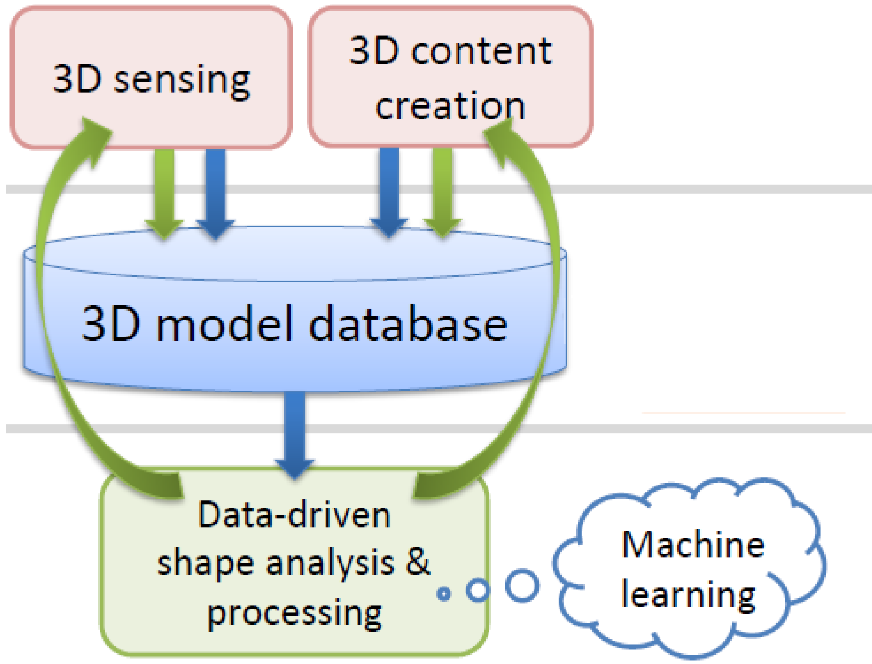
Machine Learning for Shape Analysis and Processing



Evangelos Kalogerakis

UMASS AMHERST

3D shapes for computer-aided design

Architecture

Interior design

3D shapes for information visualization

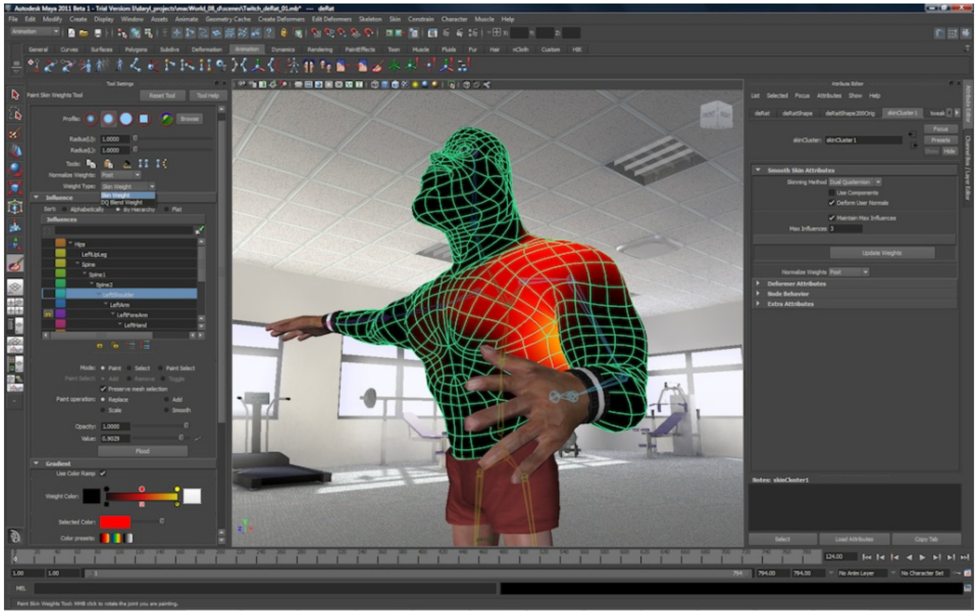
Geo-visualization

Scientific visualization

3D shapes for digital entertainment

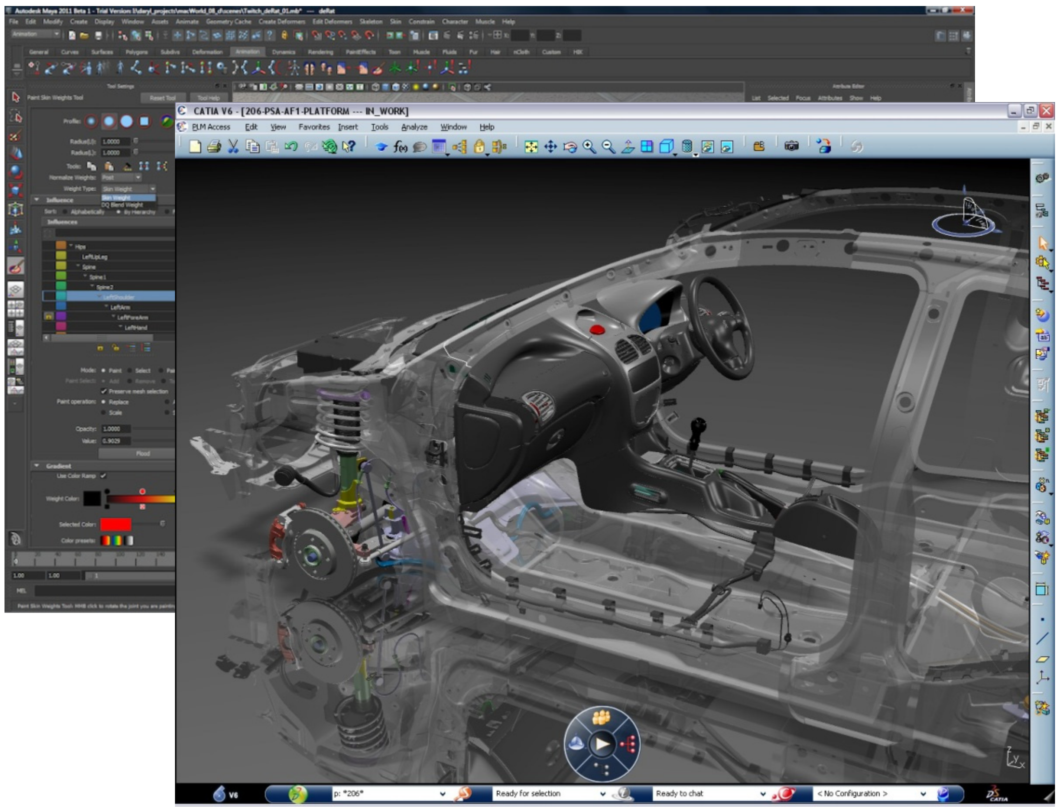
Video games

Digitizing our imagination



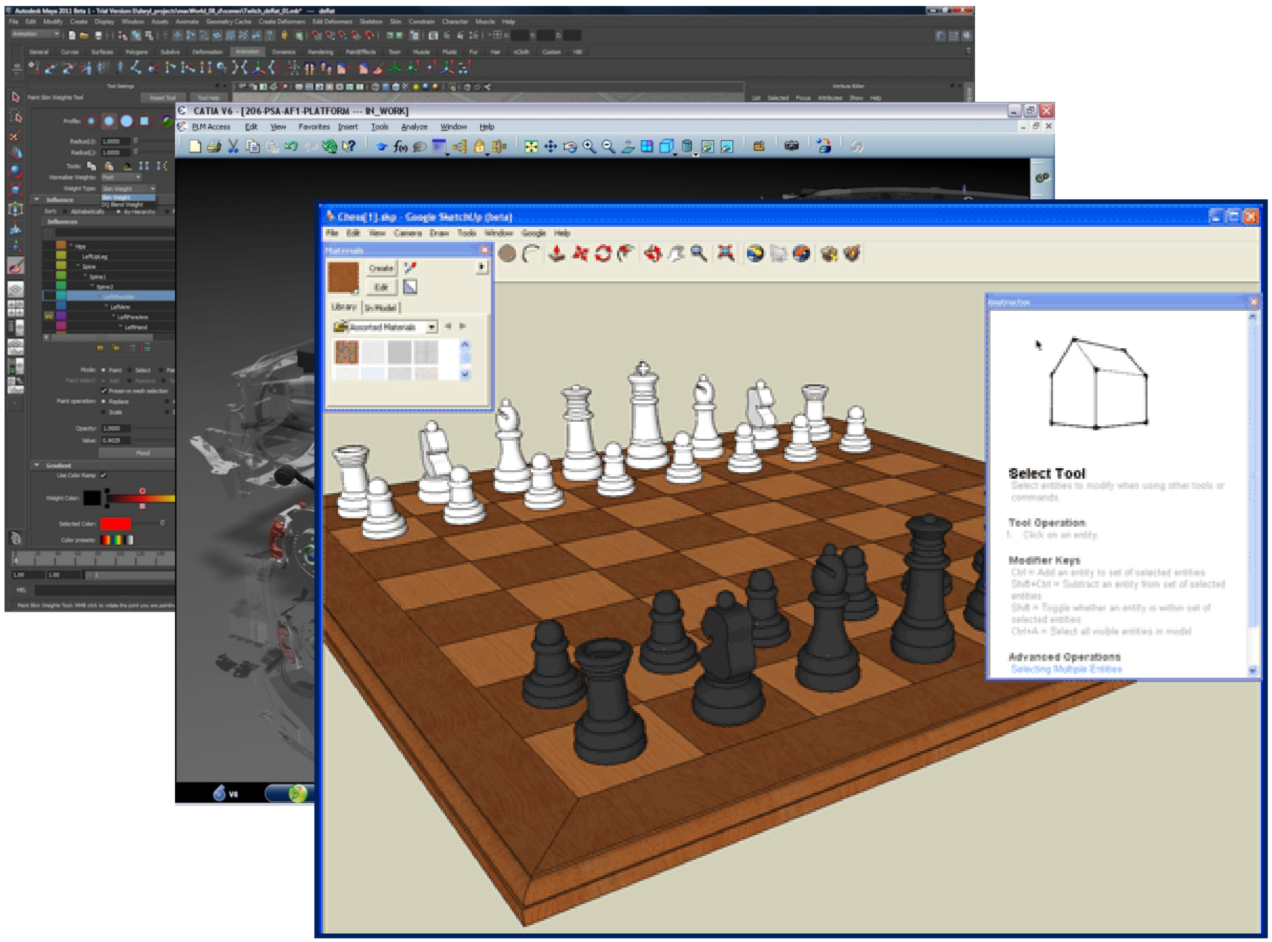
Professional 3D modeling tools [Autodesk Maya]

Digitizing our imagination



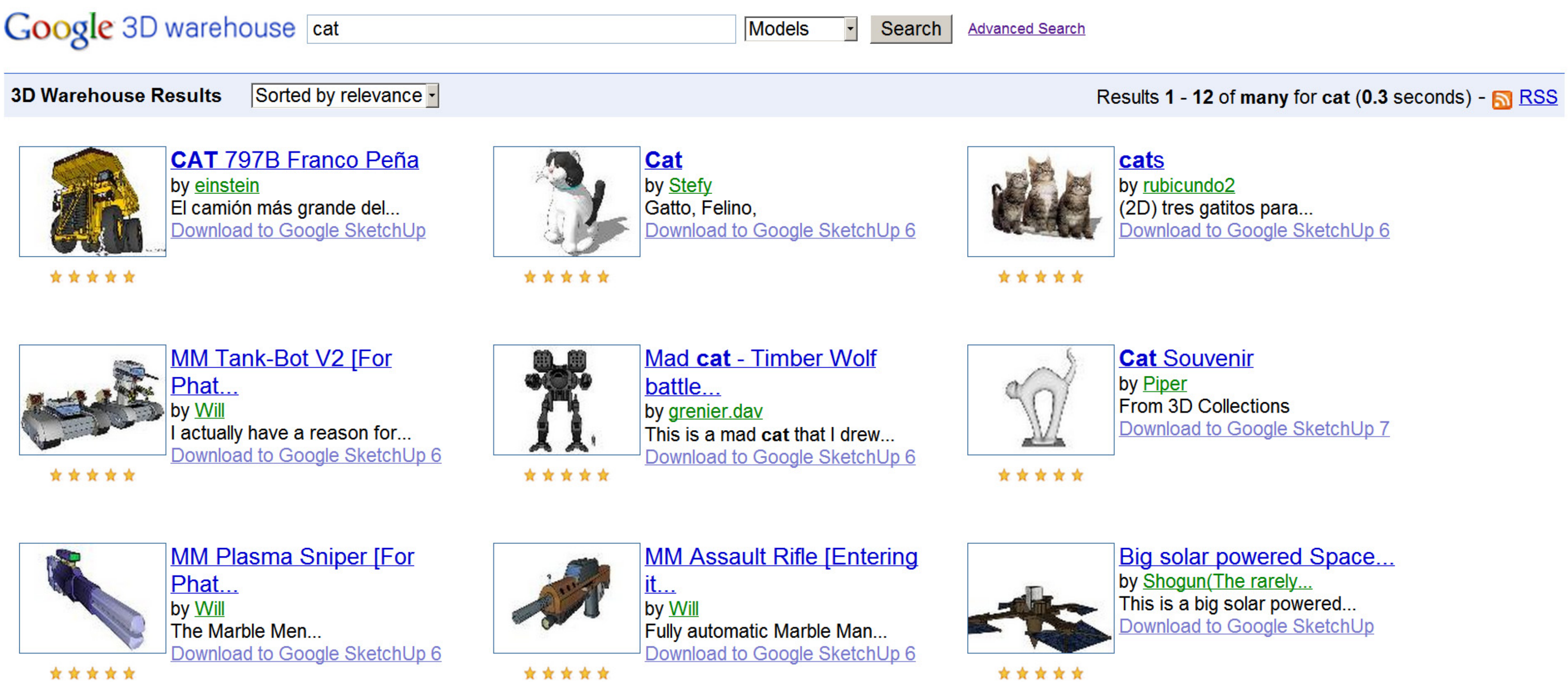
Computer-Aided Design tools
[Catia]

Digitizing our imagination



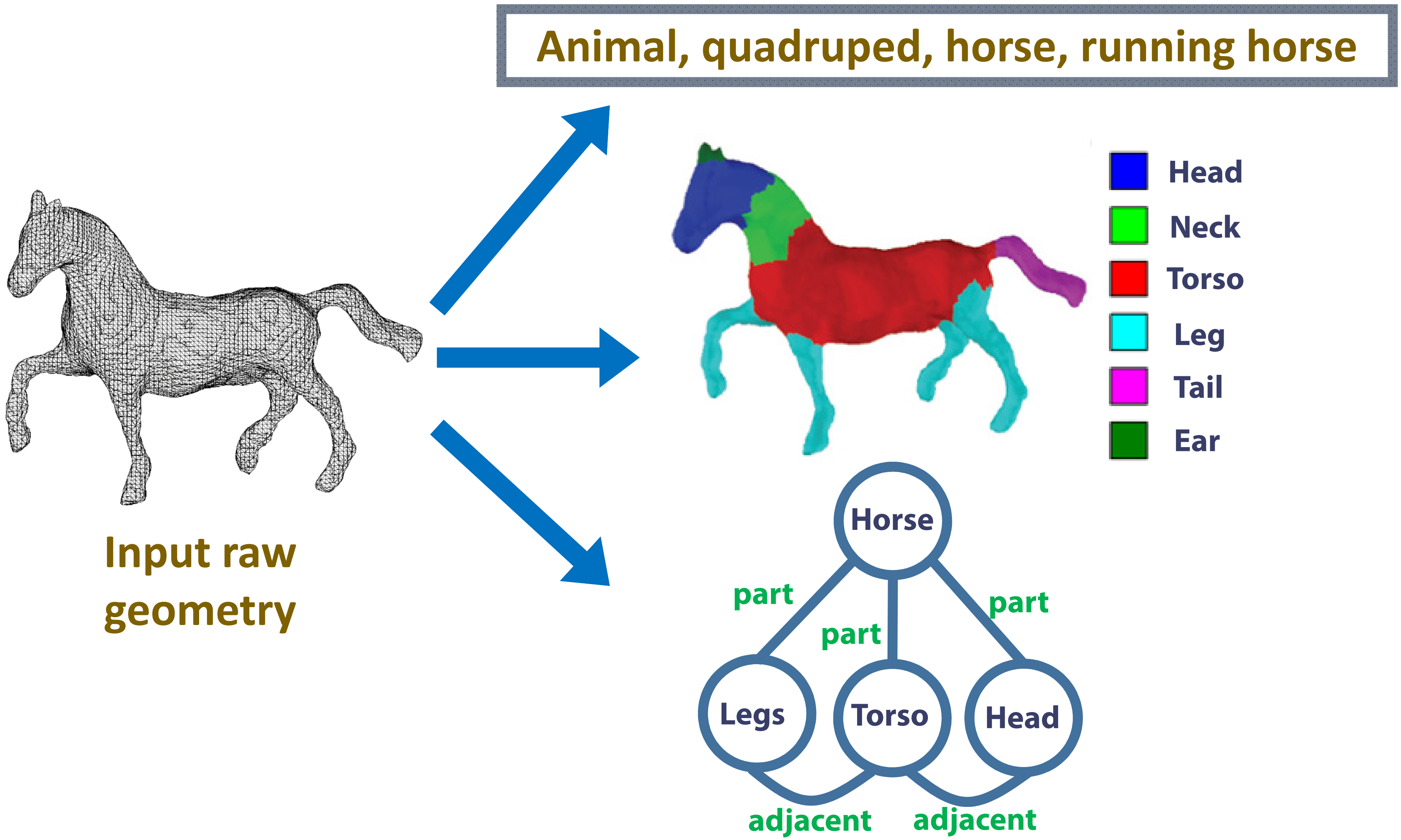
General-Purpose Modeling tools [Sketch-up]

3D shape repositories

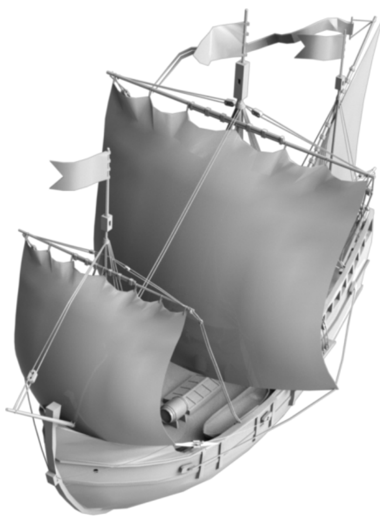


[Trimble Warehouse]

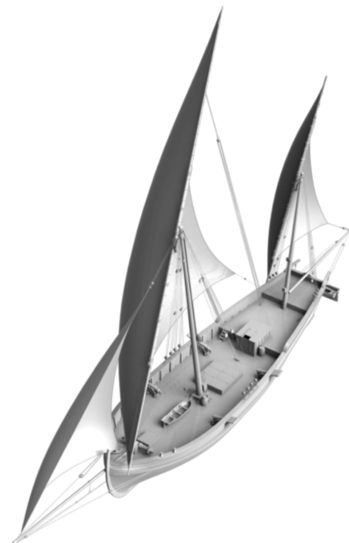
Shape understanding



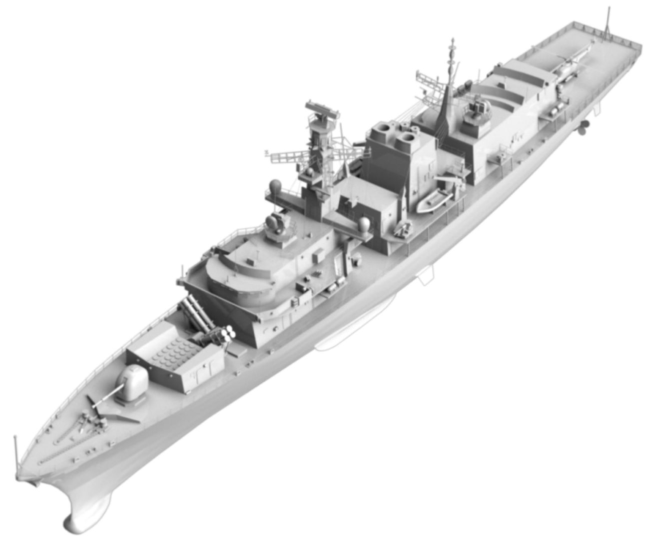
Why shape understanding?
Shape categorization



Sailing Ship, Galleon



Sailing ship, Yawl

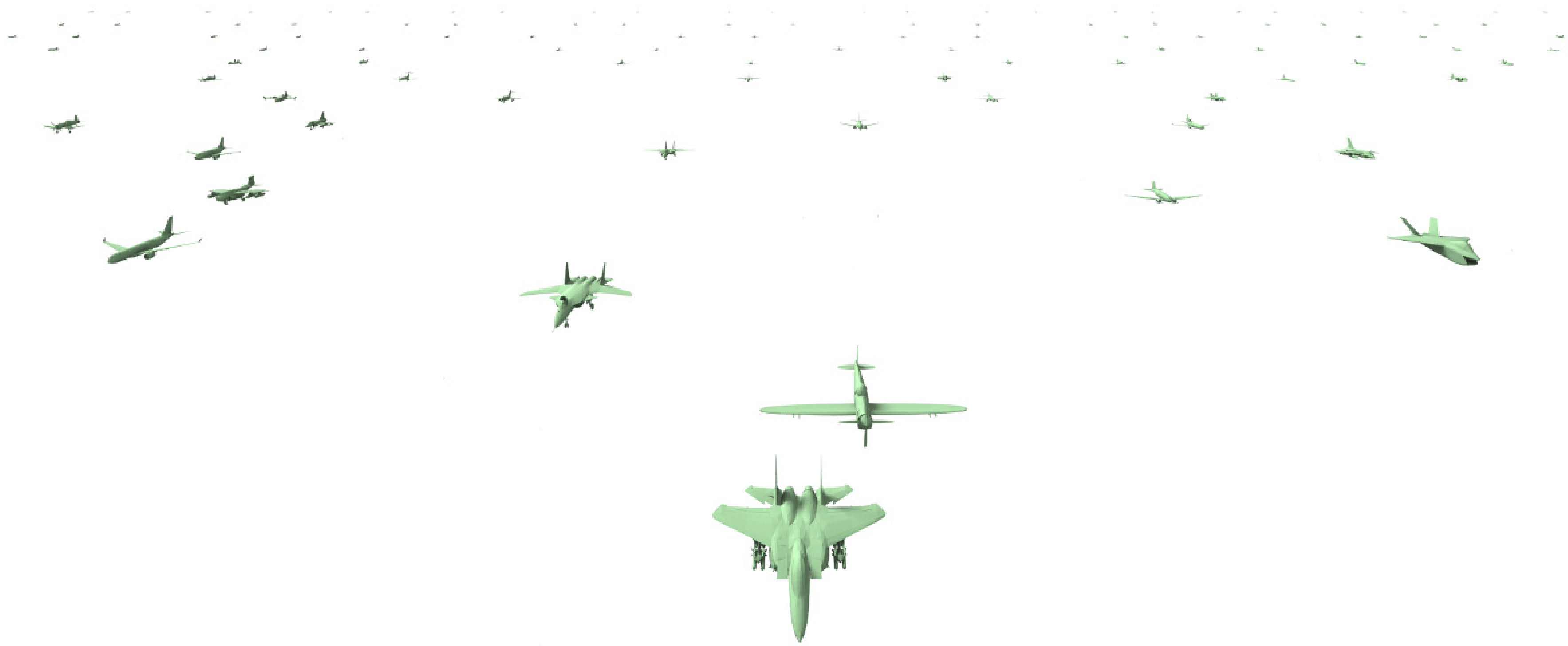


Military ship, Frigate

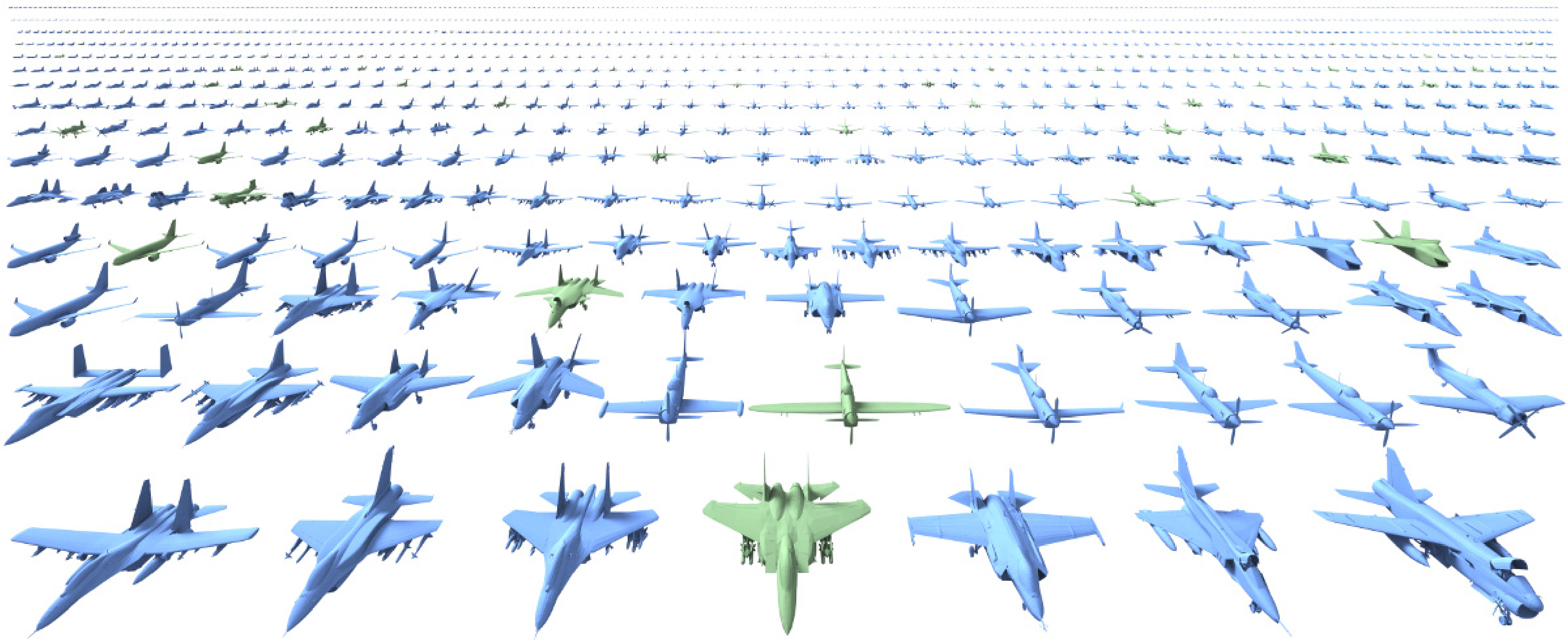
Why shape understanding?
3D Modeling

Chaudhuri, Kalogerakis, Guibas, Koltun, SIGGRAPH 2011
access video: https://www.youtube.com/watch?v=7Abki79WIOY

Why shape understanding?
Shape synthesis



Why shape understanding?
Shape synthesis

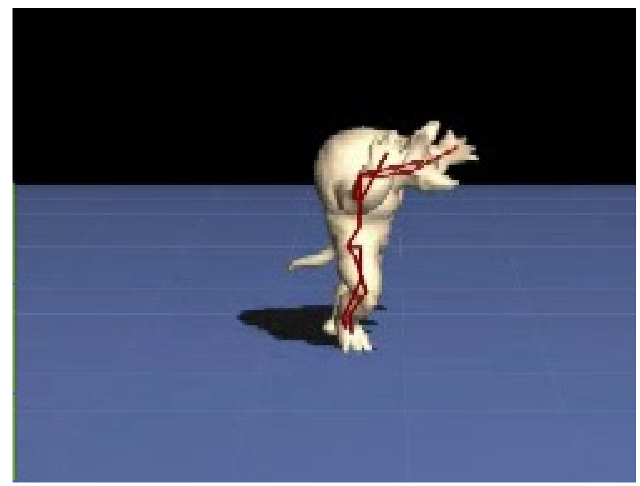


Kalogerakis, Chaudhuri, Koller, Koltun, SIGGRAPH 2012

Why shape understanding?
Texturing

Kalogerakis, Hertzmann, Singh, SIGGRAPH 2010

Why shape understanding?
Character Animation

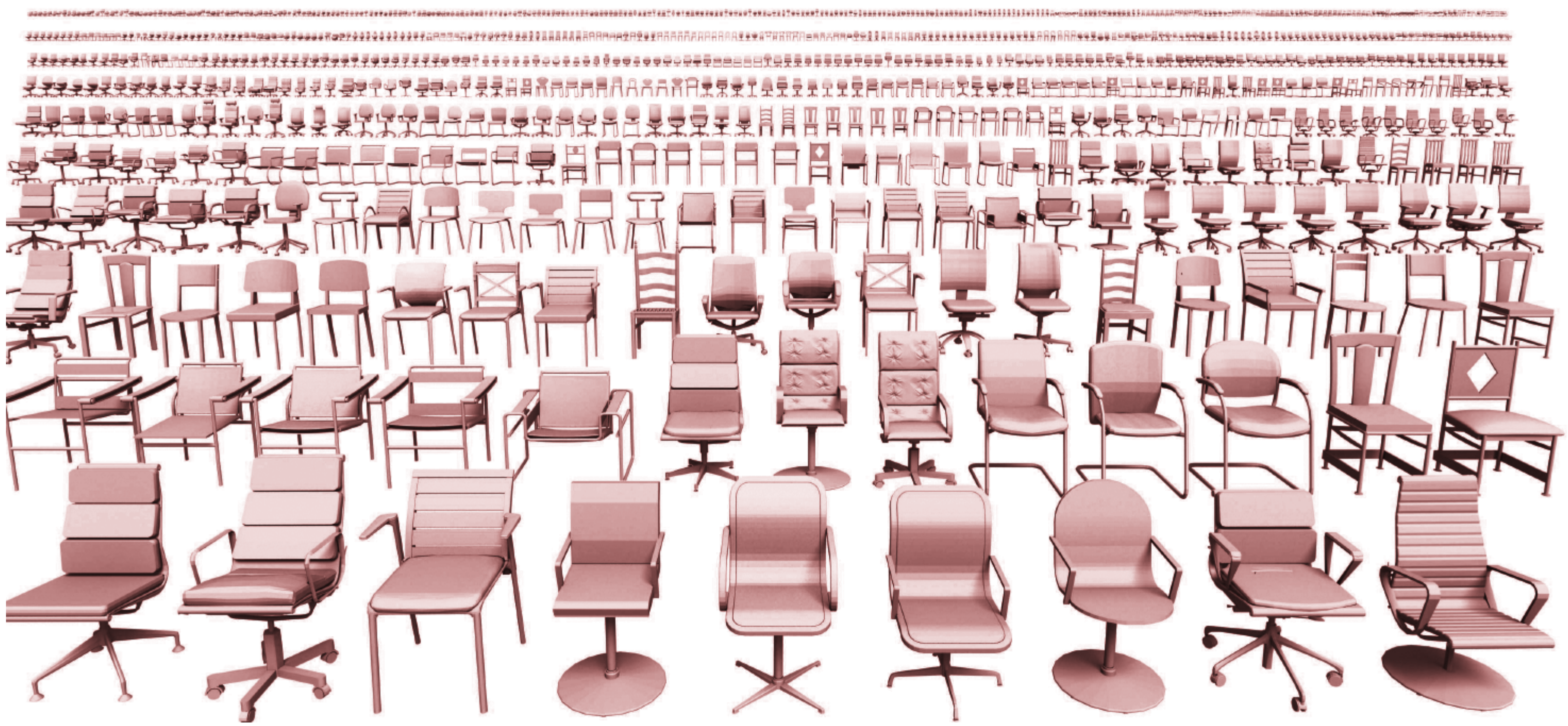


Simari, Nowrouzezahrai, Kalogerakis, Singh, SGP 2009

Why data-driven methods for shape understanding?

It is extremely hard to perform shape understanding with a set of deterministic, manually specified rules!

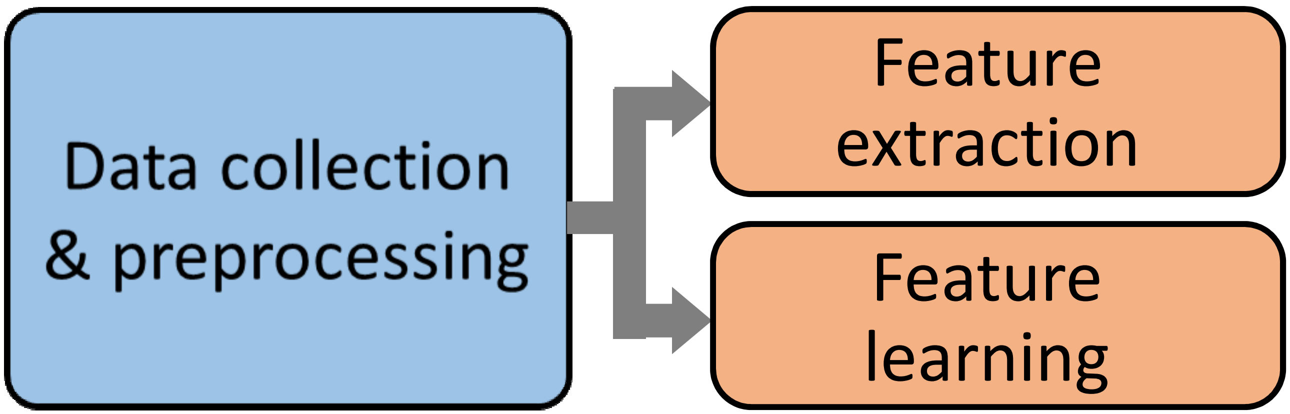
We should not treat shapes in complete isolation of all others.



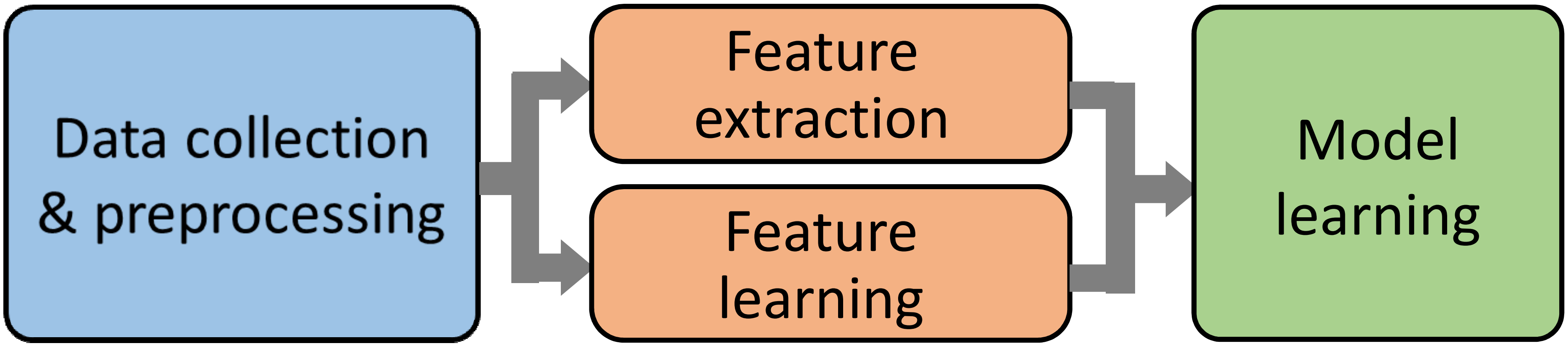
Pipeline of data-driven methods

Data collection
& preprocessing

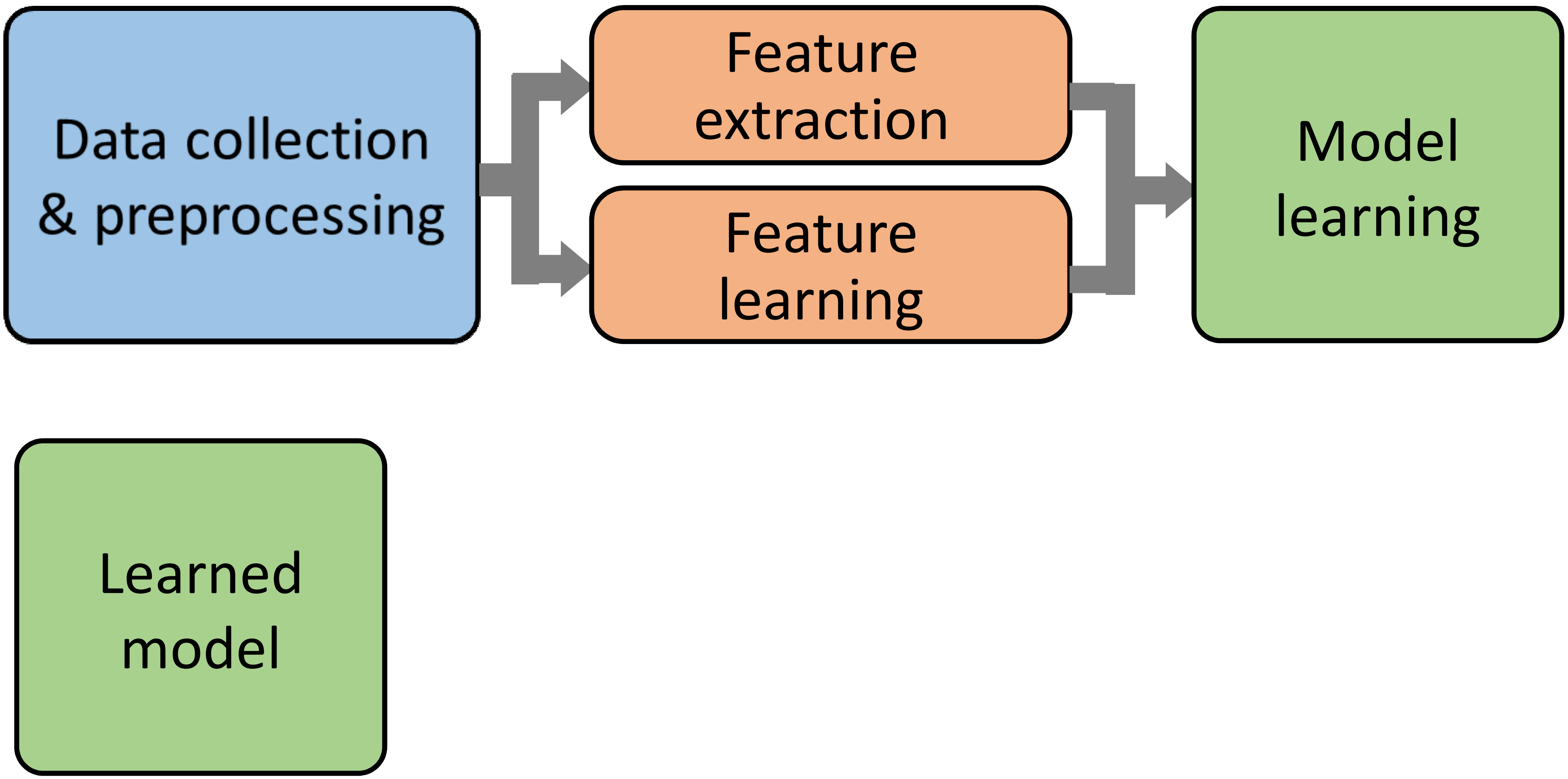
Pipeline of data-driven methods



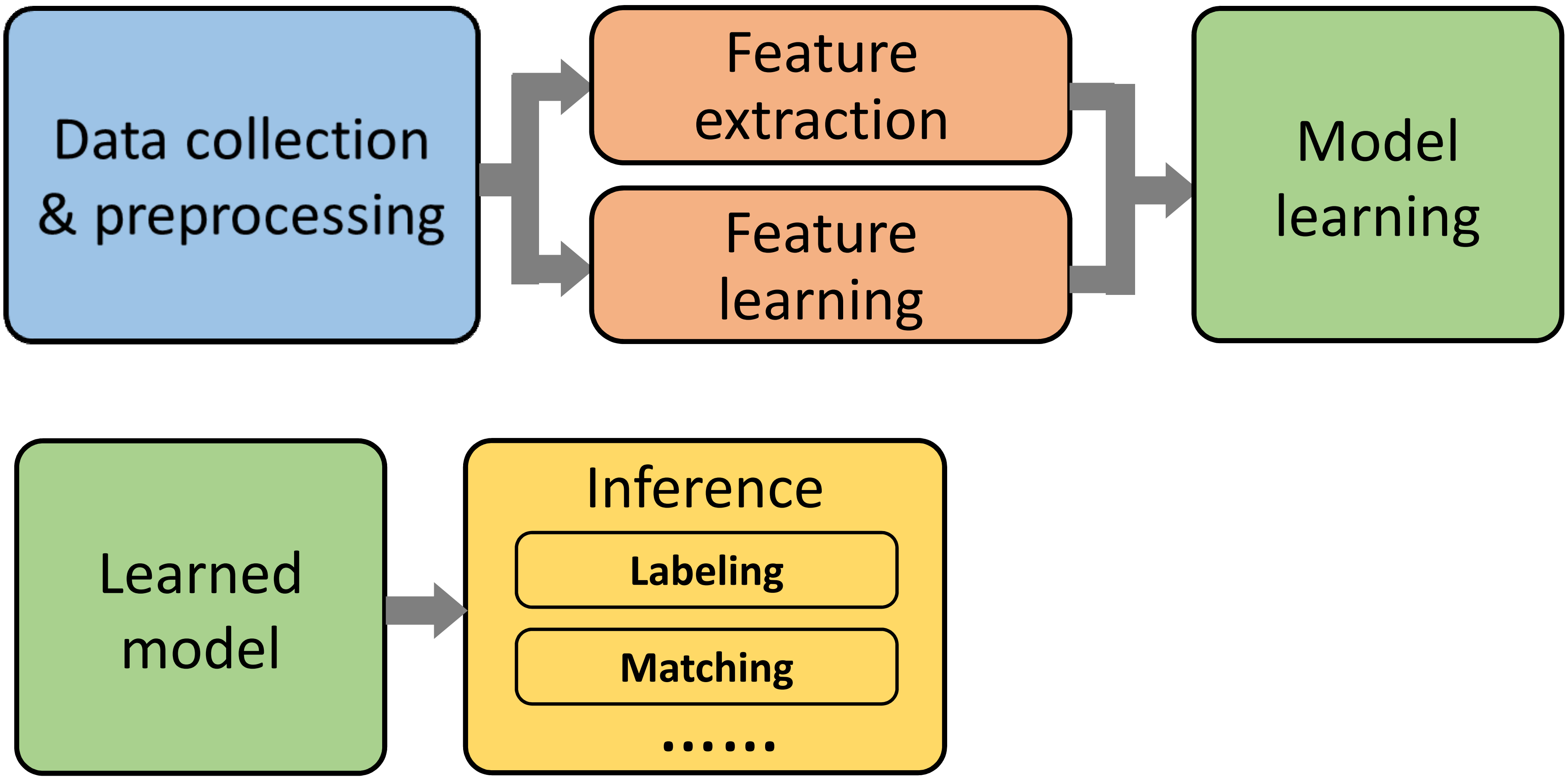
Pipeline of data-driven methods



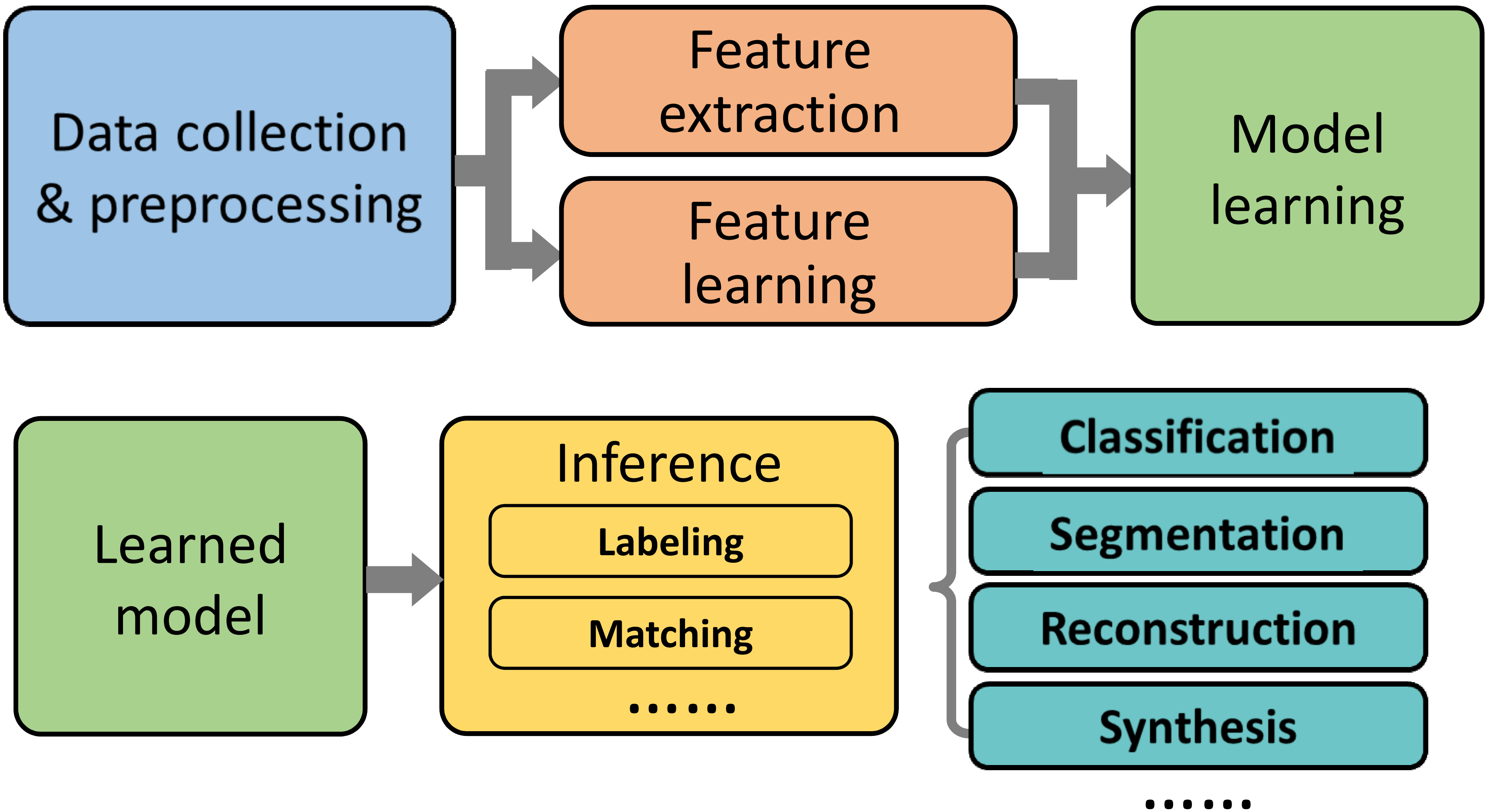
Pipeline of data-driven methods



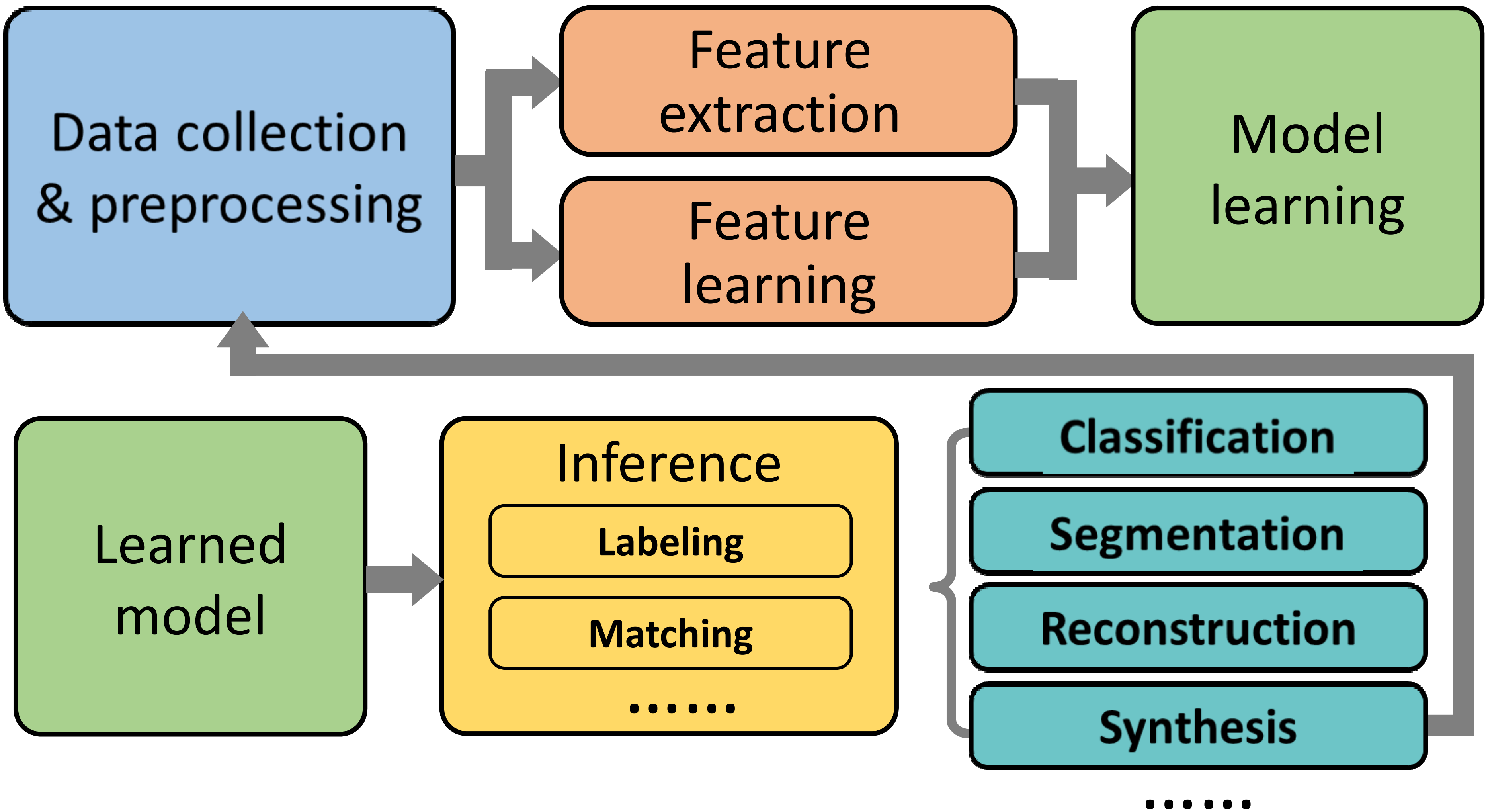
Pipeline of data-driven methods



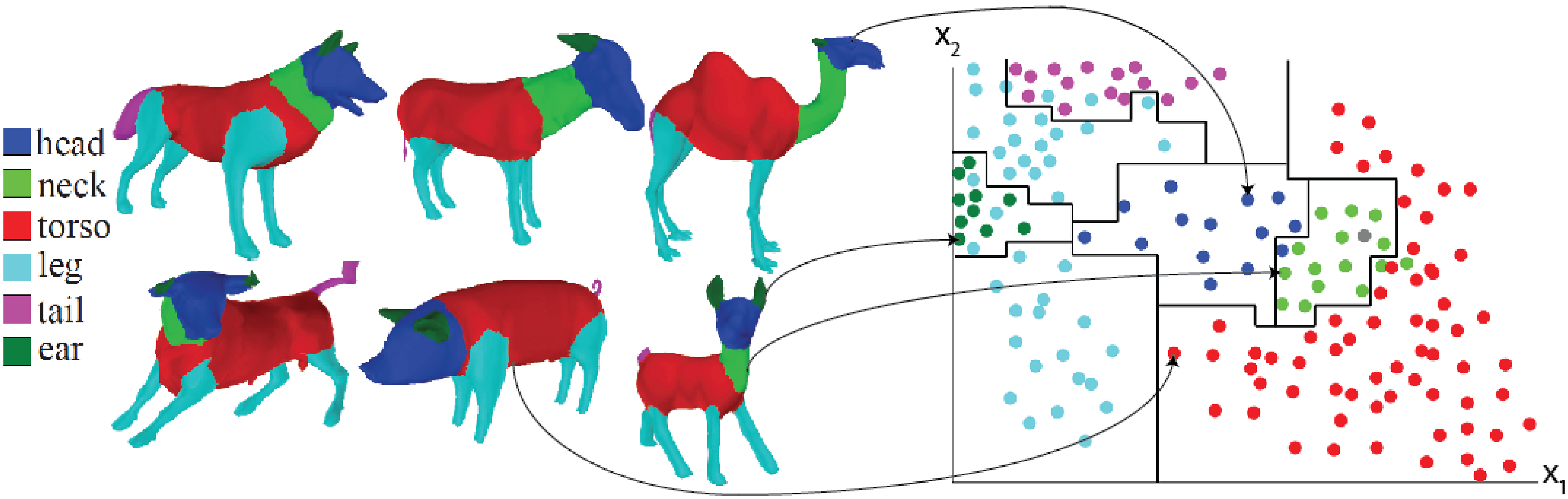
Pipeline of data-driven methods



Pipeline of data-driven methods



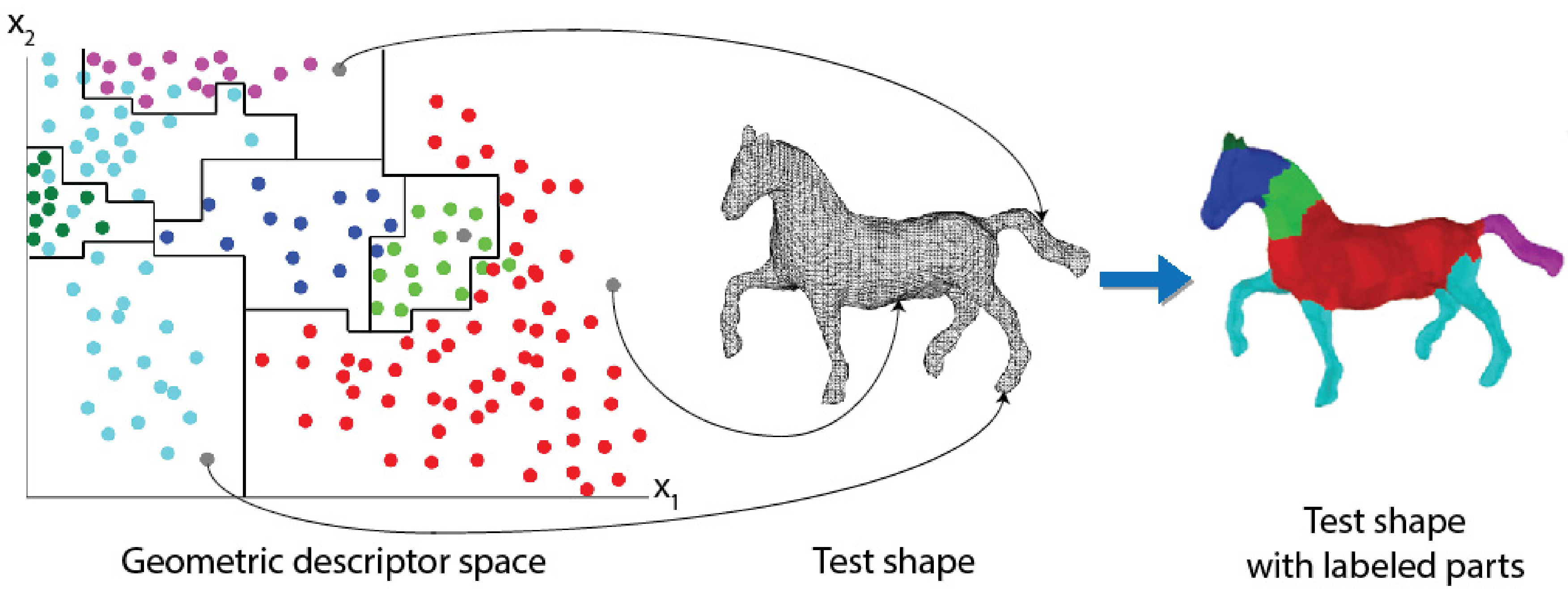
Example: part labeling [**training stage**]



Training shapes
with labeled parts

Geometric descriptor space

Example: part labeling [**test stage**]



STAR report

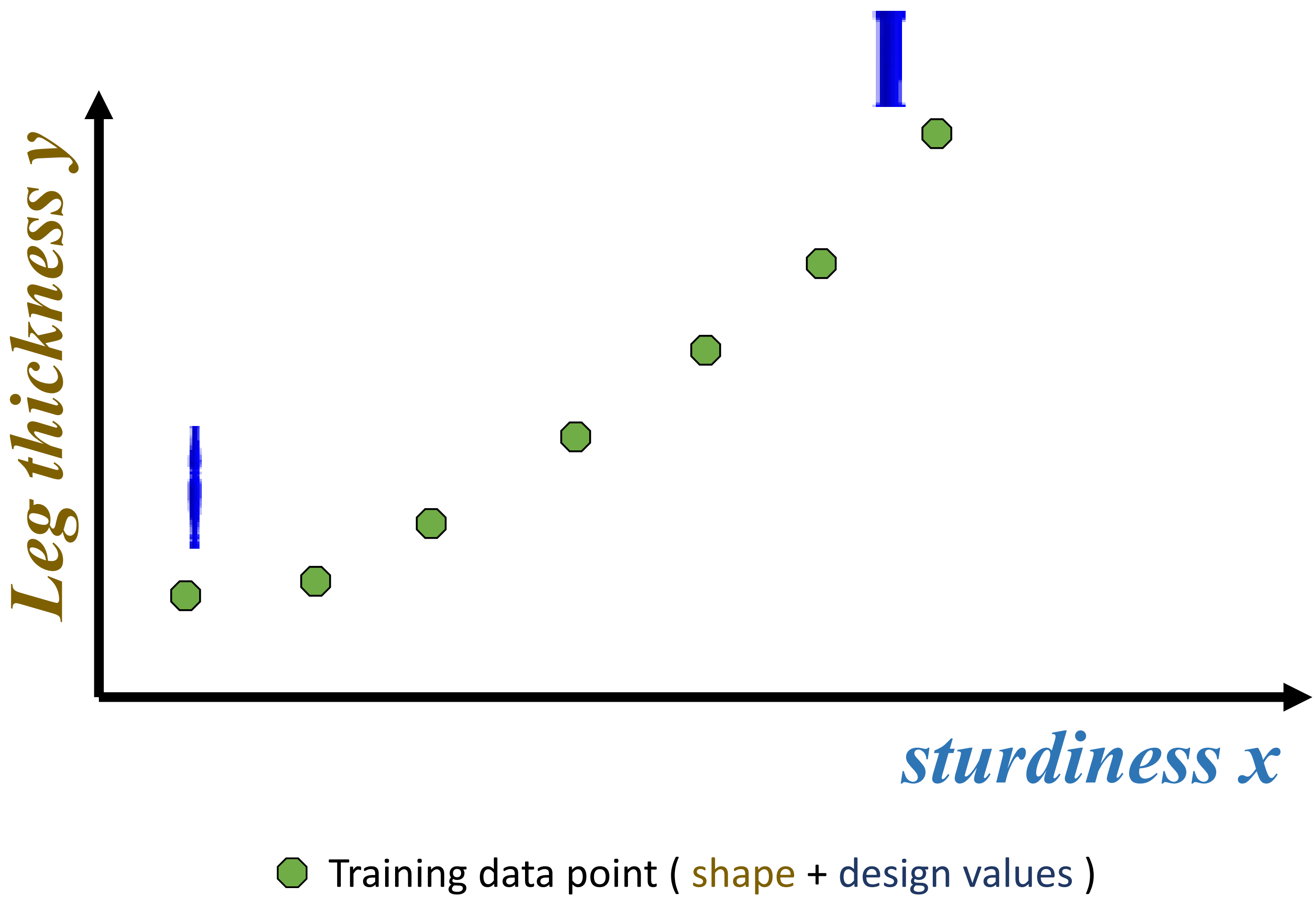
Data-Driven Shape Analysis and Processing, CGF STAR report
Kai Xu, Vladimir Kim, Qixing Huang, Evangelos Kalogerakis
Overview of 200+ works published in the field, 30 pages. Enjoy!

1. Pipeline of data-driven shape processing techniques
2. Learning
3. Shape classification and retrieval
4. Shape segmentation
5. Shape correspondences
6. Shape reconstruction
7. Shape modeling and synthesis
8. Scene analysis and synthesis
9. Exploration and organization of 3D model collections

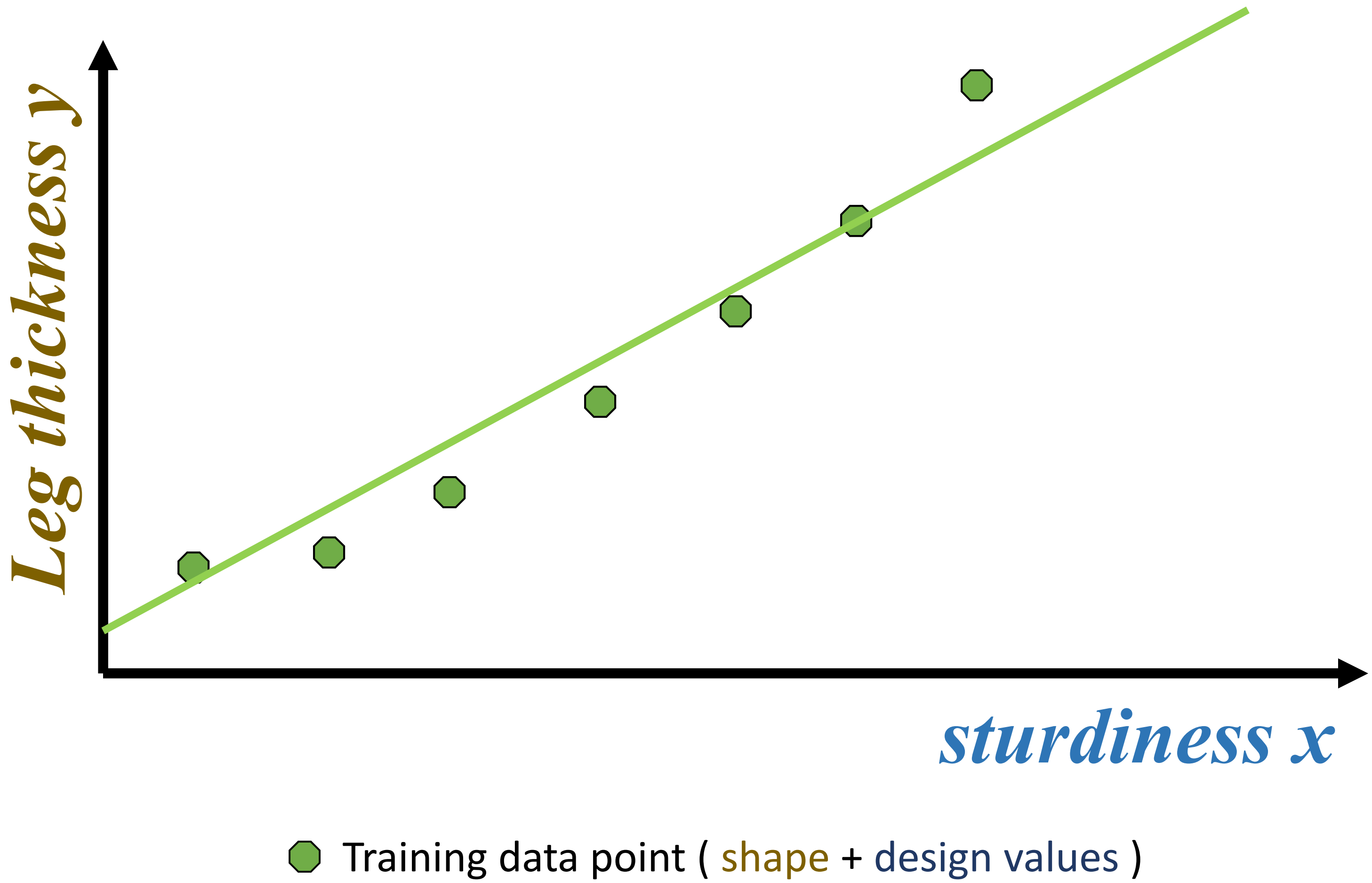
In this talk....

1. **Pipeline of data-driven shape processing techniques**
2. **Learning**
3. **Shape classification and retrieval**
4. Shape segmentation
5. Shape correspondences
6. Shape reconstruction
7. **Shape modeling and synthesis**
8. Scene analysis and synthesis
9. Exploration and organization of 3D model collections

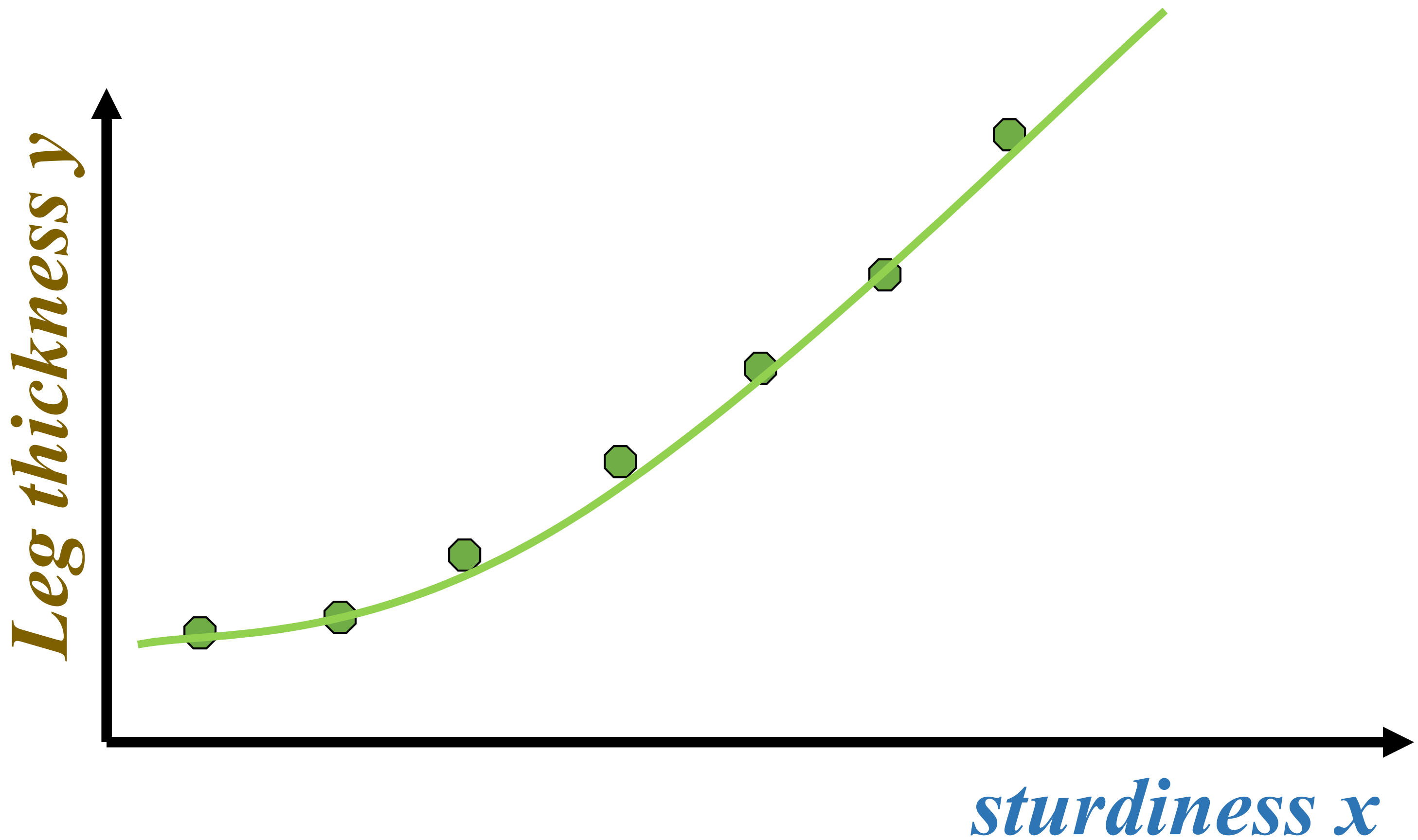
Learning basics: regression



Learning basics: regression

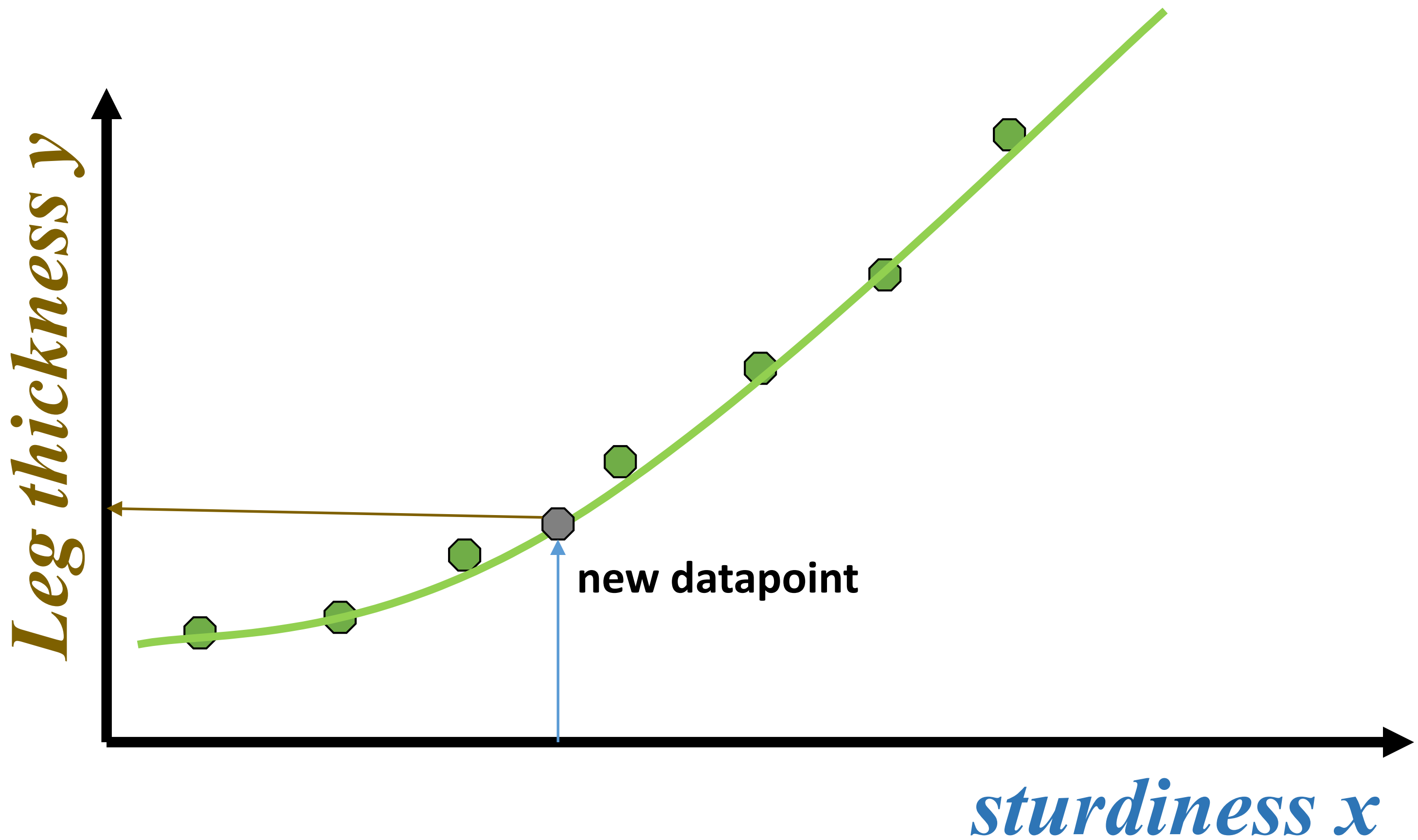


Learning basics: regression



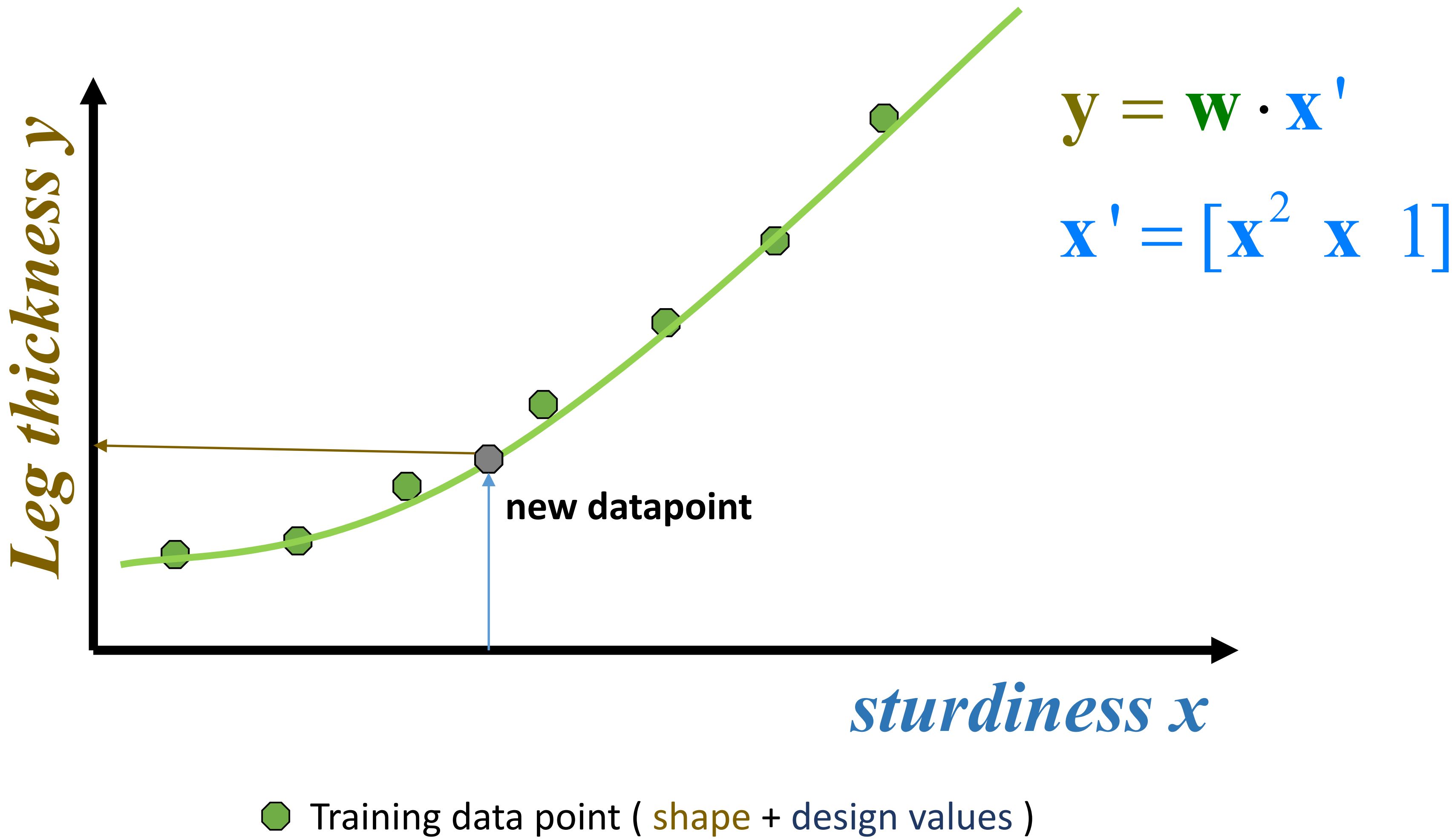
Training data point (shape + design values)

Learning basics: regression

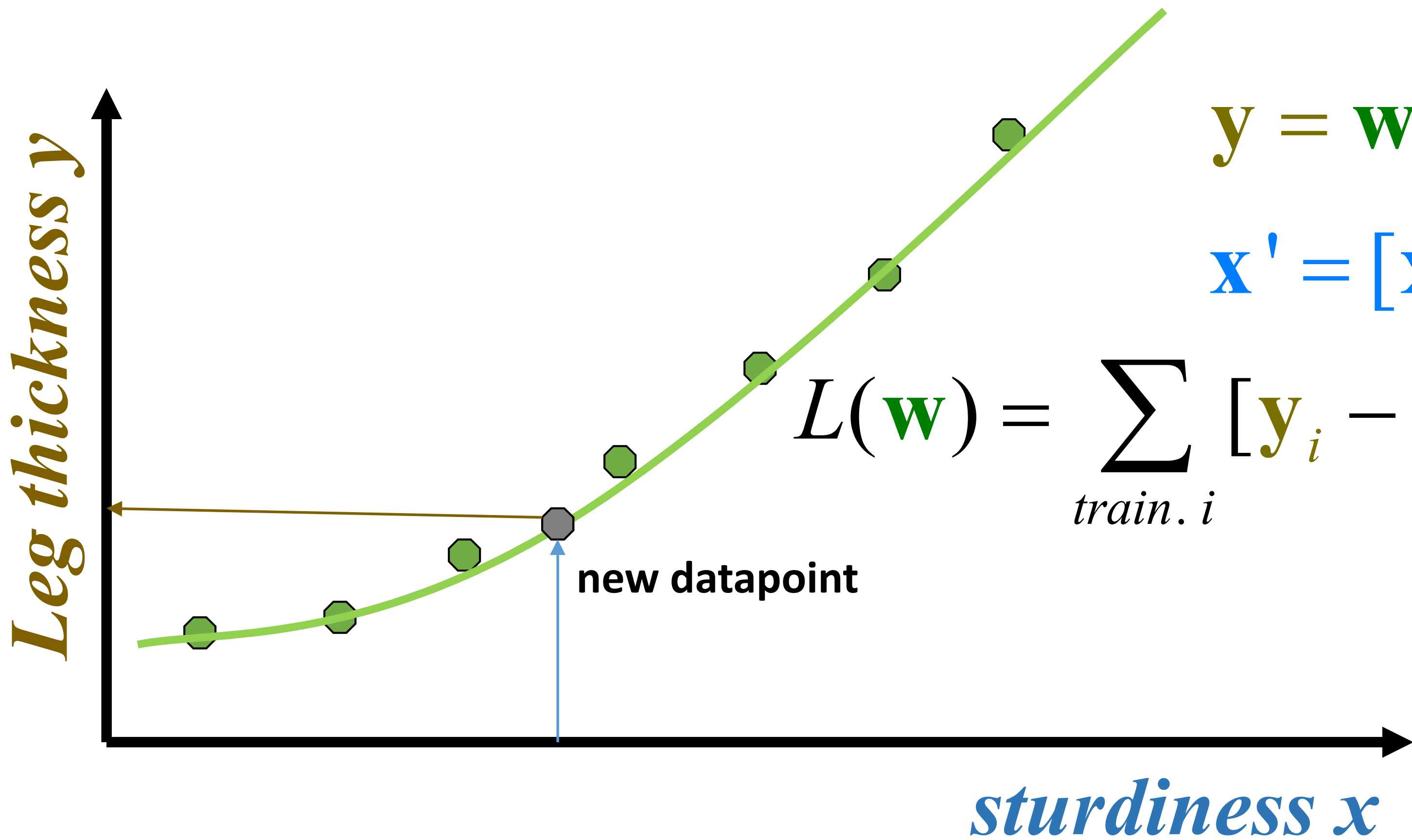


Training data point (shape + design values)

Learning basics: regression



Learning basics: regression



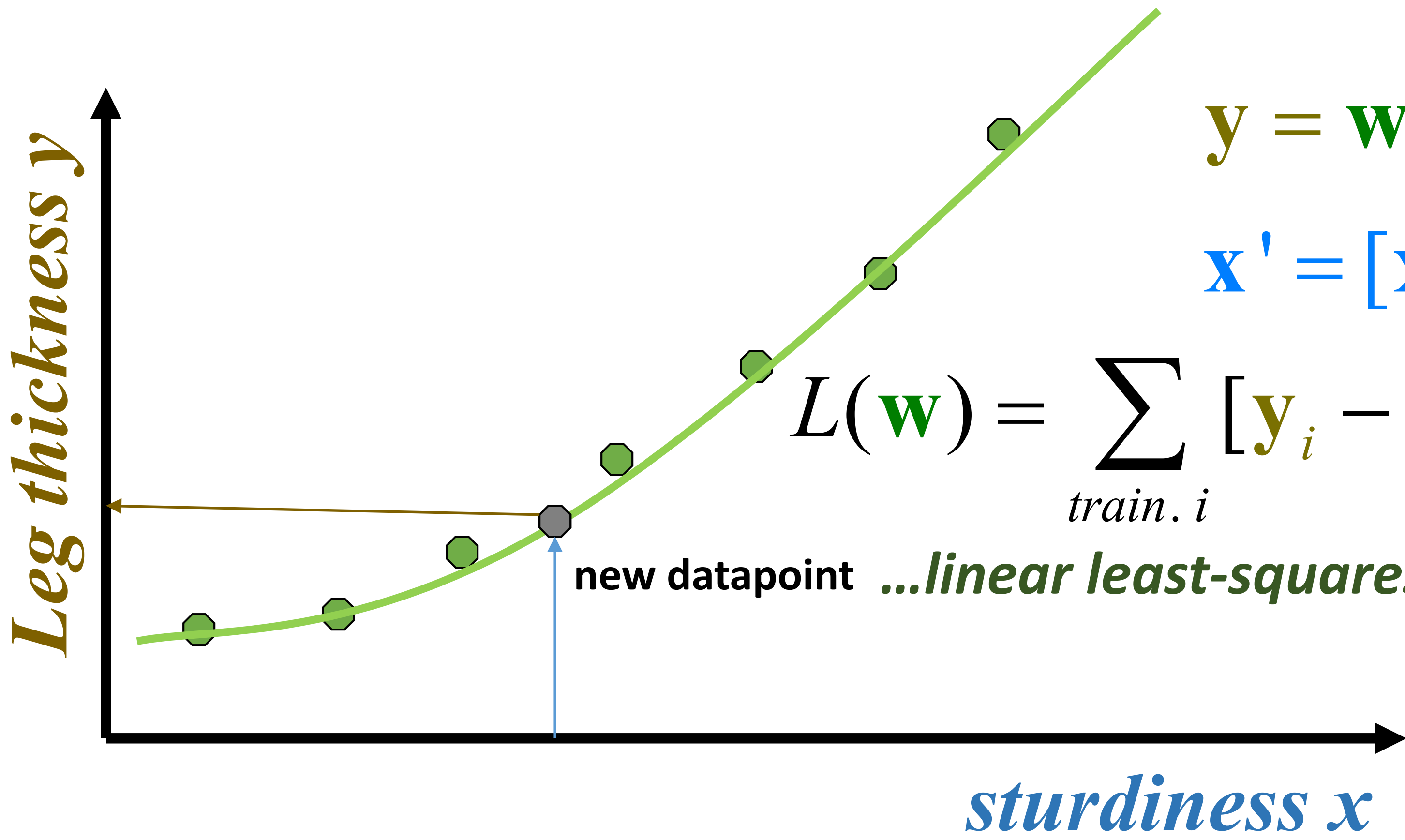
$$\mathbf{y} = \mathbf{w} \cdot \mathbf{x}'$$

$$\mathbf{x}' = [\mathbf{x}^2 \ \ \mathbf{x} \ \ 1]$$

$$L(\mathbf{w}) = \sum_{train.\,i} [\mathbf{y}_i - \mathbf{w} \cdot \mathbf{x}_i']^2$$

Training data point (shape + design values)

Learning basics: regression



$$\mathbf{y} = \mathbf{w} \cdot \mathbf{x}'$$

$$\mathbf{x}' = [\mathbf{x}^2 \;\; \mathbf{x} \;\; 1]$$

$$L(\mathbf{w}) = \sum_{train.\, i} [\mathbf{y}_i - \mathbf{w} \cdot \mathbf{x}_i']^2$$

...linear least-squares solution...

Training data point (shape + design values)

Overfitting

Important to select a function that would **avoid overfitting** & **generalize** (produce reasonable outputs for inputs not encountered during training)

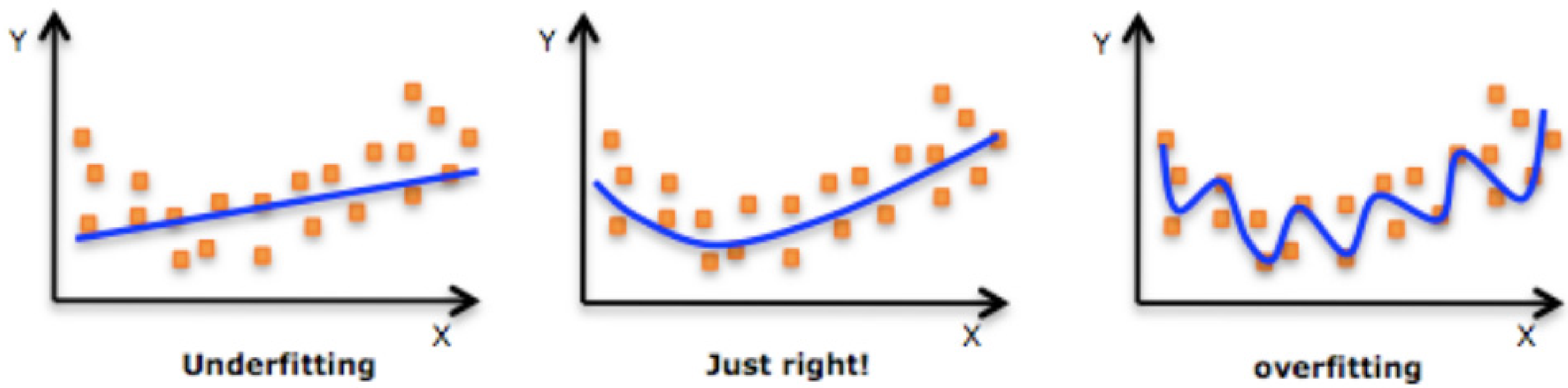


image from Andrew Ng's ML class

Learning basics: Logistic Regression

Suppose you want to predict **mug** or **no mug** for a shape.

Output: $\mathbf{y}=\mathbf{1}\ [\mathit{coffee\,mug}],\quad \mathbf{y}=0\ [\mathit{no\,coffee\,mug}]$

Input: $\mathbf{x}=\{x_1,x_2,\ldots\}$ [curvature histograms, HKS *etc*]

Learning basics: Logistic Regression

Suppose you want to predict **mug** or **no mug** for a shape.

Output: $\mathbf{y} = \mathbf{1}\ [\mathit{coffee\ mug}],\quad \mathbf{y} = 0\ [\mathit{no\ coffee\ mug}]$

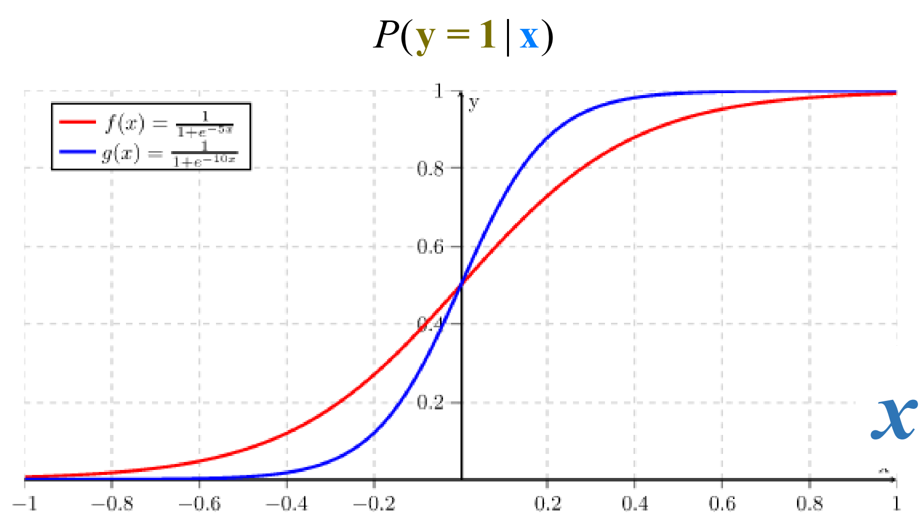
Input: $\mathbf{x} = \{x_1, x_2, \ldots\}$ [curvature histograms, HKS *etc*]

Classification function:

$$P(\mathbf{y} = \mathbf{1} \mid \mathbf{x}) = \mathbf{f}(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

where $\mathbf{w}$ is a **weight vector**

$$\sigma(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})}$$



Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…N* training shapes*)*

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{N} P(\mathbf{y}=1 \mid \mathbf{x}_i)^{[\mathbf{y_i}==1]} [1 - P(\mathbf{y}=1 \mid \mathbf{x}_i)]^{[\mathbf{y_i}==0]}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…N* training shapes*)*

Find parameters that **maximize probability of training data**

$$\max_{\mathbf{w}} \prod_{i=1}^{N} \sigma(\mathbf{w} \cdot \mathbf{x}_i)^{[\mathbf{y_i}==1]}[1-\sigma(\mathbf{w} \cdot \mathbf{x}_i)]^{[\mathbf{y_i}==0]}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x_i}$ and given labels $\mathbf{y_i}$ (mugs/no mugs) *(i=1…N* training shapes*)*

Find parameters that **maximize the log prob. of training data**

$$\max_{\mathbf{w}} \log\{\prod_{i=1}^{N} \sigma(\mathbf{w}\cdot\mathbf{x}_i)^{[\mathbf{y_i}==1]}[1-\sigma(\mathbf{w}\cdot\mathbf{x}_i)]^{[\mathbf{y_i}==0]}\}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x}_\mathbf{i}$ and given labels $\mathbf{y}_\mathbf{i}$ (mugs/no mugs) *(i=1…N* training shapes*)*

Find parameters that **maximize the log prob. of training data**

$$\max_{\mathbf{w}} \sum_{i=1}^{N} [\mathbf{y}_\mathrm{i} == 1] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + [\mathbf{y}_\mathrm{i} == 0] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))$$

Logistic regression: training

Need to estimate parameters ***w*** from training data e.g., shapes of objects $\mathbf{x}_i$ and given labels $\mathbf{y}_i$ (mugs/no mugs) *(i=1…N* training shapes*)*

This is called **log-likelihood**

$$\max_{\mathbf{w}} \underbrace{\sum_{i=1}^{N} [\mathbf{y}_i == 1] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + [\mathbf{y}_i == 0] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))}_{L(w)}$$

Logistic regression: training

Need to estimate parameters $\boldsymbol{w}$ from training data e.g., shapes of objects $\mathbf{x}_\mathbf{i}$ and given labels $\mathbf{y}_\mathbf{i}$ (mugs/no mugs) $(i=1 \ldots N$ training shapes$)$

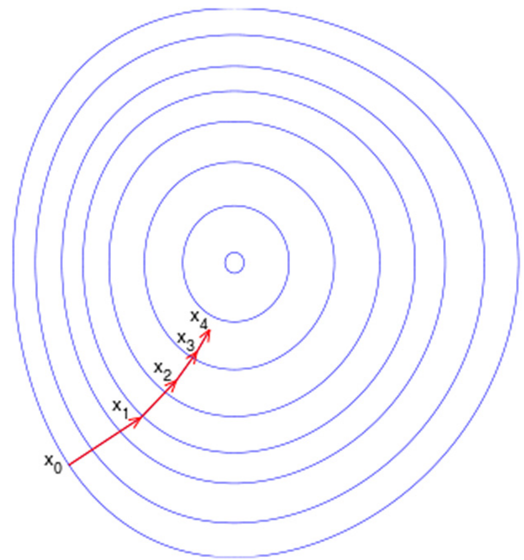
We have an **optimization problem.**

$$\max_{\mathbf{w}} \overbrace{\sum_{i=1}^{N} [\mathbf{y}_i == 1]\log\sigma(\mathbf{w}\cdot\mathbf{x}_i) + [\mathbf{y}_i == 0]\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}_i))}^{\boldsymbol{L(w)}}$$

$$\frac{\partial L(\mathbf{w})}{\partial w_d} = \sum_i x_{i,d}[\, y_i - \sigma(\mathbf{w}\cdot\mathbf{x}_i)\,]$$

(partial derivative for d^{th} parameter)

How can we minimize/maximize a function?



Gradient descent: Given a random initialization of parameters and a step rate η, update them according to:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \eta \nabla L(\mathbf{w})$$

See also **quasi-Newton** and **IRLS** methods

Regularization

Overfitting:
few training data vs large number of parameters!

Penalize large weights:

$$\min_{\mathbf{w}} \; -L(\mathbf{w}) + \lambda \sum_d \mathrm{w}_d{}^2$$

Called **ridge regression (or L2 regularization)**

Regularization

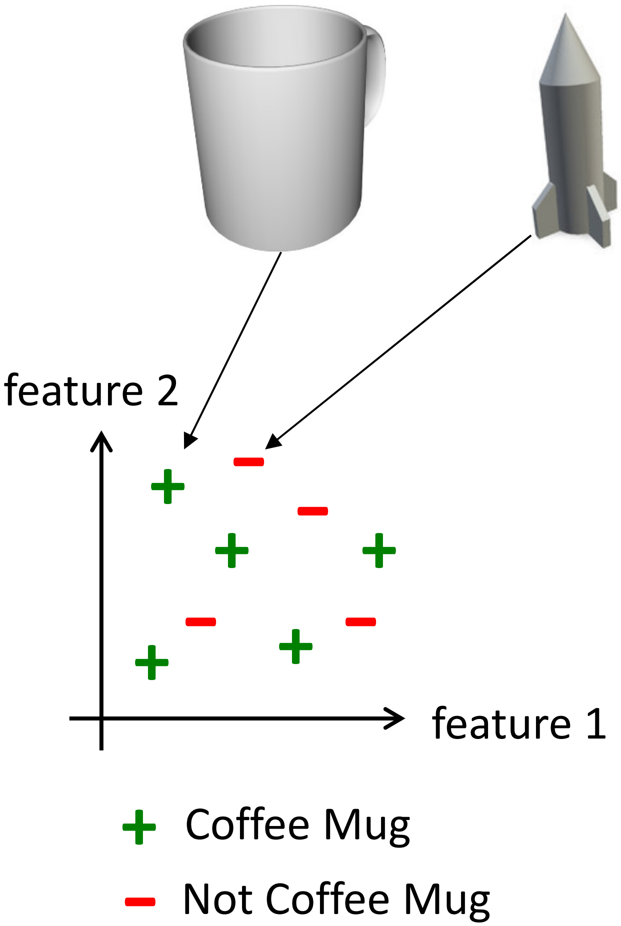
Overfitting:
few training data vs large number of parameters!

Penalize non-zero weights - push as many as possible to **0**:

$$\min_{\mathbf{w}} \quad -L(\mathbf{w}) + \lambda \sum_{d} |\, \mathrm{w}_d \,|$$

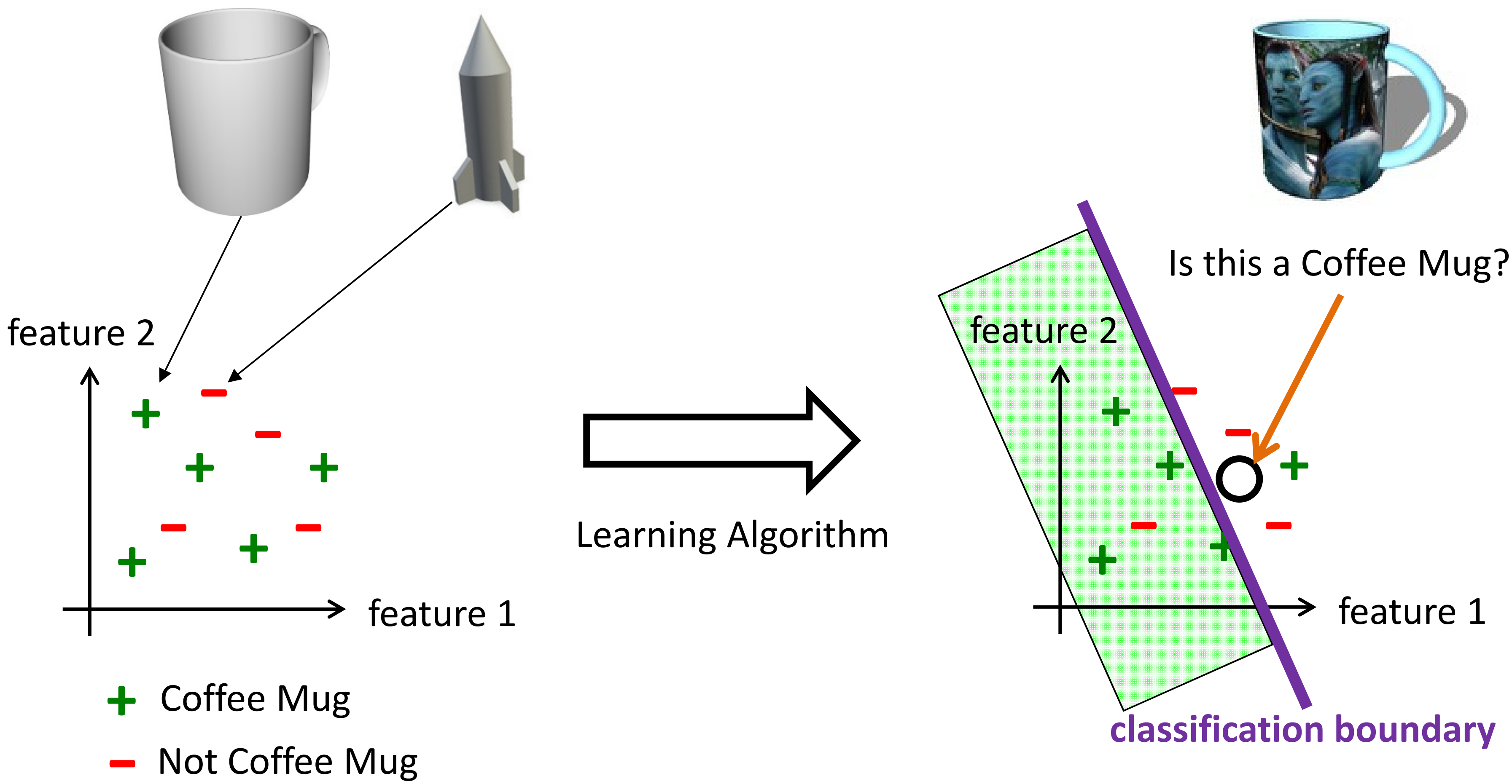
Called **Lasso (or L1 regularization)**

The importance of choosing good features…



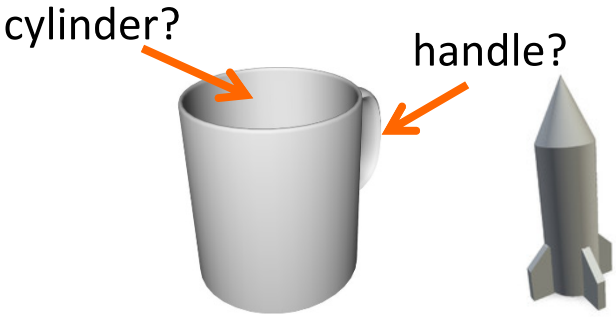
modified slides originally by Adam Coates

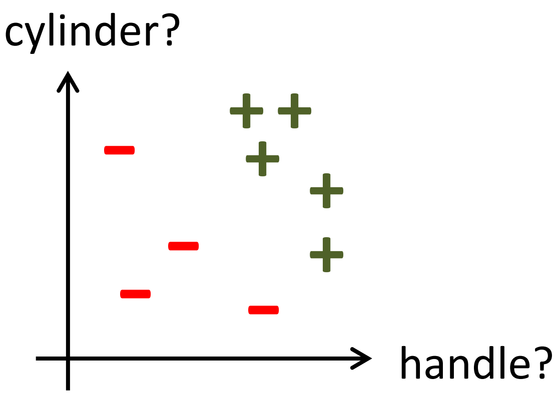
The importance of choosing good features...



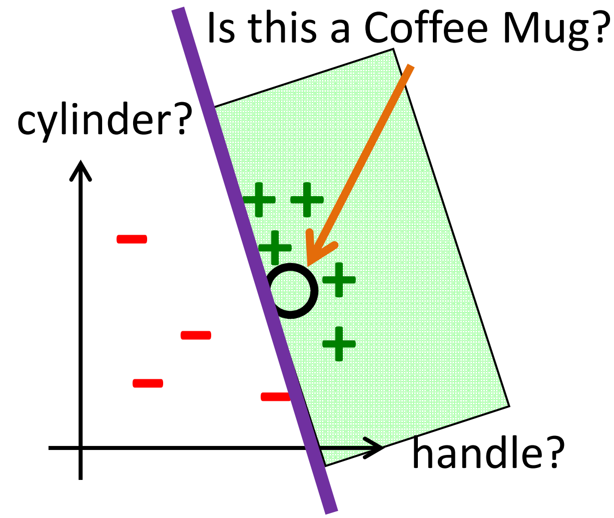
modified slides originally by Adam Coates

The importance of choosing good features...





Learning Algorithm



+ Coffee Mug

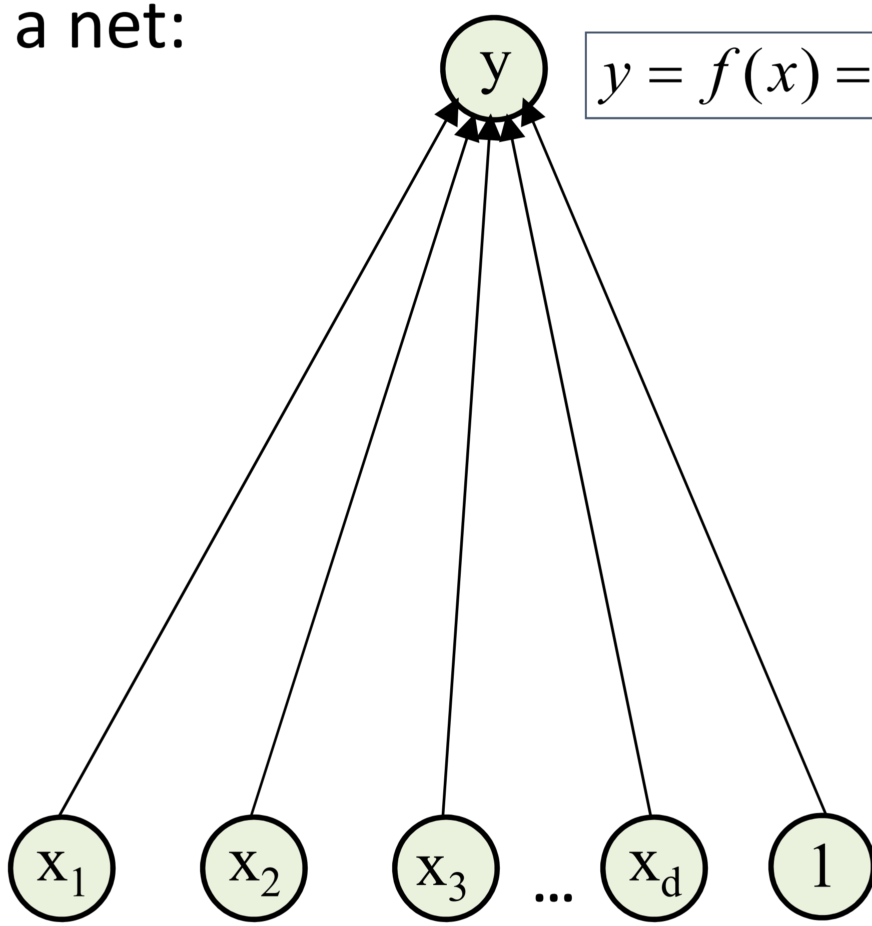
− Not Coffee Mug

modified slides originally by Adam Coates

From “shallow” to “deep” mappings

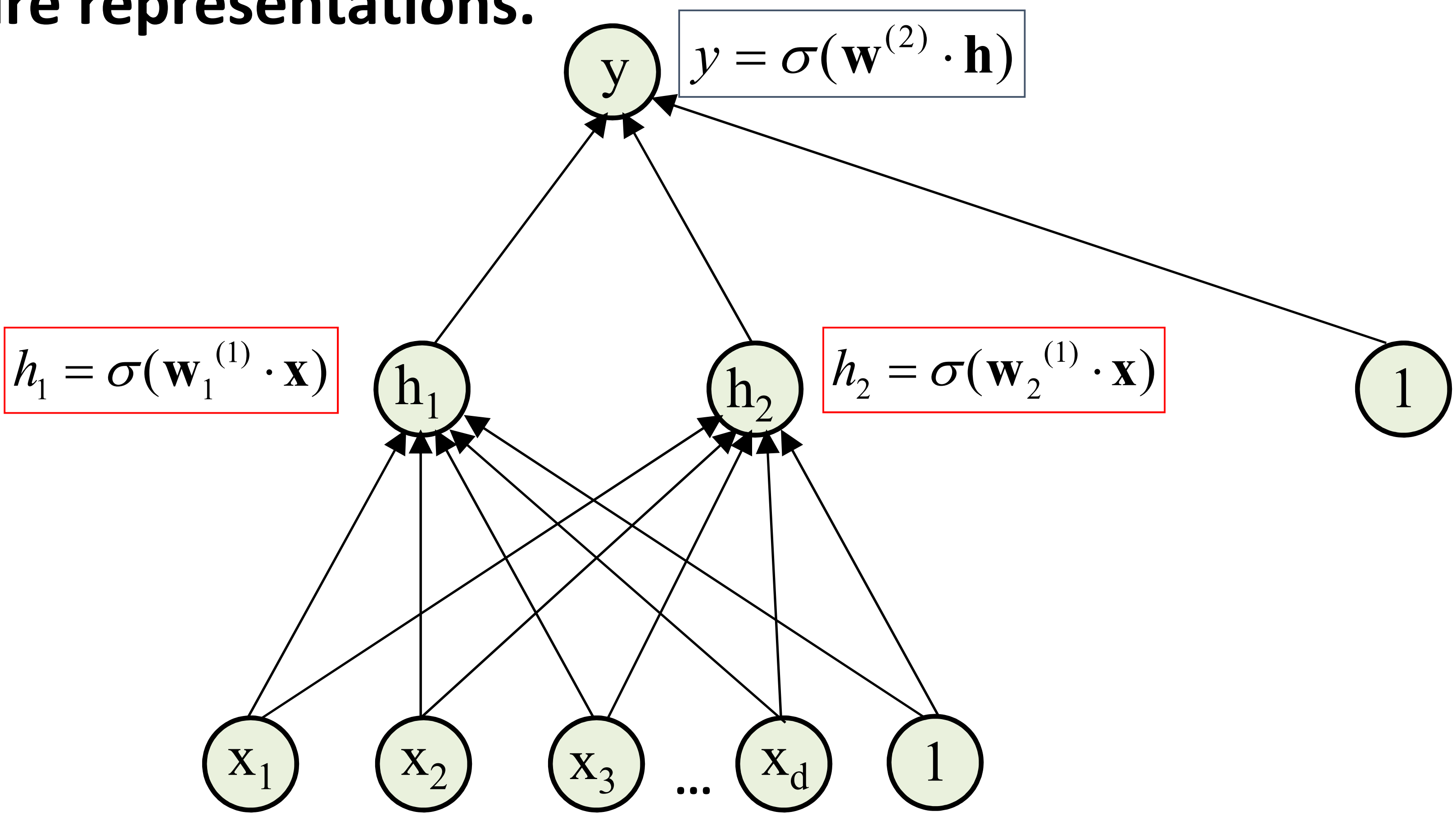
Logistic regression: output is a **direct function of inputs**.
Think of it as a net:

$$y = f(x) = \sigma(\mathbf{w} \cdot \mathbf{x})$$



Neural network

Introduce latent nodes that play the role of **learned feature representations.**



Neural network

Same as logistic regression but now our output function has **multiple stages** ("layers", "modules").

$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h} \longrightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \longrightarrow y$$

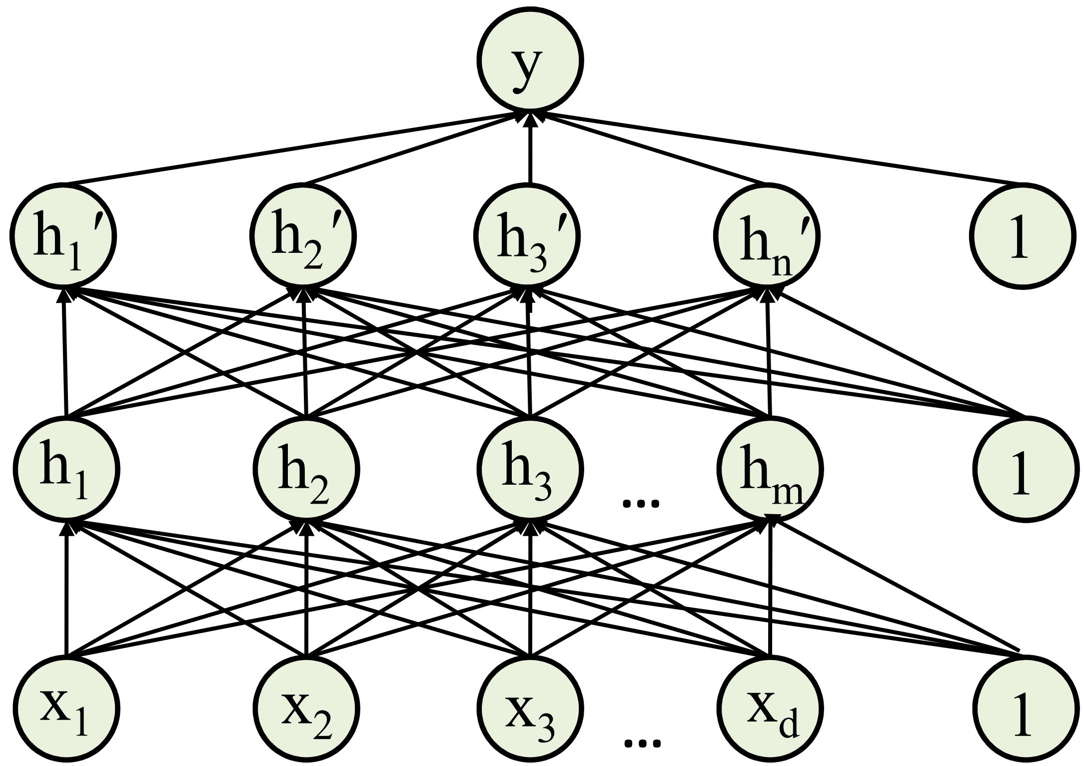
Intermediate representation **Prediction**

$$where \quad \mathbf{W}^{(\cdot)} = \begin{bmatrix} \mathbf{w_1}^{(\cdot)} \\ \mathbf{w_2}^{(\cdot)} \\ \cdots \\ \mathbf{w}_m^{(\cdot)} \end{bmatrix}$$

modified slides originally by Adam Coates

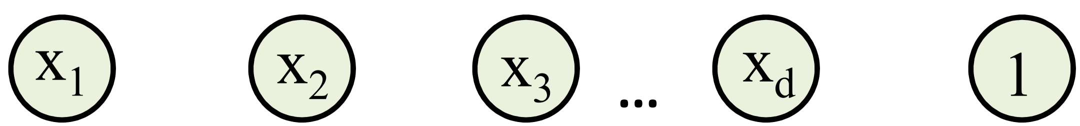
Neural network

Stack up several layers:



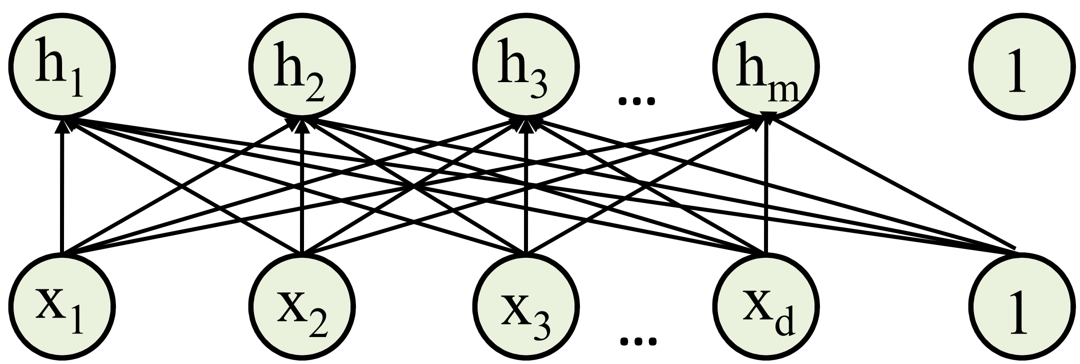
Forward propagation

Process to compute output:



Forward propagation

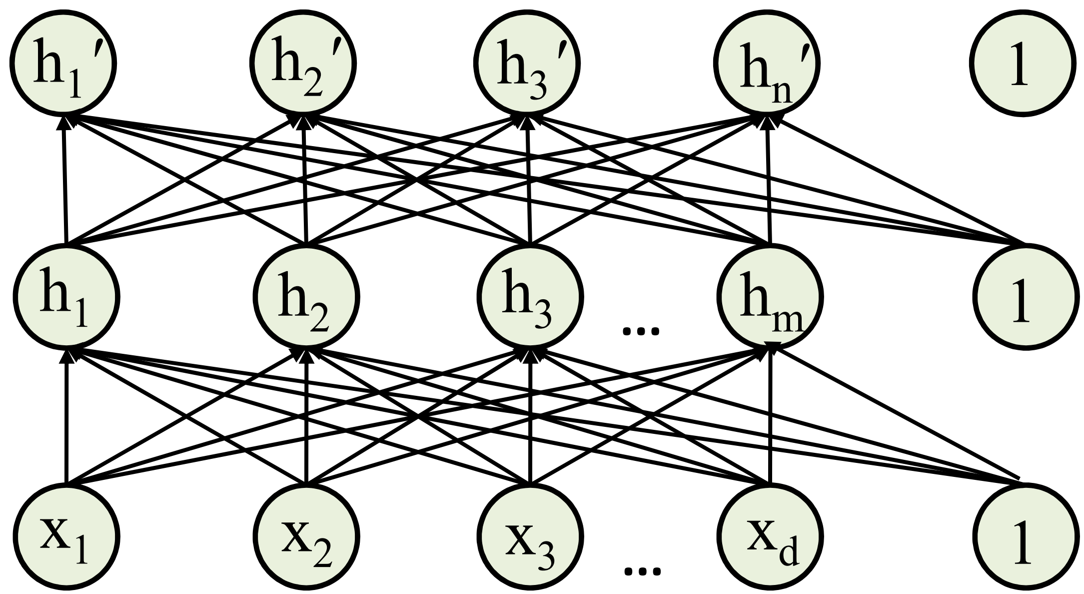
Process to compute output:



$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h}$$

Forward propagation

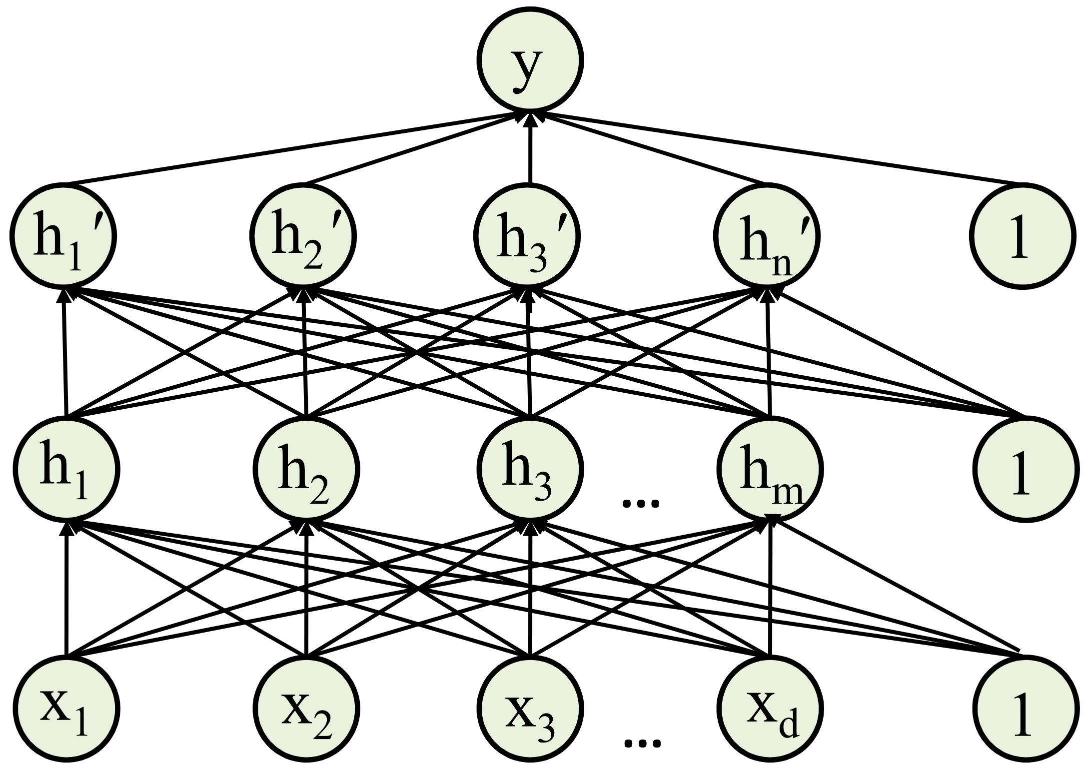
Process to compute output:



$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h} \longrightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \longrightarrow \mathbf{h}'$$

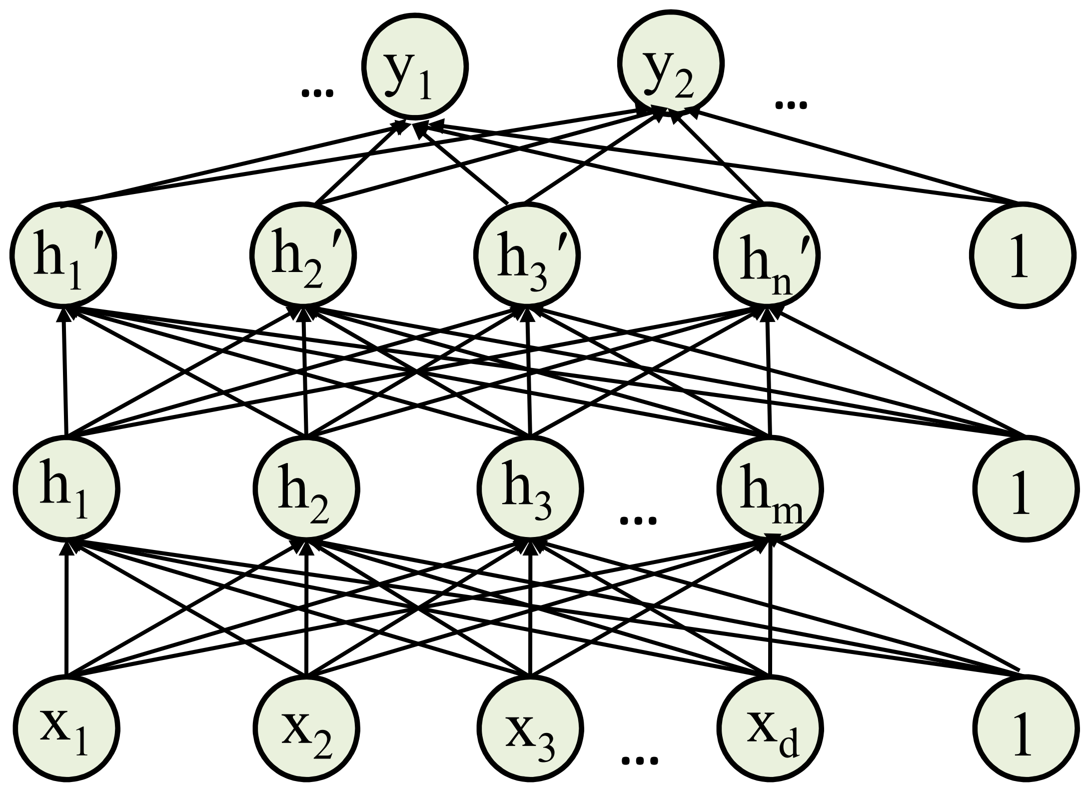
Forward propagation

Process to compute output:



$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h} \longrightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \longrightarrow \mathbf{h}' \longrightarrow \boxed{\sigma(\mathbf{W}^{(3)} \cdot \mathbf{h}')} \longrightarrow y$$

Multiple outputs



$$\mathbf{x} \longrightarrow \boxed{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \longrightarrow \mathbf{h} \longrightarrow \boxed{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \longrightarrow \mathbf{h}' \longrightarrow \boxed{\sigma(\mathbf{W}^{(3)} \cdot \mathbf{h}')} \longrightarrow \mathbf{y}$$

How can we learn the parameters?

Use a loss function e.g., for classification:

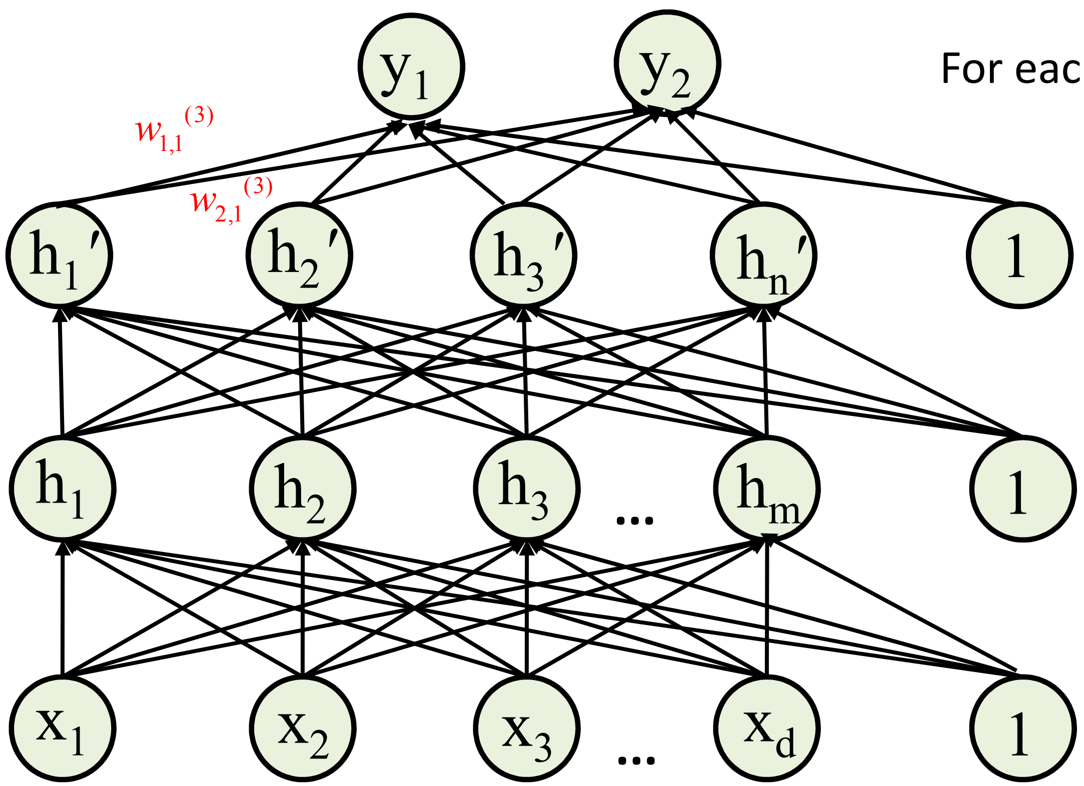
$$L(\mathbf{w}) = -\sum_{i} \sum_{output\ t} [\mathbf{y}_{\mathbf{i,t}} == 1] \log f_t(\mathbf{x}_i) + [\mathbf{y}_{\mathbf{i,t}} == 0] \log(1 - f_t(\mathbf{x}_i))$$

In case of regression i.e., for predicting continuous outputs:

$$L(\mathbf{w}) = \sum_{i} \sum_{output\ t} [\mathbf{y}_{i,t} - f_t(\mathbf{x}_i)]^2$$

Backpropagation

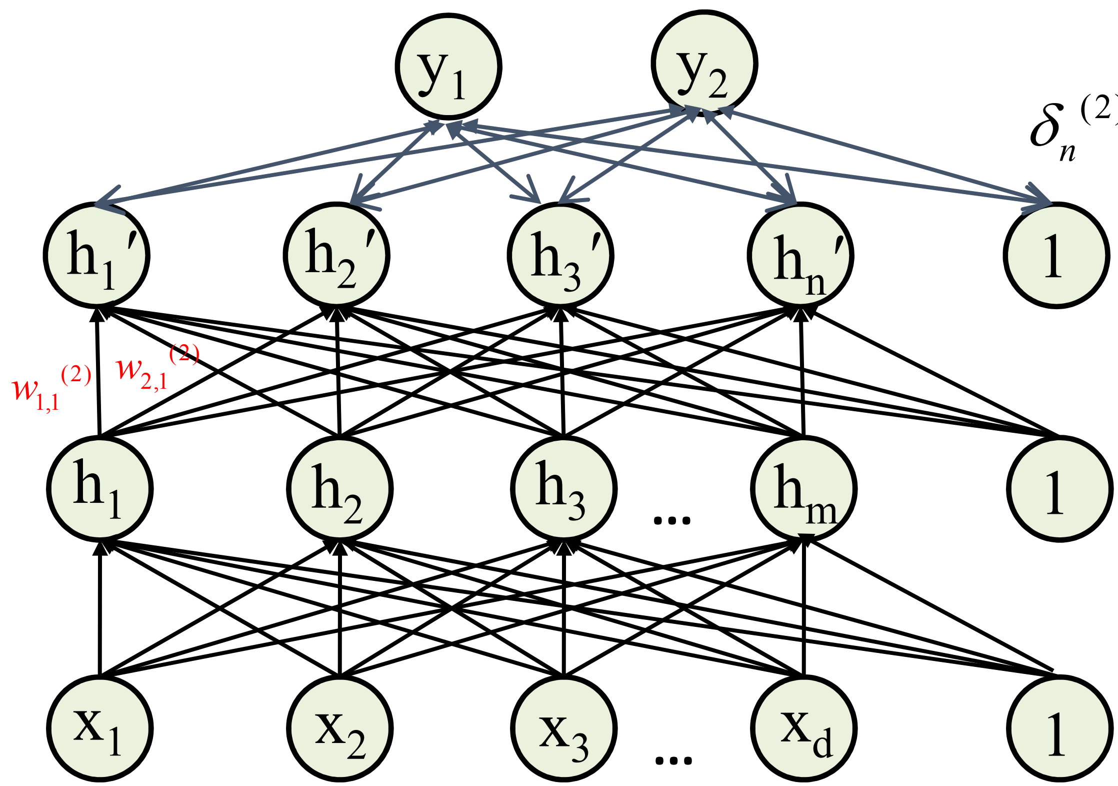
For each training example i:



For each output: $\delta_t^{(3)} = f_t(\mathbf{x}) - y_t$

$$\frac{\partial L(\mathbf{w})}{\partial w_{t,n}^{(3)}} = \delta_t^{(3)} h_n$$

Backpropagation

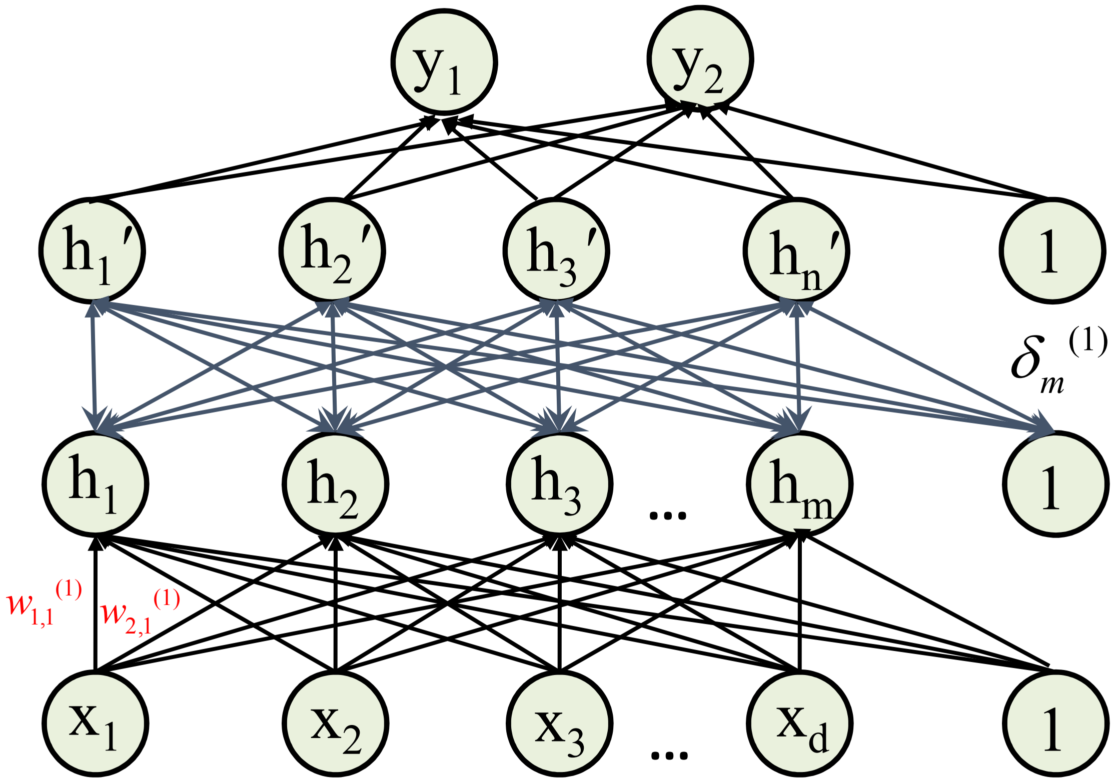


$$\delta_n^{(2)} = \sigma'(\mathbf{w_n}^{(2)} \cdot \mathbf{h}) \sum_t w_{t,n}^{(3)} \delta_t^{(3)}$$

Note: $\sigma'(\cdot) = \sigma(\cdot)[1 - \sigma(\cdot)]$

$$\frac{\partial L(\mathbf{w})}{\partial w_{n,m}^{(2)}} = \delta_n^{(2)} h_m$$

Backpropagation



$$\delta_m^{(1)} = \sigma'(\mathbf{w}_m^{(1)} \cdot \mathbf{x}) \sum_n w_{n,m}^{(2)} \delta_n^{(2)}$$

$$\frac{\partial L(\mathbf{w})}{\partial w_{m,d}^{(1)}} = \delta_m^{(1)} x_d$$

Is this magic?

All these are derivatives derived analytically using the **chain rule**!

Gradient descent is expressed through **backpropagation of messages** δ following the structure of the model

Training algorithm

For each training example [in a batch]

1. **Forward propagation** to compute outputs per layer
2. **Back propagate** messages δ from top to bottom layer
3. Multiply messages δ with inputs to compute **derivatives per layer**
4. **Accumulate the derivatives** from that training example

Apply the gradient descent rule

Yet, this does not work so easily...

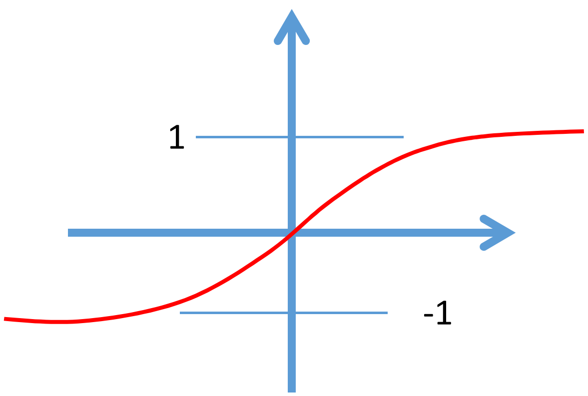
Yet, this does not work so easily...

- **Non-convex:** Local minima; convergence criteria.
- Optimization becomes difficult with **many layers**.
- Hard to diagnose and **debug malfunctions**.
- **Many things turn out to matter**:
  - Choice of nonlinearities.
  - Initialization of parameters.
  - Optimizer parameters: step size, schedule.

Non-linearities

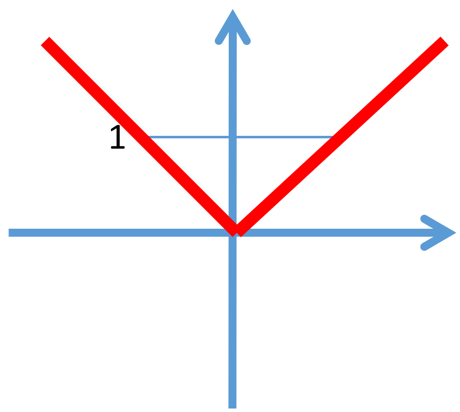
- **Choice of functions inside network matters.**
  - Sigmoid function yields highly non-convex loss functions
  - Some other choices often used:

tanh(·)



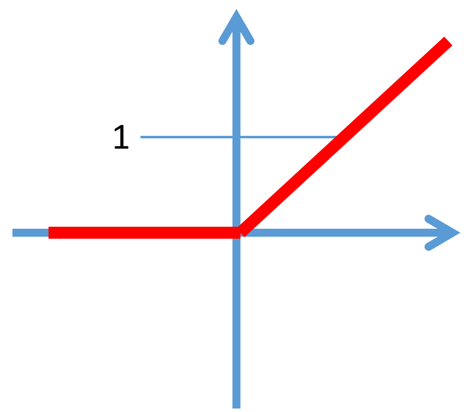
$tanh'(\cdot) = 1 - tanh(\cdot)^2$

abs(·)



$abs'(\cdot) = sign(\cdot)$

ReLu(·) = max{0, ·}

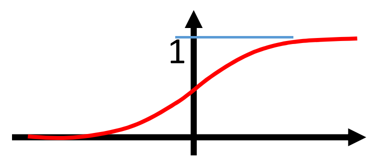


$ReLu'(\cdot) = [\, \cdot > 0 \,]$

"Rectified Linear Unit"
→ Most popular.
[Nair & Hinton, 2010]

Initialization

- **Usually small random values.**
  - Try to choose so that typical input to a neuron avoids saturating

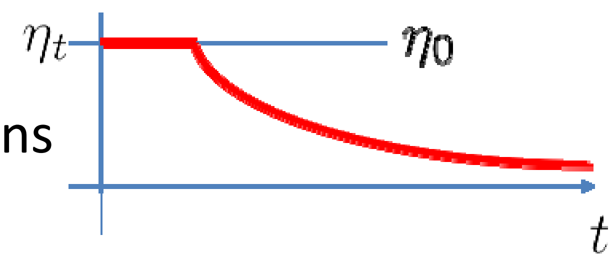


- **Initialization schemes for weights used as input to a node:**
  - tanh units: Uniform[-r, r]; sigmoid: Uniform[-4r, 4r].
  - See [Glorot et al., AISTATS 2010]

$$r = \sqrt{6/(\text{fan-in} \ + \ \text{fan-out})}$$

Step size

- **Fixed step-size**
  - try many, choose the best...
  - pick size with least test error on a validation set after T iterations
- **Dynamic step size**
  - decrease after T iterations



  - if simply the objective is not decreasing much, cut step by half

Momentum

Modify stochastic/batch gradient descent:

$$\text{Before}: \quad \Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} L(\mathbf{w}), \quad w = w - \Delta \mathbf{w}$$

$$\text{With momentum}: \Delta \mathbf{w} = \mu \Delta \mathbf{w}_{previous} + \eta \nabla_{\mathbf{w}} L(\mathbf{w}), \quad w = w - \Delta \mathbf{w}$$

"Smooth" estimate of gradient from several steps of gradient descent:

- High-curvature directions cancel out, low-curvature directions "add up" and accelerate.
- **Other techniques: Adagrad, Adadelta, batch normalization…**

Momentum+L2 regularization

Modify stochastic/batch gradient descent:

$$\text{Before}: \quad \Delta\mathbf{w} = \eta\nabla_{\mathbf{w}}L(\mathbf{w}), \quad w = w - \Delta\mathbf{w}$$

$$\text{With momentum}: \Delta\mathbf{w} = \mu\Delta\mathbf{w}_{previous} + \eta\nabla_{\mathbf{w}}L(\mathbf{w}), \quad w = w - \Delta\mathbf{w}$$

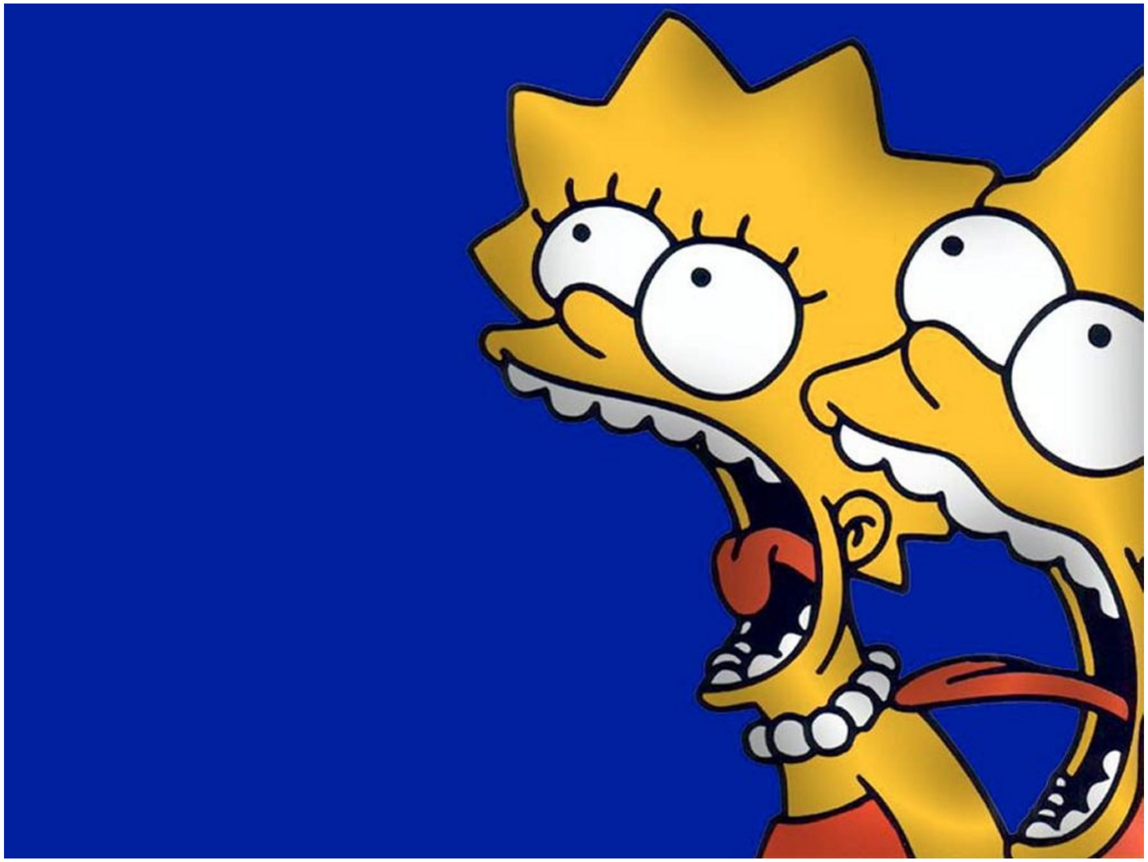
"Smooth" estimate of gradient from several steps of gradient descent:

- High-curvature directions cancel out, low-curvature directions "add up" and accelerate.
- **Other techniques: Adagrad, Adadelta, batch normalization…**

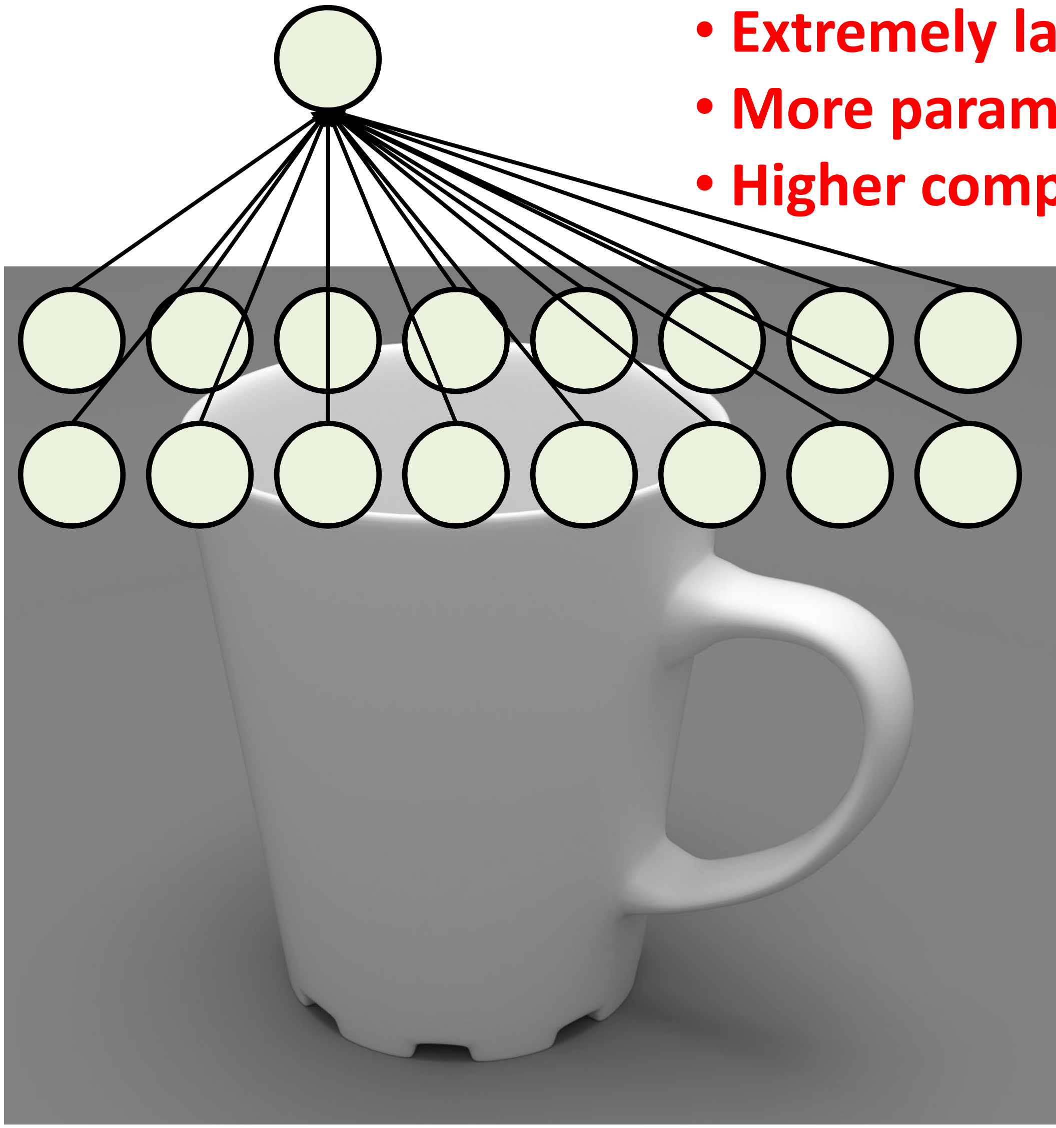
Add **L2 regularization** to the loss function:

$$\Delta\mathbf{w} = \eta\nabla_{\mathbf{w}}(L(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2)$$

Yet, things will not still work well!

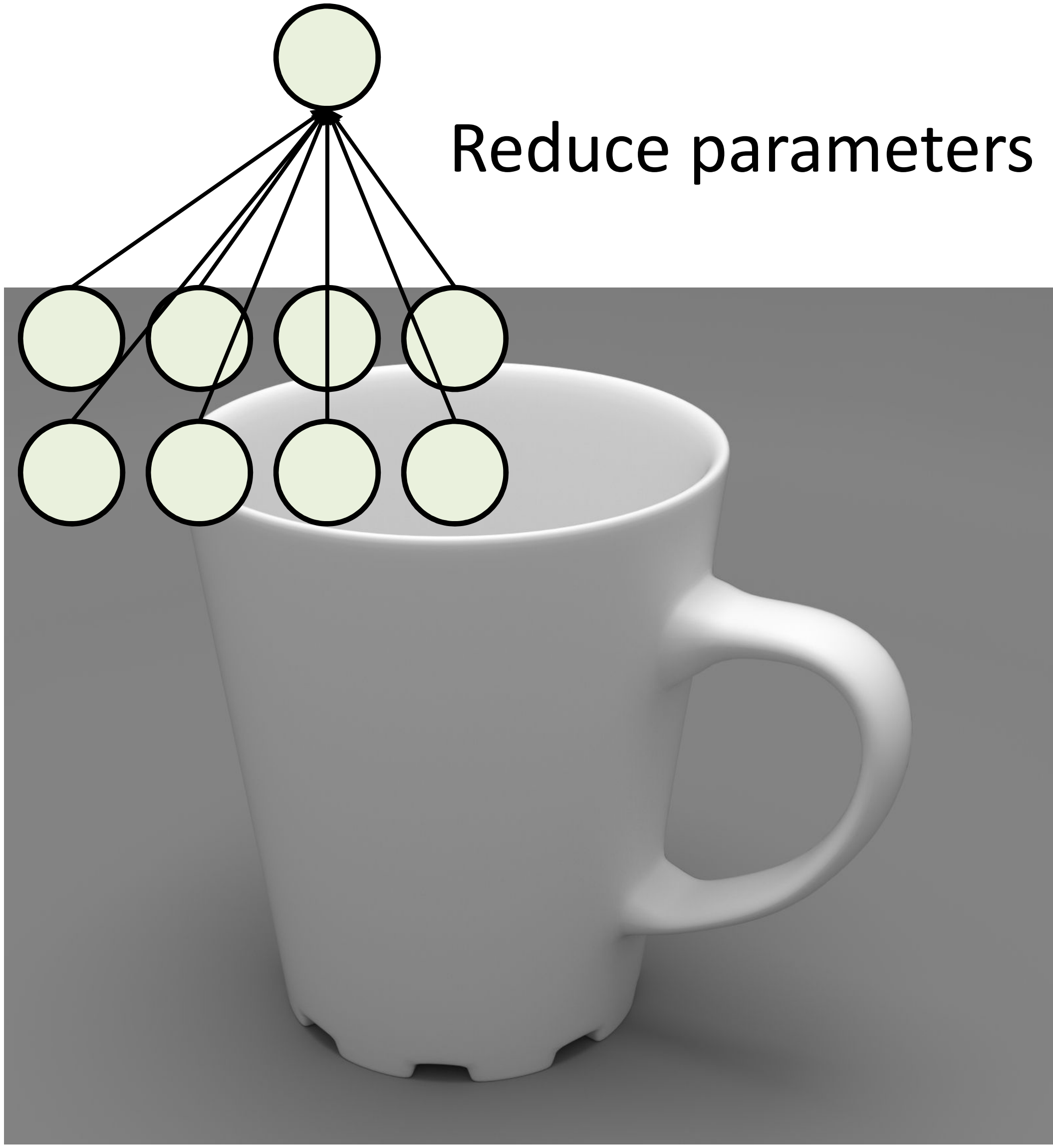


Main problem



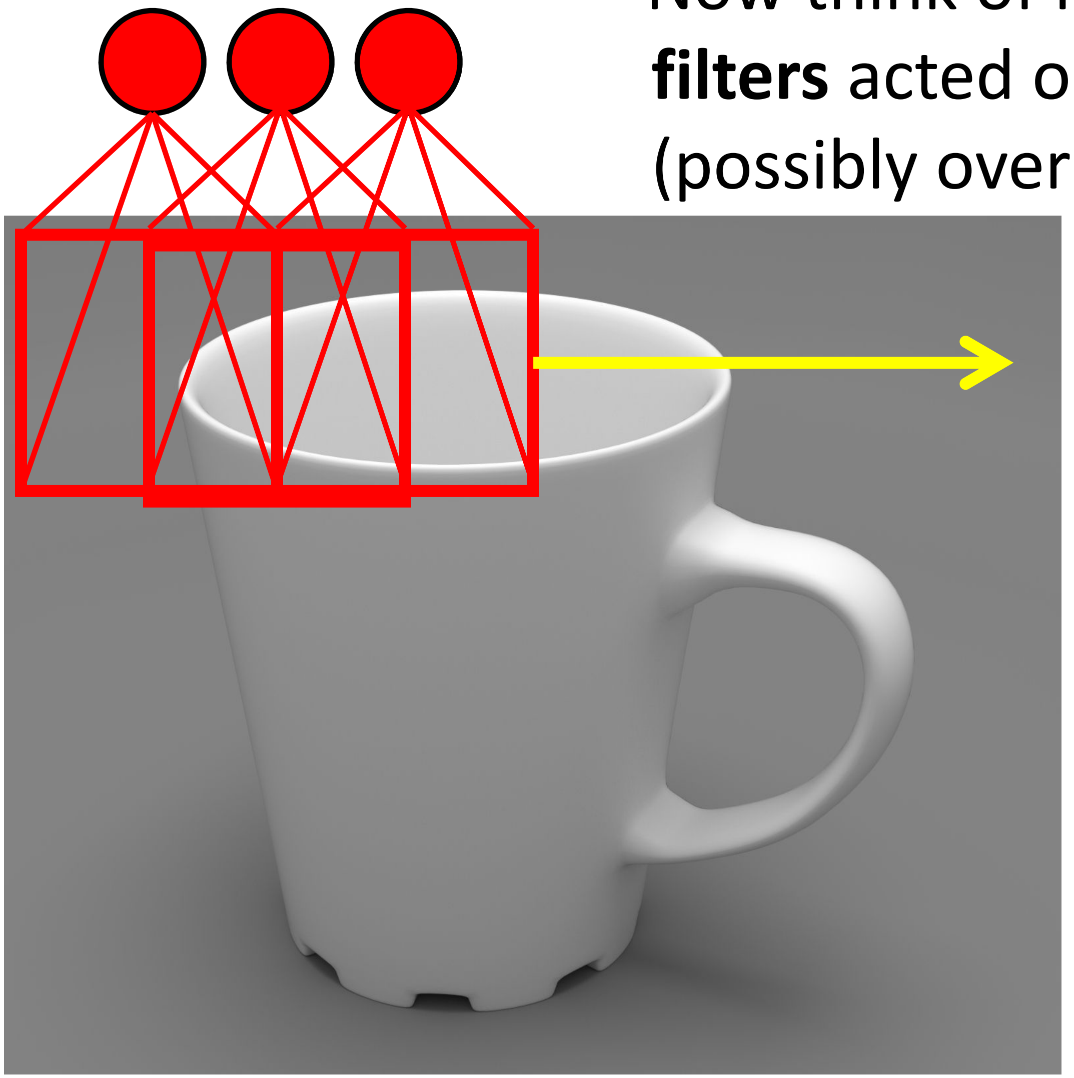
- **Extremely large number of connections.**
- **More parameters to train.**
- **Higher computational expense.**

Local connectivity



Reduce parameters with **local connections!**

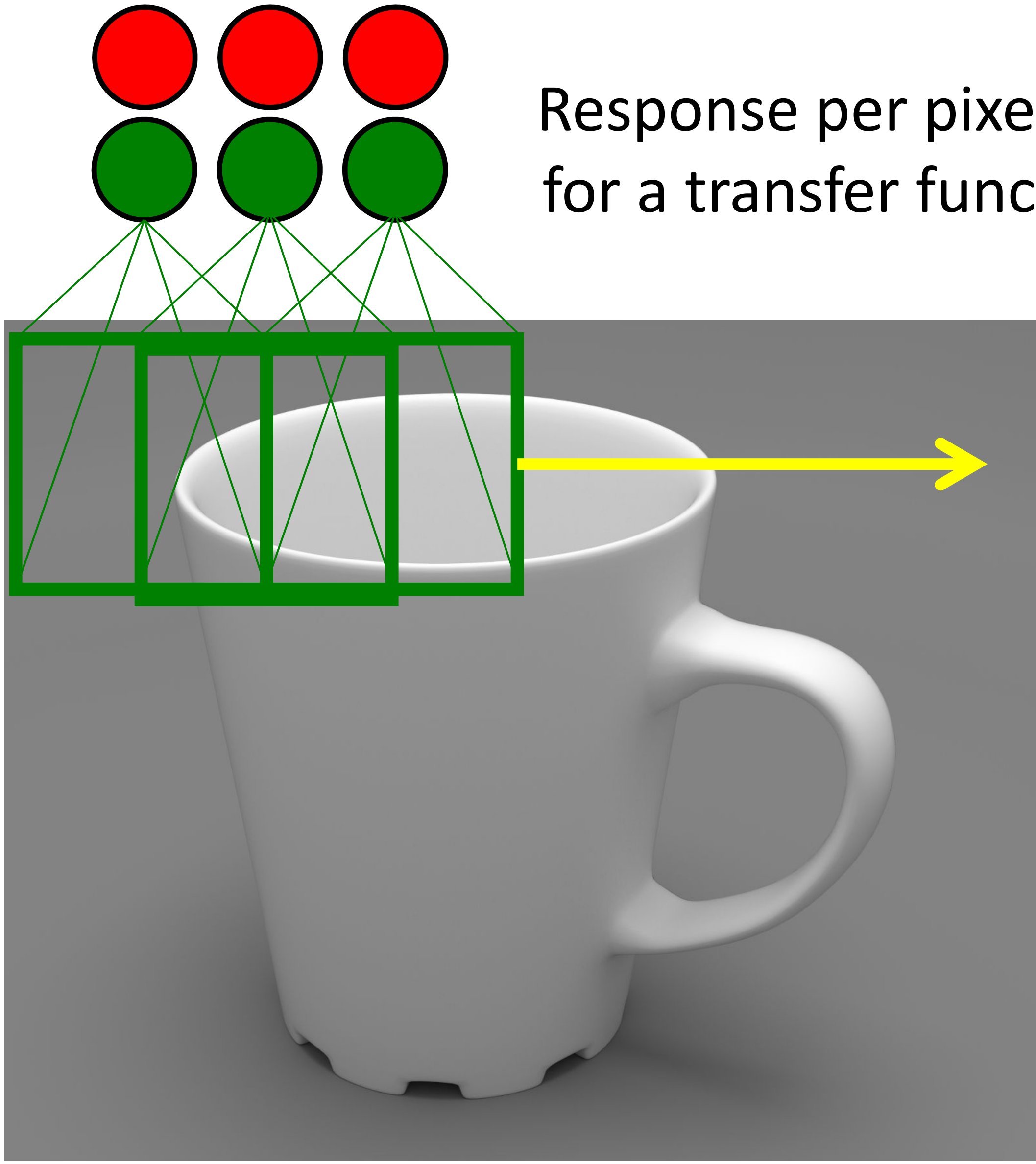
Neurons as convolution filters



Now think of neurons as convolutional **filters** acted on small adjacent (possibly overlapping) windows

Window size is called **"receptive field" size** and spacing is called **"step" or "stride"**

Can have many filters!

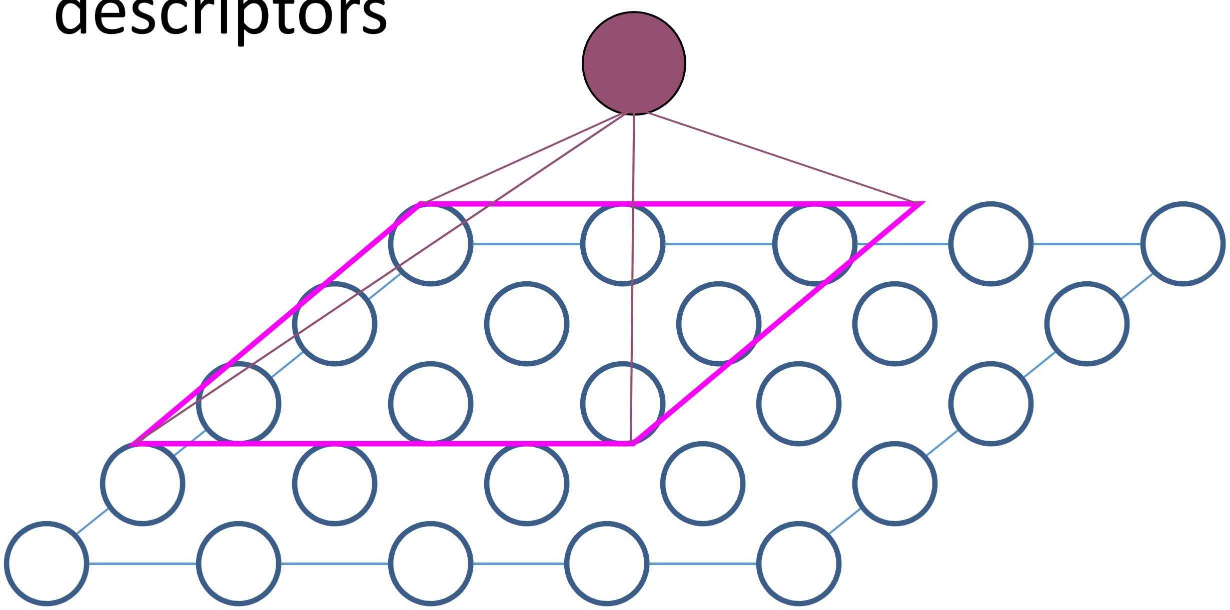


Response per pixel p, per filter f
for a transfer function g: $\boldsymbol{h_{p,f}} = \mathbf{g}(\mathbf{w_f} \cdot \mathbf{x_p})$

modified slides originally
by Adam Coates

Pooling

Apart from hidden layers dedicated to convolution, we can have layers dedicated to extract **locally invariant** descriptors



Max pooling:

$$h_{p',f} = \max_{p}(\mathbf{x}_{\mathbf{p}})$$

Mean pooling:

$$h_{p',f} = \underset{p}{avg}(\mathbf{x}_{\mathbf{p}})$$

Fixed filter (e.g., Gaussian):

$$h_{p',f} = w_{gaussian} \cdot \mathbf{x}_{\mathbf{p}}$$

Progressively reduce the resolution of the image, so that the next convolutional filters are applied on larger scales

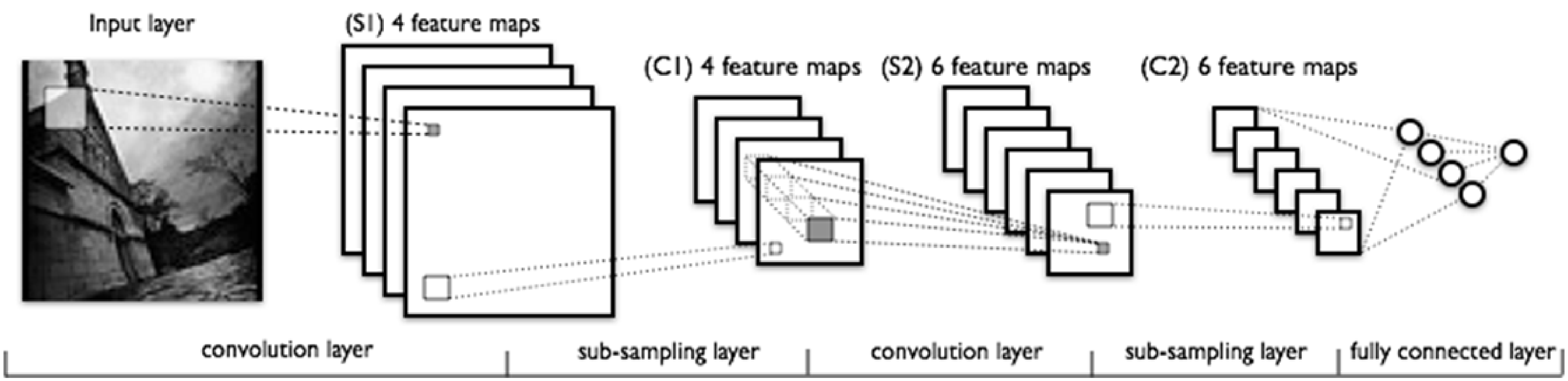
[Scherer et al., ICANN 2010]

[Boureau et al., ICML 2010]

A mini convolutional neural network

Interchange convolutional and pooling (subsampling) layers.

In the end, **unwrap all feature maps into a single feature vector** and pass it through the classical (**fully connected**) neural network.

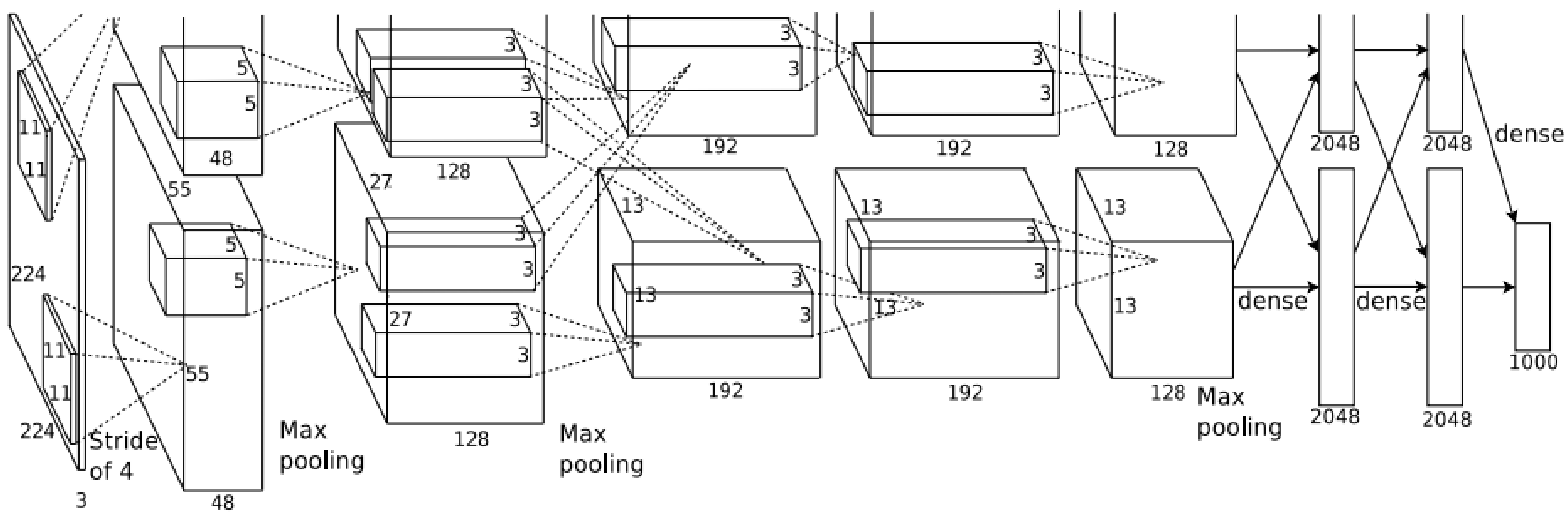


Source: http://deeplearning.net/tutorial/lenet.html

AlexNet

Proposed architecture from Krizhevsky et al., NIPS 2012:

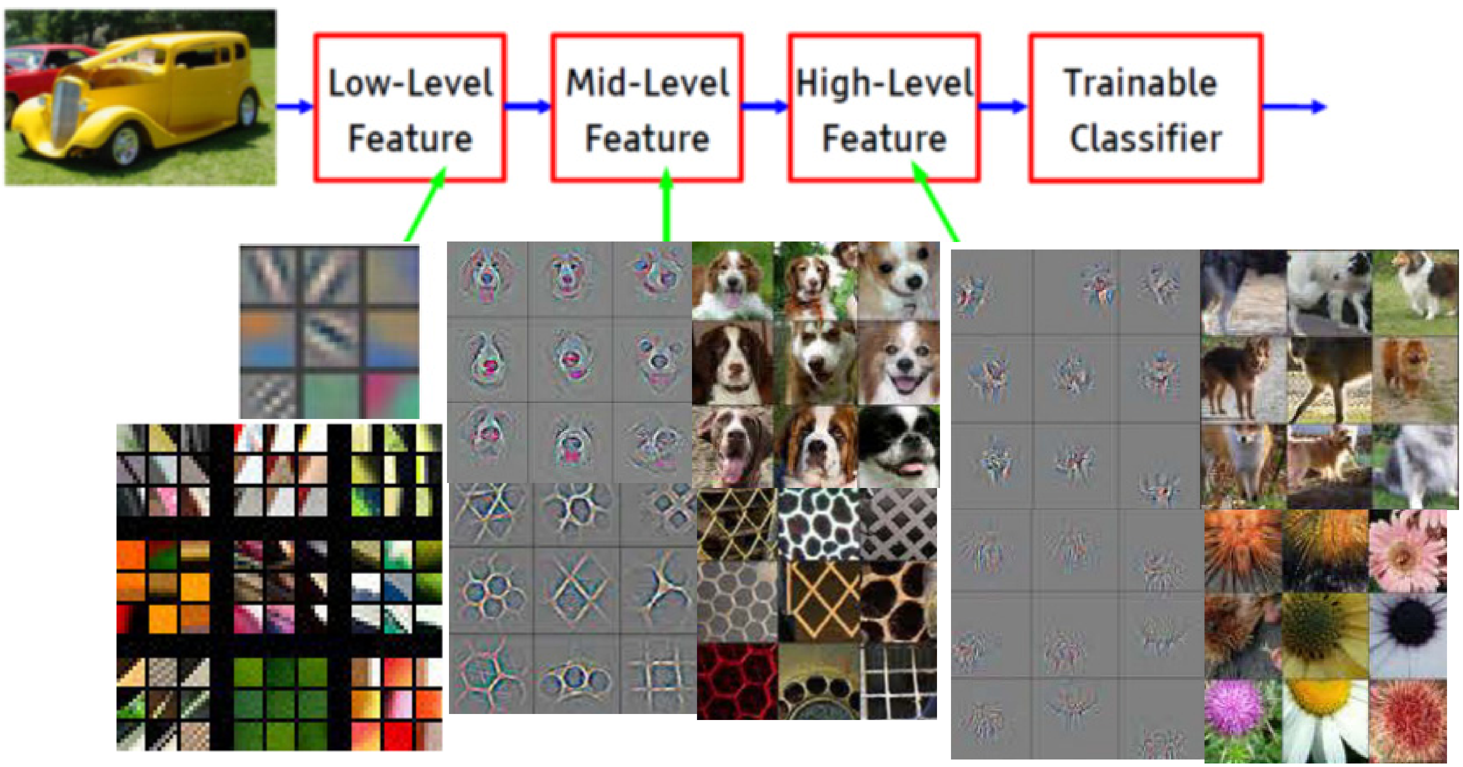
- Convolutional layers with Rectified linear units
- Max-pooling layers
- Stochastic gradient descent on GPU with momentum, L2 regularization, dropout
- Applied to image classification (ImageNet competition – top runner & game changer)



Krizhevsky et al., NIPS 2012

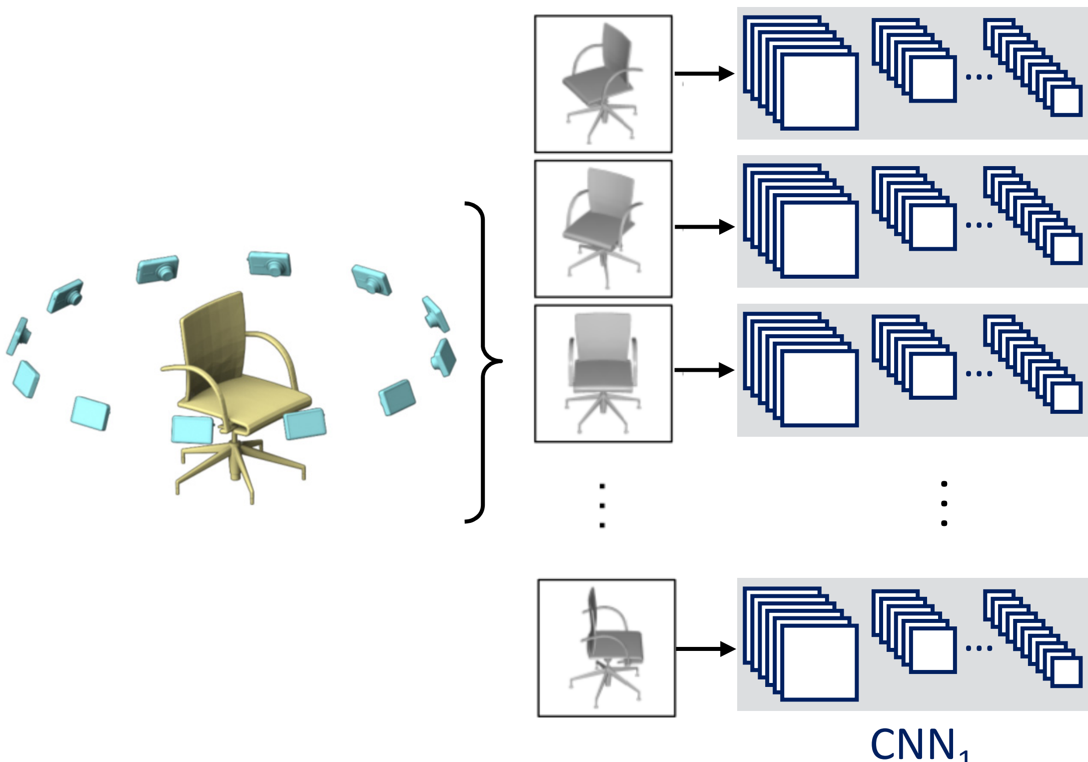
Learned representations

Think of convolution filters as optimized **feature templates capturing various hierarchical patterns** (edges, local structures, sub-parts, parts…)



see Matthew D. Zeiler and Rob Fergus, Visualizing and Understanding Convolutional Networks, 2014

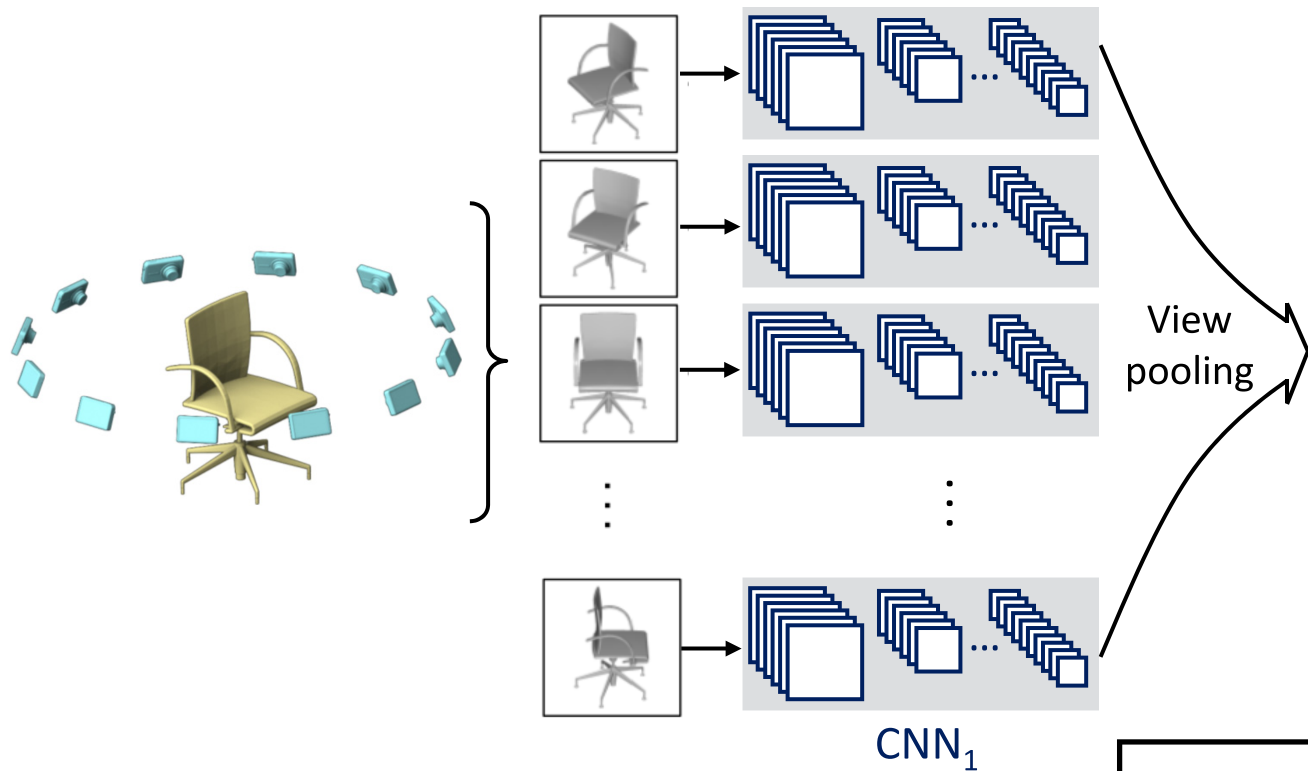
Multi-view CNNs for shape analysis



CNN$_1$: a ConvNet extracting image features per view

Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015

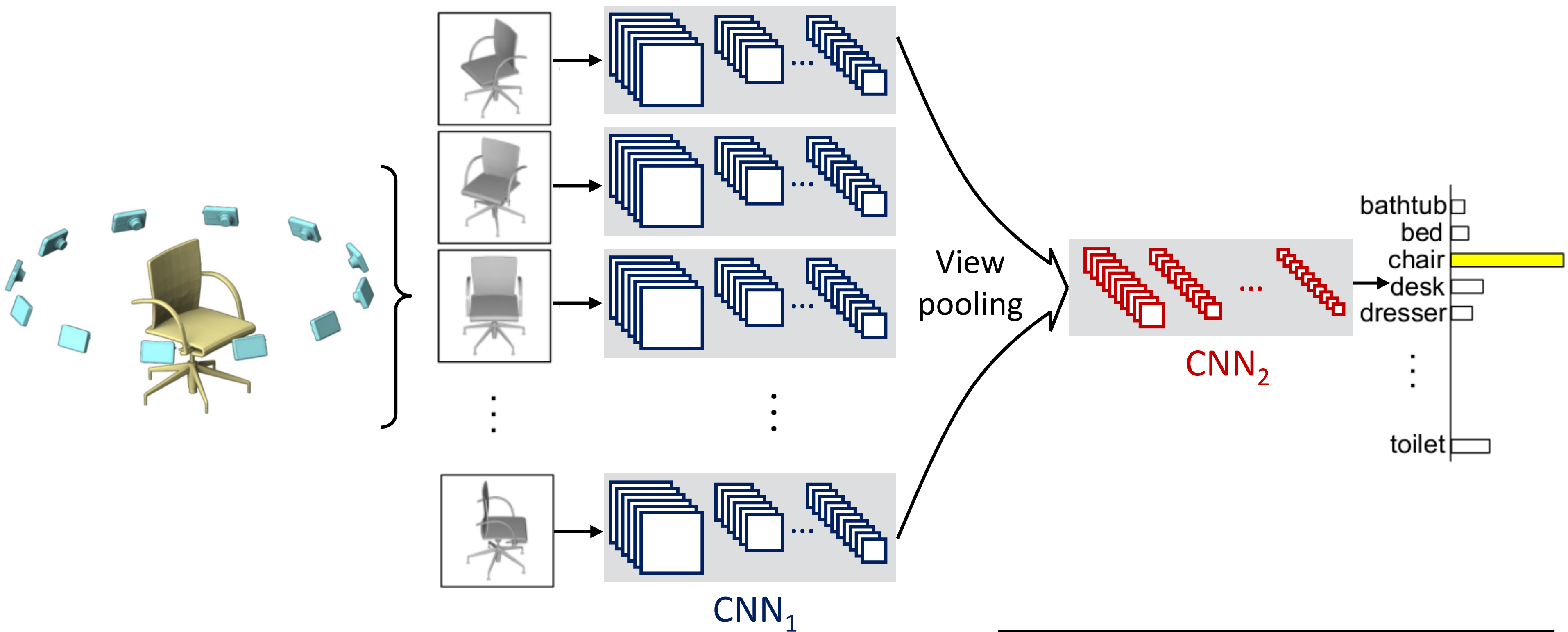
Multi-view CNNs for shape analysis



View pooling:
element-wise max-pooling
across all views

Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller
Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015

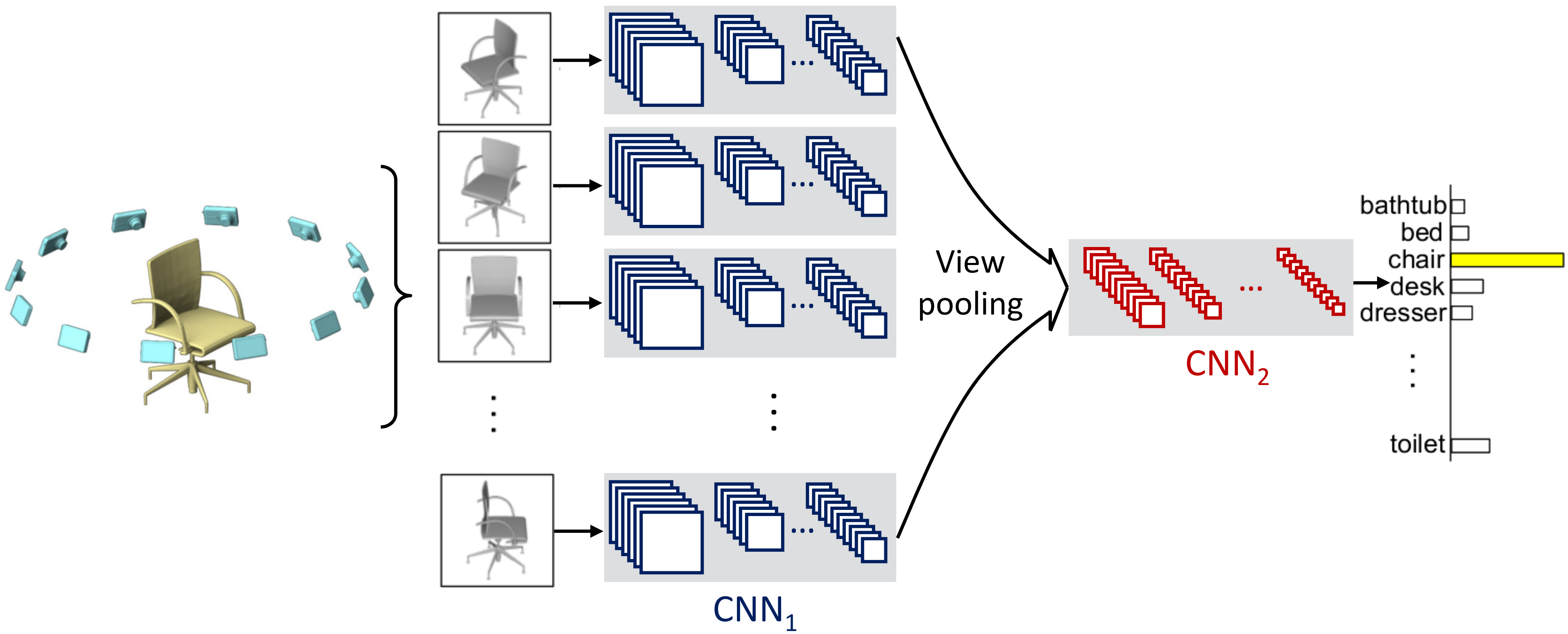
Multi-view CNNs for shape analysis



CNN$_2$: a second ConvNet producing shape descriptors

Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller
Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015

Multi-view CNNs for shape analysis



CNNs pre-trained on ImageNet **(leverage large image datasets for training shape analysis techniques!)**

Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015

Multi-view CNNs for shape analysis

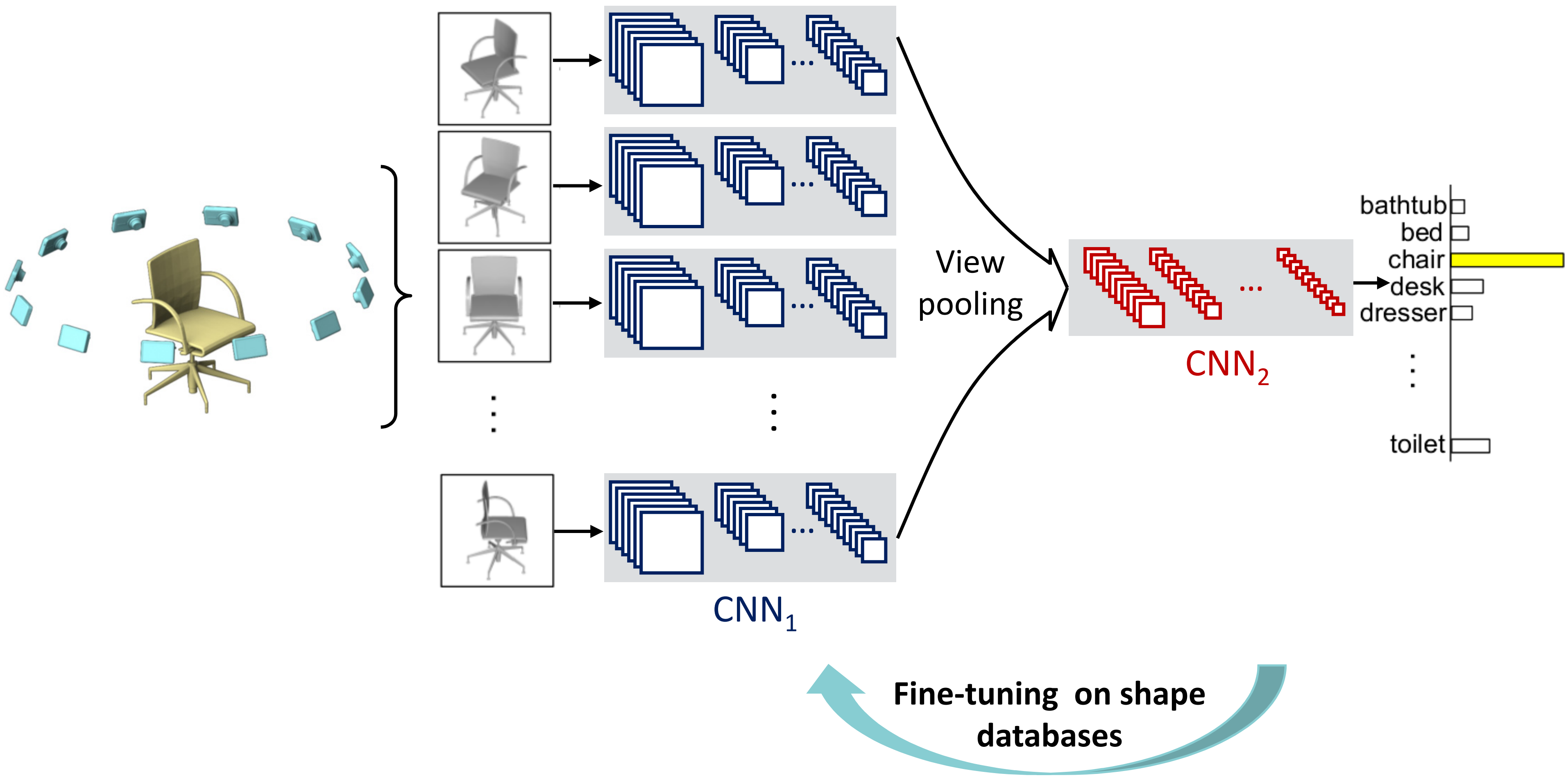


Image from Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller
Multi-view Convolutional Neural Networks for 3D Shape Recognition, ICCV 2015

Volumetric CNNs

Key idea: represent a shape as a volumetric image with binary voxels.

Learn filters operating on these volumetric data.

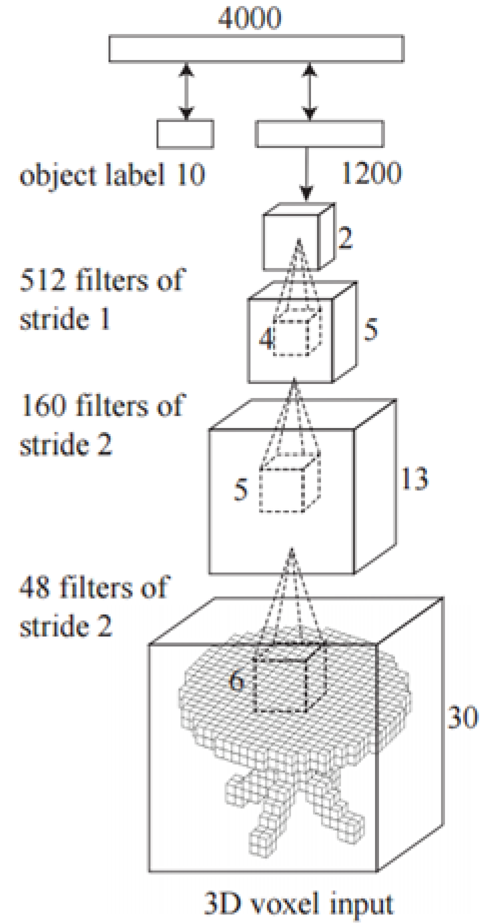
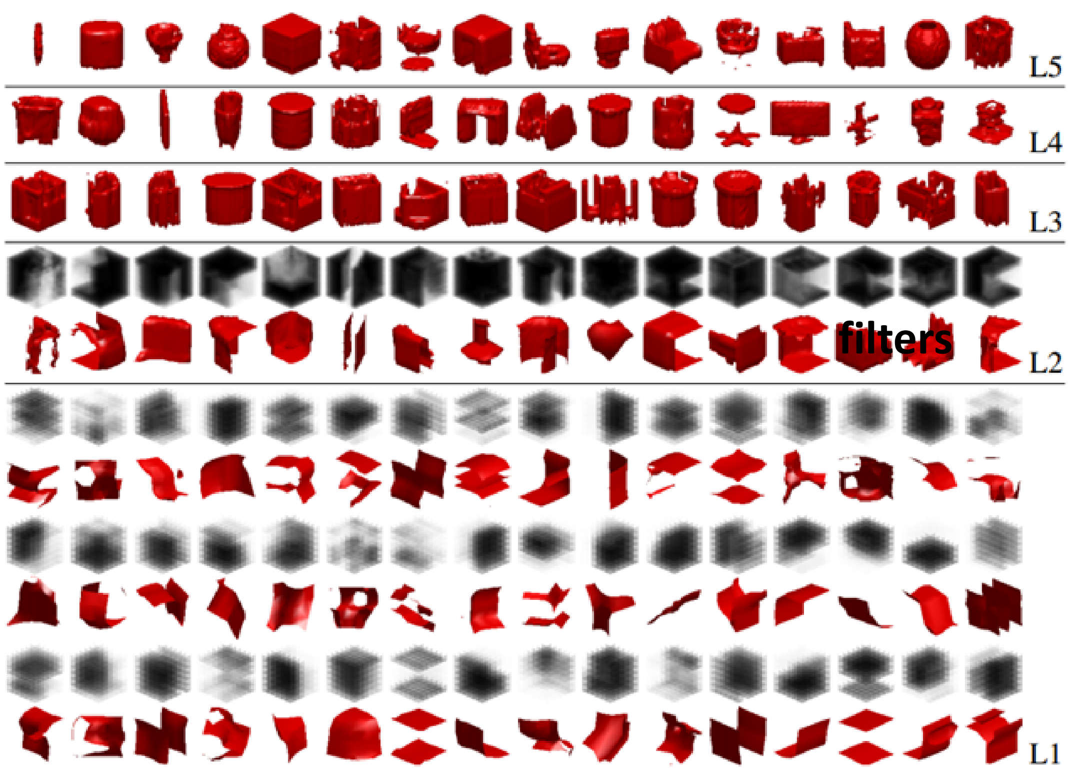


Image from Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao 3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015

Volumetric CNNs



Learned filters

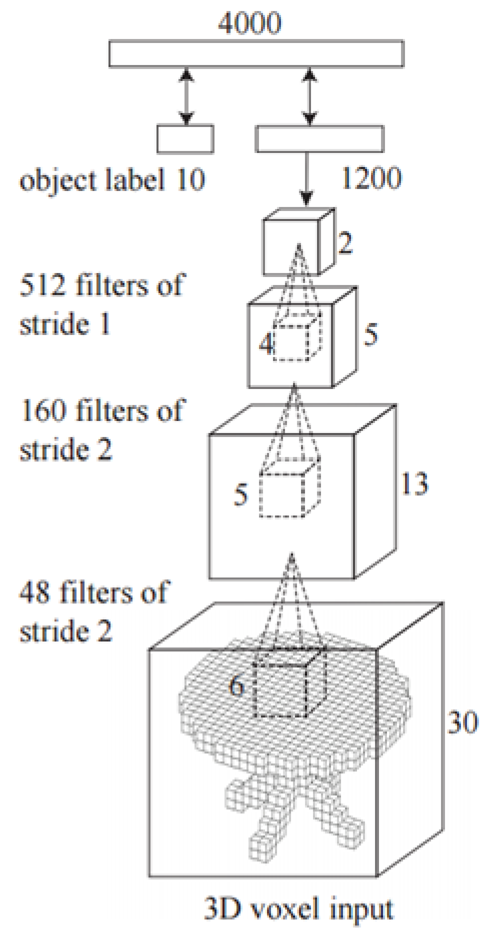


Image from Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao
3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015

Sketch-based 3D Shape Retrieval using CNNs

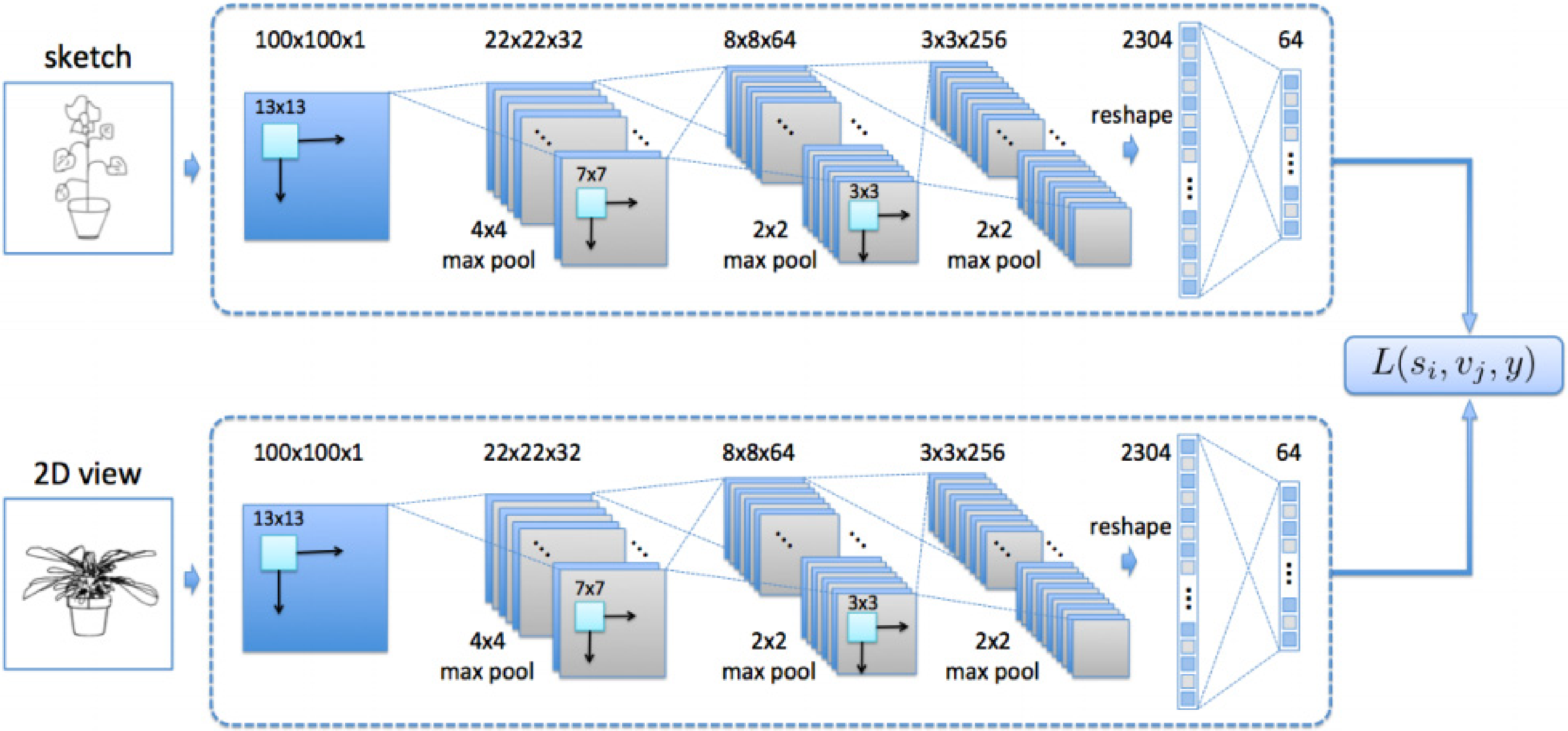


Image from Fang Wang, Le Kang, Yi Li,
Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, 2015

Sketch-based 3D Shape Retrieval using CNNs

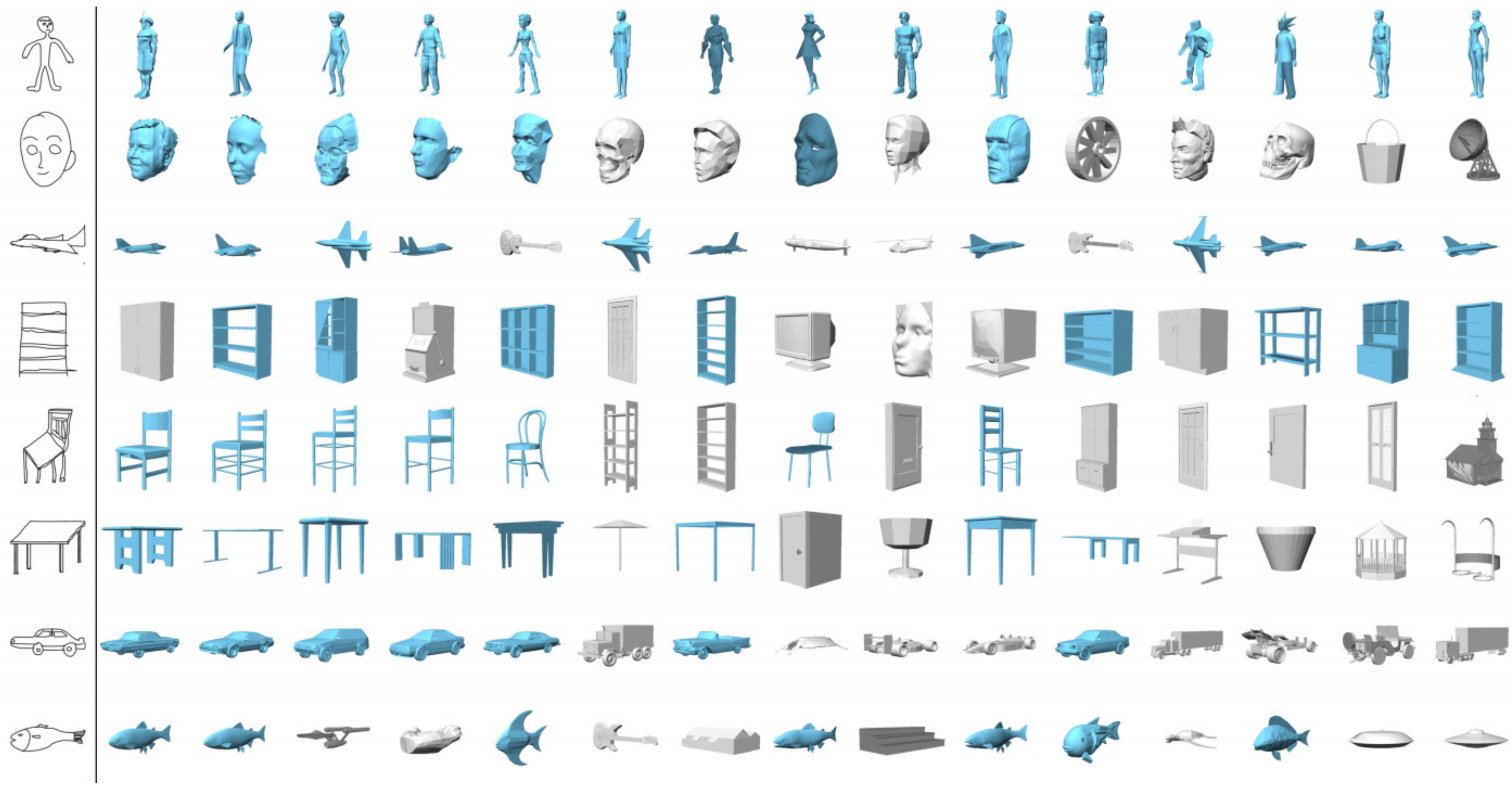
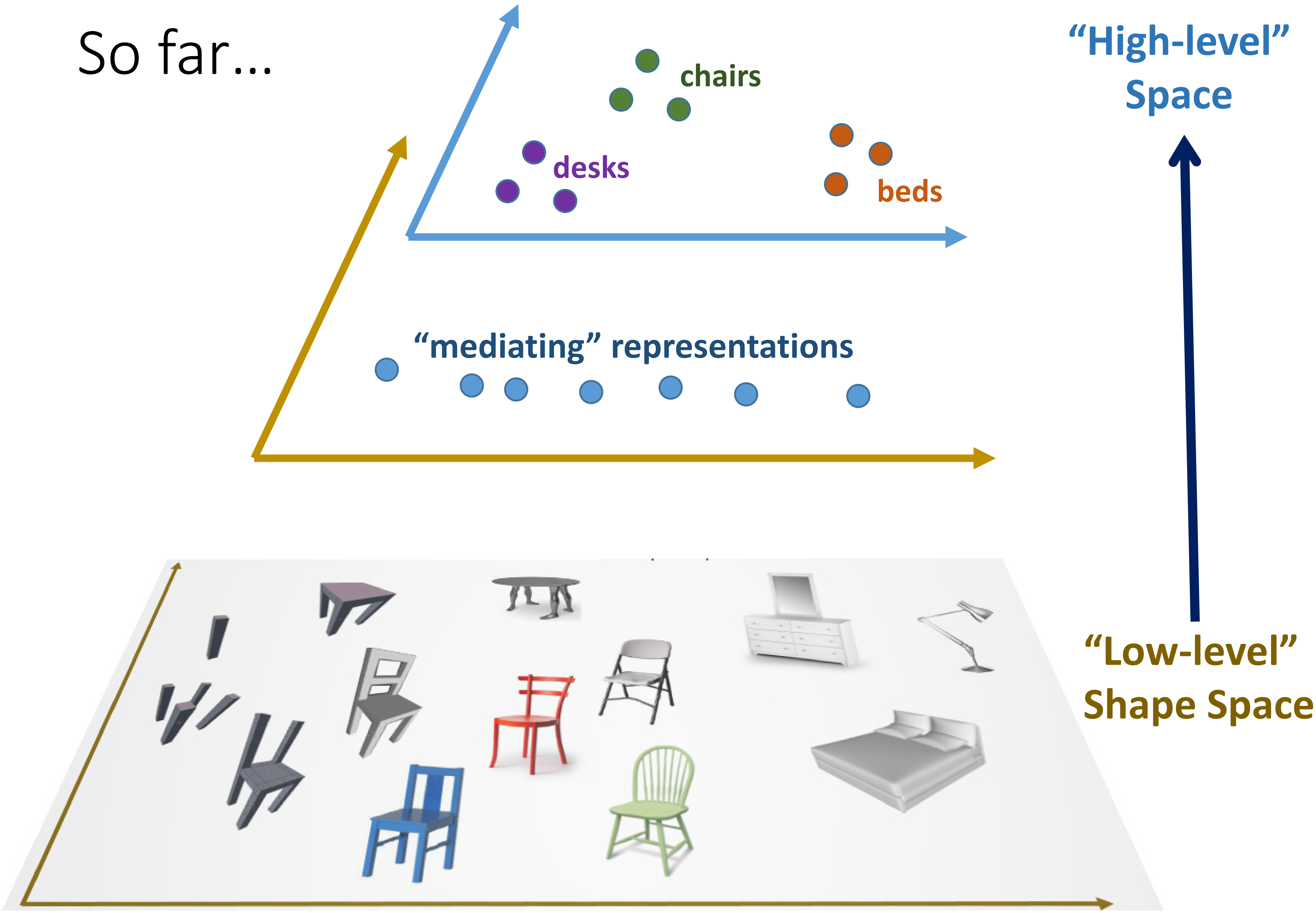
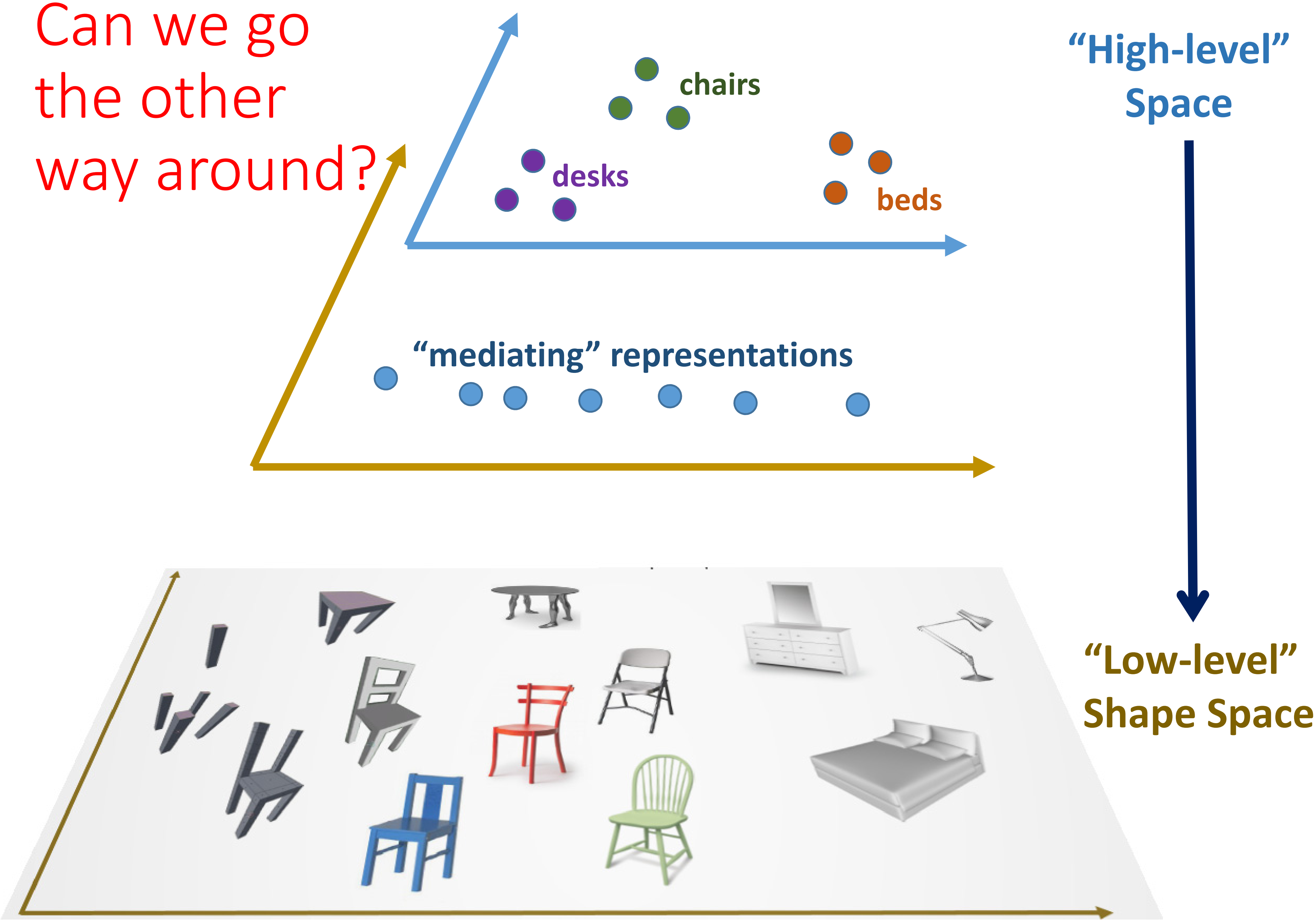


Image from Fang Wang, Le Kang, Yi Li,
Sketch-based 3D Shape Retrieval using Convolutional Neural Networks, 2015

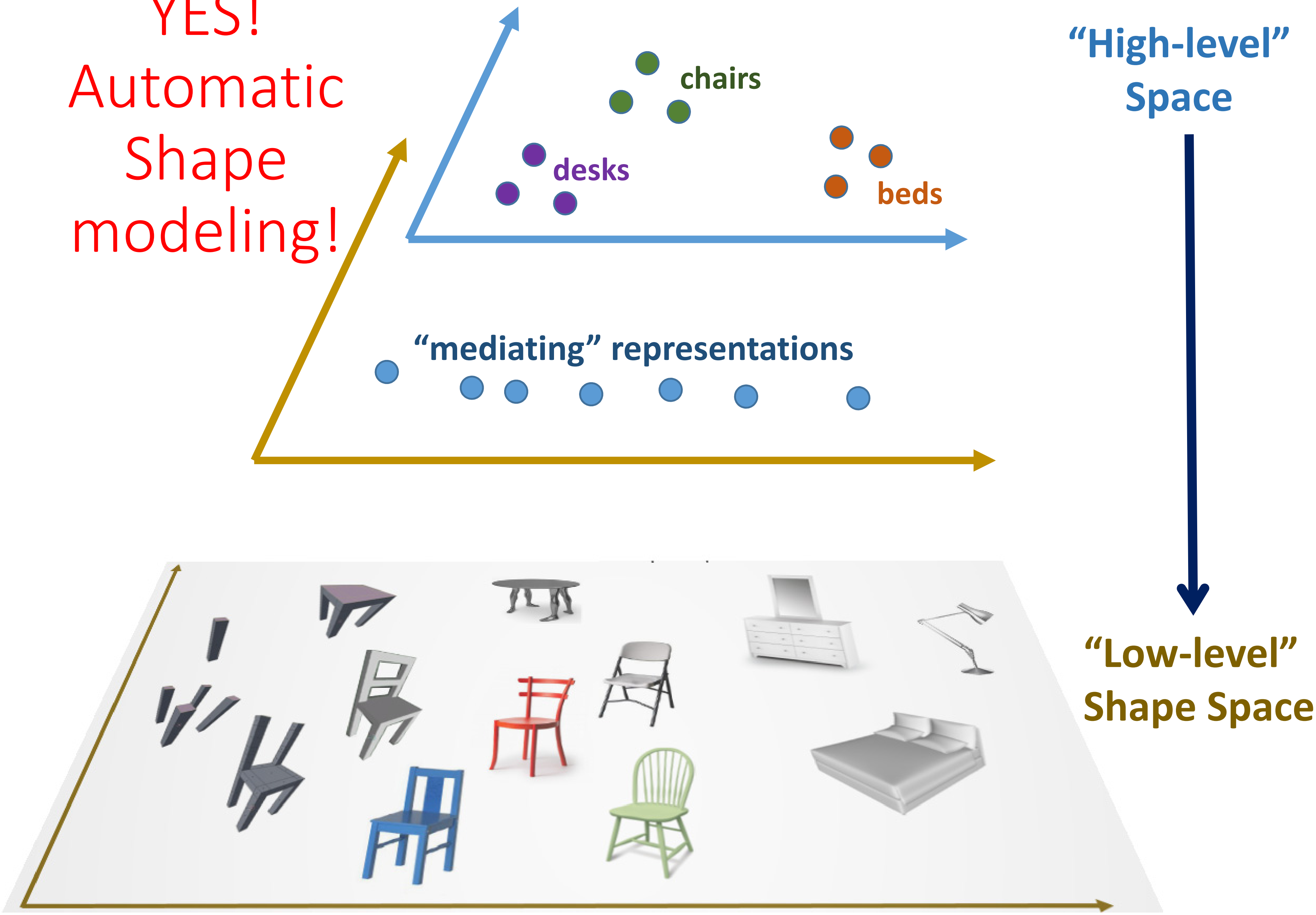
So far...


chairs
desks
beds
"High-level" Space
"mediating" representations
"Low-level" Shape Space

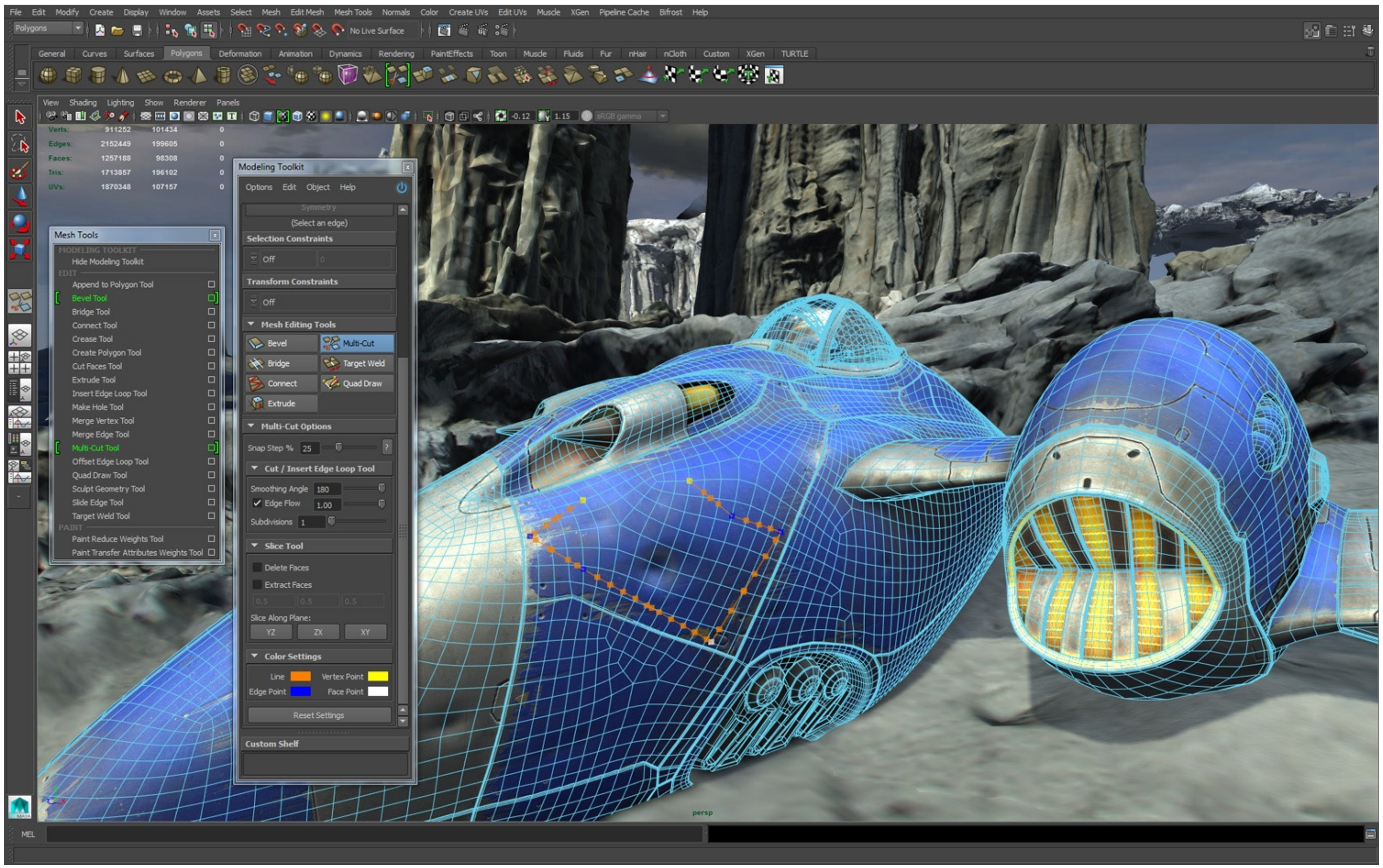
Can we go the other way around?



YES! Automatic Shape modeling!



Why automatic geometric modeling?
Because it is not easy!



Autodesk Maya 2015

“Traditional” Geometric Modeling

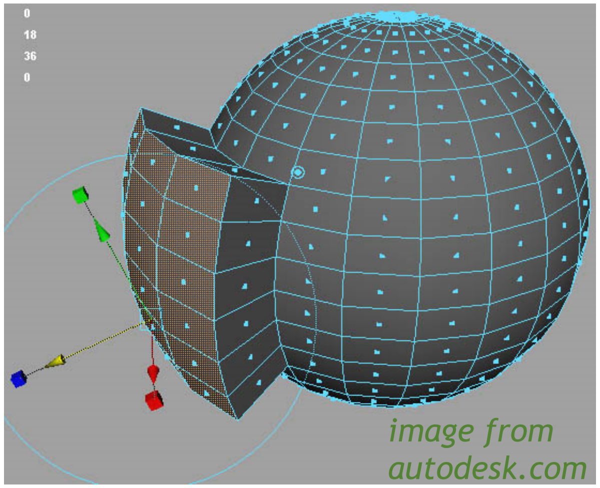


image from autodesk.com

Manipulating polygons

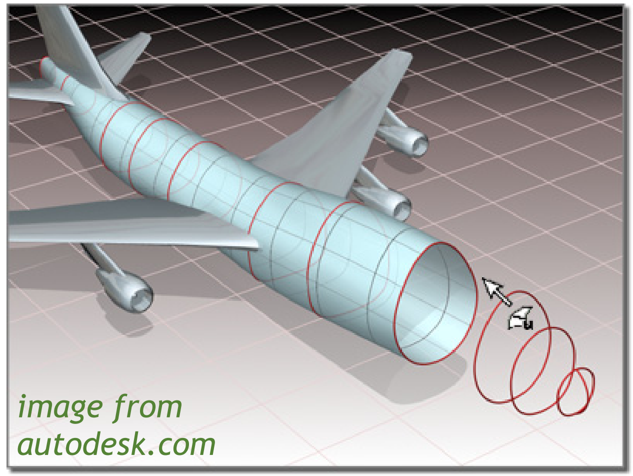


image from autodesk.com

Manipulating curves

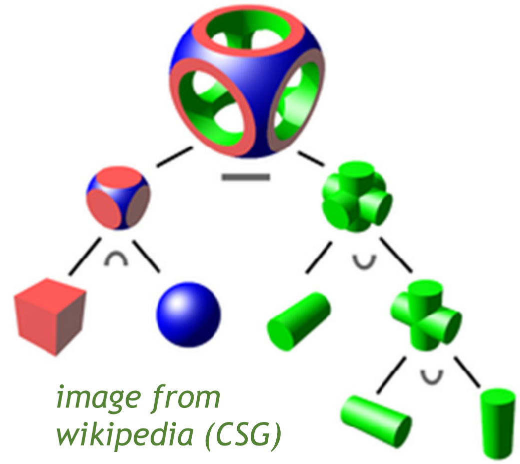
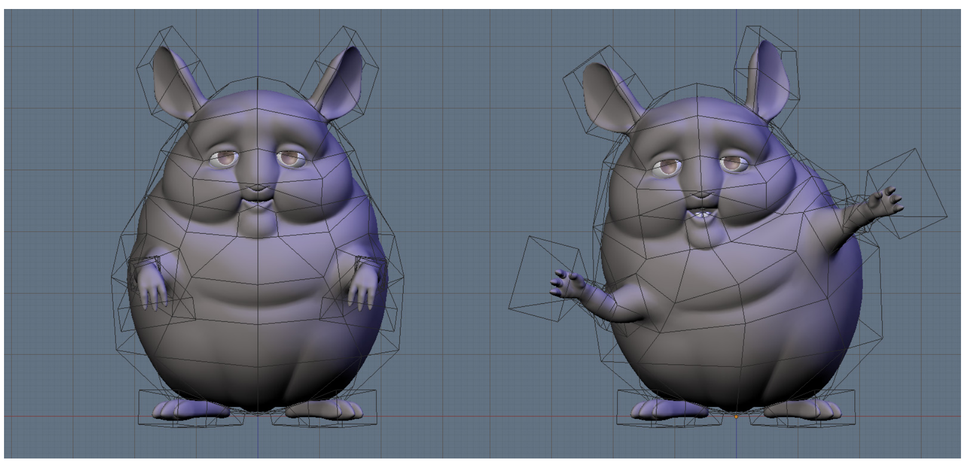
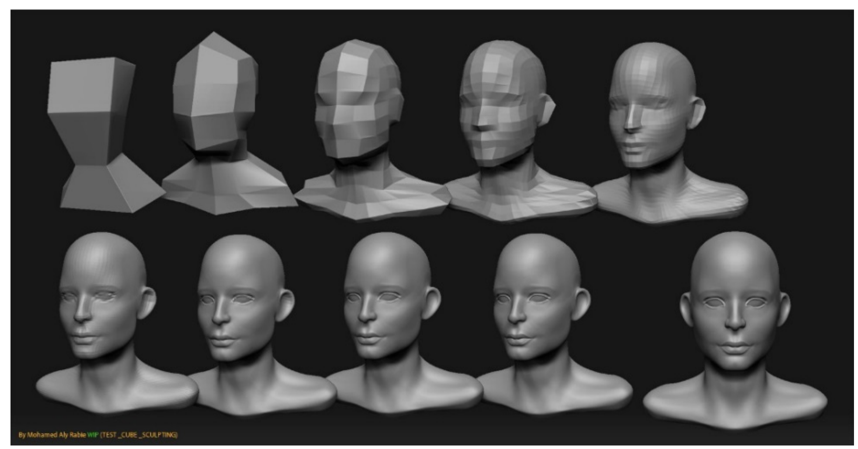


image from wikipedia (CSG)

Manipulating 3D primitives



Manipulating control points, cages *image from Blender*



Digital Sculpting *image from Mohamed Aly Rable*

“Traditional” Geometric Modeling

Impressive results at the hands of **experienced users**

Operations requires **exact and accurate input**

Creating compelling 3D models **takes lots of time**

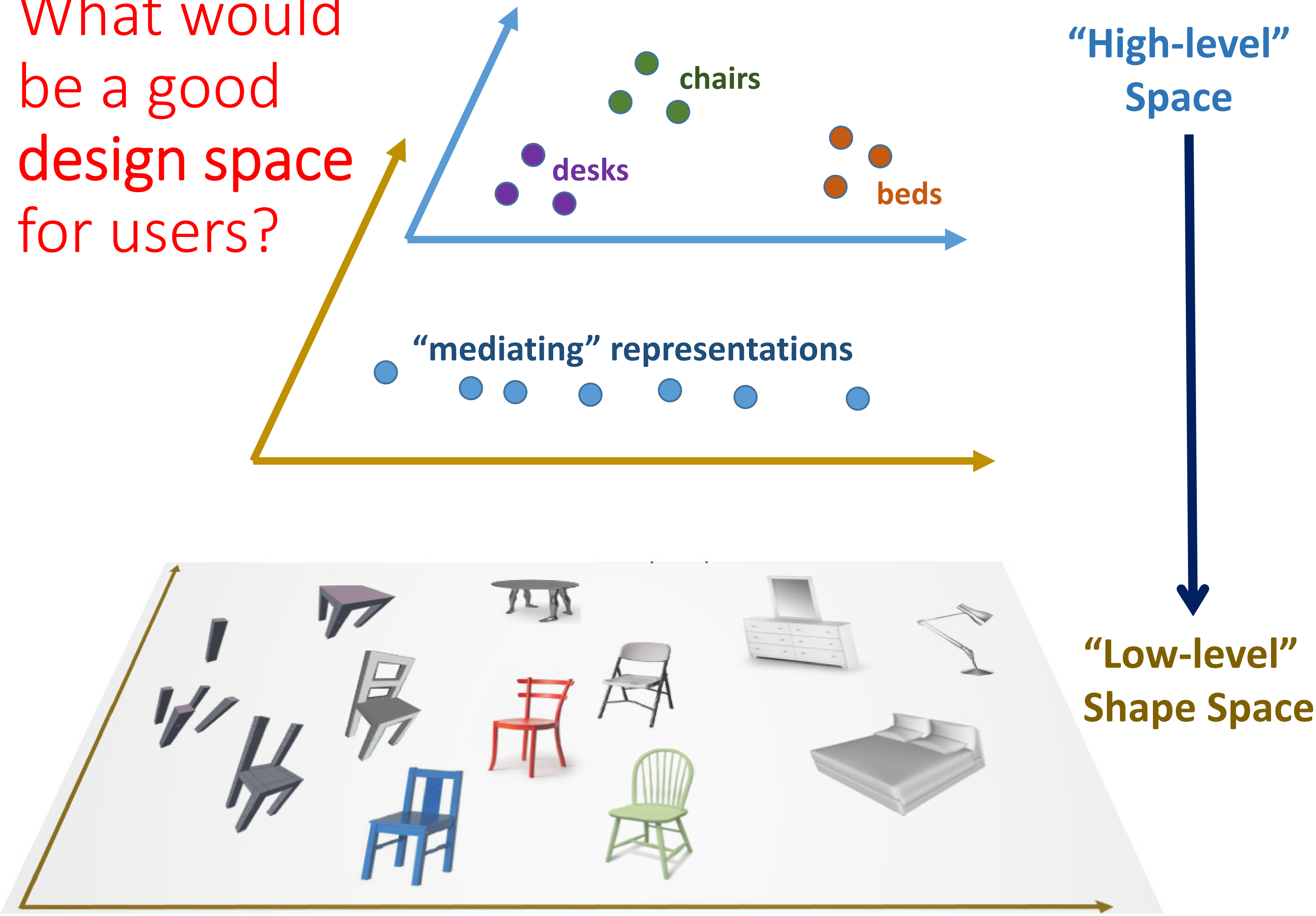
Tools usually have **steep learning curves**

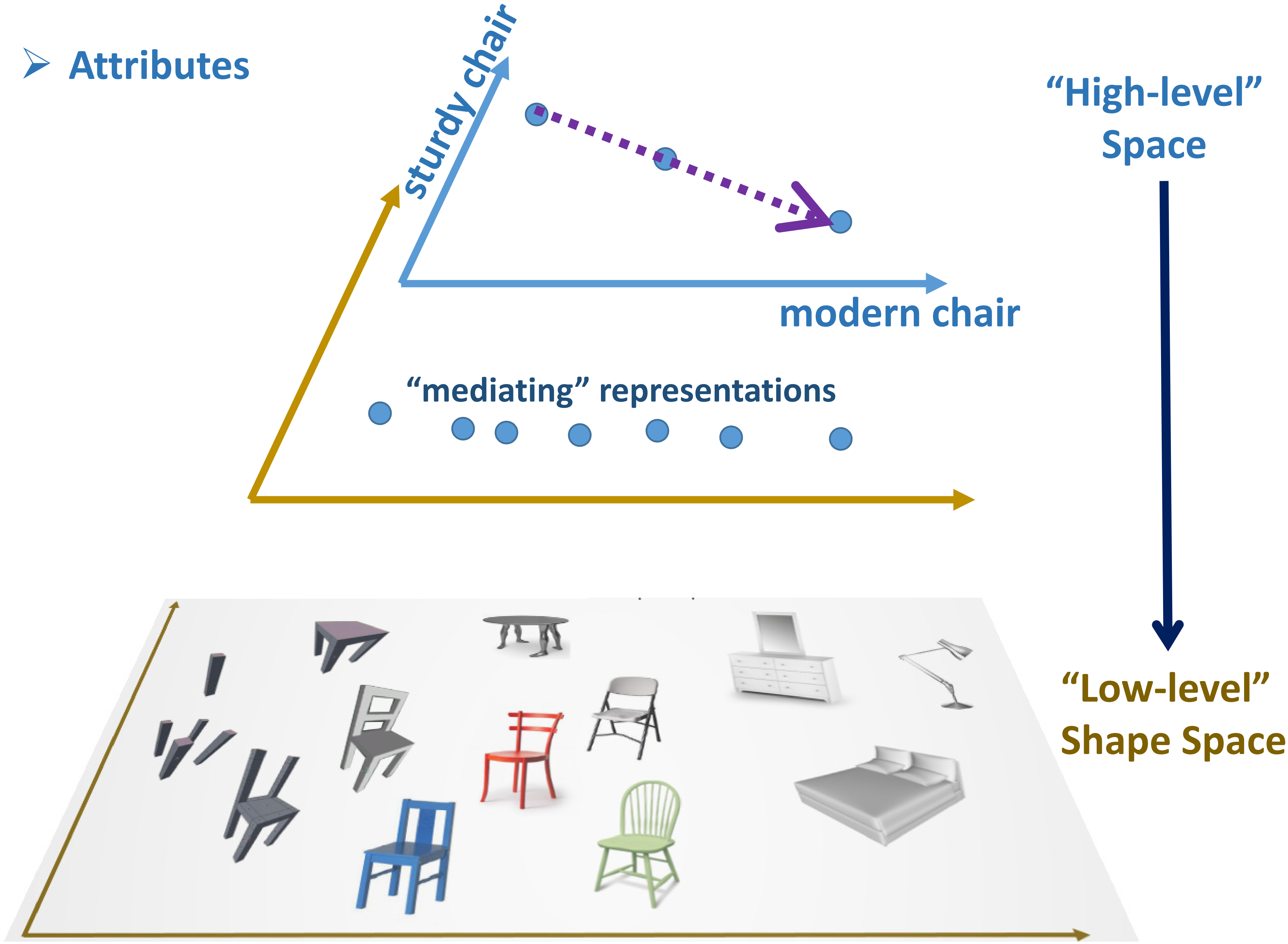
An alternative approach…

- Users provide high-level, possibly approximate input
- Computers learn to generate low-level, accurate geometry

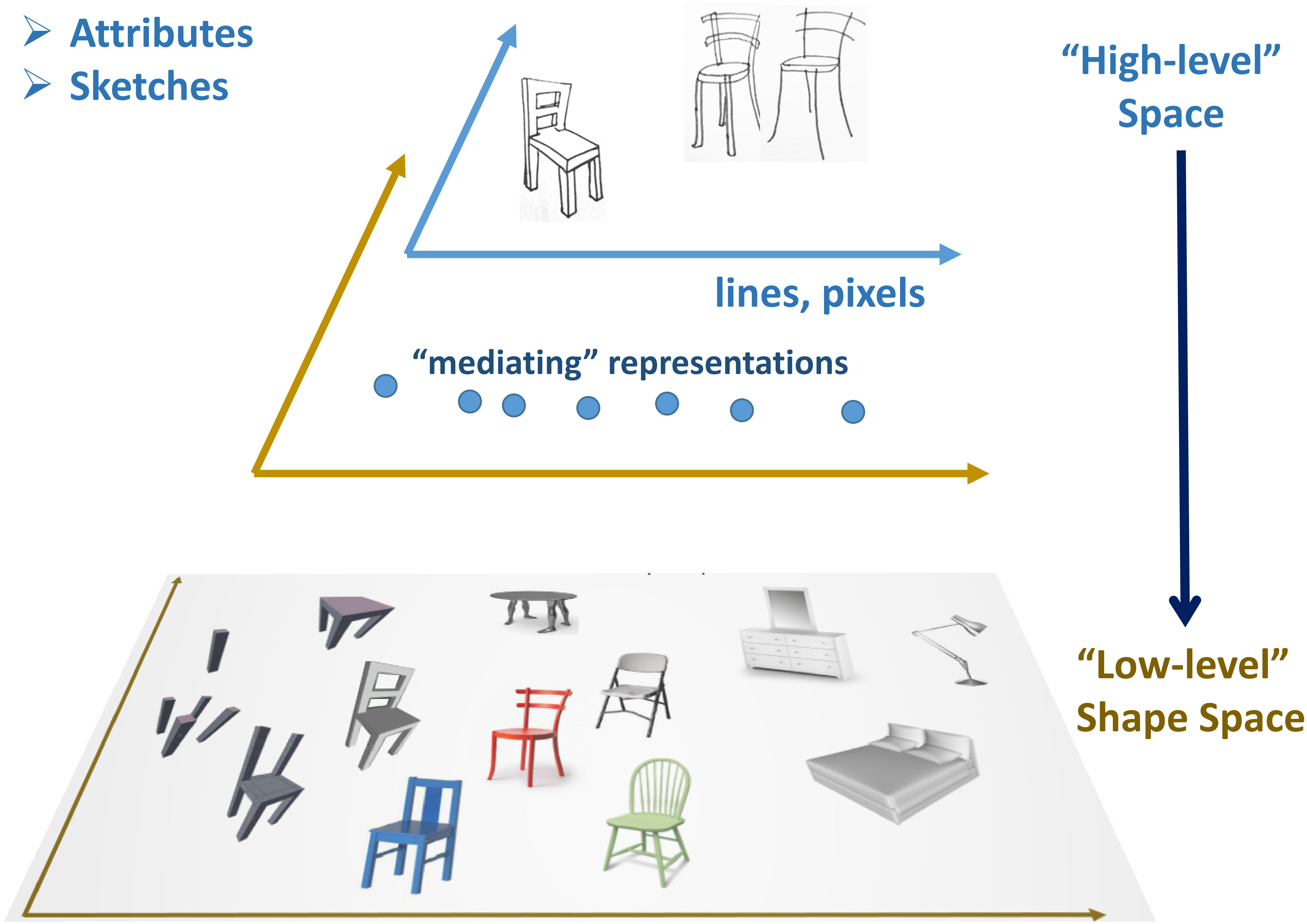
➢ **Machine learning!**

What would be a good **design space** for users?

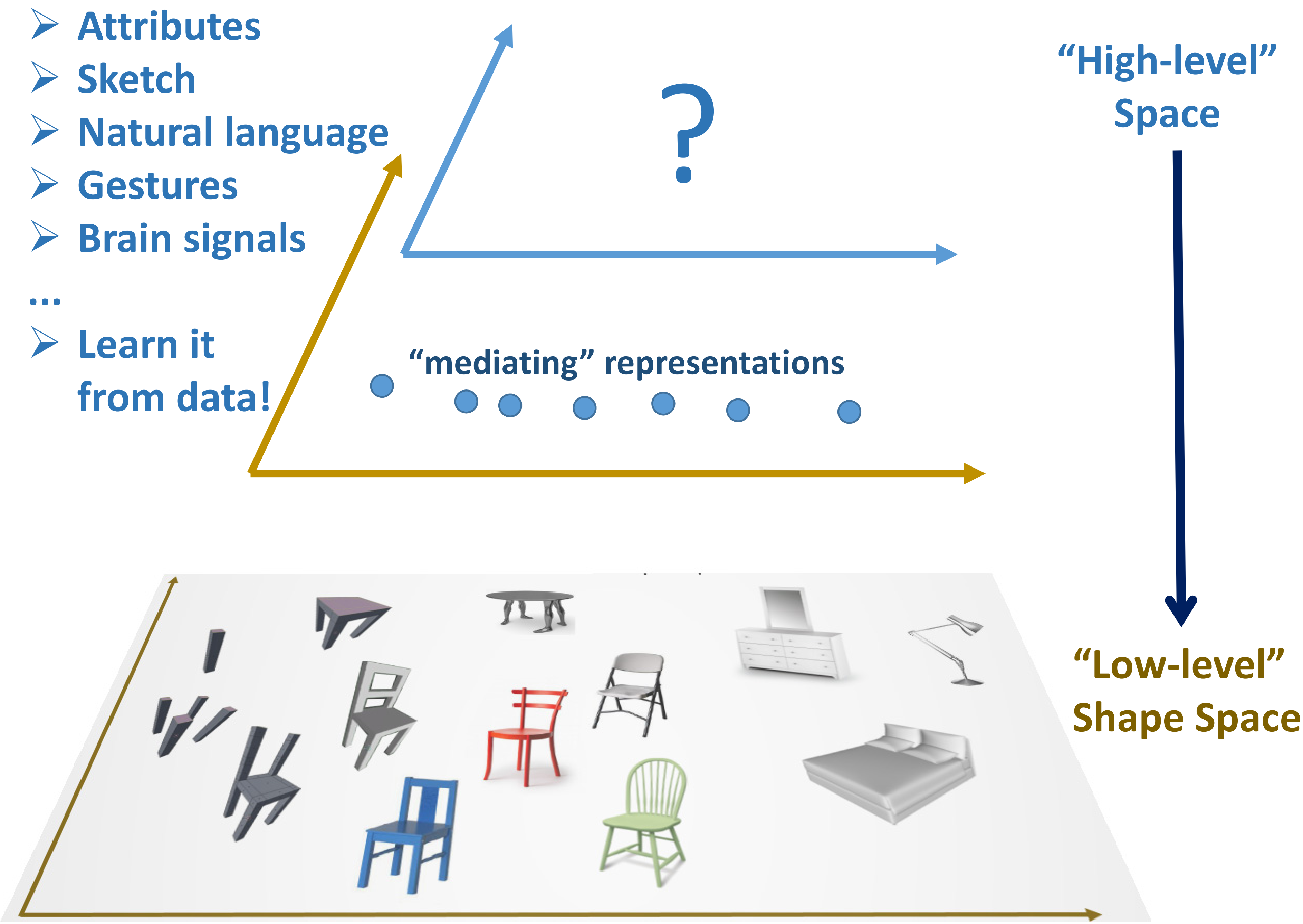




Attributes
sturdy chair
modern chair
"High-level" Space
"mediating" representations
"Low-level" Shape Space



Attributes
Sketches
"High-level"
Space
lines, pixels
"mediating" representations
"Low-level"
Shape Space



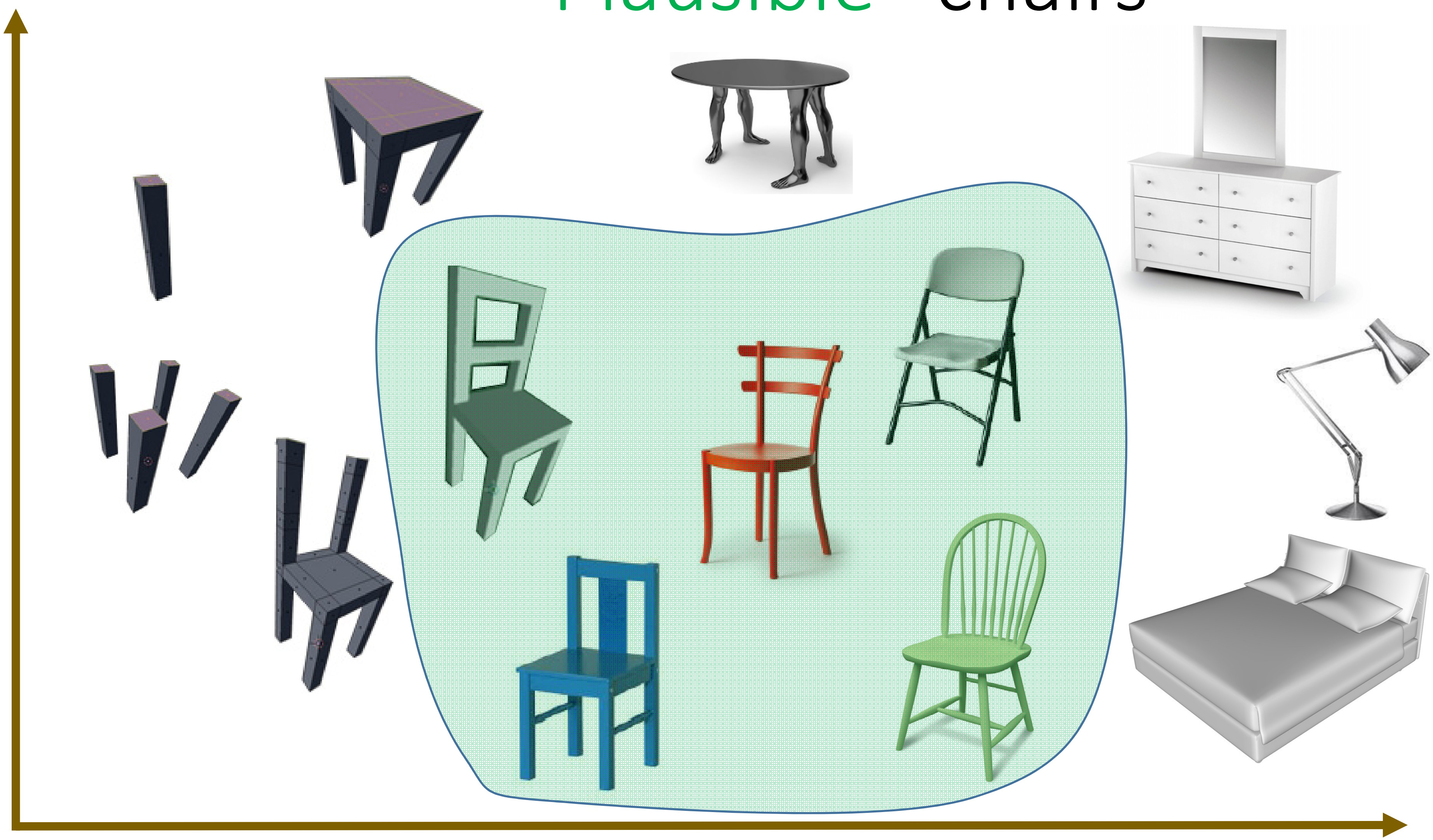
➢ Attributes
➢ Sketch
➢ Natural language
➢ Gestures
➢ Brain signals
…
➢ Learn it from data!
?
“High-level” Space
“mediating” representations
“Low-level” Shape Space

Machine learning for Geometric Modeling

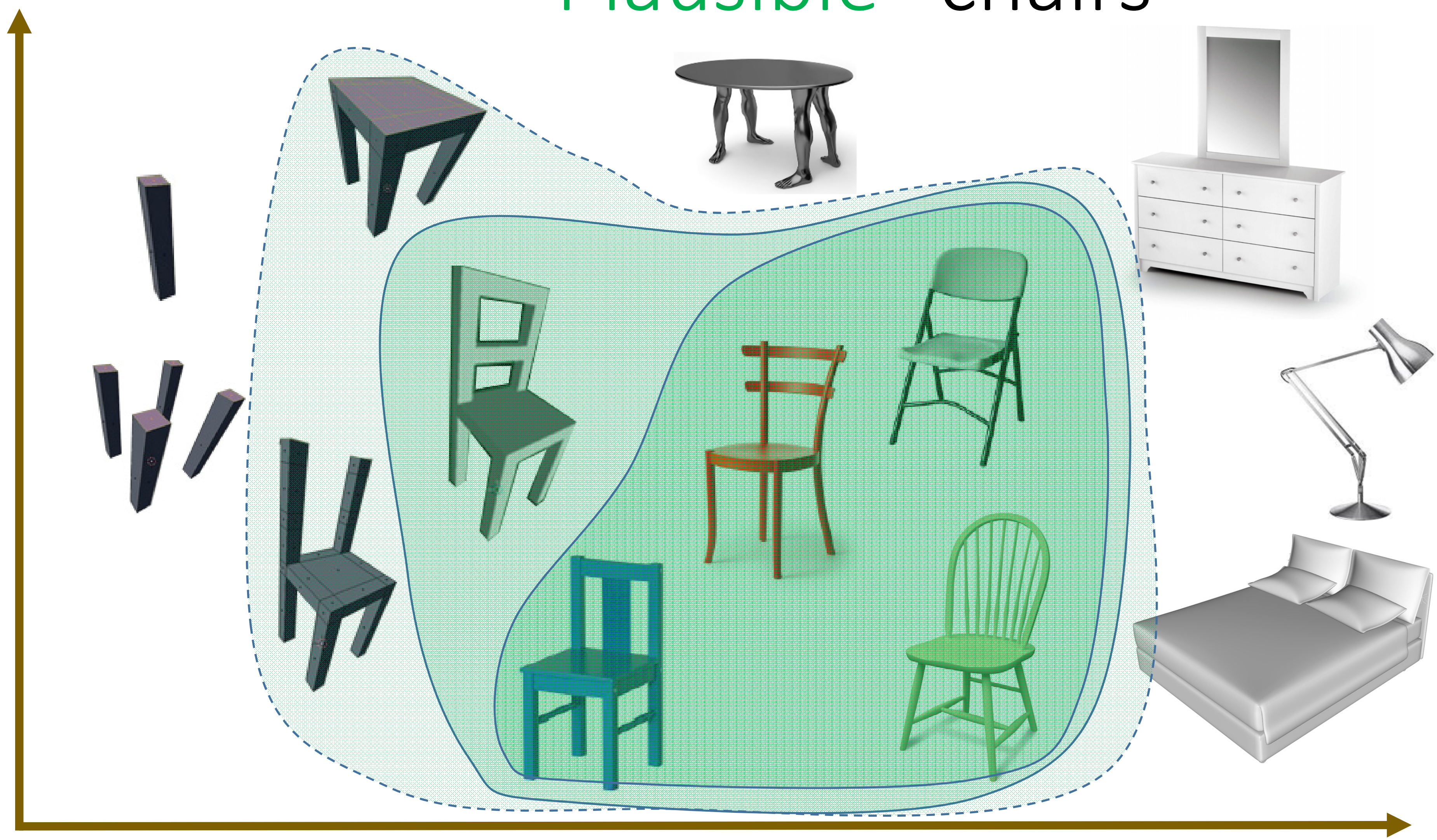
- Learn mappings from design ("high-level") to "low-level" space: $y = f(x)$

- Learn which shapes are probable ("plausible") given input: $P(y \mid f(x))$

"Plausible" chairs



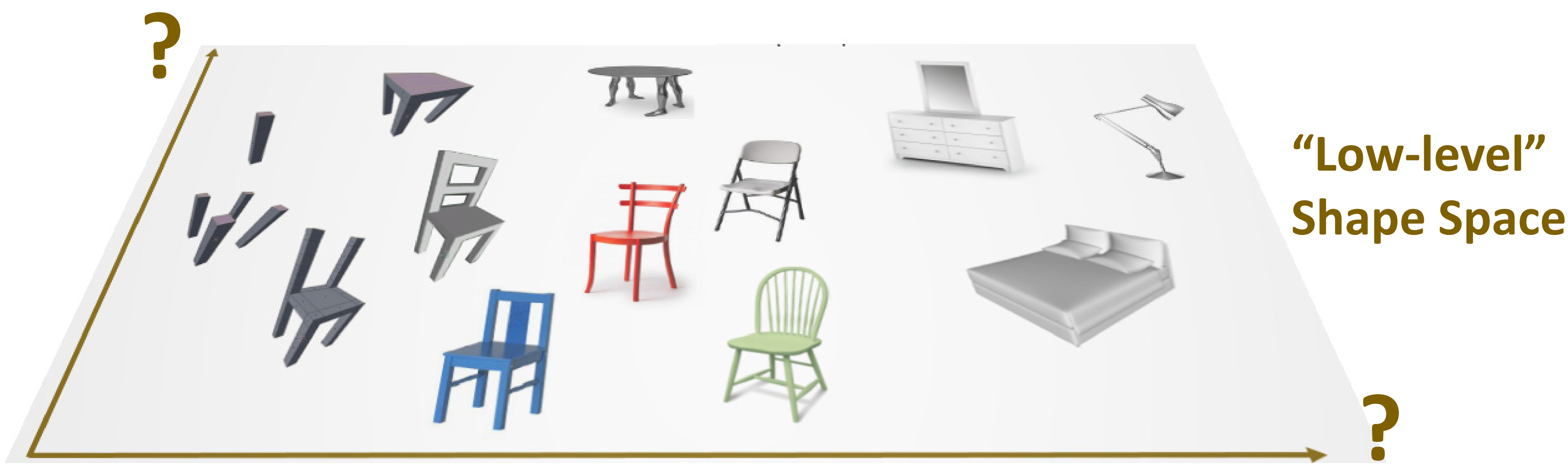
“Plausible” chairs



(not a binary or even an objective choice!)

The representation challenge

How do we represent the **shape space**?

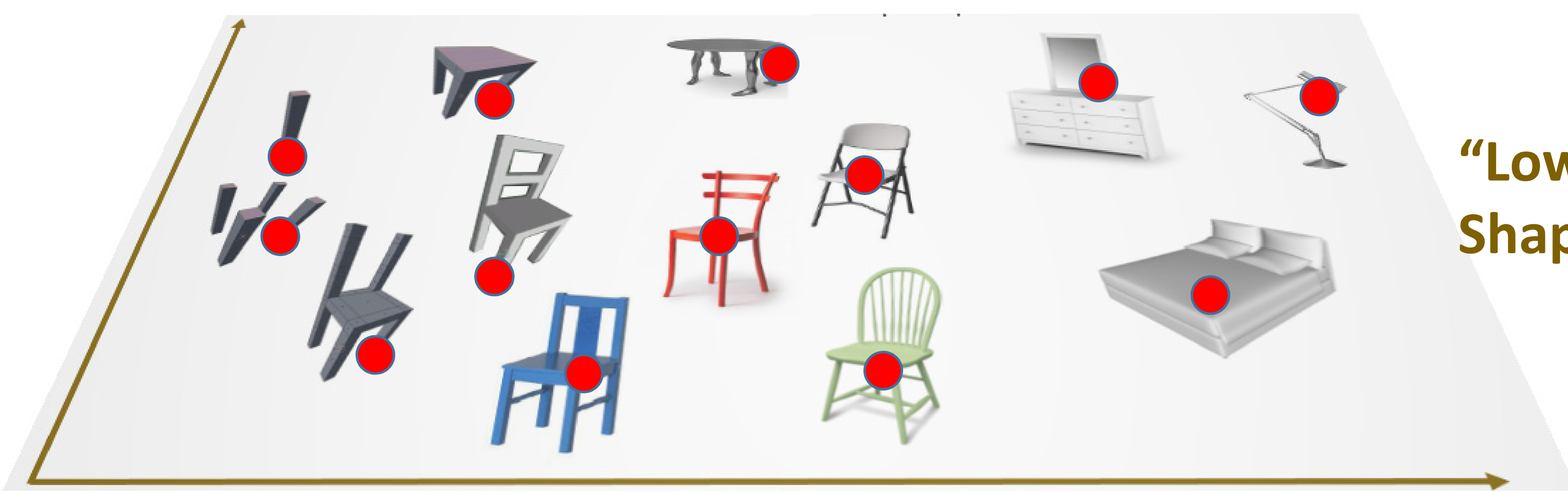


“Low-level” shape space representation

Can we use the polygon meshes as-is for our shape space?

No. Take the first vertex on each mesh. Where is it?

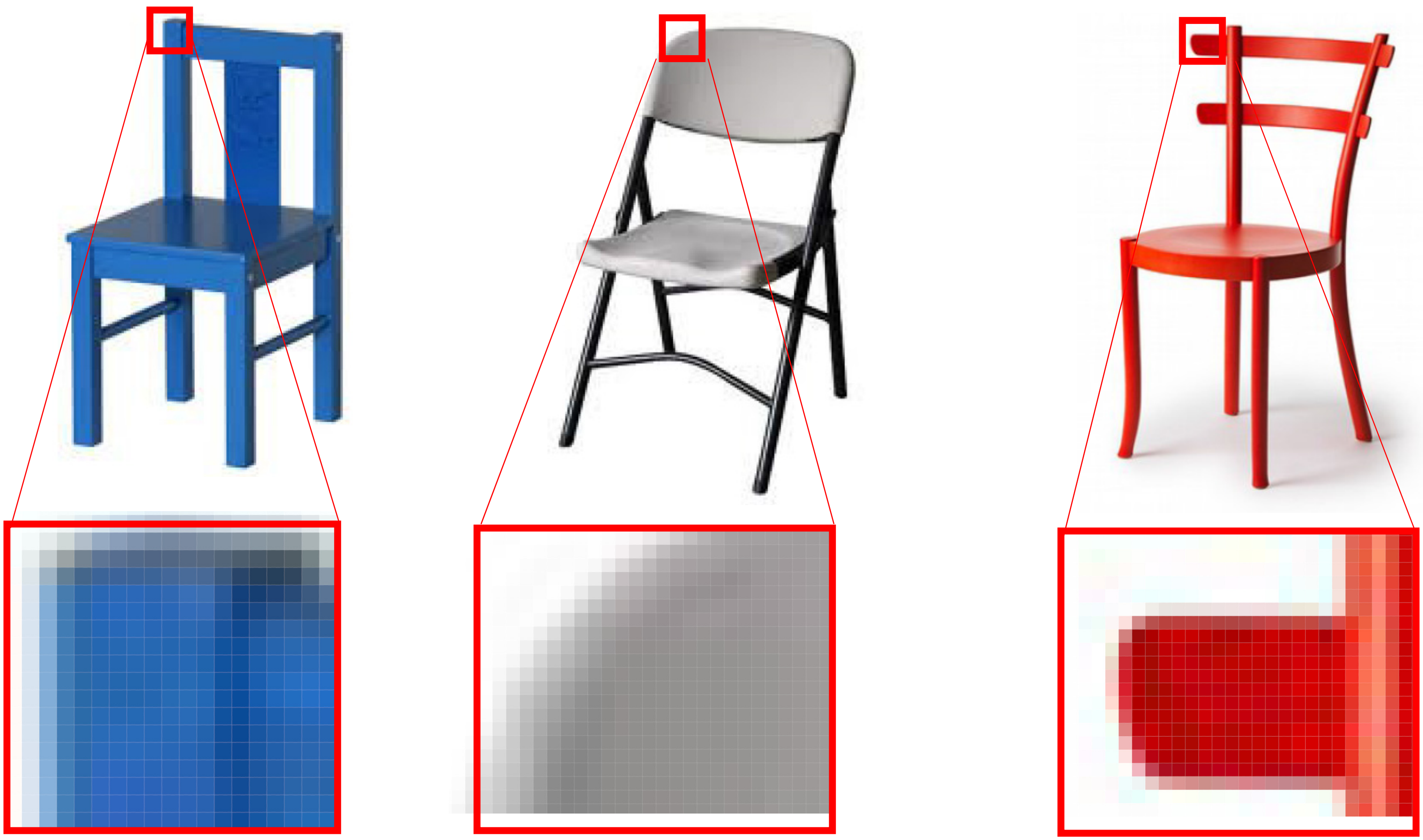
Meshes have different number of vertices, faces etc



“Low-level” Shape Space

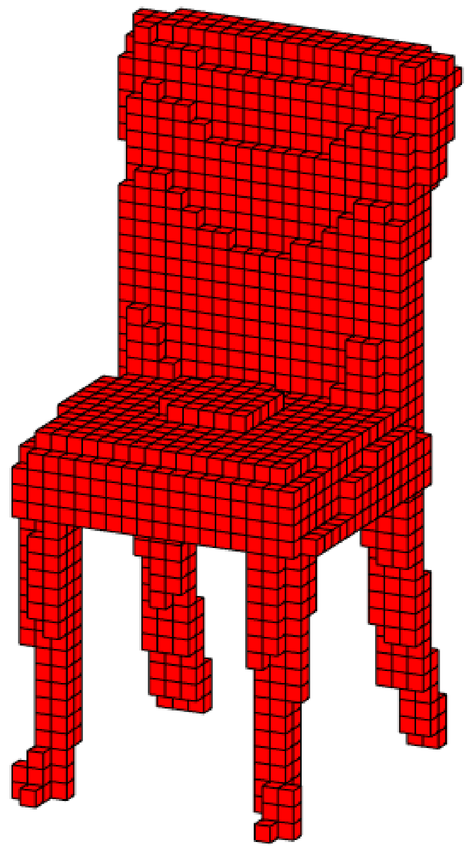
The “computer vision” approach

Learn mappings to pixels & multiple views!



The "volumetric" approach

Learn mappings to voxels!



The “correspondences” approach

Find point correspondences between 3D surface points. Can do aligment. Can we always have dense correspondences?

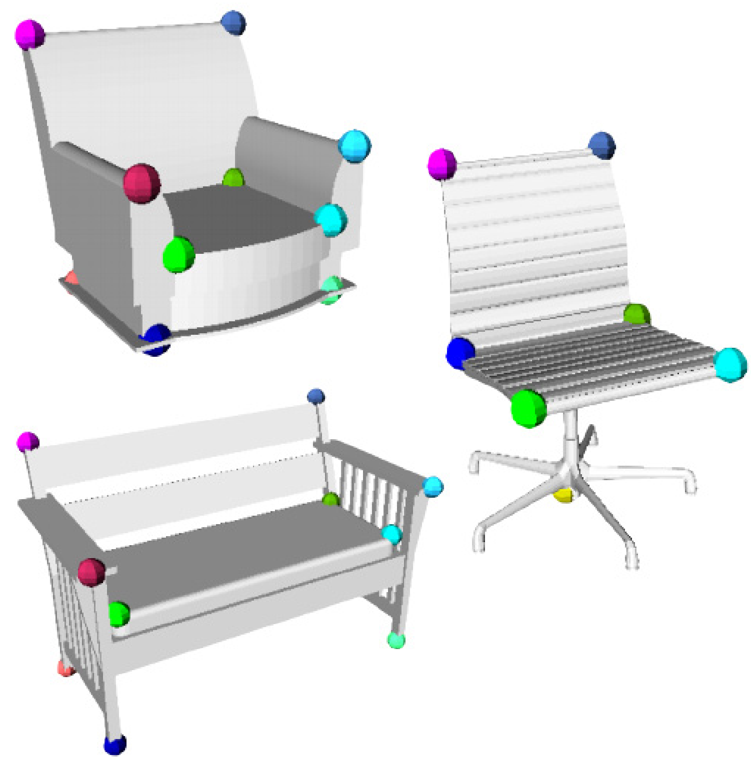


Image from Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser, “Learning Part-based Templates from Large Collections of 3D Shapes”, 2013

The “abstractions” approach

Parameterize shapes with primitives (cuboids, cylinders etc)
How can we capture surface detail?

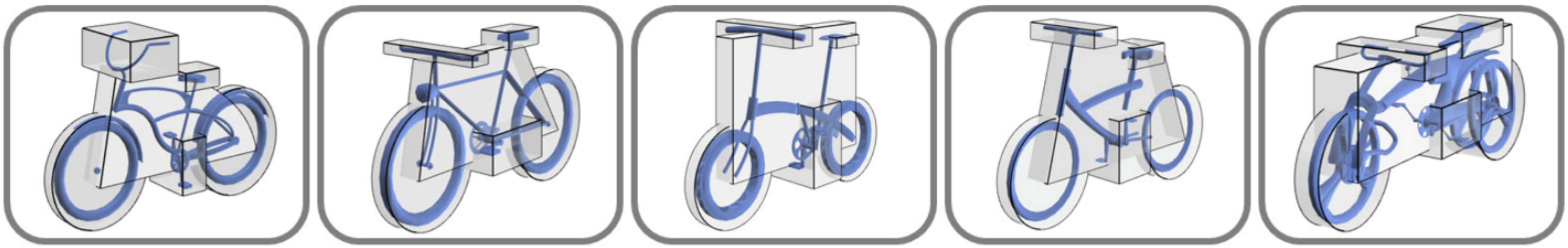


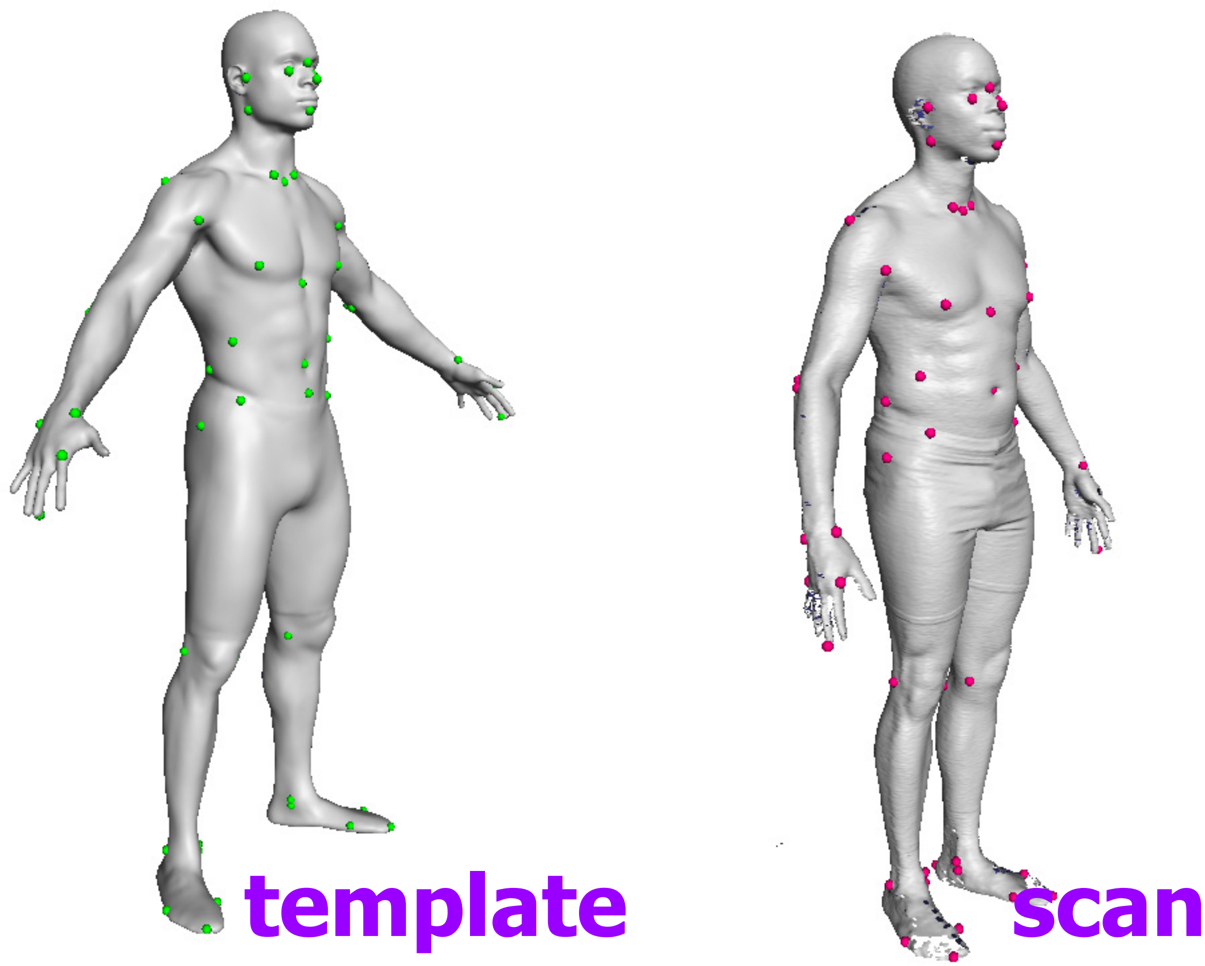
Image from E. Yumer., L. Kara, Co-Constrained Handles for Deformation in Shape Collections, 2014

Case study: the space of human bodies

Training shapes: 125 male + 125 female scanned bodies

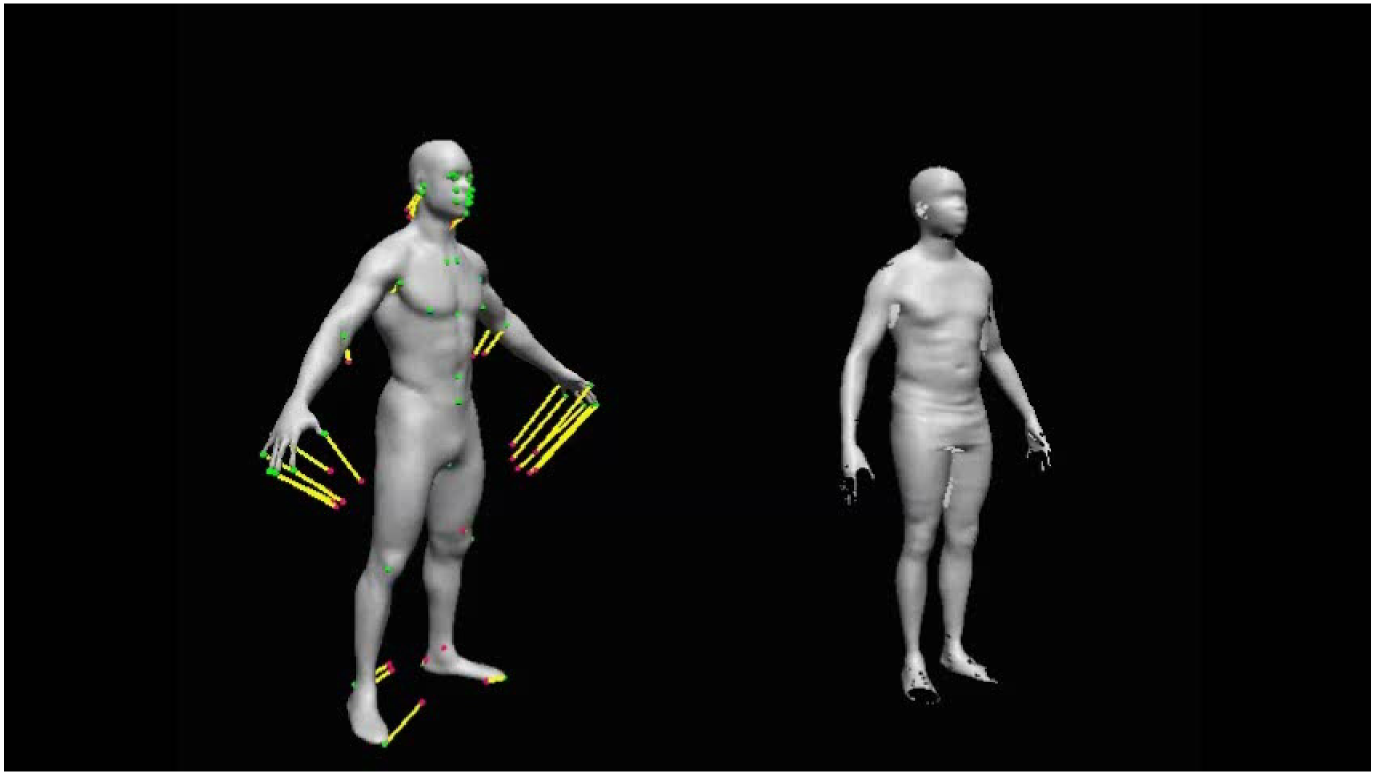
Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Matching algorithm



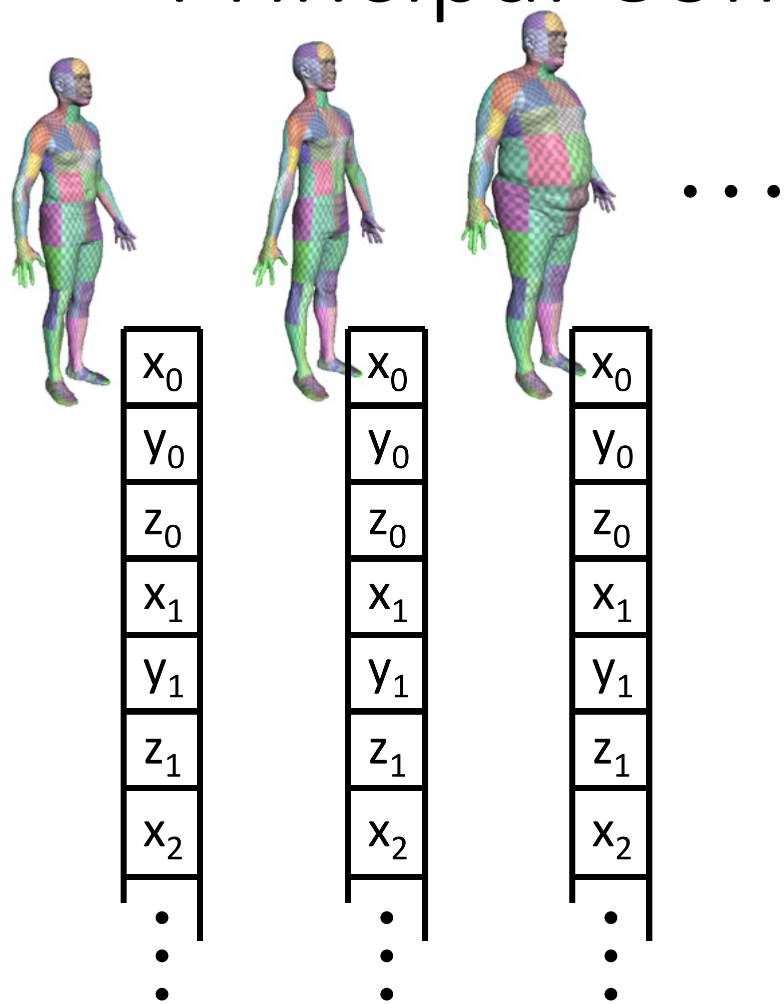
Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Matching algorithm



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

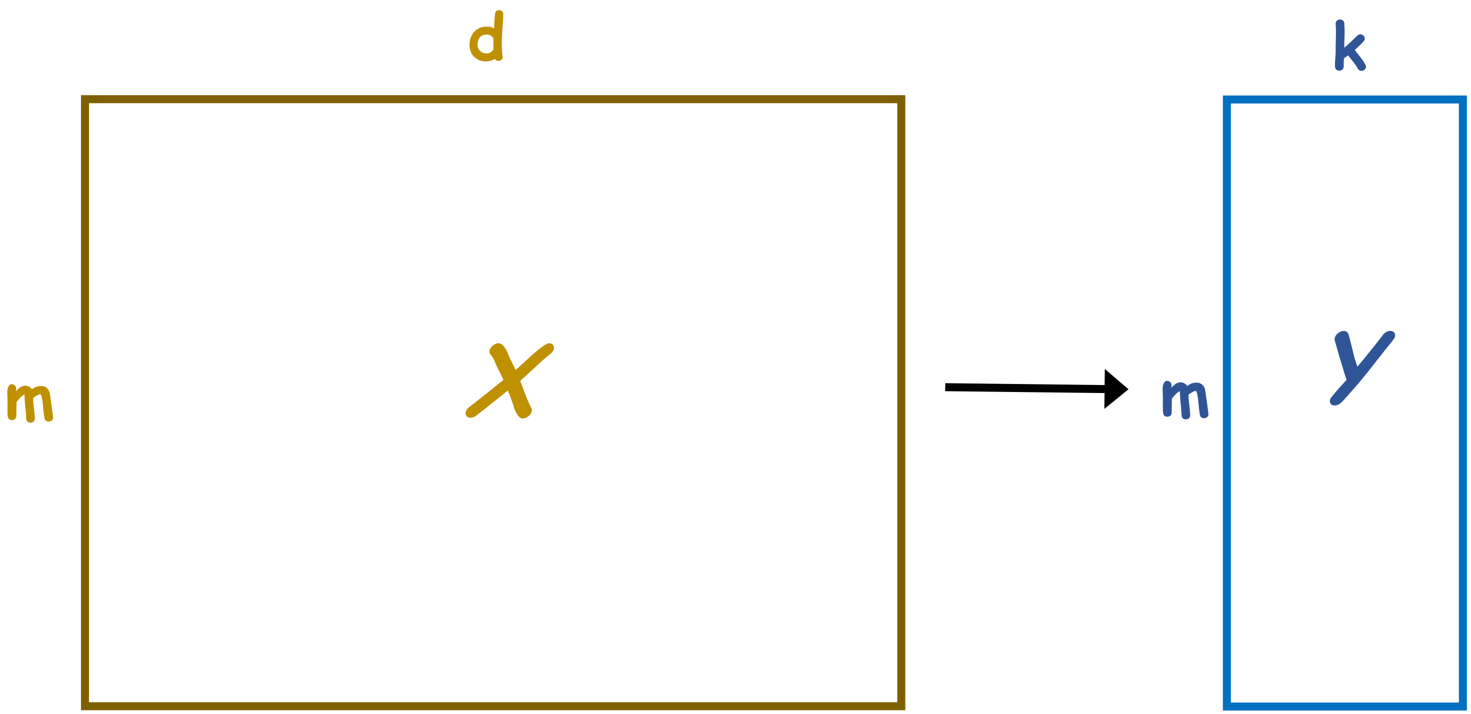
Principal Component Analysis



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

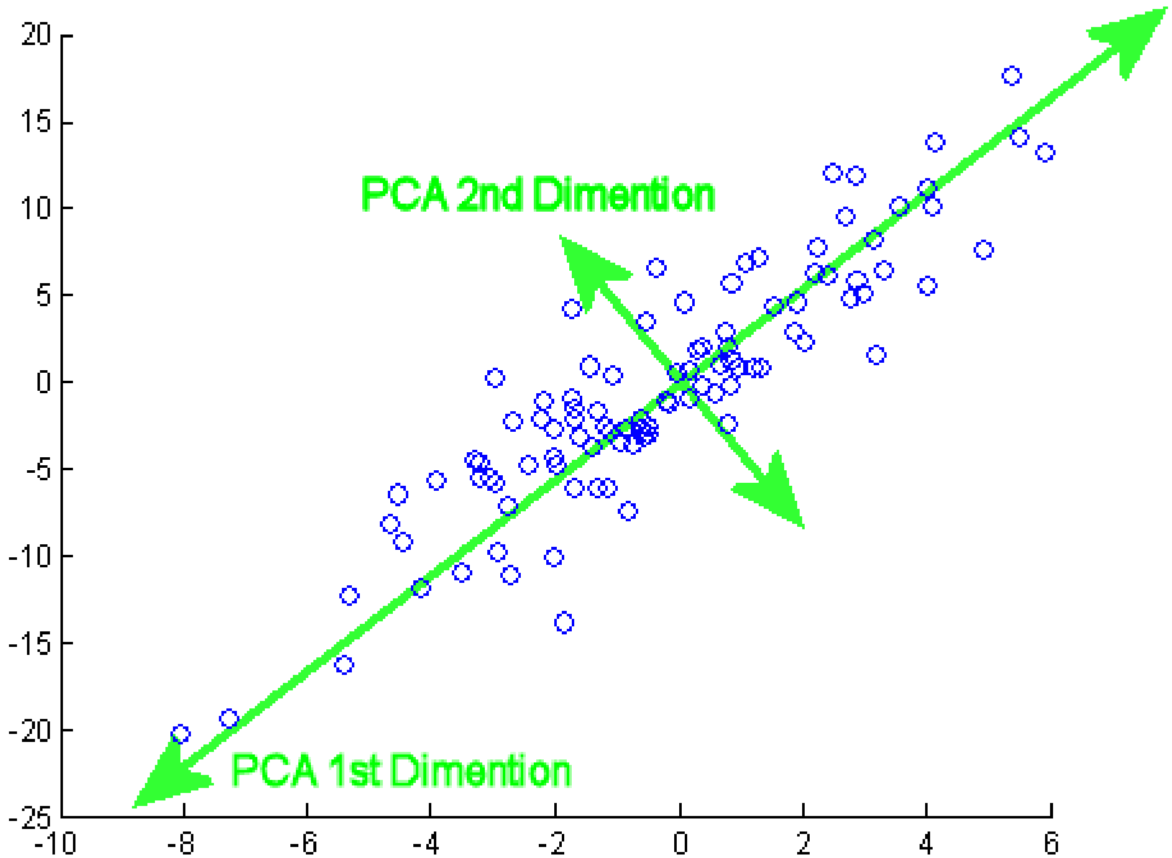
Dimensionality Reduction

Summarization of data with many (d) variables by a smaller set of (k) derived latent variables.

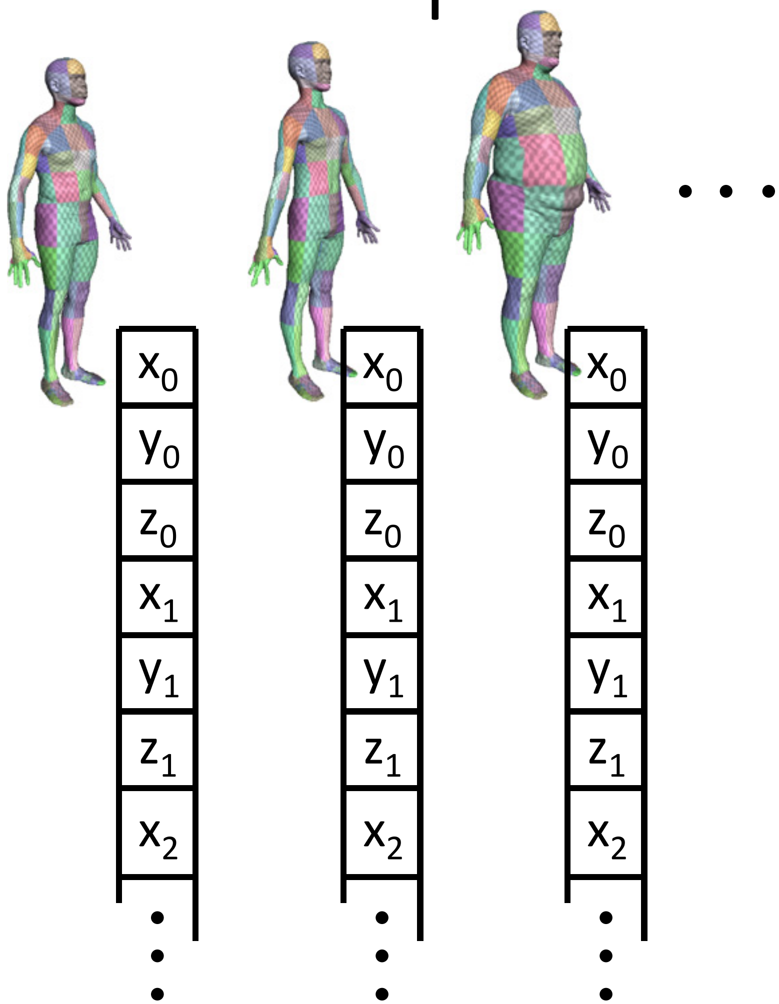


Principal Component Analysis

Each principal axis is a linear combination of the original variables



Principal Component Analysis

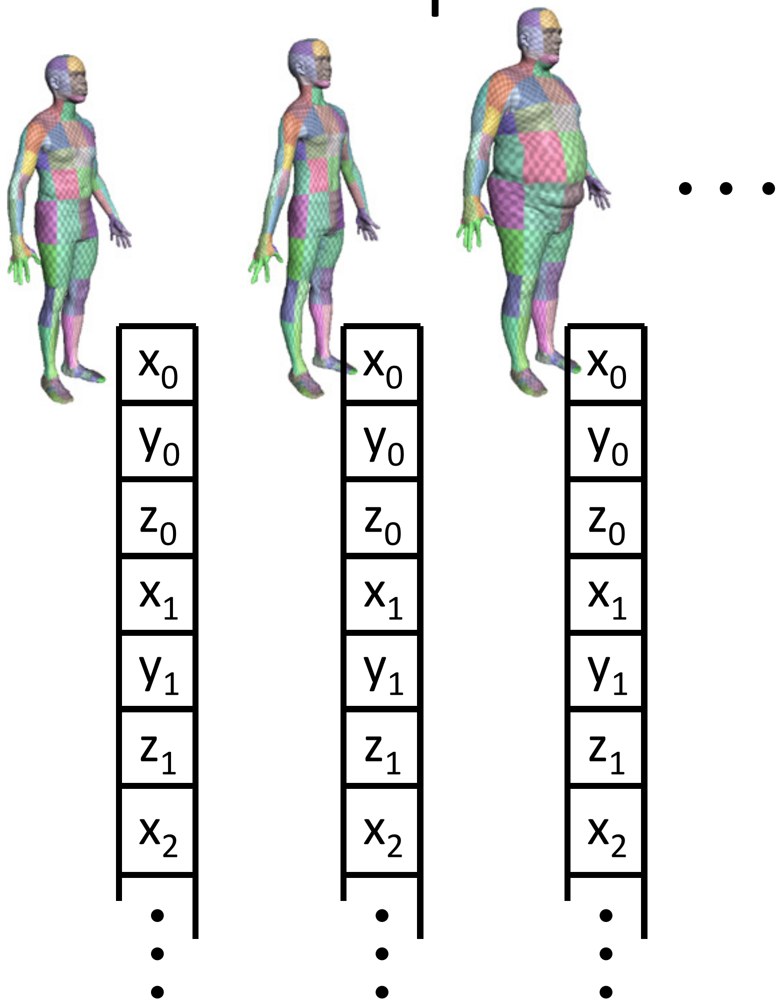


average male

mean + PCA component #1

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

Principal Component Analysis



average male

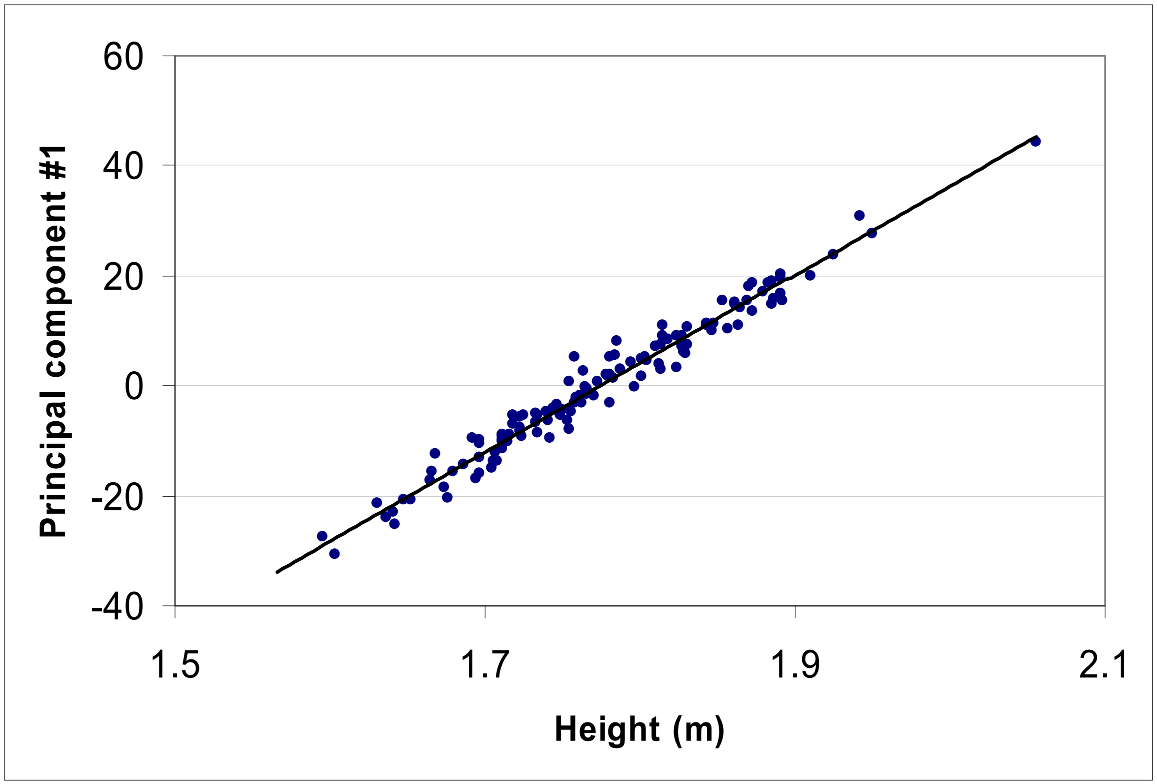
mean + PCA component #3

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

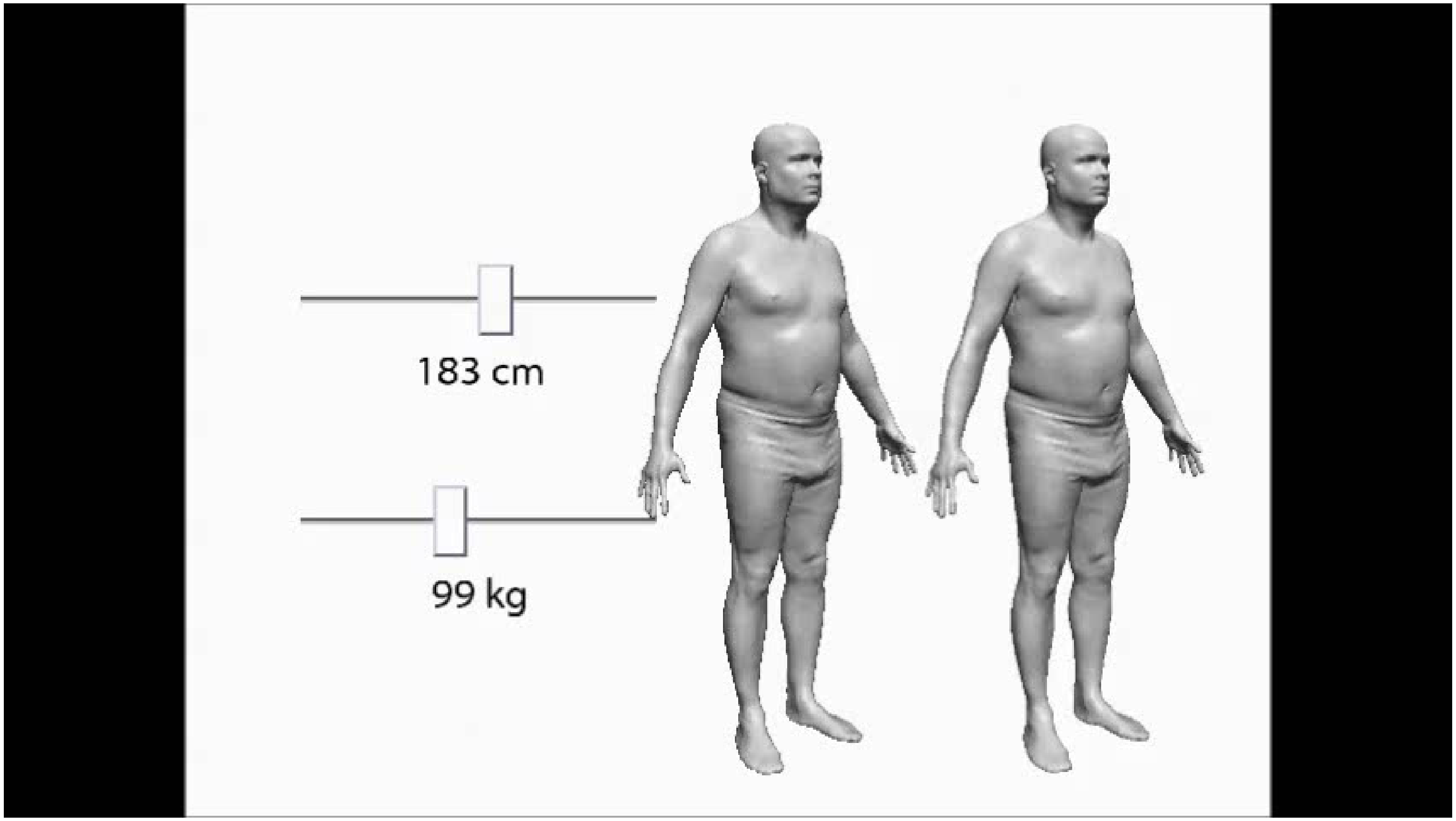
Fitting to attributes

Correlate PCA space with known attributes:



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003

Fitting to attributes



Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
to access the video: http://grail.cs.washington.edu/projects/digital-human/pub/allen04exploring.html

Case study: a probabilictic model for component-based synthesis

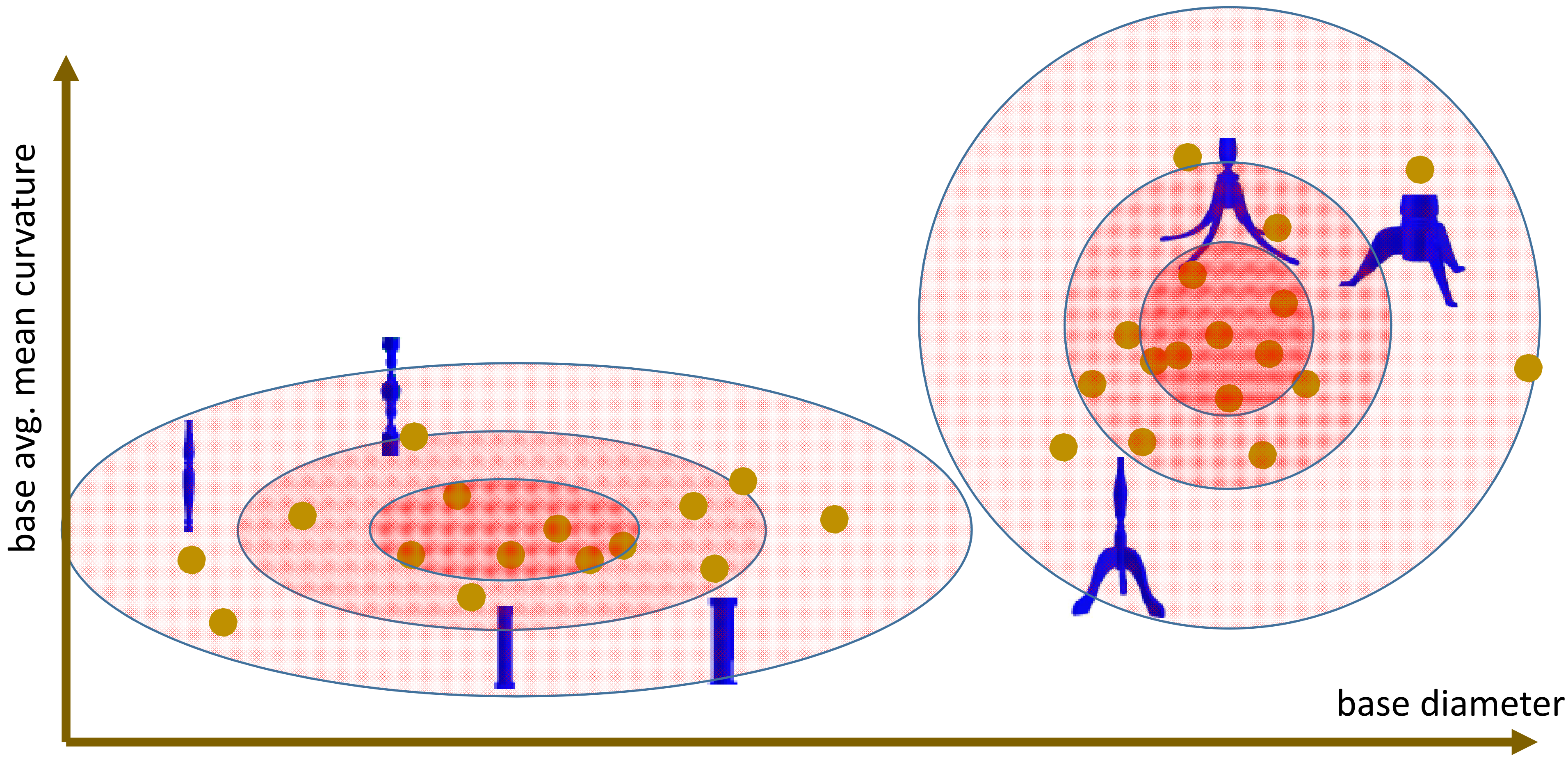
Given some training segmented shapes:

… and more ….

Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

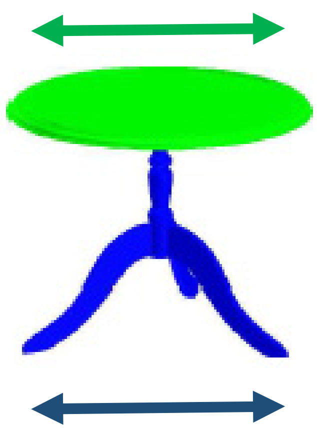
Describe shape space of parts with a probability distribution

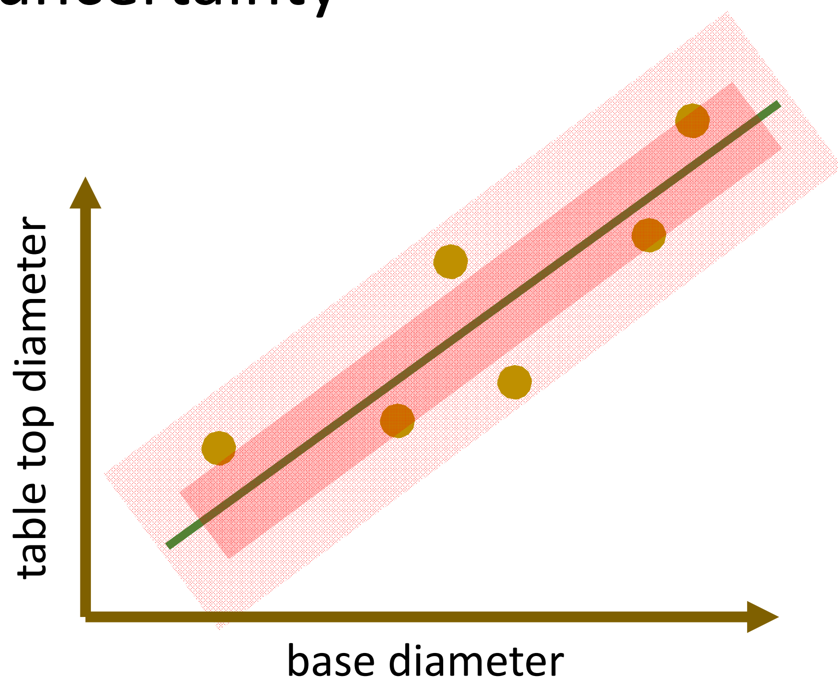


Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

Learn relationships between different part parameters within each cluster e.g. diameter of table top is related to scale of base plus some uncertainty

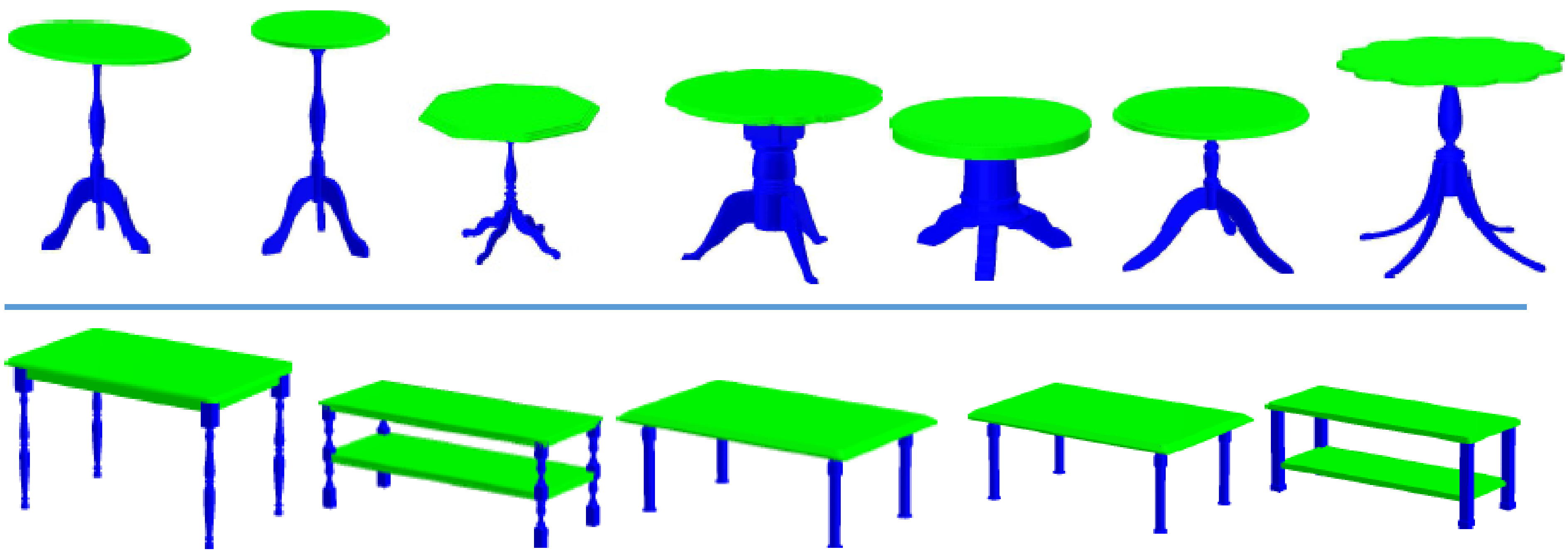




Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Case study: a probabilictic model for component-based synthesis

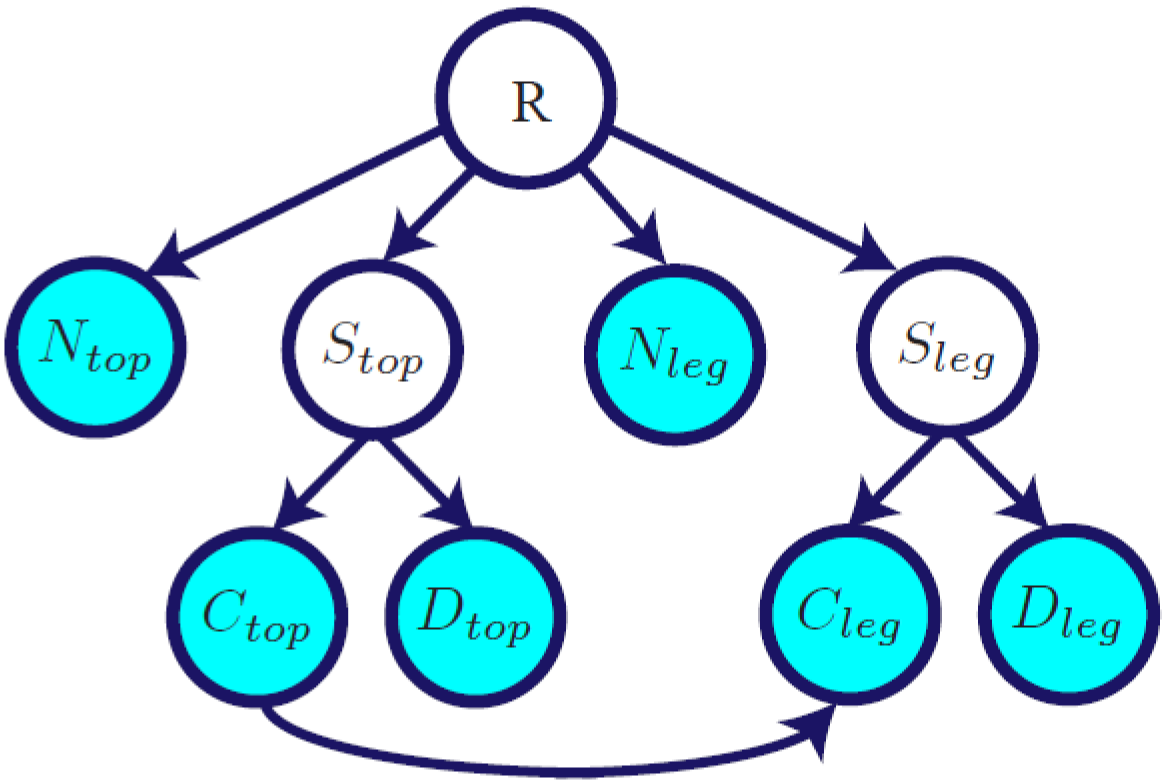
Learn relationships between part clusters e.g. circular table tops are associated with bases with split legs



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

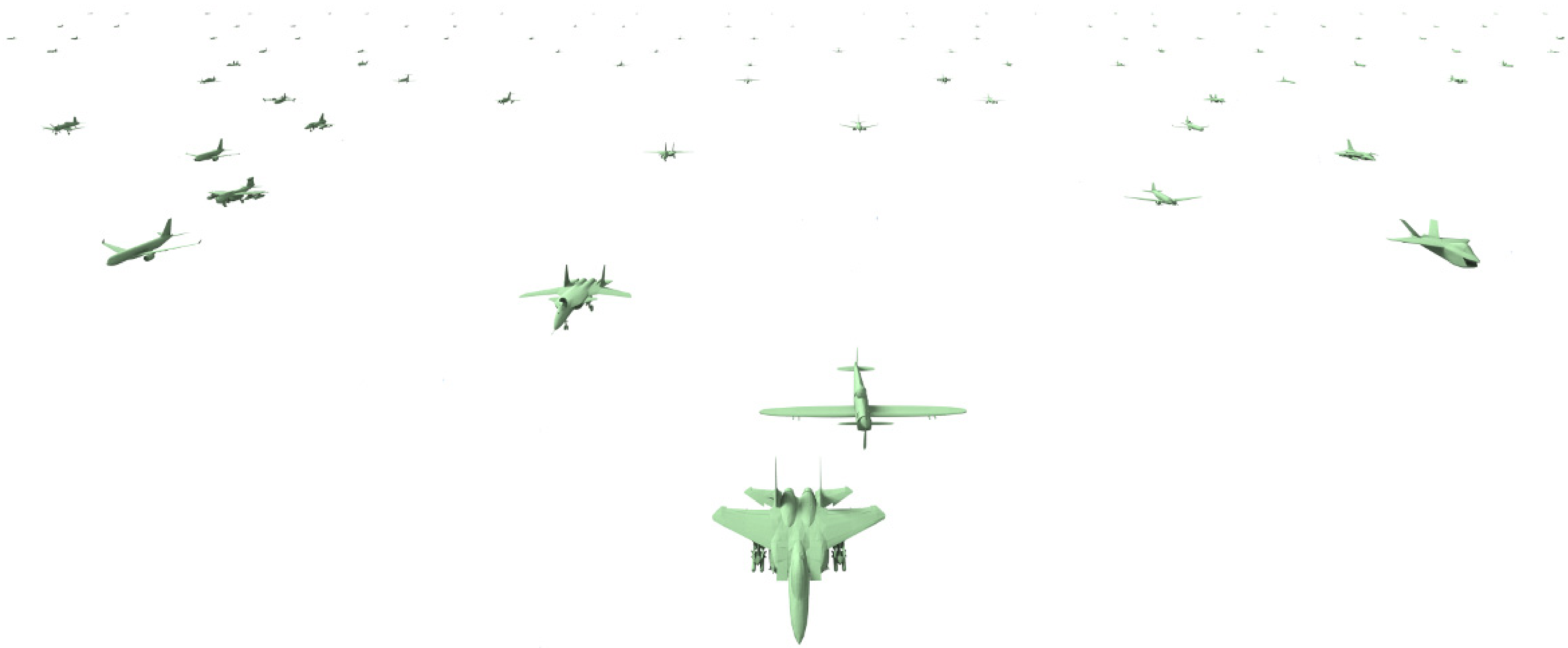
Case study: a probabilictic model for component-based synthesis

Represent all these relationships within a structured probability distribution (probabilistic graphical model)



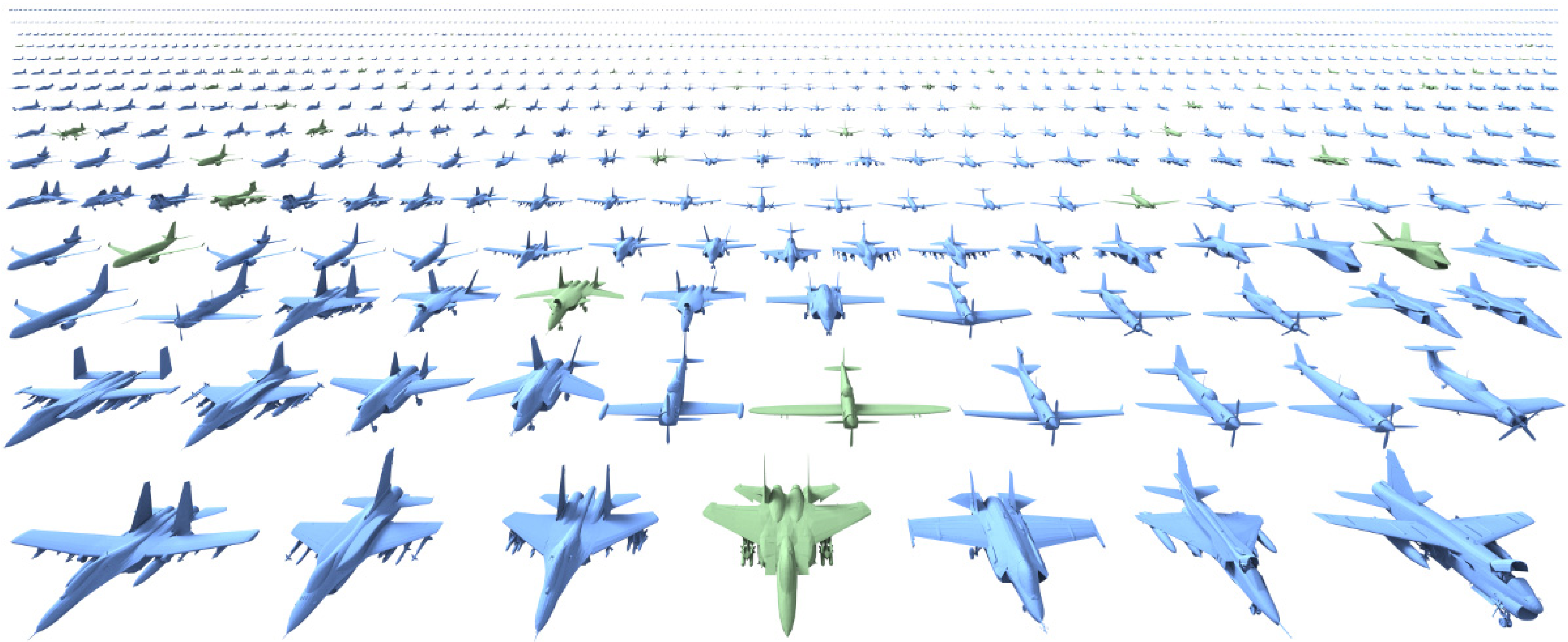
Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes

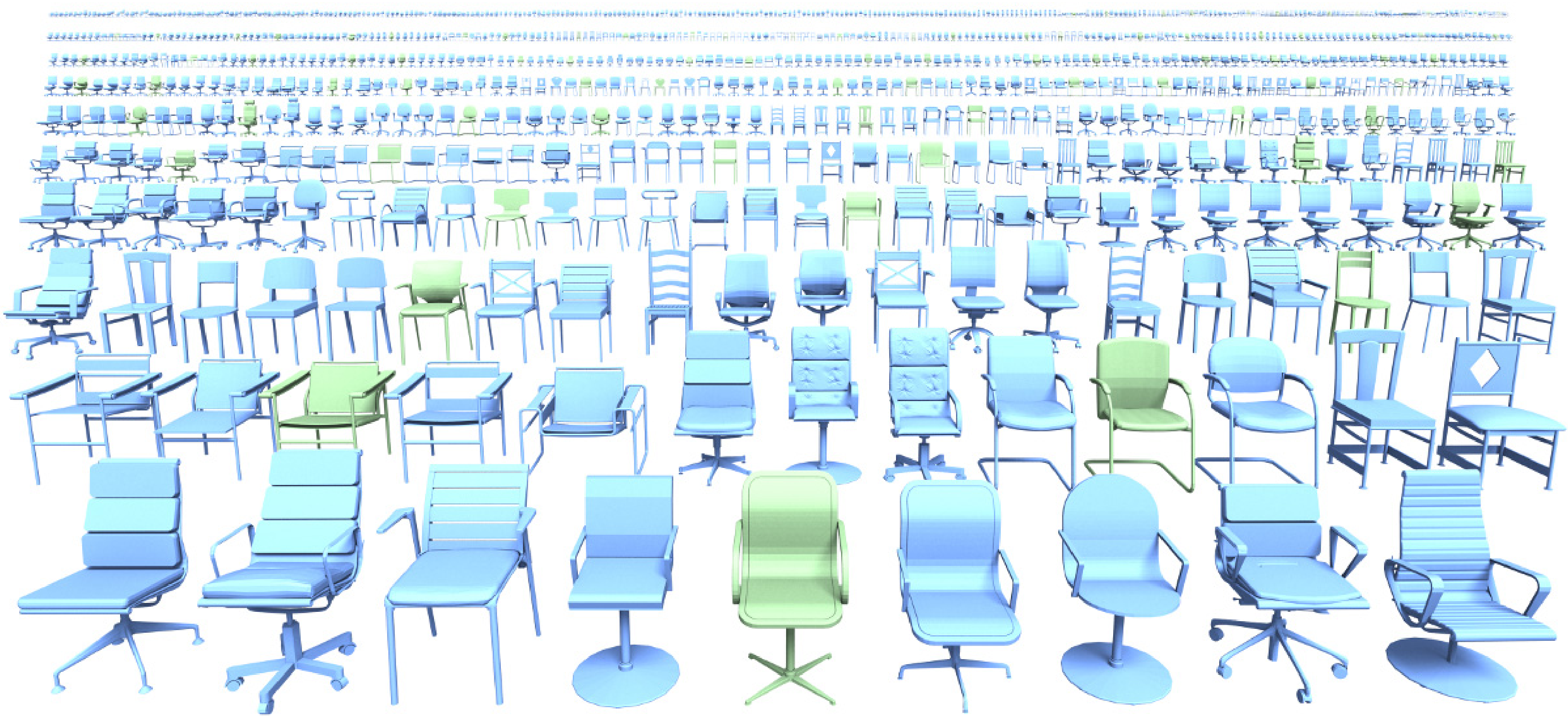


Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes

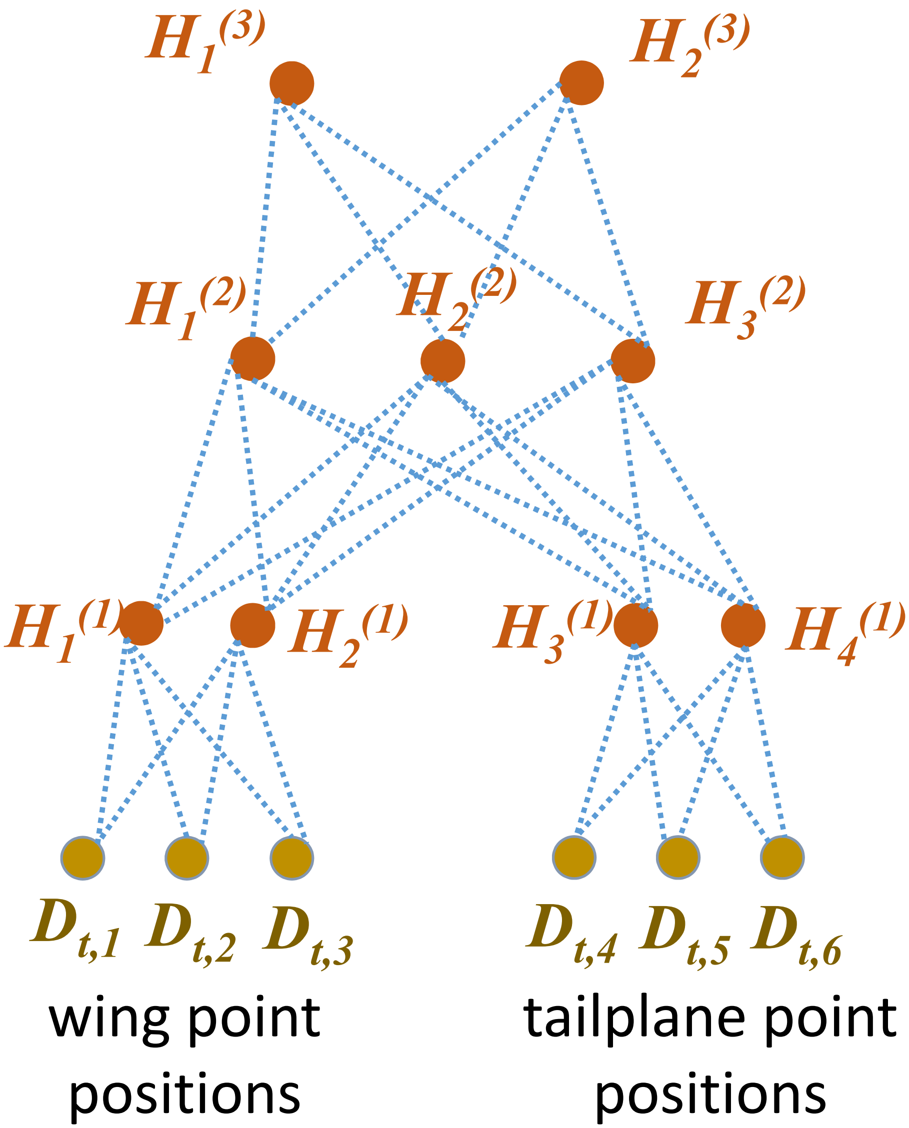
Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Shape Synthesis - Airplanes



Slides from Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, Vladlen Koltun
A Probabilistic Model for Component-Based Synthesis, 2012

Generative models of surface geometry



Slides from Haibin Huang, Evangelos Kalogerakis, Benjamin Marlinn
Analysis and Synthesis of 3D Shape Families via Deep-Learned Generative Models of Surfaces, 2015

Learning to Generate Chairs

Inverting the CNN…

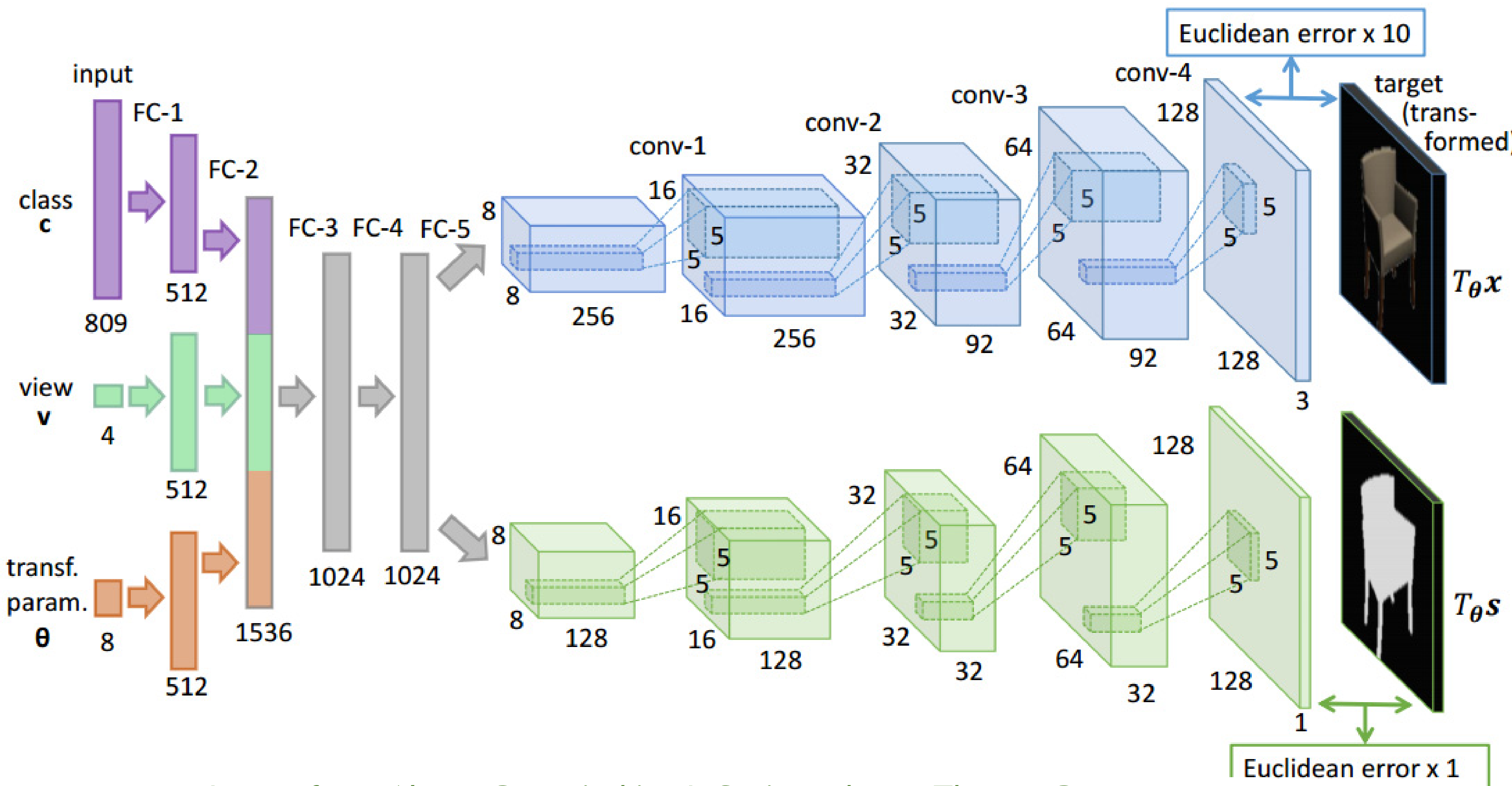


Image from Alexey Dosovitskiy, J. Springenberg, Thomas Brox
Learning to Generate Chairs with Convolutional Neural Networks 2015

Learning to Generate Chairs

Inverting the CNN…

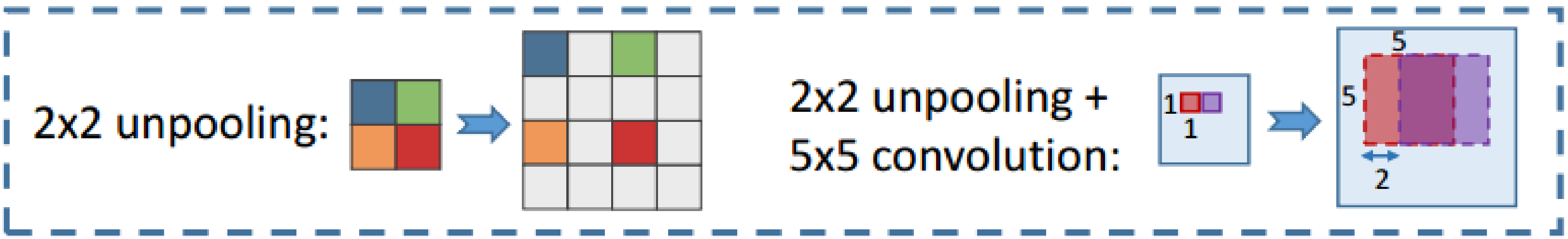


Image from Alexey Dosovitskiy, J. Springenberg, Thomas Brox
Learning to Generate Chairs with Convolutional Neural Networks 2015
to access video: http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15/

Summary

Welcome to the era where machines **learn to generate 3D visual content!**

Data-driven techniques with (deep) learning are highly promising directions

Summary

Welcome to the era where machines **learn to generate 3D visual content!**

Data-driven techniques with (deep) learning are highly promising directions

Some challenges:

- Generate **plausible, detailed, novel** 3D geometry from **high-level specifications, approximate** directions
- What **shape representation** should deep networks operate on?
- Integrate with approaches that optimize for **function, style** and **human-object interaction**