Machine Learning for Shape Analysis and Processing

Evangelos Kalogerakis
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

Animal, quadruped, horse, running horse

Input raw geometry

Head
Neck
Torso
Leg
Tail
Ear

Horse

part

Legs
Torso
Head

adjacent
adjacent
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

Data collection & preprocessing

Feature extraction

Feature learning
Pipeline of data-driven methods

Data collection & preprocessing

Feature extraction

Feature learning

Model learning
Pipeline of data-driven methods

- Data collection & preprocessing
- Feature extraction
- Feature learning
- Model learning

Learned model
Pipeline of data-driven methods

Data collection & preprocessing

Feature extraction

Feature learning

Model learning

Inference

Labeling

Matching

......
Pipeline of data-driven methods

Data collection & preprocessing

Feature extraction

Feature learning

Model learning

Inference

Labeling

Matching

……

Classification

Segmentation

Reconstruction

Synthesis

……
Pipeline of data-driven methods

Data collection & preprocessing

Feature extraction

Feature learning

Model learning

Learned model

Inference

Labeling

Matching

......

Classification

Segmentation

Reconstruction

Synthesis

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

Legend:
- Training data point (shape + design values)
Learning basics: regression

Leg thickness $y$

Sturdiness $x$

Training data point (shape + design values)
Learning basics: regression

Leg thickness $y$

sturdiness $x$

Training data point (shape + design values)
Learning basics: regression

- Leg thickness $y$
- Sturdiness $x$
- New datapoint
- Training data point (shape + design values)
Learning basics: regression

\[ y = w \cdot x' \]
\[ x' = [x^2 \ x \ 1] \]

- Leg thickness \( y \)
- Sturdiness \( x \)
- Training data point (shape + design values)
Learning basics: regression

$$y = w \cdot x'$$

$$x' = [x^2 \ x \ 1]$$

$$L(w) = \sum_{train. \ i} [y_i - w \cdot x_i']^2$$

Leg thickness $y$

Sturdiness $x$

Training data point (shape + design values)
Learning basics: regression

\[ y = w \cdot x' \]
\[ x' = [x^2 \ x \ 1] \]
\[ L(w) = \sum_{\text{train. } i} [y_i - w \cdot x_i']^2 \]

...linear least-squares solution...

- Leg thickness \( y \)
- Sturdiness \( x \)

New datapoint

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:** $y = 1$ [*coffee mug*], $y = 0$ [*no coffee mug*]

**Input:** $x = \{x_1, x_2, ...\}$ [*curvature histograms, HKS etc*]
Learning basics: Logistic Regression

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

Output: \( y = 1 \) [coffee mug], \( y = 0 \) [no coffee mug]

Input: \( x = \{x_1, x_2, \ldots \} \) [curvature histograms, HKS etc]

Classification function:

\[
P(y = 1 \mid x) = f(x) = \sigma(w \cdot x)
\]

where \( w \) is a weight vector

\[
\sigma(w \cdot x) = \frac{1}{1 + \exp(-w \cdot x)}
\]
Logistic regression: training

Need to estimate parameters $\mathbf{w}$ 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

$$\max_{\mathbf{w}} \prod_{i=1}^{N} P(\mathbf{y} = 1 | \mathbf{x}_i)_{[\mathbf{y}_i=1]} [1 - P(\mathbf{y} = 1 | \mathbf{x}_i)]_{[\mathbf{y}_i=0]}$$
Logistic regression: training

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

Find parameters that maximize probability of training data

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

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

Find parameters that maximize the log prob. of training data

$$\max_w \log \left\{ \prod_{i=1}^{N} \sigma(w \cdot x_i)^{[y_i=1]} \left[ 1 - \sigma(w \cdot x_i) \right]^{[y_i=0]} \right\}$$
Logistic regression: training

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

Find parameters that maximize the log prob. of training data

$$\max_{\mathbf{w}} \sum_{i=1}^{N} \left[ \mathbf{y}_i = 1 \right] \log \sigma(\mathbf{w} \cdot \mathbf{x}_i) + \left[ \mathbf{y}_i = 0 \right] \log(1 - \sigma(\mathbf{w} \cdot \mathbf{x}_i))$$
Logistic regression: training

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

This is called log-likelihood

$$
\max_w \sum_{i=1}^{N} [y_i = 1] \log \sigma(w \cdot x_i) + [y_i = 0] \log(1 - \sigma(w \cdot x_i))
$$
Logistic regression: training

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

We have an **optimization problem**.

$$
\max_w \sum_{i=1}^{N} [y_i = 1] \log \sigma(w \cdot x_i) + [y_i = 0] \log(1 - \sigma(w \cdot x_i))
$$

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

(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 $\eta$, update them according to:

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

See also quasi-Newton and IRLS methods
Regularization

Overfitting: few training data vs large number of parameters!

Penalize large weights:

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

Called ridge regression (or L2 regularization)
Overfitting:
few training data vs large number of parameters!

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

\[
\min_w - L(w) + \lambda \sum_d |w_d|
\]

Called **Lasso (or L1 regularization)**
The importance of choosing good features...

+ Coffee Mug
- Not Coffee Mug

modified slides originally by Adam Coates
The importance of choosing good features...

Is this a Coffee Mug?

Learning Algorithm

classification boundary

modified slides originally by Adam Coates
The importance of choosing good features...
From “shallow” to “deep” mappings

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

\[ y = f(x) = \sigma(w \cdot x) \]
Neural network

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

$$h_1 = \sigma(w^{(1)}_1 \cdot x)$$

$$h_2 = \sigma(w^{(1)}_2 \cdot x)$$

$$y = \sigma(w^{(2)} \cdot h)$$
Neural network

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

\[
\begin{align*}
X & \rightarrow \sigma (W^{(1)} \cdot x) \rightarrow h \rightarrow \sigma (W^{(2)} \cdot h) \rightarrow y \\
\end{align*}
\]

Intermediate representation  Prediction

\[
W^{(\cdot)} = \begin{bmatrix}
w_1^{(\cdot)} \\
w_2^{(\cdot)} \\
\vdots \\
w_m^{(\cdot)}
\end{bmatrix}
\]

*modified slides originally by Adam Coates*
Neural network

Stack up several layers:

```
x1  x2  x3  ...  x_d
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>h2</td>
<td>h3</td>
<td>...  hm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>h1'</td>
<td>h2'</td>
<td>h3'</td>
<td>h_n'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>y</td>
<td>h1</td>
<td>h2</td>
<td>h3</td>
</tr>
</tbody>
</table>
```

Diagram of a neural network with several layers.
Forward propagation

Process to compute output:

\[ x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_d \quad 1 \]
Forward propagation

Process to compute output:

\[ x \xrightarrow[\sigma(W^{(1)} \cdot x)]{} h \]
Forward propagation

Process to compute output:

\[ x \xrightarrow{\sigma(W^{(1)} \cdot x)} h \xrightarrow{\sigma(W^{(2)} \cdot h)} h' \]
Forward propagation

Process to compute output:

\[
\begin{align*}
X & \rightarrow \sigma(W^{(1)} \cdot x) \rightarrow h \rightarrow \sigma(W^{(2)} \cdot h) \rightarrow h' \rightarrow \sigma(W^{(3)} \cdot h') \rightarrow y
\end{align*}
\]
Multiple outputs

\[
\begin{align*}
\mathbf{x} &\xrightarrow{\sigma(\mathbf{W}^{(1)} \cdot \mathbf{x})} \mathbf{h} \xrightarrow{\sigma(\mathbf{W}^{(2)} \cdot \mathbf{h})} \mathbf{h}' \xrightarrow{\sigma(\mathbf{W}^{(3)} \cdot \mathbf{h}')} \mathbf{y}
\end{align*}
\]
How can we learn the parameters?

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

\[
L(w) = -\sum_{i} \sum_{\text{output } t} [y_{i,t} == 1] \log f_t(x_i) + [y_{i,t} == 0] \log(1 - f_t(x_i))
\]

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

\[
L(w) = \sum_{i} \sum_{\text{output } t} [y_{i,t} - f_t(x_i)]^2
\]
Backpropagation

For each training example $i$:

$$\delta_t^{(3)} = y_t - f(x)$$

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

For each output:

$$\delta_t^{(3)} = y_t - f(x)$$

$$\frac{\partial L(w)}{\partial w_{t,n}^{(3)}} = \delta_t^{(3)} h_n$$
$\delta^{(2)}_n = \sigma'(w^{(2)}_n \cdot h) \sum_{t} w^{(3)}_{t,n} \delta^{(3)}_t$

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

$\frac{\partial L(w)}{\partial w_{n,m}^{(2)}} = \delta^{(2)}_n h_m$
Backpropagation

\[ \delta_m^{(1)} = \sigma'(w_m^{(1)} \cdot x) \sum_n w_{n,m}^{(2)} \delta_n^{(2)} \]

\[ \frac{\partial L(w)}{\partial w_{m,d}^{(1)}} = \delta_m^{(1)} x_d \]
Is this magic?

All these are derivatives derived analytically using the chain rule!

Gradient descent is expressed through backpropagation of messages $\delta$ 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 $\delta$ from top to bottom layer
3. Multiply messages $\delta$ 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:

\[
tanh(\cdot) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]
\[
abs(\cdot) = \begin{cases} 
  x & \text{for } x \geq 0 \\
  -x & \text{for } x < 0 
\end{cases}
\]
\[
ReLu(\cdot) = \max\{0, \cdot\}
\]

\[
tanh'(\cdot) = 1 - \tanh(\cdot)^2
\]
\[
abs'(\cdot) = \text{sign}(\cdot)
\]
\[
ReLu'(\cdot) = \begin{cases} 
  1 & \text{for } \cdot > 0\\
  0 & \text{otherwise}
\end{cases}
\]

“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:

Before: \( \Delta w = \eta \nabla_w L(w), \quad w = w - \Delta w \)

With momentum: \( \Delta w = \mu \Delta w_{\text{previous}} + \eta \nabla_w L(w), \quad w = w - \Delta w \)

“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:

Before: \( \Delta w = \eta \nabla_w L(w), \ w = w - \Delta w \)

With momentum: \( \Delta w = \mu \Delta w_{\text{previous}} + \eta \nabla_w L(w), \ w = w - \Delta w \)

“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:

\[
\Delta w = \eta \nabla_w (L(w) + \lambda \| w \|_2)
\]
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 $g$: $h_{p,f} = g(w_f \cdot x_p)$

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:**
  \[ h_{p',f} = \max_p(x_p) \]

- **Mean pooling:**
  \[ h_{p',f} = \text{avg}_p(x_p) \]

- **Fixed filter (e.g., Gaussian):**
  \[ h_{p',f} = w_{\text{gaussian}} \cdot x_p \]

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

$\text{CNN}_1$: 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

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.

3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015*
Volumetric CNNs

Learned filters

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
So far...

“High-level” Space

“Low-level” Shape Space

“mediating” representations

chairs

desks

beds

So far...
Can we go the other way around?
YES! Automatic Shape modeling!

“High-level” Space

“Low-level” Shape Space

“mediating” representations

desks
chairs
beds
Why automatic geometric modeling?
Because it is not easy!
“Traditional” Geometric Modeling

- Manipulating polygons
  - image from autodesk.com

- Manipulating curves
  - image from autodesk.com

- Manipulating 3D primitives
  - image from wikipedia (CSG)

- 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?
“High-level” Space

“Low-level” Shape Space

Attributes

sturdy chair

modern chair

“mediating” representations

modern chair
Low-level Shape Space

Attributes

Sketches

“High-level” Space

“mediating” representations

lines, pixels

“Low-level” Shape Space
- Attributes
- Sketch
- Natural language
- Gestures
- Brain signals

... Learn it from data!

“High-level” Space

“Low-level” Shape Space

“mediating” representations
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
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 alignment. Can we always have dense correspondences?

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
Principal Component Analysis

\[ x_0 \quad y_0 \quad z_0 \]
\[ x_1 \quad y_1 \quad z_1 \]
\[ x_2 \quad \ldots \quad \ldots \]

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

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

Slides from Brett Allen, Brian Curless, Zoran Popović, Exploring the space of human body shapes, 2003
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
Case study: a probabilistic 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 probabilistic model for component-based synthesis

Describe shape space of parts with a probability distribution
Case study: a probabilistic 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 probabilistic 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 probabilistic 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

$H_1^{(3)}$, $H_2^{(3)}$

$H_1^{(2)}$, $H_2^{(2)}$

$H_1^{(1)}$, $H_2^{(1)}$

$D_{t,1}$, $D_{t,2}$, $D_{t,3}$

wing point positions

$D_{t,4}$, $D_{t,5}$, $D_{t,6}$

tailplane point positions

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
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**.