\ E_{n1} & \dots & E_{nn} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} = E \tilde{X}$$

$$\tilde{X} = E^T X$$

$$\tilde{x}_i = \mathbf{e}_i^T \cdot X.$$

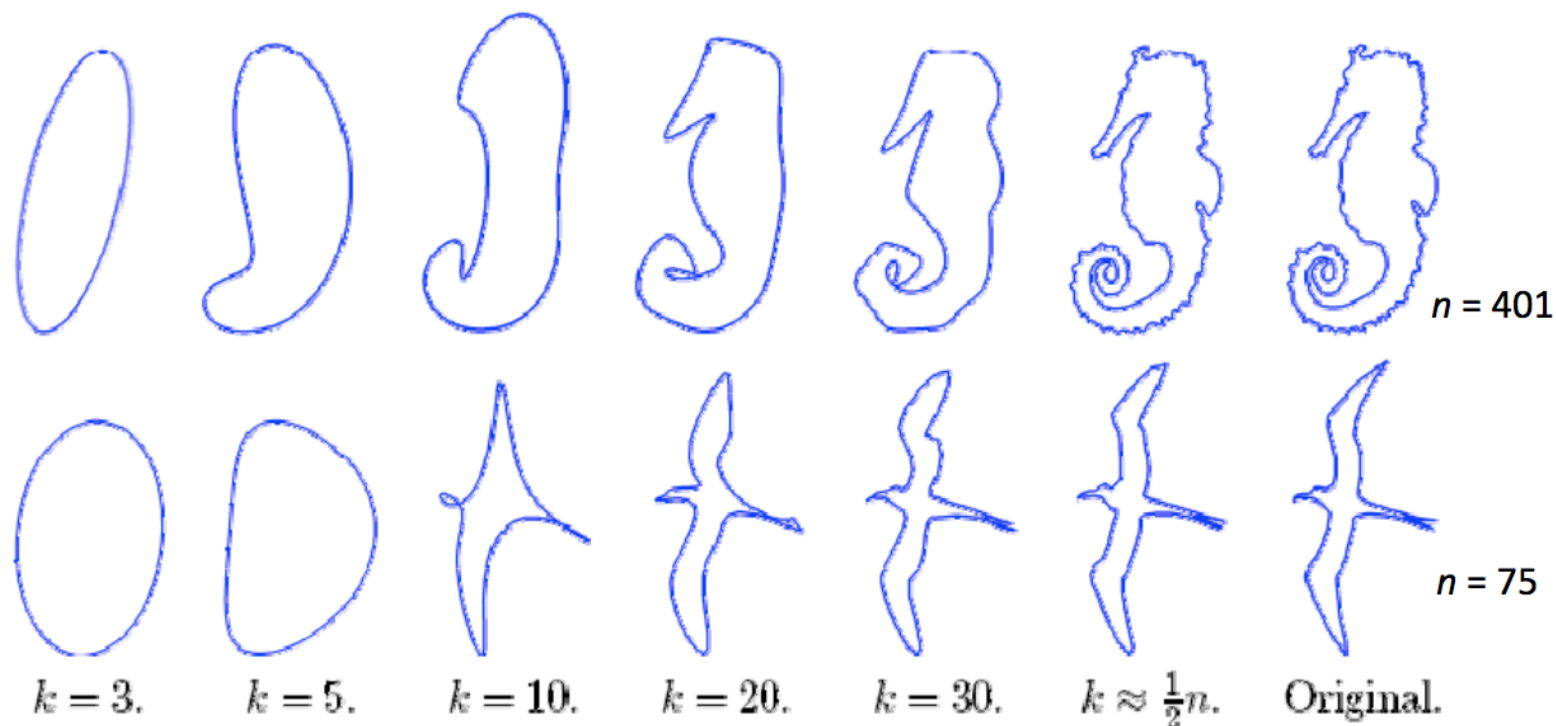
Projection of X
along eigenvector

The spectral transform



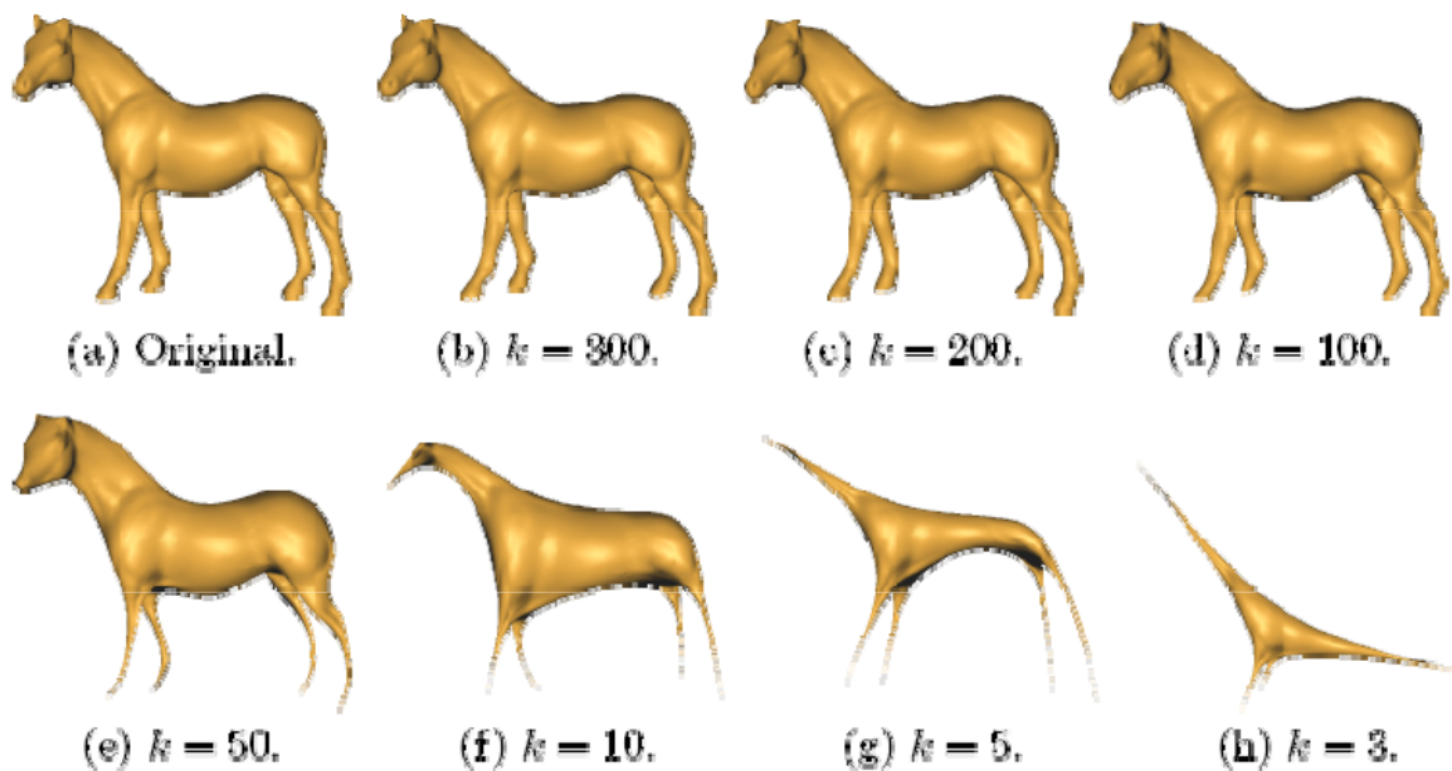
Reconstruction in 2D

- More accuracy with more eigenfunctions
- Function is the coordinate function
 - We're reconstructing the extrinsic shape of the object



Reconstruction in 3D

- More accuracy with more eigenfunctions
- Function is the coordinate function
 - We're reconstructing the extrinsic shape of the object



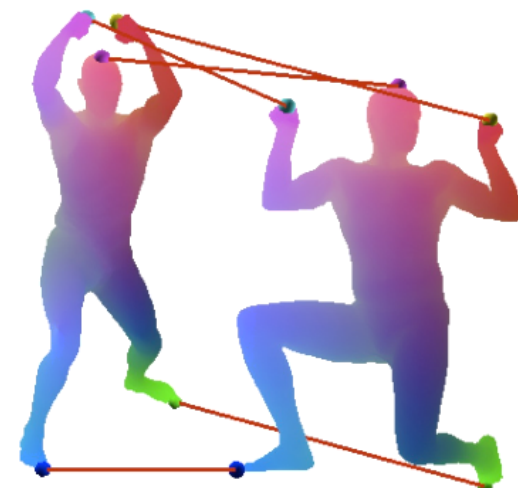
“Mid-Level” Geometric Analysis



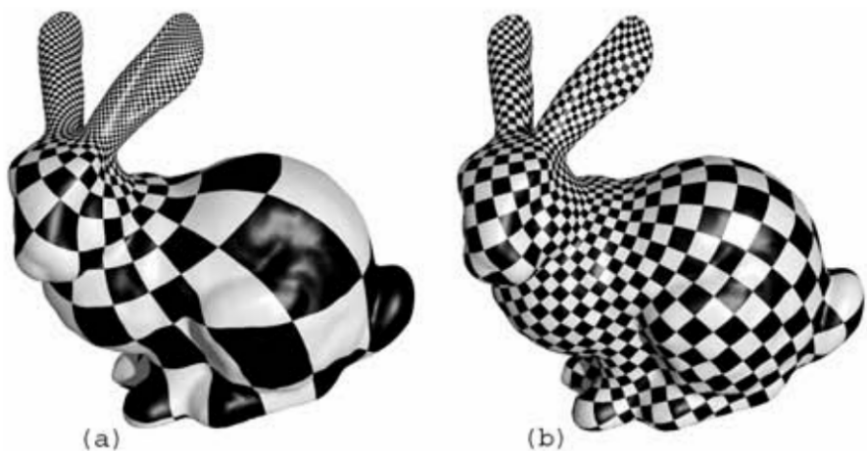
Retrieval



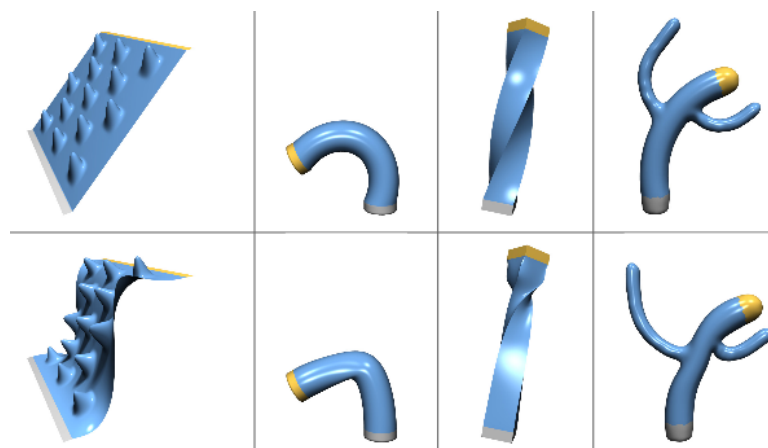
Segmentation



Correspondences



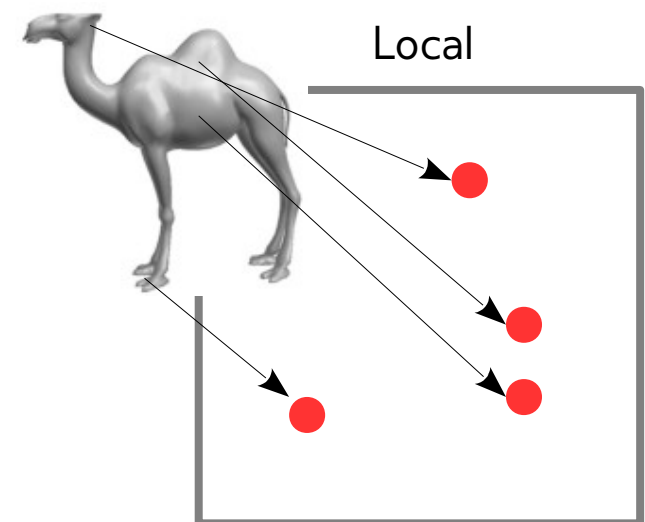
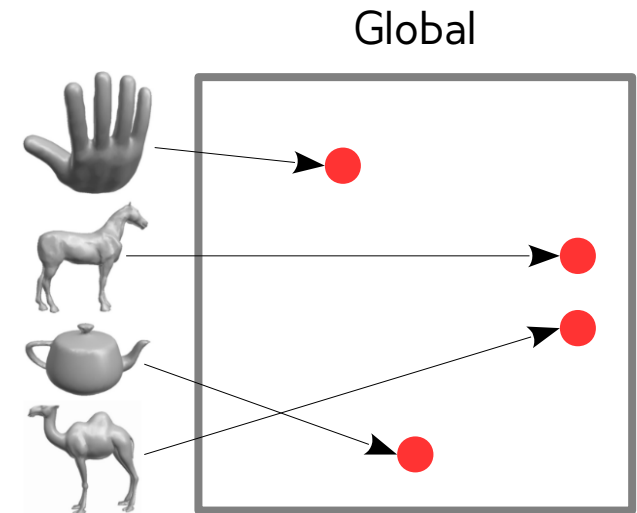
Parametrization



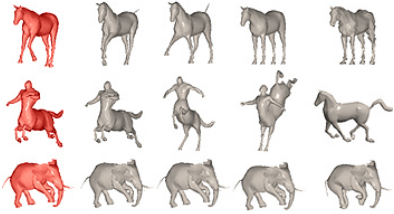
Deformation

Shape Descriptors

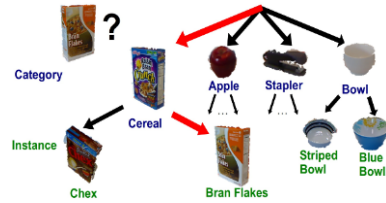
- A **shape descriptor** is a set of numbers that describes a shape in a way that is
 - **Concise**
 - **Quick to compute**
 - **Efficient to compare**
 - **Discriminative**
- **Global descriptors** describe whole objects
- **Local descriptors** describe (neighborhoods around) points
- Typically, the descriptors form a **vector space** with a **meaningful distance metric**



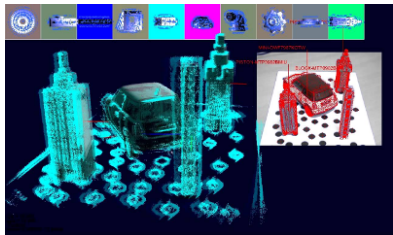
Global



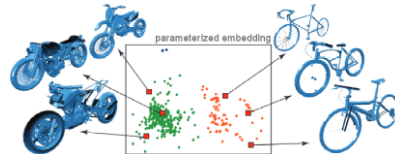
Retrieval



Classification



Recognition

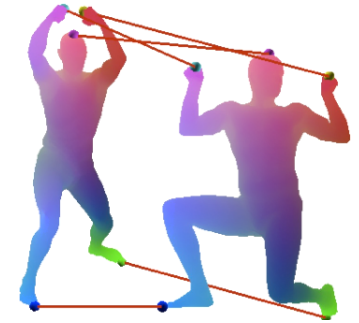


Clustering

Local



Feature detection



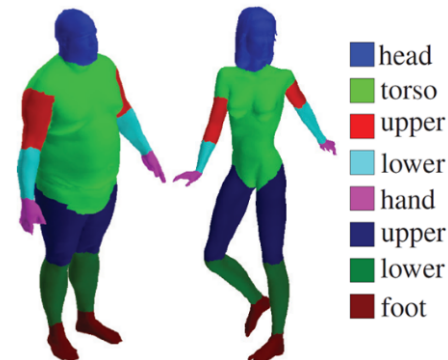
Correspondences



Registration



Symmetry detection

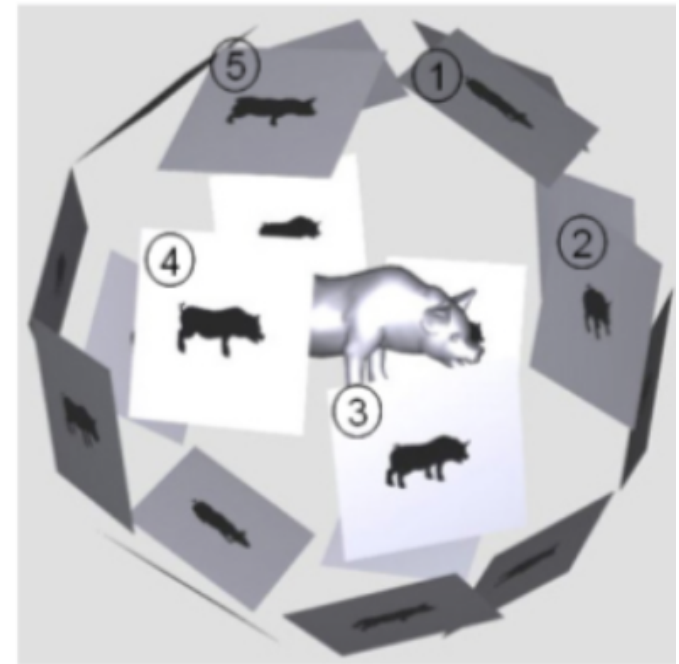


Segmentation

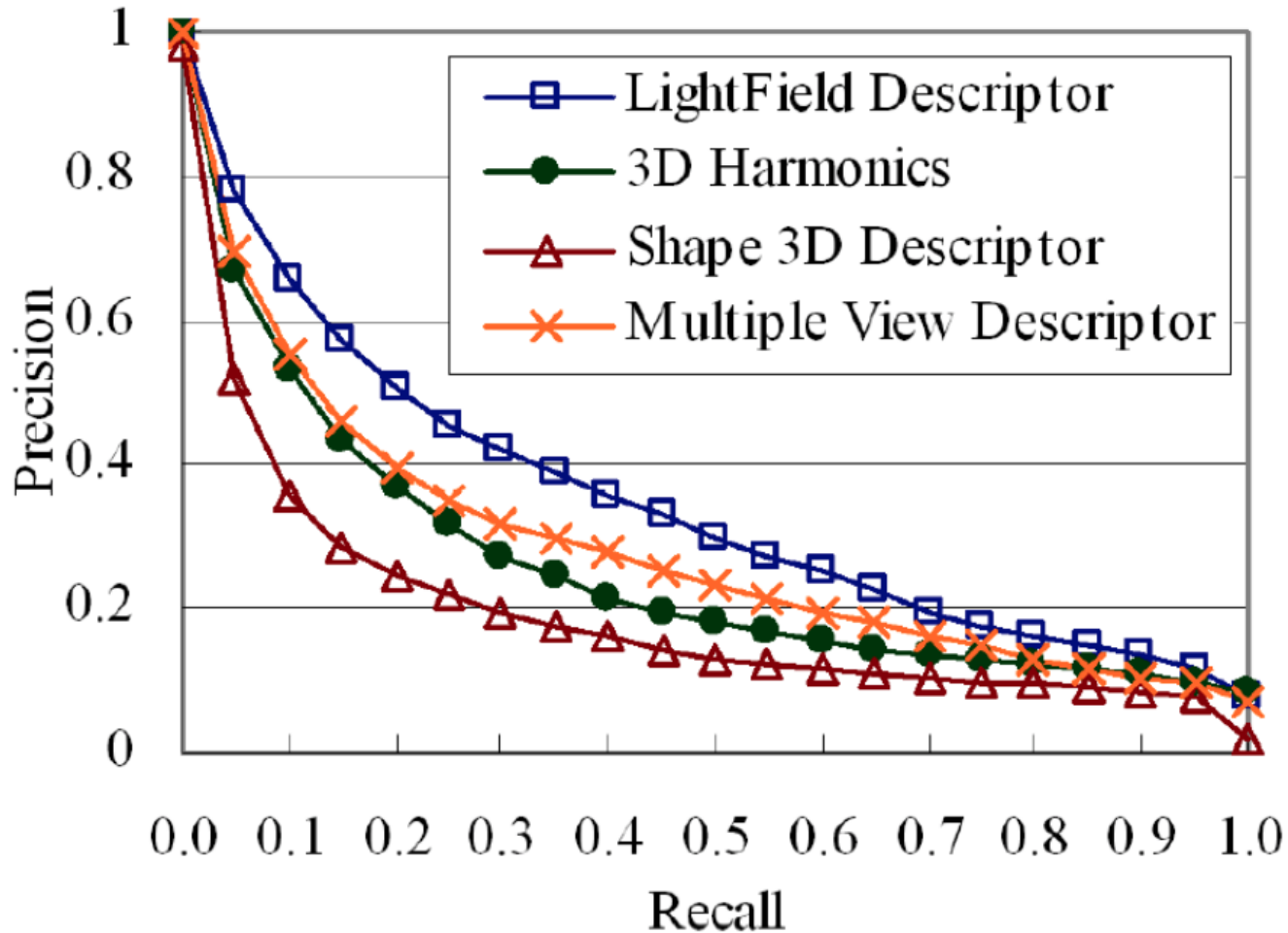
Labeling

LFD: a classic global descriptor

- The **Light Field Descriptor** (LFD) of a 3D shape is a set of 2D images of it, taken with a camera array
 - E.g. 20 cameras positioned at the vertices of a regular dodecahedron
 - Images rendered as silhouettes, so 10 unique views (say from a hemisphere)
 - Instead of the actual images, store their **Zernike Moments** and **Fourier Descriptors**
 - Compare shapes over all possible relative rotations of image clouds



Retrieval Results



3D Harmonics:
spectral signature of
the shape

Shape 3D Descriptor:
curvature histograms

**Multiple View
Descriptor:** align
shapes using PCA,
compare views along
principal axes

Test database: 1833 shapes, with 549 shapes classified into 47 functional categories, the remaining shapes classified as "miscellaneous"

What if we use better image descriptors?

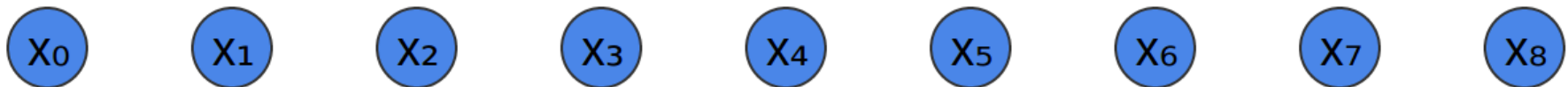
- ZMD/FD are ok, but hardly the state of the art in modern computer vision (circa 2016)
- Convolutional Neural Nets (CNNs) have revolutionized image recognition tasks

Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
CNN	37.5%	17.0%

In 2012, the error rate in the ImageNet visual recognition challenge was halved by a deep CNN (gains are typically incremental). There are 1000 categories: the baseline of random guessing would have a 99.9% error.

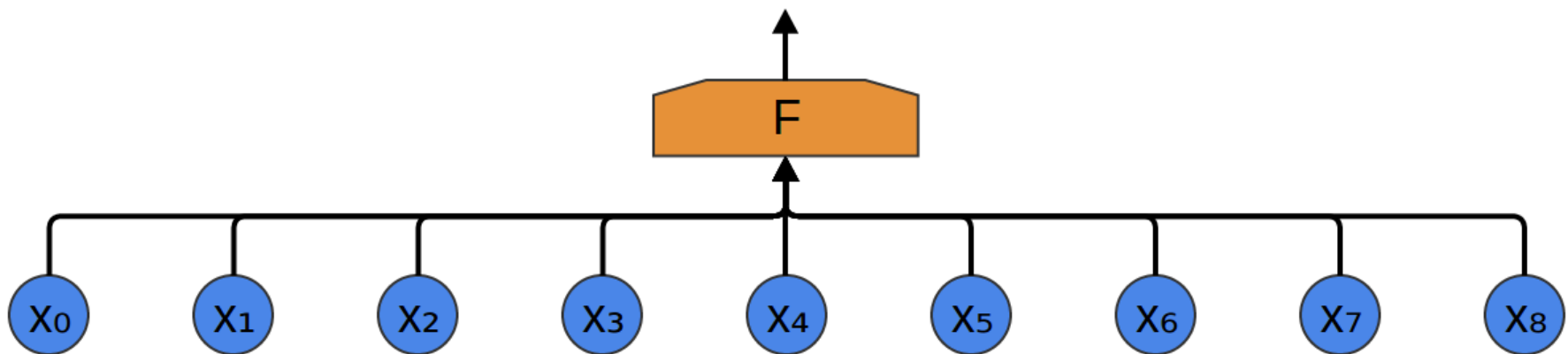
What is a Convolutional Neural Network?

- Imagine we have a set of N samples from some signal
- We want to produce a prediction, e.g. whether the signal represents a human voice, or a picture of a cat, or a depth image of a building



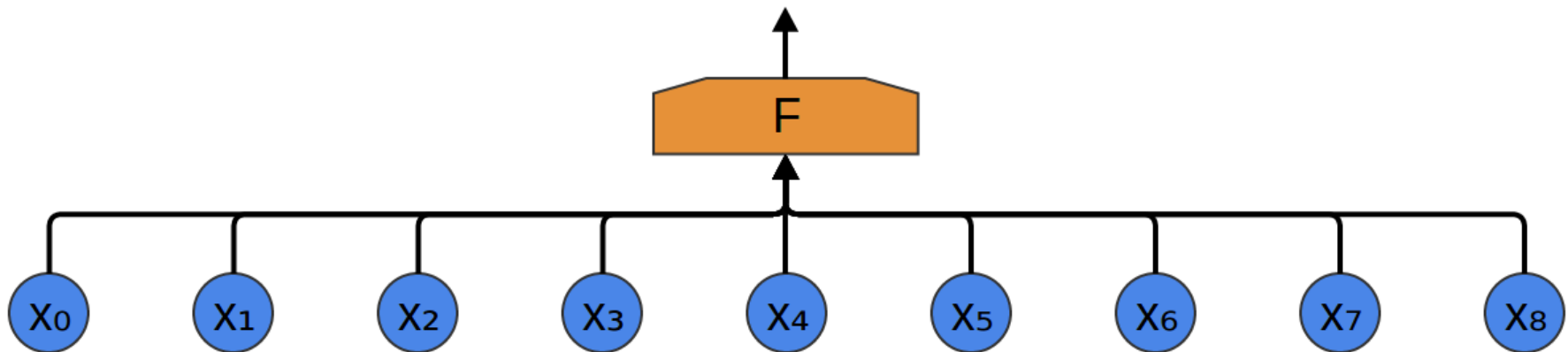
What is a Convolutional Neural Network?

- We can compute the probability as a function F of these values
 - In a **fully-connected** network, the function takes in all the inputs at once, e.g. as $g(\mathbf{w} \cdot \mathbf{x})$, where \mathbf{w} is a weight vector and g is some nonlinear transformation such as a sigmoid function



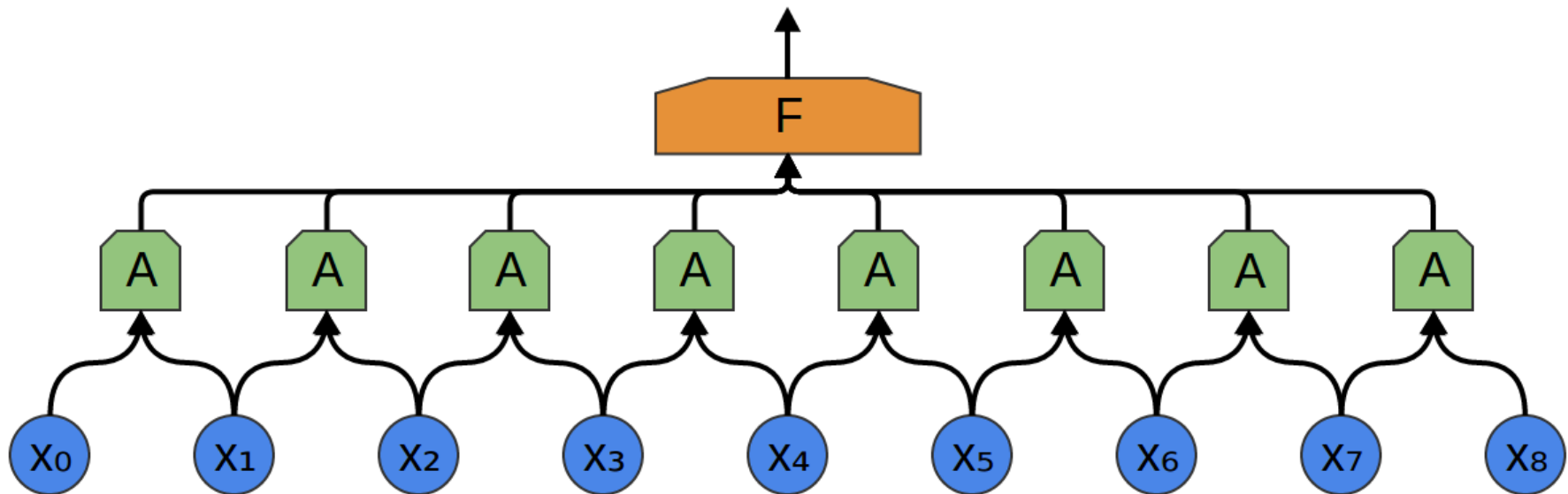
What is a Convolutional Neural Network?

- Fully-connected networks have some drawbacks
 - The function is **very high-dimensional** (all inputs processed at once)
 - **No complex relationships** between inputs are modeled (just a dot product)
 - Local information is **not captured in a “translation-invariant” way** (a feature of the signal at the left end of the sequence must be learned independently of the same feature occurring at the right end)



What is a Convolutional Neural Network?

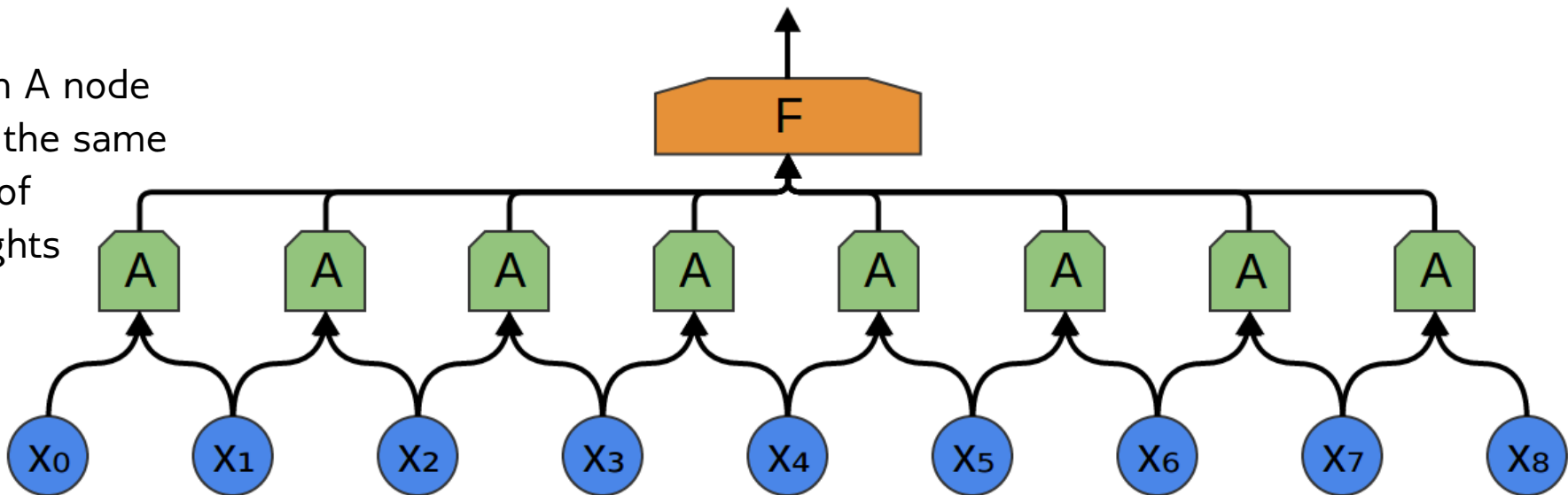
- **Solution:** a **convolutional layer**
- A filter (again, a dot product followed by a nonlinear transformation) is applied on local neighborhoods of the signal



What is a Convolutional Neural Network?

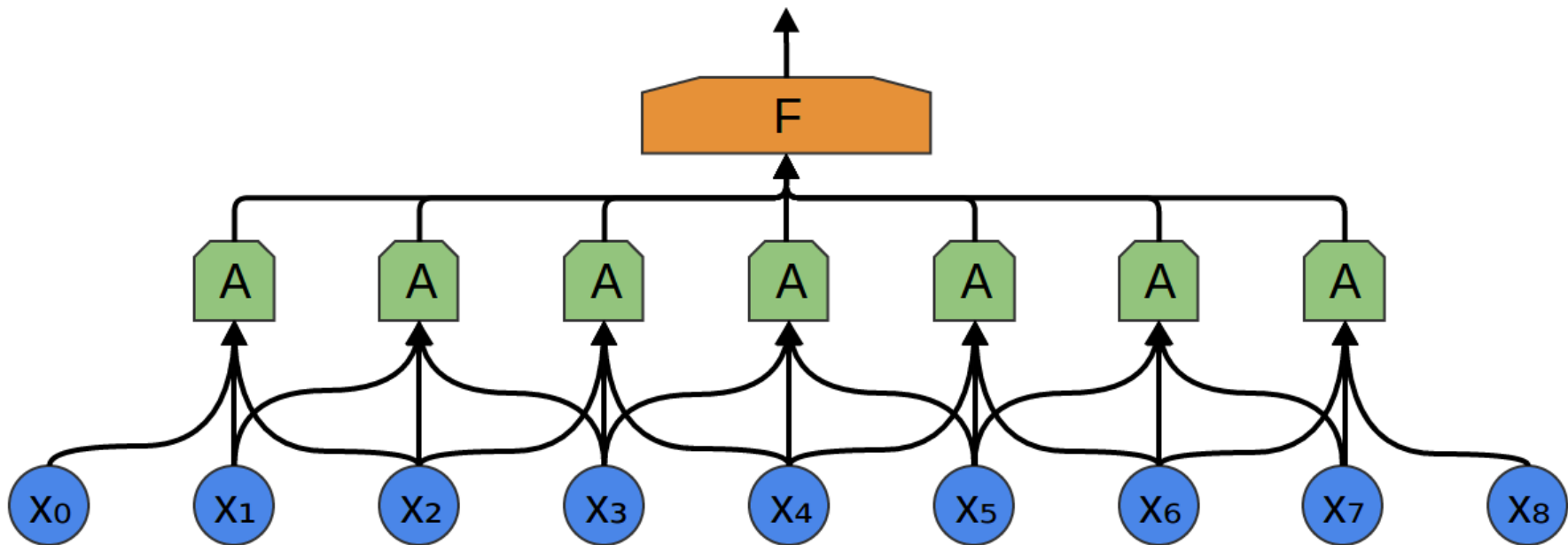
- All filters **share the same weights!**
 - Dramatically reduces number of parameters of the network
- The final output is a function of the filter responses

Each A node
has the same
set of
weights



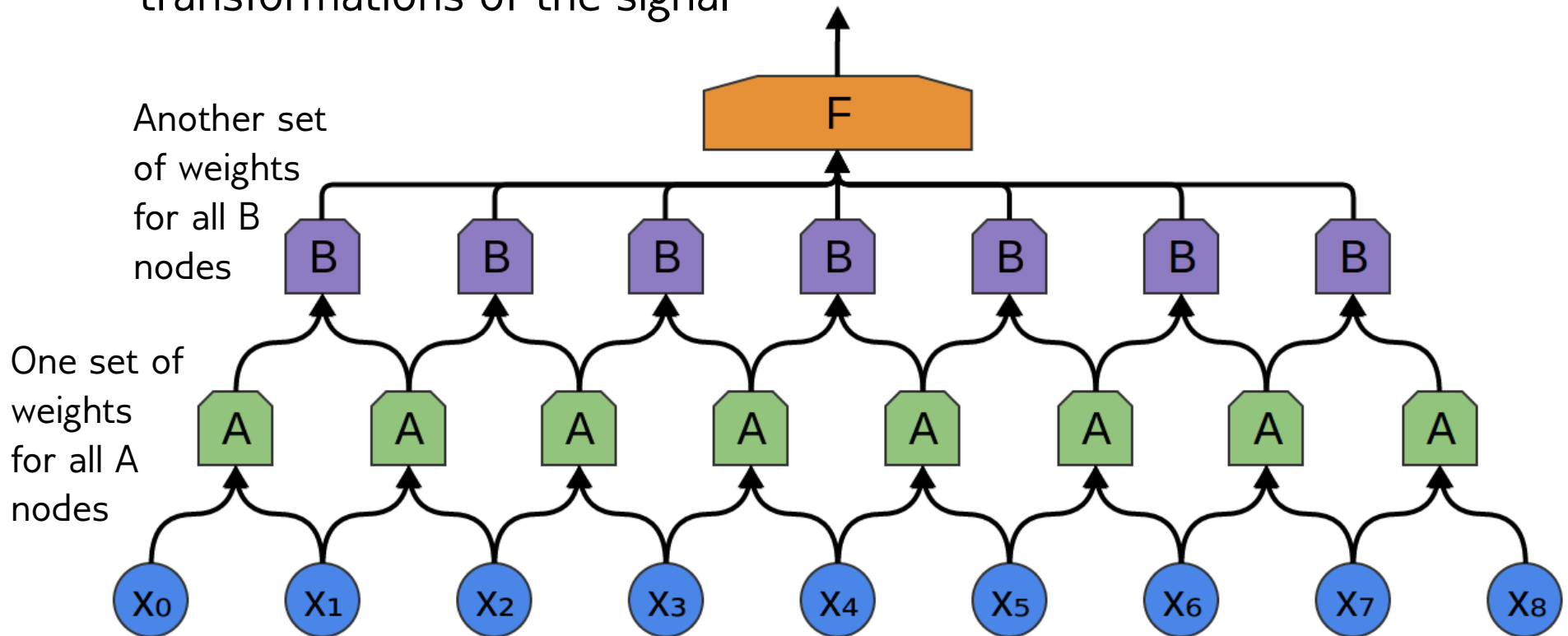
What is a Convolutional Neural Network?

- We can make the neighborhoods larger, to capture broader local features



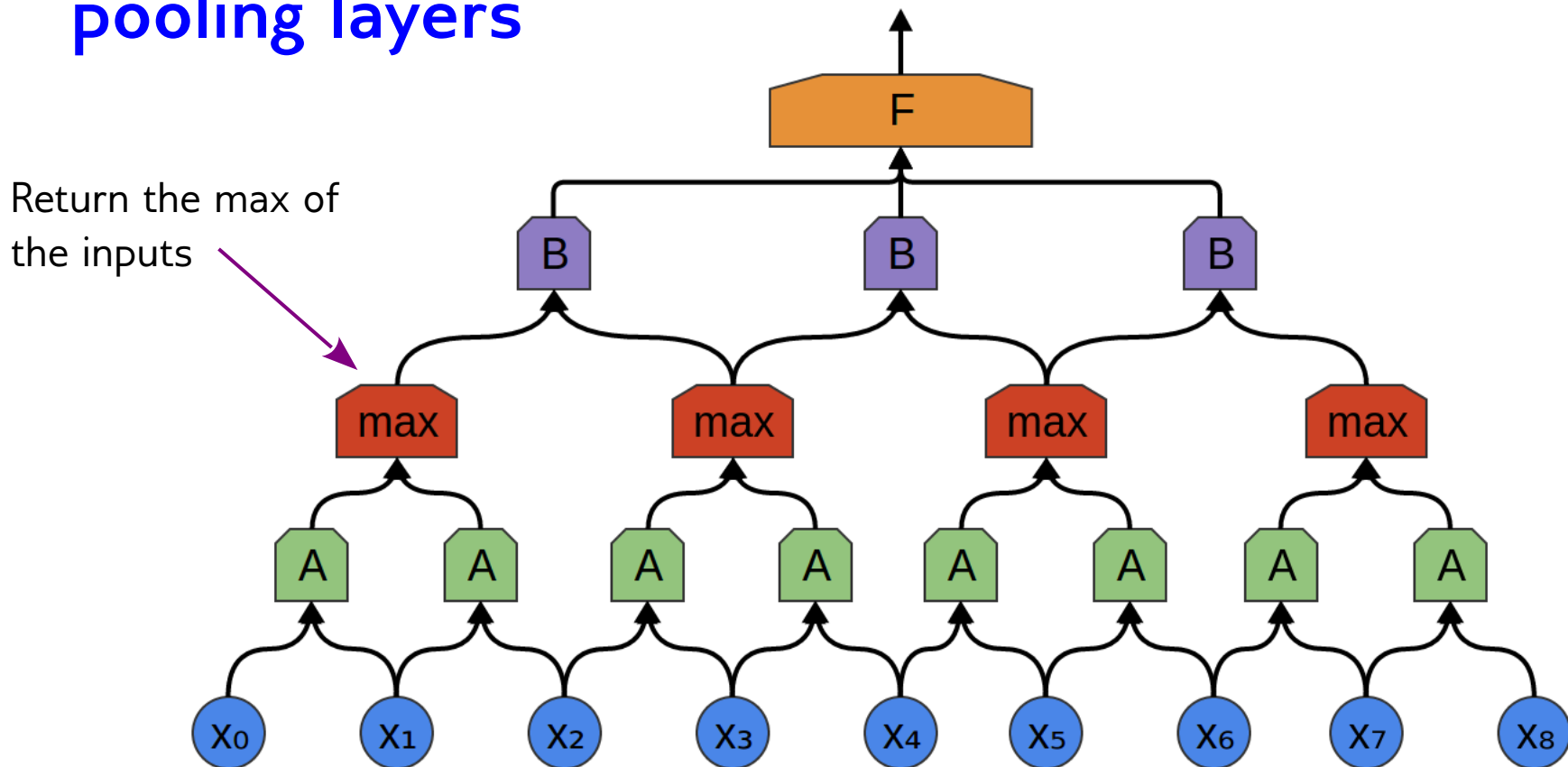
What is a Convolutional Neural Network?

- Convolutional layers are **composable**: they can be stacked with each layer providing inputs for the next layer
 - Higher layers can capture more abstract features since they effectively cover larger neighborhoods, and combine multiple different nonlinear transformations of the signal



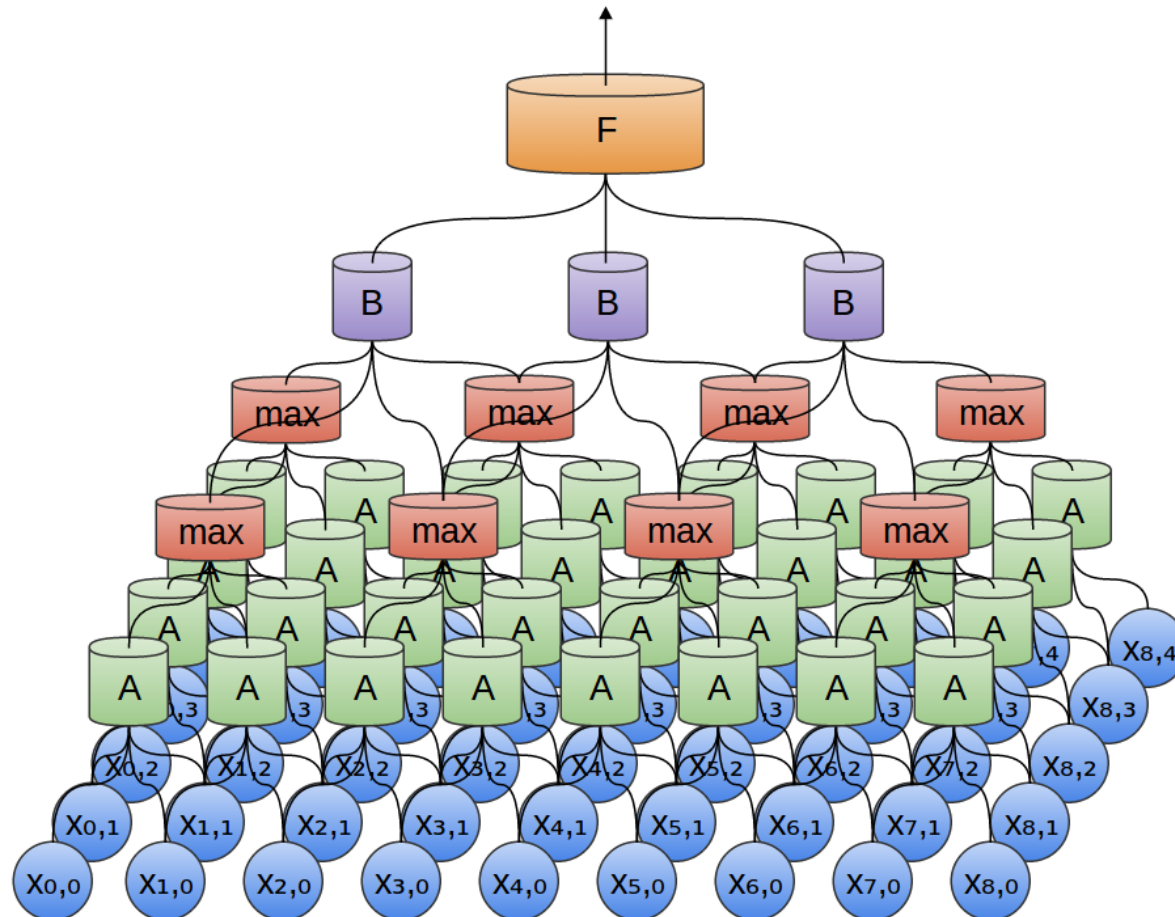
What is a Convolutional Neural Network?

- To make the network robust to small translations in detected features, and to reduce the amount of redundant data fed into higher layers, we introduce **pooling layers**



What is a Convolutional Neural Network?

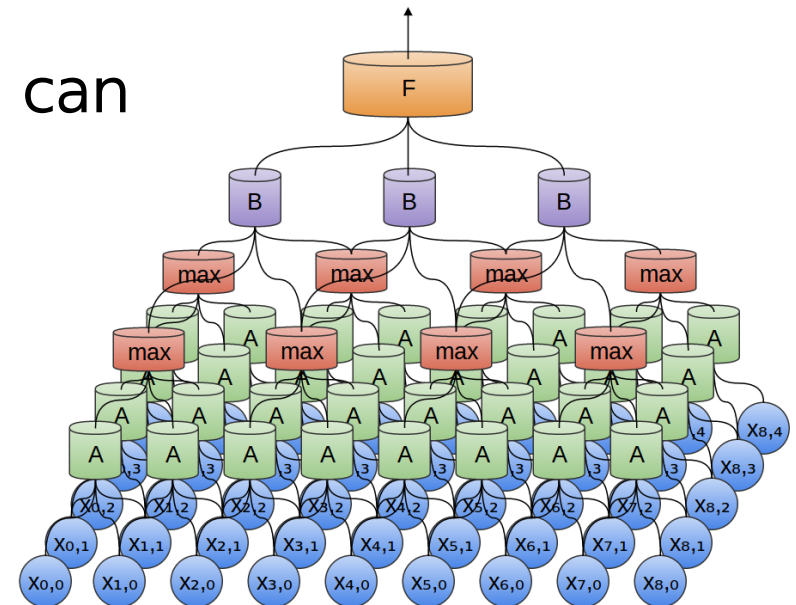
- The signal can be 2D or 3D: the filters are now also 2D/3D, but it's all essentially the same



What is a Convolutional Neural Network?

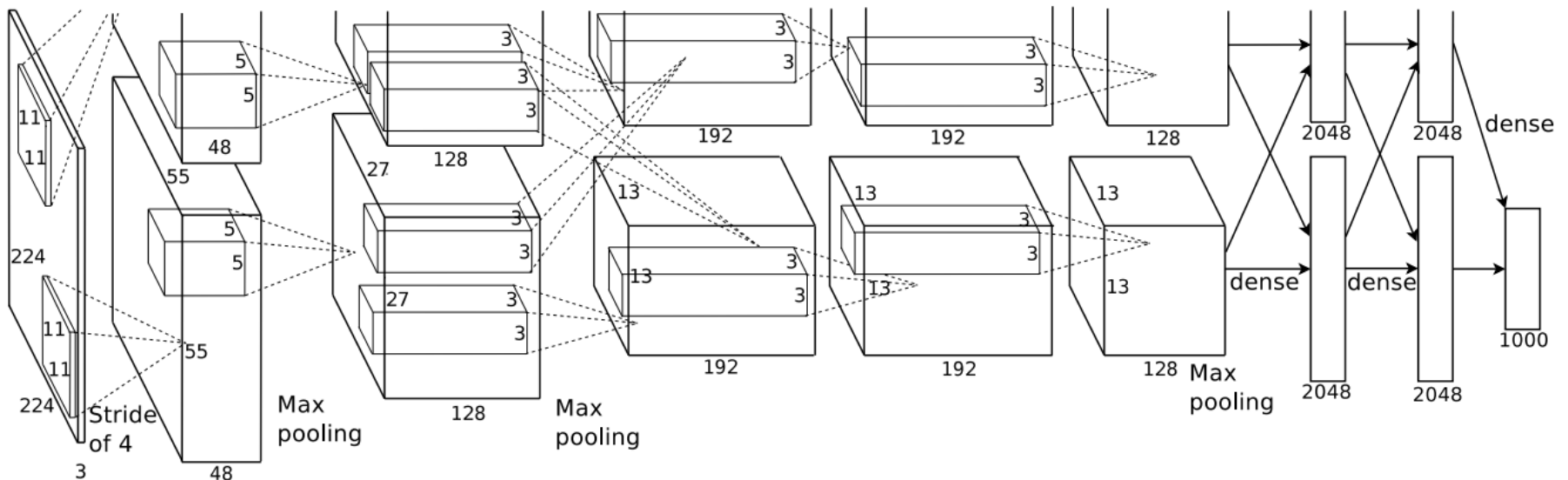
- The function computed by this gigantic model is **differentiable*** w.r.t. the weights
 - Given training data and a **loss function** measuring the deviation between predicted and actual values, we can optimize the weights by gradient descent
 - The gradient of the loss function can be found efficiently by a method called **back-propagation**

* nearly everywhere

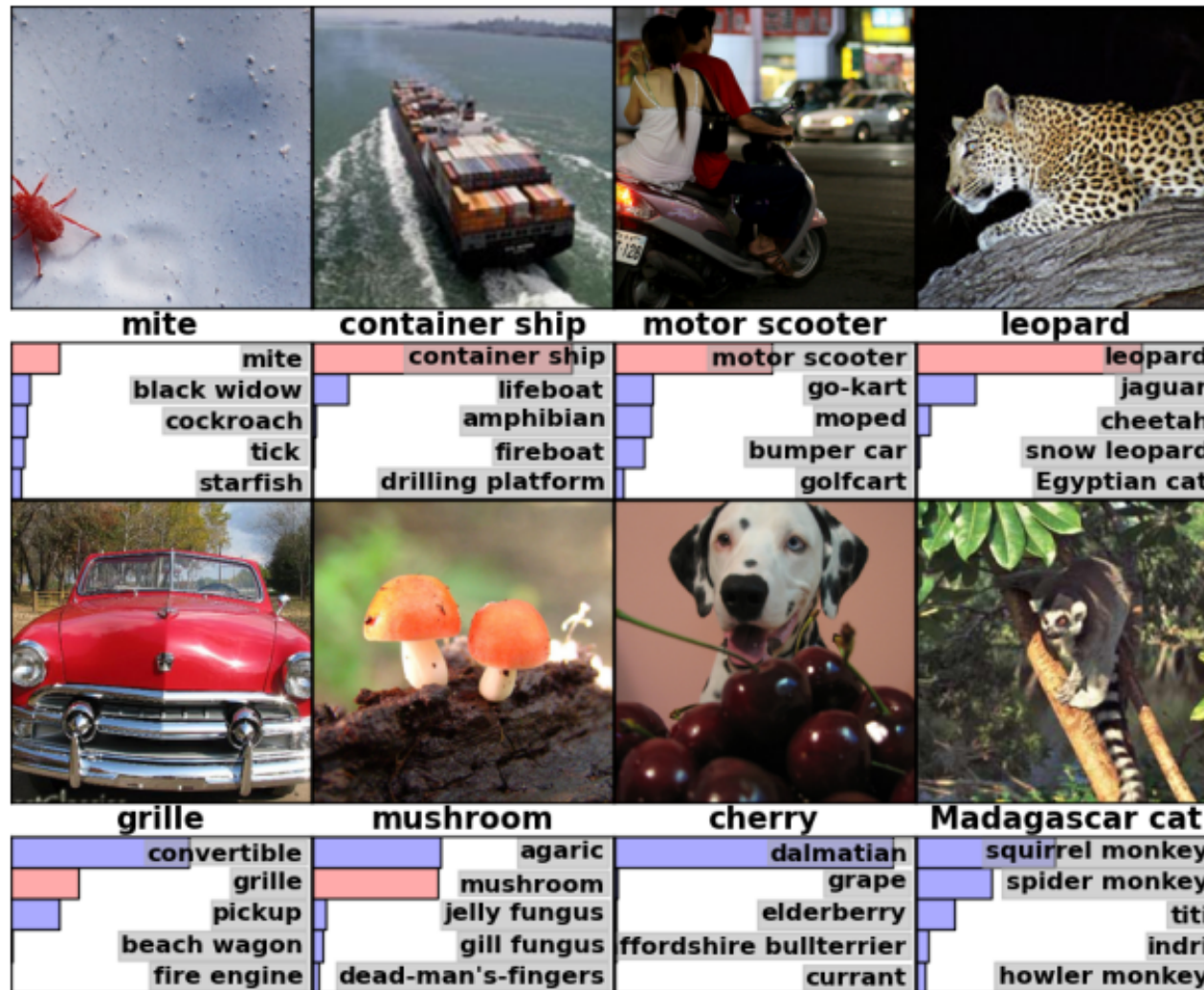


A real-world CNN

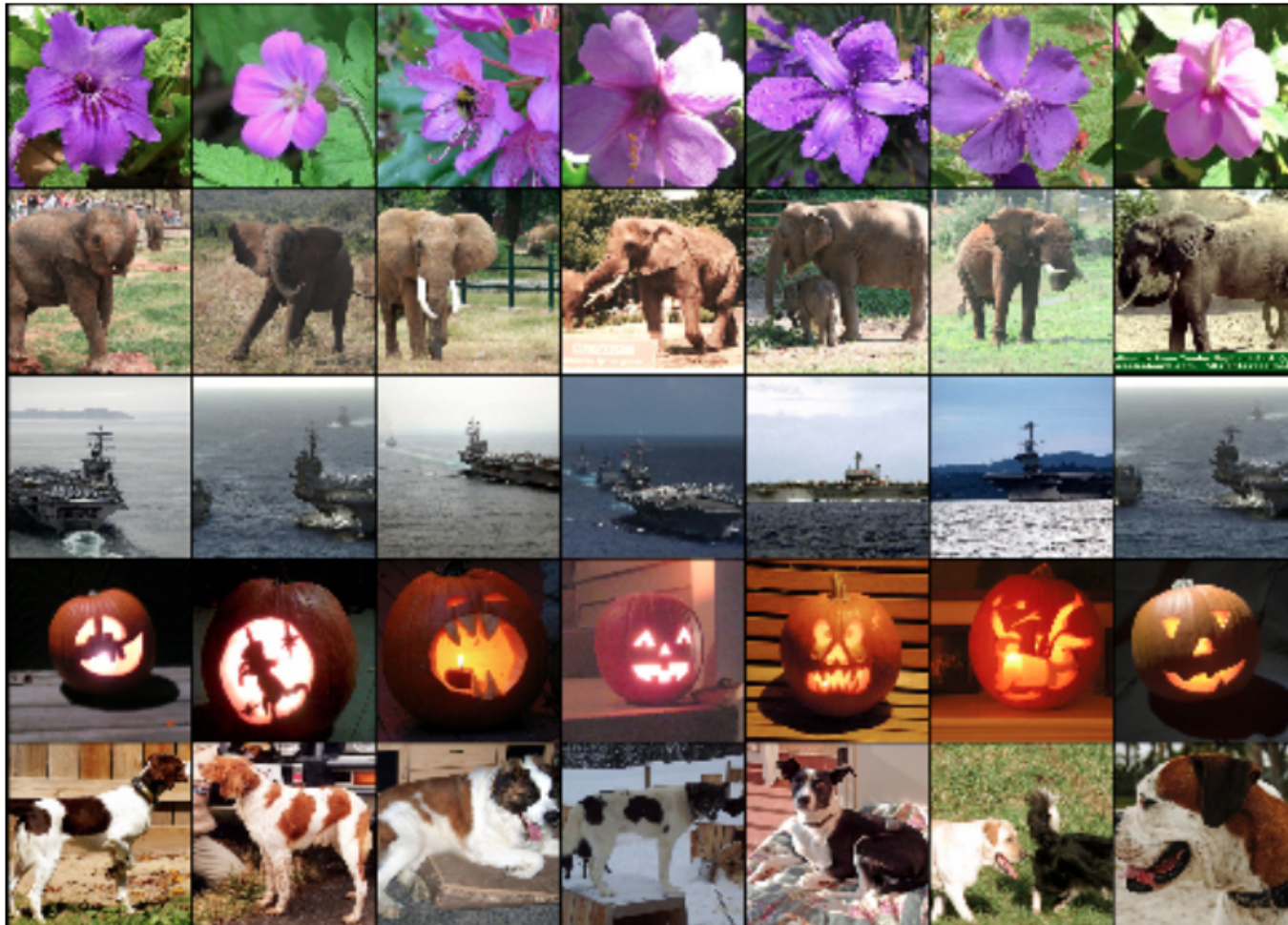
- 5 convolutional layers, 3 max-pooling layers, 3 fully-connected layers
- ~60 million parameters (despite the weight sharing!)



Using the CNN for classification



Using the CNN for retrieval



The descriptor is the vector of neuron activations in the second last layer

Query

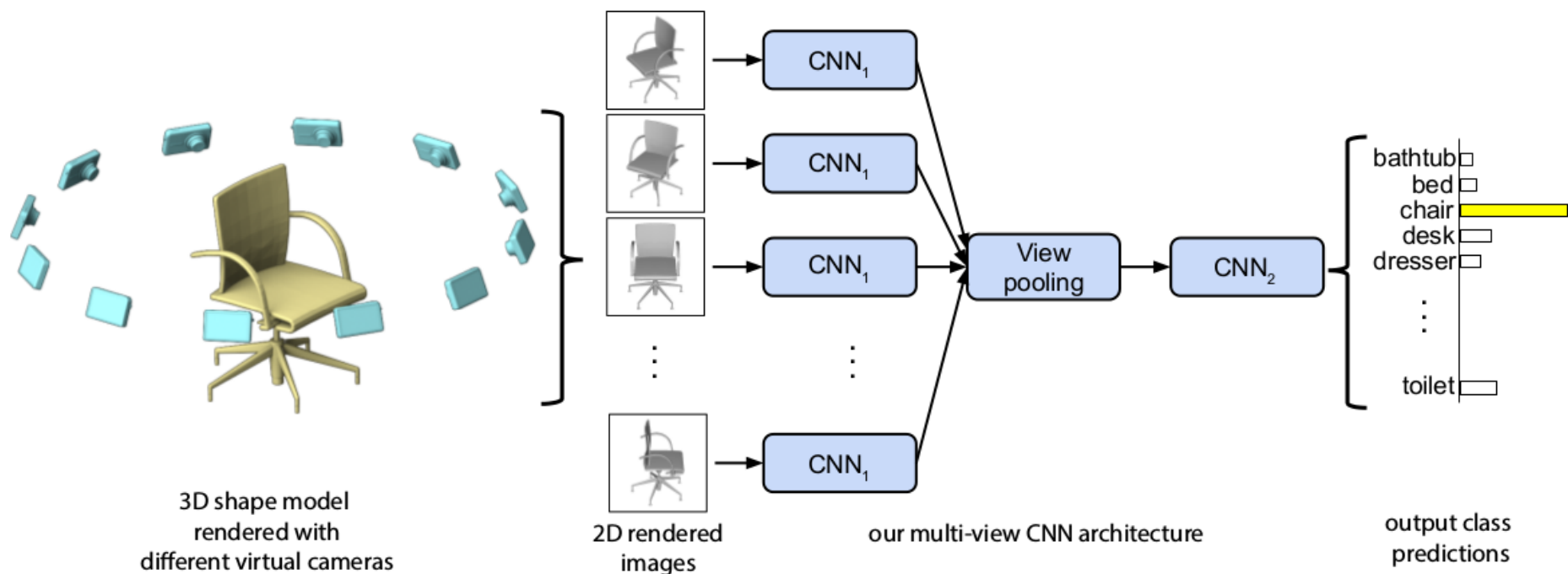
Top 6 results

Image CNN for 3D shapes

- Let's take a CNN trained on a (huge) image database, and use it to analyze views of 3D shapes
 - **Render** a 3D shape from an arbitrary viewpoint
 - Pass it through the **pre-trained CNN** and take the neuron activations in the second-last layer as the descriptor
 - For more accuracy, **fine-tune** the network on a training set of rendered shapes before testing
- Just this alone, with a single view (from an unknown direction) of the shape, bumps up the mAP retrieval accuracy (area under PR curve) on a 40-class, 12K-shape collection from 40.9% (LFD) to **61.7%**.
 - An LFD-like approach with 12 views/shape further improves to **62.8%**

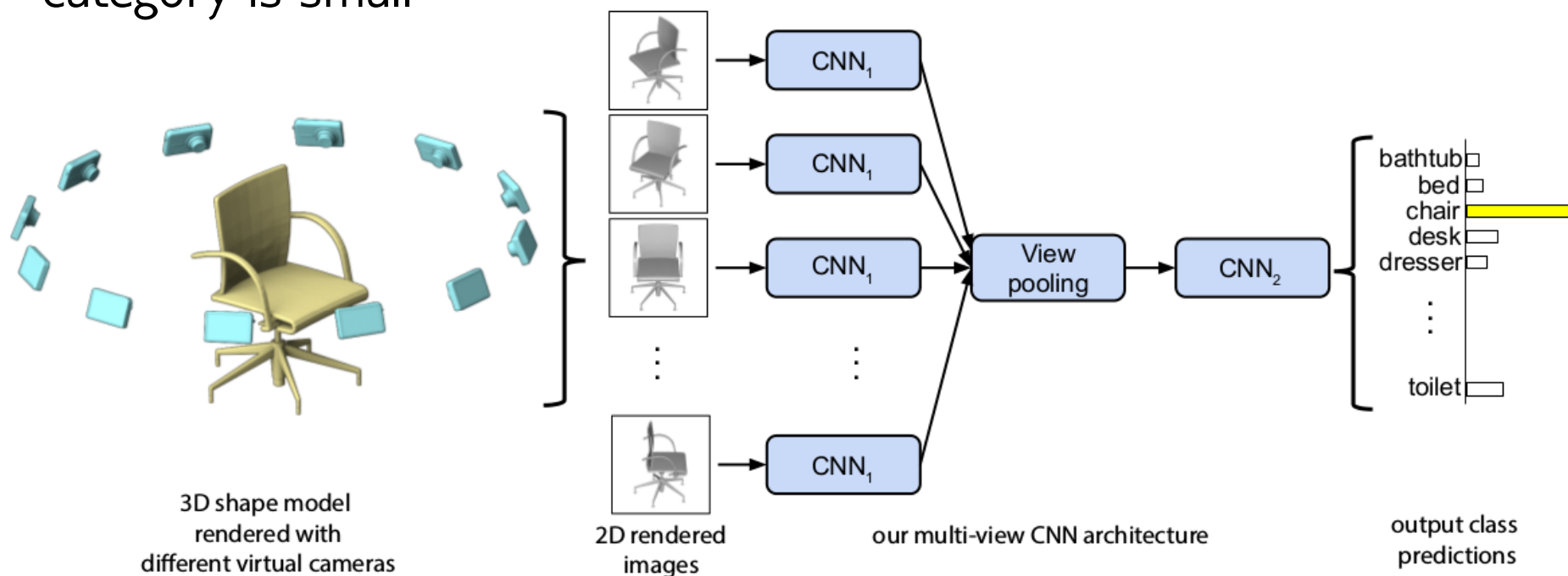
Combining Views

- A smarter way to aggregate information from multiple views
 - Take the output signal of the last convolutional layer of the base network (CNN_1) from each view, and combine them, element-by-element, using a max-pooling operation
 - Pass this **view-pooled** signal through the rest of the network (CNN_2)

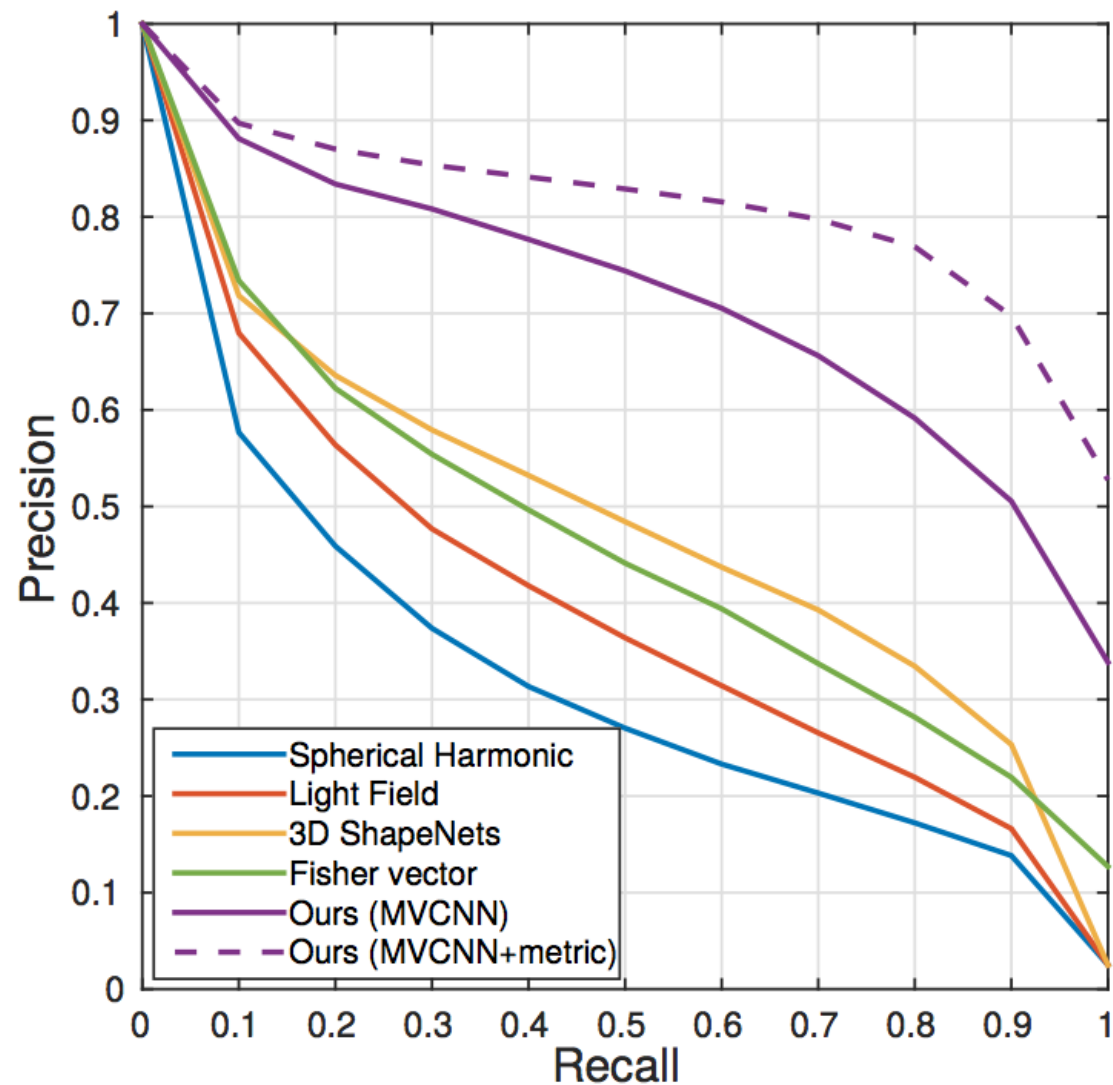


Combining Views

- The view-pooled CNN can still be trained (in exactly the same way) using back-propagation and gradient descent
- For retrieval, the descriptor from the second-last layer can be further tuned by learning a Mahalanobis metric (a projection of the descriptors) where the distance between shapes of the same training category is small

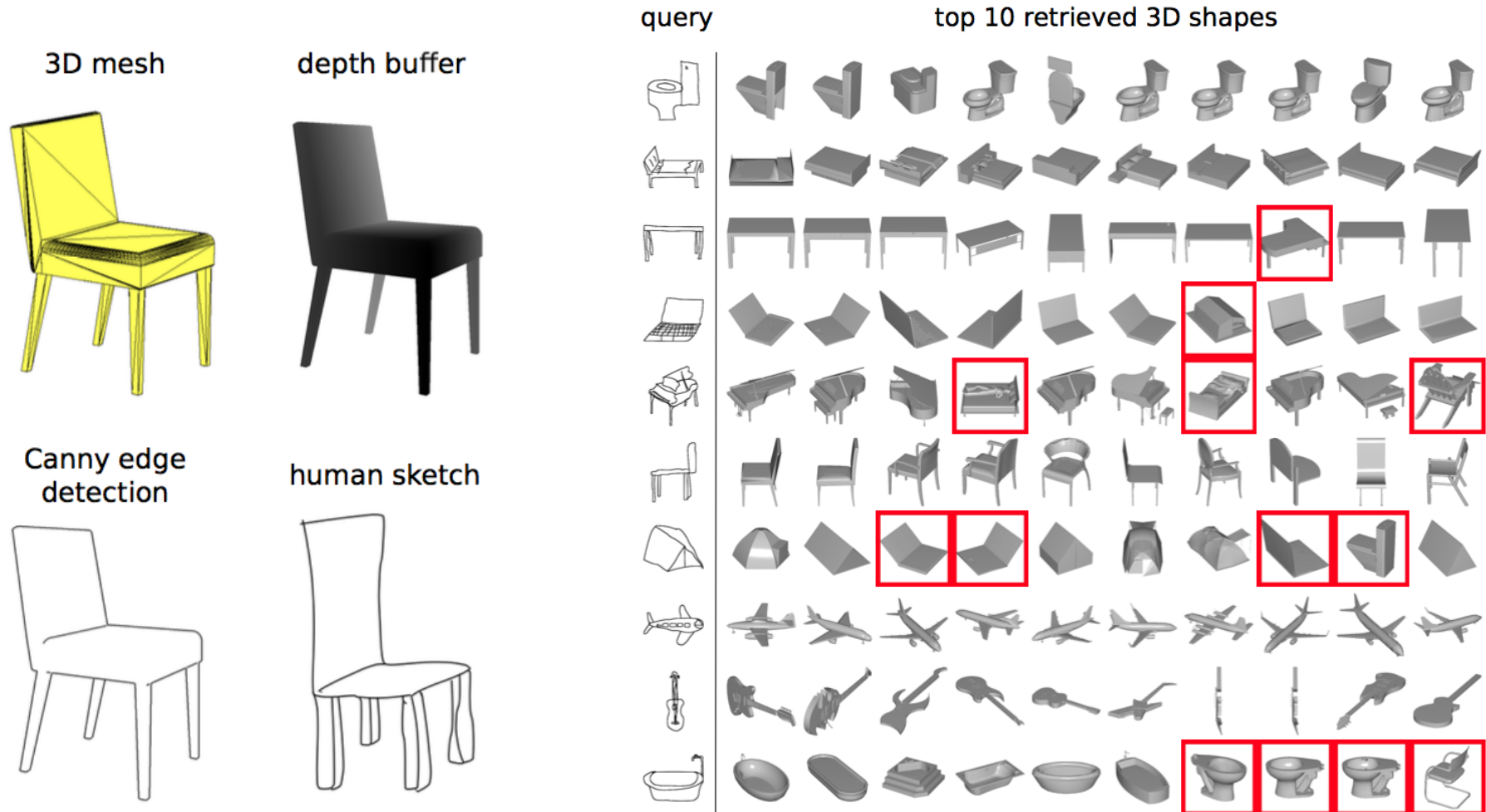


How well does this work?



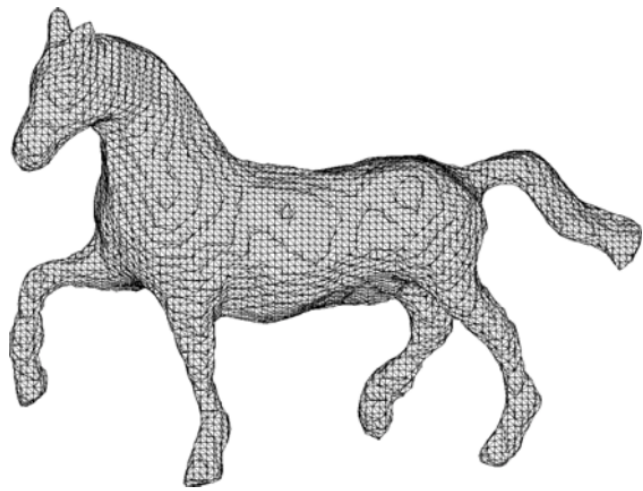
A side benefit of view-based representations

- The MVCNN can be fine-tuned to retrieve 3D models based on hand-drawn 2D sketches



Global descriptors enable retrieval.
Let's look at an application enabled by
good *local* descriptors.

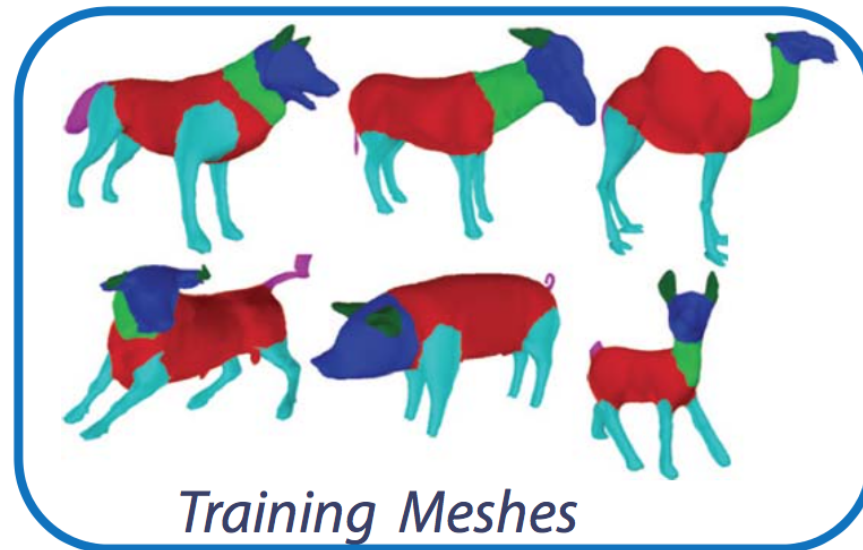
Shape Segmentation and Labeling








Input Mesh



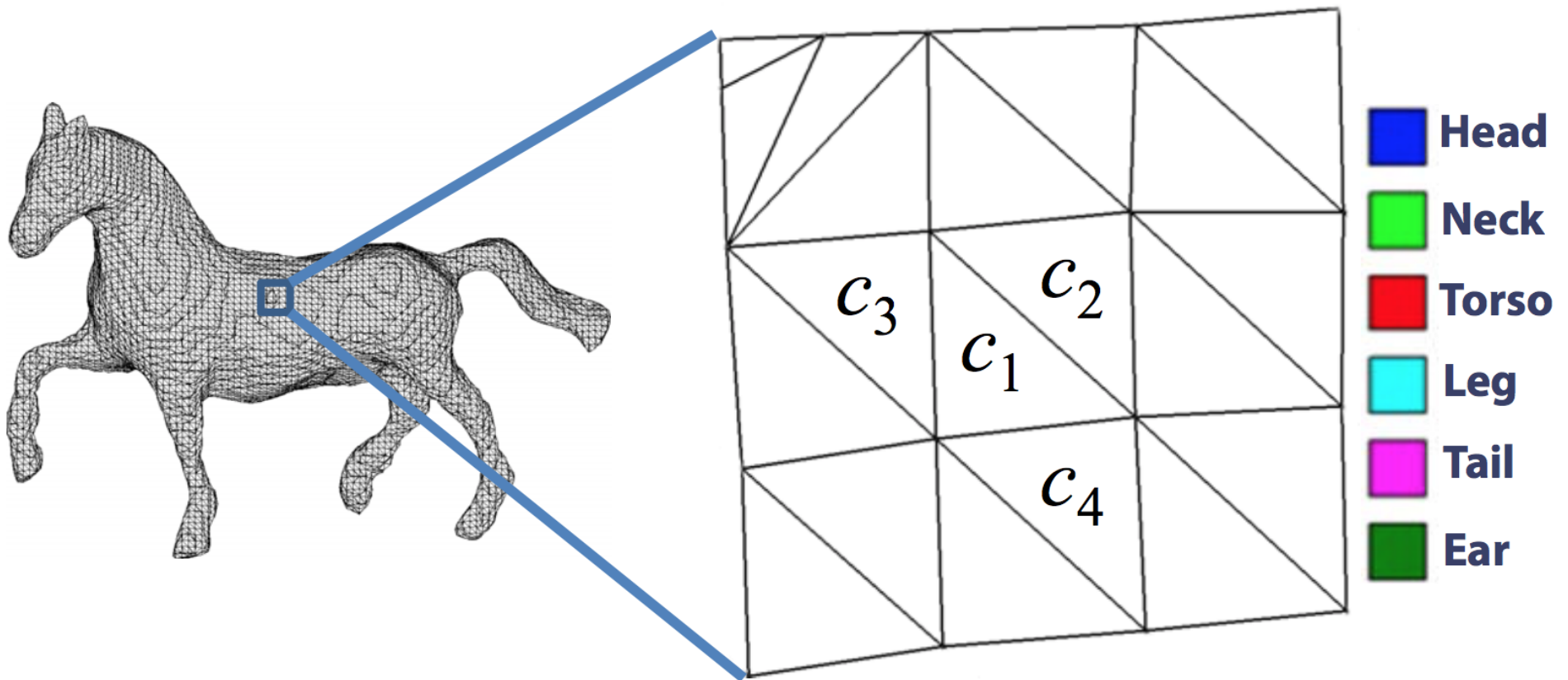
Labeled Mesh



Training Meshes

-  **Head**
-  **Neck**
-  **Torso**
-  **Leg**
-  **Tail**
-  **Ear**

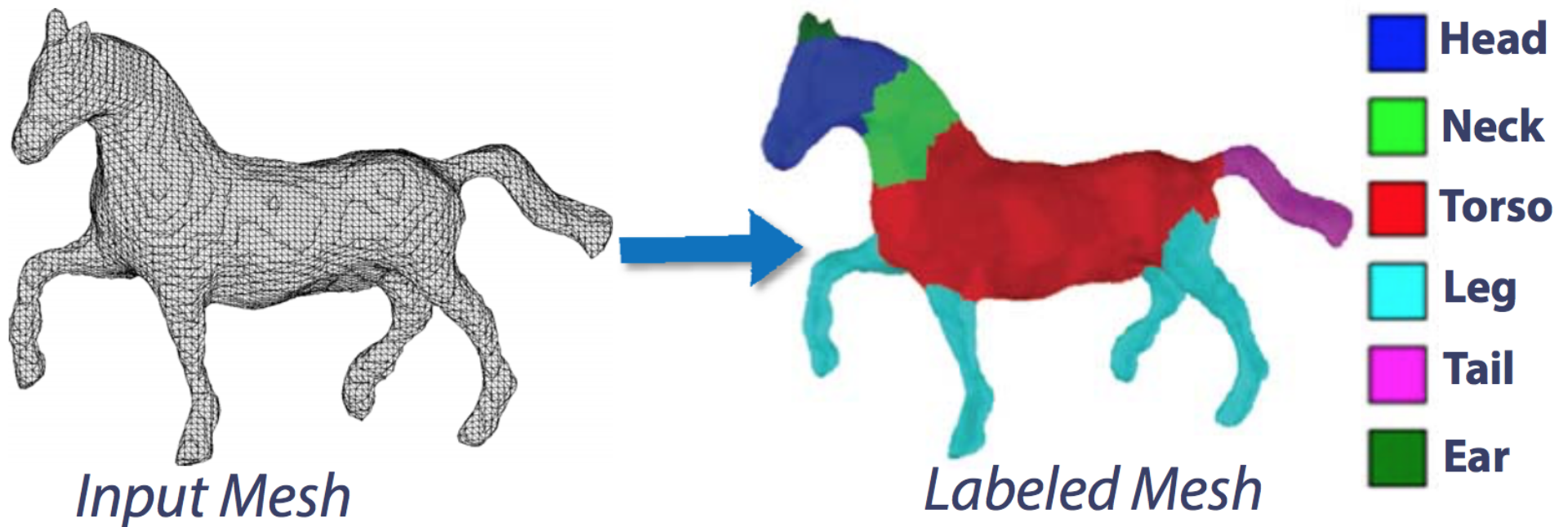
Shape Segmentation and Labeling



$$c_1, c_2, c_3 \in C$$

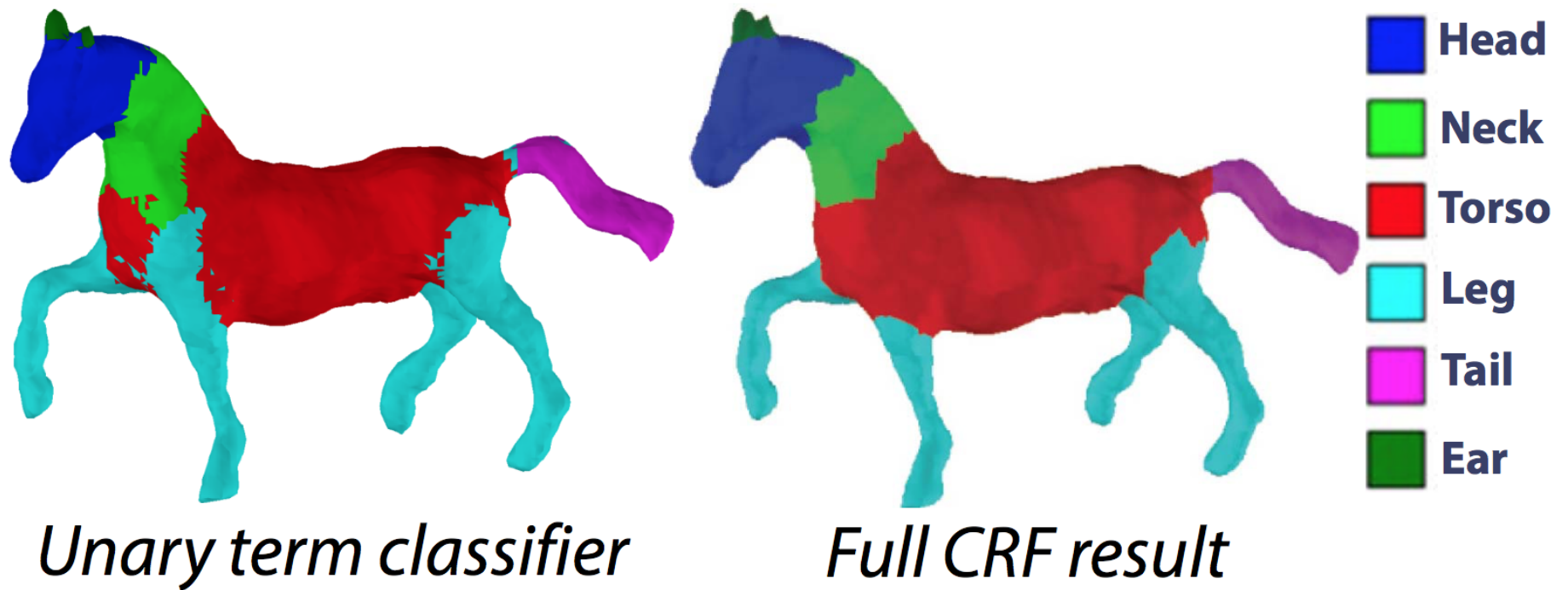
$$C = \{ \textit{head}, \textit{neck}, \textit{torso}, \textit{leg}, \textit{tail}, \textit{ear} \}$$

Conditional Random Field for Segmentation and Labeling



$$c^* = \arg \min_{\mathbf{c}} \left\{ \underbrace{\sum_i \alpha_i E_1(c_i; \mathbf{x}_i)}_{\text{Unary term}} + \underbrace{\sum_{i,j} l_{ij} E_2(c_i, c_j; \mathbf{y}_{ij})}_{\text{Pairwise term}} \right\}$$

Effect of the pairwise term



View-based local descriptors?

- CNNs can also yield local descriptors
- If multi-view CNNs dramatically improve retrieval accuracy, can they also improve segmentation accuracy?
- The answer appears to be yes (more details coming soon!)

“High-Level” Geometric Analysis



- What **type** of object is this?
- How can we **generate** more objects like this?
- What **attributes** does it have?
- What **functions** does it serve?

Outline

- Learning shape structure
 - **Probabilistic models** of shape

Outline

- Learning shape structure
 - **Probabilistic models** of shape
- Learning shape semantics
 - Semantic **attributes** (*scary, artistic, ...*)
 - Mechanical **function** (*this airplane should fly...*)
 - Human **interaction** (*sit comfortably in a chair...*)

What is the role of data?



Google/Trimble 3D Warehouse (~millions of downloadable models)

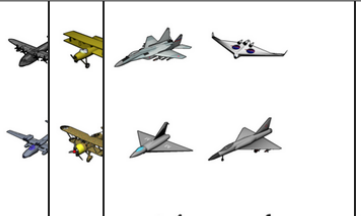


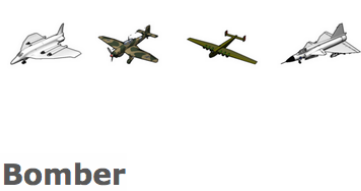







What is the role of data?

SHAPENET Search Options About Download Publications

Choose a taxonomy: ShapeNetCore

- bathtub,bathing tub,bath,tub(0,85)
- bed(13,233)
- bench(5,1813)
- bicycle,bike,wheel,cycle(0,59)
- birdhouse(0,73)
- bookshelf(0,452)
- bottle(6,498)
- bowl(1,186)
- bus,autobus,coach,charabanc,dol
- cabinet(9,1571)
- camera,photographic camera(4,11)
- can,tin,tin can(2,108)
- cap(4,56)
- car,auto,automobile,machine,motr
- chair(23,6778)
- clock(3,651)
- computer keyboard,keypad(0,65)
- dishwasher,dish washer,dishwashi
- display,video display(5,1093)
- earphone,earpiece,headphone,ph
- faucet,spigot(2,744)
- file,file cabinet,filing cabinet(1,298)

Synset Models **TreeMap** Stats Measures

 See Bip Delta wing	 Propeller plane	 Straight wing
 Bomber		
 Swept wing	 Fighter	 Jet
 Transport airplane	 Airliner	

← → 🔍

<http://shapenet.cs.stanford.edu>

What is the role of data?

- **Reuse** (of existing components)
- **Training** (of computational models)
- **Inspiration** (for new designs)

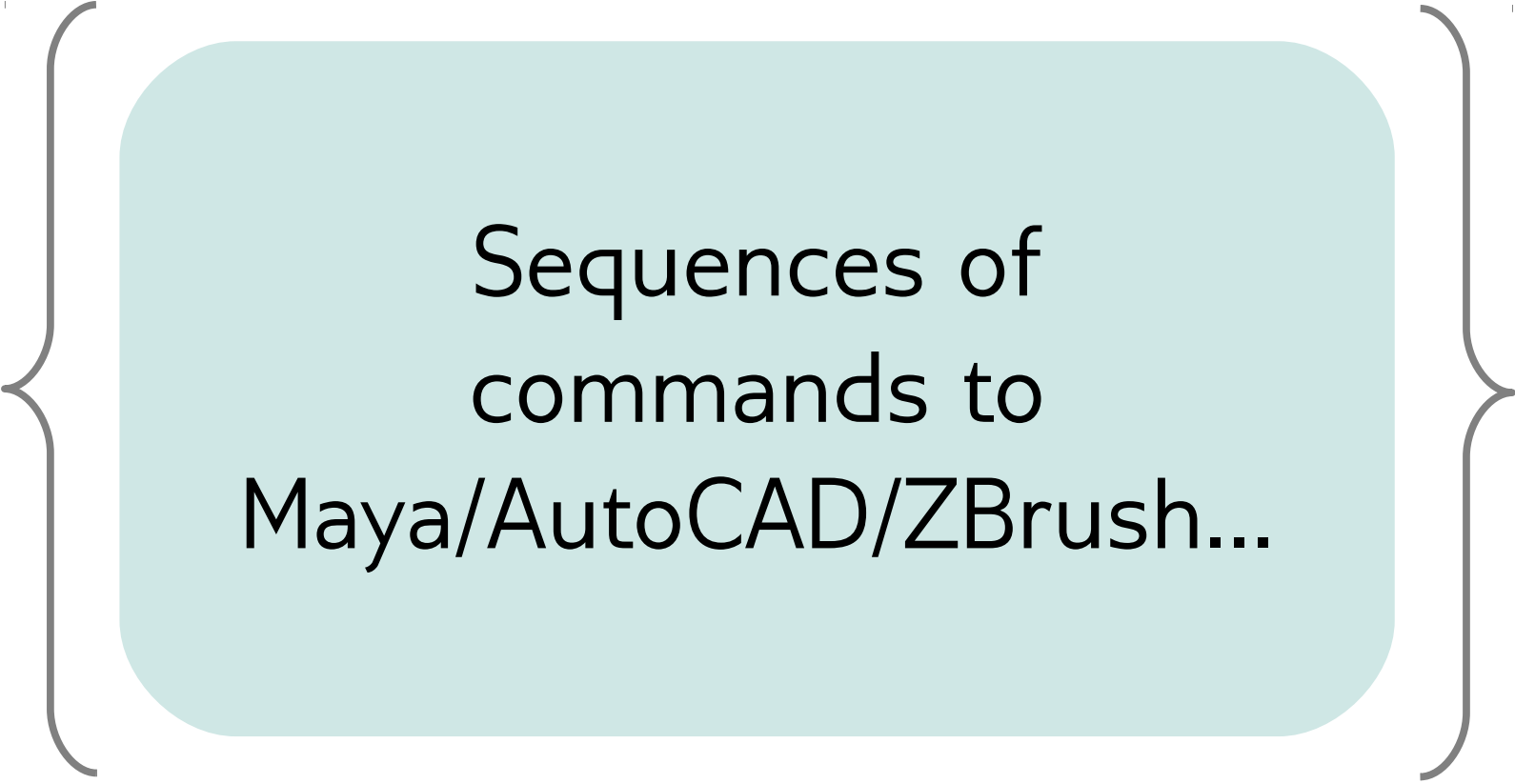
Outline

- Learning shape structure
 - **Probabilistic models** of shape
- Learning shape semantics
 - Semantic **attributes** (*scary, artistic, ...*)
 - Mechanical **function** (*this airplane should fly...*)
 - Human **interaction** (*sit comfortably in a chair...*)

Shape spaces should be...

- **General**
 - Topological/geometric/configurational variety
- **Probabilistic**
 - Some shapes are more plausible than others
- **Generative**
 - Can be used to produce new shapes
- **Meaningfully Parametrized**
 - Design intent readily maps to “suitable” shapes

Shape Space: Maya



Sequences of
commands to
Maya/AutoCAD/ZBrush...

Generality: **High**

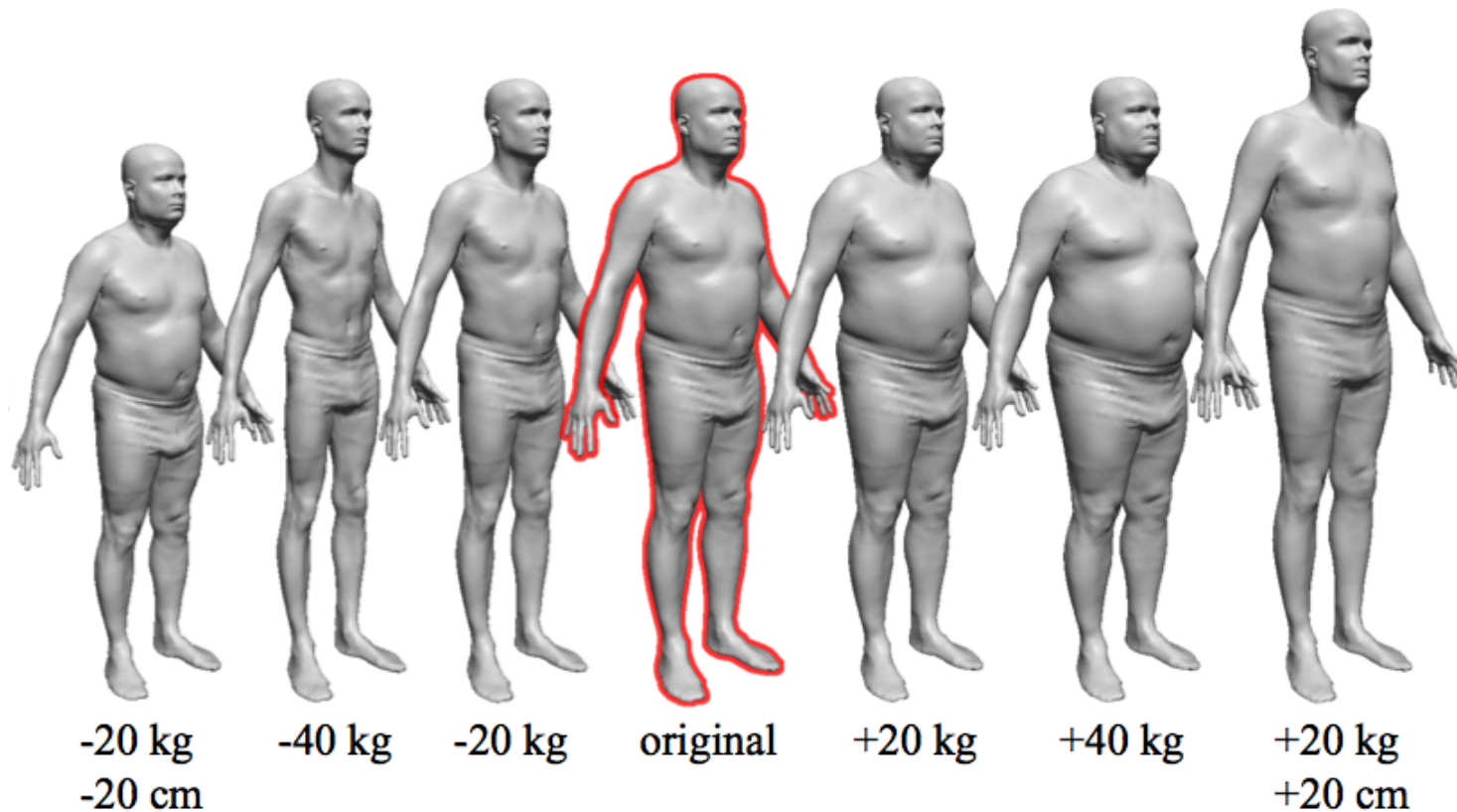
Probabilistic: **No**

Meaningful parametrization: **No**

Data-driven: **No**

Shape Space: Deformable Template

(one topology, plus parameters for body type)

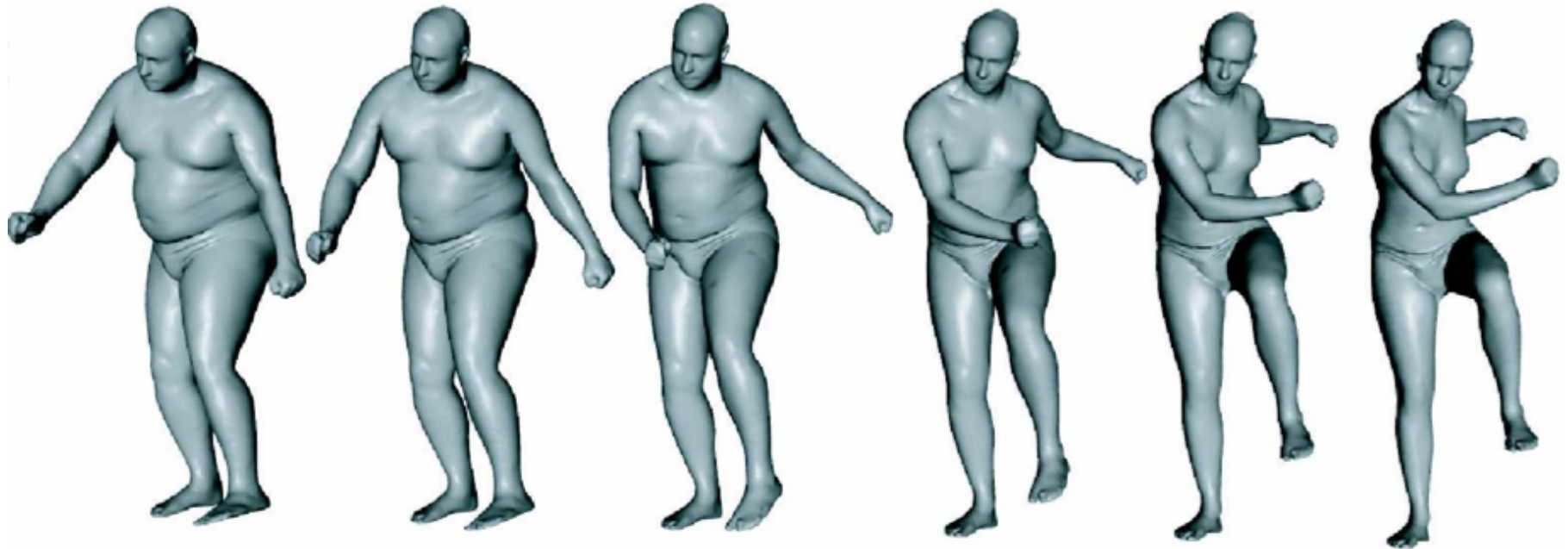


Generality: **Low**
Probabilistic: **Yes**

Meaningful parametrization: **Moderate**
Data-driven: **Yes**

Shape Space: Deformable Template

(one topology, plus parameters for both body type and pose)



Generality: **Low-ish**
Probabilistic: **Yes**

Meaningful parametrization: **Moderate**
Data-driven: **Yes**

Shape Space: Parametrized Procedure

(fixed set of parameters)



Generality: **Moderate**

Probabilistic: **No**

Meaningful parametrization: **Yes**

Data-driven: **No**

Shape Space: Probabilistic Procedure

(probability distribution on parameters)



Generality: **Moderate**

Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Partially**

Shape Space: Probabilistic Grammar

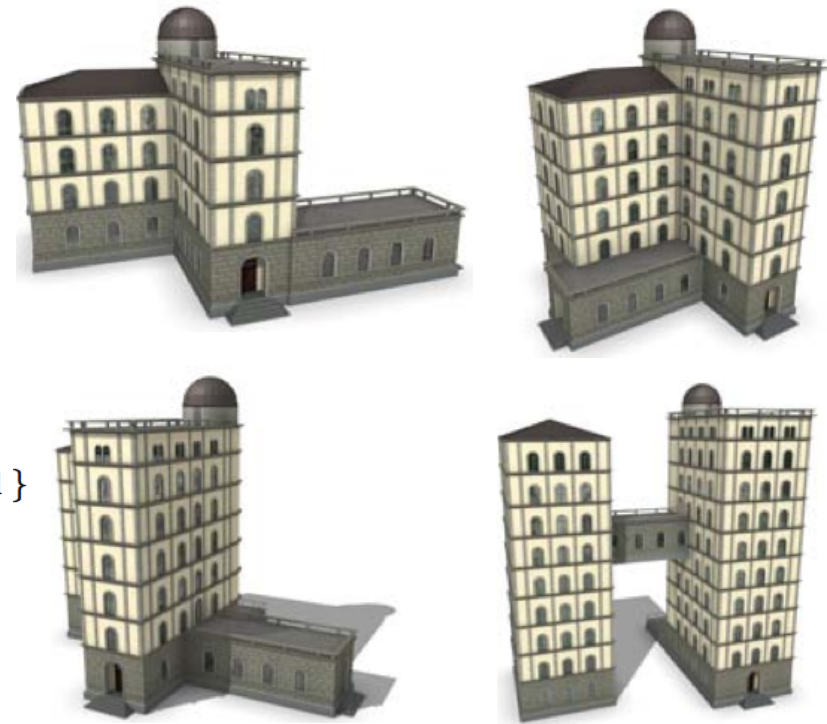
(hierarchical generation)

PRIORITY 1:

- 1: footprint \leadsto S(1r, *building_height*, 1r) facades
 T(0, *building_height*, 0) Roof("hipped", *roof_angle*) { roof }

PRIORITY 2:

- 2: facades \leadsto Comp("sidefaces") { facade }
 3: facade : Shape.visible("street")
 \leadsto Subdiv("X", 1r, *door_width**1.5) { tiles | entrance } : 0.5
 \leadsto Subdiv("X", *door_width**1.5, 1r) { entrance | tiles } : 0.5
 4: facade \leadsto tiles
 5: tiles \leadsto Repeat("X", *window_spacing*) { tile }
 6: tile \leadsto Subdiv("X", 1r, *window_width*, 1r) { wall |
 Subdiv("Y", 2r, *window_height*, 1r) { wall | window | wall } | wall }
 7: window : Scope.occ("noparent") != "none" \leadsto wall
 8: window \leadsto S(1r, 1r, *window_depth*) I("win.obj")
 9: entrance \leadsto Subdiv("X", 1r, *door_width*, 1r) { wall |
 Subdiv("Y", *door_height*, 1r) { door | wall } | wall }
 10: door \leadsto S(1r, 1r, *door_depth*) I("door.obj")
 11: wall \leadsto I("wall.obj")



Generality: **Moderate**

Probabilistic: **Yes**

Meaningful parametrization:

Data-driven:

Yes

Reuse

Shape Space: Shape Grammar

(learned from a single example)



Generality: **Moderate**

Probabilistic: **No**

Meaningful parametrization:

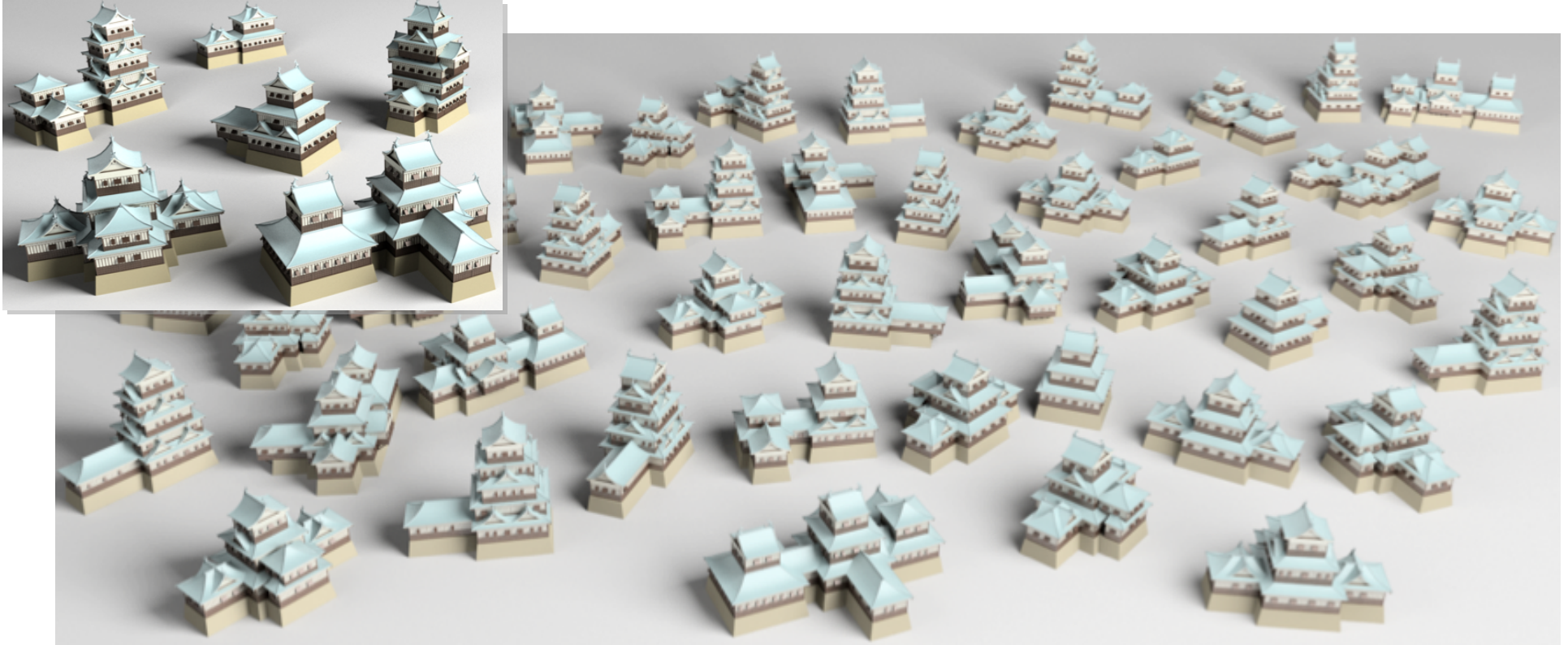
Data-driven:

Moderate

Moderate

Shape Space: Probabilistic Grammar

(learned from examples)



Generality: **Moderate**

Probabilistic: **Yes**

Meaningful parametrization: **Moderate**

Data-driven: **Yes**

Shape Space: Assembly-Based Modeling

(piece together existing components)



Generality: **Moderate**

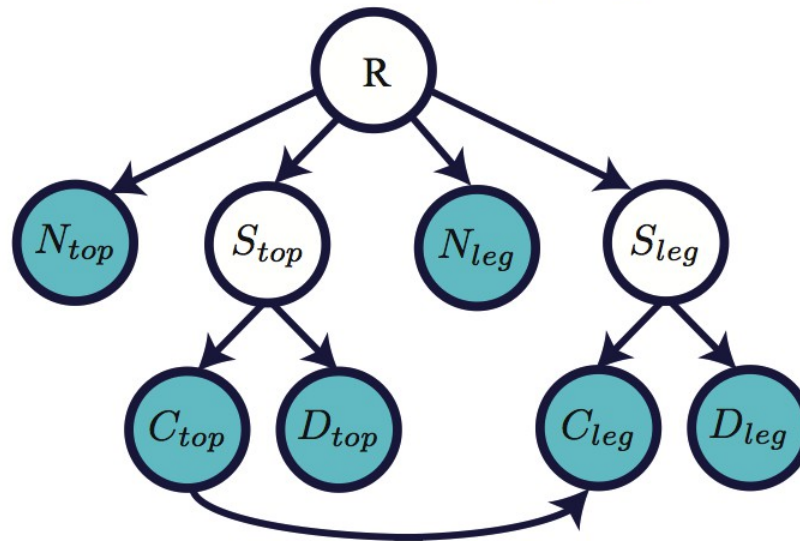
Probabilistic: **No**

Meaningful parametrization: **Yes**

Data-driven: **Reuse**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

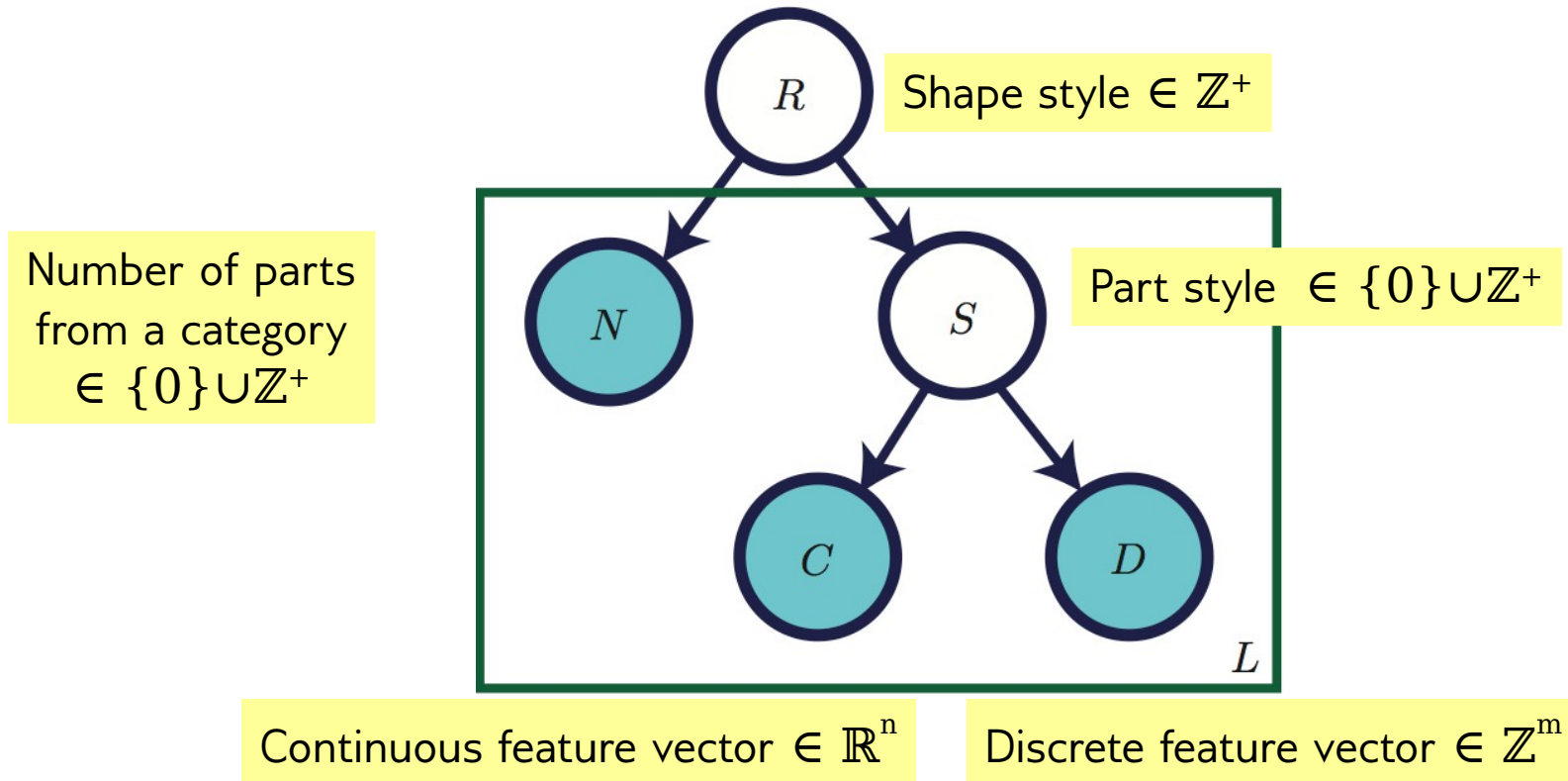
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



$$P(\mathbf{X}) = P(R) \prod_{l \in \mathcal{L}} [P(S_l | R) P(N_l | R, \pi(N_l)) P(\mathbf{C}_l | S_l, \pi(\mathbf{C}_l)) P(\mathbf{D}_l | S_l, \pi(\mathbf{D}_l))]$$

Generality: **Moderate**

Meaningful parametrization: **Yes**

Probabilistic: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Learned shape styles



Learned component styles

Generality: **Moderate**

Probabilistic: **Yes**

Meaningful parametrization: **Yes**

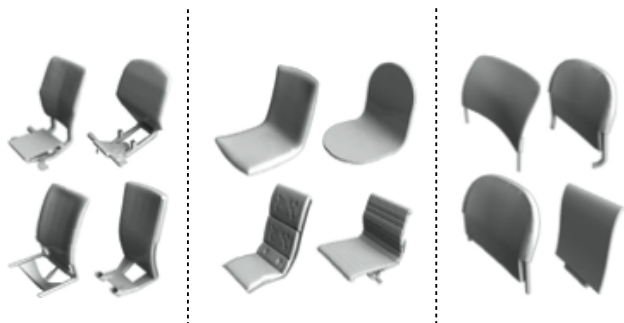
Data-driven: **Yes**

Shape Space: Probabilistic Assembly

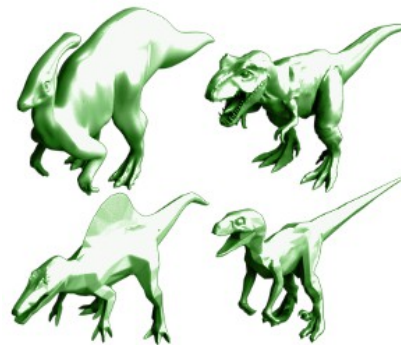
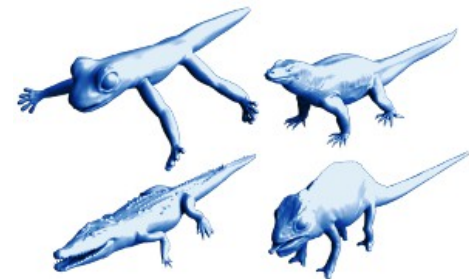
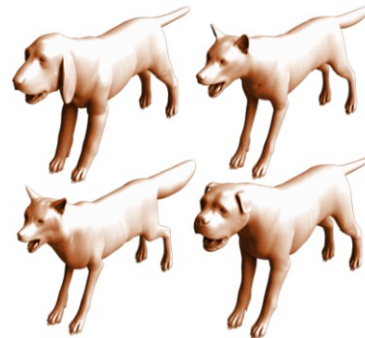
(some assemblies are better than others)



Learned shape styles



Learned component styles



More learned shape “styles”

Generality: **Moderate**
Probabilistic: **Yes**

Meaningful parametrization: **Yes**
Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**
Probabilistic: **Yes**

Meaningful parametrization: **Yes**
Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

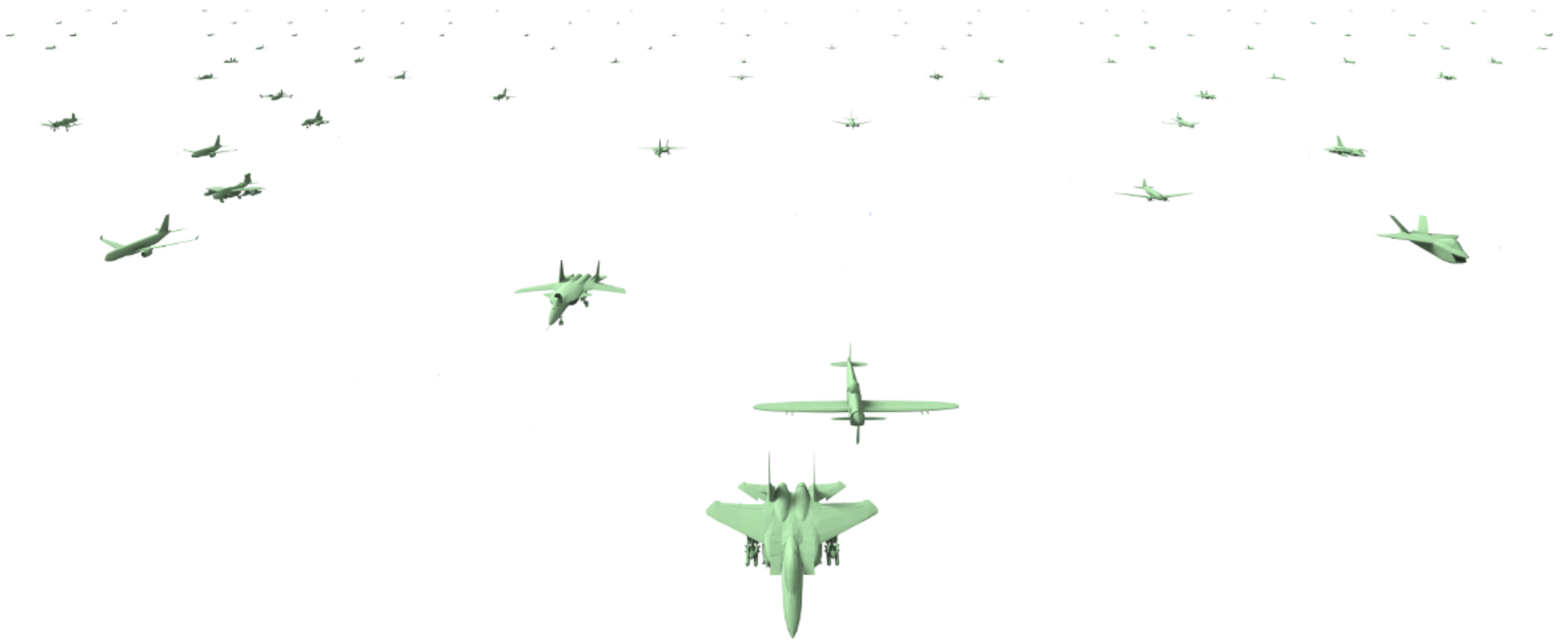
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

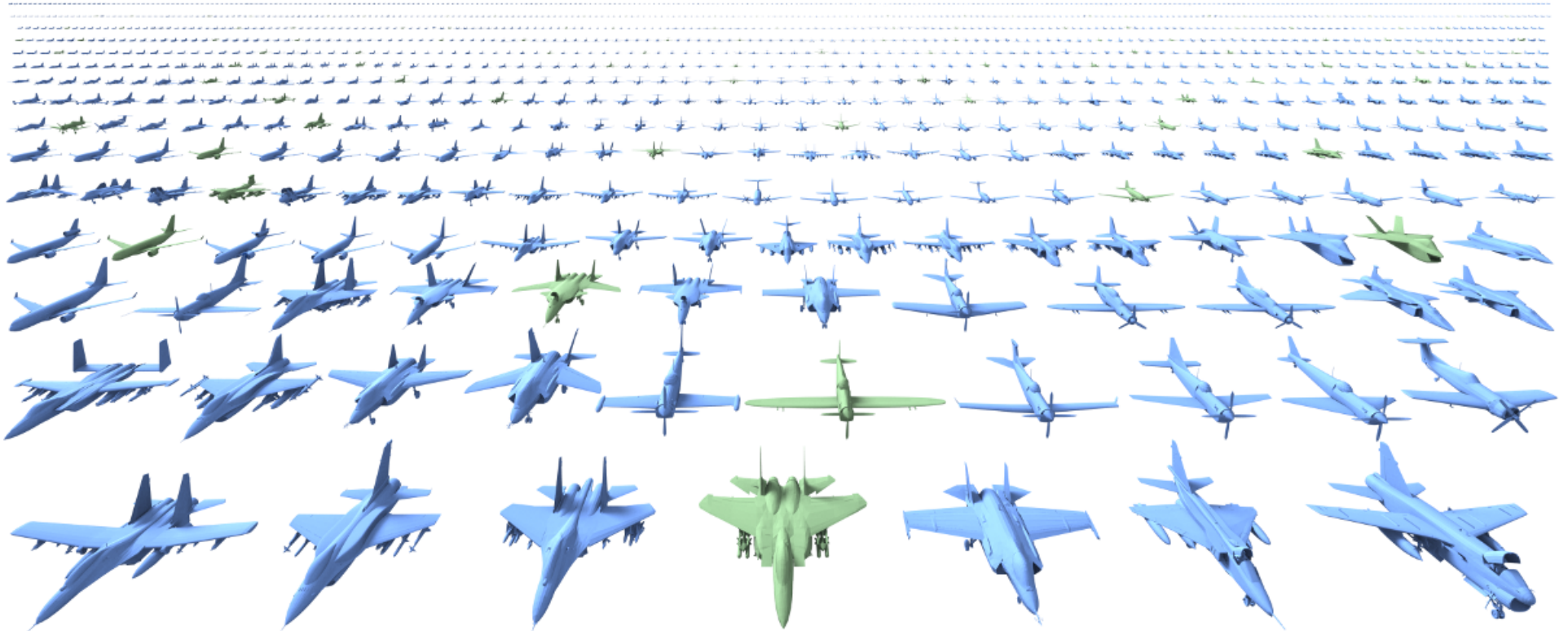
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

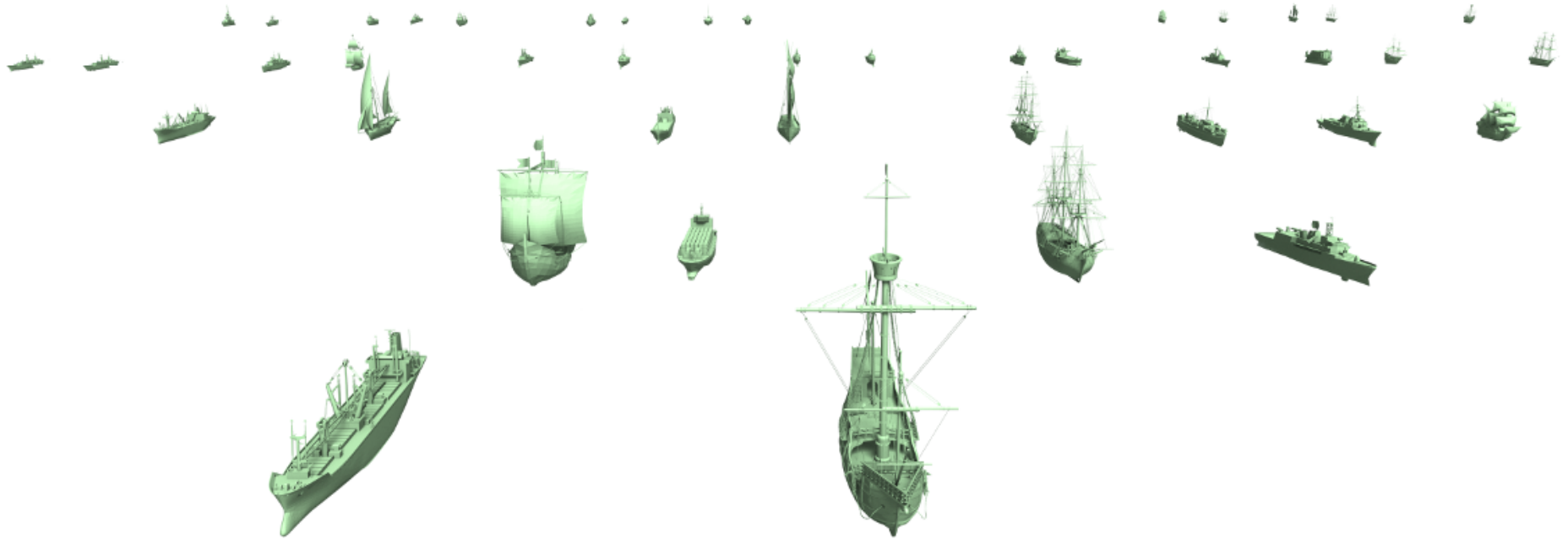
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

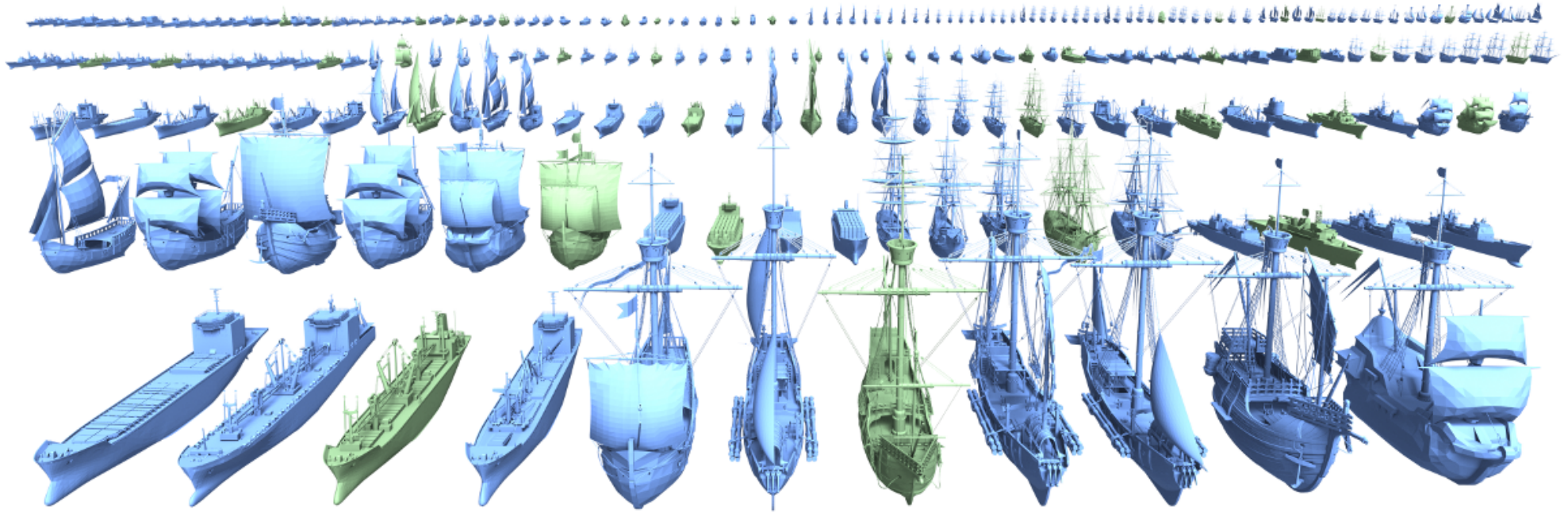
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

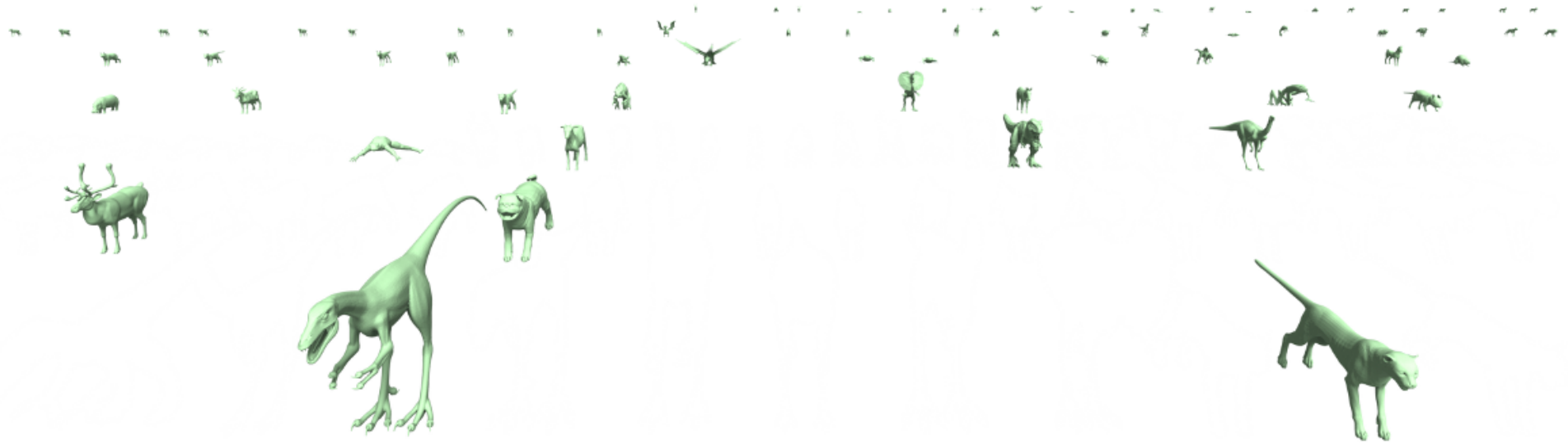
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

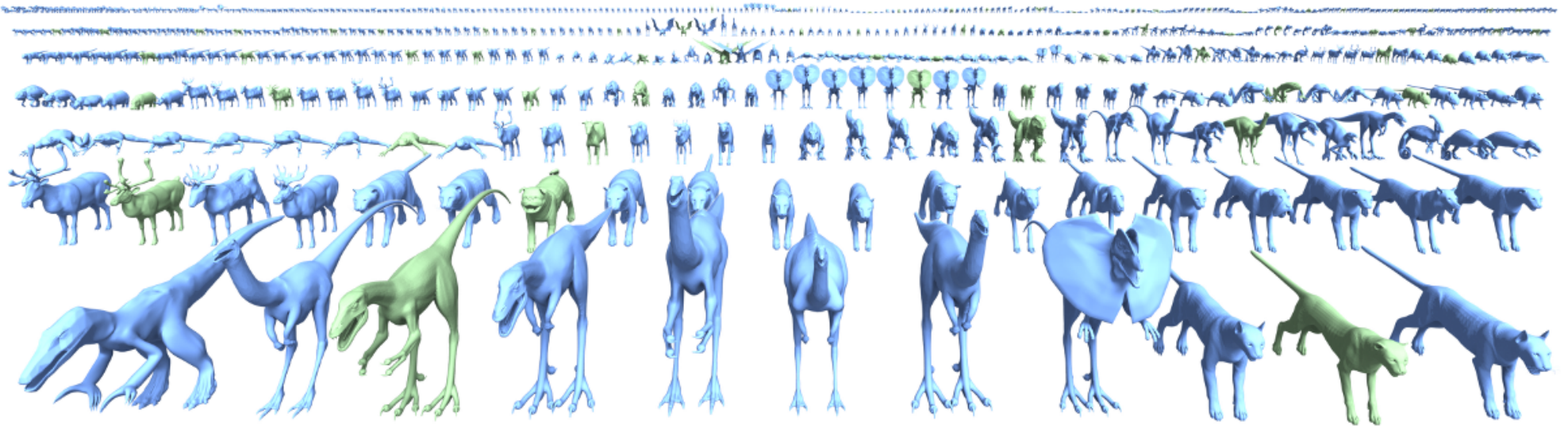
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: Probabilistic Assembly

(some assemblies are better than others)



Generality: **Moderate**

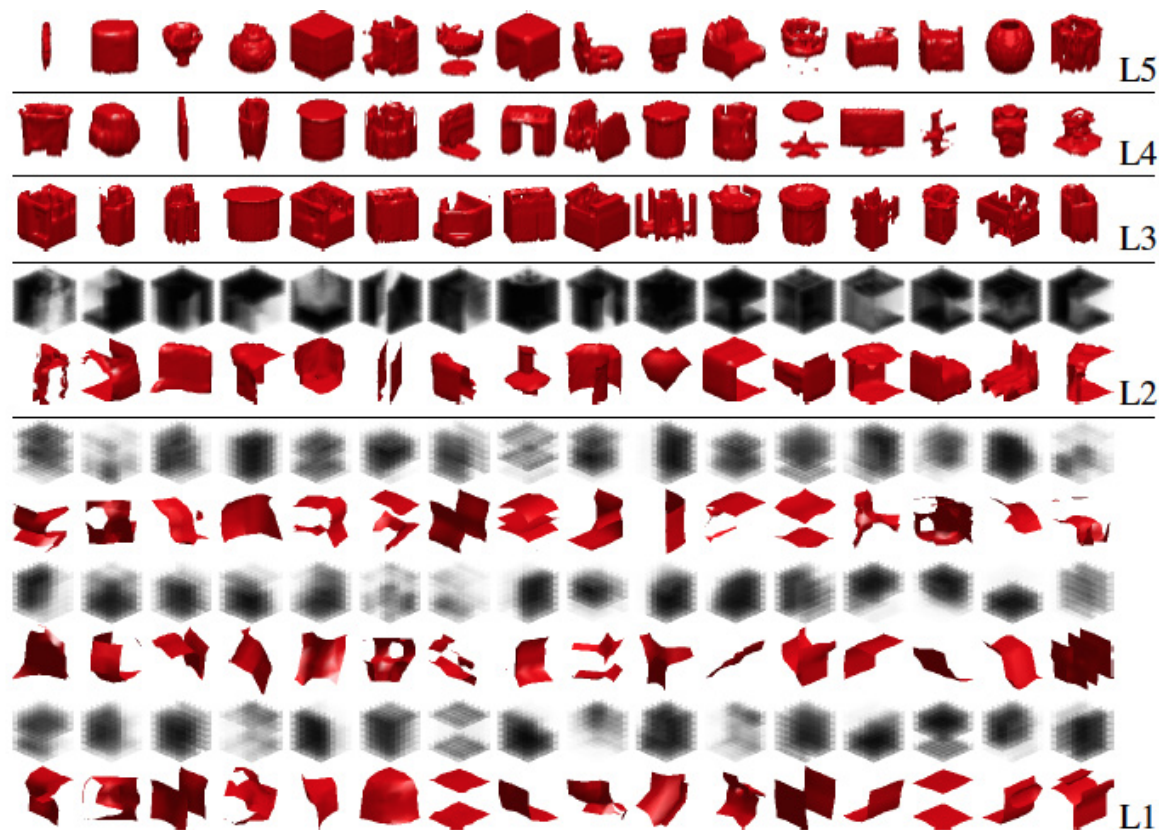
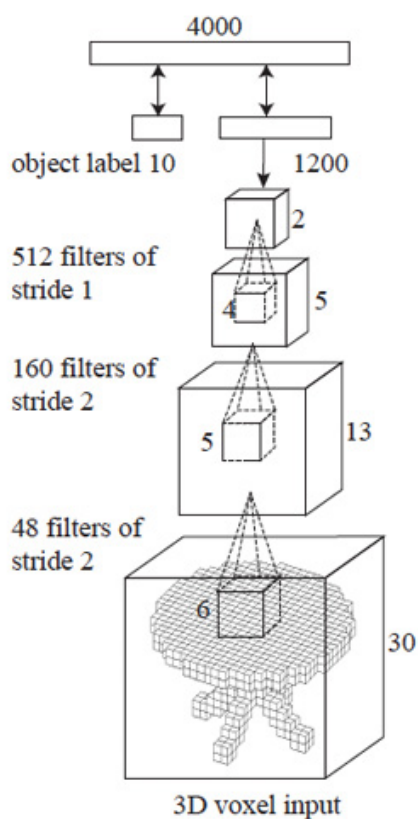
Probabilistic: **Yes**

Meaningful parametrization: **Yes**

Data-driven: **Yes**

Shape Space: 3D Deep Belief Network

(convolutional + fully-connected RBM, stacked layers)



Generality: **High**

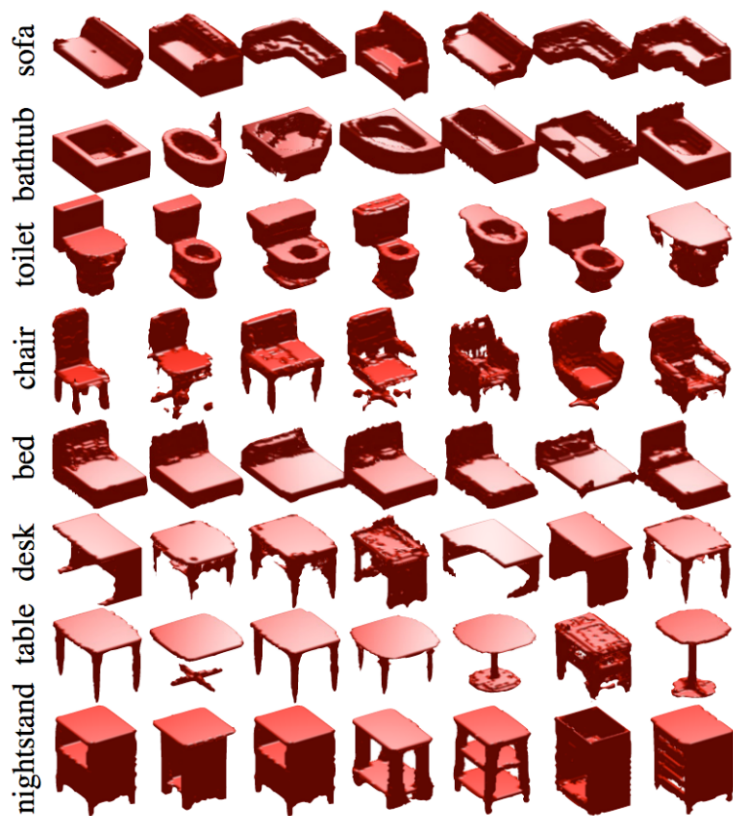
Probabilistic: **Yes**

Meaningful parametrization: **No**

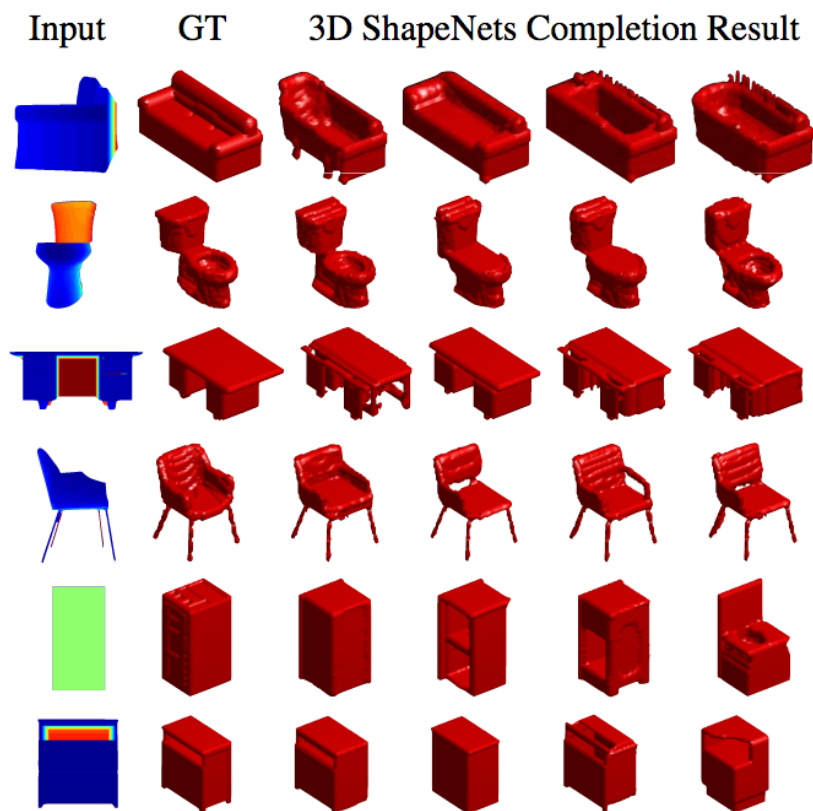
Data-driven: **Yes**

Shape Space: 3D Deep Belief Network

(convolutional + fully-connected RBM, stacked layers)



Sampled shapes



Completed shapes

Generality: **High**

Probabilistic: **Yes**

Meaningful parametrization: **No**

Data-driven: **Yes**

- Make a cute toy

- Make a cute toy
- Make an aerodynamic airplane

- Make a cute toy
- Make an aerodynamic airplane
- Make a comfortable chair

- Make a cute toy
- Make an aerodynamic airplane
- Make a comfortable chair
- Make an efficient bicycle

- Make a cute toy
- Make an aerodynamic airplane
- Make a comfortable chair
- Make an efficient bicycle
- Make a professional-looking webpage

- Make a cute **toy**
- Make an aerodynamic **airplane**
- Make a comfortable **chair**
- Make an efficient **bicycle**
- Make a professional-looking **webpage**

- Make a **cute** toy
- Make an **aerodynamic** airplane
- Make a **comfortable** chair
- Make an **efficient** bicycle
- Make a **professional**-looking webpage

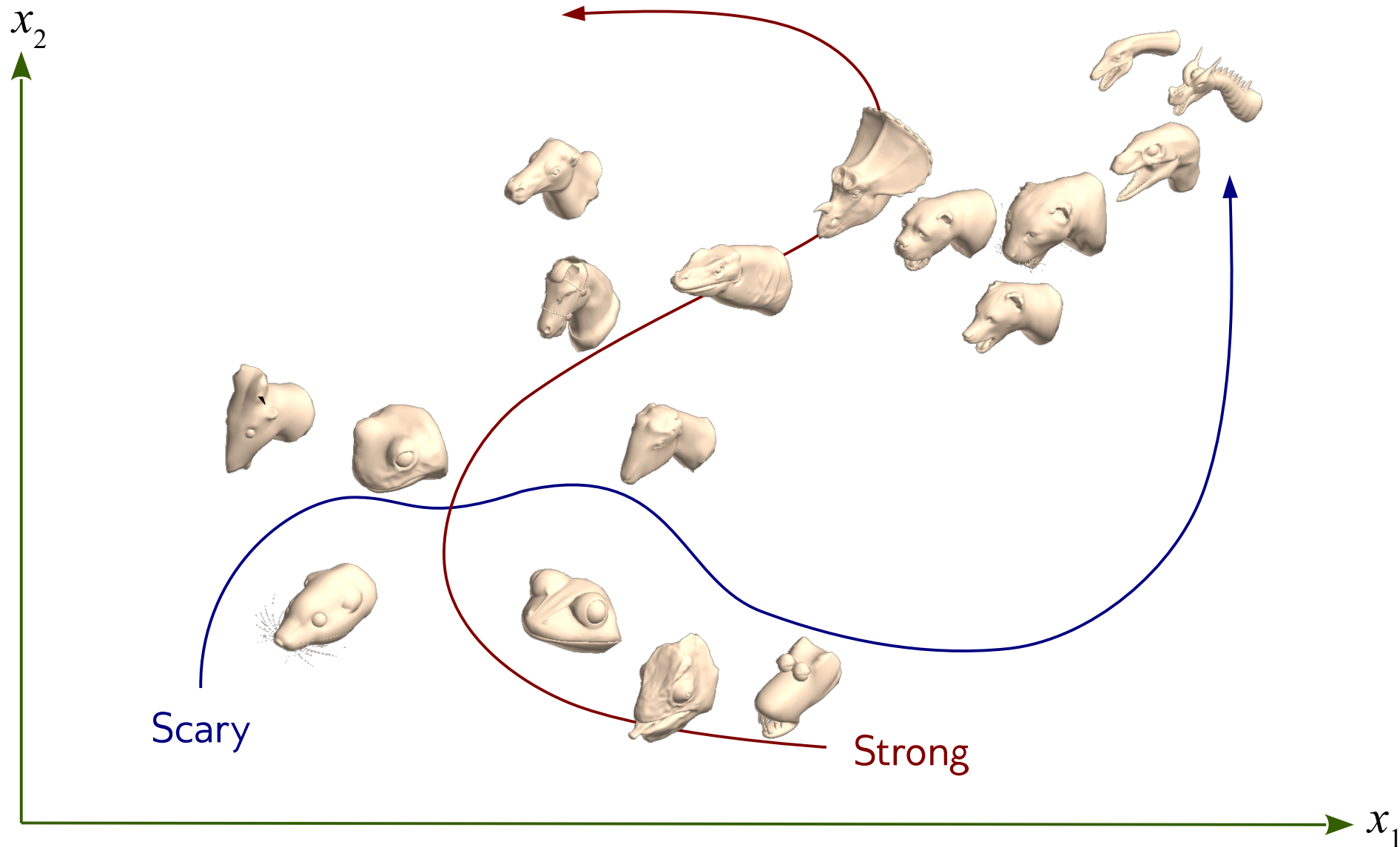
Outline

- Learning shape structure
 - Probabilistic models of shape
- Learning shape semantics
 - Semantic **attributes** (*scary, artistic, ...*)
 - Mechanical **function** (*this airplane should fly...*)
 - Human **interaction** (*sit comfortably in a chair...*)

Semantic Basis for Shape Space



Semantic Basis for Shape Space



A cute toy for a small child



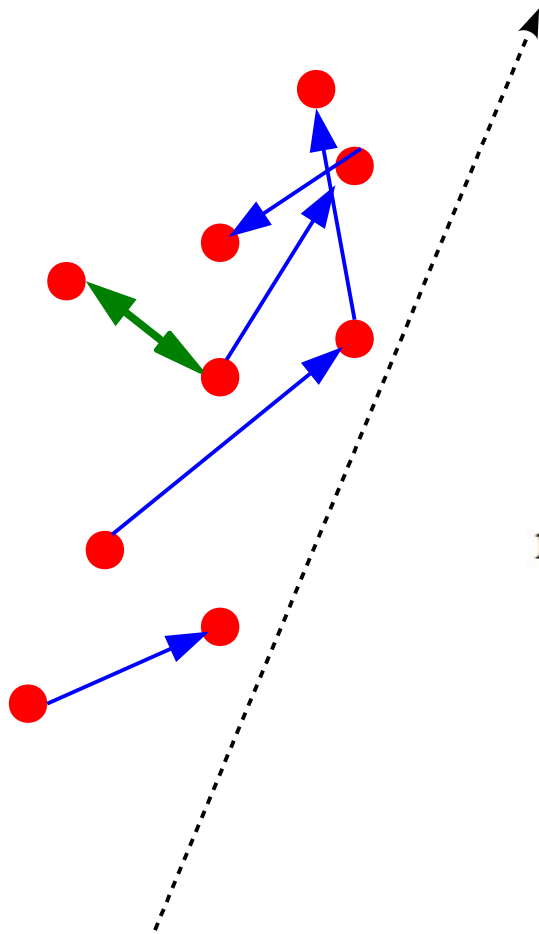
(Video)

Learning Semantic Attributes

- Crowdsource **comparative adjectives**
 - Amazon Mechanical Turk
 - Schelling survey
- Crowdsource comparisons for **training pairs**
 - A is more [.....] than B
- Learn **ranking functions**
 - f : shape features $\rightarrow \mathbb{R}$
 - Rank-SVM with transformed features & sigmoid loss
 - Iterate with cross-correlation between attributes
 - Extend to multi-component rankings

Learning Semantic Attributes

- **Rank-SVM:** Project features onto linear subspace that best preserves pairwise orderings



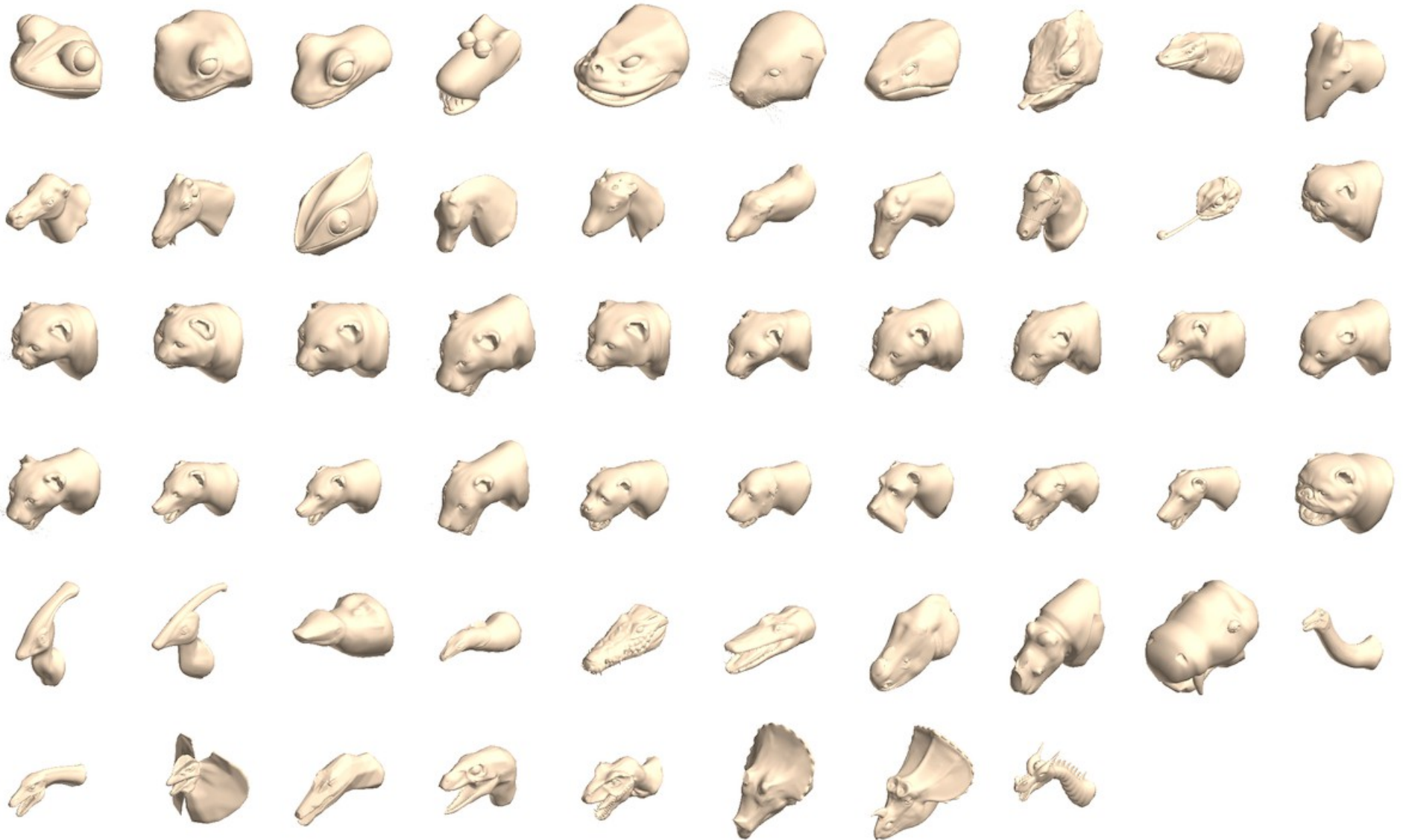
Learn $r_m(\mathbf{X}) = \mathbf{W}_m \cdot \mathbf{X}$

s.t. $\forall (i, j) \in O_m : \mathbf{w}_m \cdot \mathbf{x}_i > \mathbf{w}_m \cdot \mathbf{x}_j$
 $\forall (i, j) \in S_m : \mathbf{w}_m \cdot \mathbf{x}_i = \mathbf{w}_m \cdot \mathbf{x}_j$

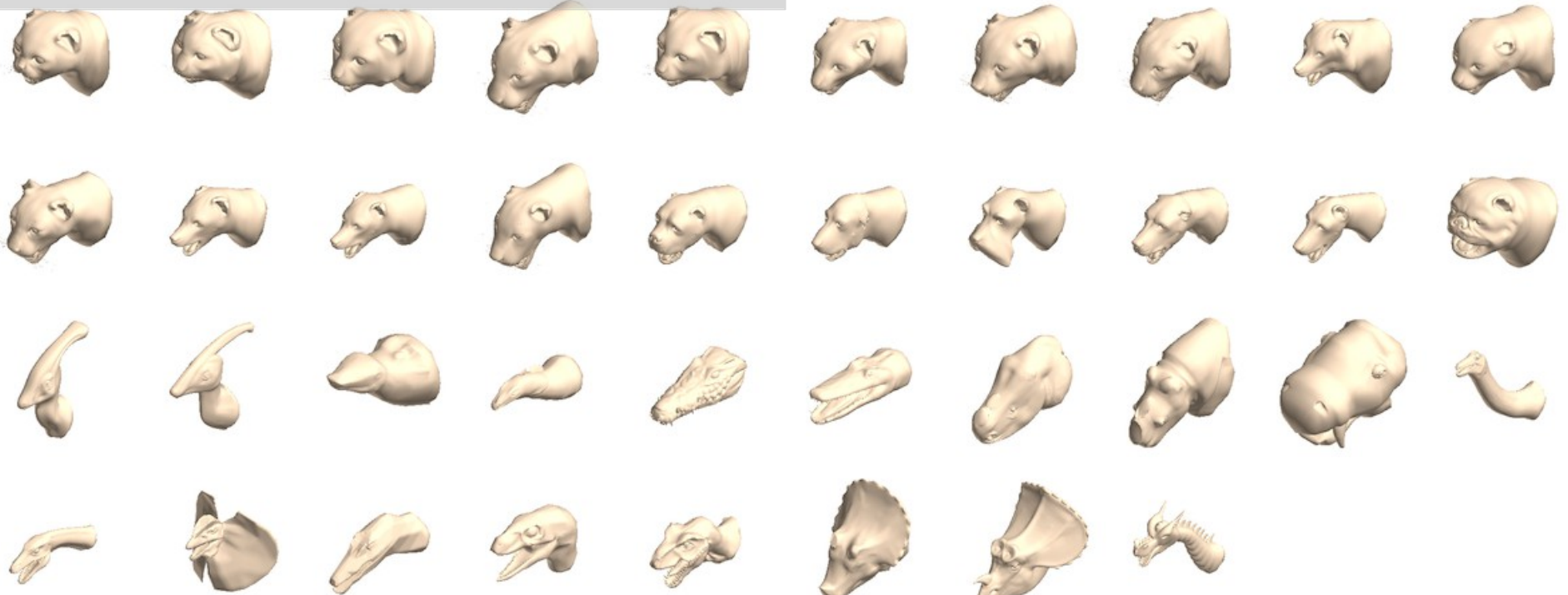
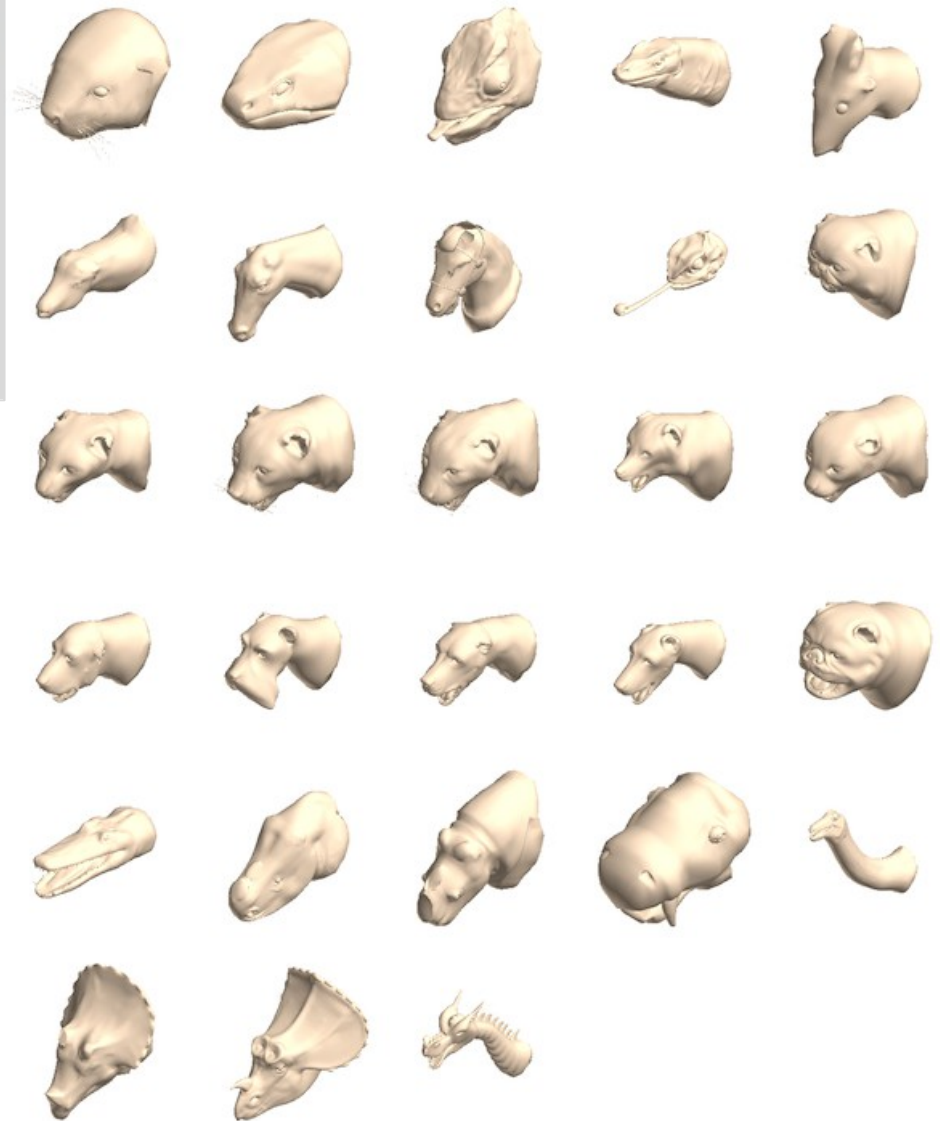
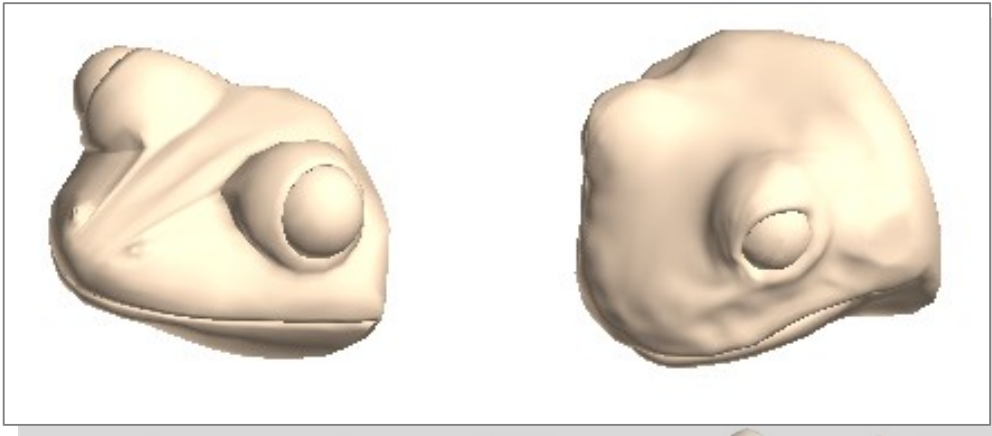


minimize $\|\mathbf{w}_m\|_2^2 + \mu \sum_{i, j \in O_m} c_{ij} (1 - \sigma(\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)))$
 $+ \nu \sum_{i, j \in S_m} c_{ij} \sigma(|\mathbf{w}_m(\mathbf{x}_i - \mathbf{x}_j)|)$

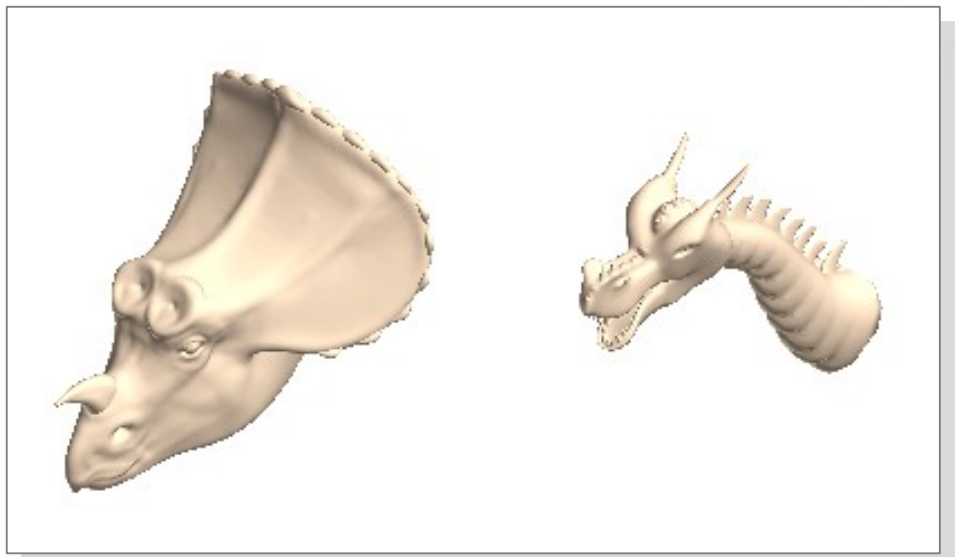
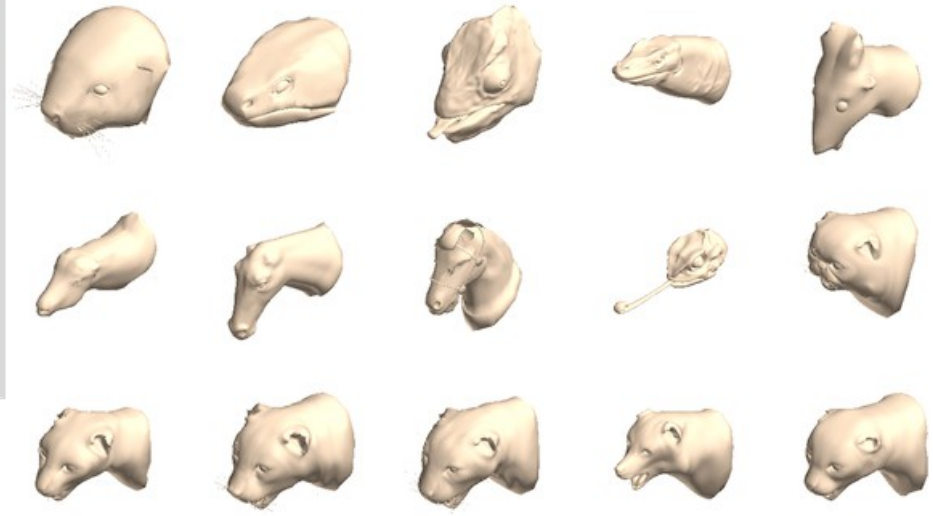
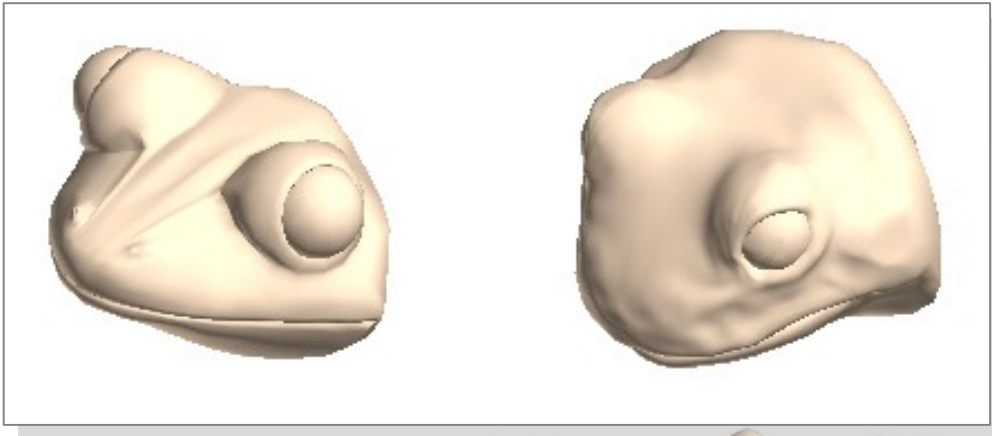
“Dangerous”



“Dangerous”



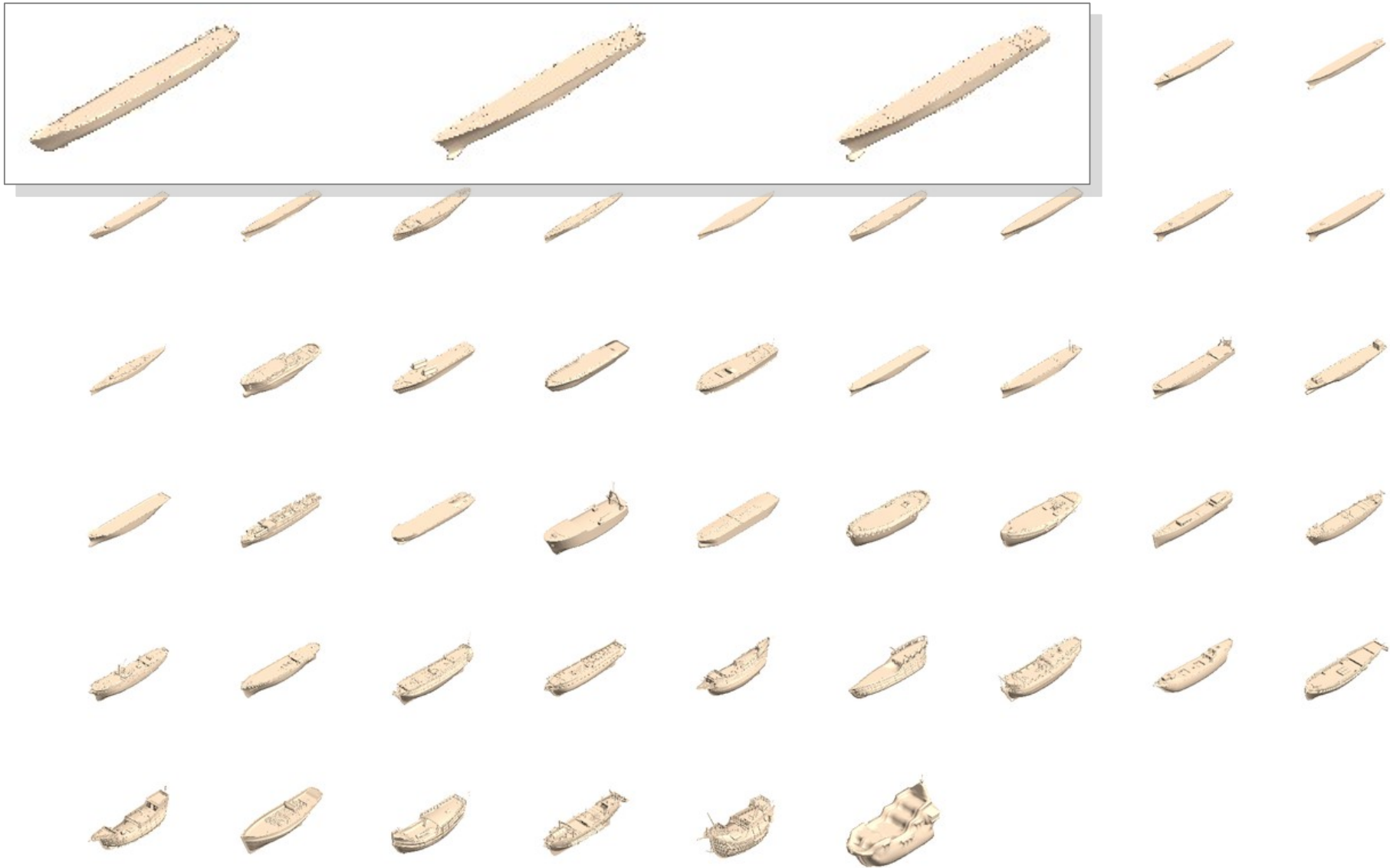
“Dangerous”



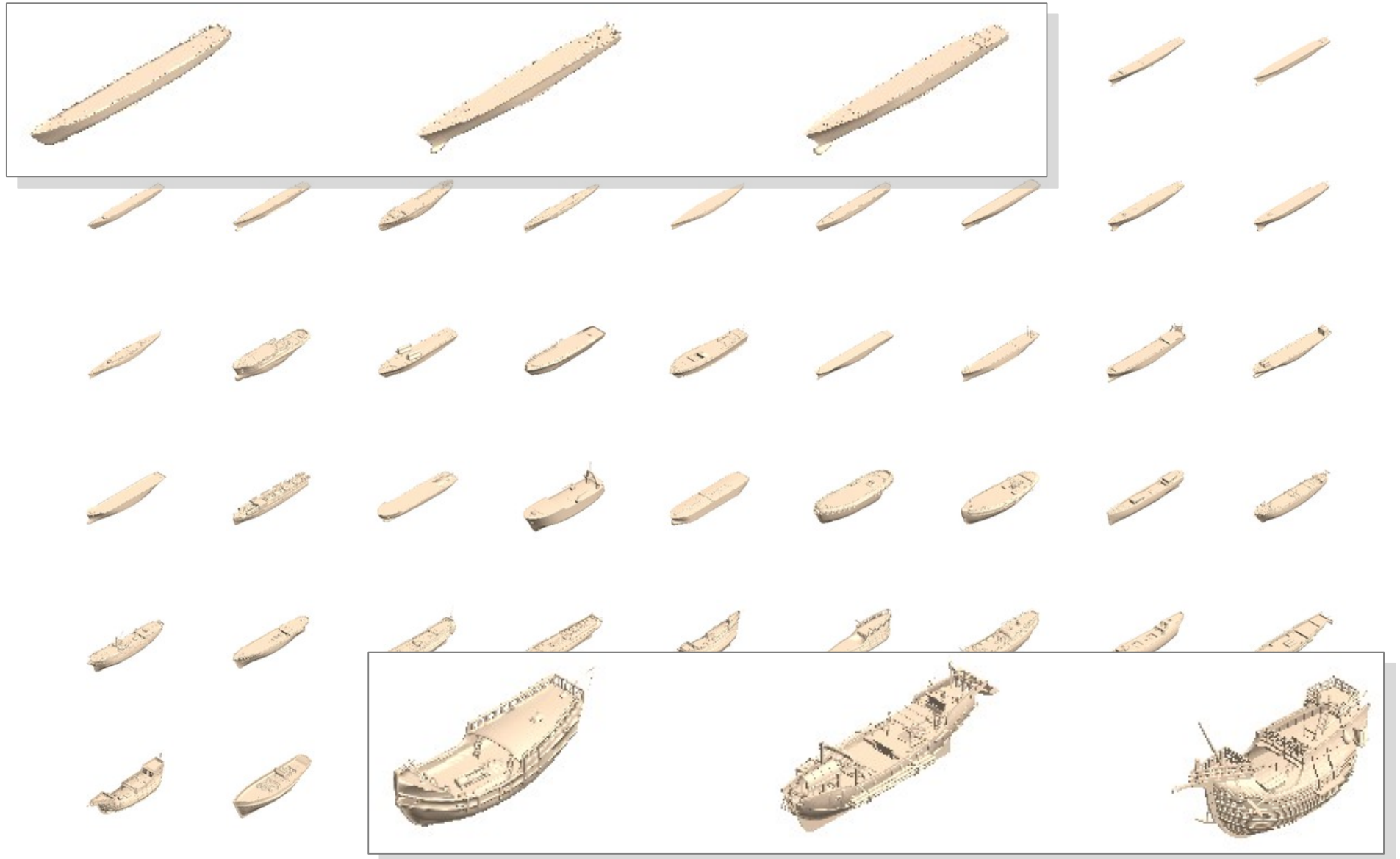
“Old-fashioned”



“Old-fashioned”



“Old-fashioned”



Semantic Shape Editing

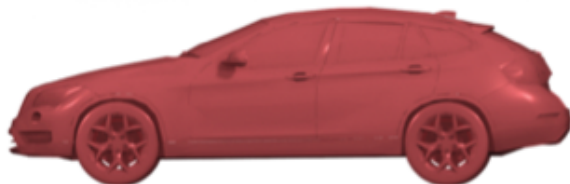
(Video)



Less *compact*



More *muscular*



More *sporty*

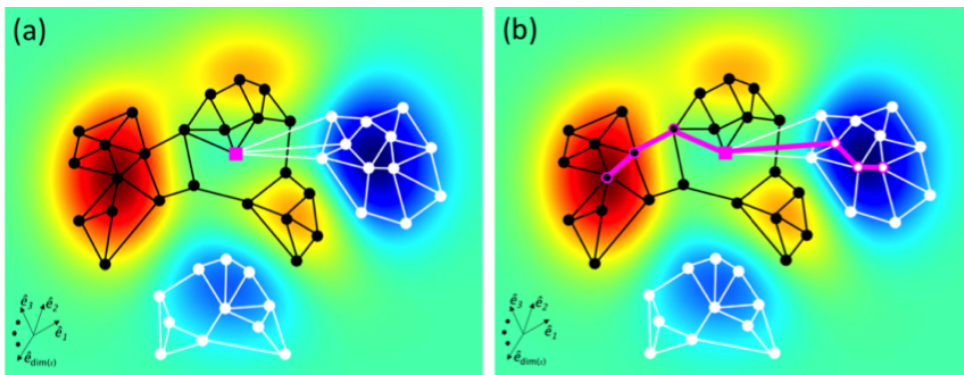
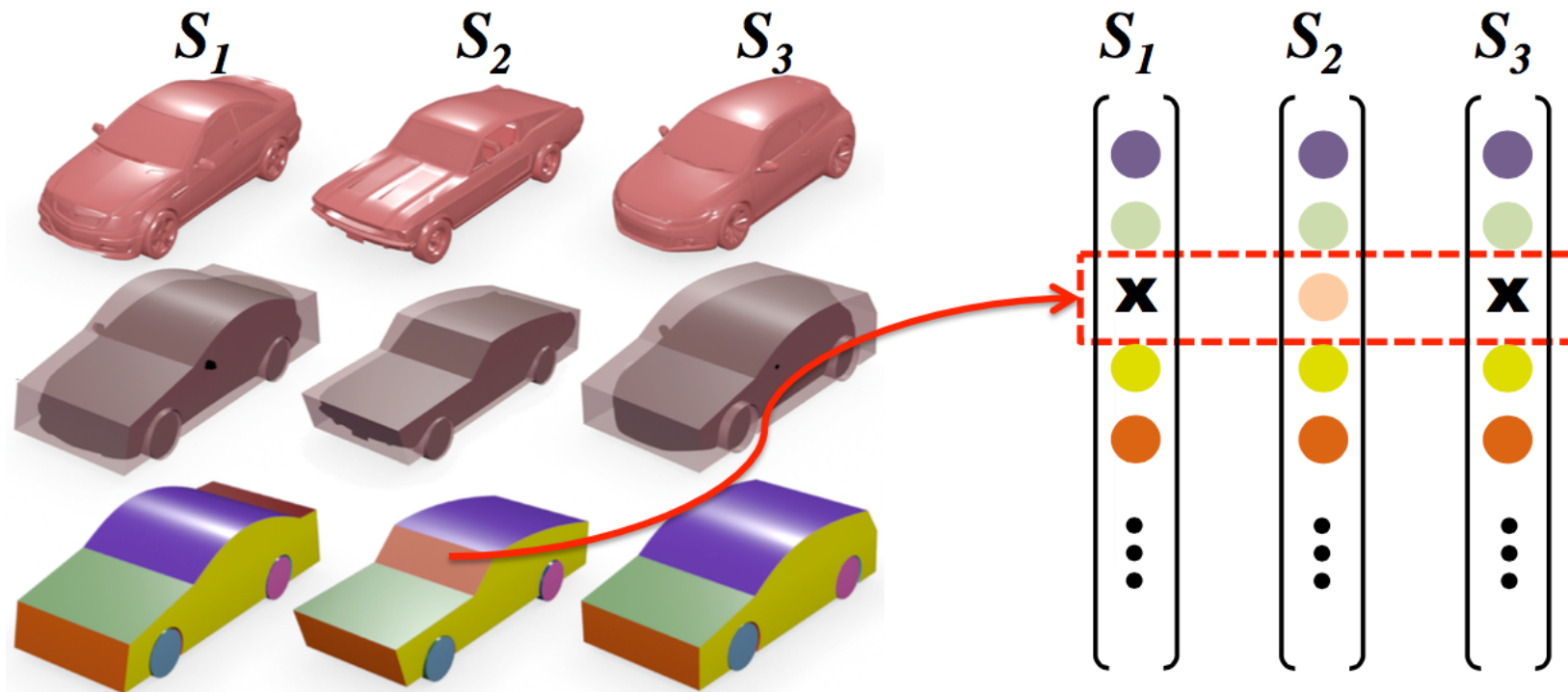


Less *modern*



More *luxurious*

Semantic Shape Editing

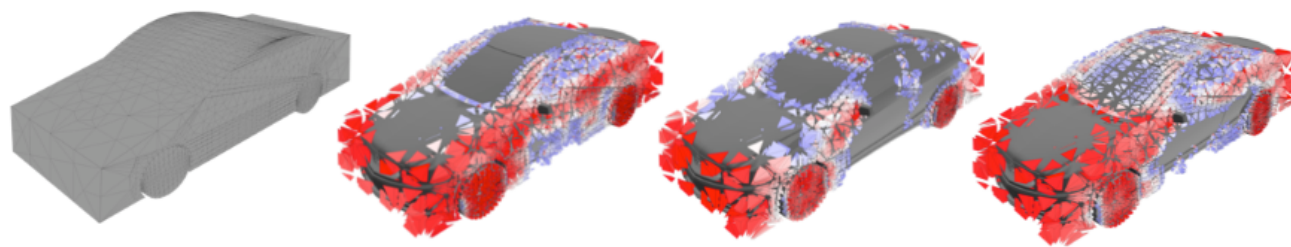
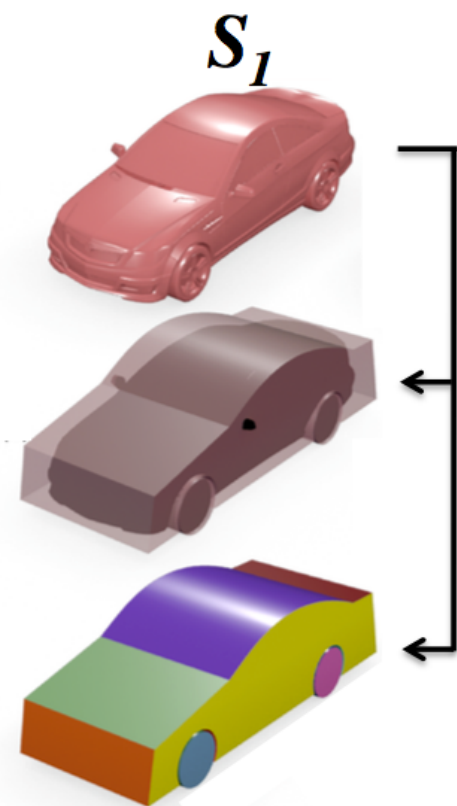


$$\underset{p_i}{\text{minimize}} \sum_{\{\mathcal{V}_h, \mathcal{E}_h\} \in \mathcal{H}} \left(\sum_{i \in \mathcal{V}_h} |p_i - f_i| + \lambda \sum_{j \in \mathcal{E}_h} -\log \left(\frac{\beta_j}{\pi} \right) \right)$$

Deformation Space = Feature Space

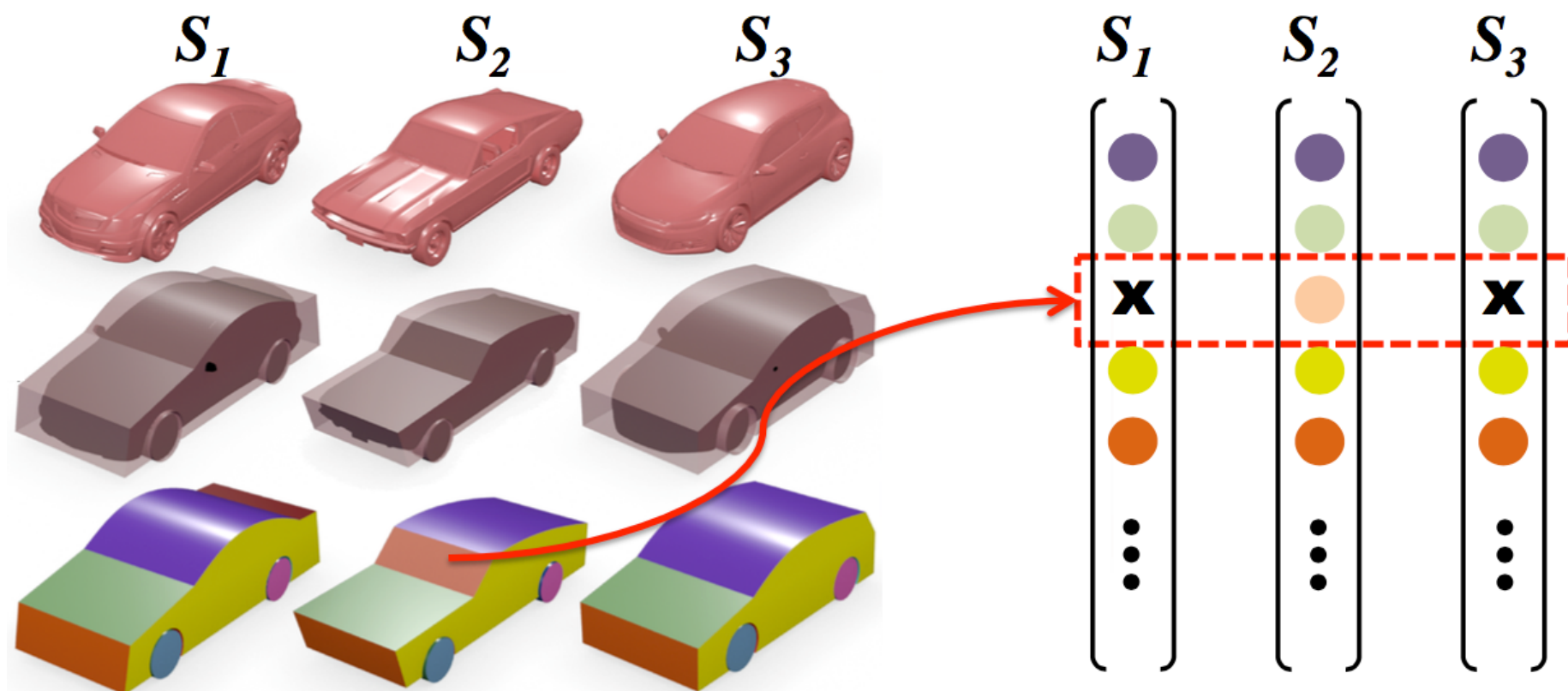
Use deformation handle parameters as shape features

- Sphere: $r, p_i | i \in \{x, y, z\}$
- Cylinder: $r, p_i, \theta_i | i \in \{x, y, z\}$
- Circular Cone: $r, \beta, p_i, \theta_i | i \in \{x, y, z\}$
- Quadric: $k_1, k_2, p_i, \theta_i | i \in \{x, y, z\}$

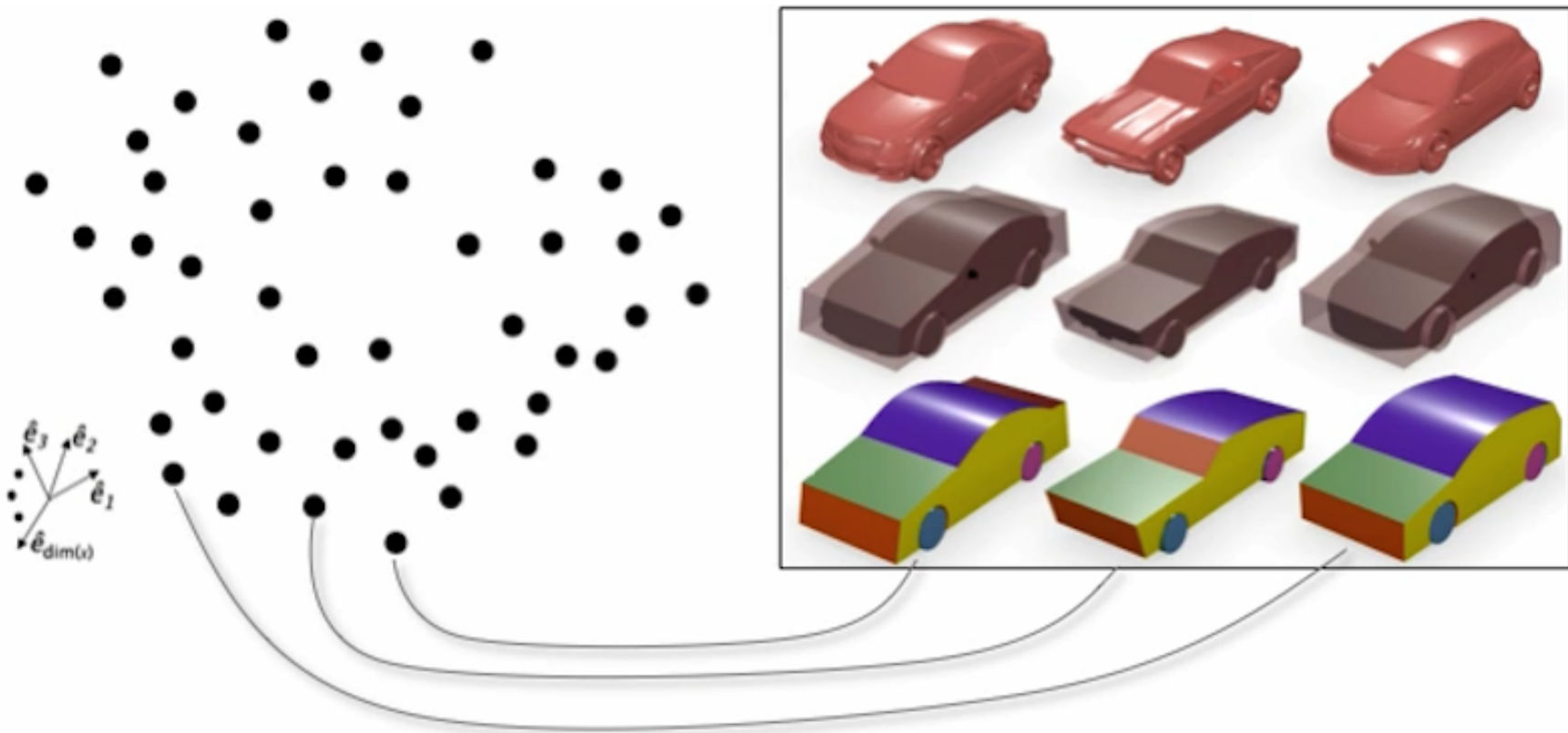


Deformation Space = Feature Space

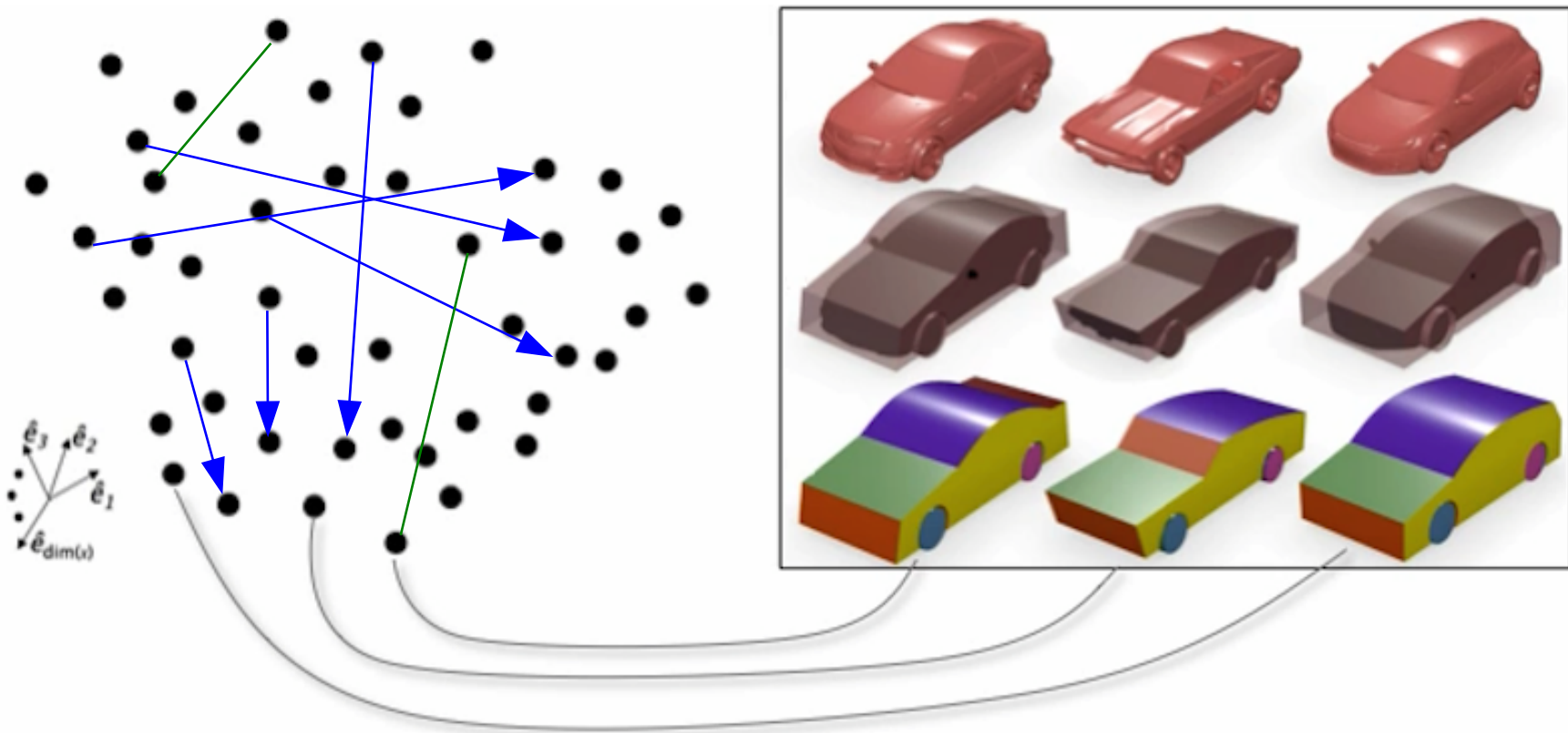
Use deformation handle parameters as shape features



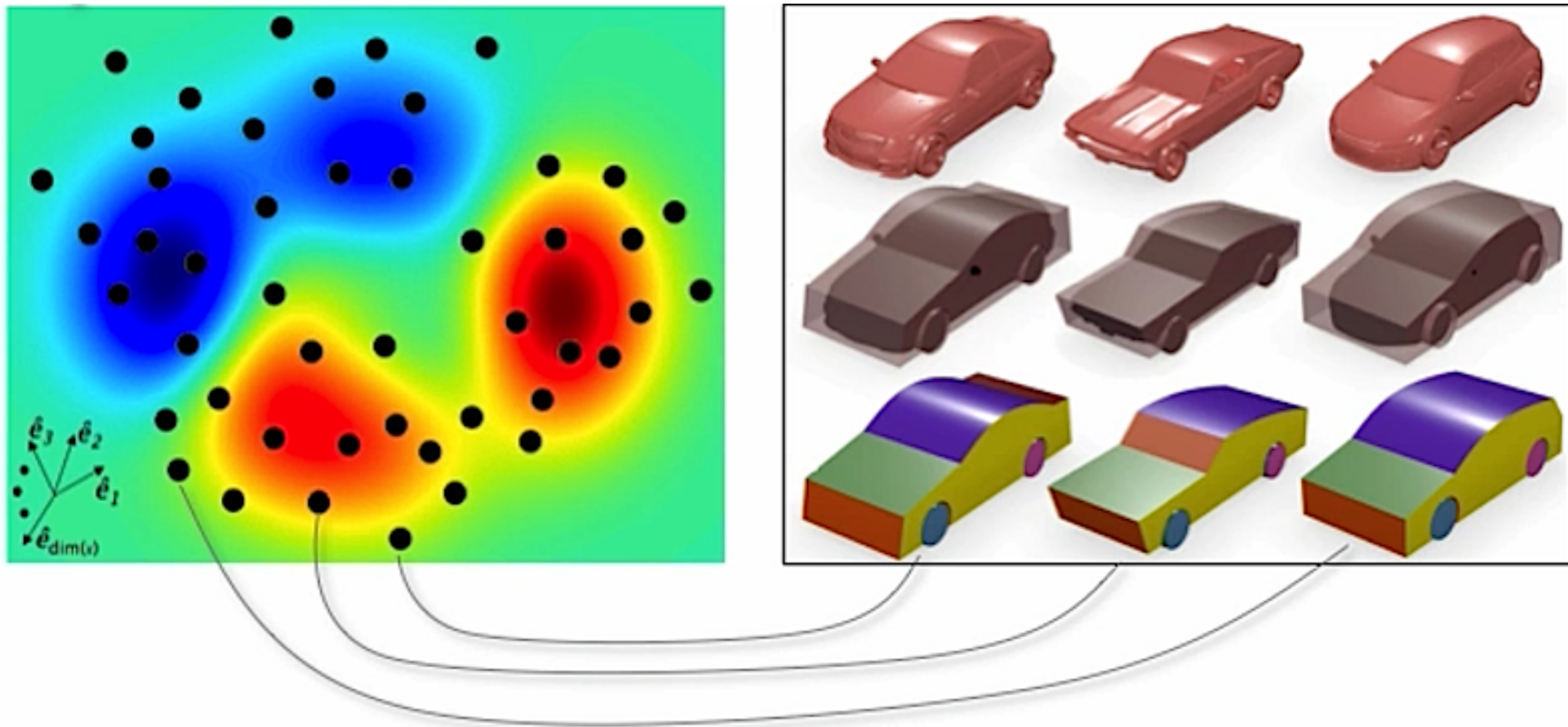
Deformation Space = Feature Space



Comparisons in Feature Space



Attribute Distribution over Feature Space



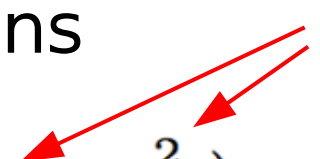
Attribute Learning: Absolute Scores

- Assume normally distributed absolute scores

$${}_a P_i \sim N({}_a \mu_i, {}_a \sigma_i^2) = \frac{1}{{}_a \sigma_i \sqrt{2\pi}} e^{-\frac{(x - {}_a \mu_i)^2}{2 {}_a \sigma_i^2}}$$

- Pairwise comparisons modeled as difference of normal distributions

from user study statistics

$${}_a P_i - {}_a P_j \sim N({}_a \mu_{ij}, {}_a \sigma_{ij}^2) = N({}_a \mu_i - {}_a \mu_j, {}_a \sigma_i^2 + {}_a \sigma_j^2)$$


- Solve overdetermined linear system

$${}_a \mu_i - {}_a \mu_j = {}_a \mu_{ij} \quad {}_a \sigma_i^2 + {}_a \sigma_j^2 = {}_a \sigma_{ij}^2$$

Attribute Learning: Scoring Function

$$\tilde{f}_a(\mathbf{x}_s) = \sum_{t \in \mathcal{T}} \frac{w_t(\mathbf{x}_s)}{\sum_j w_j(\mathbf{x}_s)} f_a(\mathbf{x}_t)$$

$$w_t(\mathbf{x}_s) = {}_a r_t \|\mathbb{1}_s \cdot \mathbb{1}_t \cdot (\mathbf{x}_s - \mathbf{x}_t)\|^{-p}$$

\mathbf{x}_s : feature vector of the new shape

\mathbf{x}_t : feature vector of shape t from database

$\tilde{f}_a(\mathbf{x}_s)$: attribute score of the new shape

$f_a(\mathbf{x}_t)$: attribute score of shape t from database

\mathcal{T} : set of all shapes in the database

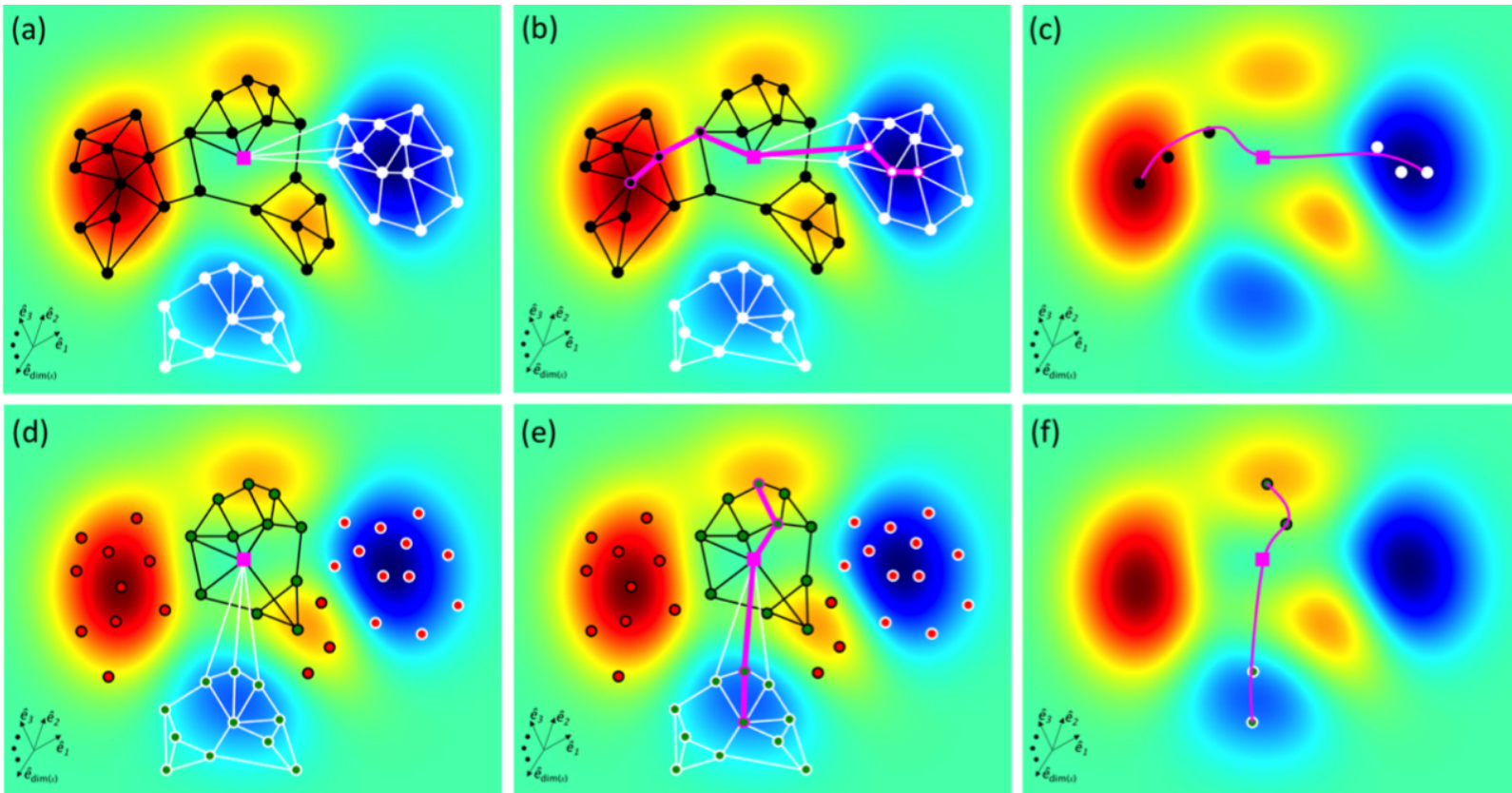
$\mathbb{1}_i$: indicator function of feature vector i

${}_a r_t$: reliability factor

$$f_a(\mathbf{x}_t) = {}_a \mu_t$$

$${}_a r_t = 1 / {}_a \sigma_k^2$$

Constrained Path Traversal



Less

Attribute Values (for which the slider is being designed)

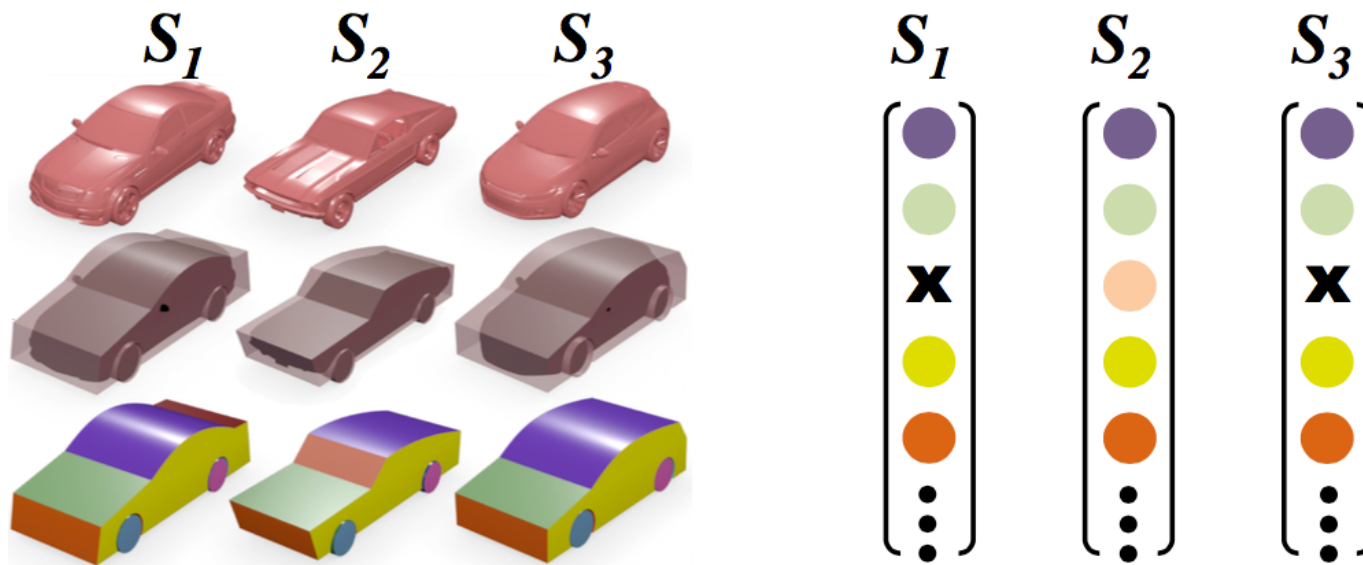
More



- : Edited shape's current location in feature space
- : Shapes with higher attribute value
- : Shapes with lower attribute value
- : Edited shape's spline path mapped to the slider
- : Shapes that violate the active constraint
- : Shapes that do not violate the active constraint

Deformation from a Given Feature Vector

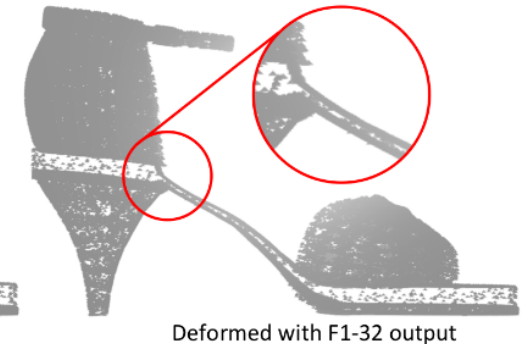
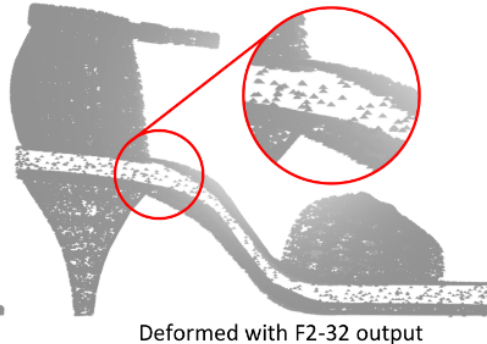
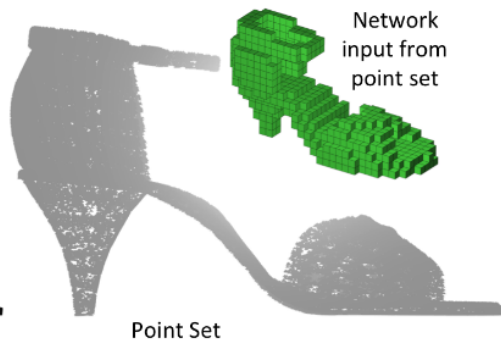
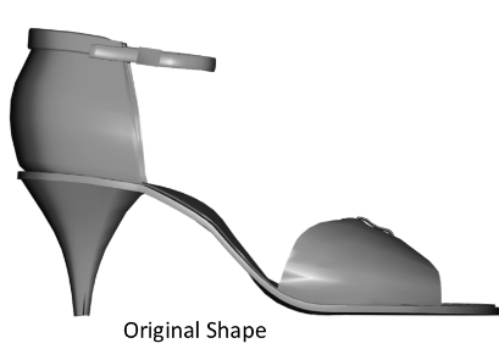
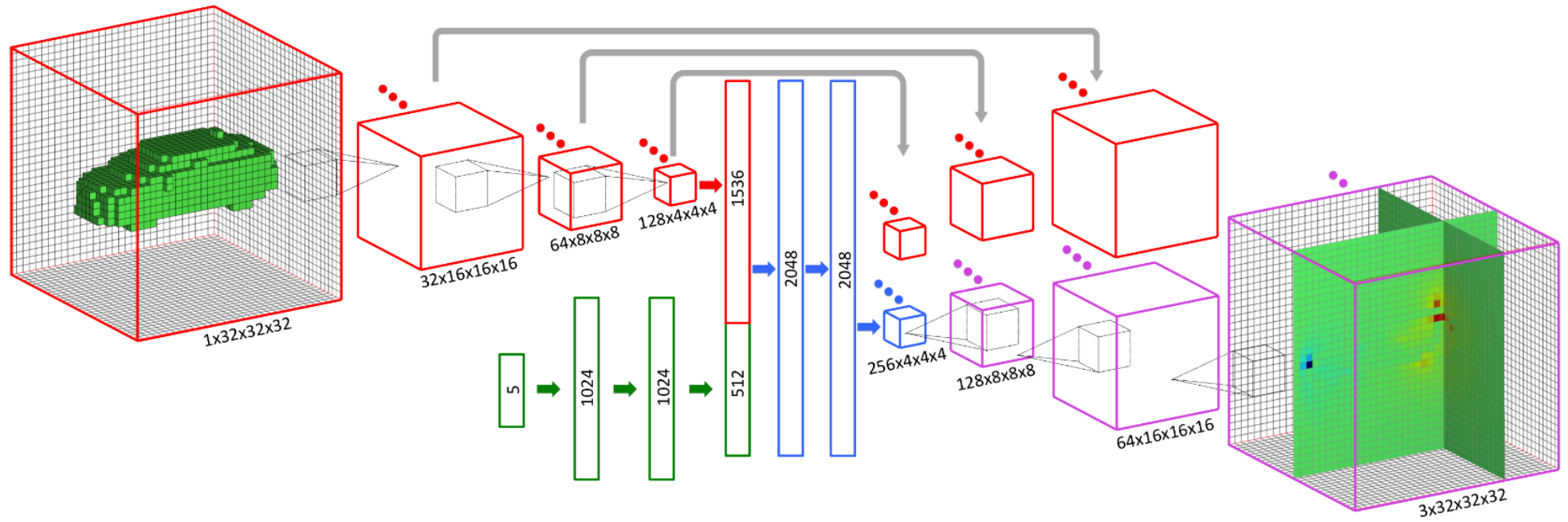
Flashback: Deformation handle parameters = shape feature vector



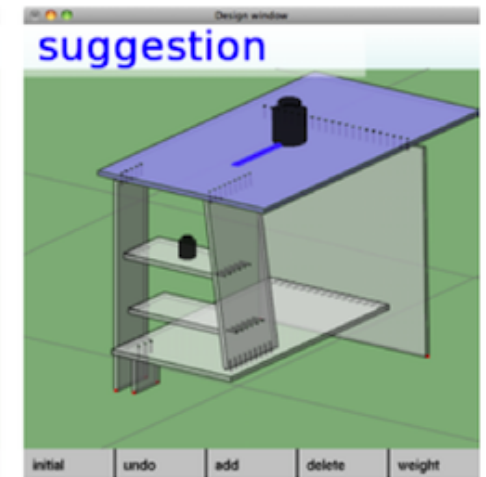
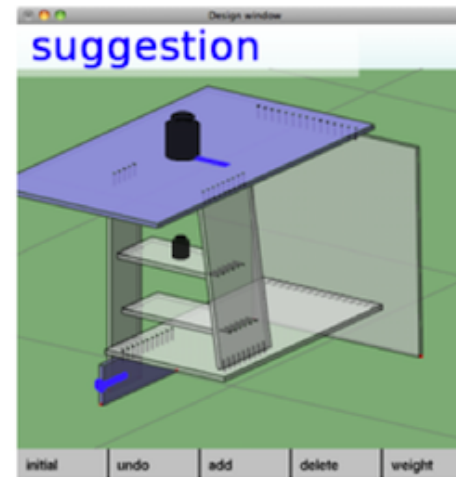
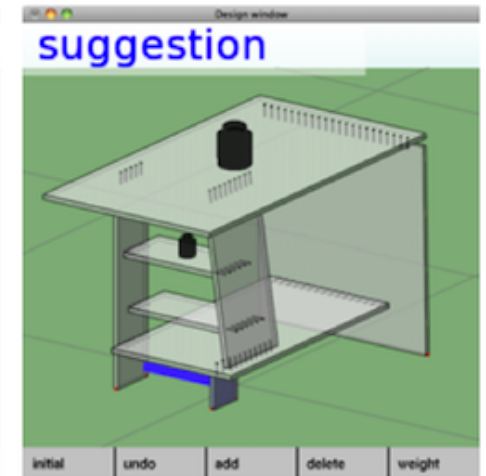
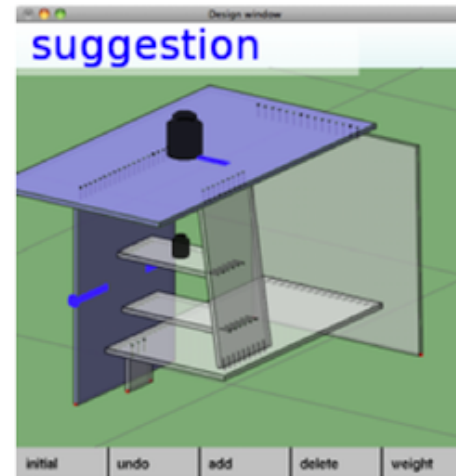
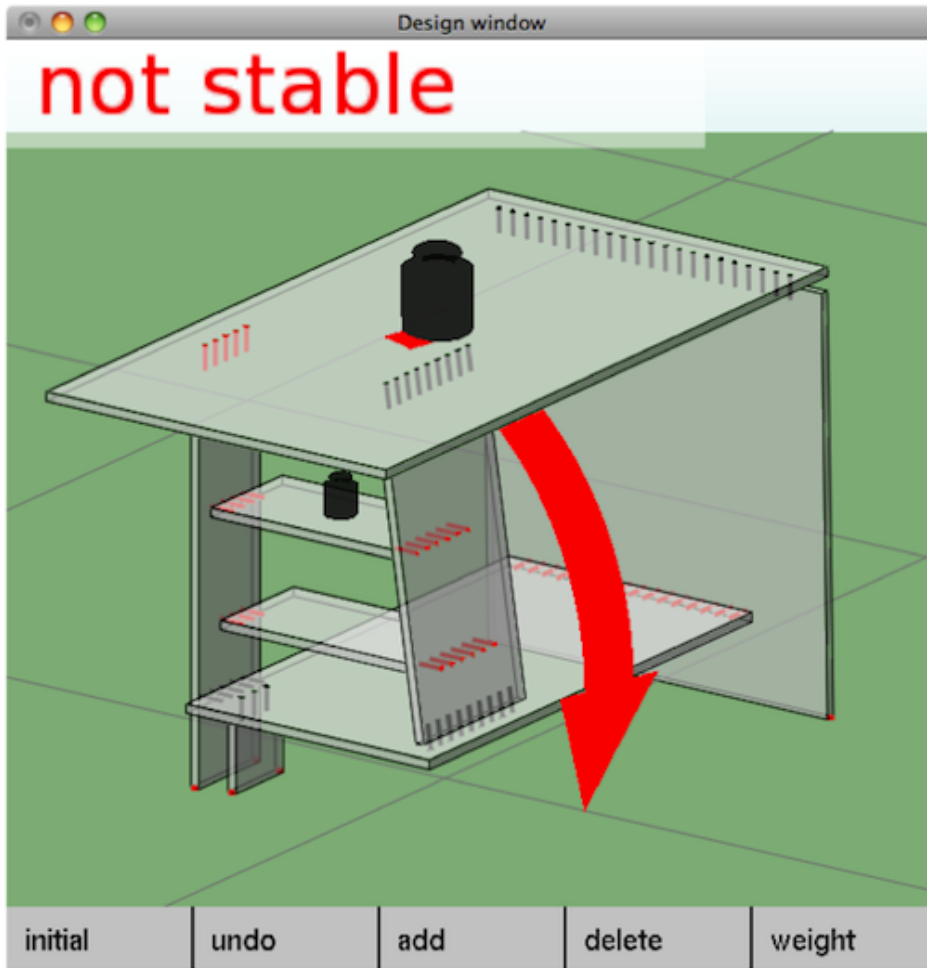
$$\underset{p_i}{\text{minimize}} \sum_{\{v_h, \varepsilon_h\} \in \mathcal{H}} \left(\sum_{i \in \mathcal{V}_h} |p_i - f_i| + \lambda \sum_{j \in \mathcal{E}_h} -\log \left(\frac{\beta_j}{\pi} \right) \right)$$



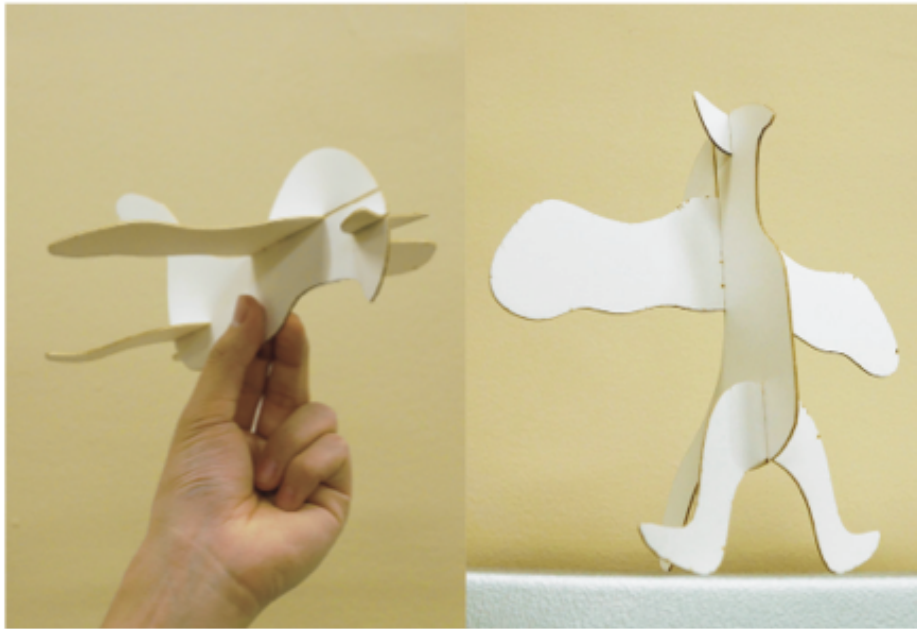
Semantic Deformation Flow



Designing for Mechanical Function



Designing for Mechanical Function

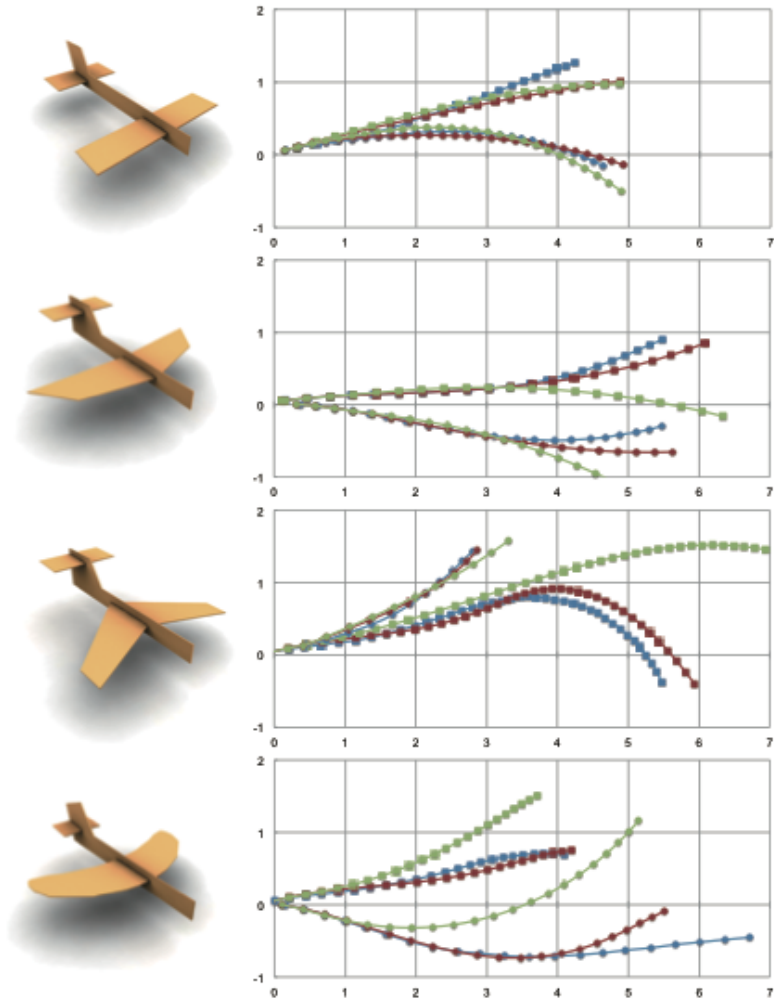


Designing for Mechanical Function

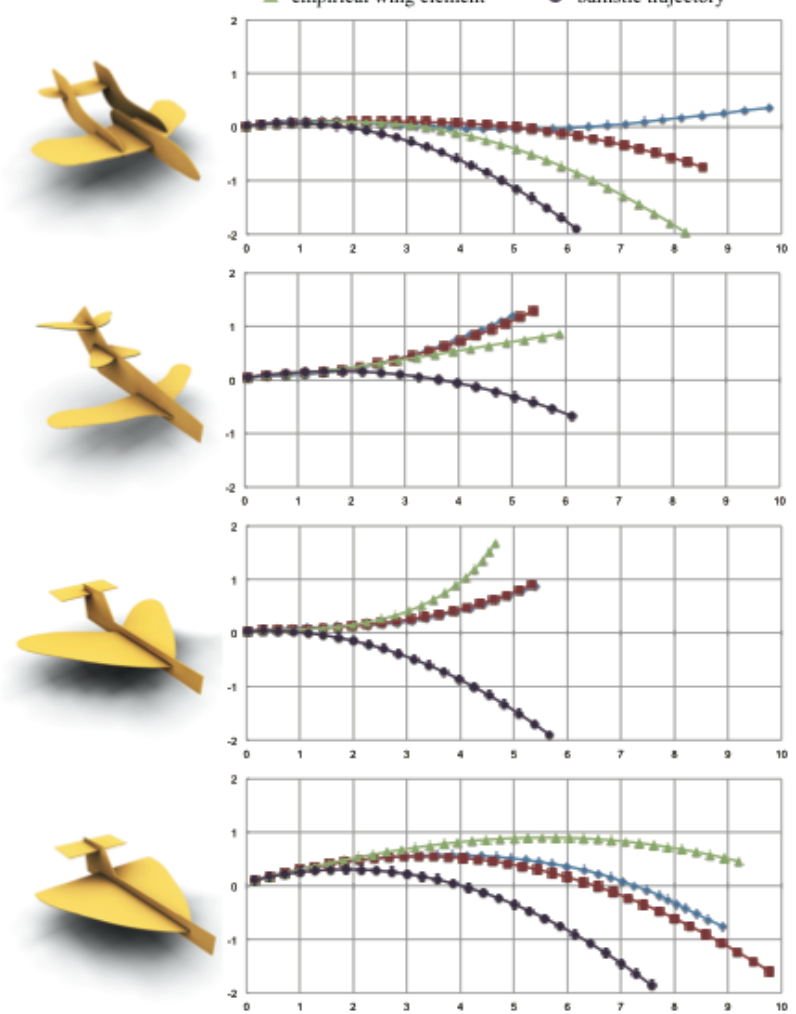


Designing for Mechanical Function

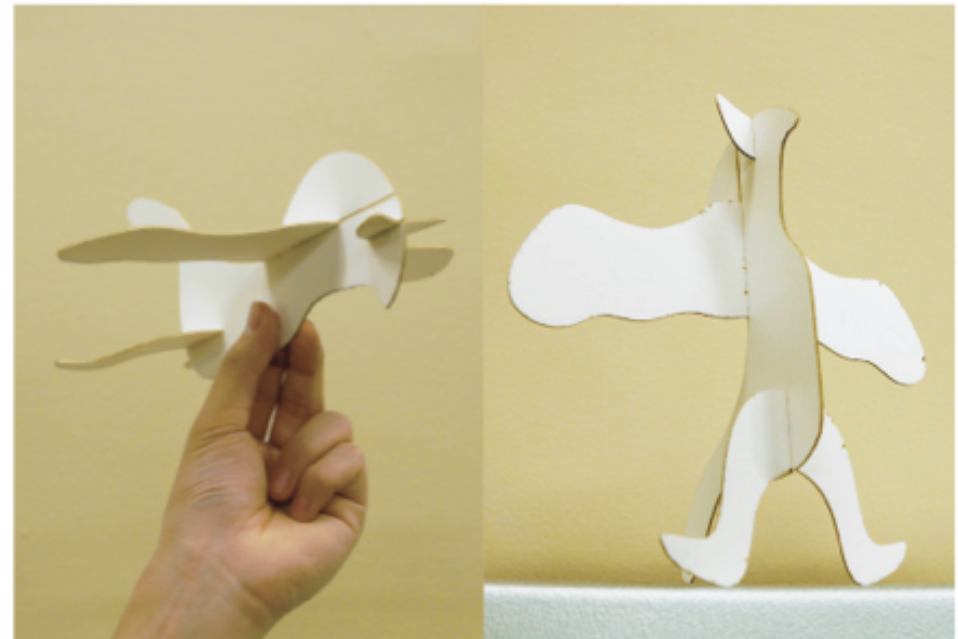
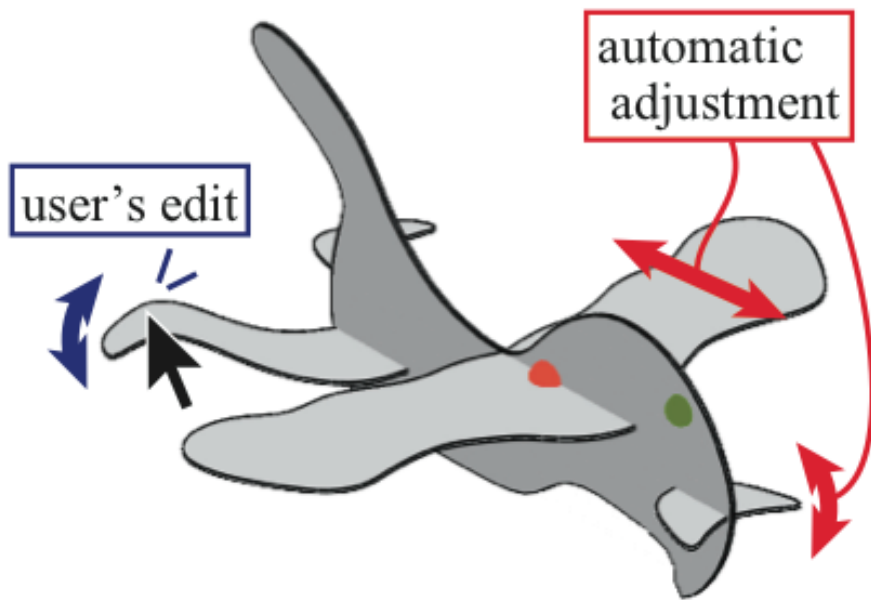
acquired trajectory fitted trajectory empirical wing element



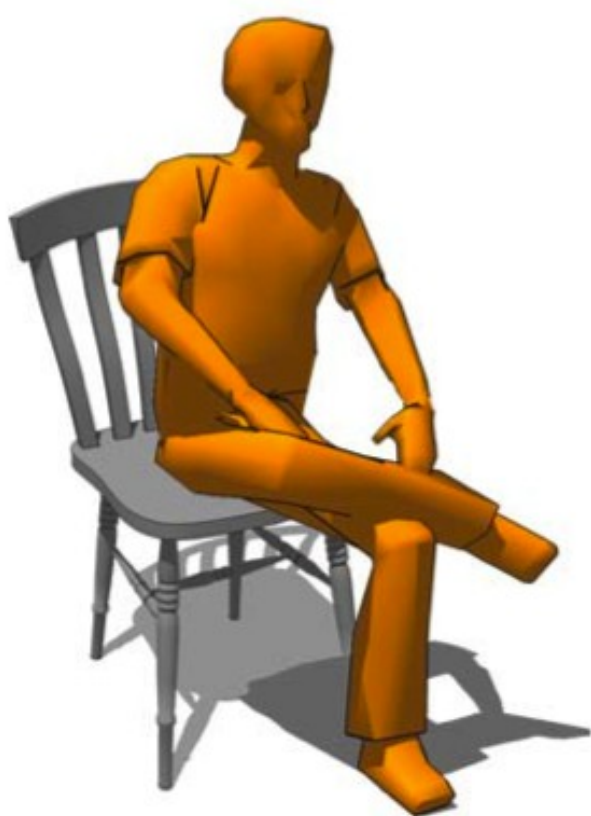
acquired trajectory our simulation empirical wing element ballistic trajectory



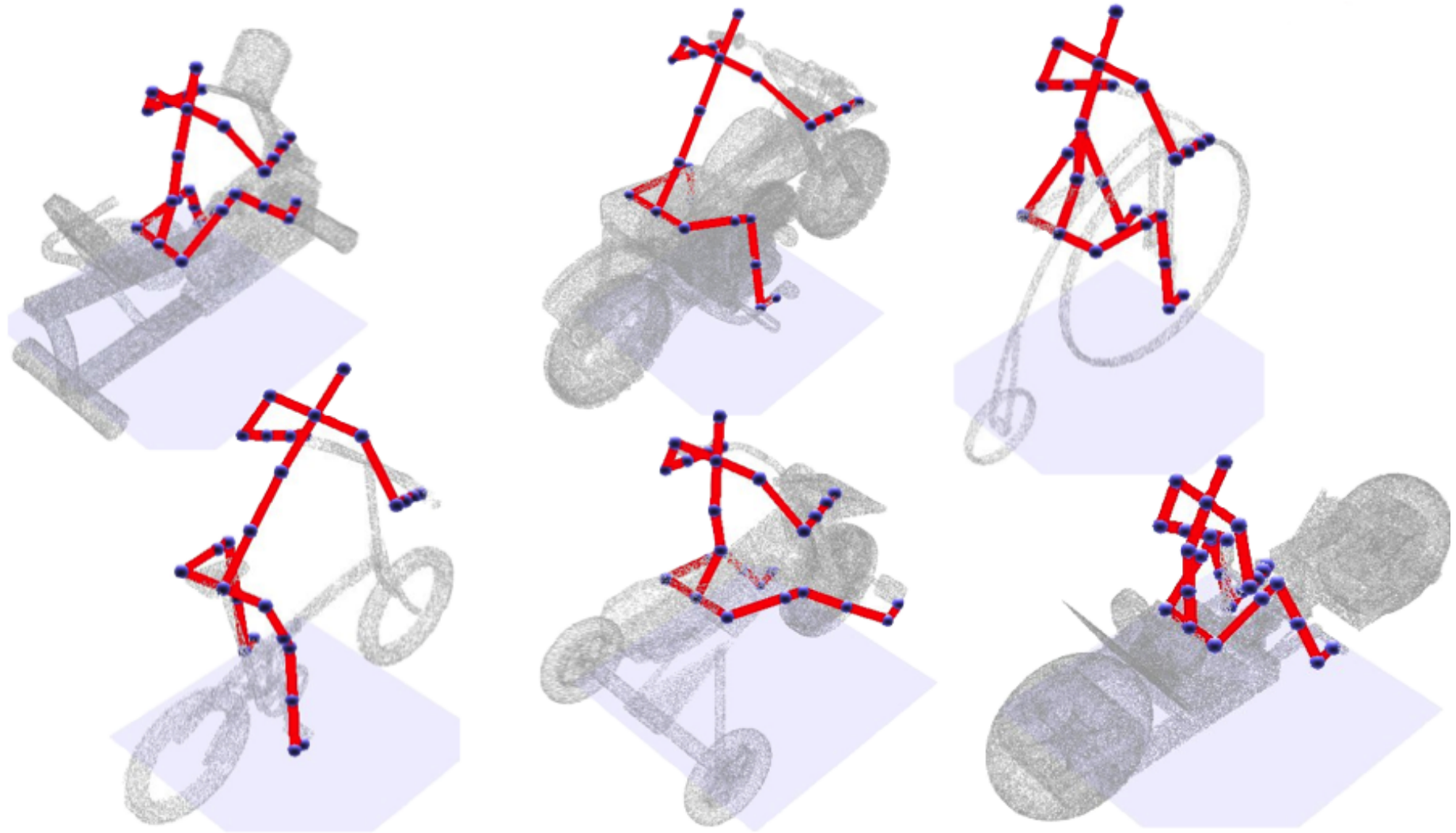
Designing for Mechanical Function



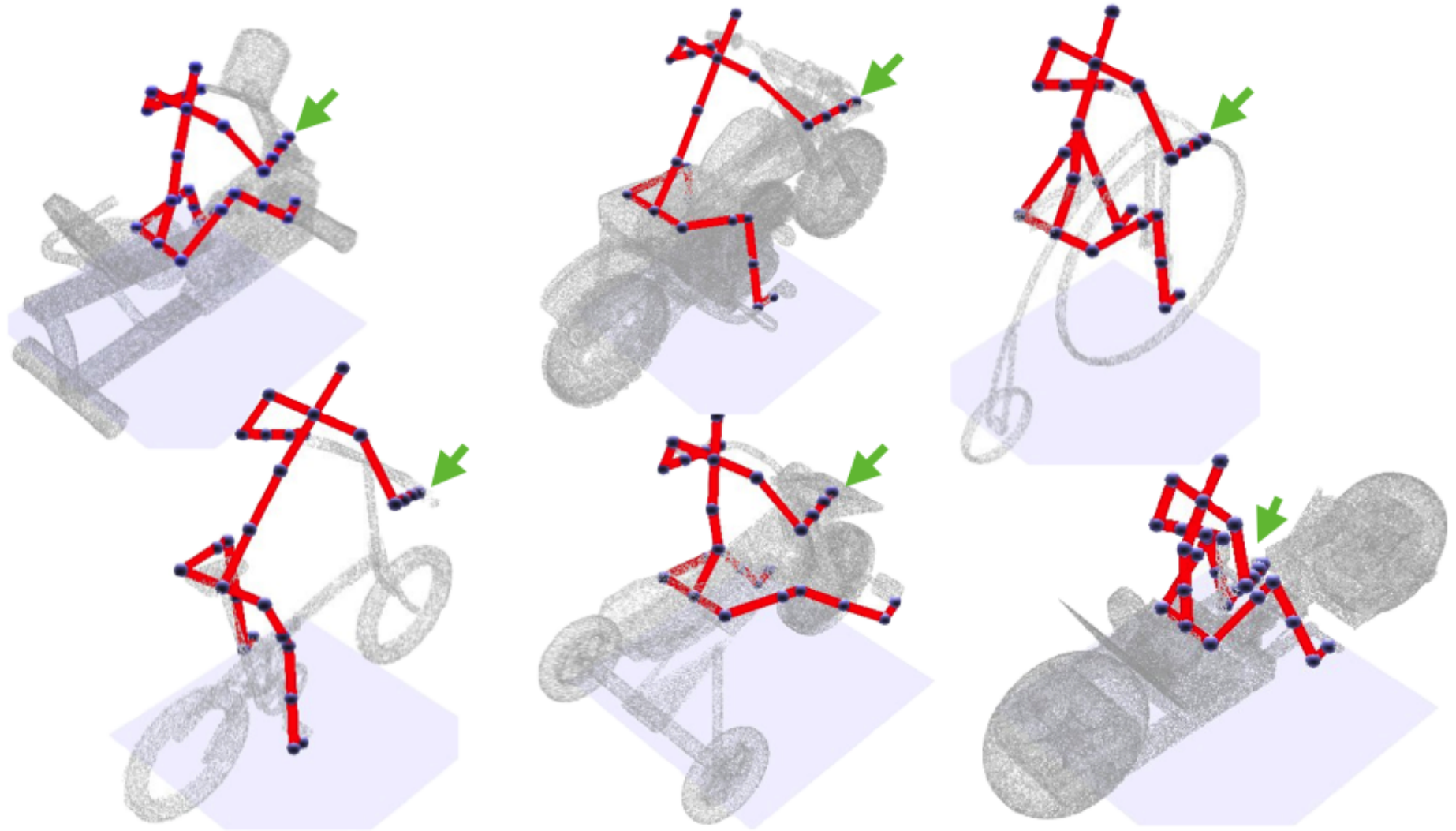
What makes a chair a chair?



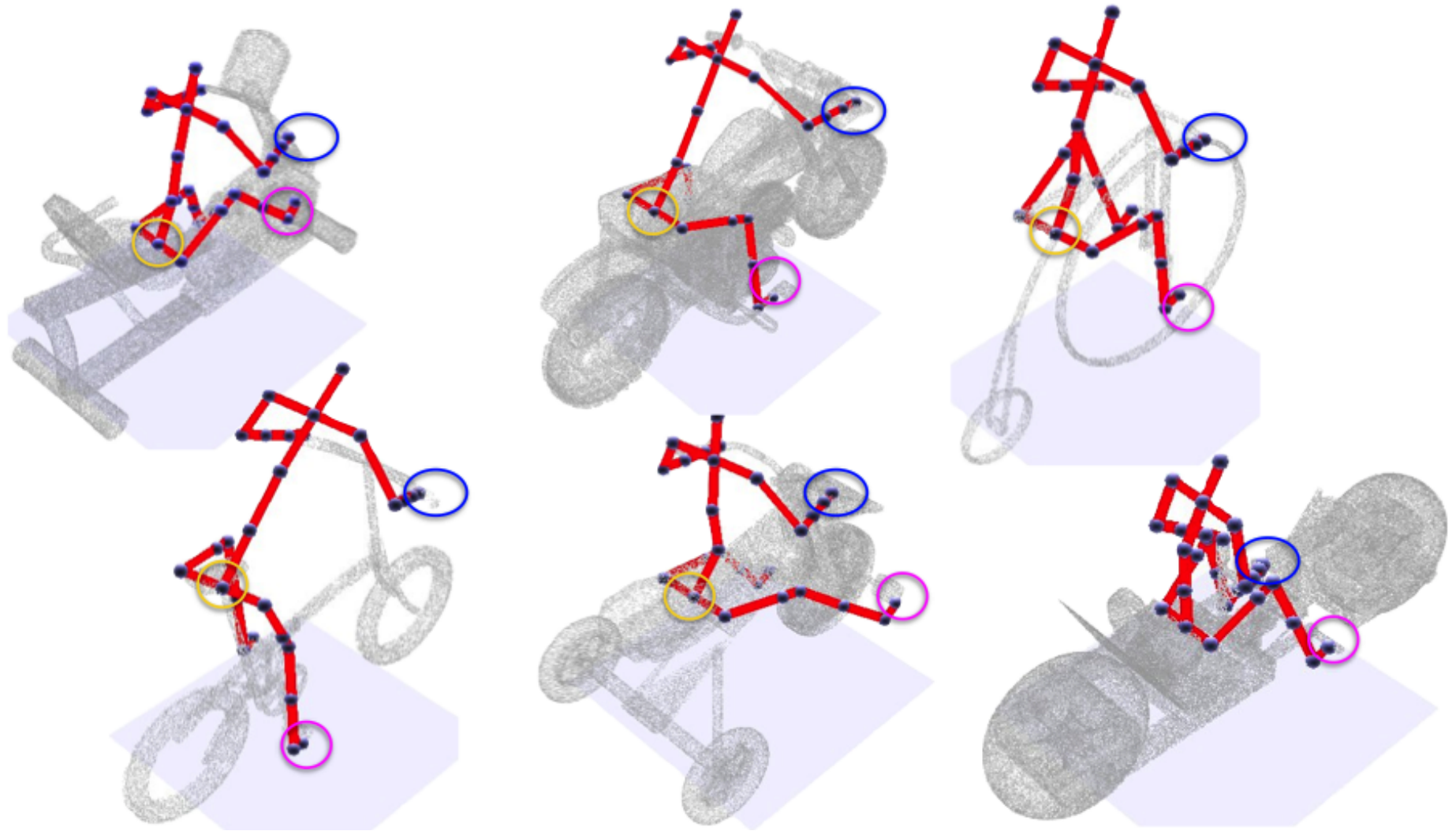
Human-Centric Shape Analysis



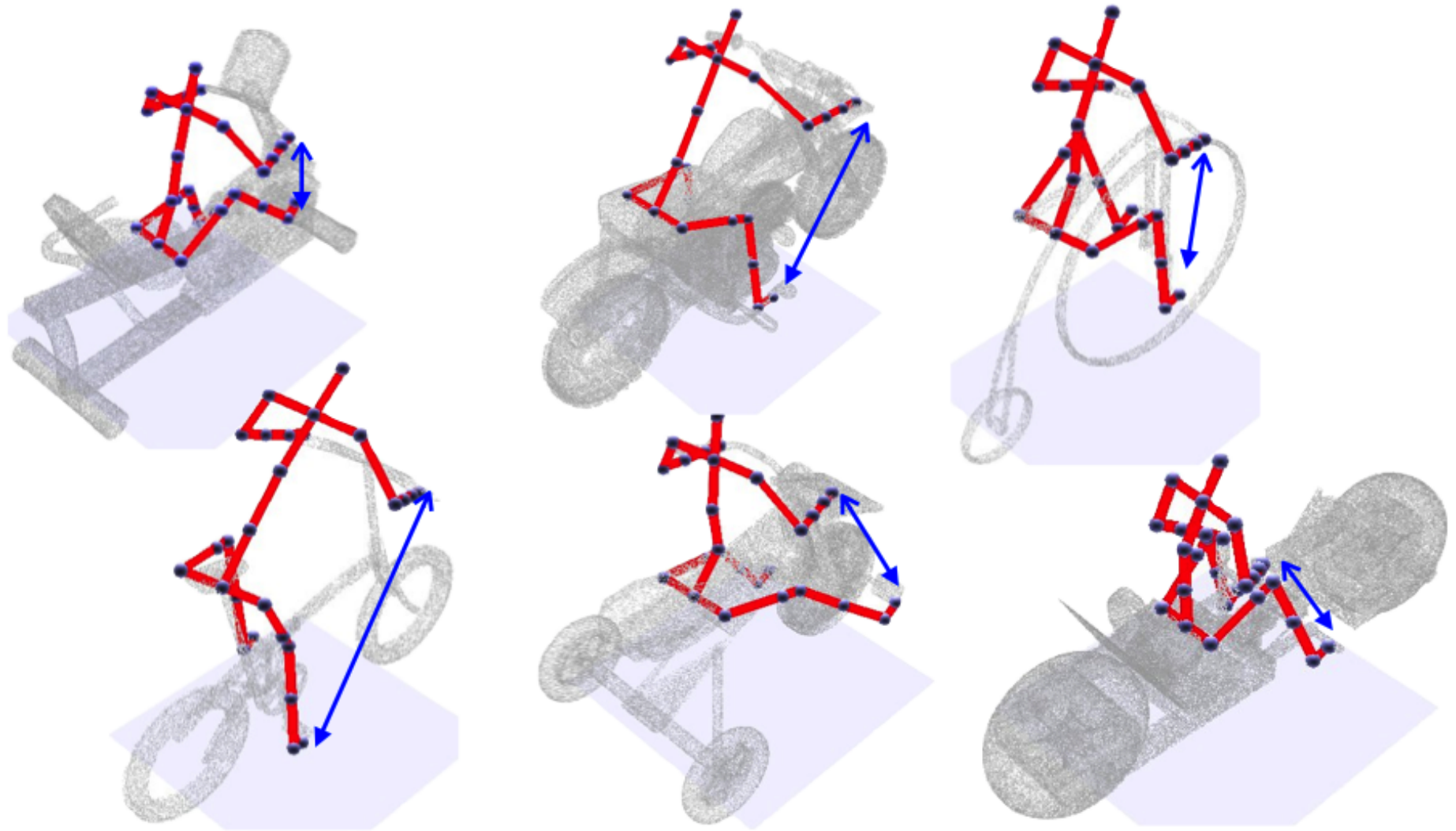
Point-to-Point Correspondences



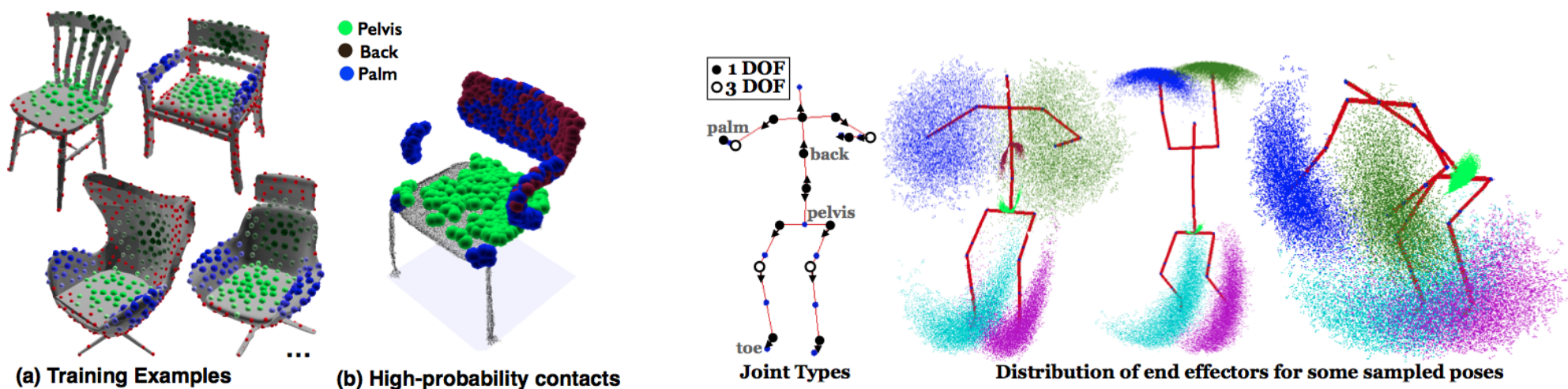
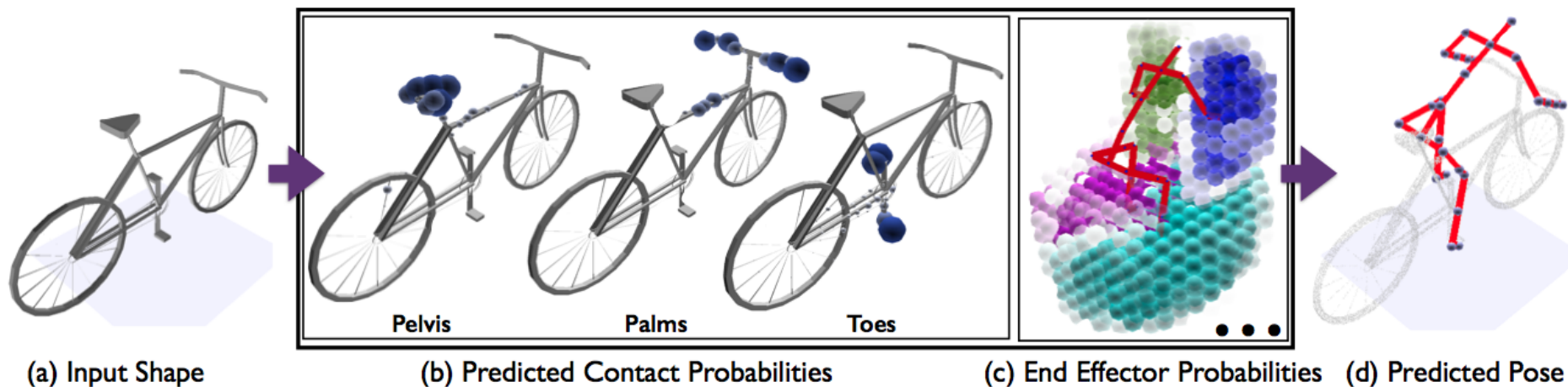
Functional Parts



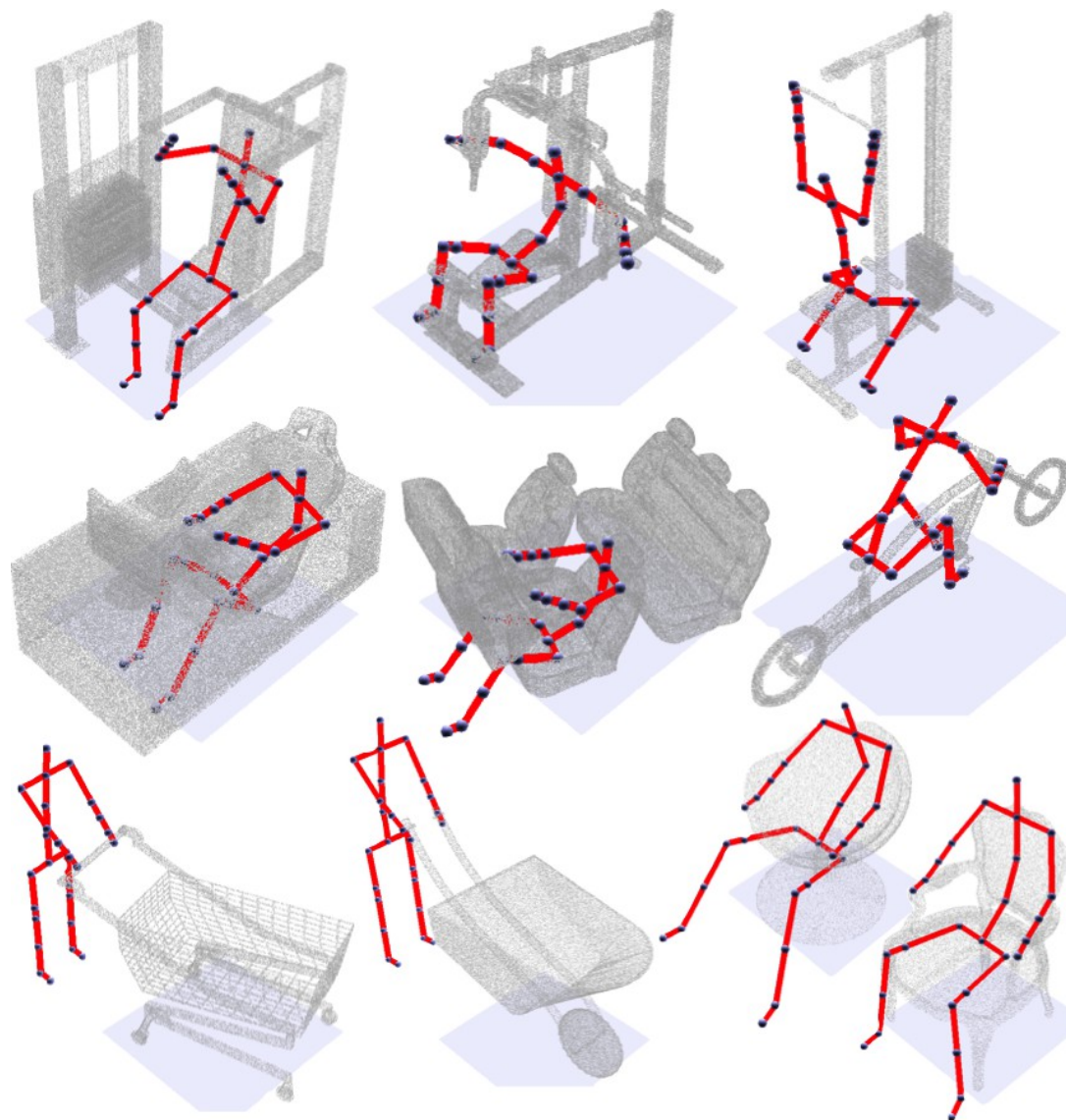
Structural Variations



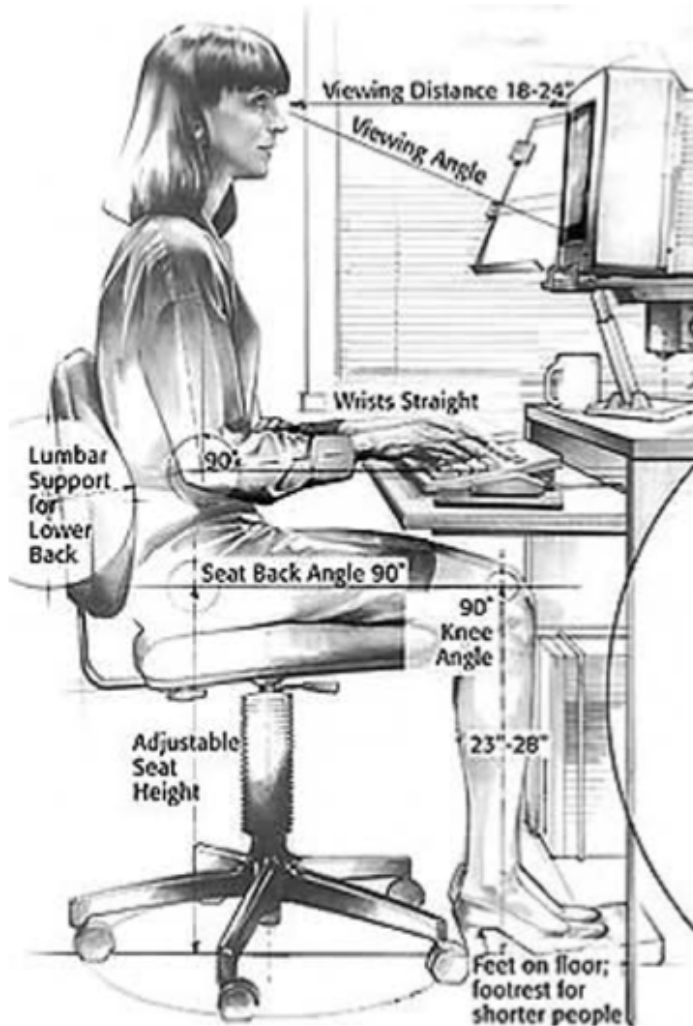
Pose Prediction Pipeline



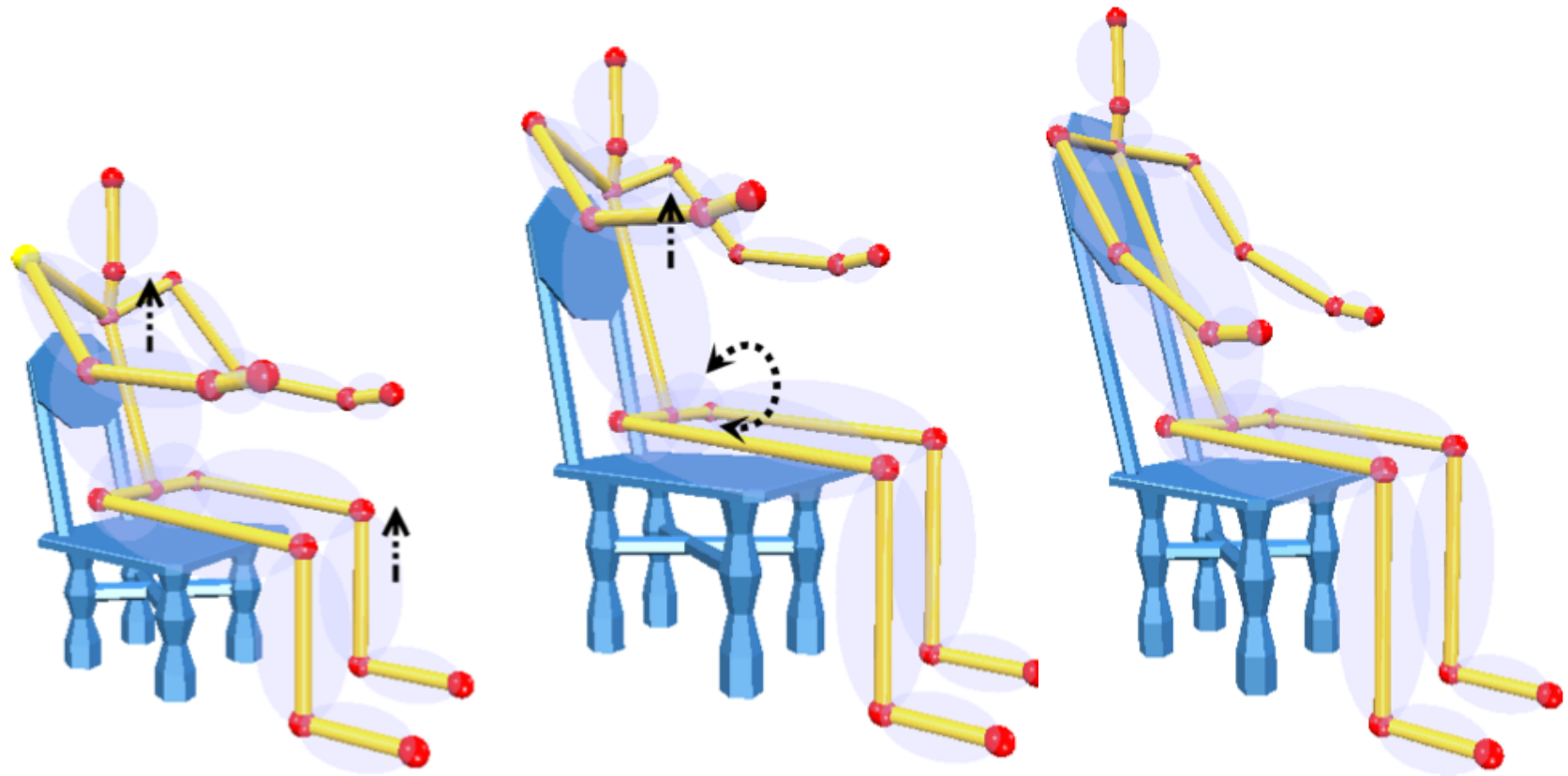
Learning to Predict Human Interaction



Designing for Human Interaction



Shape Adjustment for Body Type



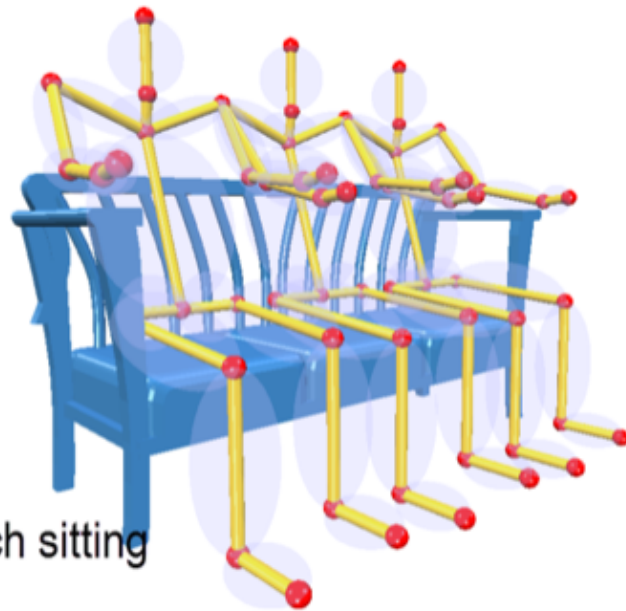
Shape Adjustment for Body Pose



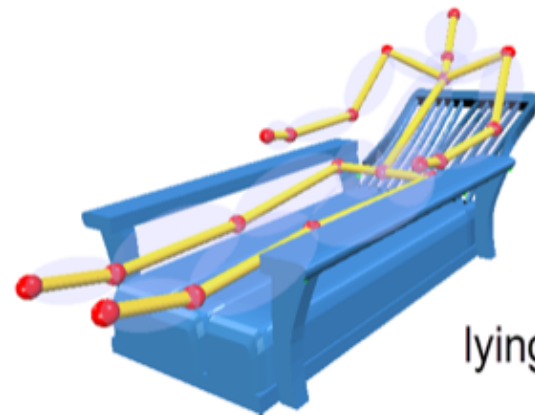
normal sitting



bar sitting



bench sitting



lying

Summary

- **“High-level” geometric analysis**
- **Probabilistic models** can characterize the structure of “plausible” objects, and generate new ones
- Design intent can be captured through **semantic attributes, mechanical function** and **human interaction**
- Models of structure, attributes, function and interaction can be automatically learned from **(big) data**