Neural Rendering

D.A. Forsyth
Neural vs Differentiable Rendering

- Differentiable rendering
  - make (relatively conventional) renderer differentiable
  - usually to support inference (shape from single image, etc.)
- Neural rendering
  - use neural networks at various points in the rendering process
  - lots of methods
    - no real consensus on what a neural rendering process looks like
Some topics...

- Reduce rendering noise
  - in MCMC rendering
  - in image based rendering
  - in performance capture
- Realistic images from approximations
- Generate novel views
  - from multiview input
- Exaggerate effects
  - eg motion fields
- Reshade and relight
Reducing noise: MCMC rendering

- **Issue:**
  - physically accurate rendering requires tracing very large numbers of complex paths; the resulting estimates can have quite high noise
  - Reducing noise by tracing “more paths” is impractical (1/\sqrt{N})

Filter noisy pixels:

\[
\hat{c}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{c}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}},
\]

Kalantari et al 15
Cross-bilateral filter

\[ \hat{c}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{c}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}}, \]

\[ d_{i,j} = \exp \left[ - \frac{\| \bar{p}_i - \bar{p}_j \|^2}{2\alpha_i^2} \right] \times \exp \left[ - \frac{D(\bar{c}_i, \bar{c}_j)}{2\beta_i^2} \right] \]

\[ \times \prod_{k=1}^{K} \exp \left[ - \frac{D_k(\bar{f}_{i,k}, \bar{f}_{j,k})}{2\gamma_{k,i}^2} \right], \]

Kalantari et al 15

Location

Pixel color

Features (eg. which surface, normal, etc.)
Natural attack

Figure 4: Our approach combines a standard MLP (Fig. 3) with a matching filter. The local mean primary features (illustrated by a stack of images) contain color, position, and additional features such as world positions, shading normals, etc. A set of secondary features \( \{x_1, \cdots, x_N\}_i \) (see Sec. 3.3) are extracted from the mean primary features in a local neighborhood of each pixel. The MLP takes the secondary features and outputs the parameters of the filter. The filter then takes the block of mean primary features and outputs a filtered pixel. During training, we minimize the error between the filtered pixel and the ground truth. Once trained, the network can generate appropriate filter parameters for an arbitrary test image.
Figure 2: Comparison between our approach and several state-of-the-art algorithms on the KITCHEN scene rendered at 4 spp. Note that the ground truth image is still noisy even at 32K spp. Non-local means filtering (NLM) [Roussel et al. 2012] is a color-based method which cannot keep geometry or texture detail. Random parameter filtering (RPF) [Sen and Darabi 2012], SURE-based filtering (SBF) [Li et al. 2012], robust denoising (RD) [Roussel et al. 2013], and weighted local regression (WLR) [Moon et al. 2014] use additional scene features (e.g., world positions) to keep the details. However, they often do not weight the importance of each feature optimally, resulting in under/over blurred regions or splotches in the final result. Our approach preserves scene detail and generates a higher-quality, noise-free result faster than most other methods. The relative mean squared error (ReIMSE) and structural similarity (SSIM) index are listed below each image. Larger SSIM values indicate better perceptual quality. Full images are available in the supplementary materials. Scene credit: Jo Ann Elliott.
Spikes...

Figure 5: The image on the left shows the result of our approach before spike removal on an inset of the KITCHEN scene. In our method, we remove high magnitude spikes in the filtered image as a post-process to produce the result shown in the middle. The ground truth image is shown on the right for comparison.
See also

Alla Chaitanya, 17 (same problem, different architecture)

<table>
<thead>
<tr>
<th>Our reconstruction result</th>
<th>MC Input</th>
<th>AAF</th>
<th>EAW</th>
<th>SBF</th>
<th>Our</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grids</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>Pillars</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Fig. 10. Closeups for shadow filtering for 1 spp input (MC), axis-aligned filter (AAF), À-Trous wavelet filter (EAW), SURE-based filter (SBF), and our result.
Some topics...

- Reduce rendering noise
  - in MCMC rendering
  - in image based rendering
  - in performance capture
- Realistic images from approximations
- Generate novel views
  - from multiview input
- Exaggerate effects
  - eg motion fields
- Reshade and relight
Noise management in IBR

• (You could see NeRF as an extreme case of this)
• Image based rendering
  • From several images of a scene, produce a rendering at new viewpoint
    • Typically, using some form of approximate geometric representation
  • Simplest cases
    • SFM yields cameras, blend on a common plane (Phototourism, Snavely et al 06)
      • https://www.youtube.com/watch?v=mTBPGuPLI5Y
      • blend can look poor, texture slides
    • SFM yields points->parametric model, texture from image, render (Facade, Debevec 1996, 1997)
      • many things remain hard to model
      • errors in recovered model lead to texture problems
The novel view is a blend; blend is driven by relief from reconstruction, normals, etc. Strategy: build the best blender.
On-line Deep Blending Pipeline

Novel Viewpoint

Voxel Grid

InsideOut Tiled N-View Selection

Per-image Depth Mesh Rendering

Warped Views

Per-pixel View Prioritization

4 View Mosaics

Global Mesh Render + 4 Mosaics

Blend Weights

Blended RGB Output

Hedman, 18
Fig. 10. Our network takes as input ranked mosaics generated from a set of warped candidate views. For each pixel, the candidates are ranked based on their expected blending contribution, and 4 color-image mosaics are formed from the top 4 rankings. Example mosaics are shown in the first two rows. The top right halves show the color mosaic, while the bottom left halves show colormaps of the selection, with each input shown in a different color. Weighted blending outputs from our network (bottom right) are trained by minimizing their difference with real images (bottom left). Our network also blends an RGB view of the global mesh (not shown).
Off-line CNN Training

Pool of Original Input Views (Multiple Scenes)

Left-out Image

Other Views

Deep Blending

Predicted Blended RGB Output

Training Loss

- Perceptual Differences
- Temporal Consistency
Training

• Losses:
  • per frame perceptual loss

\[ \mathcal{L}(I_N, I_R) = |I_N - I_R| + \\
  |\text{VGG16}_{\text{relu12}}(I_N) - \text{VGG16}_{\text{relu12}}(I_R)| + \\
  |\text{VGG16}_{\text{relu22}}(I_N) - \text{VGG16}_{\text{relu22}}(I_R)| \]

• two frame temporal consistency
  • helps prevent oscillation, flicker, etc

\[ \mathcal{L}_T(I_N, I_R) = \mathcal{L}(I^t_N, I_R) + 0.33 \times \mathcal{L}(I^{t-1}_N, W_f(I^t_N)), \]
Notes and Queries

- This mostly cleans up a very good IBR representation
  - notice how much preprocessing and detail before learning
- You should likely think of IBR repn as latent variables
  - Q: can one learn them? Why?
- There is no adversarial loss
  - Q: Why? (authors say might create temporal coherence problems)
View dependent appearance effects

• Specular effects, gloss, etc. depend on viewing direction
  • Blending multiple views will blur the effect or remove it
    • Strategy:
      • select triangle from image mesh per view (Debevec, 98) rather than blending
View dependent appearance effects

- Specular effects, gloss, etc. depend on viewing direction
  - Blending multiple views will blur the effect or remove it

Figure 14: Image synthesis on real data: we show a comparison to the IBR technique of Debevec et al. (1998). From left to right: reconstructed geometry of the object, result of IBR, our result, and the ground truth.
Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our EffectsNet architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected into the target image space. In the target image space we estimate the new view-dependent effect and add them to the warped images. An encoder-decoder network is used to blend the warped images to obtain the final output image. During training, we enforce that the output image matches the corresponding ground truth image.
We propose a learning-based image-guided rendering approach that enables novel view synthesis for arbitrary objects. Input to our approach is a set of $N$ images $\mathcal{I} = \{\mathcal{I}_k\}_{k=1}^N$ of an object with constant illumination. In a preprocess, we obtain camera pose estimates and a coarse proxy geometry using the COLMAP structure-from-motion approach (Schönberger & Frahm (2016); Schönberger et al. (2016)). We use the reconstruction and the camera poses to render synthetic depth maps $\mathcal{D}_k$ for all input images $\mathcal{I}_k$ to obtain the training corpus $\mathcal{T} = \{(\mathcal{I}_k, \mathcal{D}_k)\}_{k=1}^N$, see Fig. 8. Based on this input, our learning-based approach generates novel views based on the stages that are depicted in Fig. 1. First, we employ a coverage-based look-up to select a small number $n \ll N$ of fixed views from a subset of the training corpus. In our experiments, we are using a number of $n = 20$ frames, which we call reference images. Per target view, we select $K = 4$ nearest views from these reference images. Our EffectsNet predicts the view-dependent effects for these views and, thus, the corresponding view-independent components can be obtained via subtraction (Sec. 5). The view-independent component is explicitly warped to the target view using geometry-guided cross-projection (Sec. 6). Next, the view-dependent effects of the target view are predicted and added on top of the warped views. Finally, our CompositionNet is used to optimally combine all warped views to generate the final output (Sec. 6). In the following, we discuss details, show how our approach can be trained based on our training corpus (Sec. 4), and extensively evaluate our proposed approach (see Sec. 7 and the appendix).
Figure 2: *EffectsNet* is trained in a self-supervised fashion. In a Siamese scheme, two random images from the training set are chosen and fed into the network to predict the view-dependent effects based on the current view and the respective depth map. After re-projecting the source image to the target image space we compute the diffuse color via subtraction. We optimize the network by minimizing the difference between the two diffuse images in the valid region.

Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected into the target image space. In the target image space we estimate the new view-dependent effect and add them to the warped images. An encoder-decoder network is used to blend the warped images to obtain the final output image. During training, we enforce that the output image matches the corresponding ground truth image.

Thies et al 20
Figure 4: Prediction and removal of view-dependent effects of a highly specular real object.
Figure 6: Comparison to the IBR method InsideOut of Hedman et al. (2016) and the learned IBR blending method DeepBlending of Hedman et al. (2018). To better show the difference in shading, we computed the quotient of the resulting image and the ground truth. A perfect reconstruction would result in a quotient of 1. As can be seen our approach leads to a more uniform error, while the methods of Hedman et al. show shading errors due to the view-dependent effects.

Thies et al 20
Idea: predict these separately

Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our EffectsNet architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected into the target image space. In the target image space we estimate the new view-dependent effect and add them to the warped images. An encoder-decoder network is used to blend the warped images to obtain the final output image. During training, we enforce that the output image matches the corresponding ground truth image.
Notes and Queries

• Key idea
  • separate diffuse view prediction and view dependent components
  • notice how much preprocessing and detail before learning
    • multiple registered pix and depth maps

• There is an adversarial loss
  • Local PatchGAN loss
    • from pix2pix (Isola, 16)
    • useful trick
Performance capture (rough summary)

- Use multiple synchronized cameras to
  - come up with a surface like representation of performer(s)
    - that is photorealistic
    - to re-render from different views
    - to augment
- History
  - rough outlines clear since mid 90s
  - details fantastically important
  - quality is hard to get
Performance capture (rough summary)

Kanade et al 97

View

Depth map (stereo, I think)
Performance capture (rough summary)

Depth discontinuities create meshing problems  
Crop at discontinuities  
Fill holes with other viewpoints

Kanade et al 97
Figure 8. A baseball bat swing from the baseball’s point of view.
Quiz: what could go wrong
Quiz: what could go wrong?

- Flicker at boundaries
  - segmentation not coherent over time
- Segmentation errors lead to poor appearance
- Motion blur errors
- Matting errors
- Resolution problems
- Texture at boundaries
Performance capture (rough summary)

Depth discontinuities create meshing problems

Crop at discontinuities

Fill holes with other viewpoints

Kanade et al 97
Figure 8. A baseball bat swing from the baseball’s point of view.
Fixes

- **Cameras:**
  - more, faster, higher resolution, better synchronized
- **Reconstruction algorithms:**
  - high resolution multiview stereo reconstructions
- **Body models:**
  - skinned parametric body/hand/face models
Figure 2: Part models and the Frank model. (a) The body model [34]; (b) the face model [15]; and (c) a hand rig. In (a-c), the red dots have corresponding 3D keypoints reconstructed by detectors; (d) Body only model; (e) Face and hand models substitute the corresponding parts of the body model. Alignments are ensured by $\Gamma$s; and (f) The blending matrix $C$ is applied to produce a seamless mesh.
Issues (later…)

- Construct parametric surface deformation model from data
  - for body, hand, head+face (body - SMPL, widely used)
- Skinning
  - Link joint parameters of model to surface for control
- Blending
  - Attach hand, head+face to body
Fitting

• Recover point cloud
• Recover 3D joint (keypoint) positions
  • human pose recovery (qv)
• Fit parametric model to point cloud using
  • keypoint positions
  • ICP for points to surface
  • Minimize seams between hand/body, head/body
  • prior
• Refine parametric model to better encode sequences
Relightables - extreme capture

- Capture with
  - 12MP active IR depth sensors (specialized)
  - Fast HR RGB cameras
  - Controllable relighting during capture

Guo et al, 2019
Guo et al 19

Depth maps come both from active and from passive sensors. Essential: can’t green-screen because we’re actively relighting; CRF here leads to other small but important improvements. Standard procedure.

Fig. 8. The Relightables Pipeline (Part 1). First, raw images are used to reconstructed a high quality 3D model.
Simplified by another standard procedure

Mesh $\rightarrow$ Tracked Mesh + UV Atlas

Gives texture coordinates for each triangle in mesh

Fig. 9. The Relightables Pipeline (Part 2). This mesh then gets downsampled, tracked over time and parameterized.
Because we know triangle normals, and we see under multiple illuminations, can recover (a) albedo and (b) gloss terms.

Q: can we also refine normals, triangles, etc?

Guo et al 19
• General point
  • for rendering purposes, normals do not need to be geometric
    • ie the normal at each mesh vertex
    • does not have to be estimated from the mesh
    • could be estimated photometrically (essentially, photometric stereo)
  • photometric normals are often (usually) better

Guo et al 19
Fig. 20. Left: HDRI relighting of diffuse color and geometry such as in Collet et al. [2015]. Right: our solution using geometry, albedo, photometric normal, and material maps as input. Note the increased sharpness and amount of details with the proposed system.
There are problems in all systems...

Fig. 2. Limitations of state of the art, real-time performance capture systems. Left: low resolution textures where the final rendering does not resemble a high quality picture of the subject. Middle: coarse geometry leads to overly smooth surfaces where important details such as glasses are lost. This also limits the quality of the final texture. Right: incomplete data in the reconstruction creates holes in the final output.

Martin-Brualla et al, 18
Idea: learned beauty-renderer

- During capture, have witness cameras
- Train a beauty renderer to
  - accept predicted frames
  - produce good looking frames
  - using witness cameras

Fig. 5. LookinGood's fully convolutional deep architecture. We train the model for both left and right view that simulate a VR or AR headset. The architecture takes as input a low resolution image and produces a high quality rendering together with a foreground segmentation mask.

Martin-Brualla et al, 18
Fig. 7. Generalization on new sequences. We show here some results on known participant but unseen sequences. Notice how the method is able to in-paint missing areas correctly in the single camera case (top rows). Full body results show an improved quality and robustness to imprecision in the groundtruth mask (third row, right). The method also recovers from color and geometry inconsistencies (forth row, left).
Fig. 8. Viewpoint Robustness. Notice how the neural re-rendering generalizes well w.r.t. to viewpoint changes, despite no training data was acquired for those particular views.
Notes and queries

• What are the losses?
  • natural
  • in paper - look them up!
  • adversarial loss is part of this

• General points:
  • Beauty renderers are probably an excellent idea
  • Q: conditioning to get best balance between quality/efficiency?
    • A:?
  • Q: should this be a general part of any future “realistic” rendering system?
    • i.e. learned beauty renderer from rough to final
    • A: likely yes, only issue is pragmatics
Some topics…

• Reduce rendering noise
  • in MCMC rendering
  • in image based rendering
  • in performance capture - TBA!
• Realistic images from approximations
• Generate novel views
  • from multiview input
• Exaggerate effects
  • eg motion fields
• Reshade and relight
Some topics…

• Reduce rendering noise
  • in MCMC rendering
  • in image based rendering
  • in performance capture
• **Realistic images from approximations**
  • texture synthesis history
• Generate novel views
  • from multiview input
• Exaggerate effects
  • eg motion fields
• Reshade and relight
Texture

CS 419
Slides by Ali Farhadi
Texture scandals!!
Bush campaign digitally altered TV ad

President Bush’s campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.

This section shows a sampling of the duplication of soldiers.

Original photograph
Two crucial algorithmic points

• Nearest neighbors
  • again and again and again

• Dynamic programming
  • likely new; we’ll use this again, too
Texture Synthesis

Efros & Leung ICCV99
How to paint this pixel?

Input texture

Efros & Leung ICCV99
Neighborhood size

Efros & Leung ICCV99
Varying Window Size

Increasing window size

Efros & Leung ICCV99
More Results
Extrapolation

Efros & Leung ICCV99
Input texture

Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut

Efros & Freeman SIGGRAPH01
Minimal error boundary

overlapping blocks

vertical boundary

overlap error

min. error boundary

Efros & Freeman SIGGRAPH01
Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut

Efros & Freeman SIGGRAPH01
More Results
More Results
Texture Transfer

- Take the texture from one object and paint it on another object

Decomposing shape and texture
- Very challenging
- Walk around
- Add some constraint to the search

Efros & Freeman SIGGRAPH01
Texture Transfer

Efros & Freeman SIGGRAPH01
Image Analogies
Training

Unfiltered source (A)

Filtered source (A')
Learn to Blur

Unfiltered source ($A$)  Filtered source ($A'$)

Unfiltered target ($B$)  Filtered target ($B'$)

Hertzman. Jacobs. Oliver. Curless. and Salesin. SIGGRAPH01
Texture by Numbers

Unfiltered source (A)

Filtered source (A’)

Unfiltered (B)

Filtered (B’)

Hertzman. Jacobs. Oliver. Curless. and Salesin. SIGGRAPH01
Colorization

Unfiltered source ($A$)  Filtered source ($A'$)

Unfiltered target ($B$)  Filtered target ($B'$)
Super-resolution

A

A'

Hertzman. Jacobs. Oliver. Curless. and Salesin. SIGGRAPH01
Super-resolution (result!)

Hertzman. Jacobs. Oliver. Curless. and Salesin. SIGGRAPH01
Training images

Hertzman. Jacobs. Oliver. Curless. and Salesin. SIGGRAPH01