Neural Rendering

D.A. Forsyth

Neural vs Differentiable Rendering

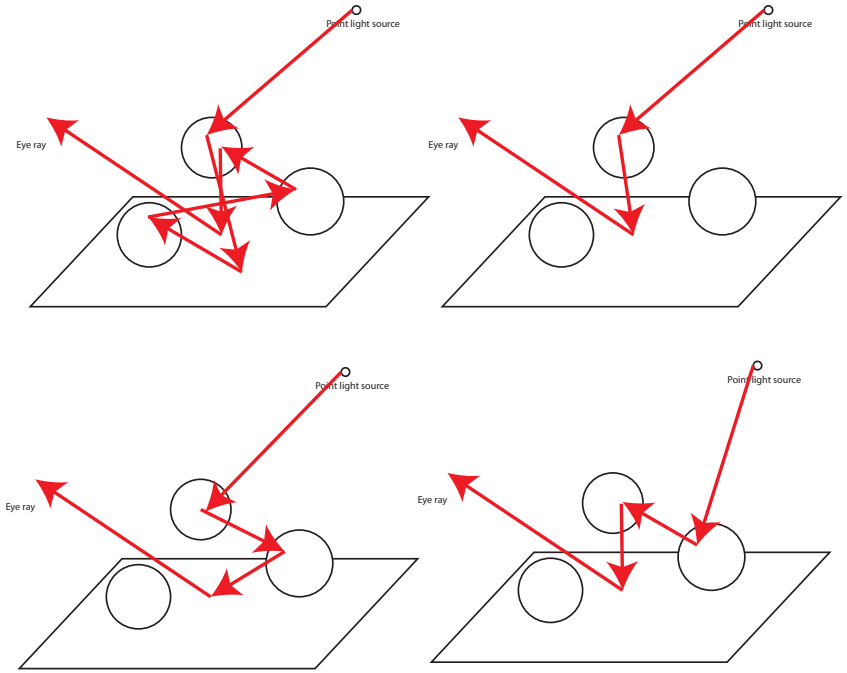
- Differentiable rendering
  - make (relatively conventional) renderer differentiable
  - usually to support inference (shape from single image, etc.)
- Neural rendering
  - use neural networks at various points in the rendering process
  - lots of methods
    - no real consensus on what a neural rendering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

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

Kalantari et al 15

Cross-bilateral filter

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

Location

Pixel color

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

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

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

Kalantari et al 15

Natural attack

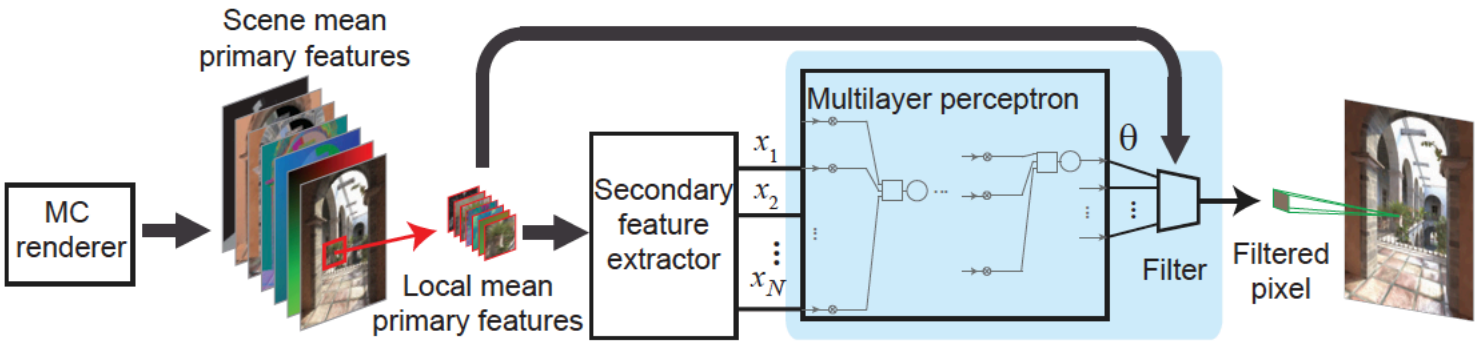


Figure 4: *Our approach combines a standard MLP (Fig. 3) with a matching filter. The local mean primary features (illustrated by a stack of images) contain color, position, and additional features such as world positions, shading normals, etc. A set of secondary features* $\{x_1, \cdots, x_N\}_i$ *(see 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.*

Kalantari et al 15

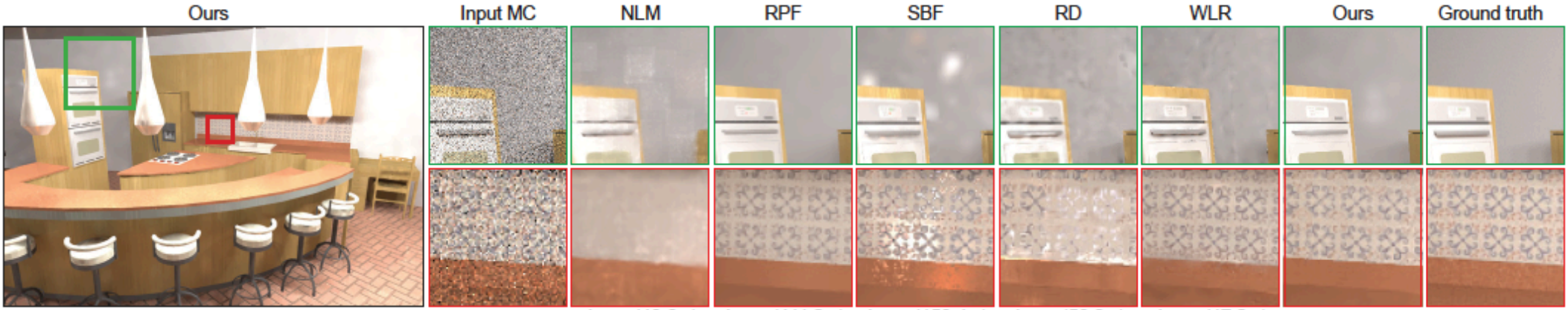


Figure 2: *Comparison between our approach and several state-of-the-art algorithms on the* KITCHEN *scene rendered at 4 spp. Note that the ground truth image is still noisy even at 32K spp. Non-local means filtering (NLM) [Rousselle et al. 2012] is a color-based method which cannot keep geometry or texture detail. Random parameter filtering (RPF) [Sen and Darabi 2012], SURE-based filtering (SBF) [Li et al. 2012], robust denoising (RD) [Rousselle 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 (RelMSE) 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.

Kalantari et al 15

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.*

Kalantari et al 15

See also

Alla Chaitanya, 17 (same problem, different architecture)

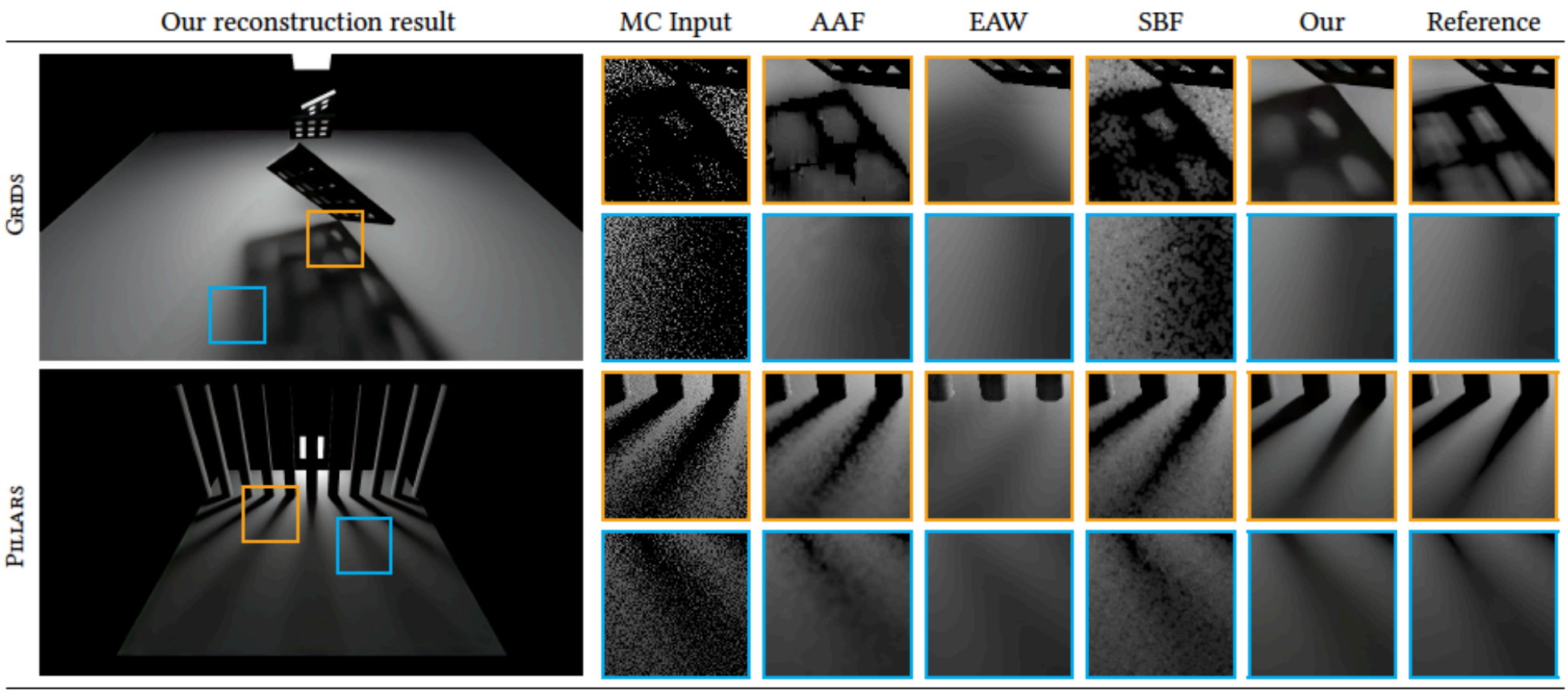


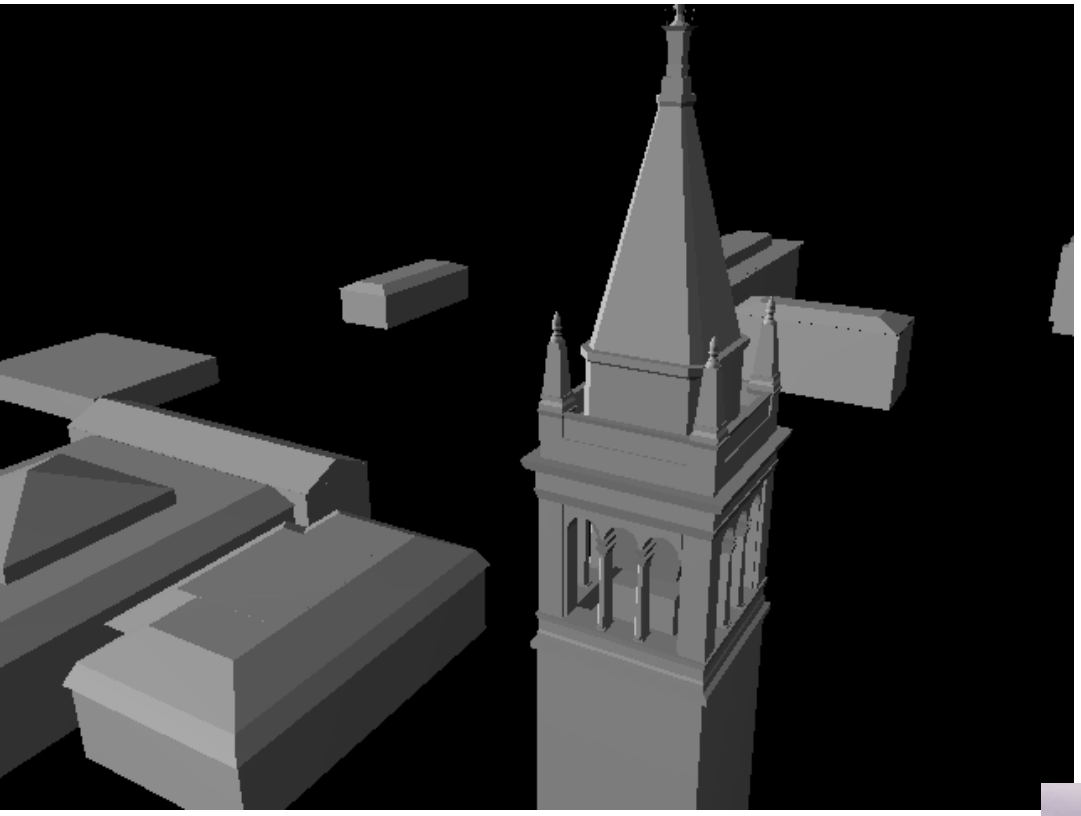
Fig. 10. Closeups for shadow filtering for 1 spp input (MC), axis-aligned filter (AAF), À-Trous wavelet filter (EAW), SURE-based filter (SBF), and our result.

Some topics…

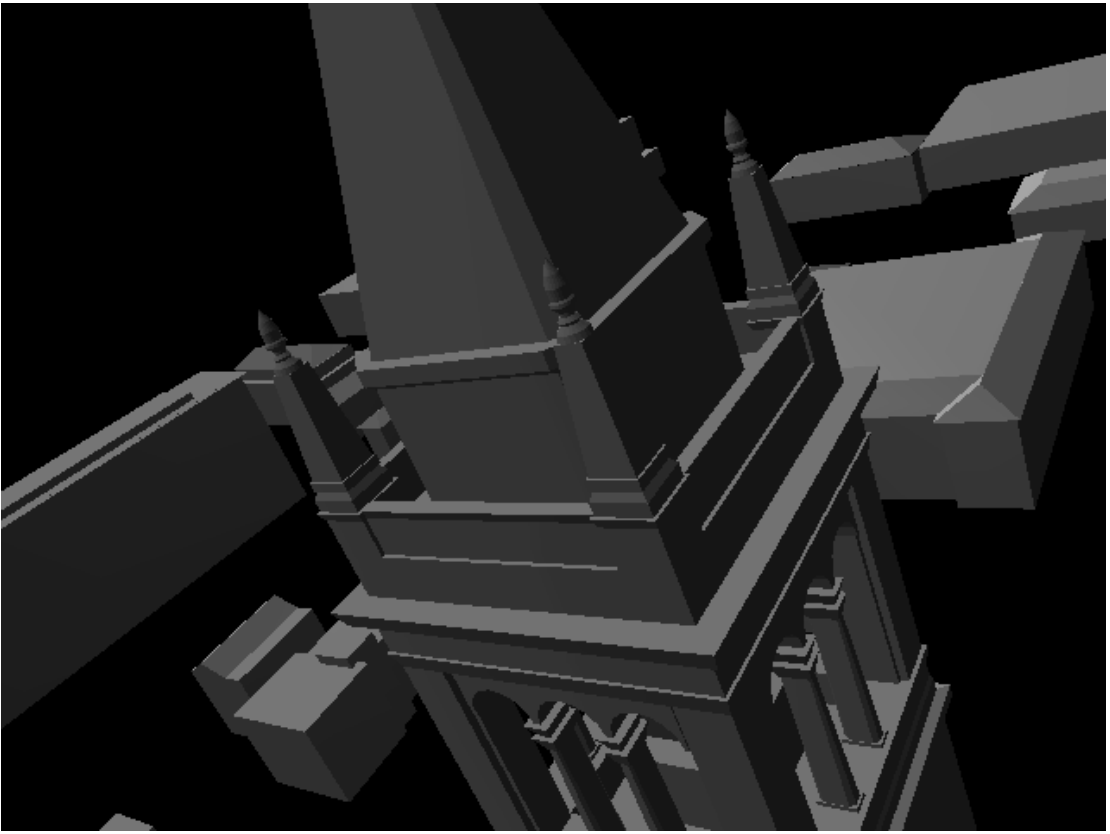
- Reduce rendering noise
  - in MCMC rendering
  - in image based rendering
  - in performance capture
- Realistic images from approximations
- Generate novel views
  - from multiview input
- Exaggerate effects
  - eg motion fields
- Reshade and relight

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



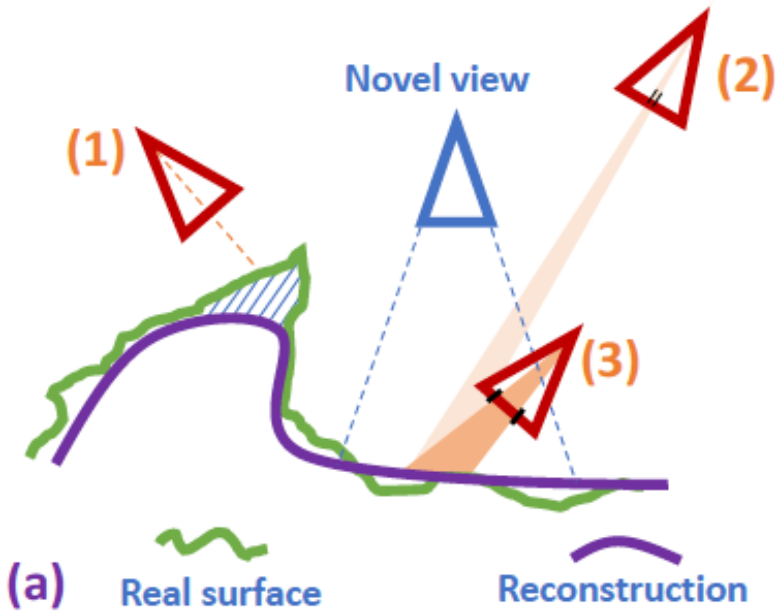
https://www.pauldebevec.com/Campanile/#movie



https://www.pauldebevec.com/Campanile/#movie

IBR as blending

The novel view is a blend;
blend is driven by relief from reconstruction, normals, etc. Strategy: build the best blender.



Hedman, 18

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

CNN

Blending

Global Mesh Render + 4 Mosaics

Blend Weights

Blended RGB Output

Hedman, 18

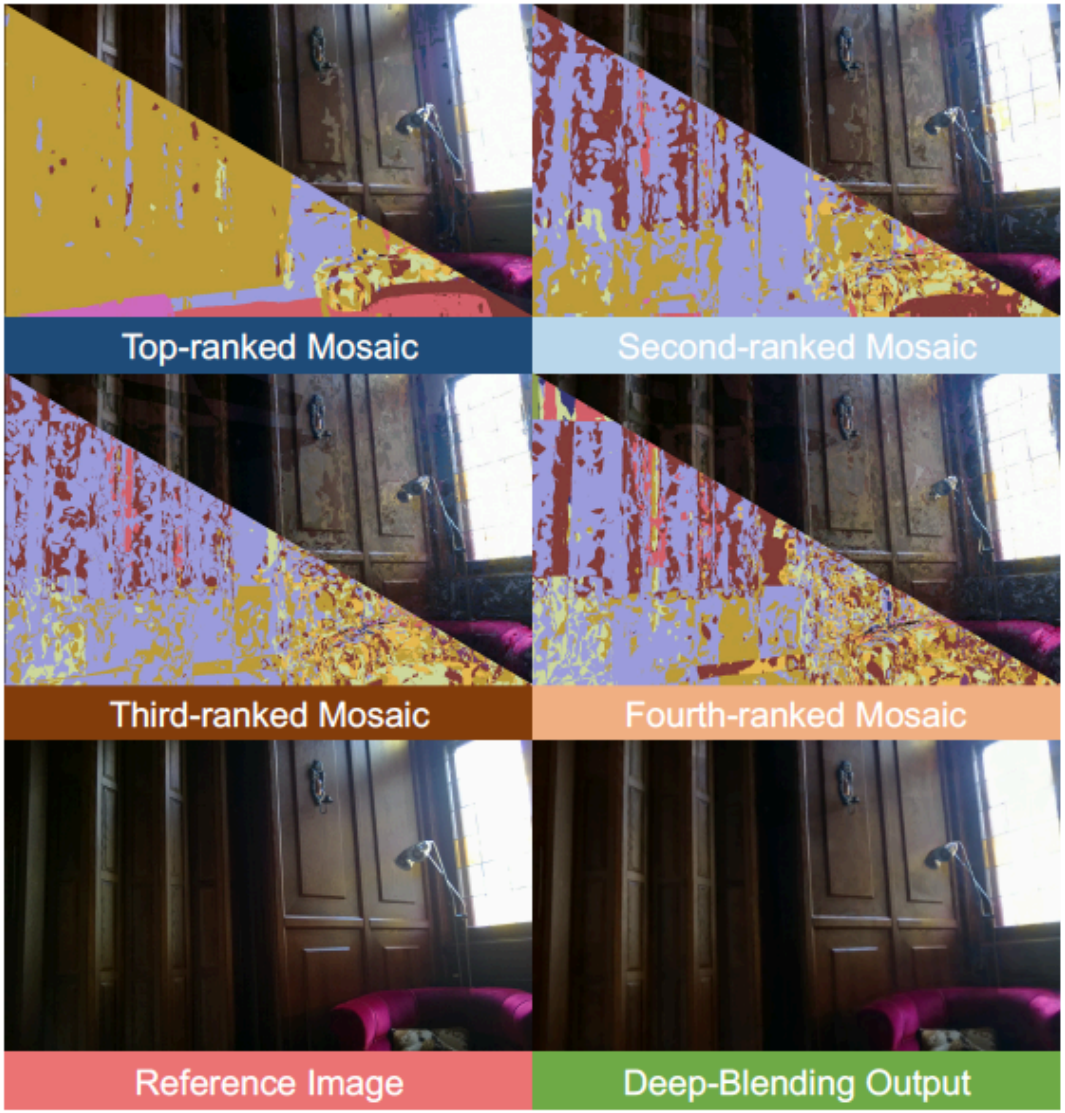
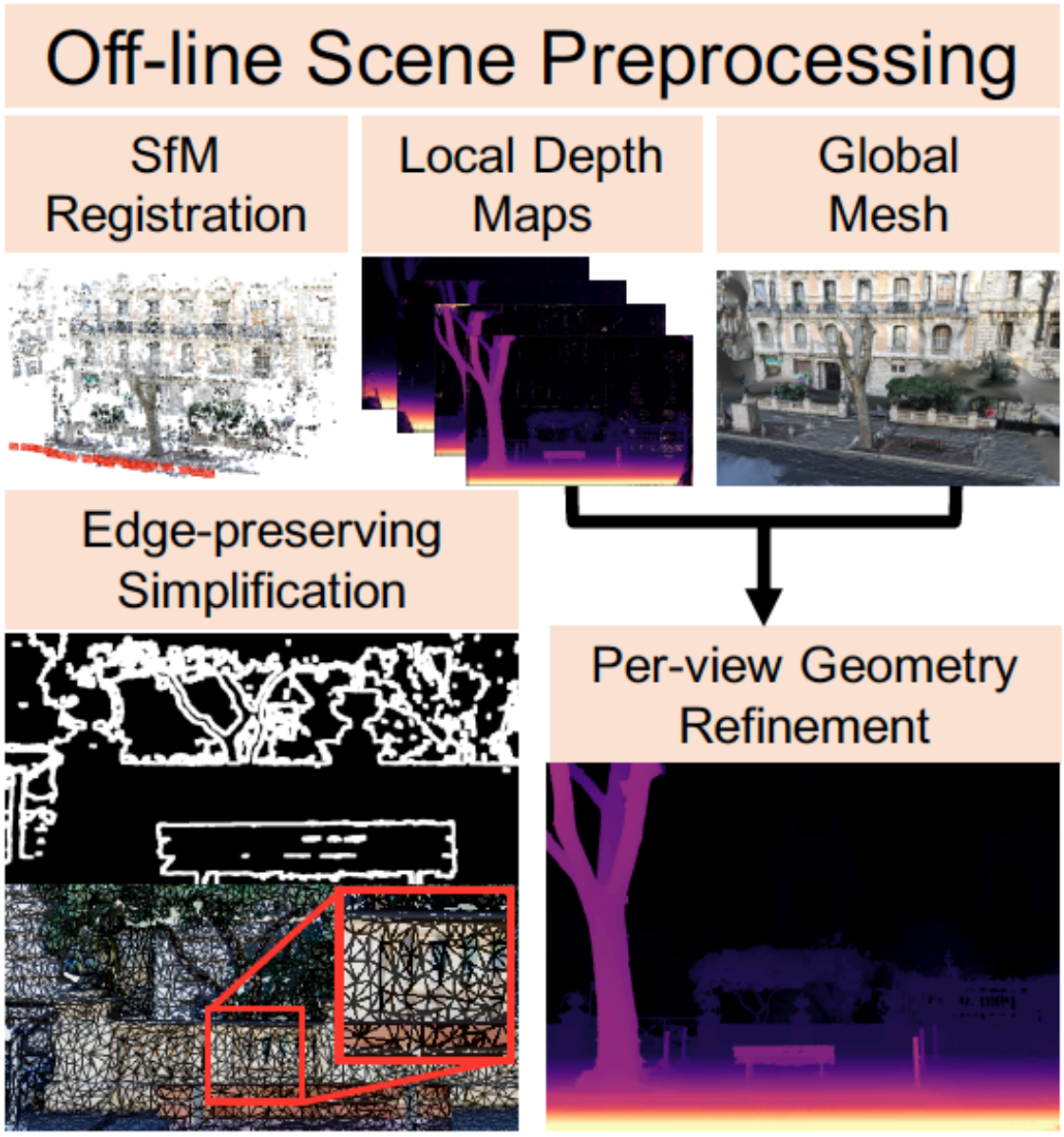
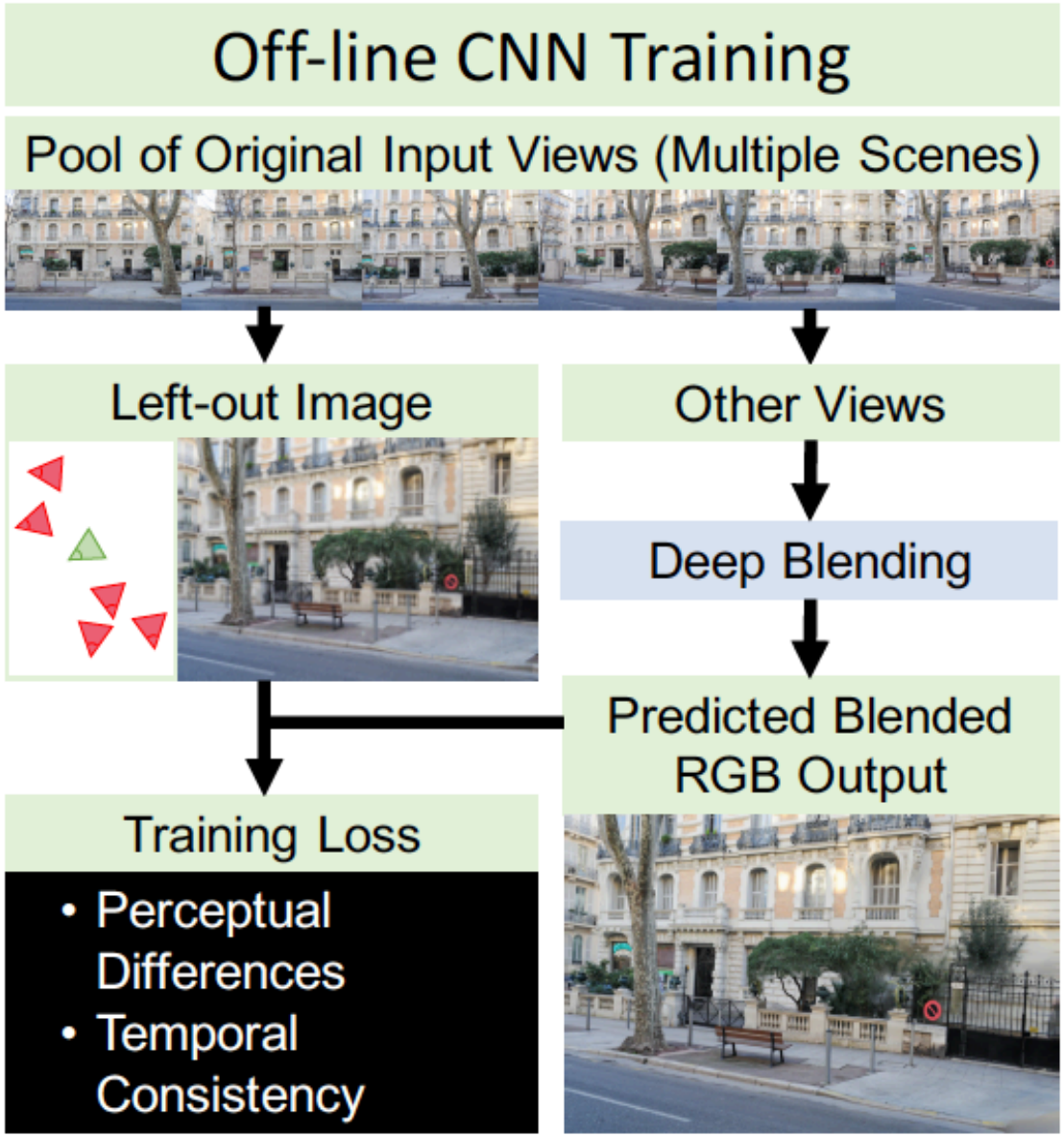


Fig. 10. Our network takes as input ranked mosaics generated from a set of warped candidate views. For each pixel, the candidates are ranked based on their expected blending contribution, and 4 color-image mosaics are formed from the top 4 rankings. Example mosaics are shown in the first two rows. The 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).

Hedman, 18


Off-line Scene Preprocessing
SfM Registration
Local Depth Maps
Global Mesh
Edge-preserving Simplification
Per-view Geometry Refinement

Hedman, 18


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

Hedman, 18

Training

- Losses:
  - per frame perceptual loss

$$\mathcal{L}(I_N, I_R) = |I_N - I_R| + |\text{VGG16}_{relu\,12}(I_N) - \text{VGG16}_{relu\,12}(I_R)| + |\text{VGG16}_{relu22}(I_N) - \text{VGG16}_{relu22}(I_R)|$$

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

$$\mathcal{L}_T(I_N, I_R) = \mathcal{L}(I_N^t, I_R) + 0.33 * \mathcal{L}(I_N^{t-1}, \mathcal{W}_f(I_N^t)),$$

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

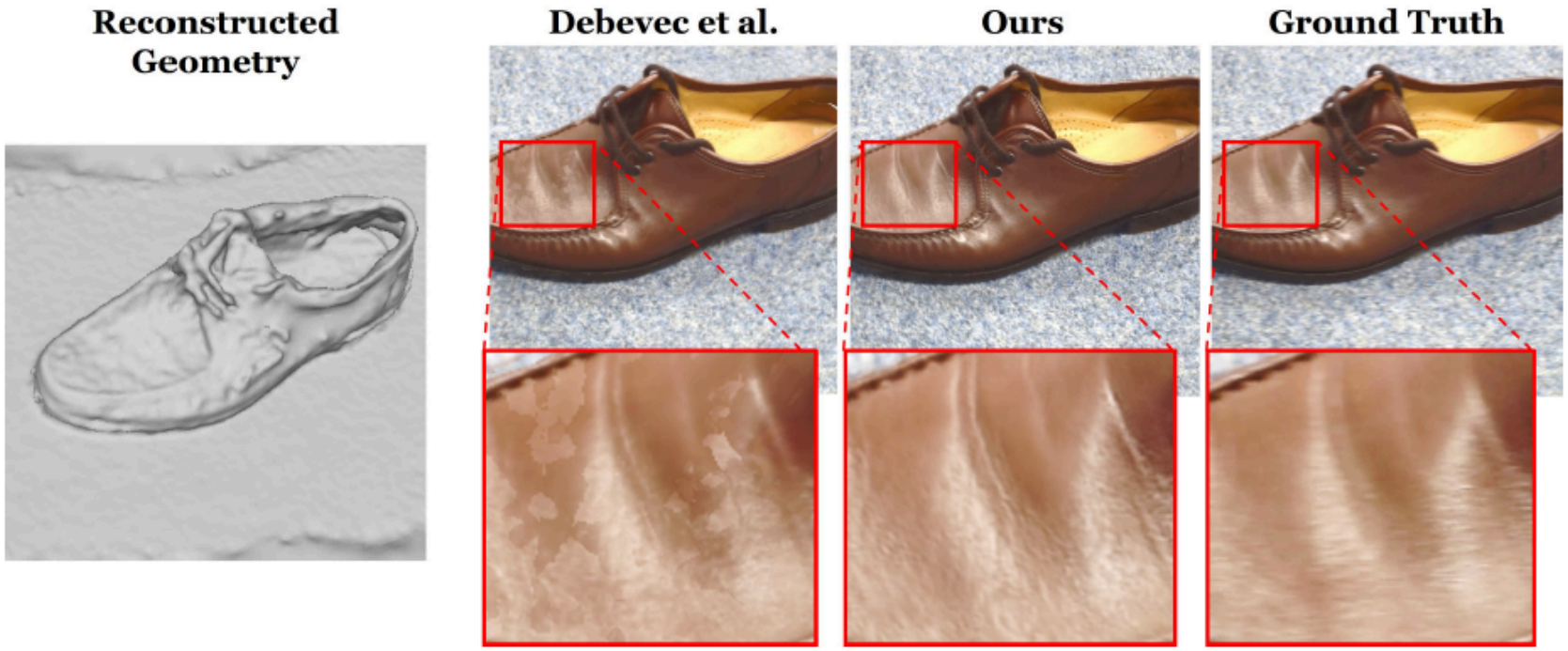


Figure 14: Image synthesis on real data: we show a comparison to the IBR technique of Debevec et al. (1998). From left to right: reconstructed geometry of the object, result of IBR, our result, and the ground truth.

Idea: predict these separately

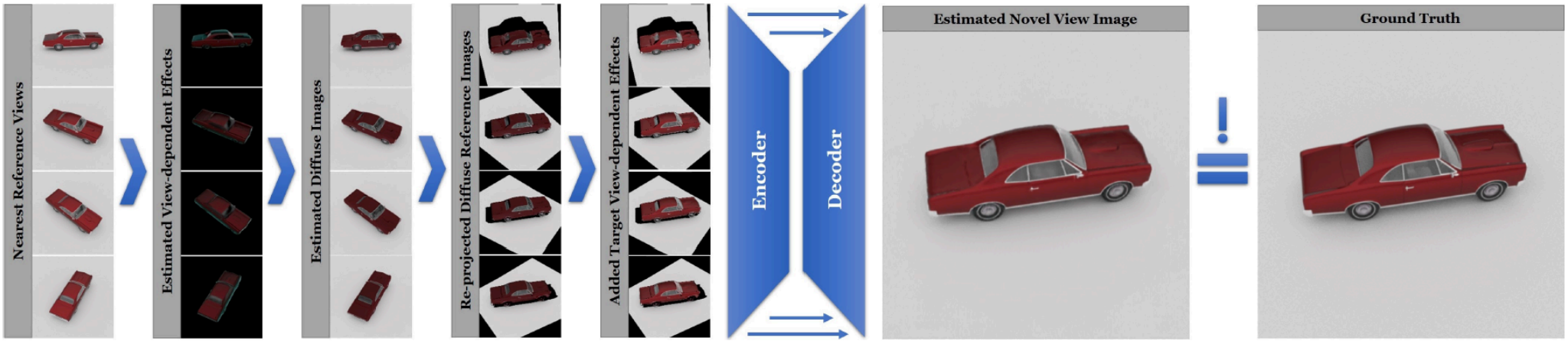


Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected 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.

Thies et al 20

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).

Thies et al 20

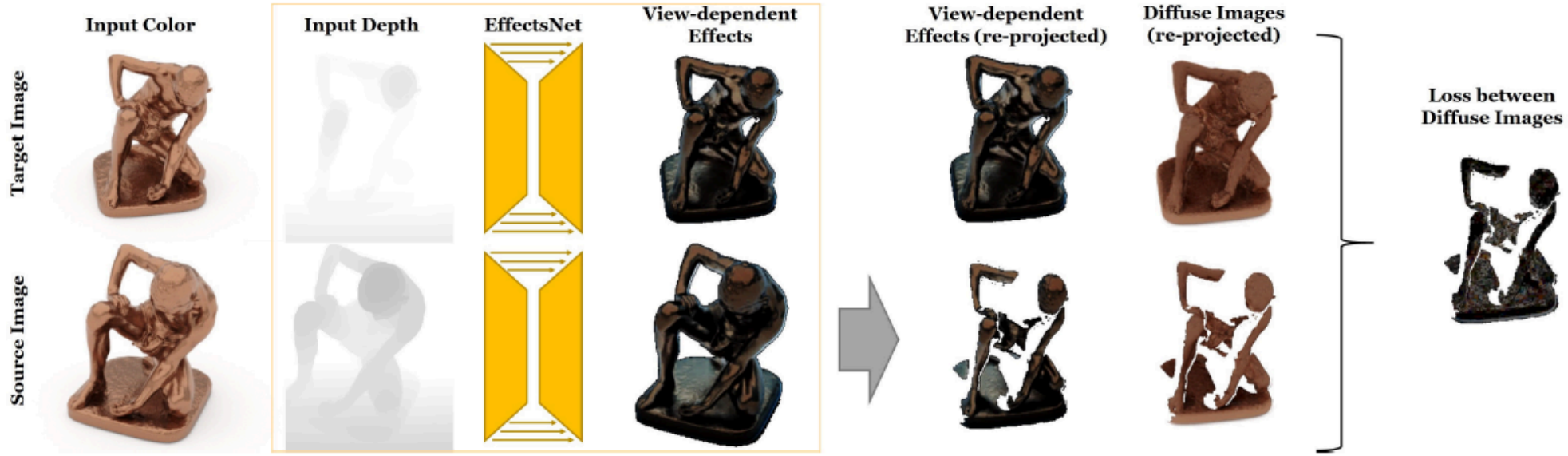


Figure 2: *EffectsNet* is trained in a self-supervised fashion. In a Siamese scheme, two random images from the training set are chosen and fed into the network to predict the view-dependent effects based on the current view and the respective depth map. After re-projecting the source image to the target 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.

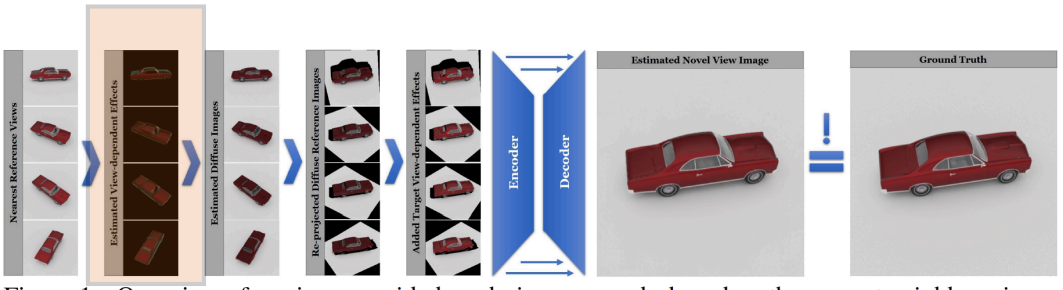


Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected 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

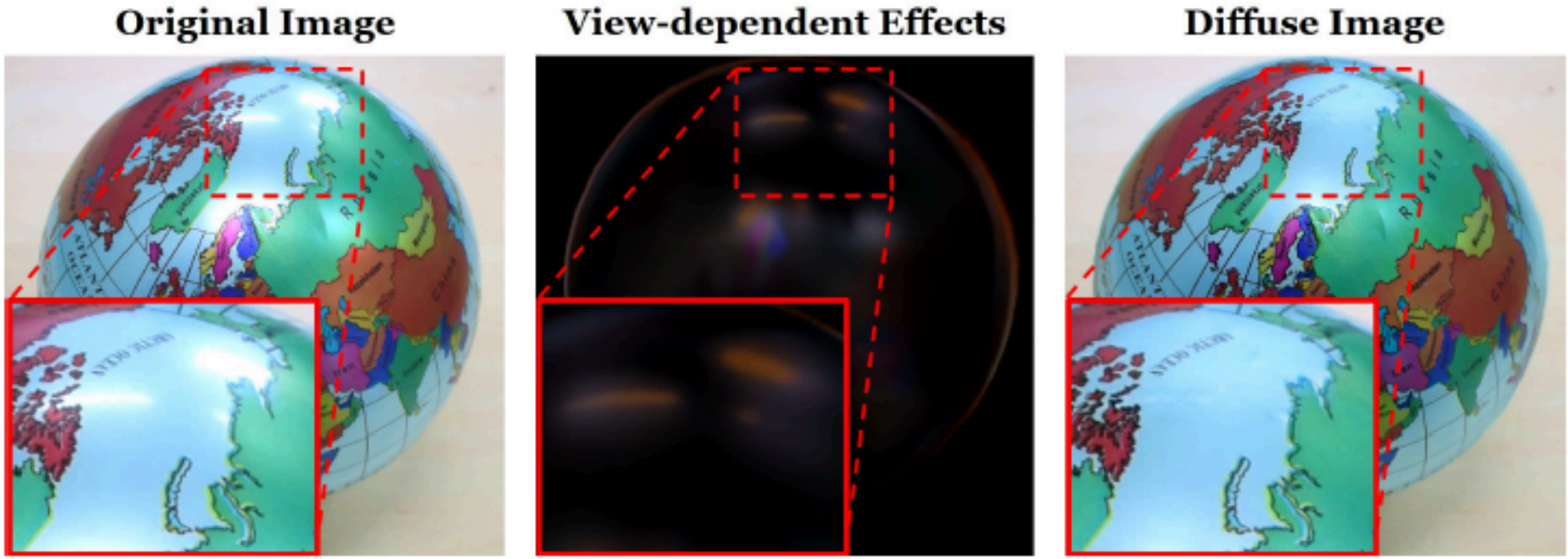


Figure 4: Prediction and removal of view-dependent effects of a highly specular real object.

Thies et al 20

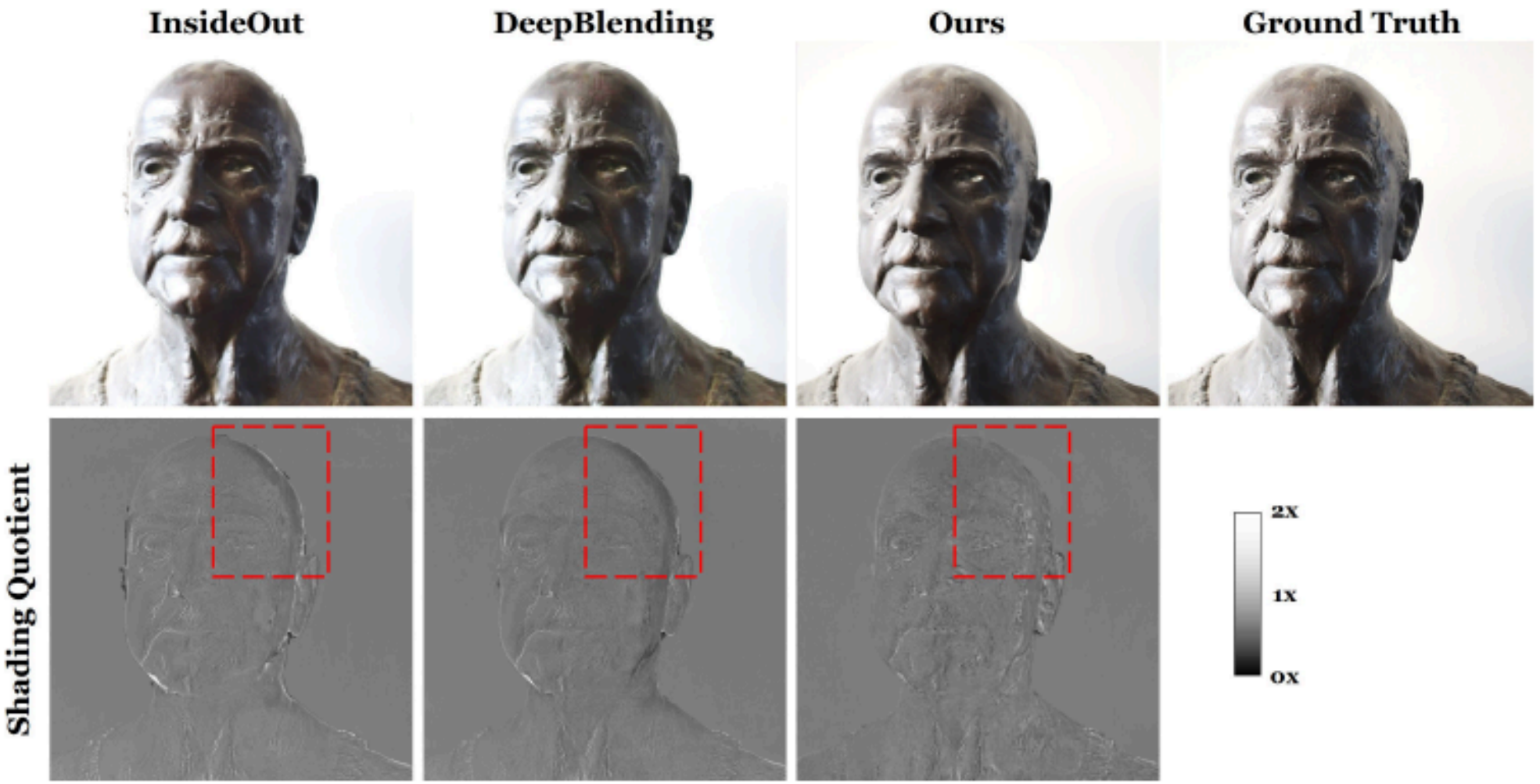


Figure 6: Comparison to the IBR method InsideOut of Hedman et al. (2016) and the learned IBR blending method DeepBlending of Hedman et al. (2018). To better show the difference in shading, we computed the quotient of the resulting image and the ground truth. A perfect reconstruction would result in a 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

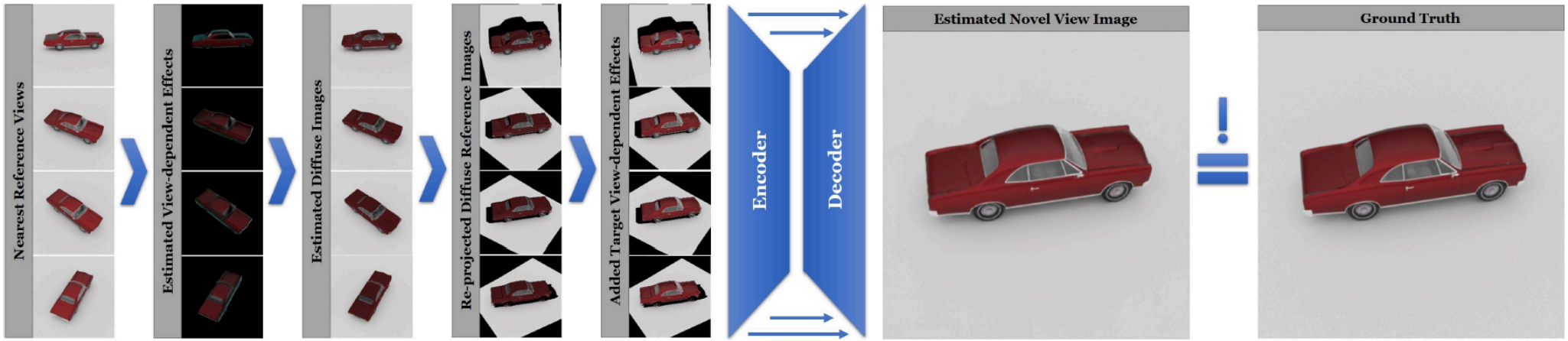


Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected 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

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…

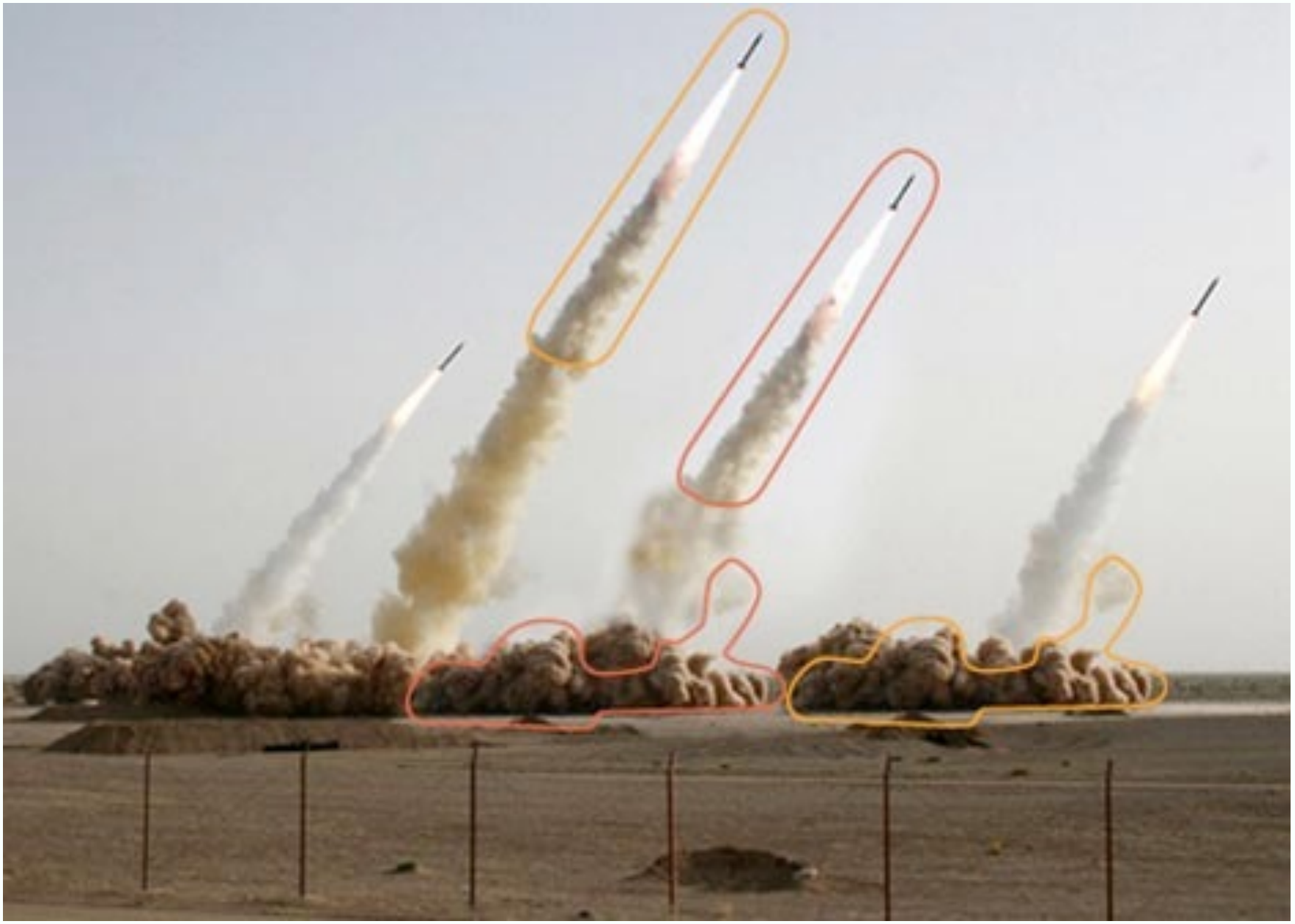
- Reduce rendering noise
  - in MCMC rendering
  - in image based rendering
  - in performance capture
- Realistic images from approximations
  - texture synthesis history
- Generate novel views
  - from multiview input
- Exaggerate effects
  - eg motion fields
- Reshade and relight

Texture

CS 419

Slides by Ali Farhadi

Texture scandals!!



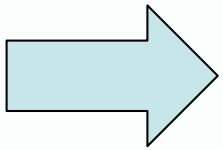
Bush campaign digitally altered TV ad

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.

Two crucial algorithmic points

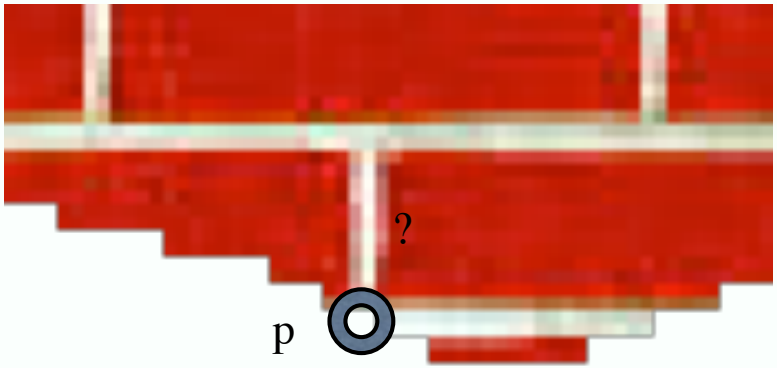
- Nearest neighbors
  - again and again and again
- Dynamic programming
  - likely new; we'll use this again, too

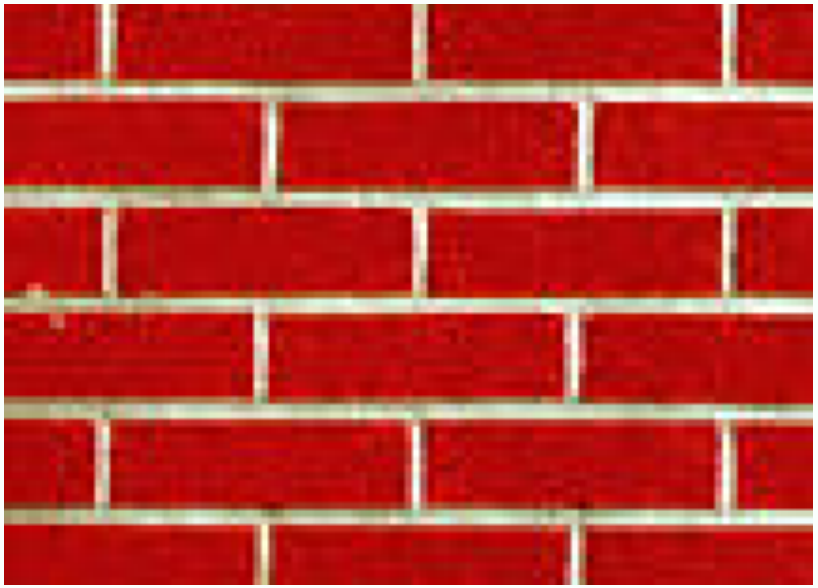
Texture Synthesis



Efros & Leung ICCV99

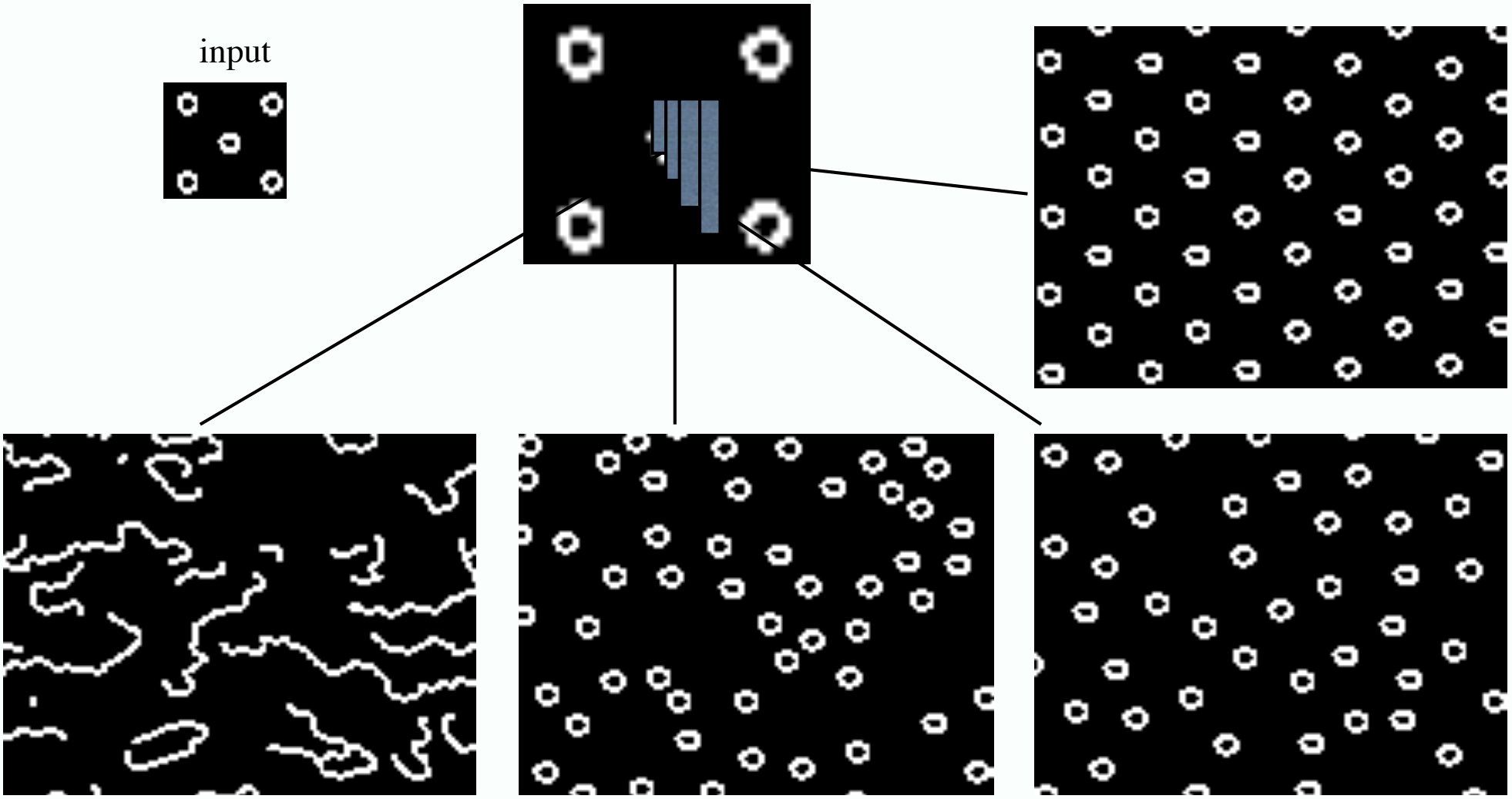
How to paint this pixel?





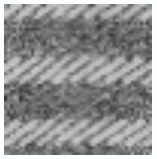
Input texture

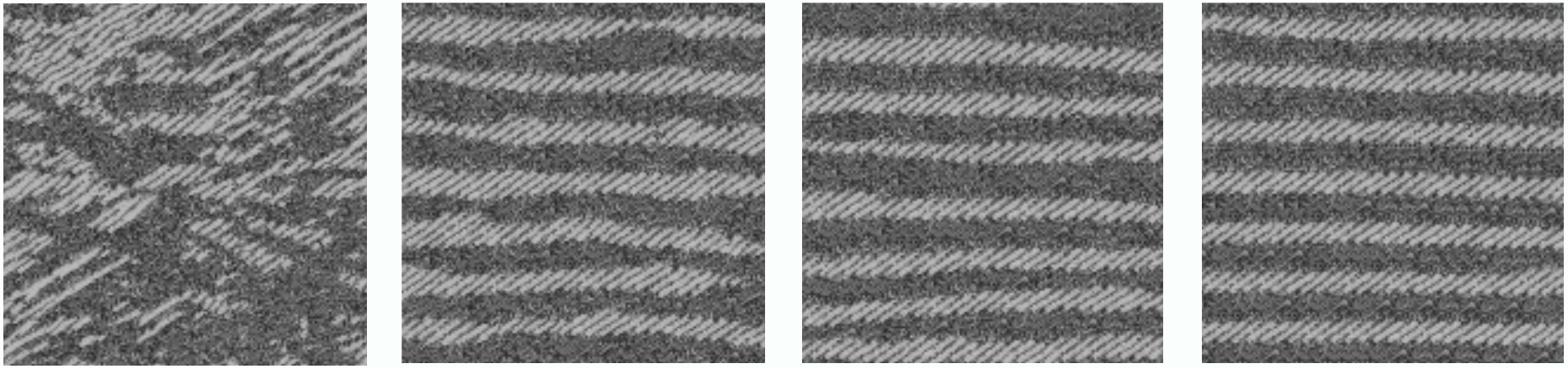
Neighborhood size

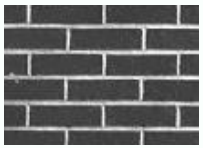


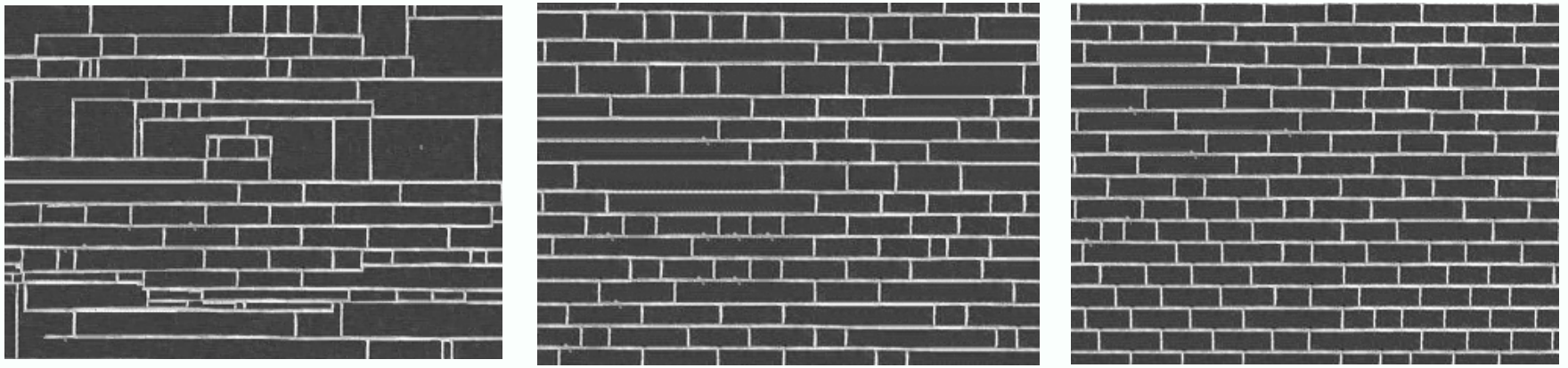
Efros & Leung ICCV99

Varying Window Size





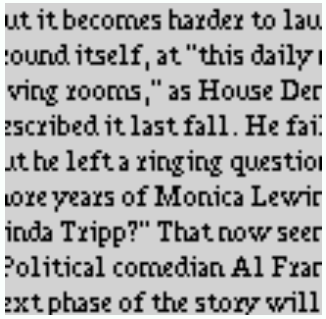


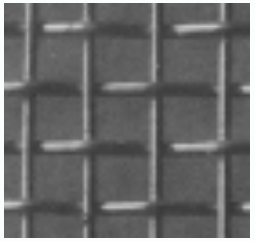


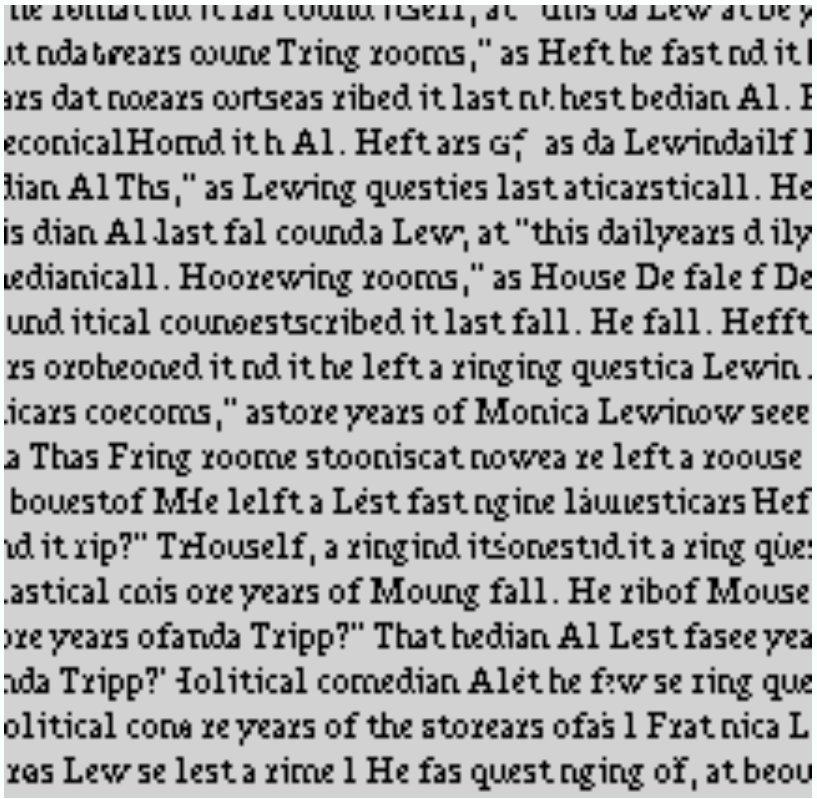
Increasing window size

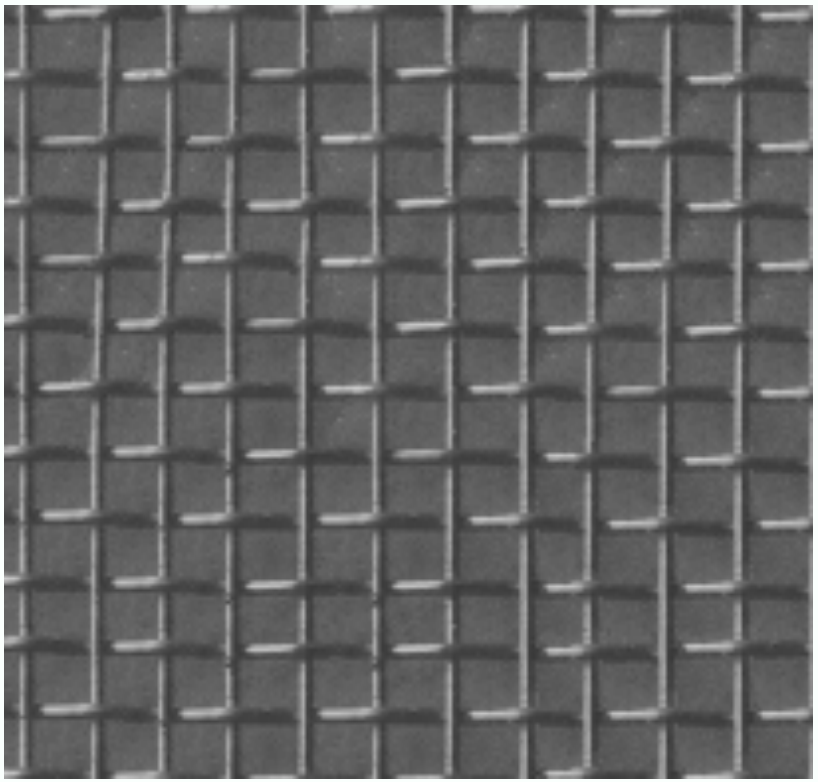
Efros & Leung ICCV99

More Results



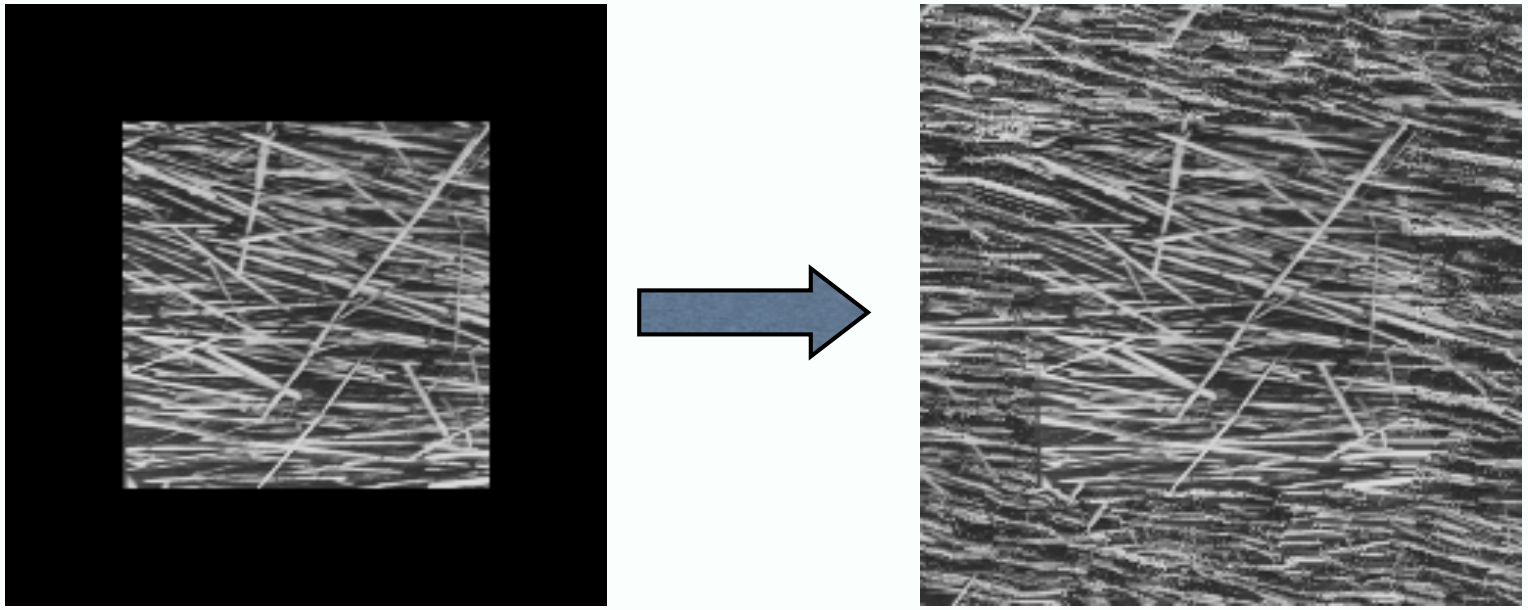


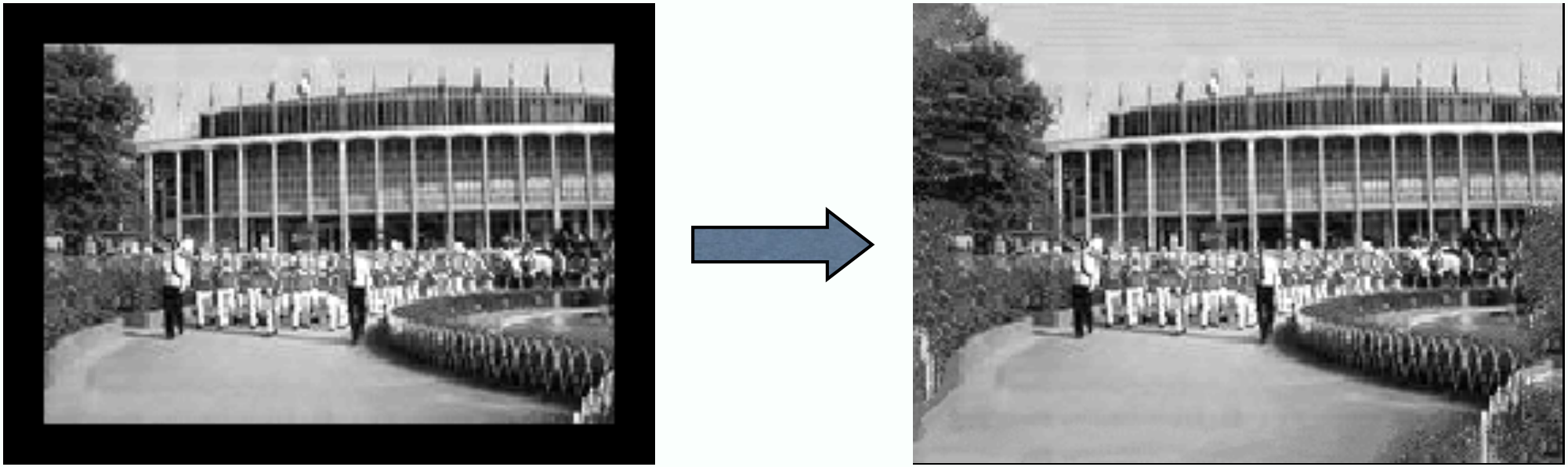




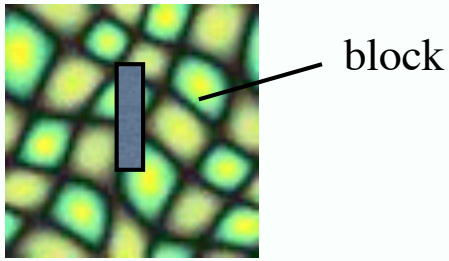
Efros & Leung ICCV99

Extrapolation

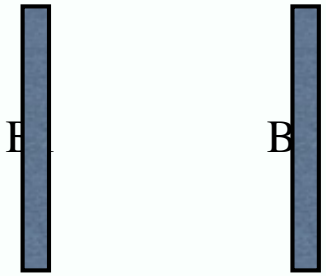




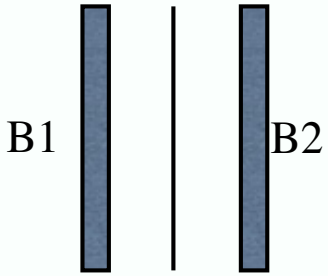
Efros & Leung ICCV99



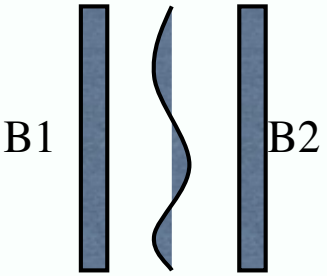
Input texture



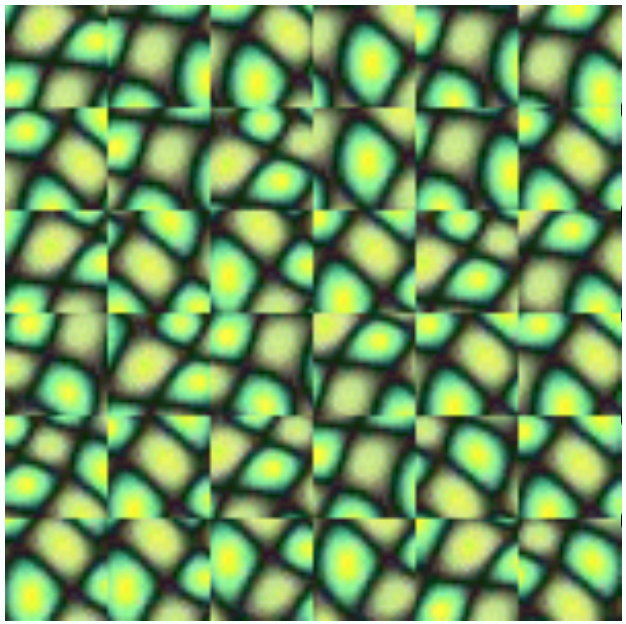
Random placement
of blocks

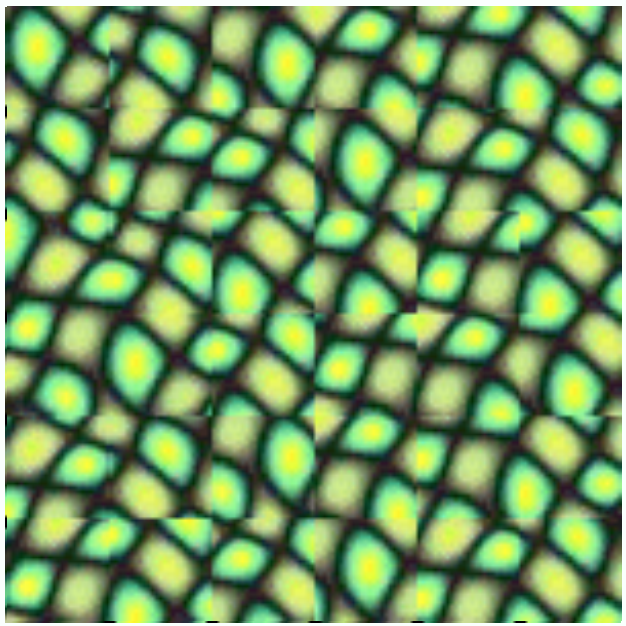


Neighboring blocks
constrained by overlap



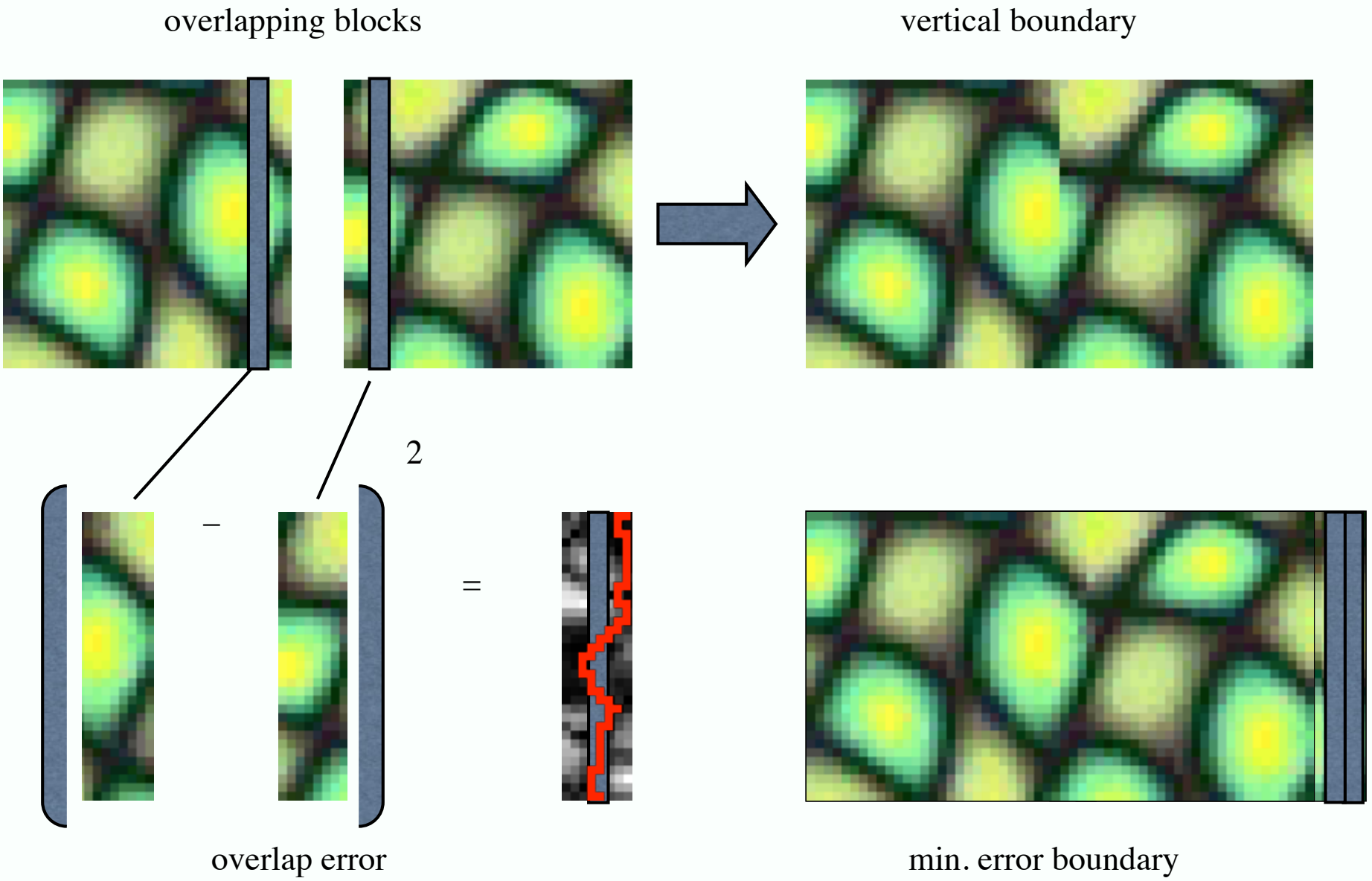
Minimal error
boundary cut

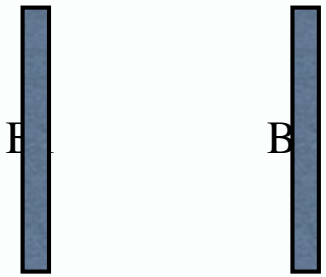




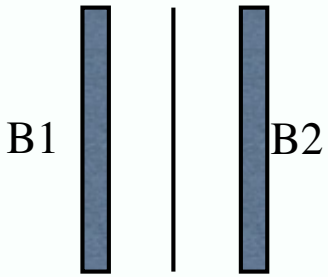
Efros & Freeman SIGGRAPH01

Minimal error boundary

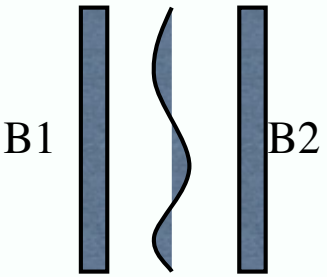




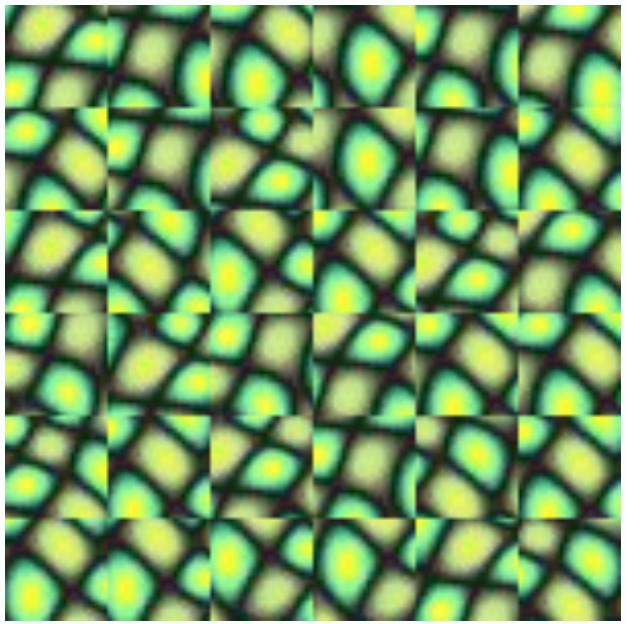
Random placement
of blocks

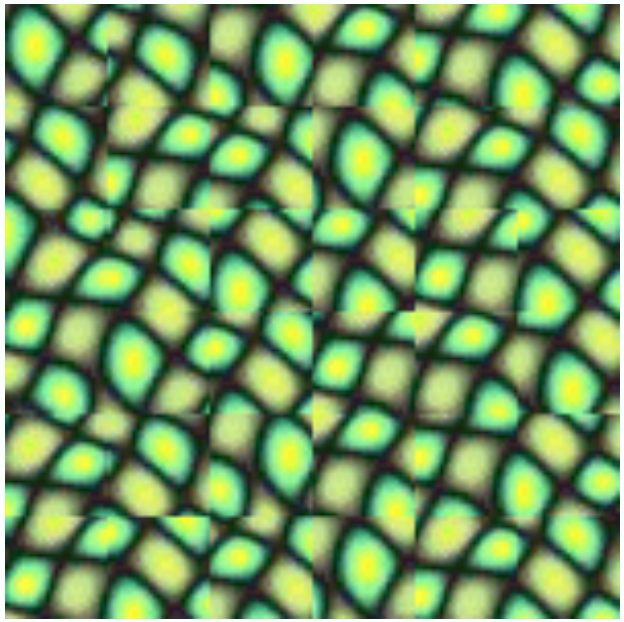


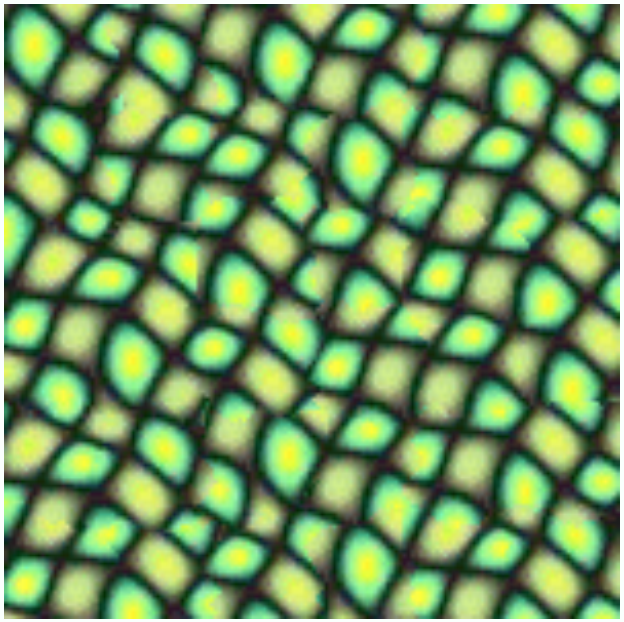
Neighboring blocks
constrained by overlap



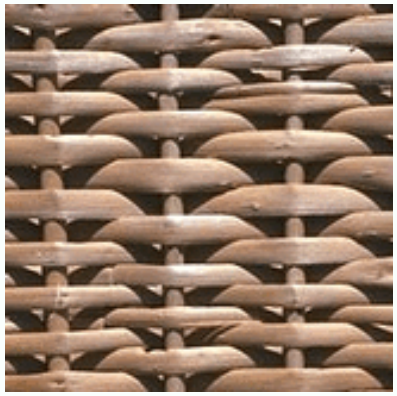
Minimal error
boundary cut

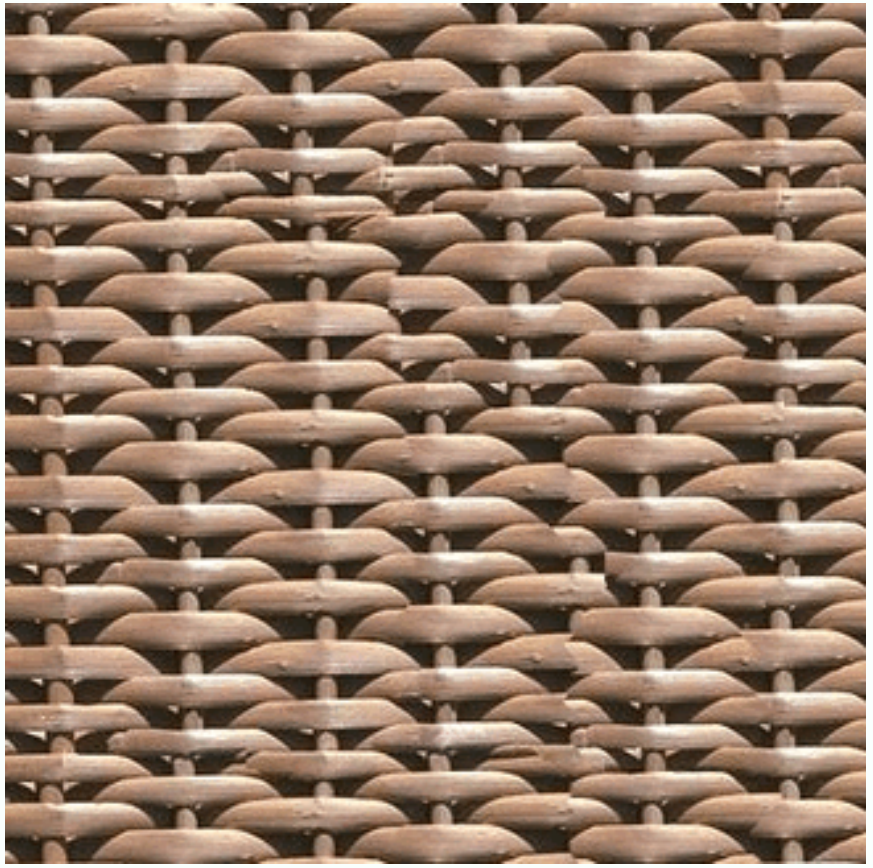




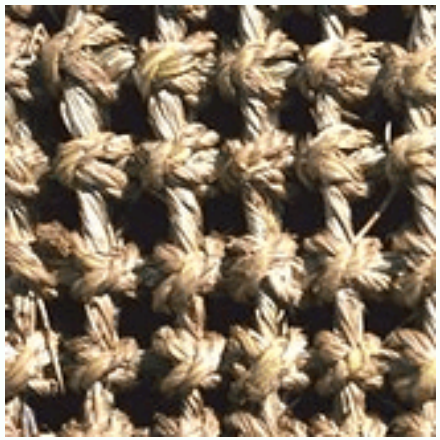


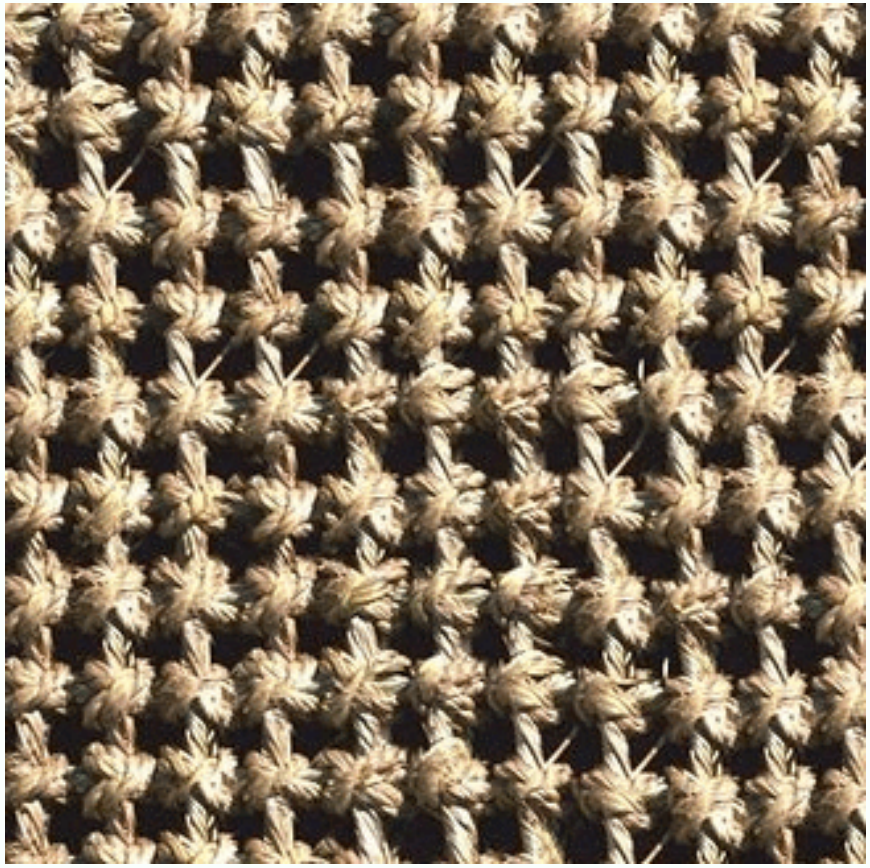
More Results





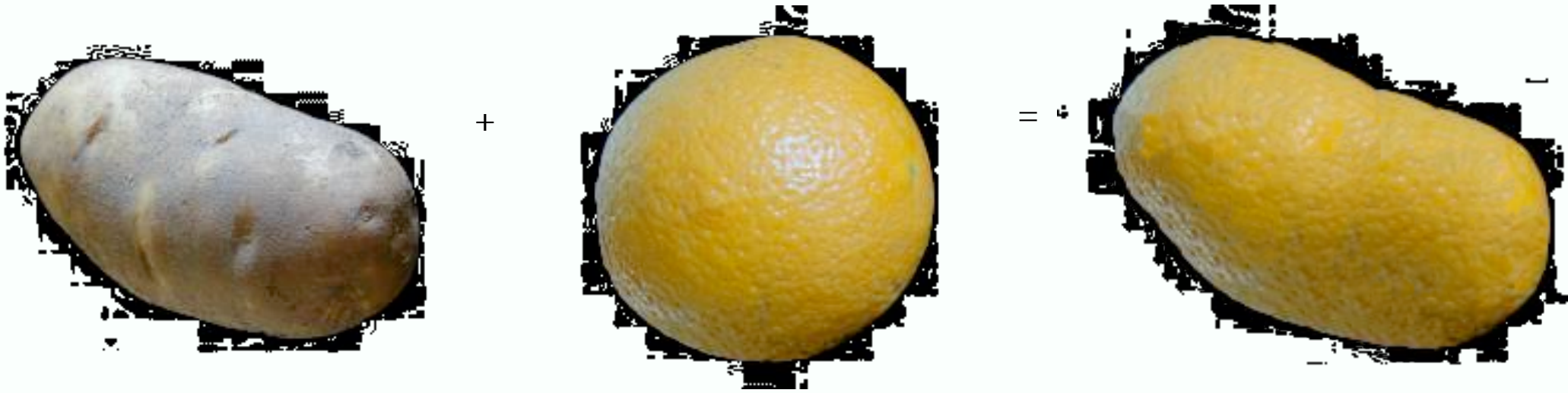
More Results





Texture Transfer

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



Decomposing shape and texture

Very challenging

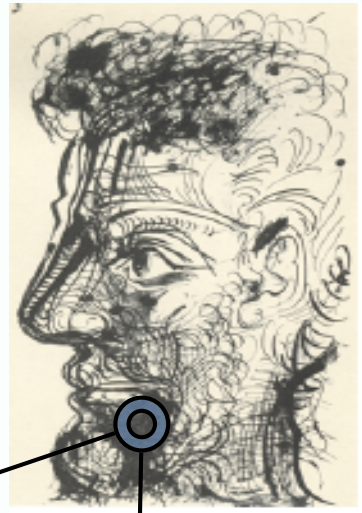
Walk around

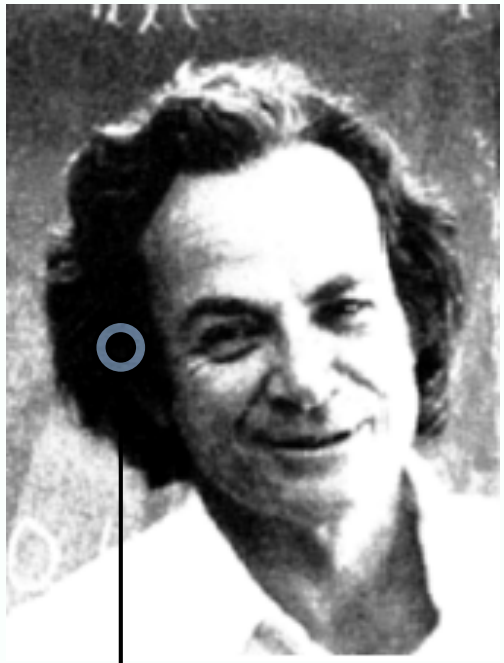
Add some constraint to the search

Efros & Freeman SIGGRAPH01

Source Texture

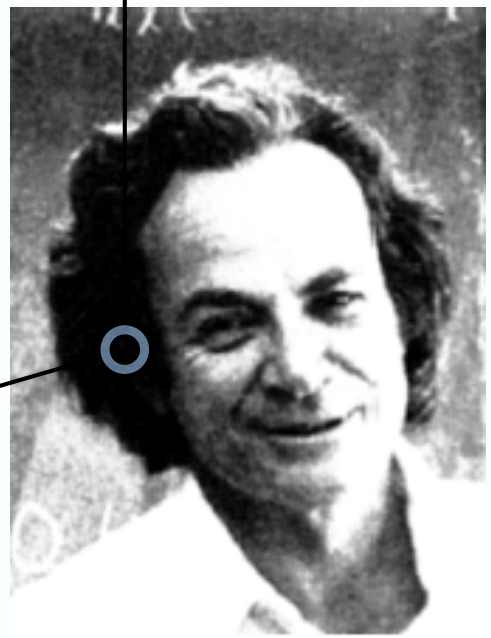
Destination

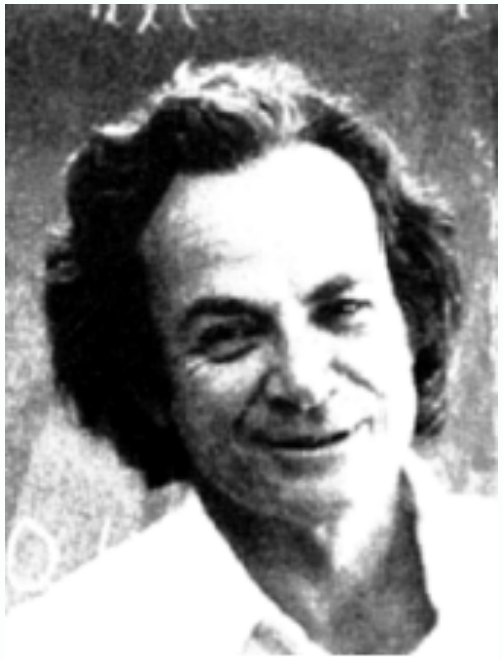




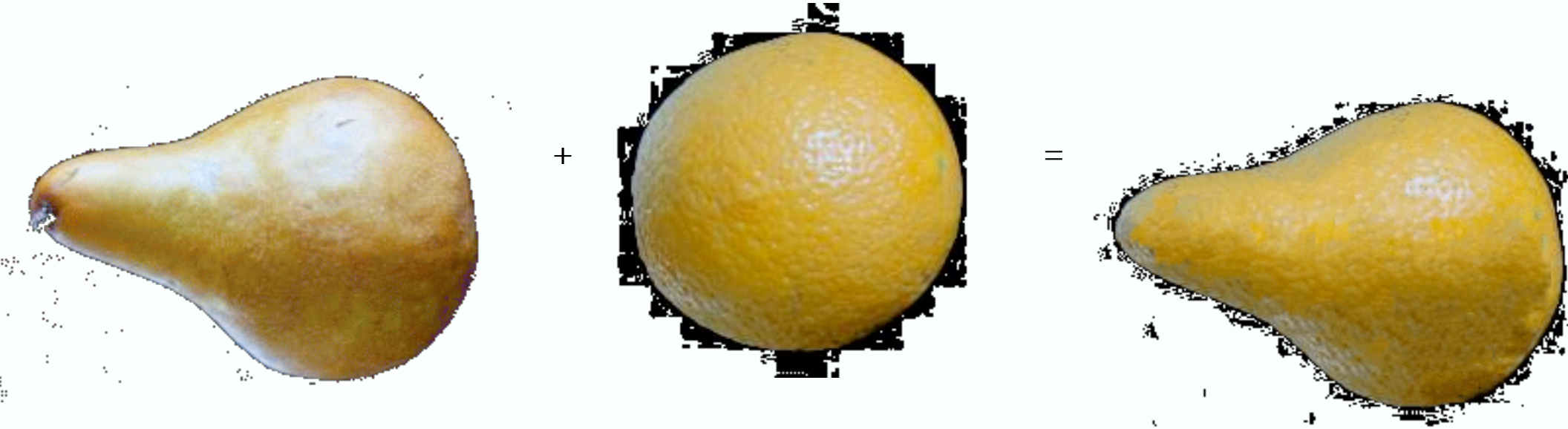
Source Map

Destination Map

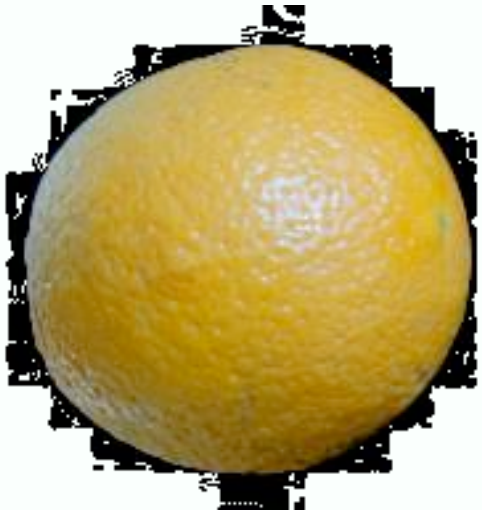




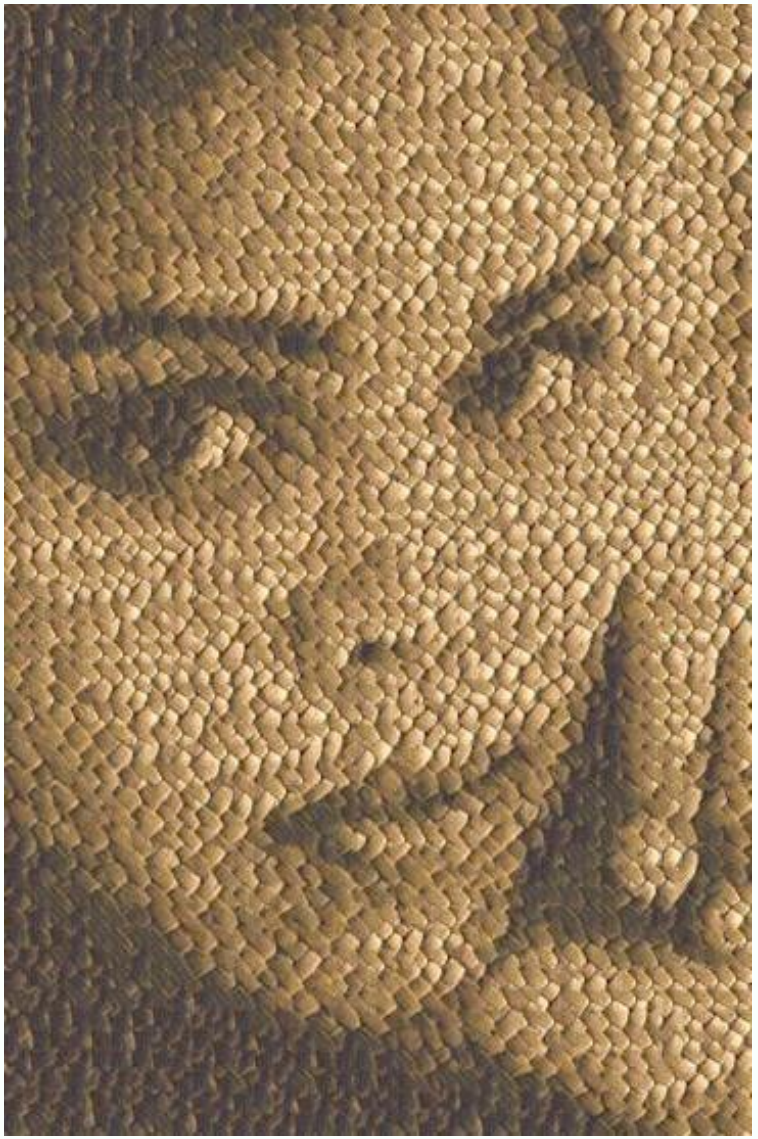
Texture Transfer



+



=



parmesan

+

=

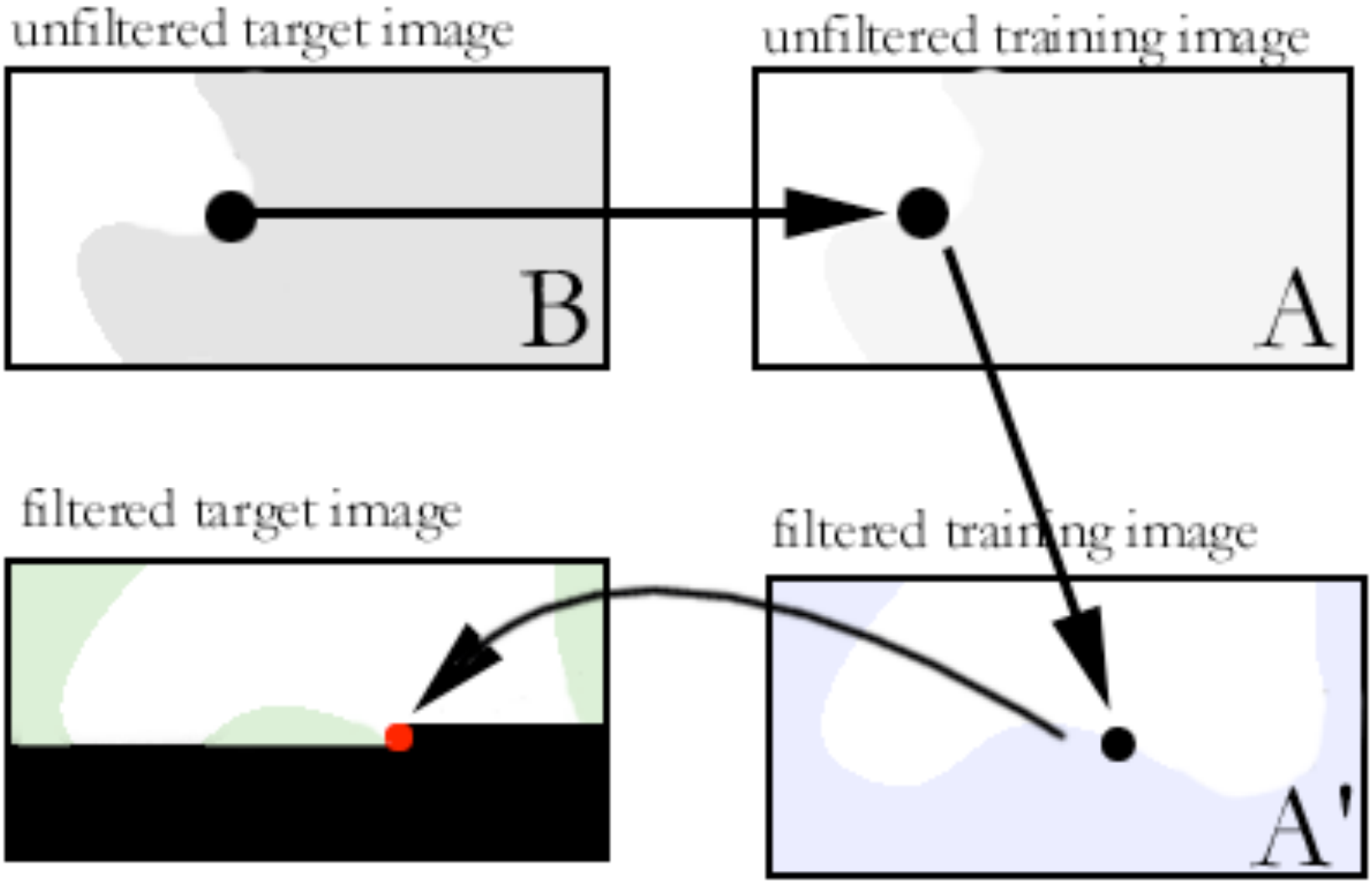
rice

+

=

Efros & Freeman SIGGRAPH01

Image Analogies



Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

Training

Unfiltered source (*A*)

Filtered source (*A'*)

Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

::

:

B

B’

Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

::

:

B

B’

Learn to Blur

Unfiltered source (A)

Filtered source (A')

Unfiltered target (B)

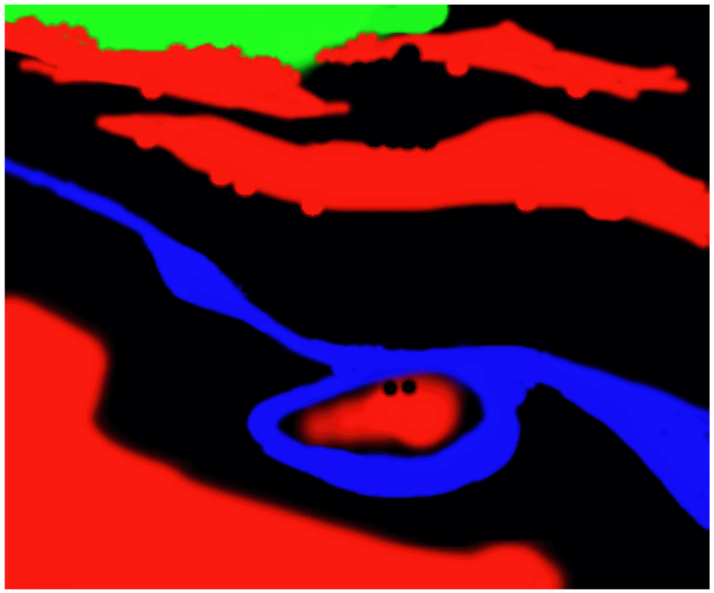
Filtered target (B')

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*)

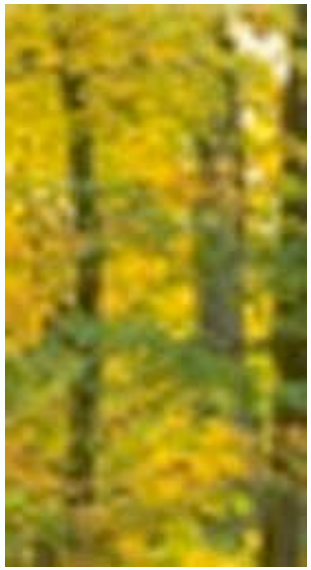
Filtered source (*A'*)

Unfiltered target (*B*)

Filtered target (*B'*)

Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

Super-resolution



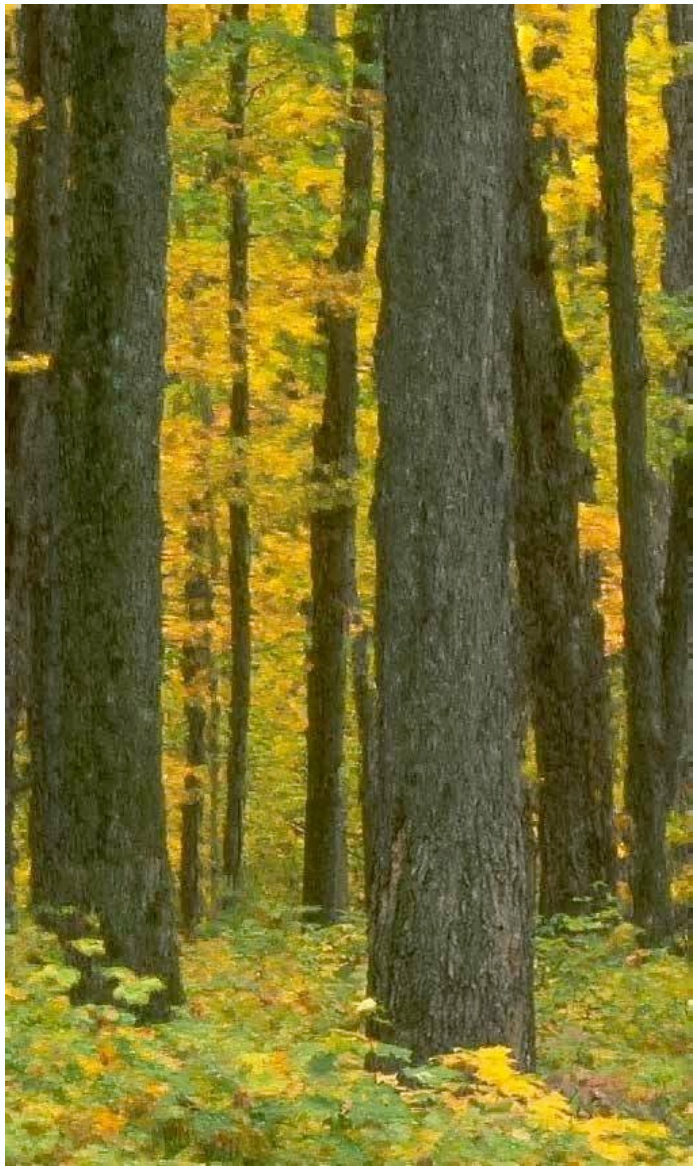
A

A’

Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

Super-resolution (result!)

B



B’

Hertzman, Jacobs, Oliver, Curless, and Salesin, SIGGRAPH01

Training images