Simulation & Animation

Niloy Mitra
UCL/Adobe

Iasonas Kokkinos
UCL/Ariel AI

Paul Guerrero
UCL/Adobe

Vladimir Kim
Adobe

Nils Thuerey
TUM

Leonidas Guibas
Stanford/FAIR
# Timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
<th>Niloy</th>
<th>Iasonas</th>
<th>Paul</th>
<th>Nils</th>
<th>Leonidas</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>Introduction</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>~9:15</td>
<td>Neural Network Basics</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>~9:50</td>
<td>Supervised Learning in CG</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>~10:20</td>
<td>Unsupervised Learning in CG</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>~10:55</td>
<td>Learning on Unstructured Data</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>~11:35</td>
<td>Learning for Animation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>12:05</td>
<td>Discussion</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Computer Animation

- Feature detection (image features, point features)
- Denoising, Smoothing, etc.
- Embedding, Distance computation
- Rendering
- Animation
- Physical simulation

- Motion over time
- Lots of data - expensive...
- Relationships between spatial and temporal changes
Character Animation

• Target *character rigs*
• Natural *reactions and transitions*
• Reinforcement Learning
Physics-Based Animation

• Leverage *physical models*

• Examples:
  • Rigid bodies
  • Cloth
  • Deformable objects
  • Fluids
Character Animation
Existing Approaches

- Motion Representations
- Controllers
Learned Motion Manifolds

Data Preprocessing

1. Motion Capture and Processing

2. Phase Extraction

3. Terrain Fitting

Training

4. PFNN Training by Backpropagation

[Learning Motion Manifolds with Convolutional Autoencoders, SGA 2015 Tech. Briefs]
[Phase–functioned neural networks for character control, SIGGRAPH 2017]
Learned Motion Manifolds

[Phase-functioned neural networks for character control, SIGGRAPH 2017]
Reinforcement Learning

• Goal: maximize *reward* by performing *actions* in an *environment*
RL for Animation

• Learn Controllers that steer character rigs
• Smooth and natural transitions
• Reactions to changes in the environment

[Terrain–Adaptive Locomotion Skills Using Deep Reinforcement Learning, SIGGRAPH 2016]
[DeepMimic: Example–Guided Deep Reinforcement Learning of Physics–Based Character Skills, SIGGRAPH 2018]
We present a framework that, given a character, reference motion, and task...
Physics-Based Animation
Physics-Based Animation

Experiment
Observations / data

Theory
Model equations

Computation
Discrete representation

Skip Theory with Deep Learning?
[No! More on that later…]
Physics-Based Animation

- Better goal: **support solving** suitable physical models
- Nature = Partial Differential Equations (PDEs)
- Hence can aim for **solving PDEs with deep learning (DL)**
- Requirement: “**regularity**” of the targeted function

“Bypass the solving of evolution equations when these equations conceptually exist but are not available or known in closed form.” [Kevrekidis et al.]
Partial Differential Equations

• Typical problem formulation: unknown function \( u(x_1, \ldots, x_n) \)

• PDE of the general form:

\[
f(\begin{pmatrix} x_1, \ldots, x_n \end{pmatrix}, \begin{pmatrix} \frac{\partial u}{\partial x_1} & \ldots & \frac{\partial u}{\partial x_n} & \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 \partial x_2} & \ldots \end{pmatrix}) = 0
\]

• Solve in domain \( \Omega \), with boundary conditions on boundary \( \Gamma \)

• Traditionally: discretize & solve numerically. Here: also discretize, but solve with DL...
Methodology 1

• Viewpoints: holistic or partial
  [partial also meaning “coarse graining” or “sub-grid / up-res”]
• Influences complexity and non-linearity of solution space
• Trade off computation vs accuracy:
  • Target most costly parts of solving
  • Often at the expense of accuracy
Methodology 2

• Consider dimensionality & structure of discretization
  • Small & unstructured
    • Fully connected NNs only choice
    • Only if necessary...
  • Large & structured
    • Employ convolutional NNs
    • Usually well suited
Solving PDEs with DL

• Practical example: airfoil flow
  • Given boundary conditions solve stationary flow problem on grid
  • Fully replace traditional solver
  • 2D data, no time dimension
  • I.e., holistic approach with structured data
Solving PDEs with DL

• Data generation
• Large number of pairs: input (BCs) - targets (solutions)
Solving PDEs with DL

- Data generation
- Example pair
- Note - boundary conditions (i.e. input fields) are typically constant
- Rasterized airfoil shape present in all three input fields
Solving PDEs with DL

• U-net NN architecture

Reduce spatial dimensions

Increase spatial dimensions

Skip connections
Solving PDEs with DL

• U-net NN architecture

• Unet structure highly suitable for PDE solving
• Makes boundary condition information available throughout
• Crucial for inference of solution
Solving PDEs with DL

- **Training**: 80,000 iterations with ADAM optimizer
- Convolutions with enough data - no dropout necessary
- Learning rate decay stabilizes models
Results

- Use knowledge about physics to simplify space of solutions: make quantities dimension-less
- Significant gains in inference accuracy
Solving PDEs with DL

• Validation and test accuracy for different model sizes
Code example

Solving PDEs with DL
Existing Approaches

- Elasticity
- Cloth
- Fluids
Neural Material - Elasticity

• Learn correction of regular FEM simulation for complex materials
• Numerical simulation with flexible NN for material behavior

[NNWarp: Neural Network-based Nonlinear Deformation, TVCG 2018]
[Neural Material: Learning Elastic Constitutive Material and Damping Models from Sparse Data, arXiv 2018]
Neural Material - Elasticity

• Learn correction of regular FEM simulation for complex materials

NeoHookean Training

GT: NeoHookean, $E = 2e4$  
Nominal: Co-rotational, $E = 3.5e4$

[Neural Material: Learning Elastic Constitutive Material and Damping Models from Sparse Data, arXiv 2018]
Latent Spaces

- Learn flexible reduced representation for physics problems

[Deep Fluids: A Generative Network for Parameterized Fluid Simulations, EG 2019]
Latent Spaces

• Learn flexible reduced representation for physics problems
  • Employ Encoder part (E) of Autoencoder network to reduce dimensions
  • Predict future state in latent space with FC network
  • Use Decoder (D) of Autoencoder to retrieve volume data

[Deep Fluids: A Generative Network for Parameterized Fluid Simulations, EG 2019]
Latent Spaces

- Learn flexible reduced representation for physics problems

[Deep Fluids: A Generative Network for Parameterized Fluid Simulations, EG 2019]
Latent Spaces • In combination with Reinforcement Learning

Cooperative ball game

Liquid jet 1

Liquid jet 2

[Fluid Directed Rigid Body Control using Deep Reinforcement Learning, SIGGRAPH 2018]
Latent Spaces

• For elasticity problems
Latent Spaces

• For elasticity problems

Full-space Comparison

- PCA Only
- Autoencoder (ours)

[Latent-space Dynamics for Reduced Deformable Simulation, EG 2019]
Latent Spaces

• For cloth (adaptation to different body shapes)

[Learning-Based Animation of Clothing for Virtual Try-On, EG 2019]
Temporal Data

• Generative model for 3D plus time
• Input domain: low resolution 3D volumes
• Output: high-resolution 3D volumes
• Auxiliary goal: match temporal evolution of target domain (high-res. data)
Temporal Data

\[ \mathbf{x}_a \rightarrow \mathbb{G} \]

Temporal Data

\[ \begin{align*}
X_a & \quad \rightarrow \quad G \quad \rightarrow \quad D_s \quad \rightarrow \\
X_{t-1} & \quad \rightarrow \quad G \quad \rightarrow \quad D_t \quad \rightarrow \\
X_t & \quad \rightarrow \quad \quad \quad \rightarrow \quad D_t \\
X_{t+1} & \quad \rightarrow \quad \quad \quad \rightarrow \\
\end{align*} \]

\[ \begin{align*}
y_{a,t-1} & \quad \rightarrow \quad y_t \quad \rightarrow \quad y_{a,t+1} \\
\end{align*} \]

**Temporal Data**

Discretized advection operator included in loss for $G$
Temporal Data

\[ y_a = G(x_a) \]

\[ y_{t-1} = G(x_{t-1}) \]

\[ y_t = G(x_t) \]

\[ y_{t+1} = G(x_{t+1}) \]
Temporal Data

Summary

• Checklist for solving PDEs with DL:

  ✓ Model? (Typically given)
  ✓ Data? Can enough training data be generated?
  ✓ Which NN Architecture?
  ✓ Fine tuning: learning rate, number of layers & features?
  ✓ Hyper-parameters, activation functions etc.?
Summary

• Approach PDE solving with DL like solving with traditional numerical methods:
  - Find closest example in literature
  - Reproduce & test
  - Then vary, adjust, refine …
• Main change: Data pipeline
Deep Learning - Outlook

• DL provides a powerful computational tool

• Open challenges:
  - Theoretical guarantees
  - Ethical questions
  - “Next level” of representation learning
The End - Thank you!

Course Information (slides/code/comments)
http://geometry.cs.ucl.ac.uk/creativeai/