Neural Rendering

D.A. Forsyth

Neural vs Differentiable Rendering

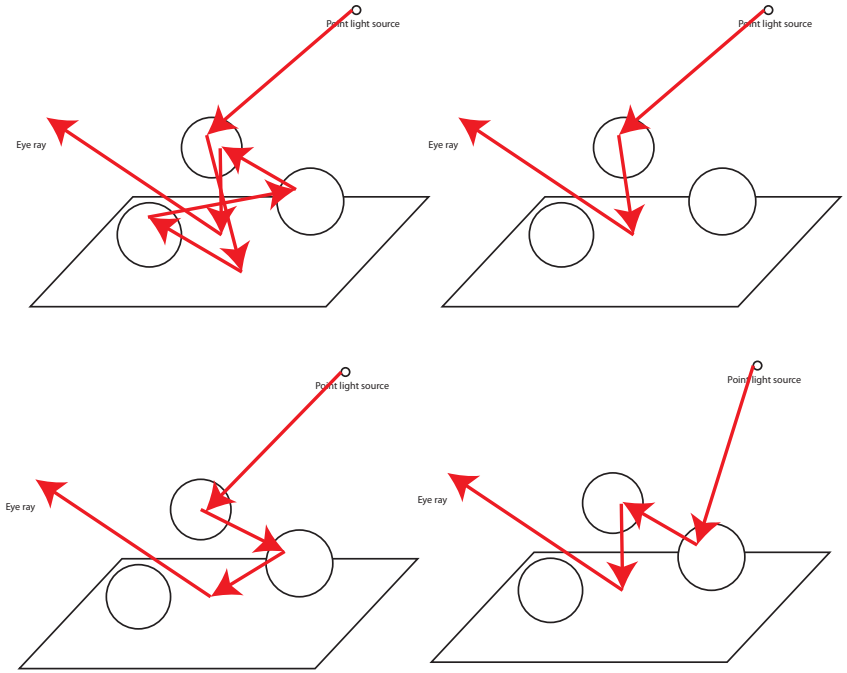
- Differentiable rendering
  - make (relatively conventional) renderer differentiable
  - usually to support inference (shape from single image, etc.)
- Neural rendering
  - use neural networks at various points in the rendering process
  - lots of methods
    - no real consensus on what a neural rendering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

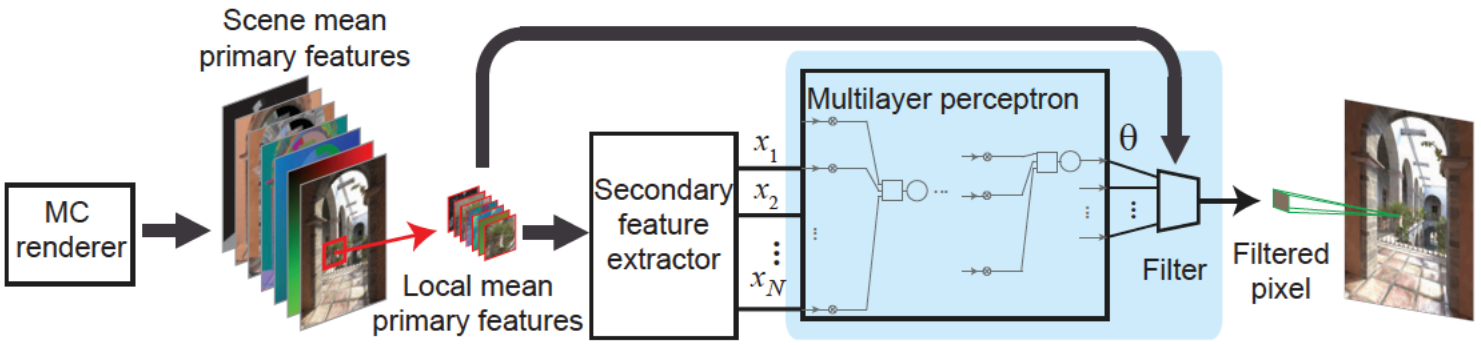


Figure 4: *Our approach combines a standard MLP (Fig. 3) with a matching filter. The local mean primary features (illustrated by a stack of images) contain color, position, and additional features such as world positions, shading normals, etc. A set of secondary features* $\{x_1, \cdots, x_N\}_i$ *(see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

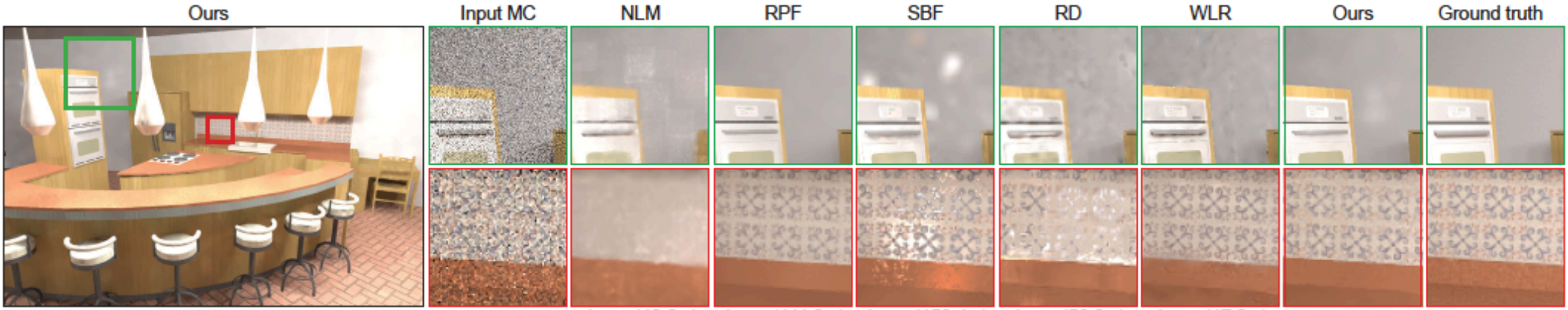


Figure 2: *Comparison between our approach and several state-of-the-art algorithms on the* KITCHEN *scene rendered at 4 spp. Note that the ground truth image is still noisy even at 32K spp. Non-local means filtering (NLM) [Rousselle et al. 2012] is a color-based method which cannot keep geometry or texture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

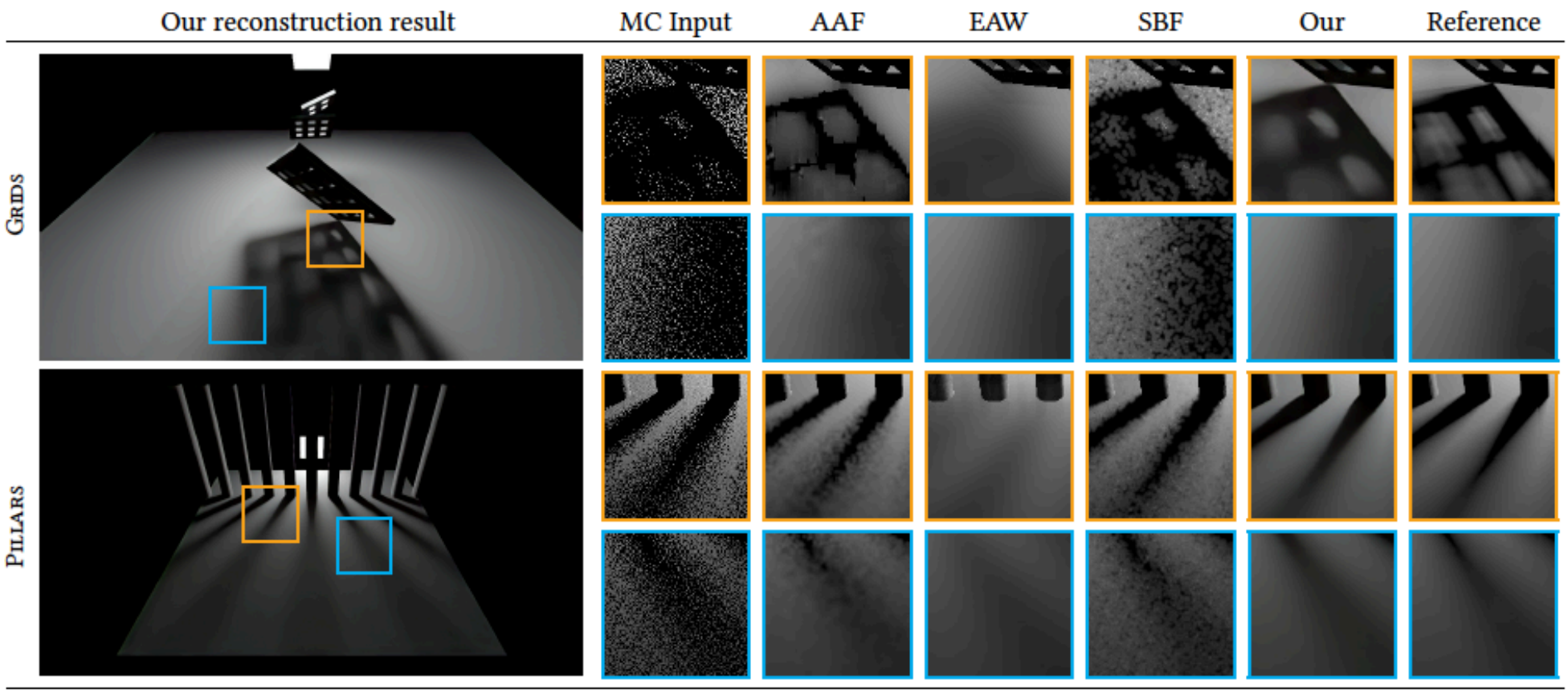


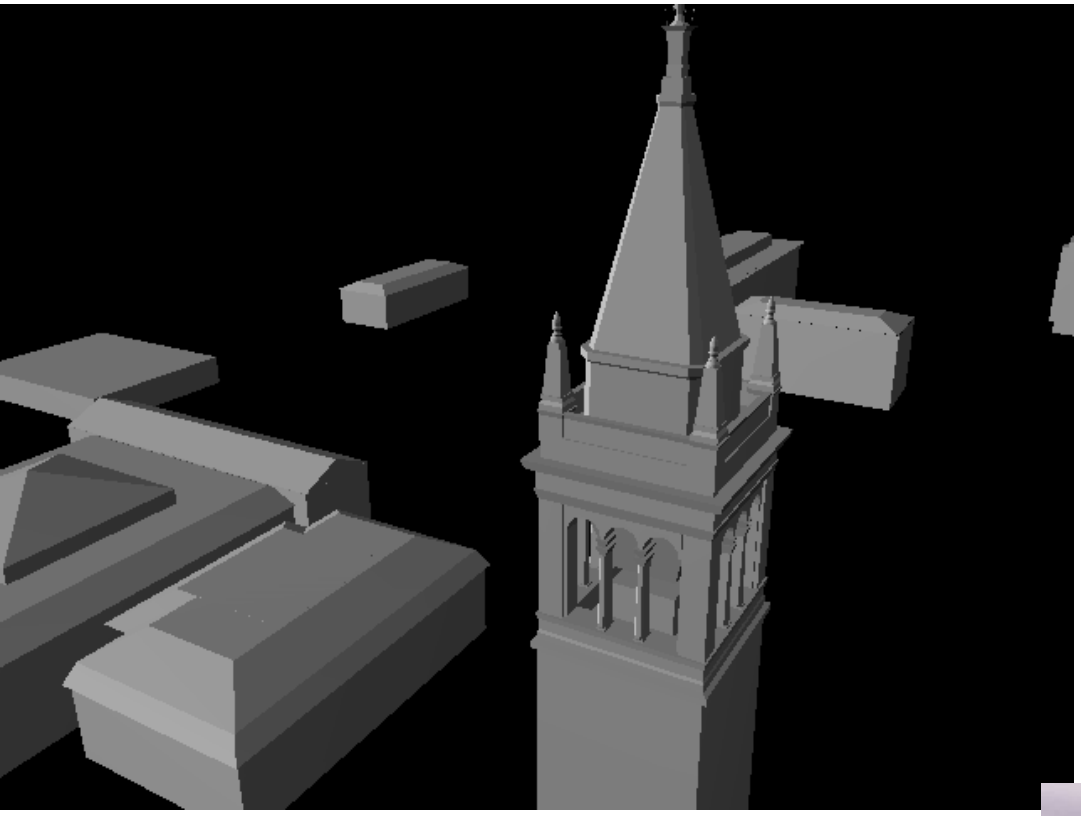
Fig. 10. Closeups for shadow filtering for 1 spp input (MC), axis-aligned filter (AAF), À-Trous wavelet filter (EAW), SURE-based filter (SBF), and our result.

Some topics…

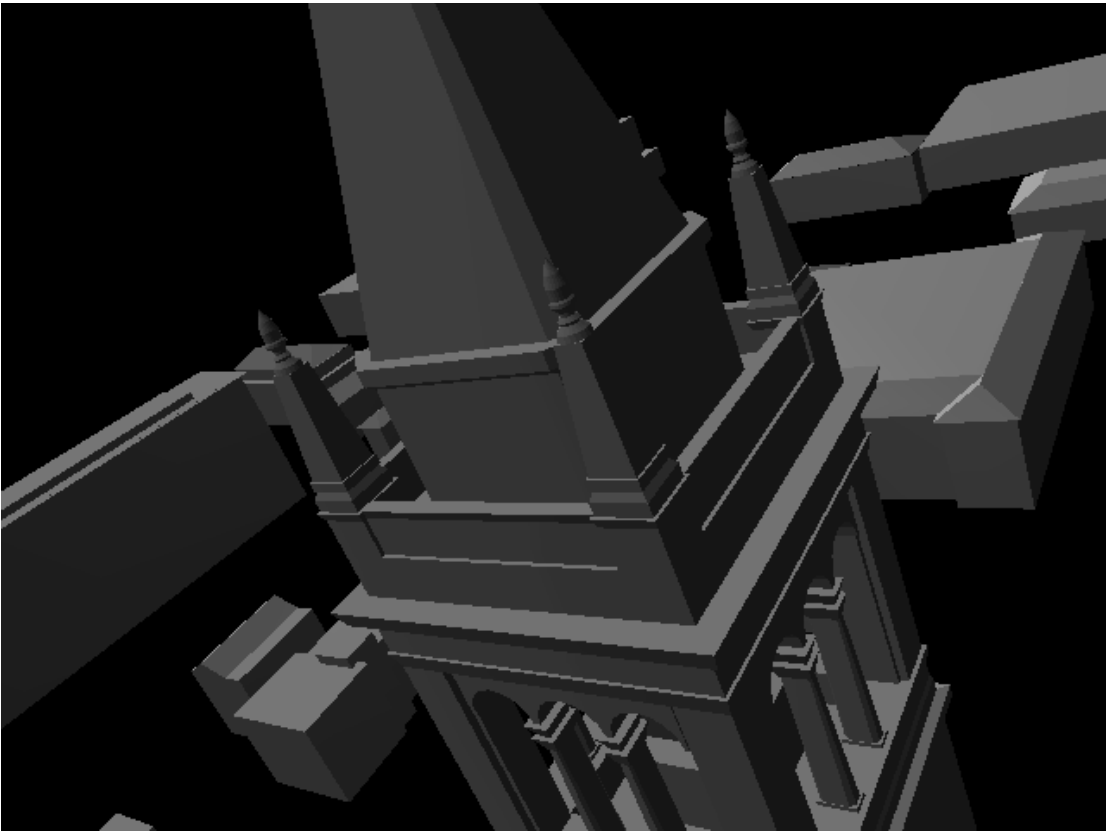
- Reduce rendering noise
  - in MCMC rendering
  - in image based rendering
  - in performance capture
- Realistic images from approximations
- Generate novel views
  - from multiview input
- Exaggerate effects
  - eg motion fields
- Reshade and relight

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



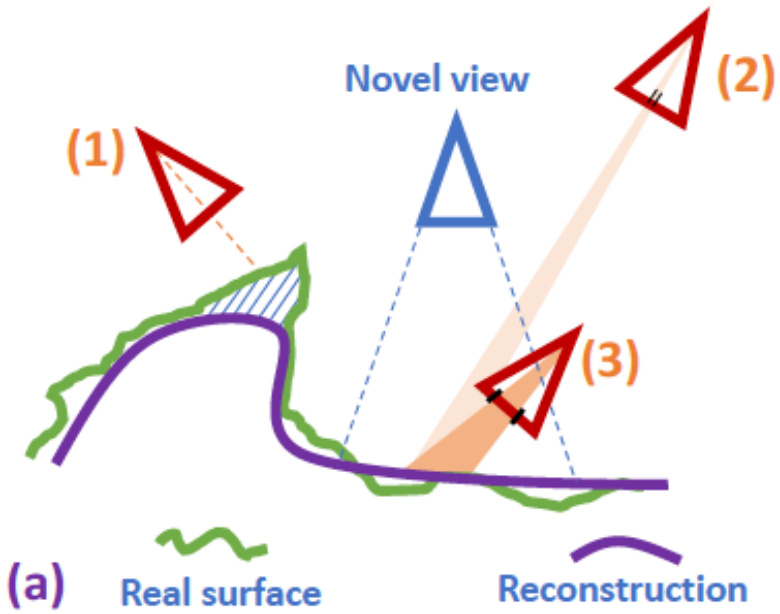
https://www.pauldebevec.com/Campanile/#movie



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

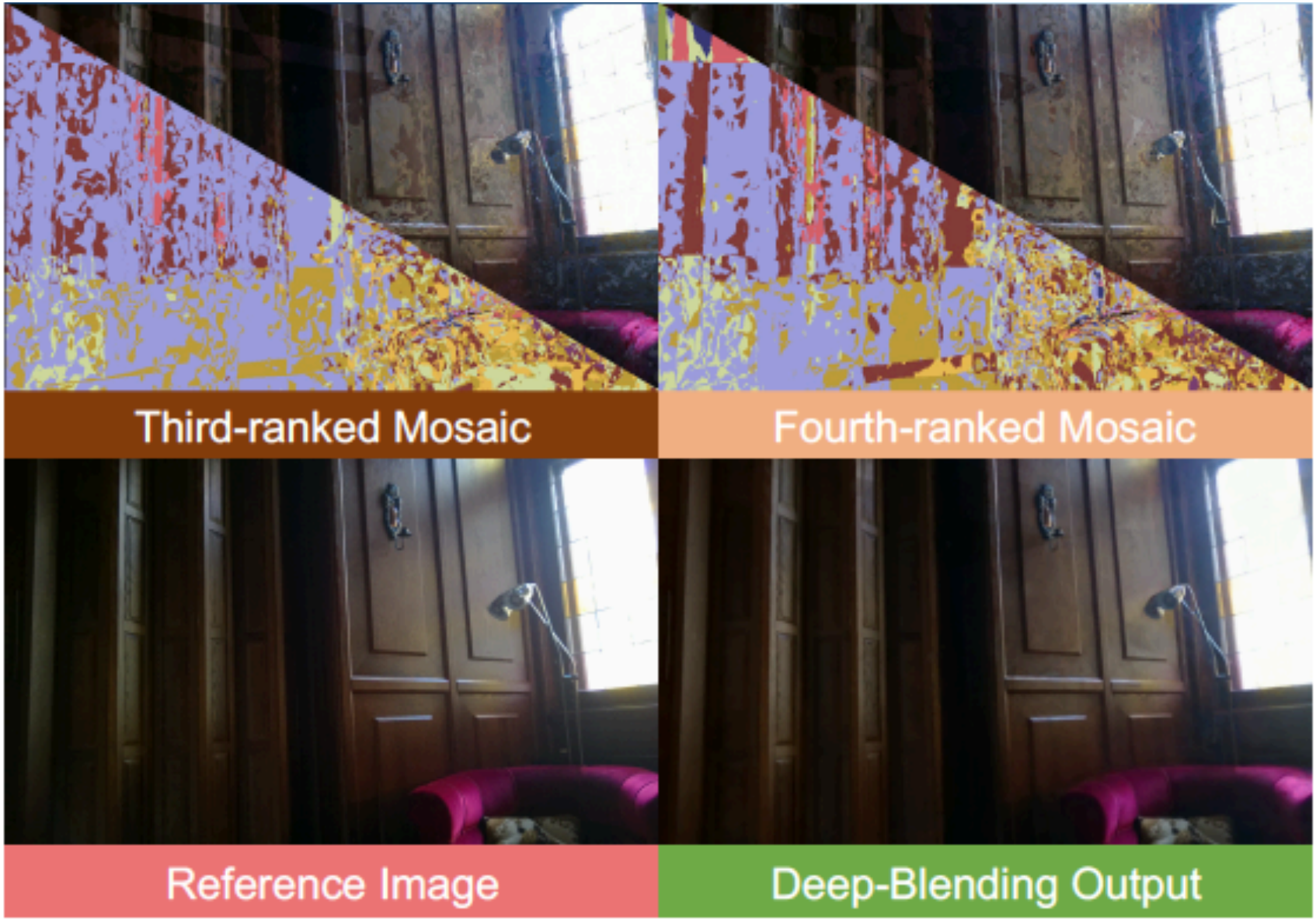
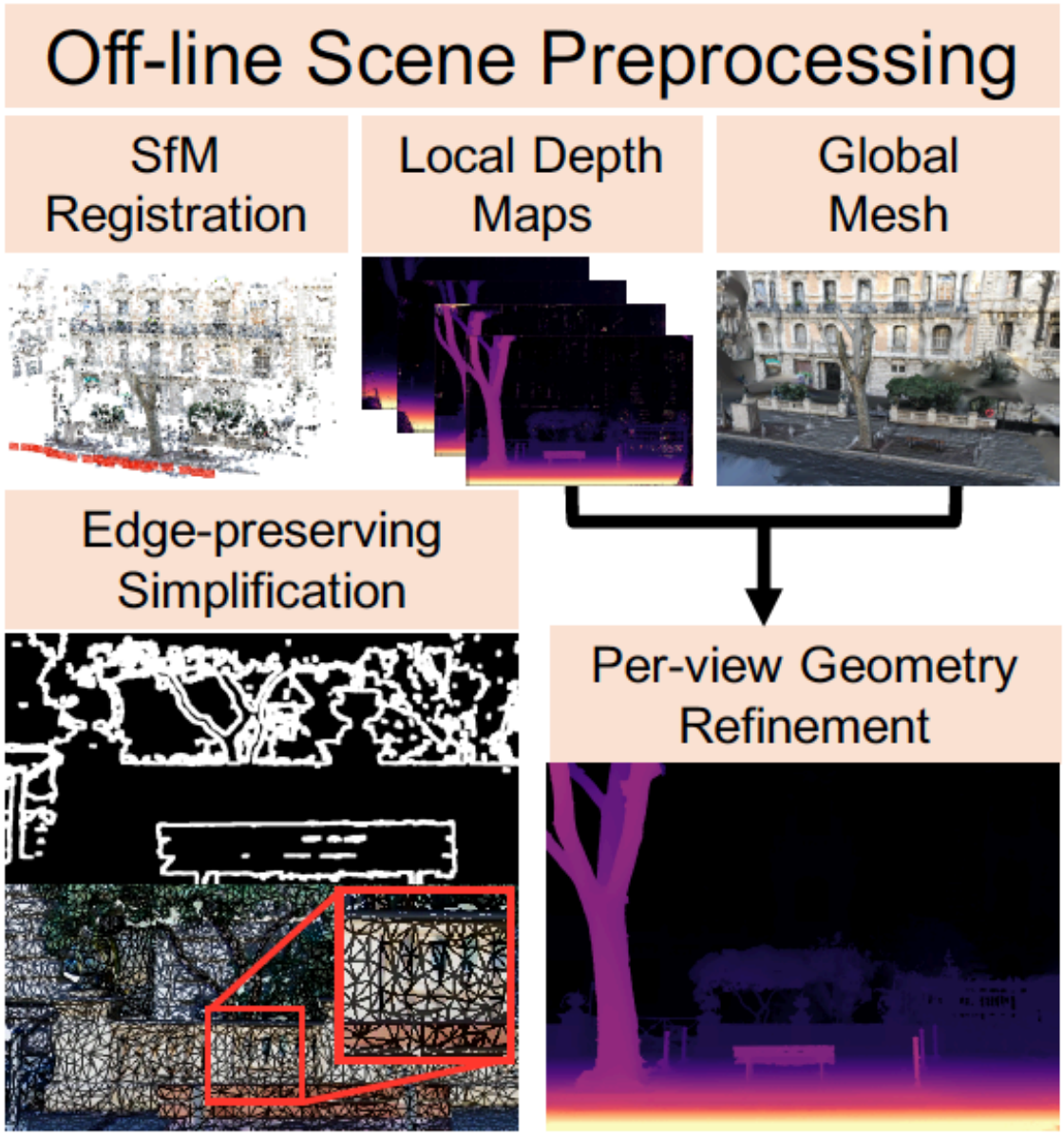
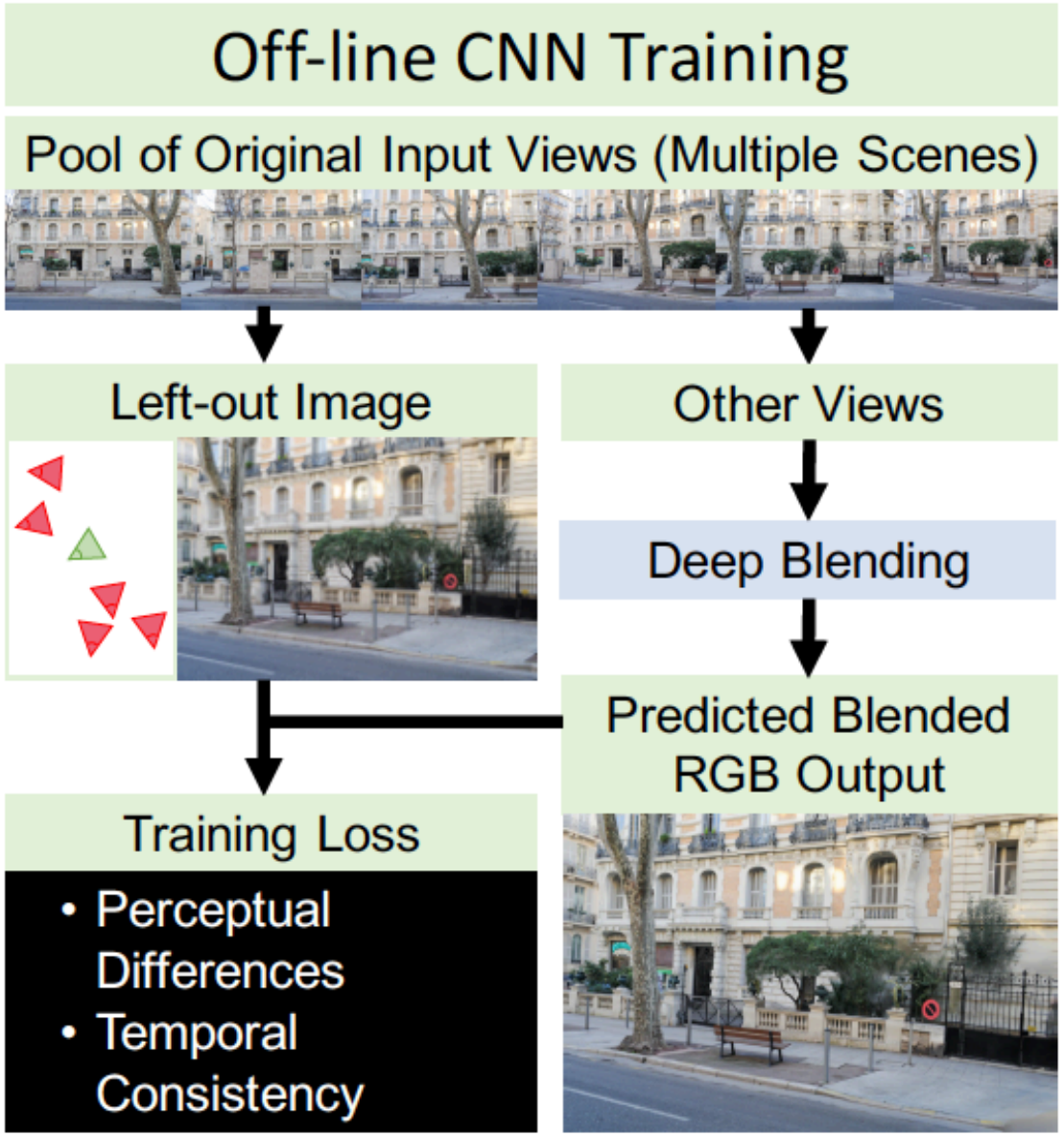


Fig. 10. Our network takes as input ranked mosaics generated from a set of warped candidate views. For each pixel, the candidates are ranked based on their expected blending contribution, and 4 color-image mosaics are formed from the top 4 rankings. Example mosaics are shown in the first two rows. The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

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

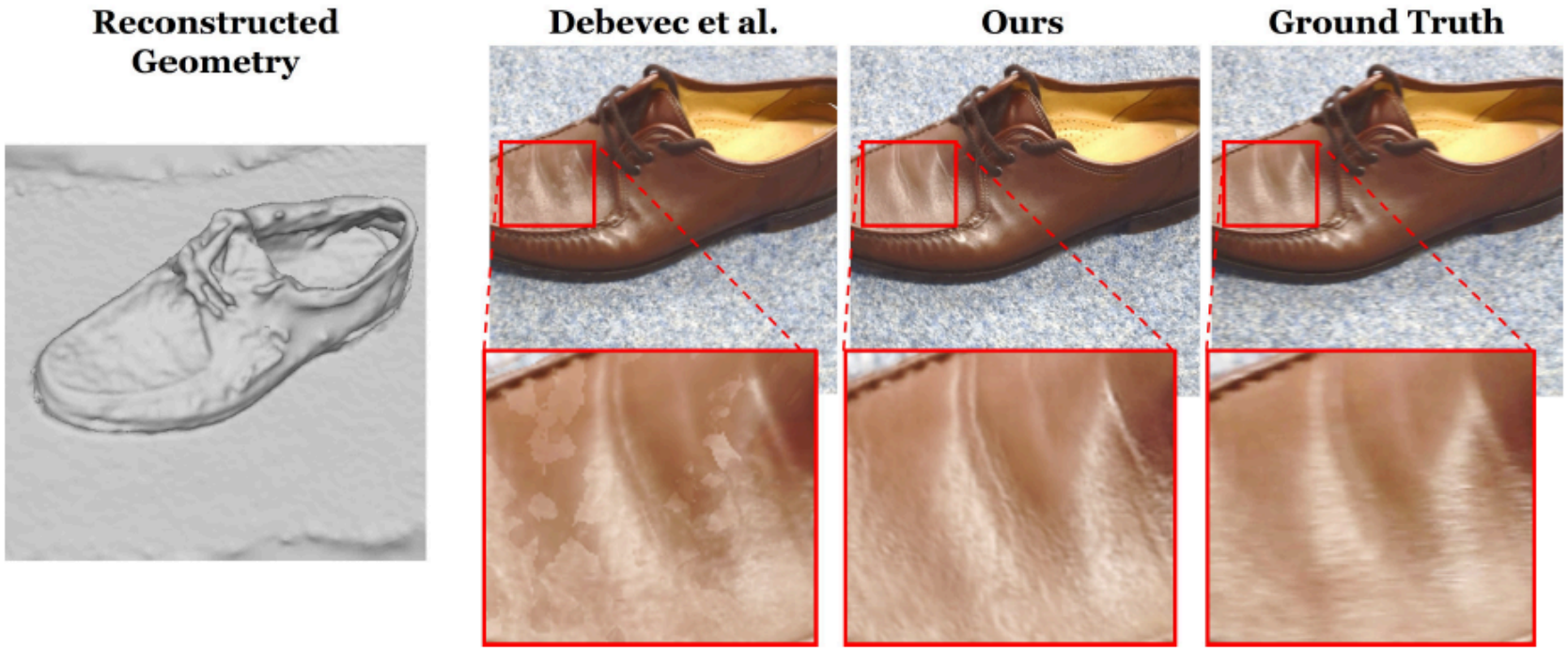


Figure 14: Image synthesis on real data: we show a comparison to the IBR technique of Debevec et al. (1998). From left to right: reconstructed geometry of the object, result of IBR, our result, and the ground truth.

Idea: predict these separately

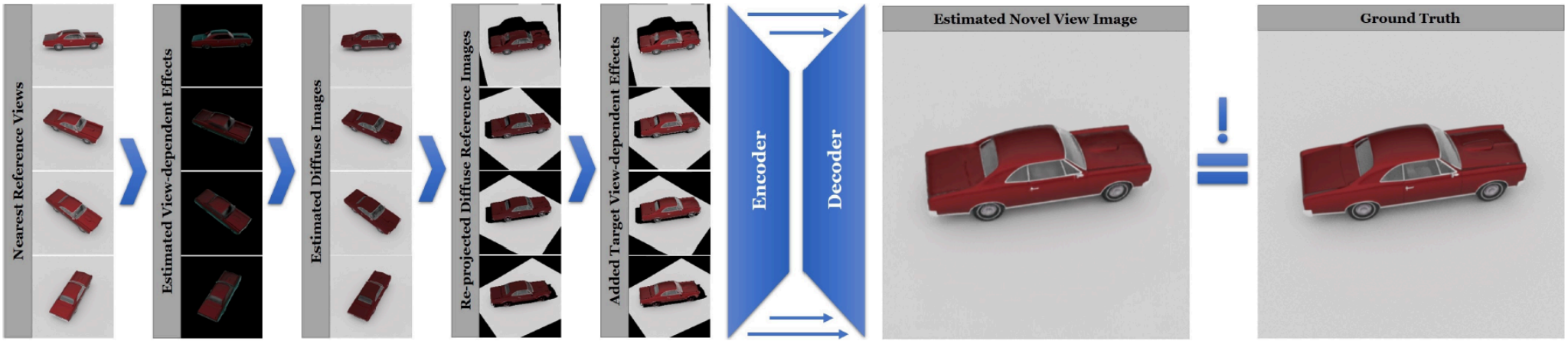


Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected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

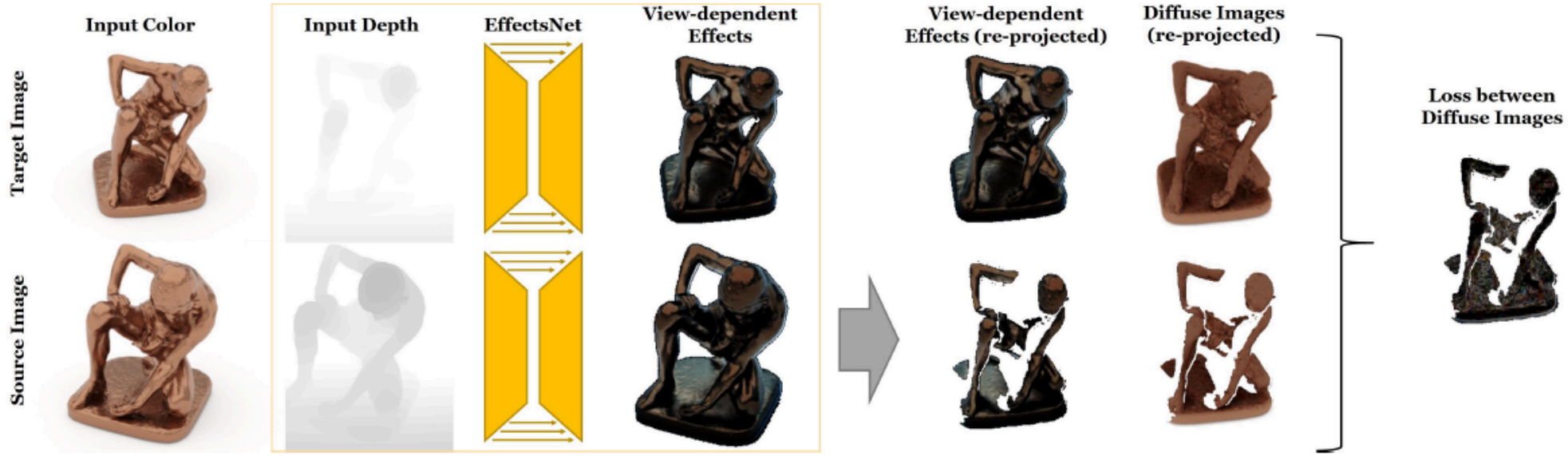


Figure 2: *EffectsNet* is trained in a self-supervised fashion. In a Siamese scheme, two random images from the training set are chosen and fed into the network to predict the view-dependent effects based on the current view and the respective depth map. After re-projecting the source image to the target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

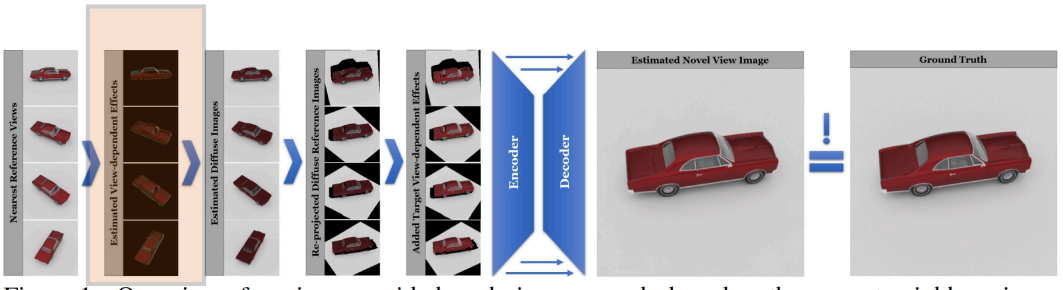


Figure 1: Overview of our image-guided rendering approach: based on the nearest neighbor views, we predict the corresponding view-dependent effects using our *EffectsNet* architecture. The view-dependent effects are subtracted from the original images to get the diffuse images that can be re-projected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

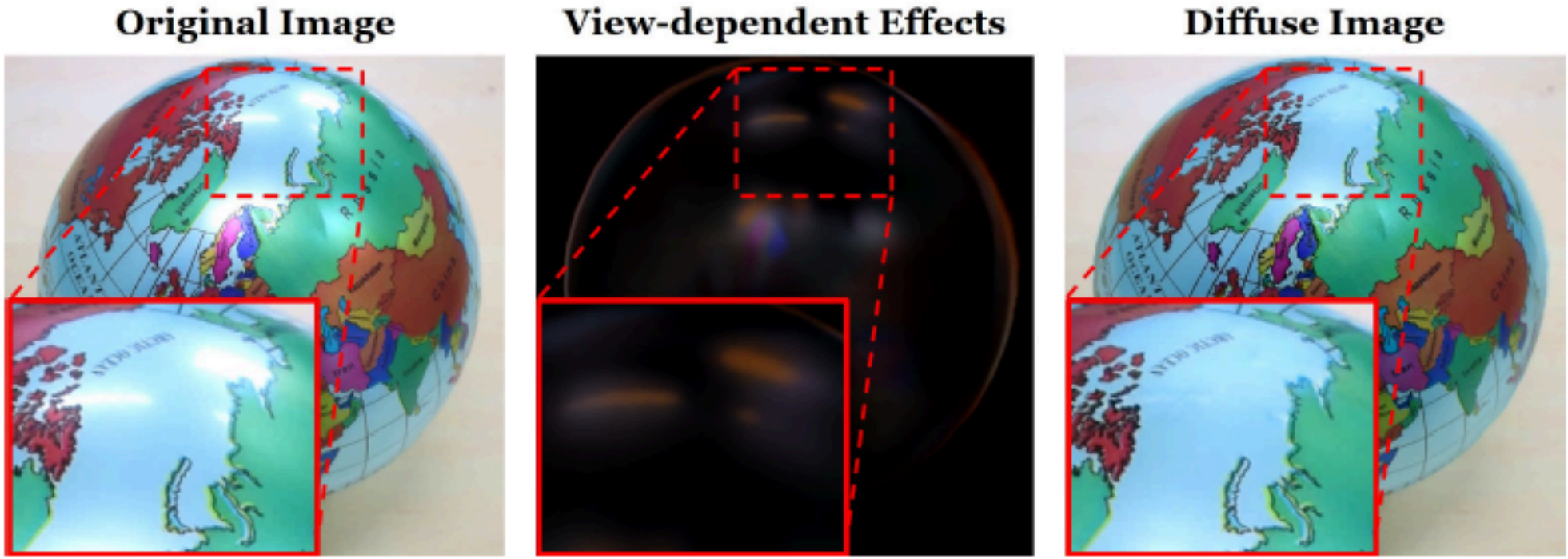


Figure 4: Prediction and removal of view-dependent effects of a highly specular real object.

Thies et al 20

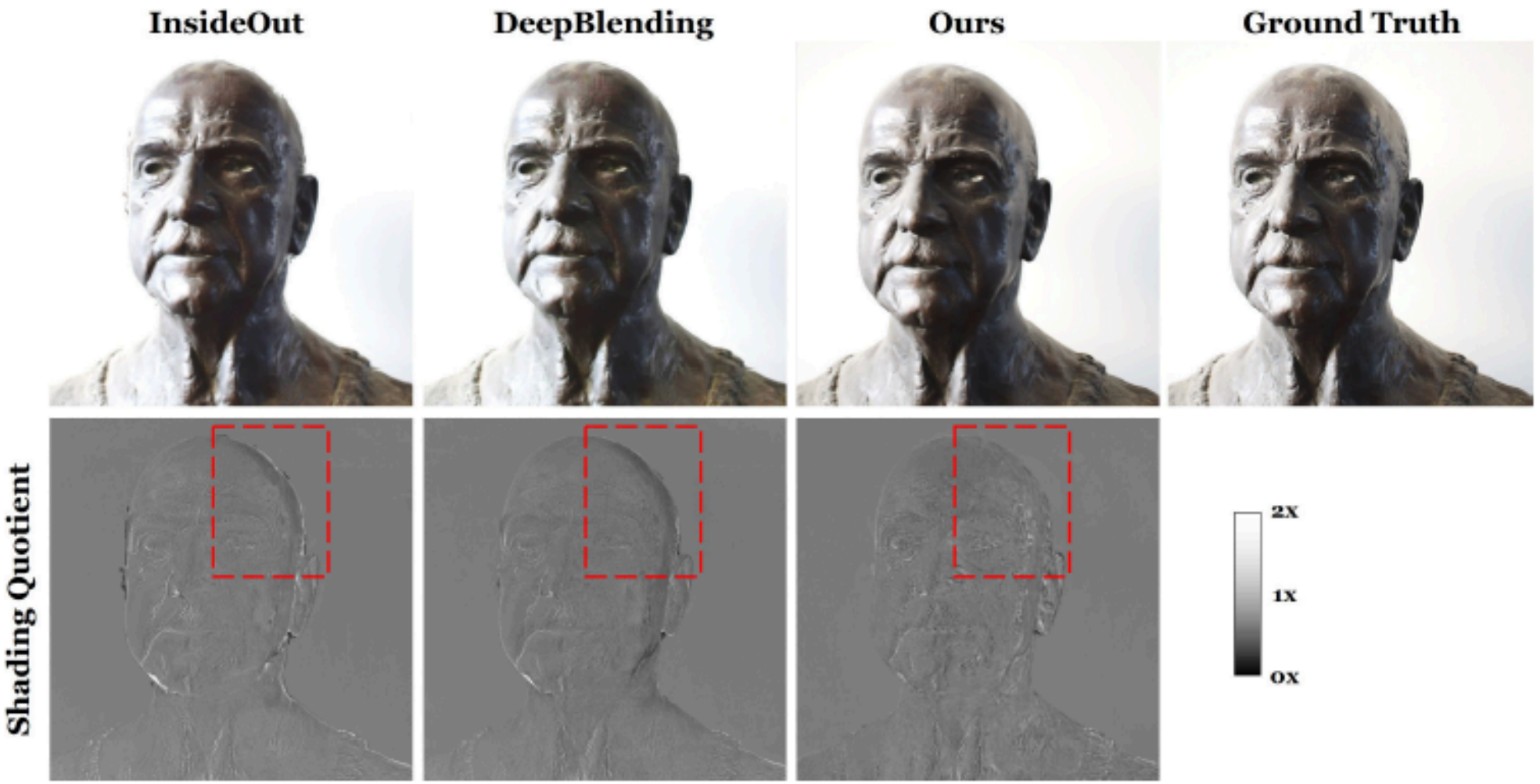


Figure 6: Comparison to the IBR method InsideOut of Hedman et al. (2016) and the learned IBR blending method DeepBlending of Hedman et al. (2018). To better show the difference in shading, we computed the quotient of the resulting image and the ground truth. A perfect reconstruction would result in a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

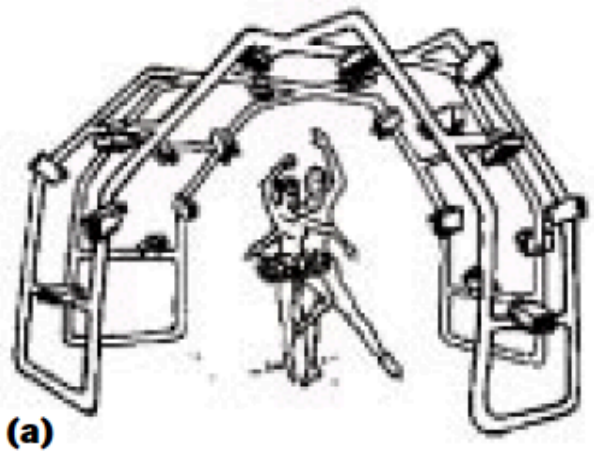
Notes and Queries

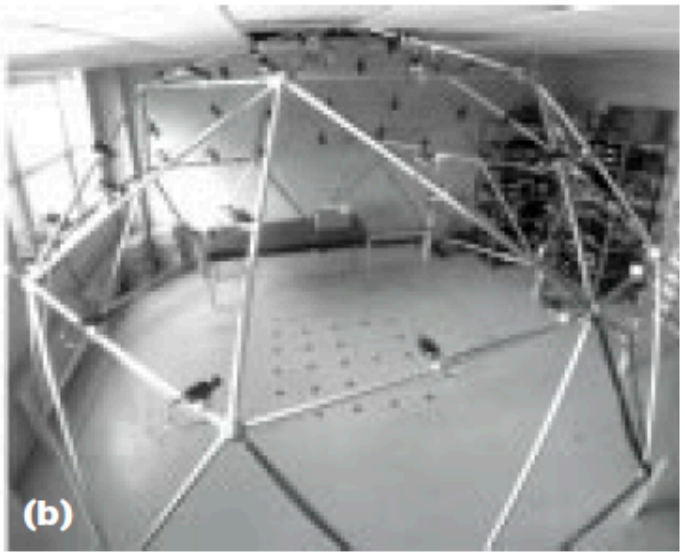
- Key idea
  - separate diffuse view prediction and view dependent components
  - notice how much preprocessing and detail before learning
    - multiple registered pix and depth maps
- There is an adversarial loss
  - Local PatchGAN loss
    - from pix2pix (Isola, 16)
      - qv
    - useful trick

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)



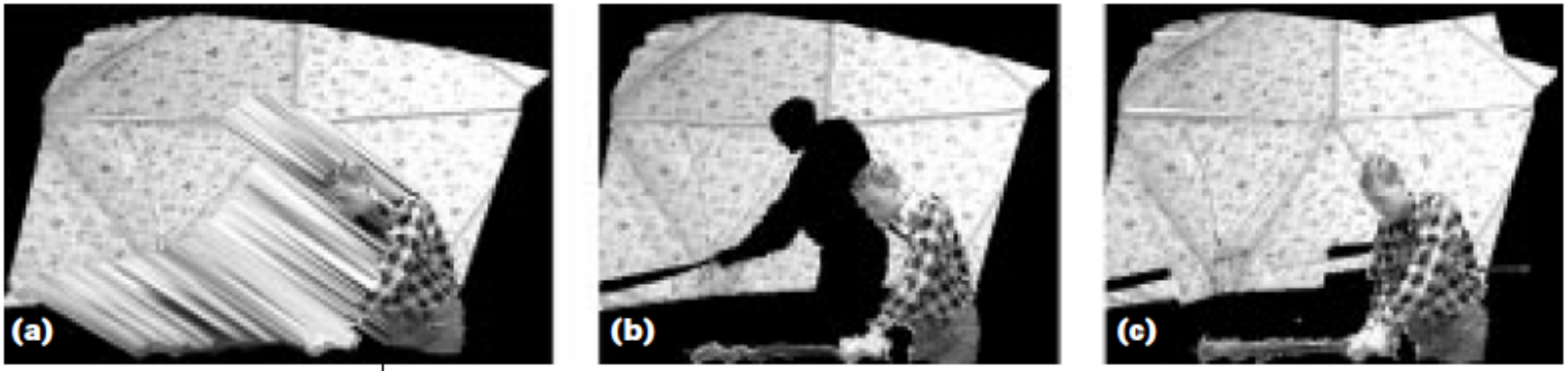


Kanade et al 97

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

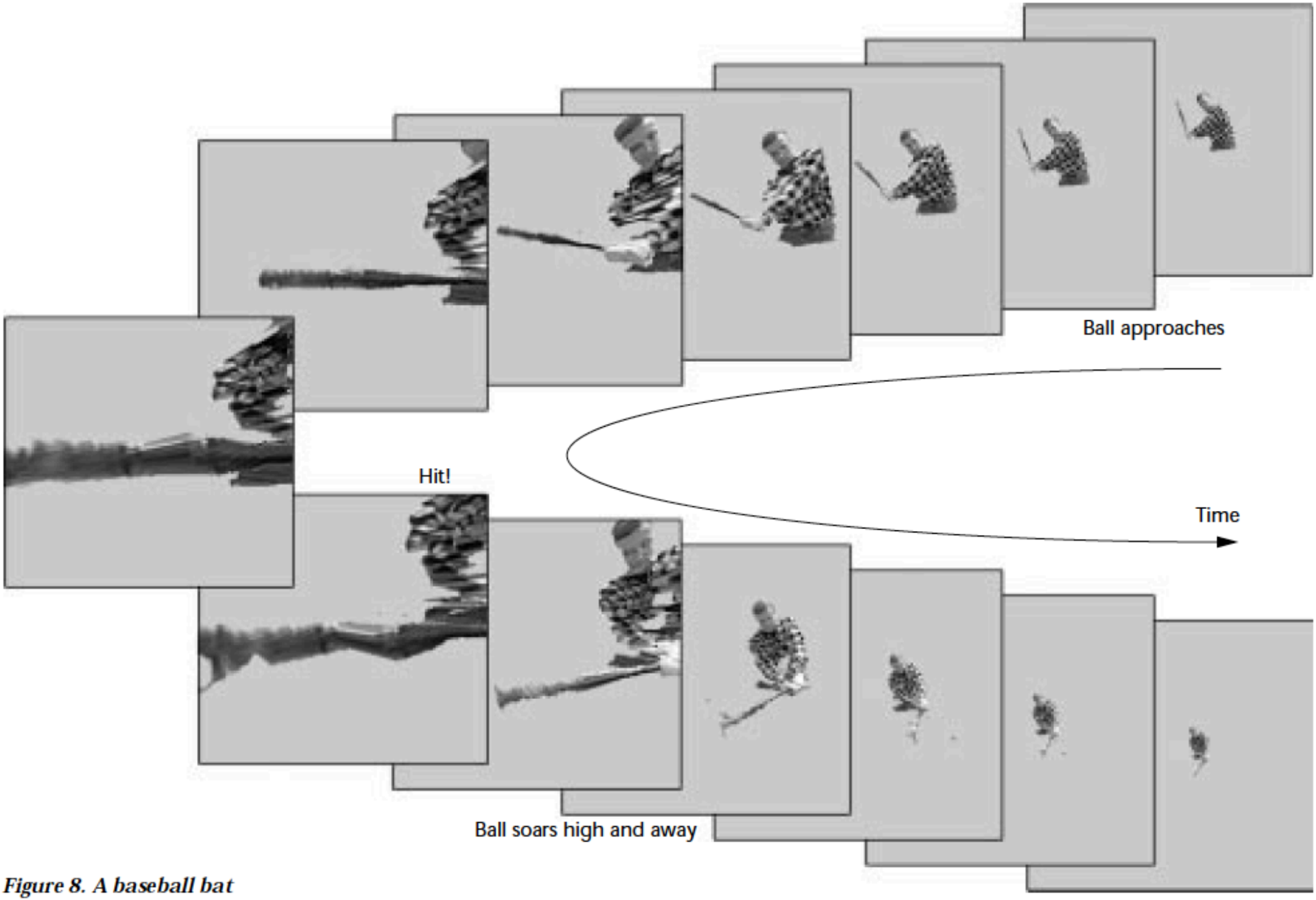


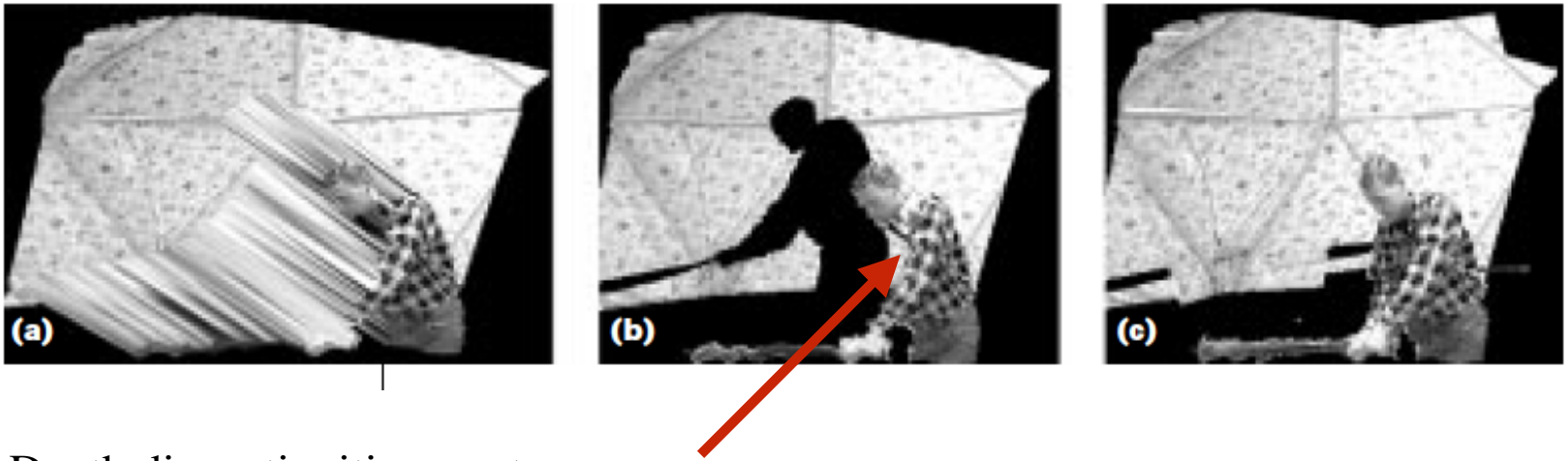
Figure 8. A baseball bat swing from the baseball's point of view.

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

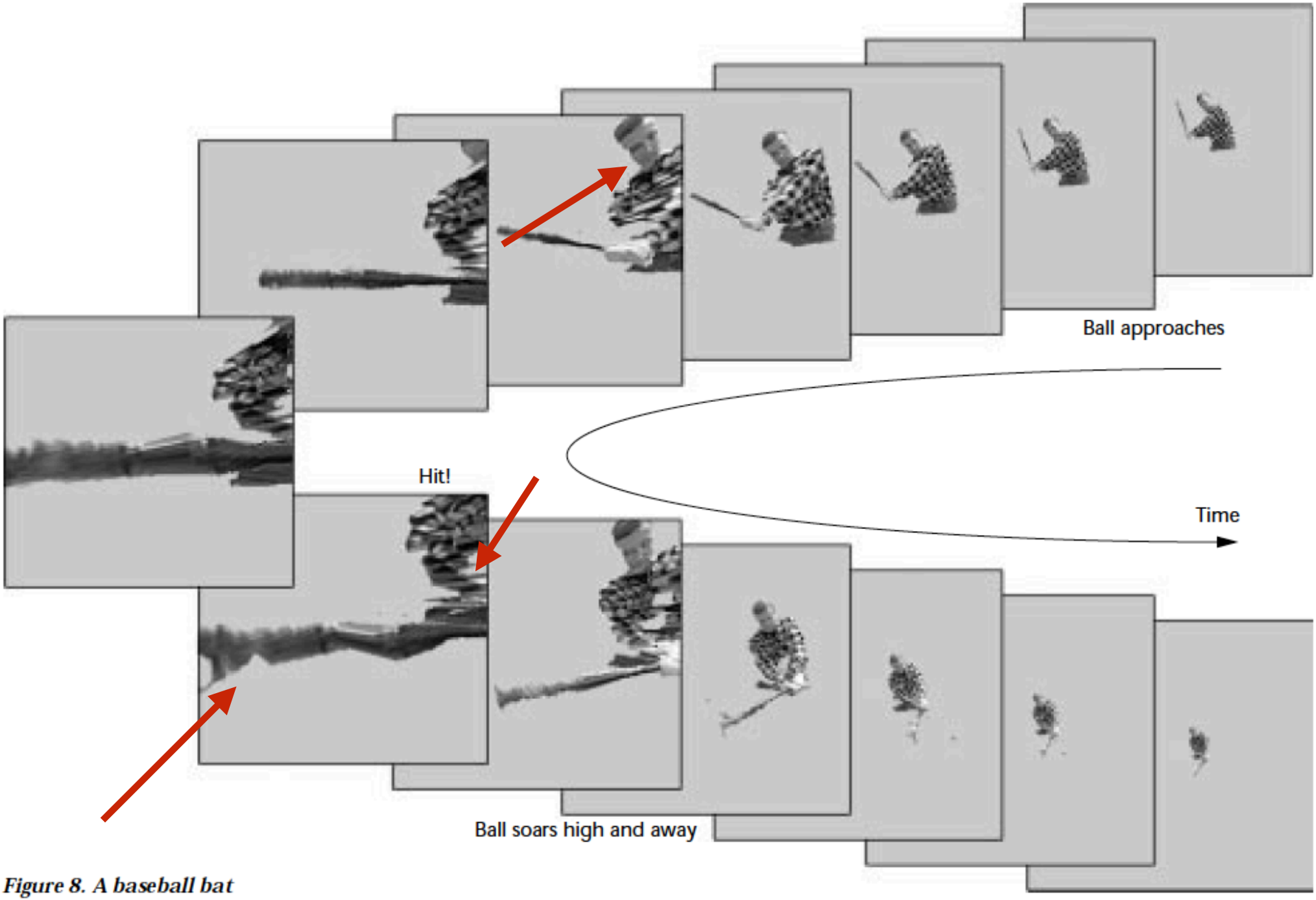
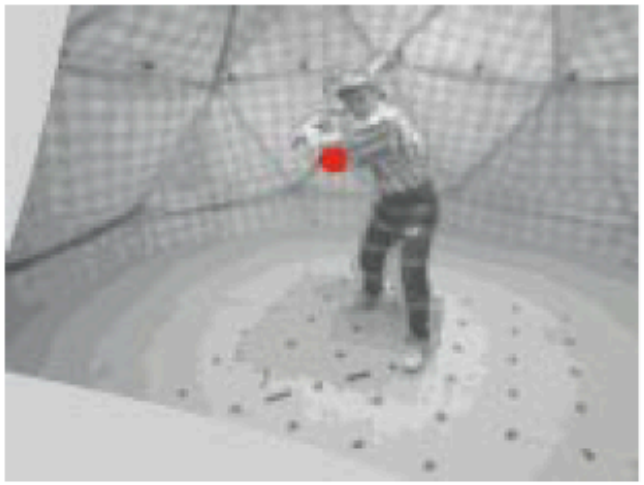
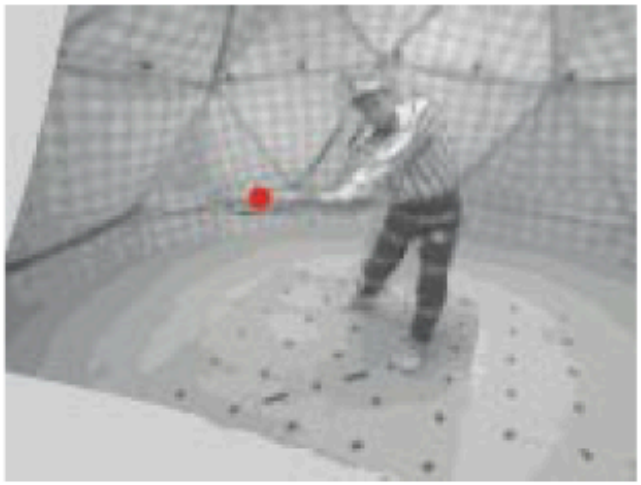
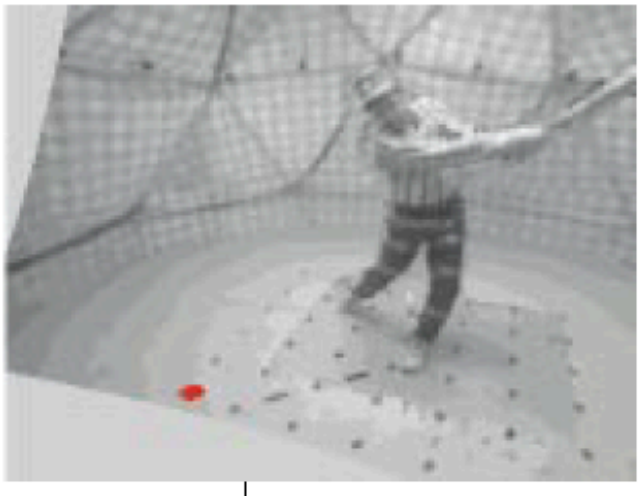


Figure 8. A baseball bat swing from the baseball's point of view.

Kanade et al 97

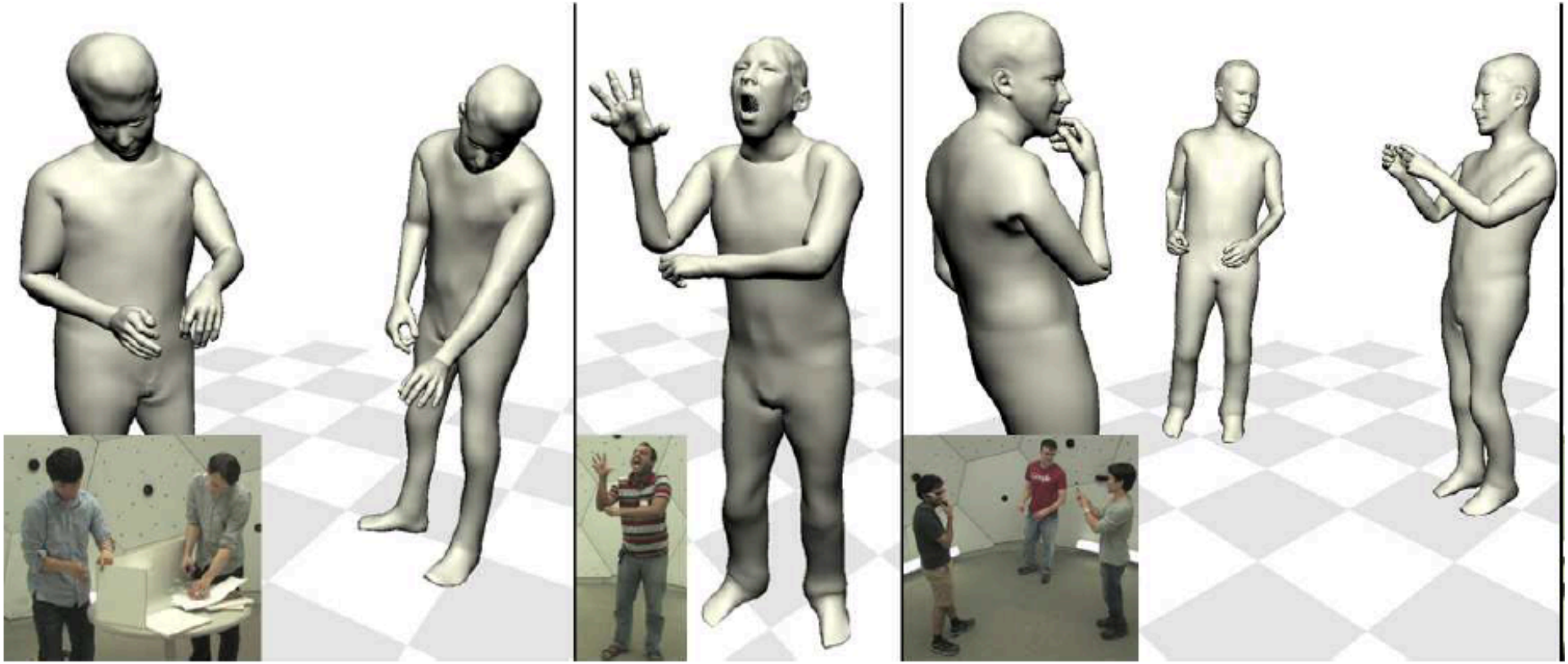
Kanade et al 97

Kanade et al 97

Fixes

- Cameras:
  - more, faster, higher resolution, better synchronized
- Reconstruction algorithms:
  - high resolution multiview stereo reconstructions
- Body models:
  - skinned parametric body/hand/face models



Joo et al 18

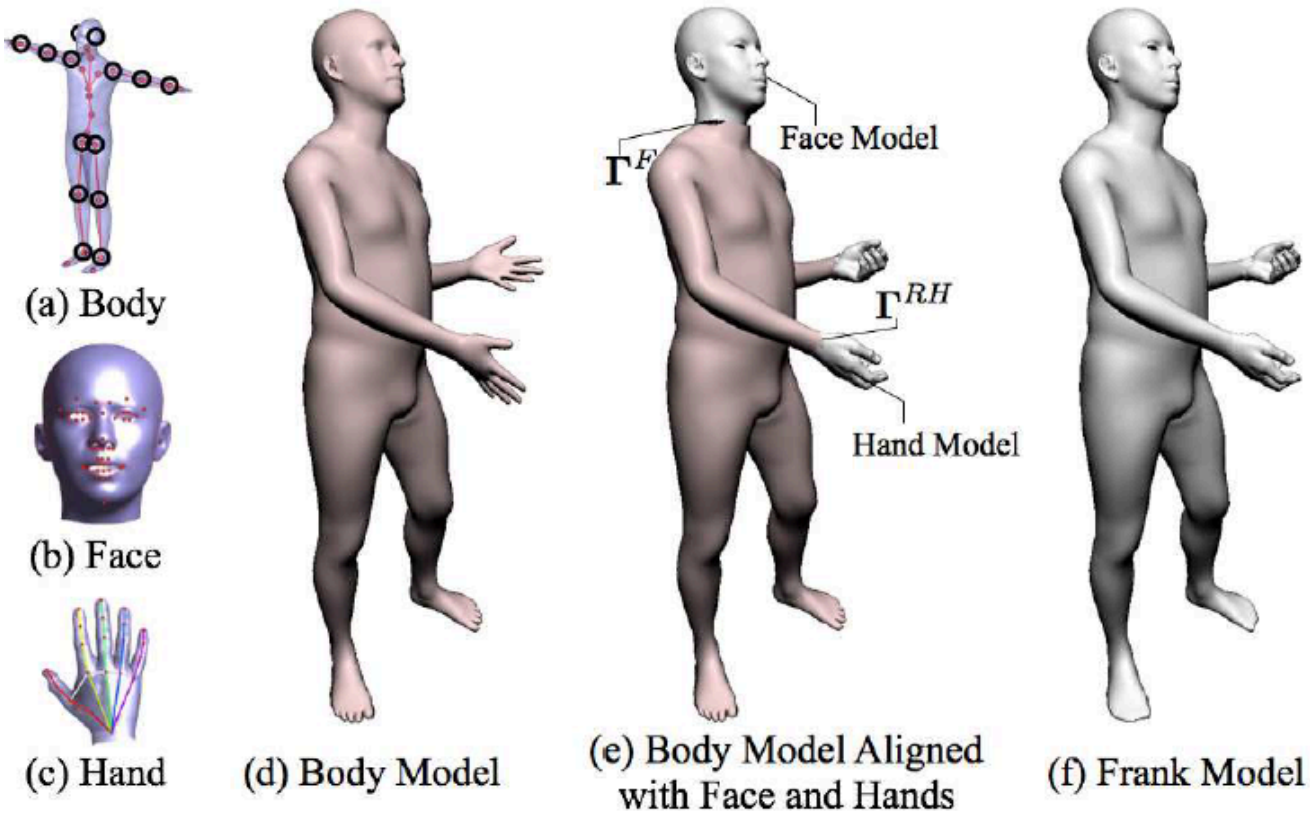


Figure 2: Part models and the Frank model. (a) The body model [34]; (b) the face model [15]; and (c) a hand rig. In (a-c), the red dots have corresponding 3D keypoints reconstructed by detectors; (d) Body only model; (e) Face and hand models substitute the corresponding parts of the body model. Alignments are ensured by $\boldsymbol{\Gamma}$s; and (f) The blending matrix $\mathbf{C}$ is applied to produce a seamless mesh.

Joo et al 18

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

Capture System with Performer

Computed Geometry

Computed Reconstruction with Texture

Relightable Volumetric Videos

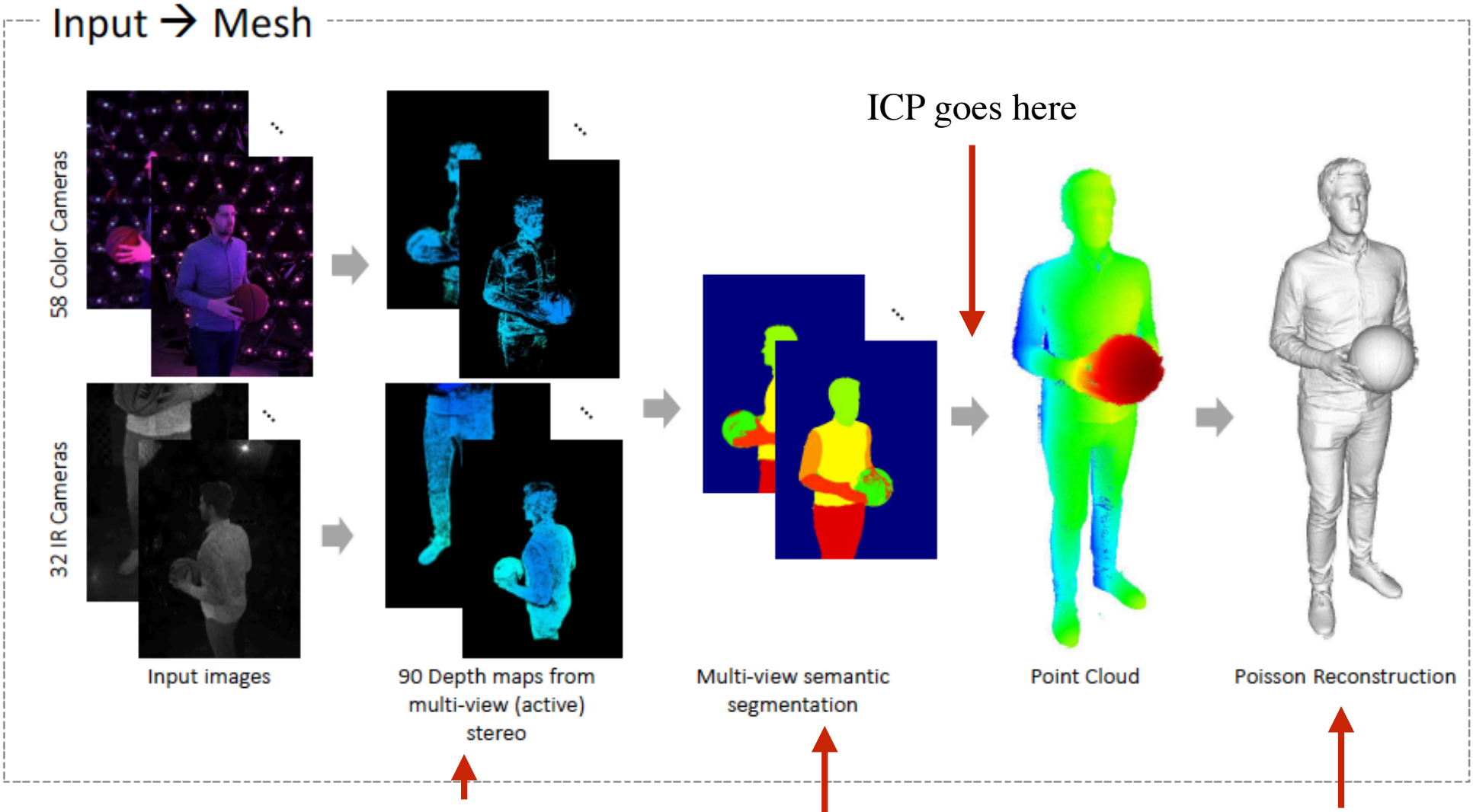


Fig. 8. The Relightables Pipeline (Part 1). First, raw images are used to reconstructed a high quality 3D model.

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

Guo et al 19

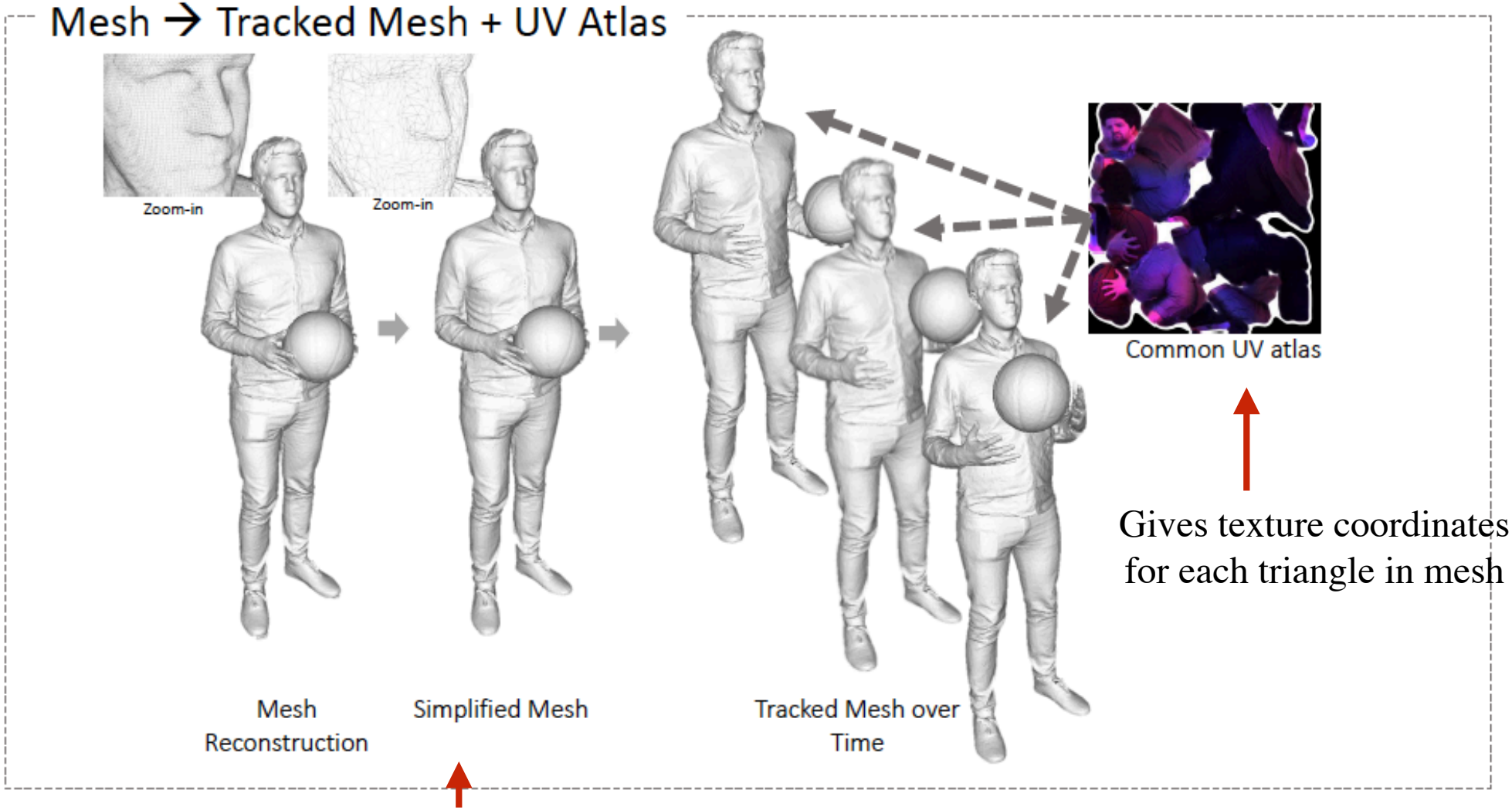


Fig. 9. The Relightables Pipeline (Part 2). This mesh then gets downsampled, tracked over time and parameterized.

Simplified by another standard procedure

Guo et al 19

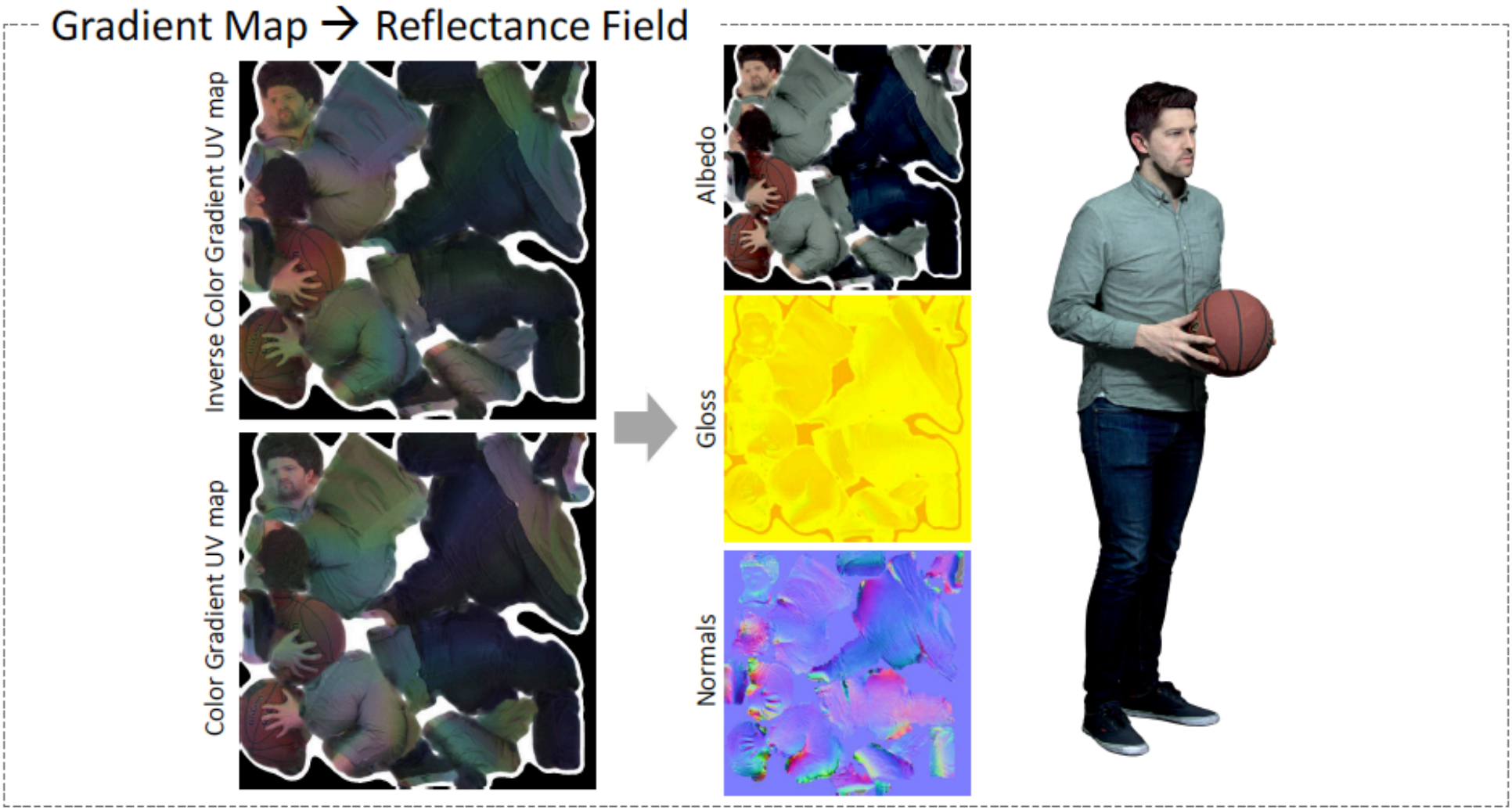


Fig. 10. The Relightables Pipeline (Part 3). Finally reflectance maps are inferred from two gradient illumination conditions.

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

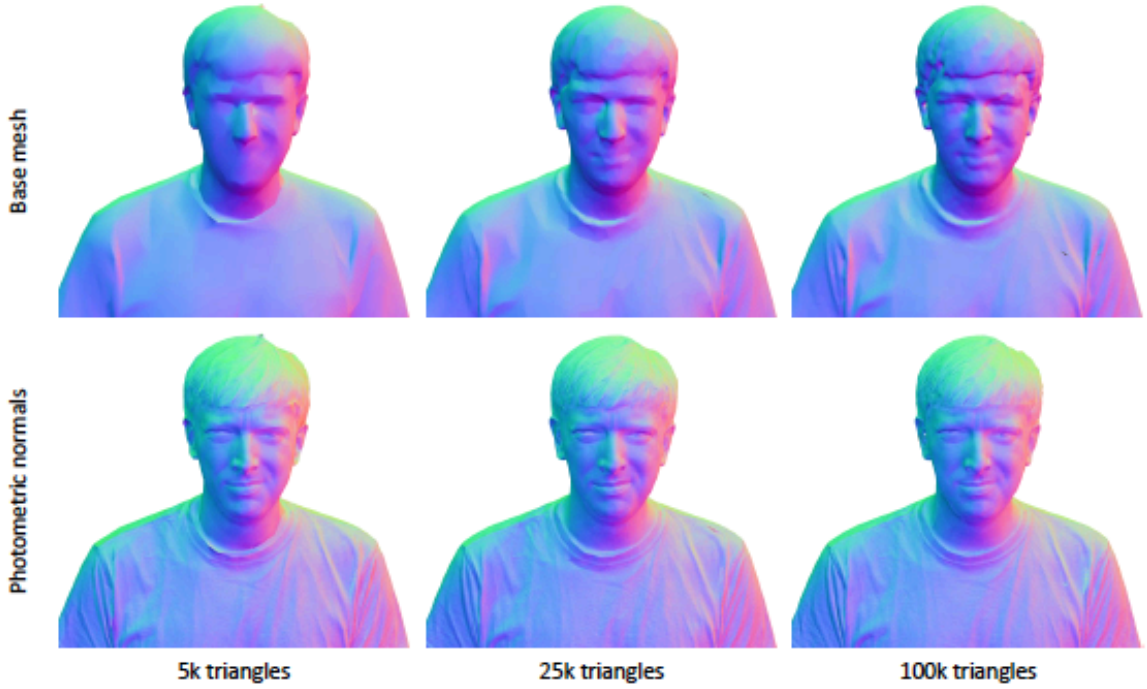
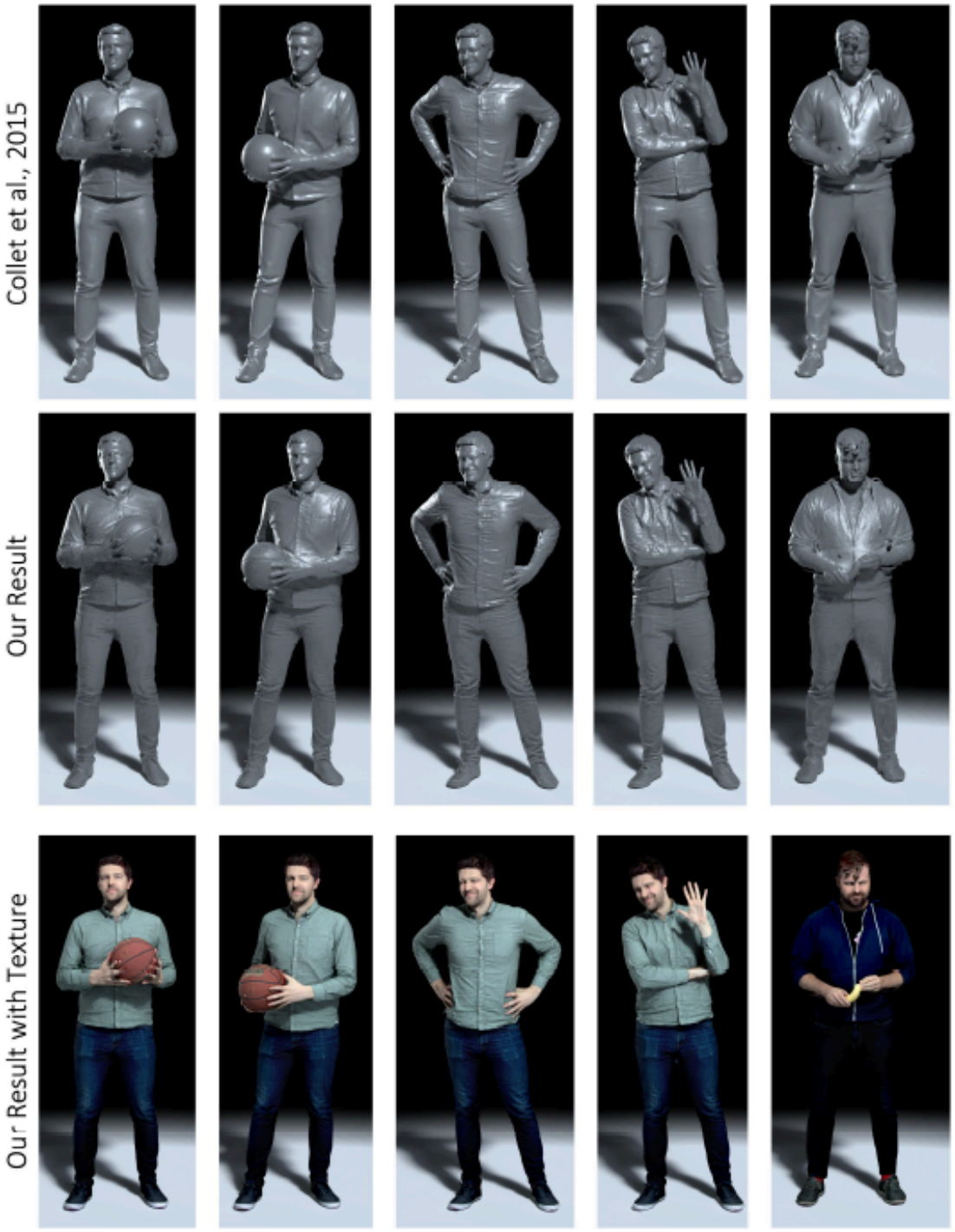


Fig. 17. A comparison of different decimated meshes (base mesh and photometric normals visualization) using 5k, 25k, and 100k triangles respectively.

- 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



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.

Guo et al 19

There are problems in all systems…

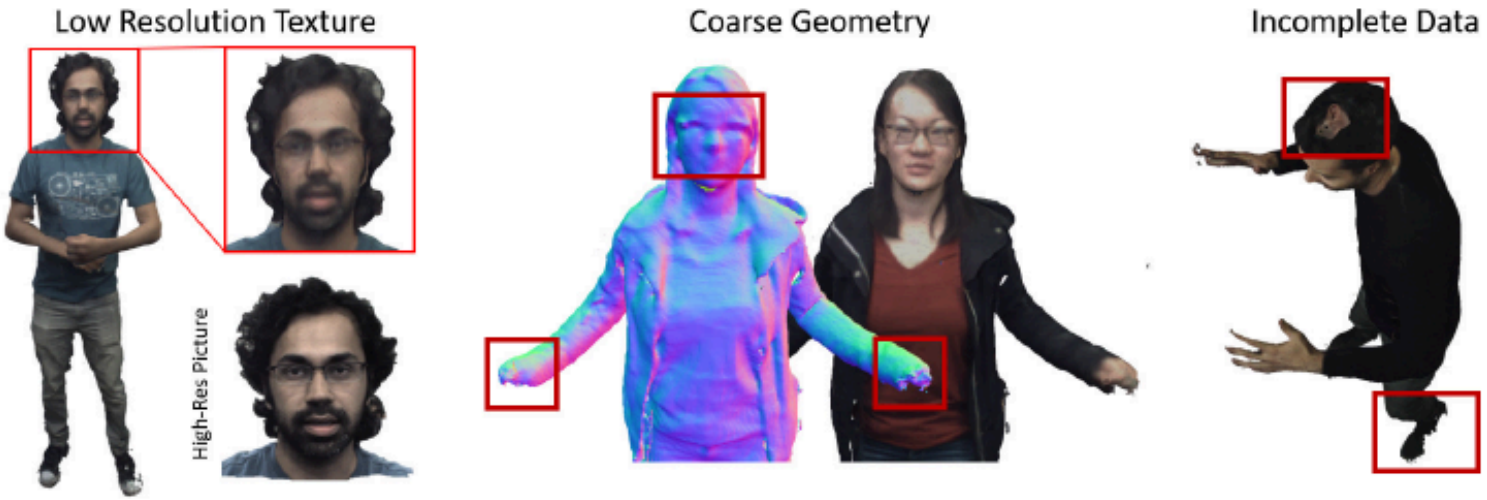


Fig. 2. Limitations of state of the art, real-time performance capture systems. Left: low resolution textures where the final rendering does not resemble a high quality picture of the subject. Middle: coarse geometry leads to overly smooth surfaces where important details such as glasses are lost. This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

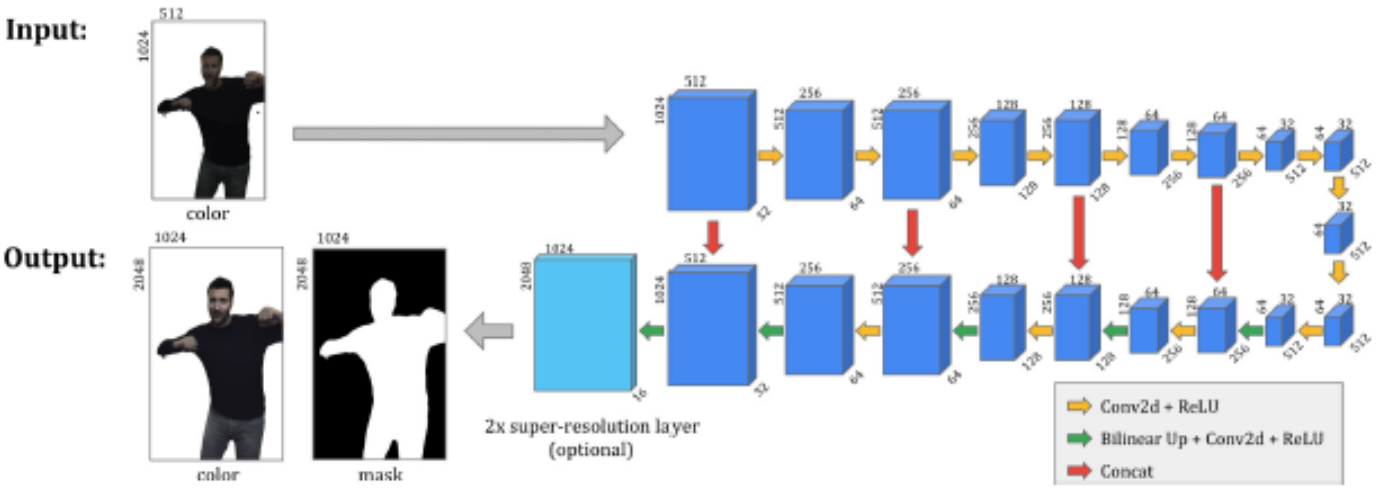


Fig. 5. LookinGood's fully convolutional deep architecture. We train the model for both left and right view that simulate a VR or AR headset. The architecture takes as input a low resolution image and produces a high quality rendering together with a foreground segmentation mask.

Martin-Brualla et al, 18

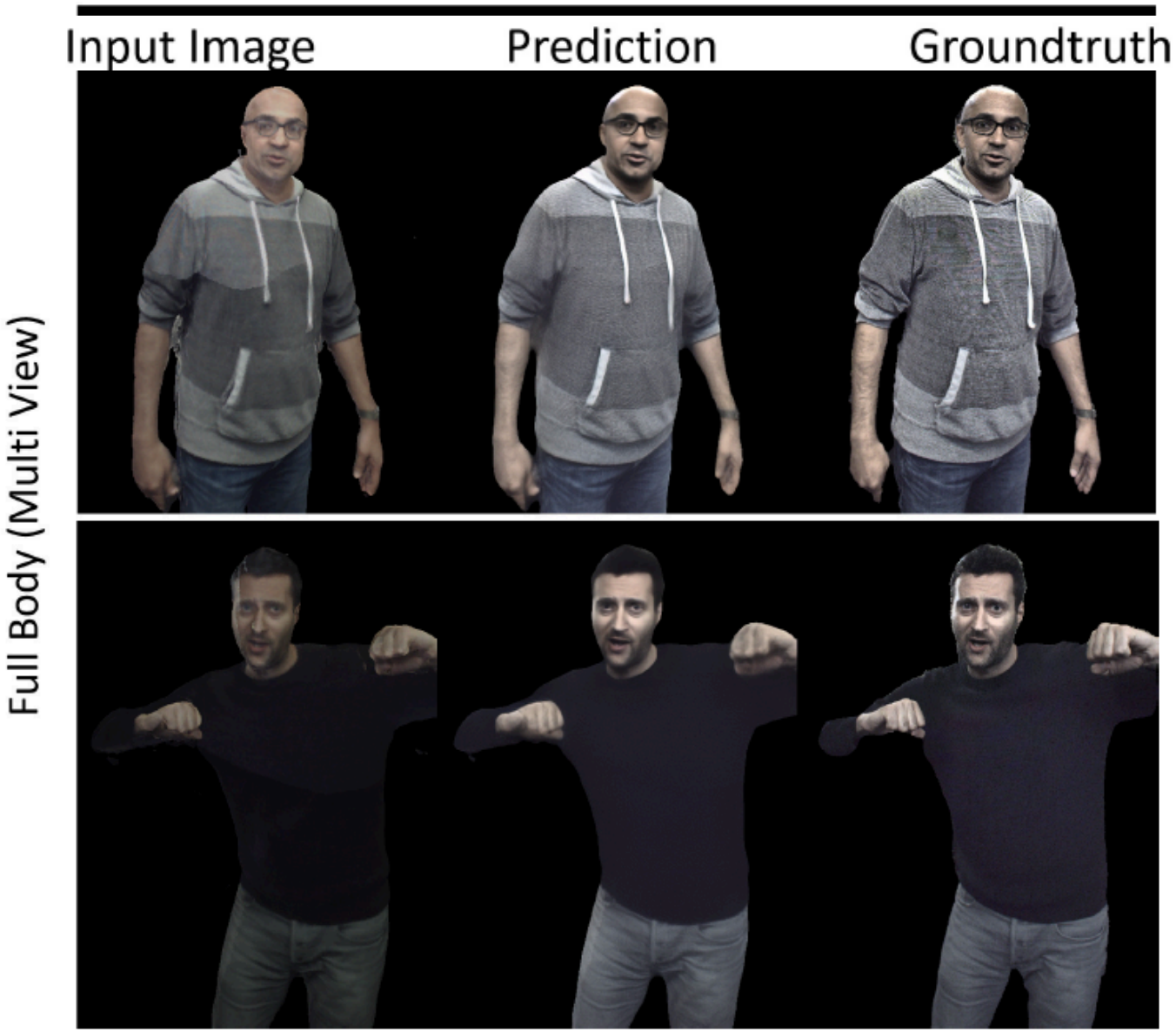


Fig. 7. Generalization on new sequences. We show here some results on known participant but unseen sequences. Notice how the method is able to in-paint missing areas correctly in the single camera case (top rows). Full body results show an improved quality and robustness to imprecision in the groundtruth mask (third row, right). The method also recovers from color and geometry inconsistencies (forth row, left).

Martin-Brualla et al, 18

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.

Martin-Brualla et al, 18

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

Realistic images from approximations

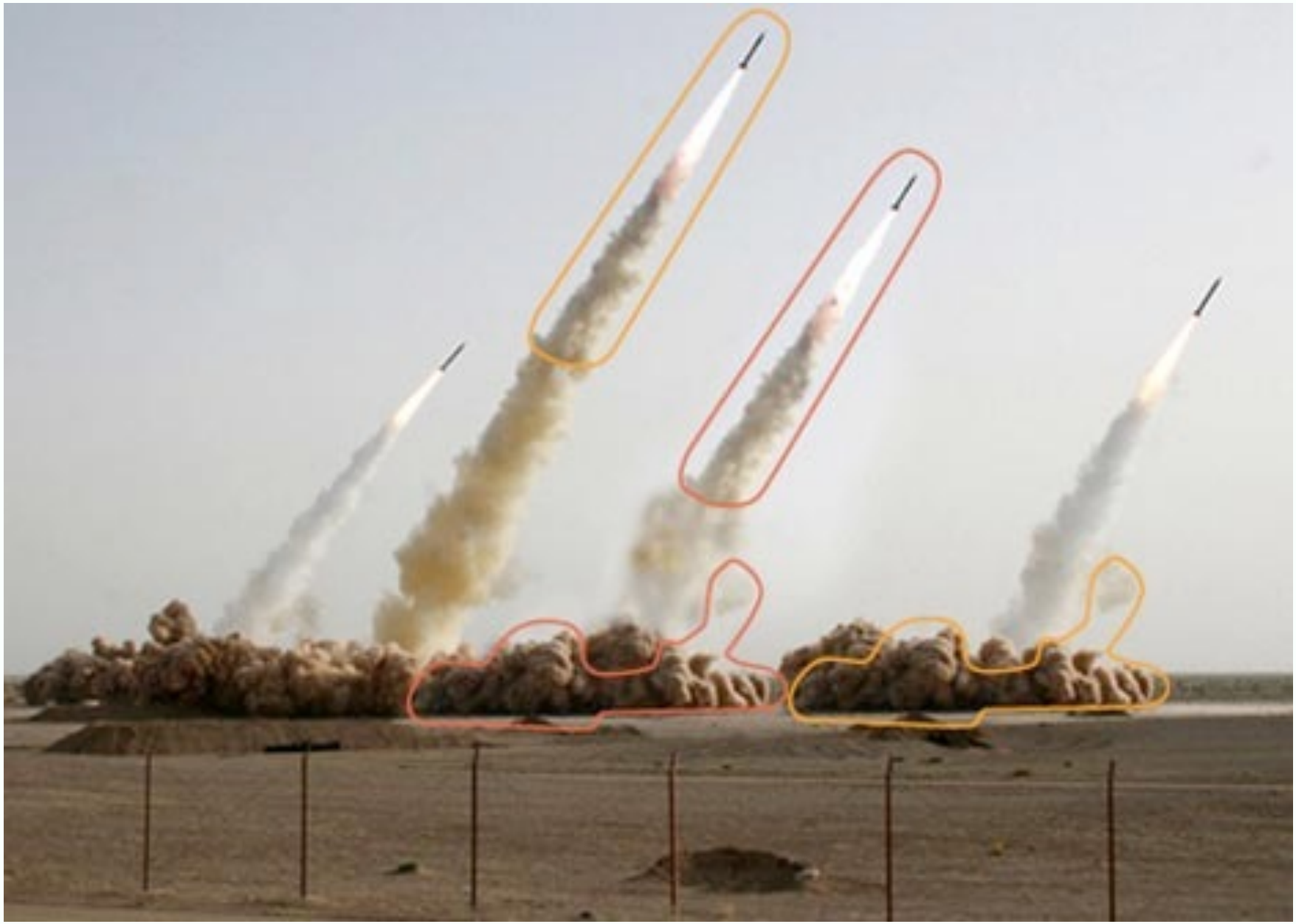
- Idea:
  - from approximate image
    - eg semantic segmenter map
  - produce “realistic” image
- In what way realistic?
  - Fools adversary
  - Obtain strong LPIPS score
    - compare deep features of local patches to those of real images (Zhang et. al)
  - etc
- Why?
  - Rendering model for backgrounds, scenes, etc.
  - Controllable to some degree
  - Possible source of training data?

Texture

CS 419

Slides by Ali Farhadi

Texture scandals!!



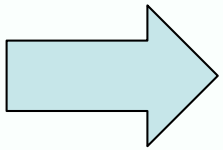
Bush campaign digitally altered TV ad

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

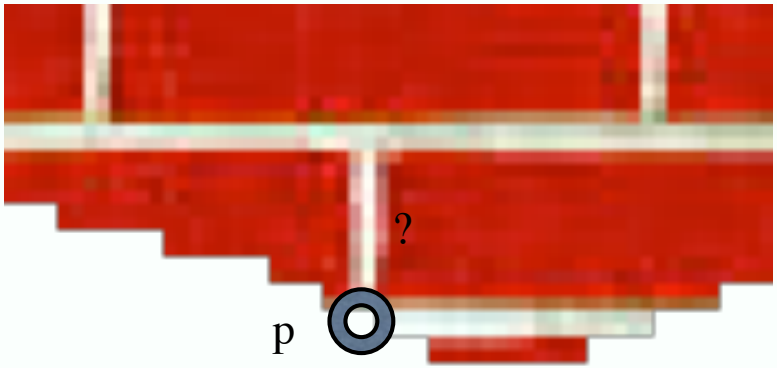
- Nearest neighbors
  - again and again and again
- Dynamic programming
  - likely new; we'll use this again, too

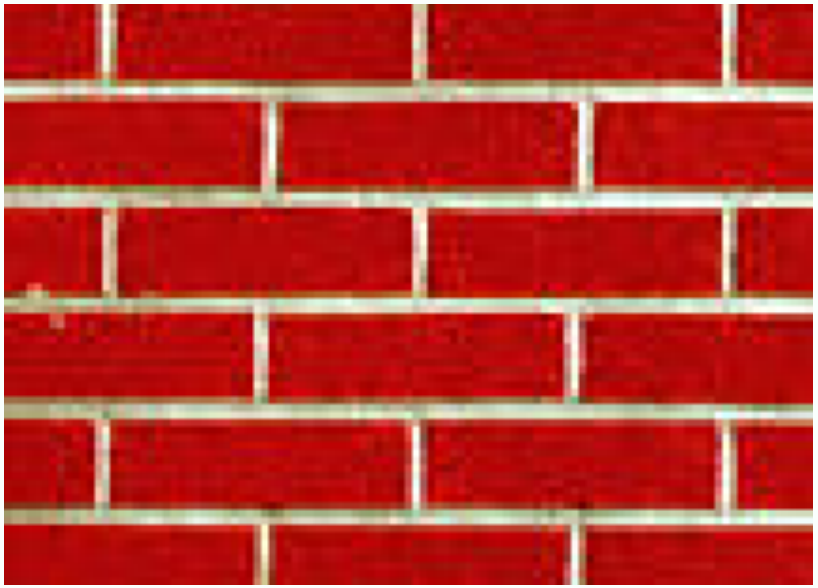
Texture Synthesis



Efros & Leung ICCV99

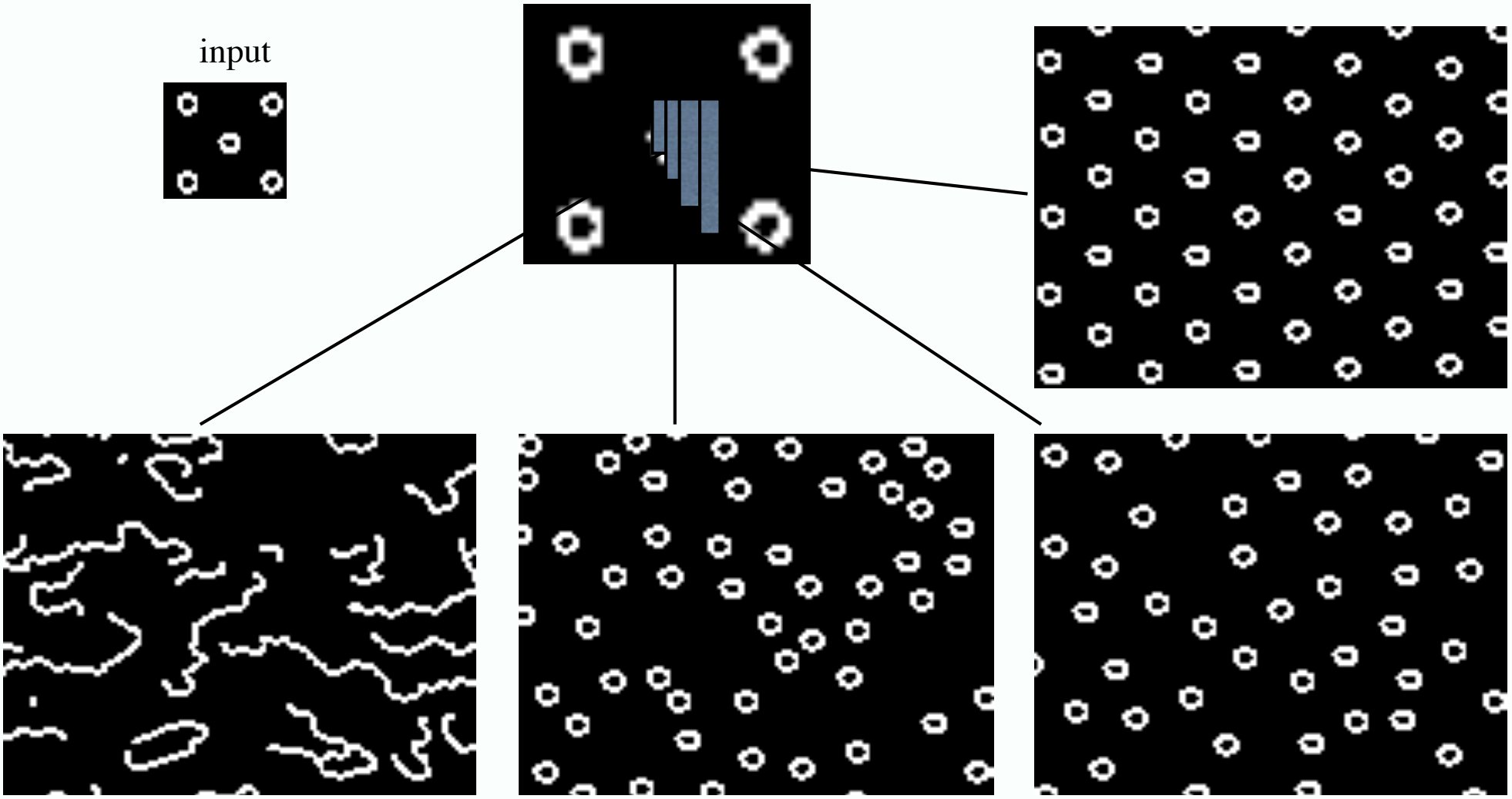
How to paint this pixel?





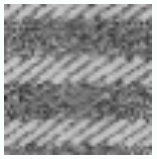
Input texture

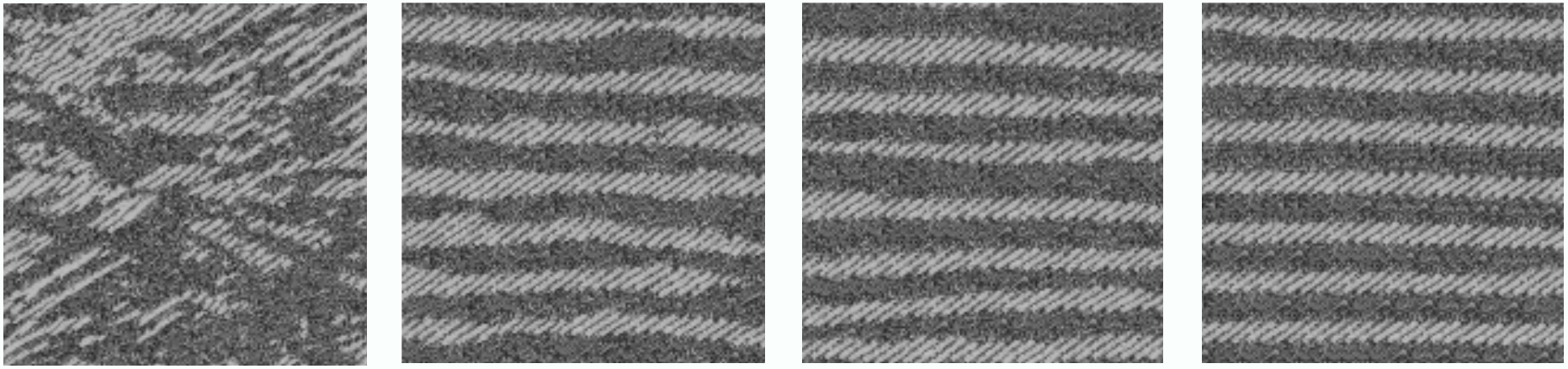
Neighborhood size

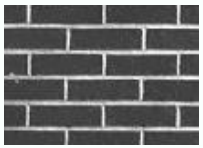


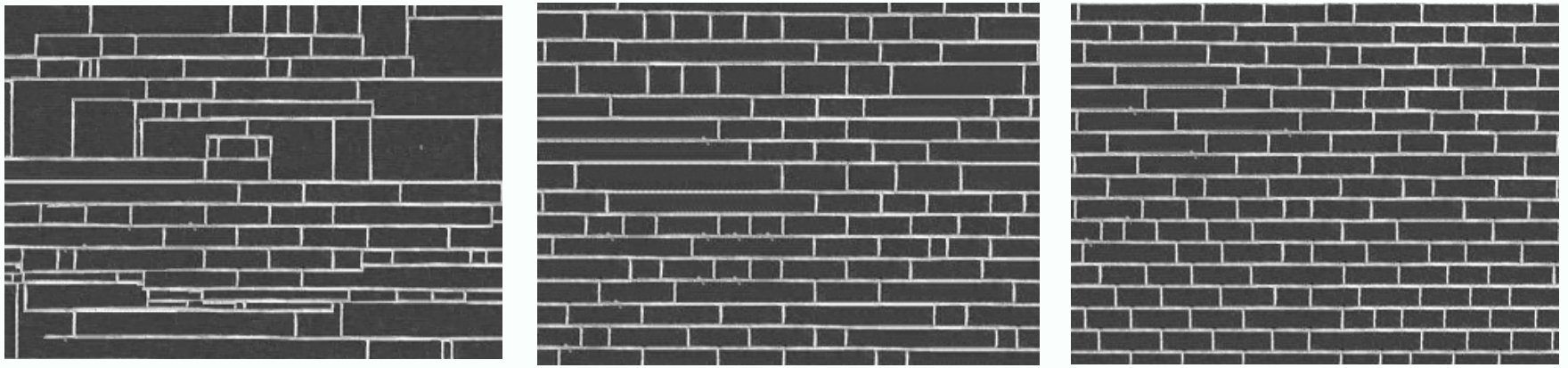
Efros & Leung ICCV99

Varying Window Size





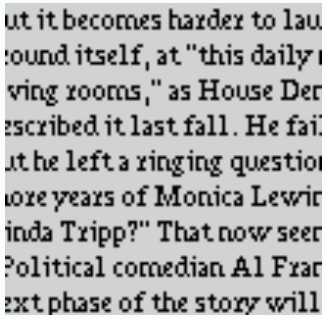


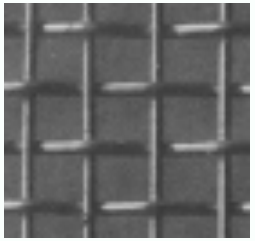


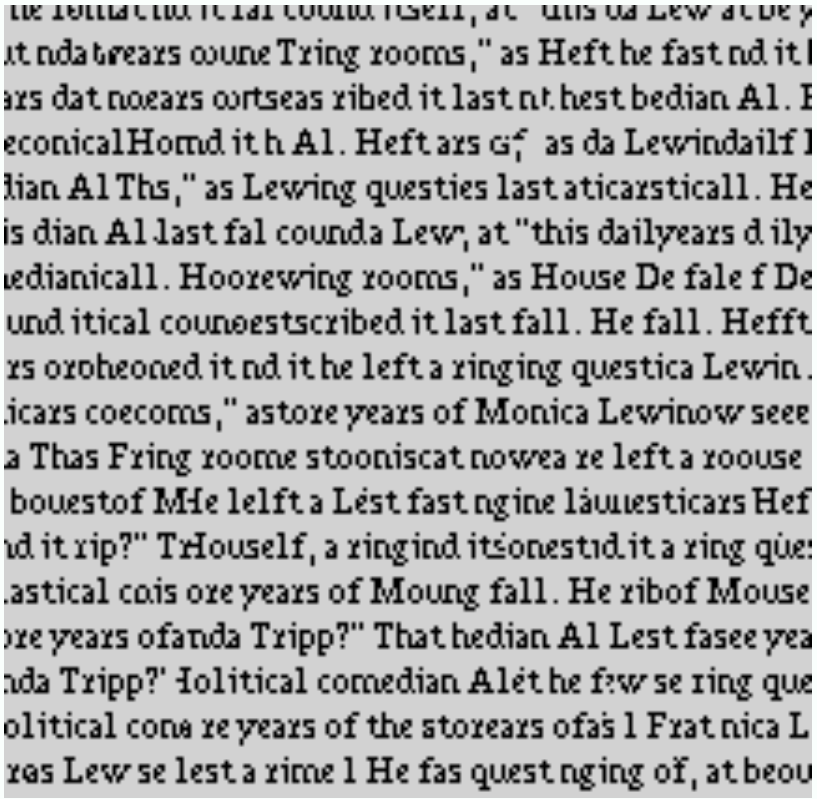
Increasing window size

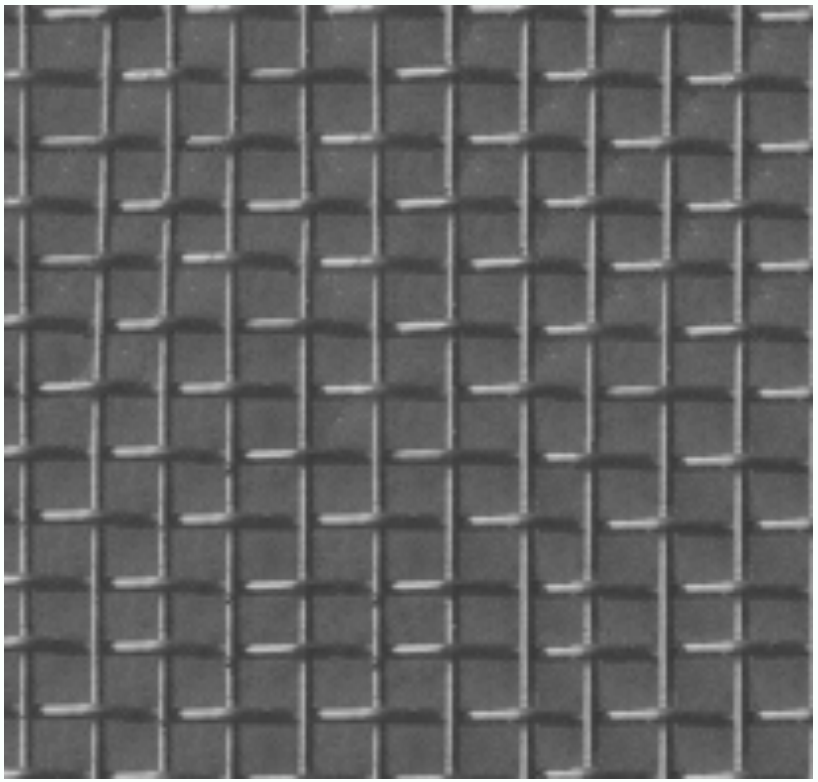
Efros & Leung ICCV99

More Results



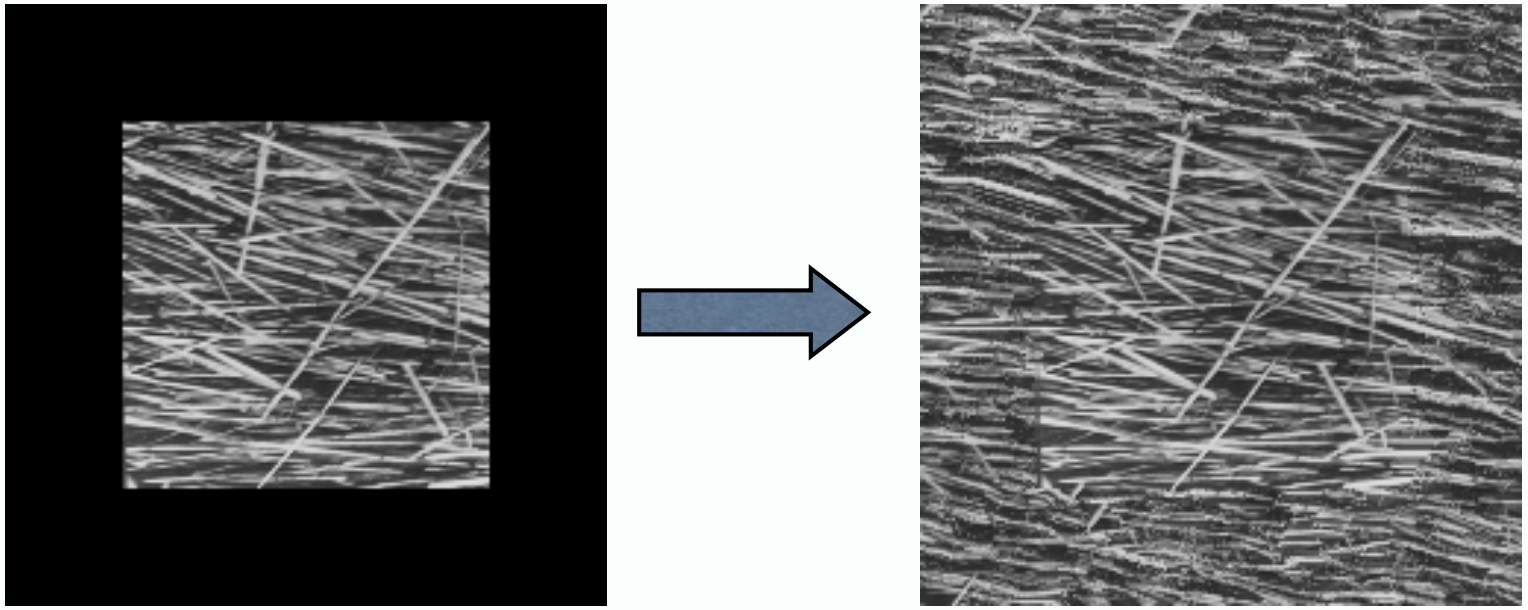


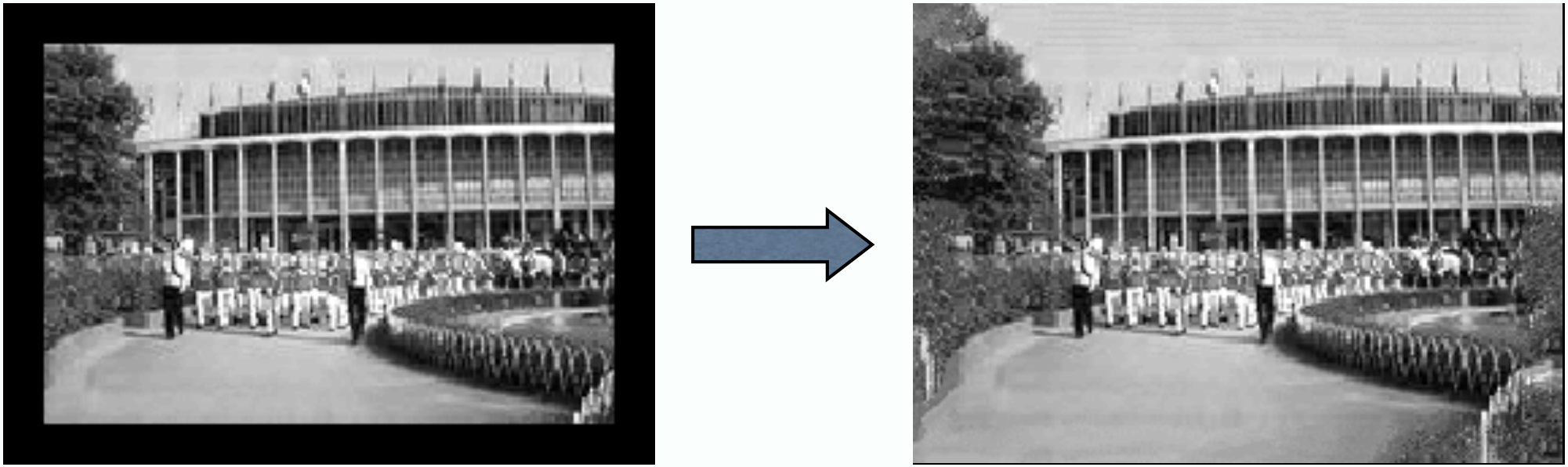




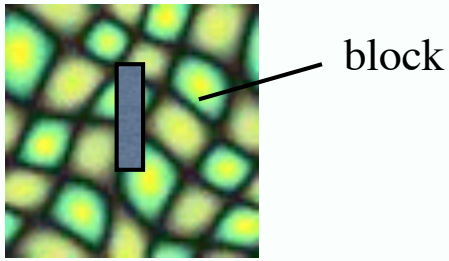
Efros & Leung ICCV99

Extrapolation

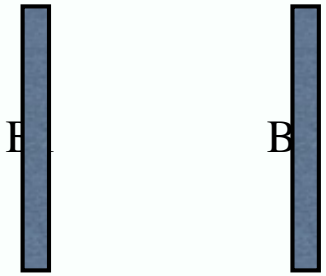




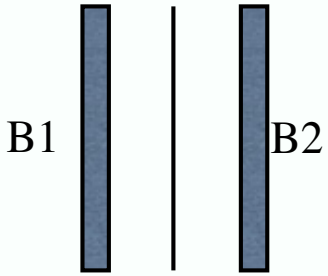
Efros & Leung ICCV99



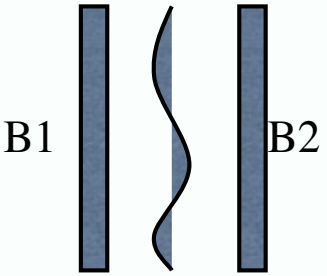
Input texture



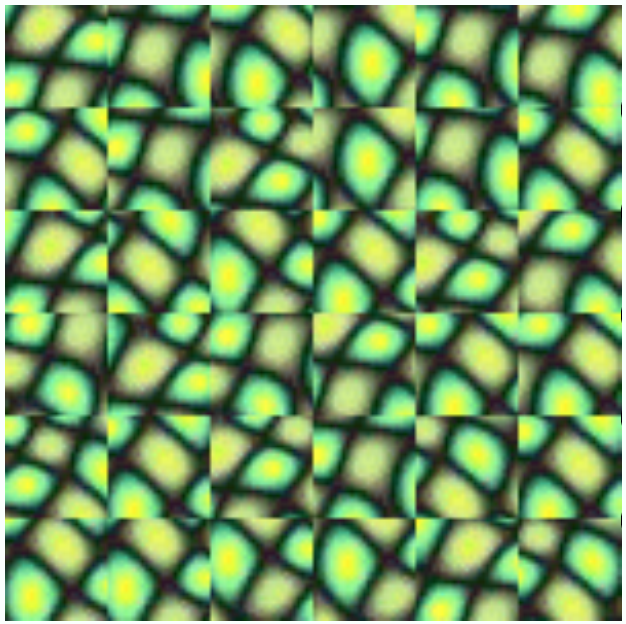
Random placement
of blocks

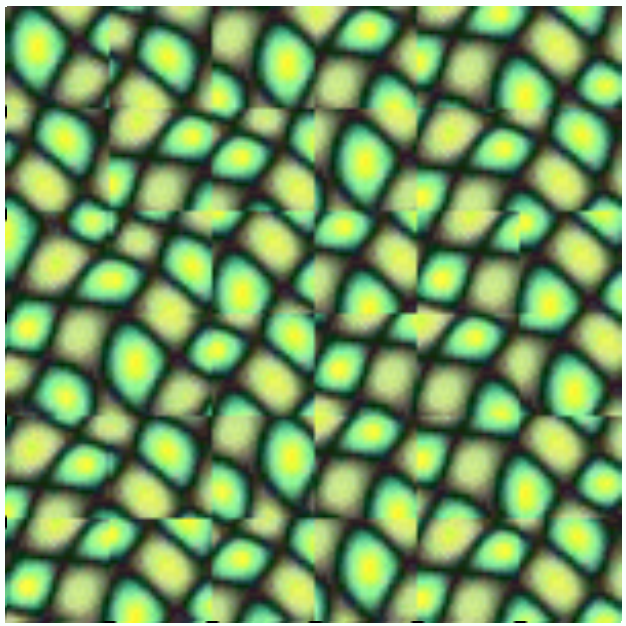


Neighboring blocks
constrained by overlap



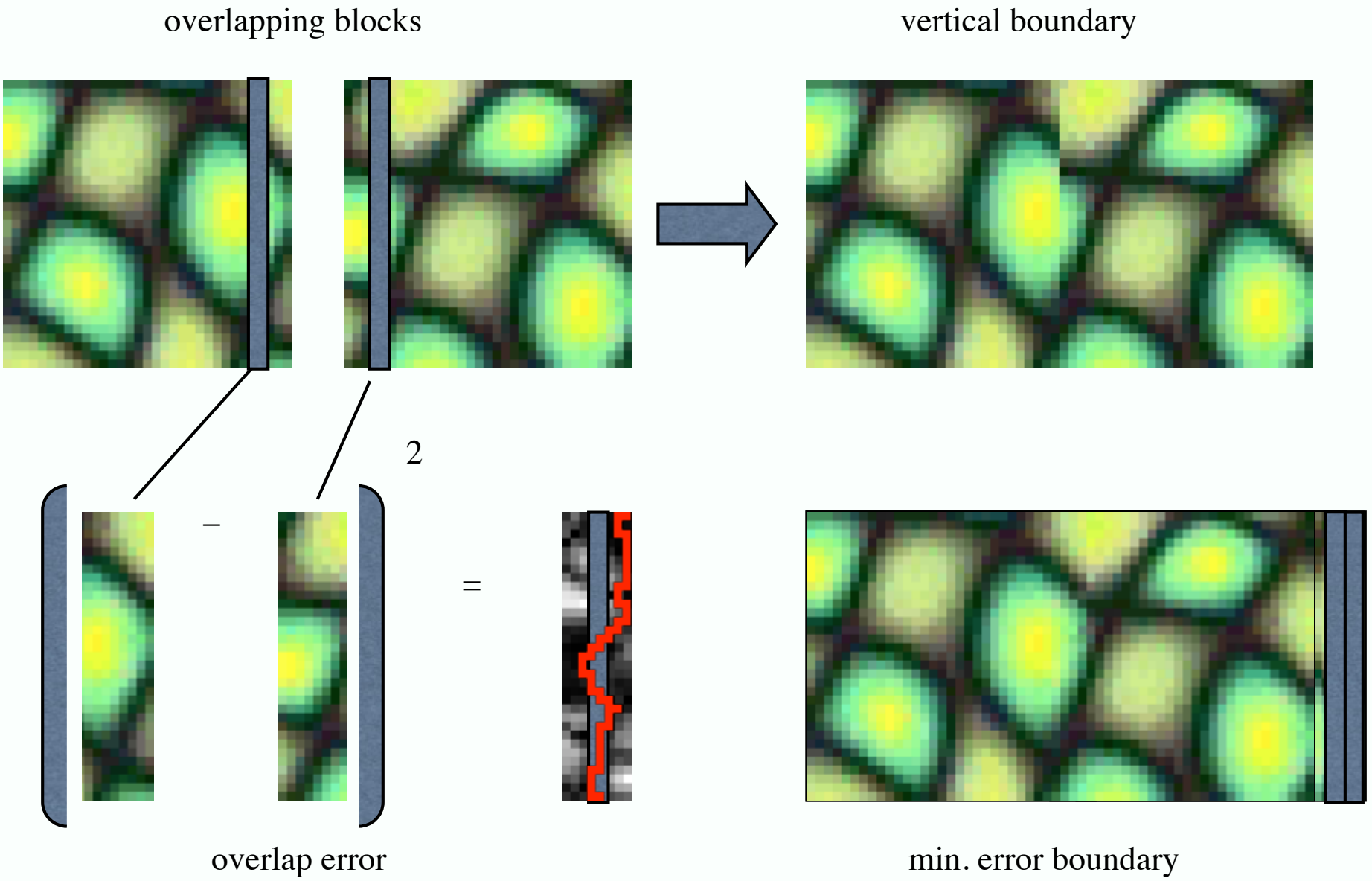
Minimal error
boundary cut

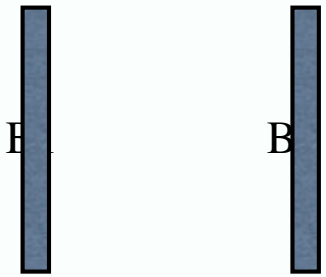




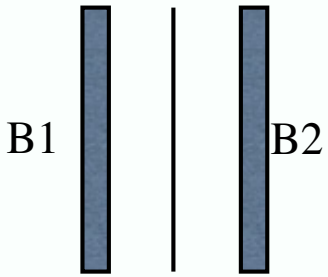
Efros & Freeman SIGGRAPH01

Minimal error boundary

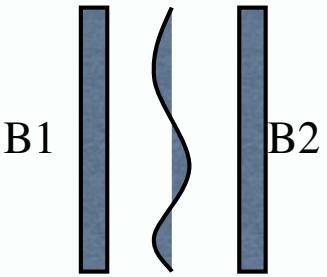




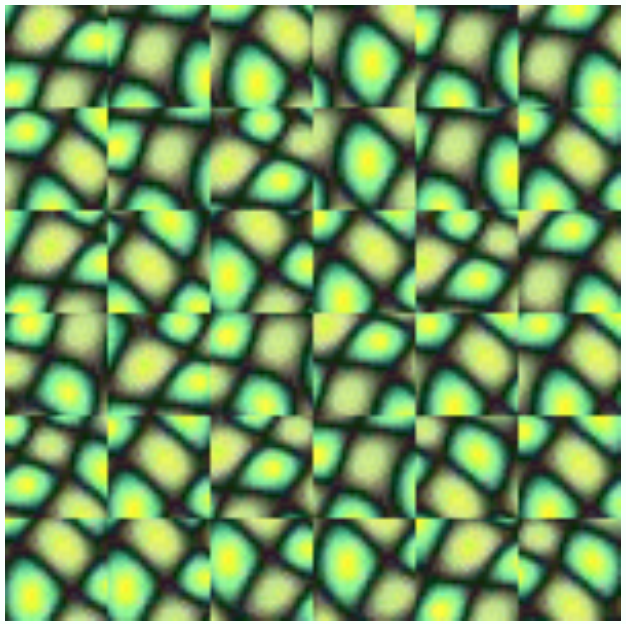
Random placement
of blocks

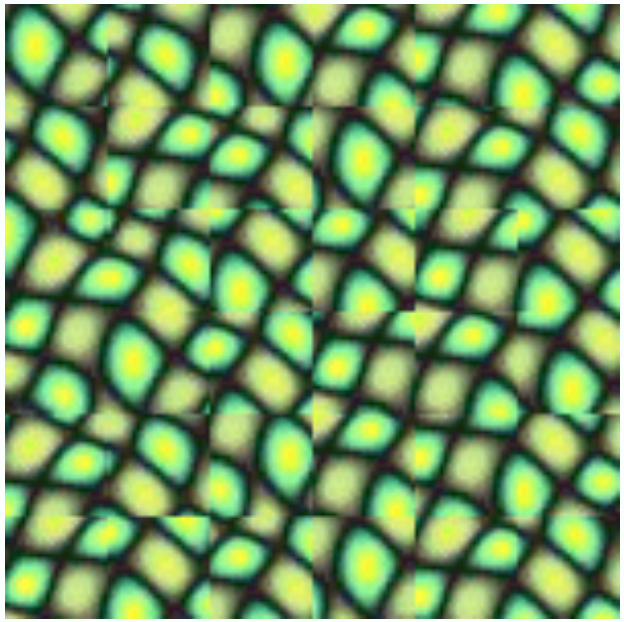


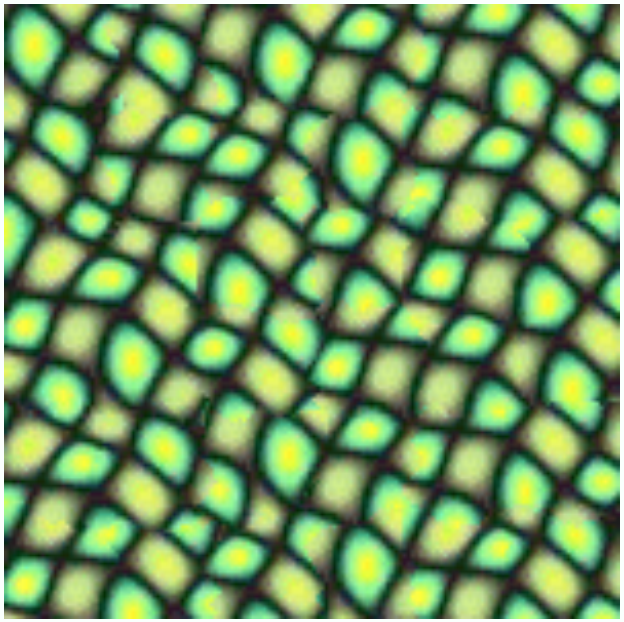
Neighboring blocks
constrained by overlap



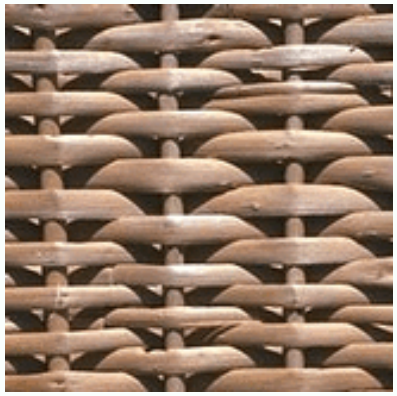
Minimal error
boundary cut

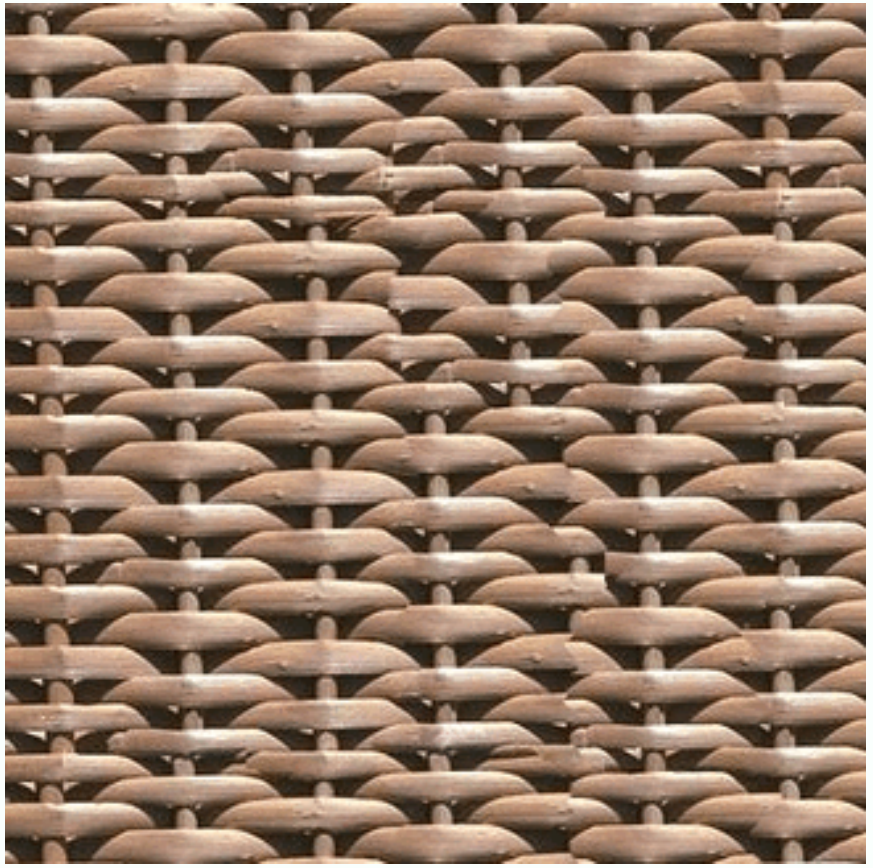




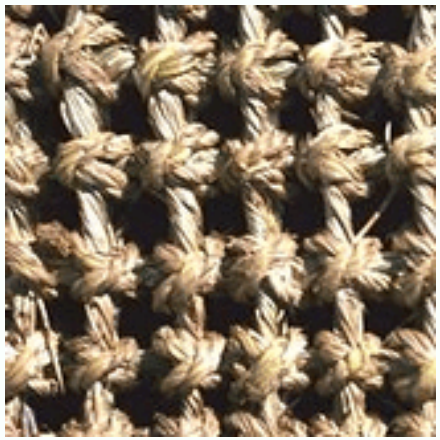


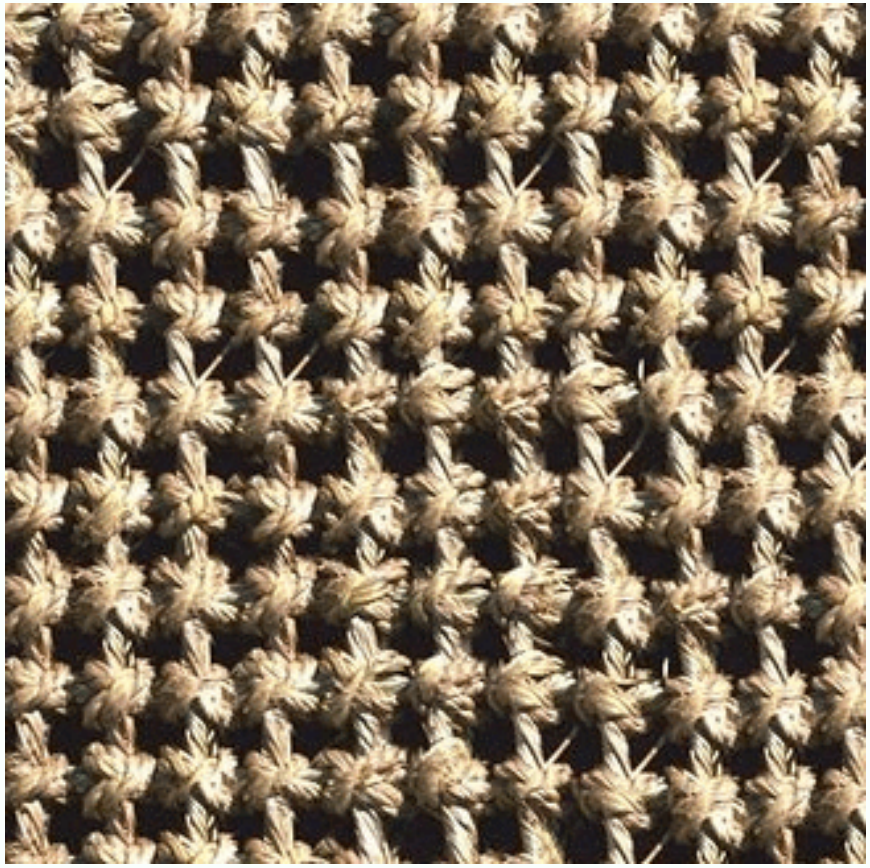
More Results





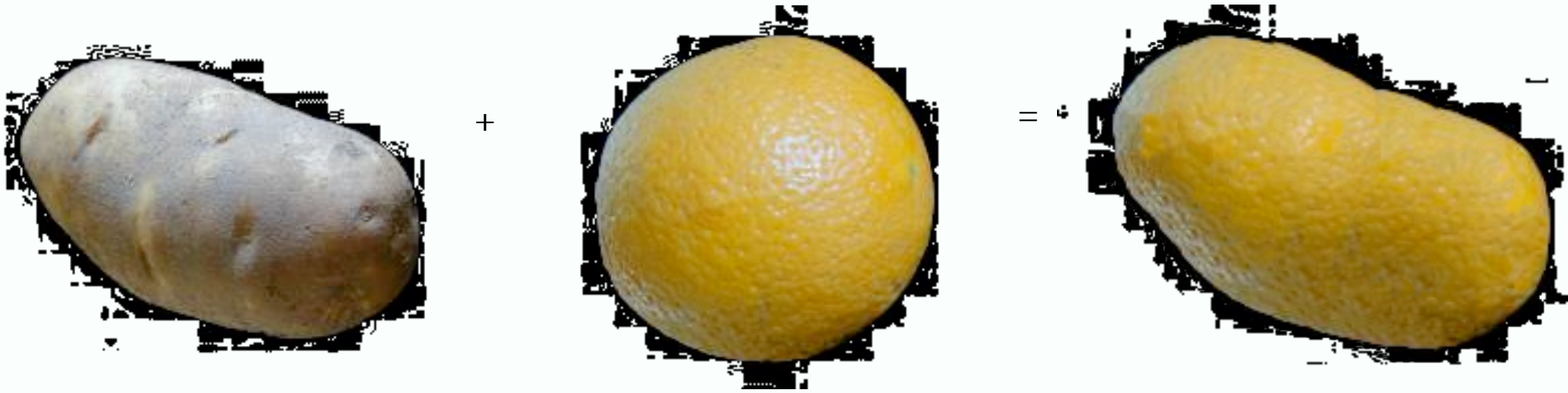
More Results





Texture Transfer

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



Decomposing shape and texture

Very challenging

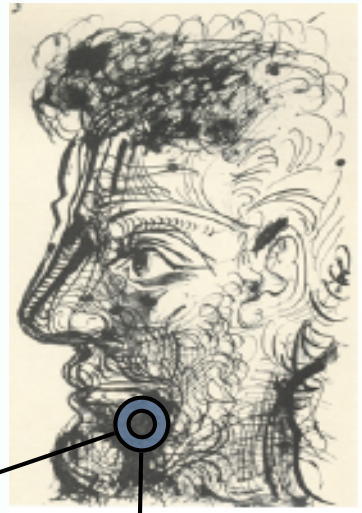
Walk around

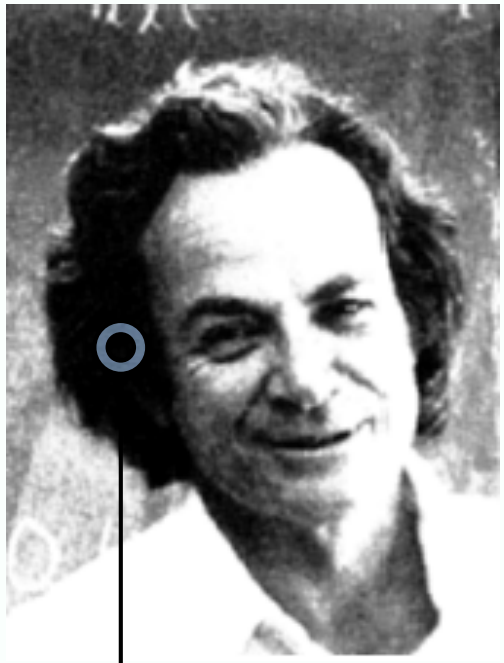
Add some constraint to the search

Efros & Freeman SIGGRAPH01

Source Texture

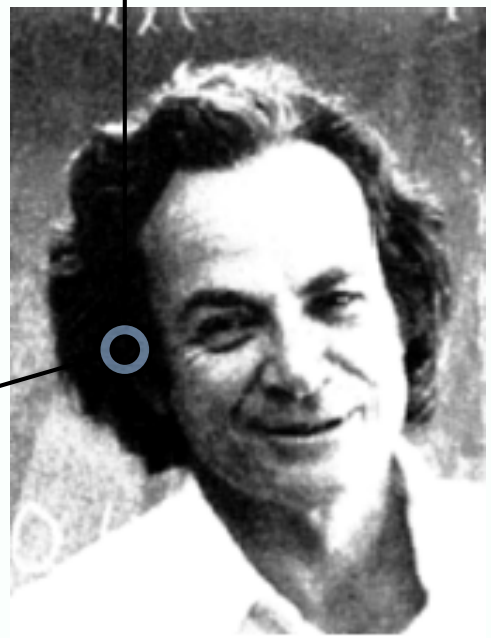
Destination

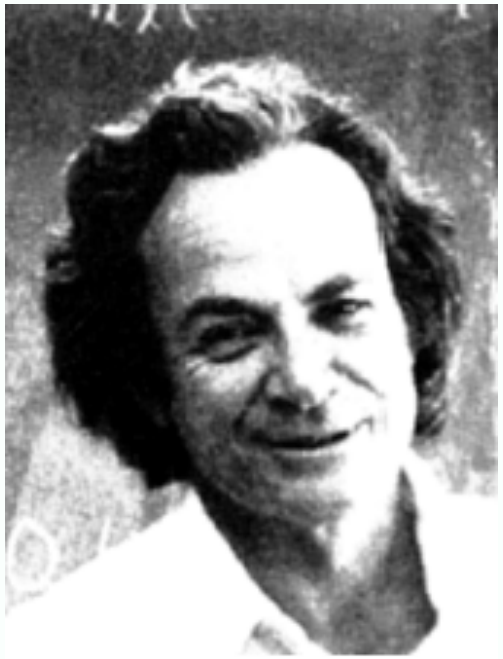




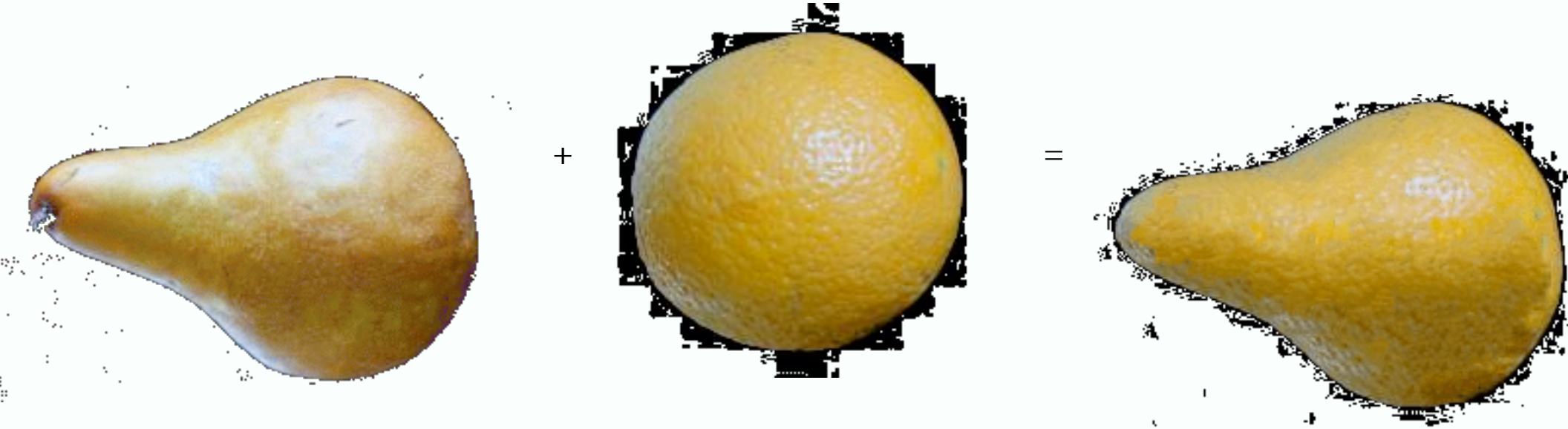
Source Map

Destination Map

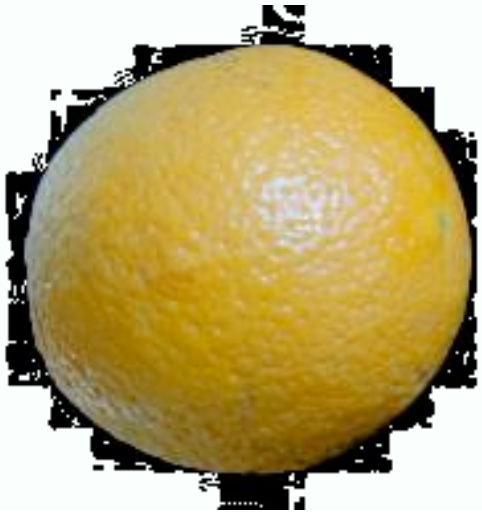




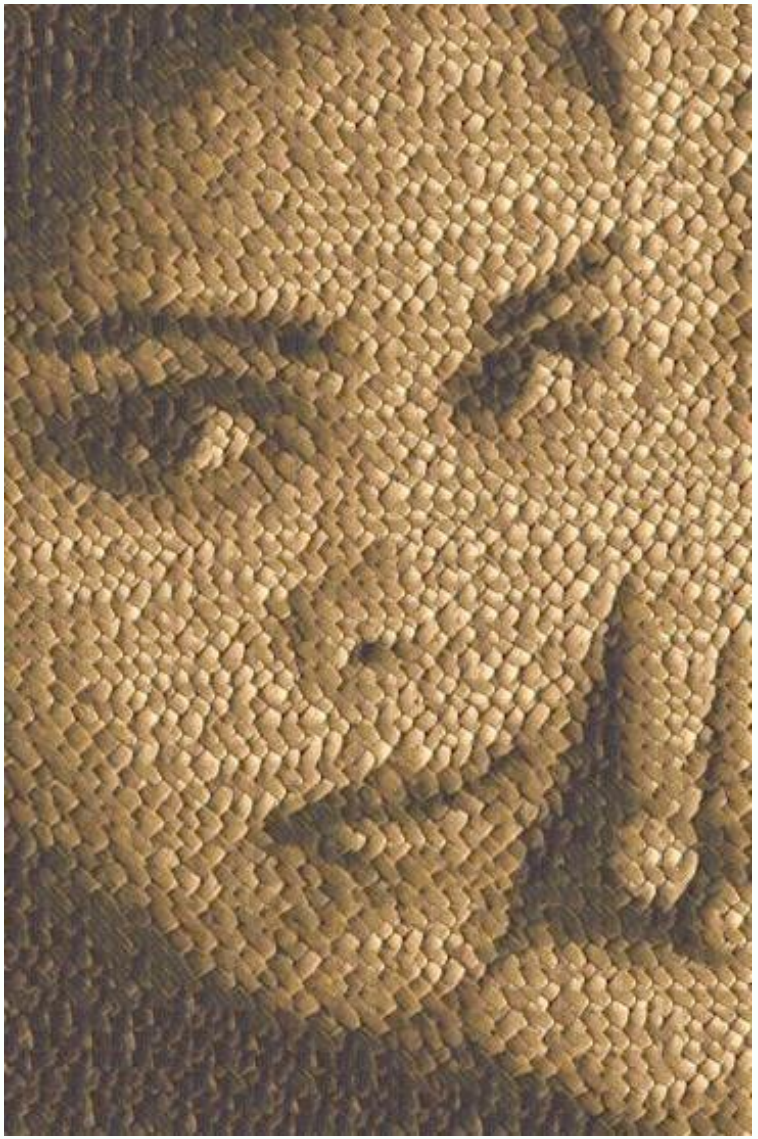
Texture Transfer



+



=



parmesan

+

=

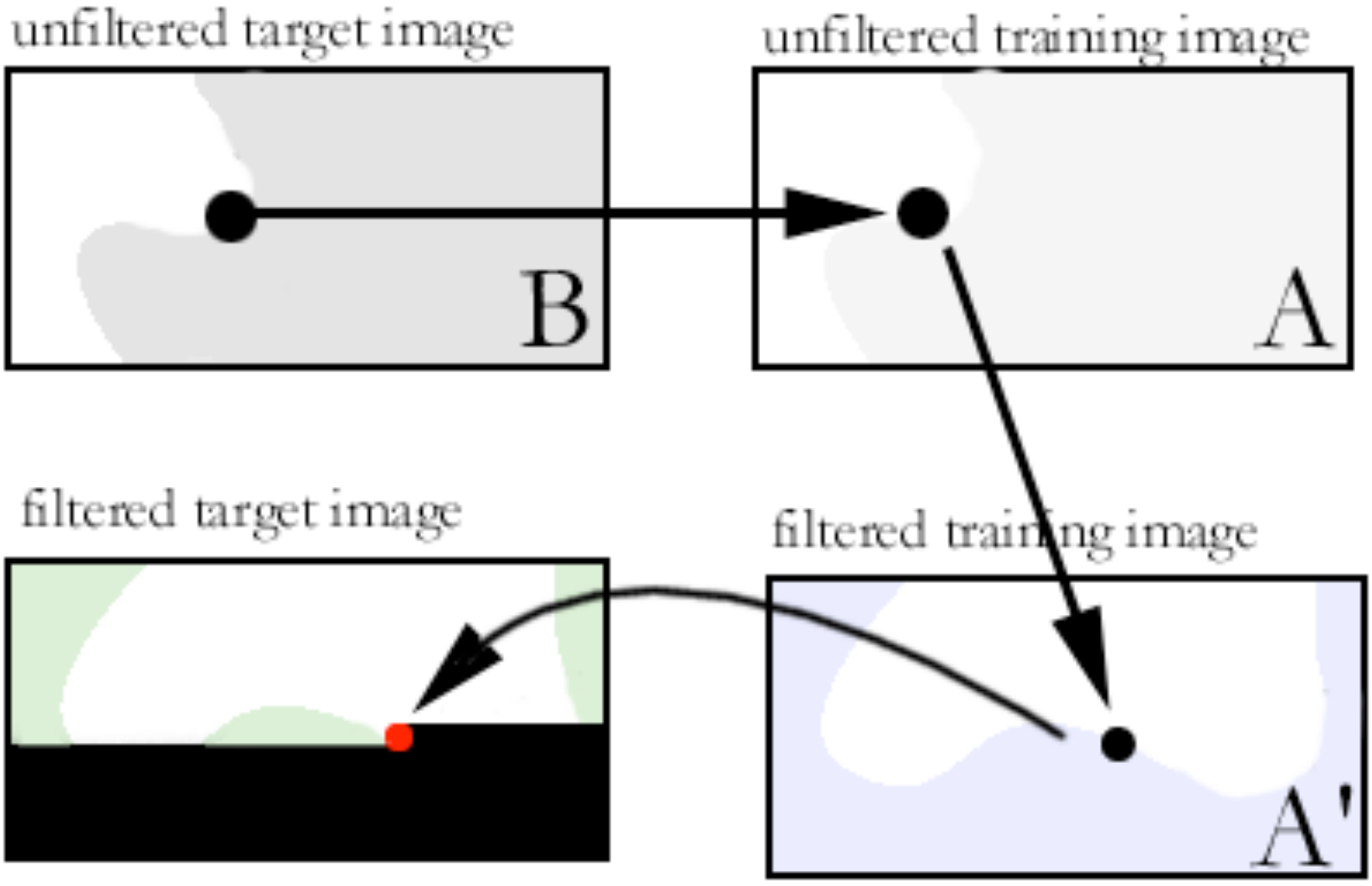
rice

+

=

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)

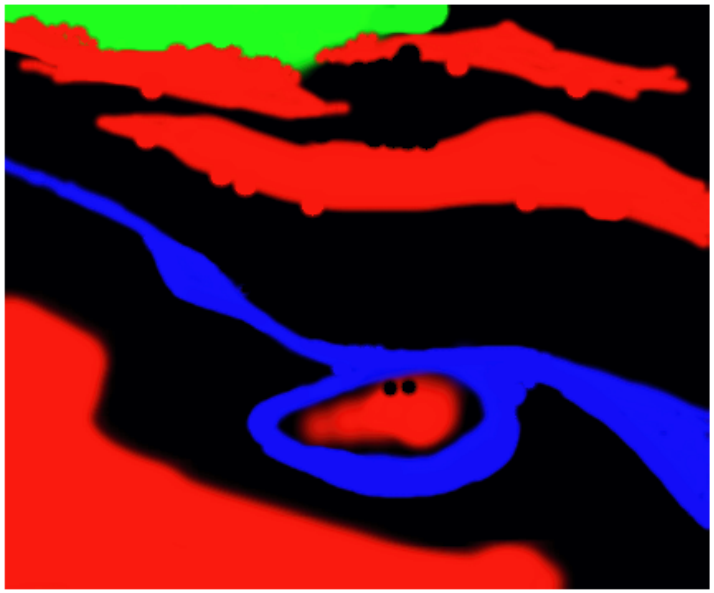
Filtered target (B')

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*)

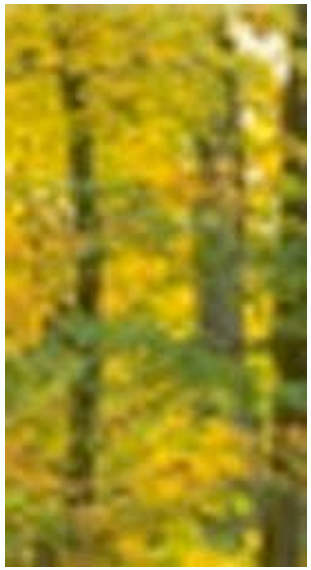
Filtered source (*A'*)

Unfiltered target (*B*)

Filtered target (*B'*)

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

Super-resolution



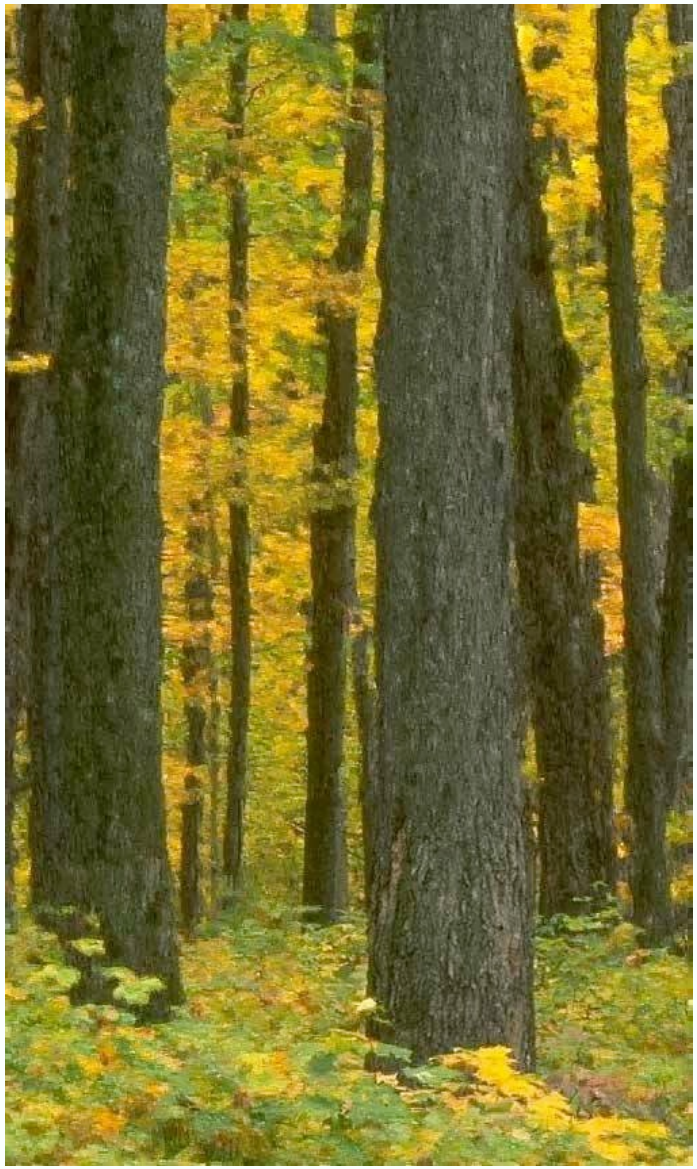
A

A’

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

Super-resolution (result!)

B



B’

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

Training images

Realistic images from approximations

- Pix2Pix
  - (Isola et al 16)
  - train with pairs of input, output, CGAN
- CycleGAN
  - (Isola et al 17)
  - train with populations, consistency losses
- CG2Real
  - (Bi et al 19)
  - fix some instabilities causing problems with cyclegan for graphics -> realistic
- SPADE
  - (Park et al 19)
  - fix problems with pix2pix and label maps
- SinGAN
  - the price of not knowing history

Labels to Street Scene

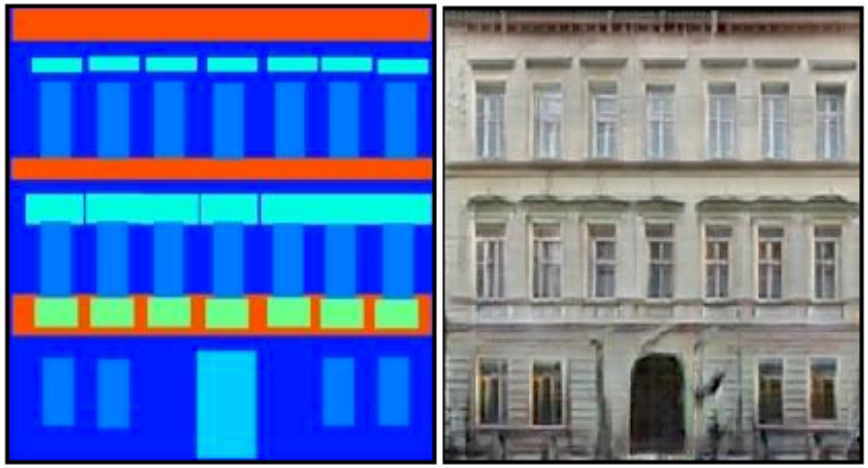
input output

Day to Night

input output

Isola et al, 16 (MUST READ)

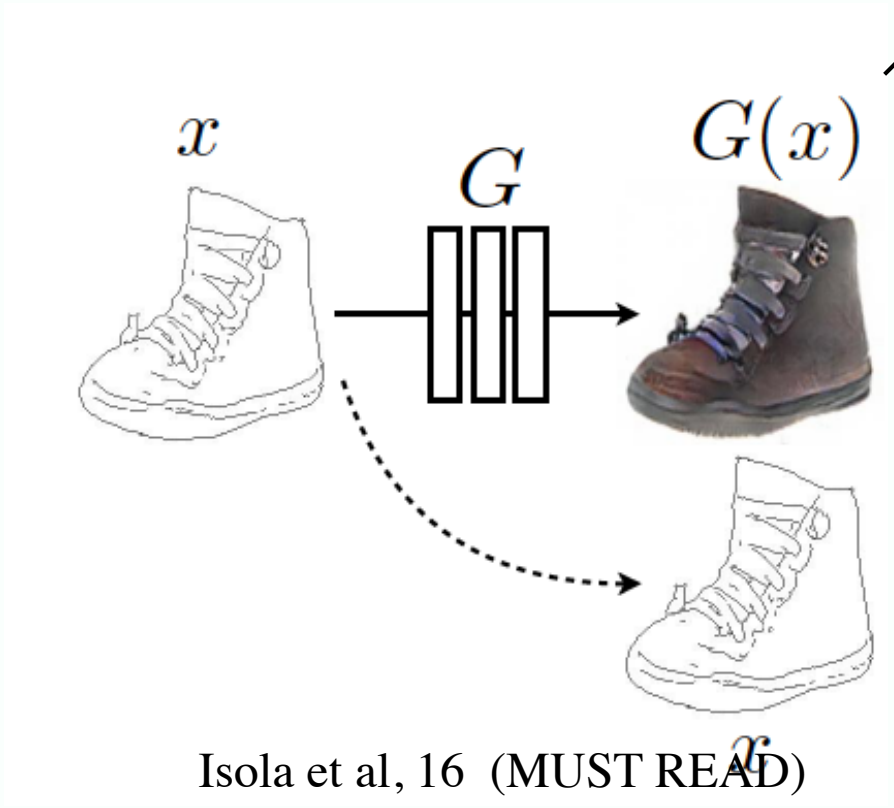
Labels to Facade



input output

BW to Color

input output



Compare to ground truth

Adversary tries to distinguish between generated and real pairs

Isola et al, 16 (MUST READ)

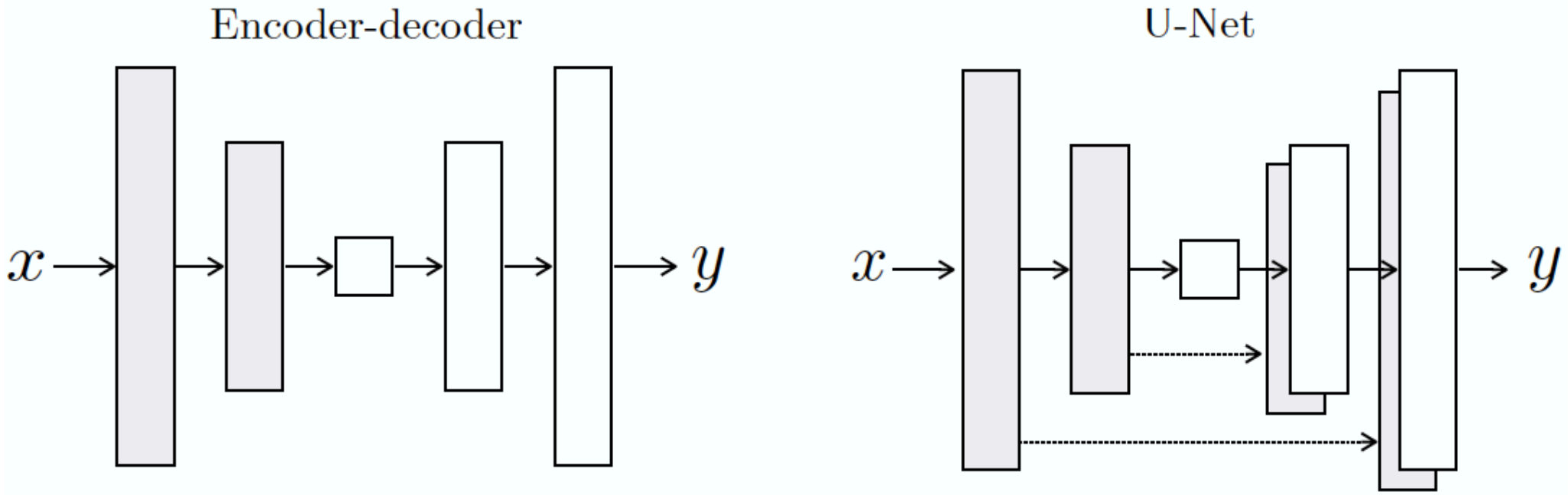


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Isola et al, 16 (MUST READ)

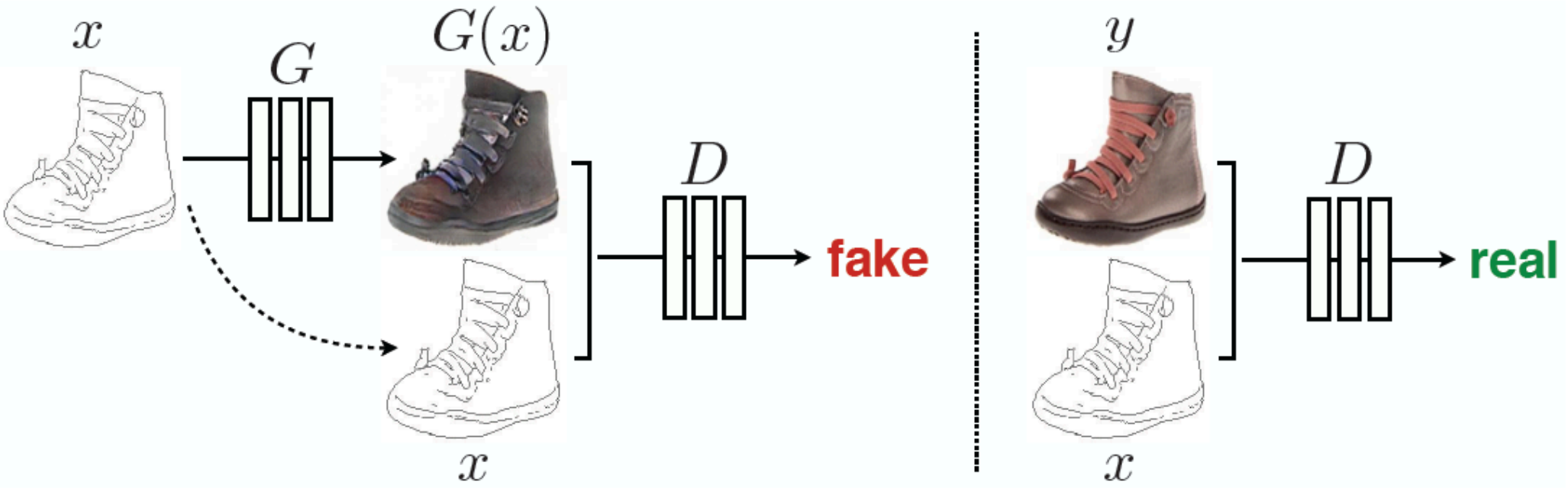


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

Isola et al, 16 (MUST READ)

Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

Isola et al, 16 (MUST READ)

Networks really like to smooth

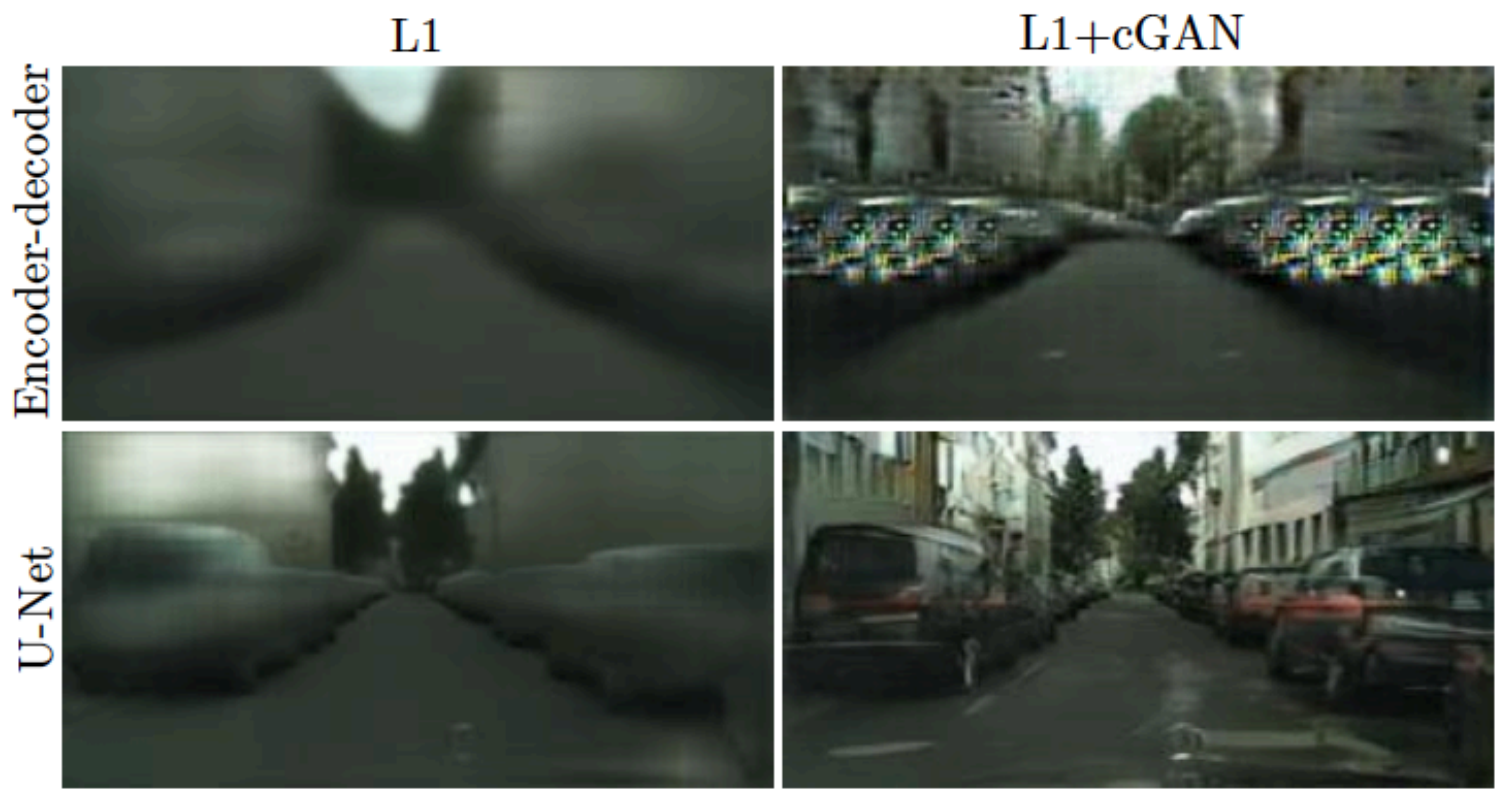


Figure 5: Adding skip connections to an encoder-decoder to create a "U-Net" results in much higher quality results.

Isola et al, 16 (MUST READ)

Patch size matters

Figure 6: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70×70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (colorfulness) dimensions. The full 286×286 ImageGAN produces results that are visually similar to the 70×70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 3). Please see `https://phillipi.github.io/pix2pix/` for additional examples.

Isola et al, 16 (MUST READ)

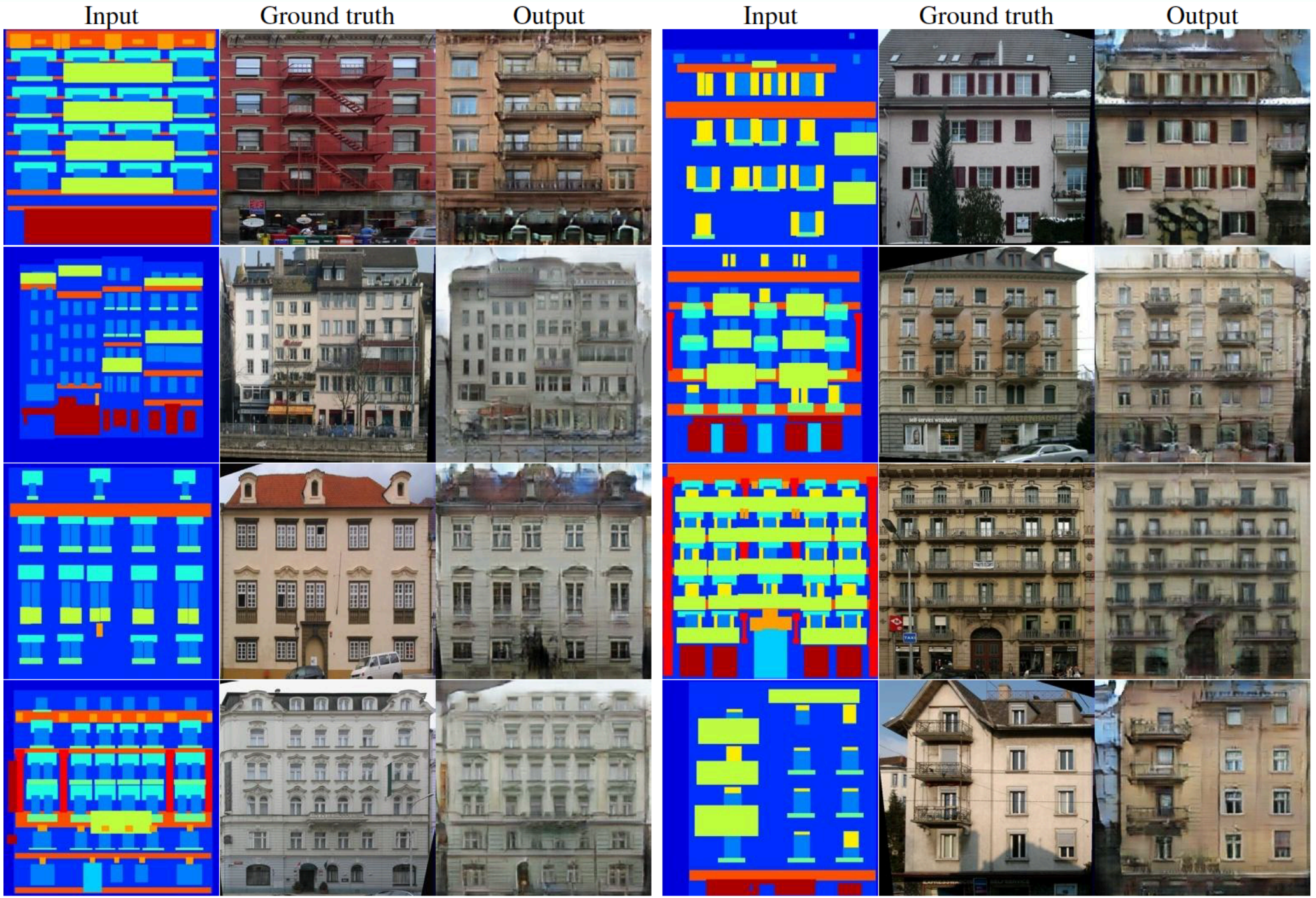


Figure 14: Example results of our method on facades labels→photo, compared to ground truth.

Isola et al, 16 (MUST READ)

Useful tricks

- Discriminator should be spectrally normalized
- Good idea for many activities to have a local discriminator
  - easily done - PatchGAN
  - benefits
    - realism at short scales likely imposes image realism
    - easier faster training
  - cautions
    - does not always apply
- D should have LeakyReLU
  - G - ReLU
- Variety of losses
  - hinge loss is quite good

Combinations known to work

- Spectral normalization, together with hinge or softplus loss
  - BigGAN,
- Non saturating loss together with gradient penalty
  - can clip gradient, can normalize
  - StyleGAN, clip OK
- Generally, UNet discriminator is helpful
  - with skips
  - with both kinds of loss
- For wasserstein, sliced wasserstein,
  - Schwing's students have a ton of knowledge
  - Ishan Deshpande
  - good for missing modes?

Issues

- What if you don't have pairs?
  - CycleGAN, next
  - CG2Real, next
- Story for semantic labels is weird
  - why pass label map through encoder?
    - SPADE, next

Realistic images from approximations

- Pix2Pix
  - (Isola et al 16)
  - train with pairs of input, output, CGAN
- CycleGAN
  - (Isola et al 17)
  - train with populations, consistency losses
- CG2Real
  - (Bi et al 19)
  - fix some instabilities causing problems with cyclegan for graphics -> realistic
- SPADE
  - (Park et al 19)
  - fix problems with pix2pix and label maps
- SinGAN
  - the price of not knowing history

What if you don't have pairs?

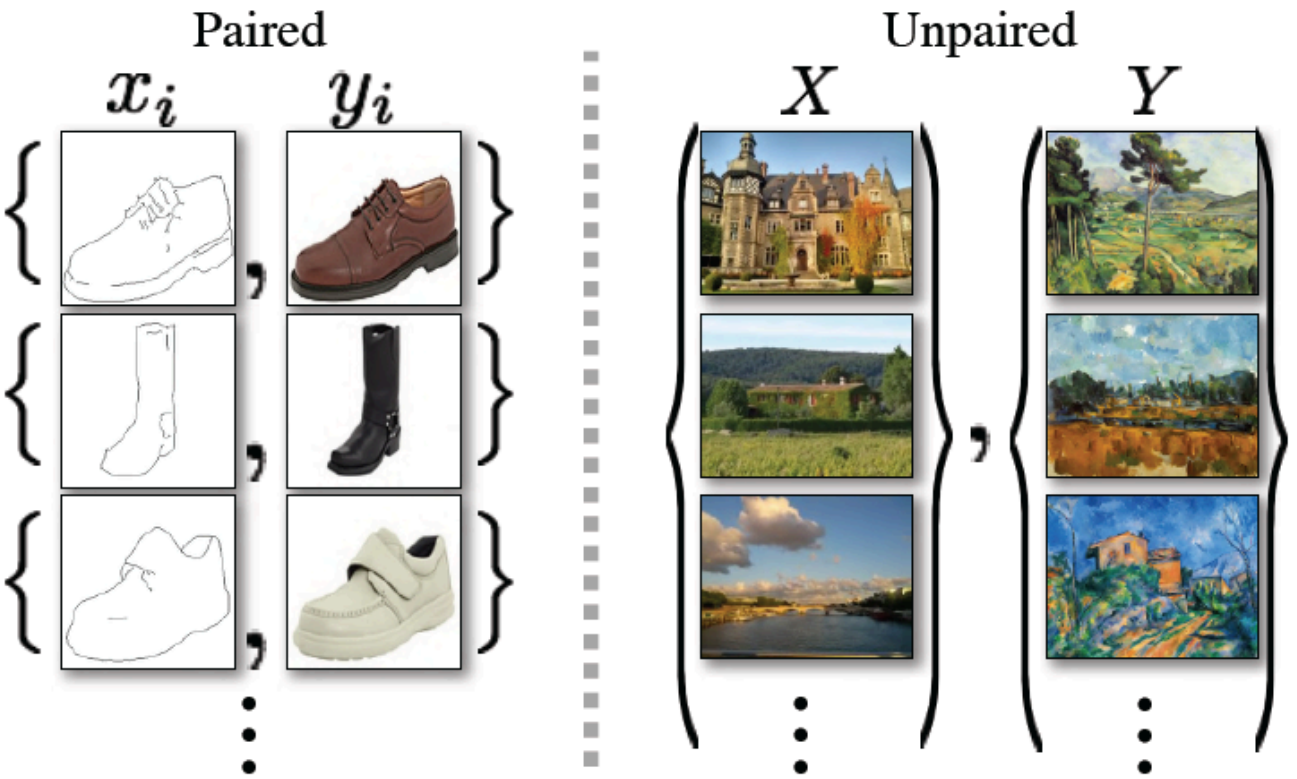


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^{N}$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^{N}$ $(x_i \in X)$ and a target set $\{y_j\}_{j=1}^{M}$ $(y_j \in Y)$, with no information provided as to which x_i matches which y_j.

Zhou et al, 17

Learn paired mappings

- G: X->Y, H: Y->X
  - G(x_i) should be “like a y” (Adversary)
  - H(y_i) should be “like an x” (Adversary)
  - Cycles work:
    - G(H(y_i)) should be about y_i
    - H(G(x_i)) should be about x_i

This isn’t innocent, or natural

This works

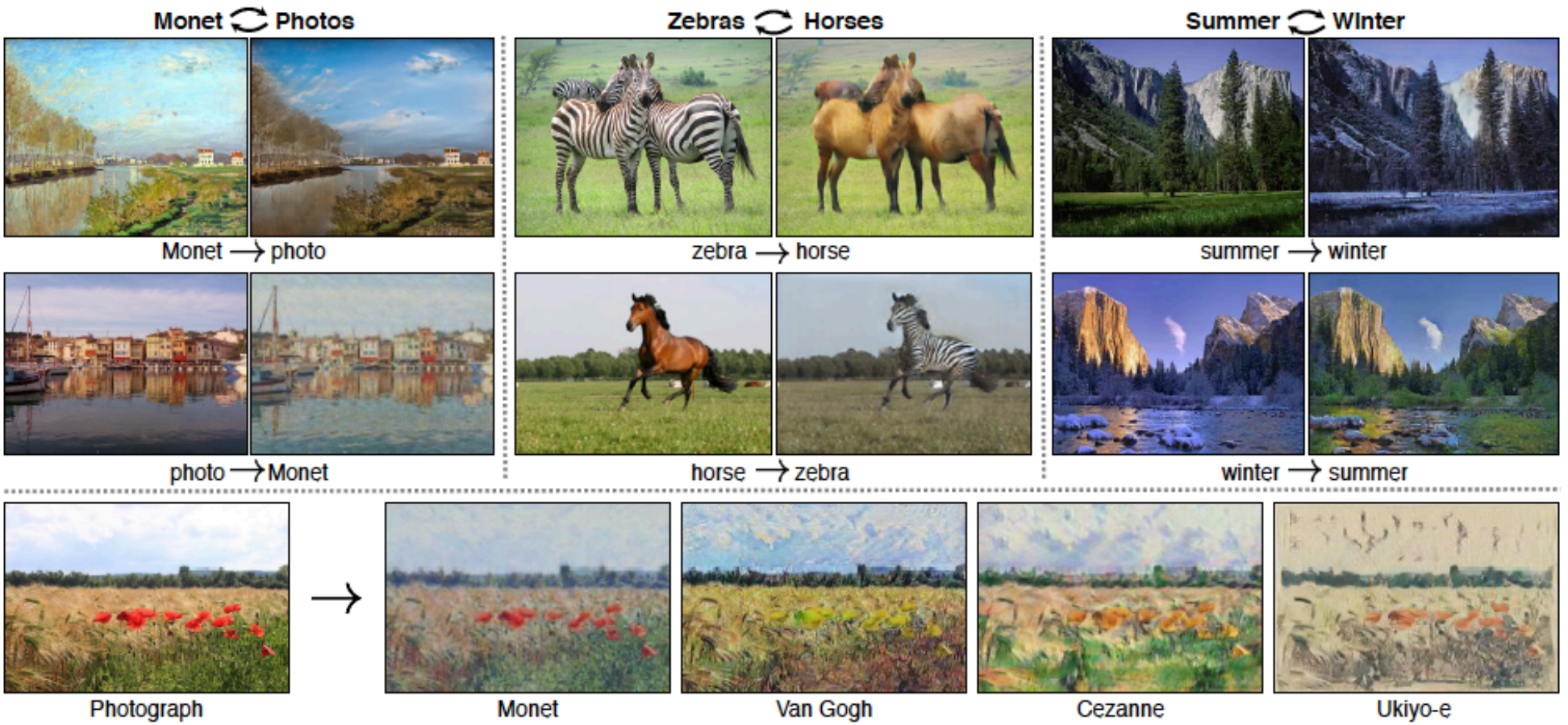
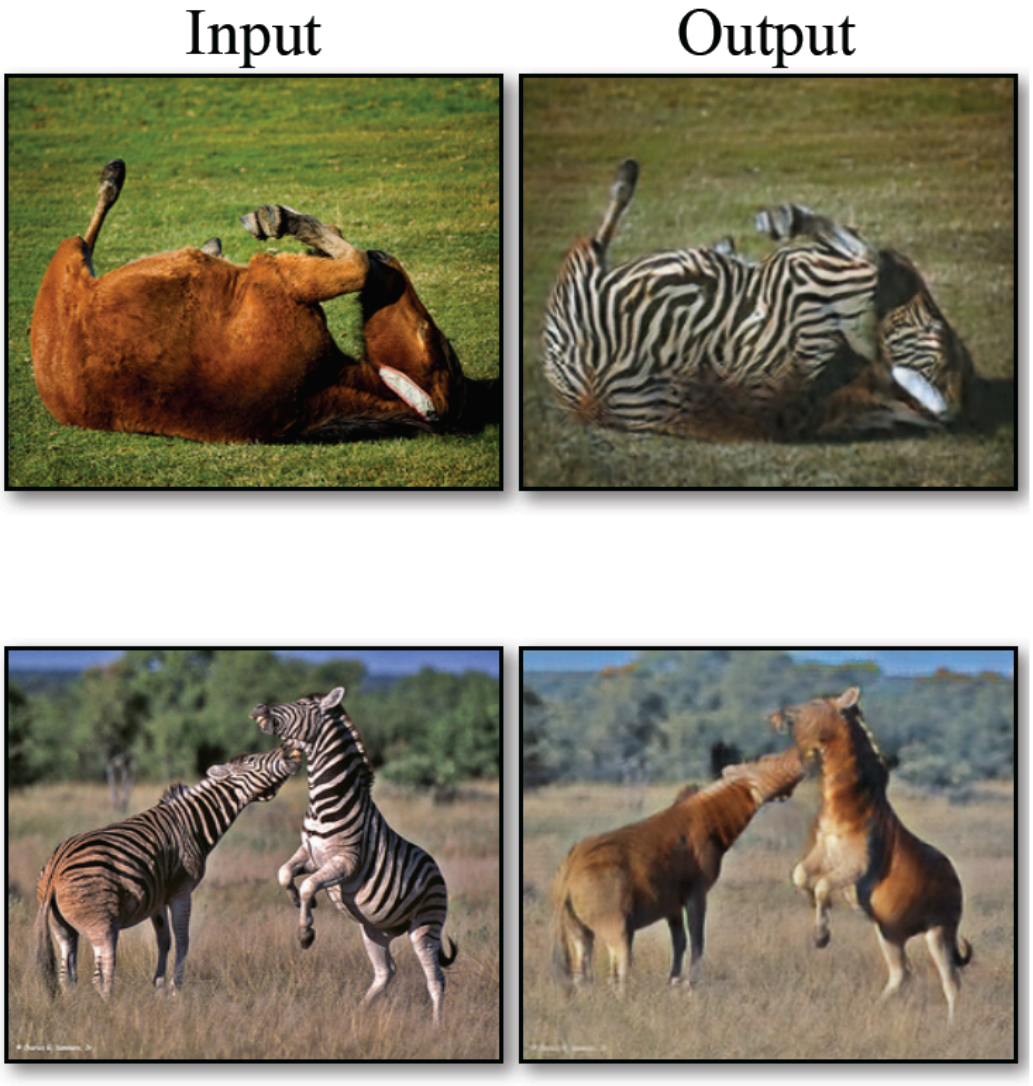


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

Zhou et al, 17

But there are issues…



Does not like to “destroy” information

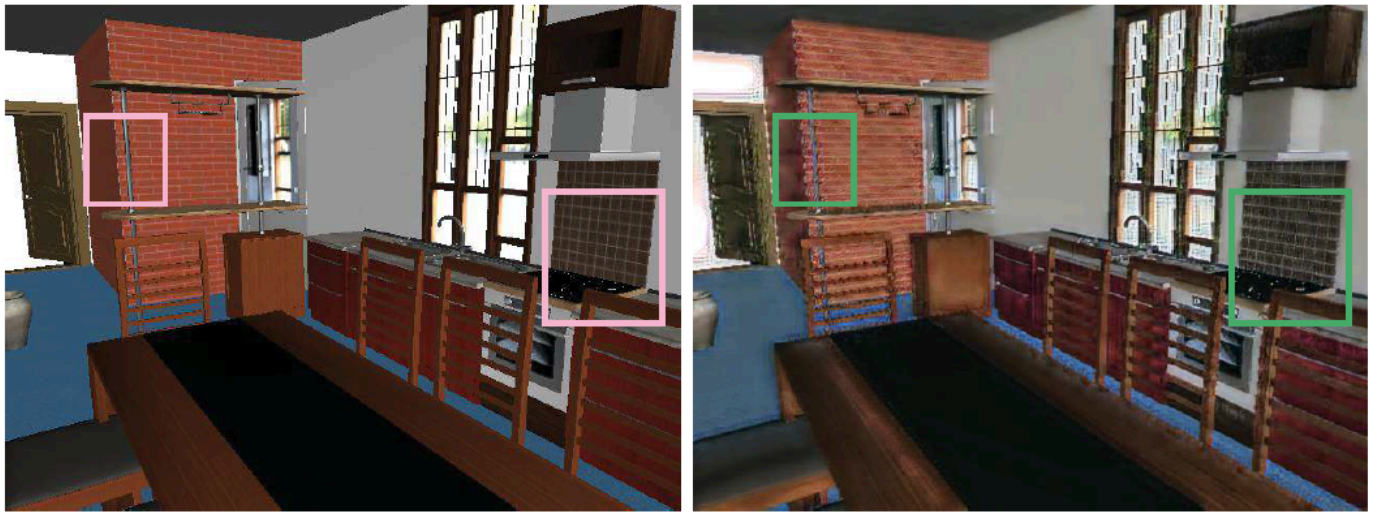
Zhou et al, 17

Realistic images from approximations

- Pix2Pix
  - (Isola et al 16)
  - train with pairs of input, output, CGAN
- CycleGAN
  - (Isola et al 17)
  - train with populations, consistency losses
- CG2Real
  - (Bi et al 19)
  - fix some instabilities causing problems with cyclegan for graphics -> realistic
- SPADE
  - (Park et al 19)
  - fix problems pix2pix has with label maps
- SinGAN
  - the price of not knowing history

Goal: OpenGL to “Realistic”

- Value:
  - control of rendering in traditional fashion
    - eg use existing assets, etc.
    - but get better images
  - possibly faster, too



(a) OpenGL image (b) CycleGAN result

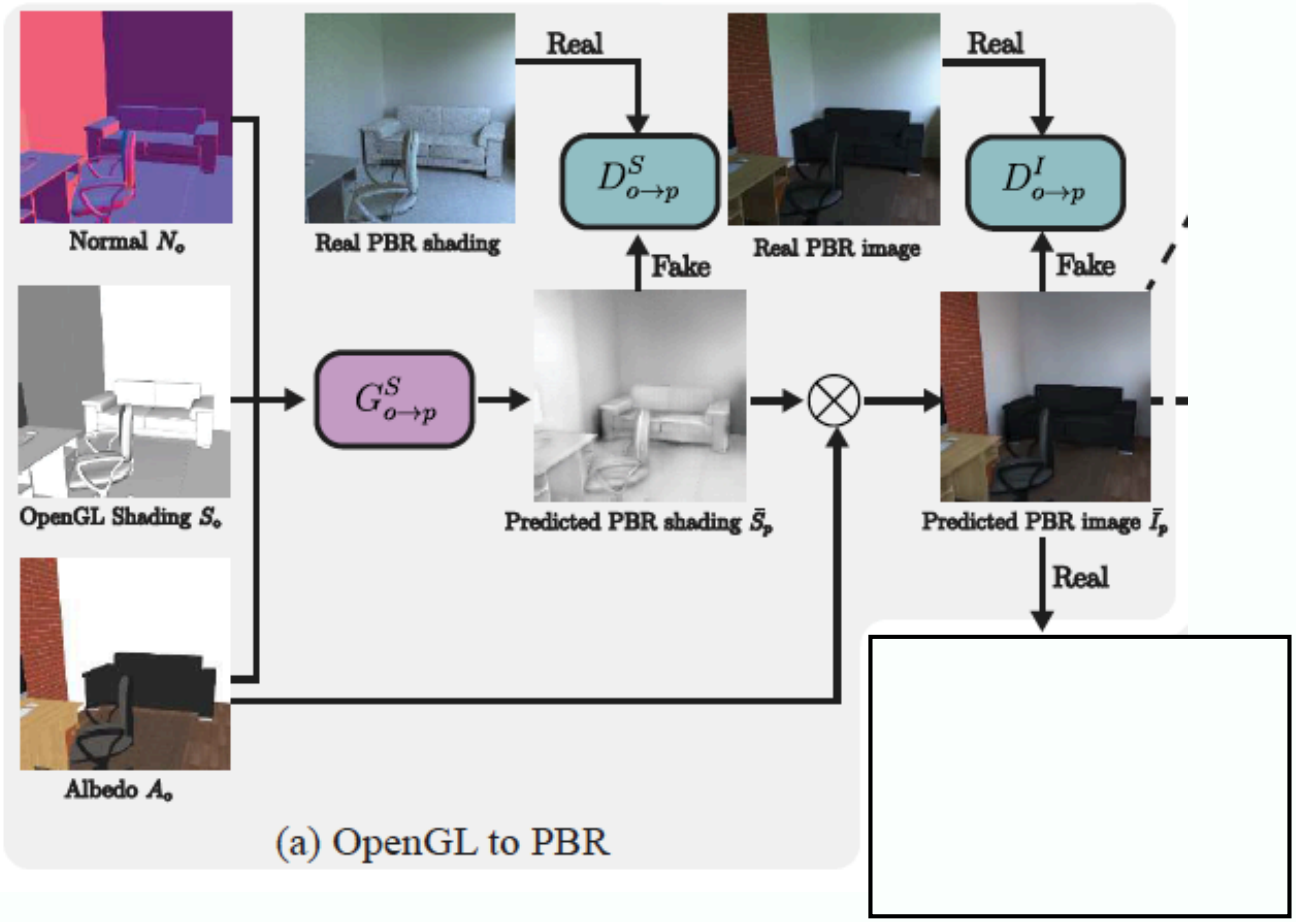
(c) Our predicted real image (a) (b) (c)

Figure 1: Our two-stage adversarial framework translates an OpenGL rendering (a) to a realistic image (c). Compared to single-stage prediction with CycleGAN (b), our result has more realistic illumination and better preserves texture details, as shown in the insets. (Best viewed in digital).

Bi et al 19

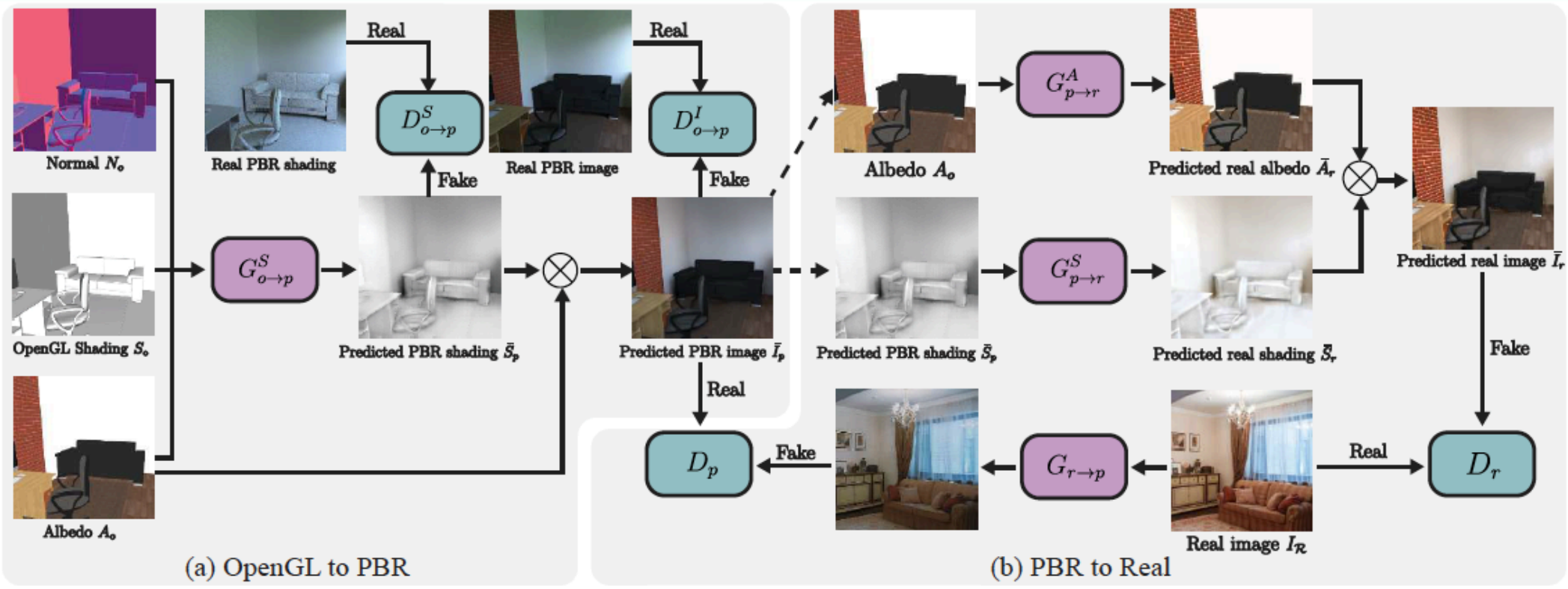
Likely complication - asymmetry

- OpenGL shading -> Physical shading
  - hard, requires long scale information
  - BUT can make paired data easily with physically accurate renderer
- OpenGL albedo -> realistic albedo
  - likely much easier, requires short scale information
  - BUT no paired data
- Strategy:
  - handle albedo and shading separately
    - shading paired, albedo unpaired



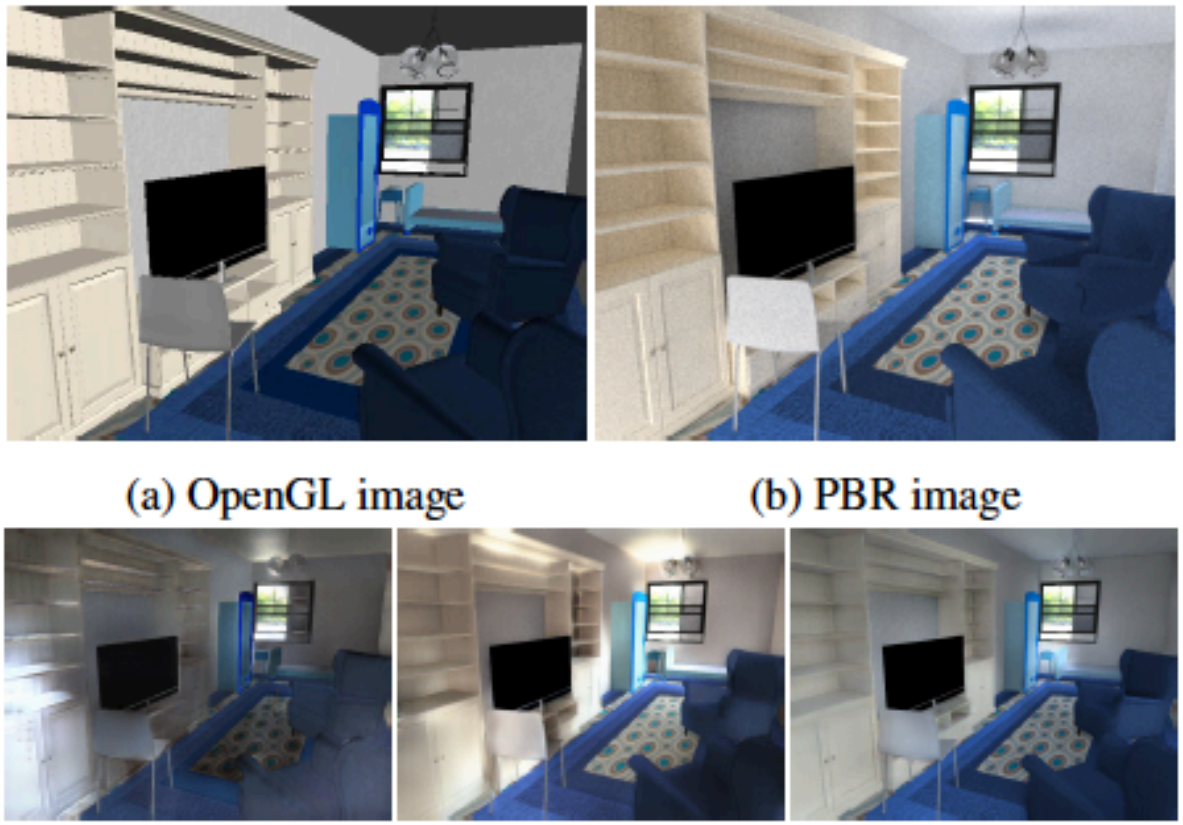
OpenGL to PBR

Bi et al 19



Whole thing

Bi et al 19



(a) OpenGL image (b) PBR image

(c) p2pHD-OpenGL (d) p2pHD-S+A+N (e) Our result

Figure 7: OpenGL to PBR comparison. A PBR renderer like Mitsuba needs around 20 mins. to render a noise-free image as in (b). In comparison our network can generate a high quality result in 0.03 secs. (e). Compared to the pix2pix-HD framework that directly predicts the output images using OpenGL (c) or auxiliary buffers (d), with inconsistent shading such as abrupt highlights on the cabinet, our method generates images with much higher visual realism.

Bi et al 19

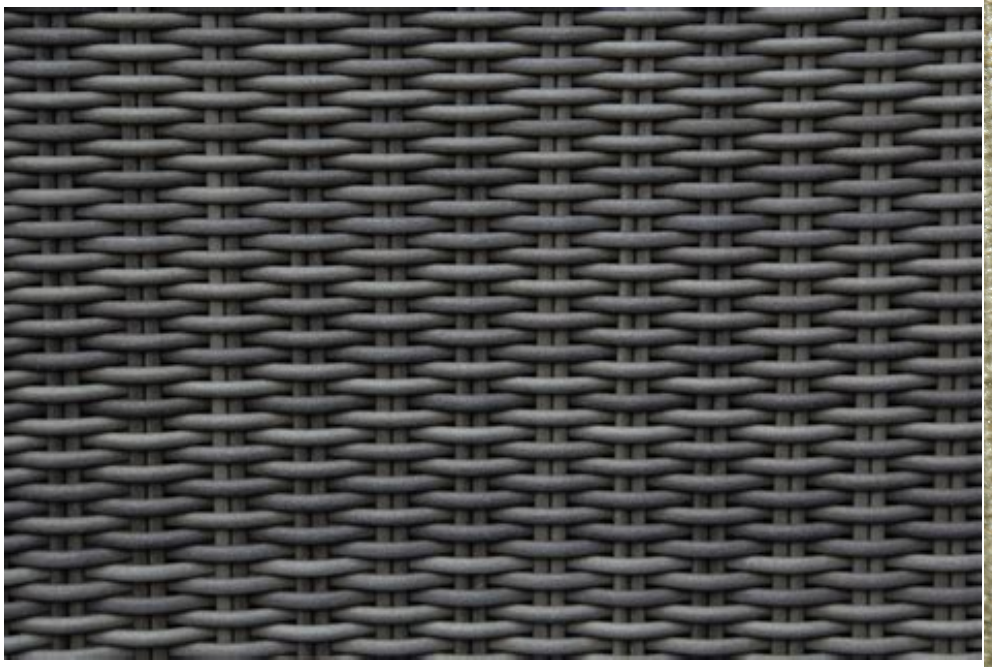
Notes and queries

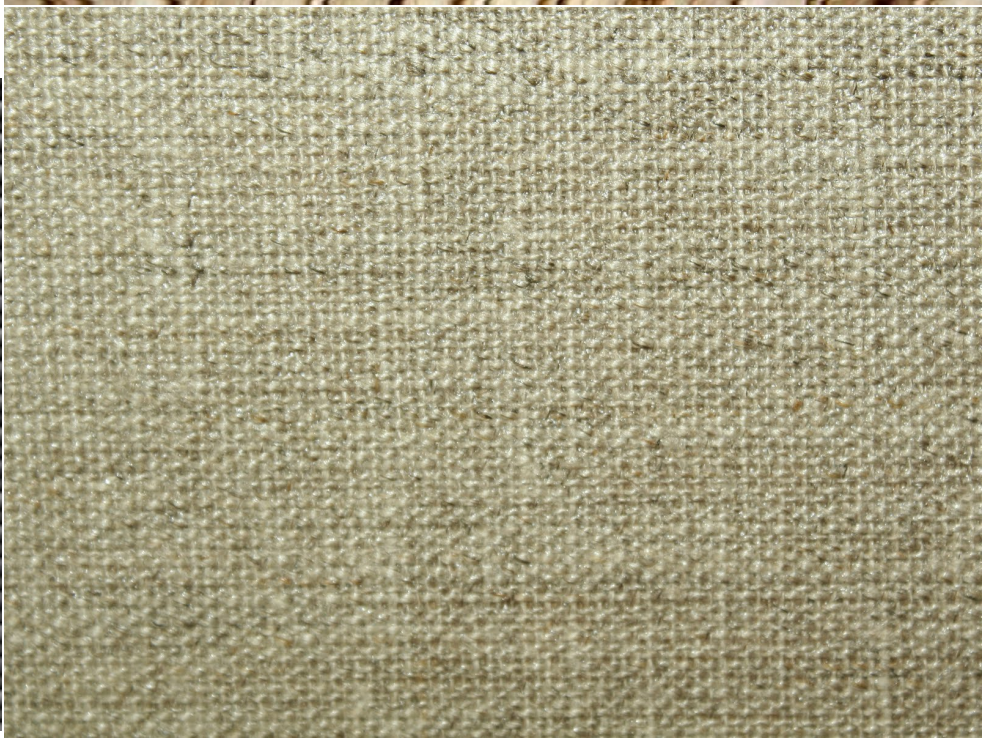
- Notice a tension between realism and control here
  - Procedure offers very good control of surface attributes
    - you do this with OpenGL textures
  - but surface attributes aren't spectacularly realistic
    - (following slides)
  - ISSUE
    - Making realistic surfaces may involve loss of control
    - How do we manage and balance?

- Physically based shading is all very well
  - but it isn't the key to realism

Relief - intrinsic, because
small local shadows do not
move with illumination
(at least Koenderink+Van Doorn, 77)

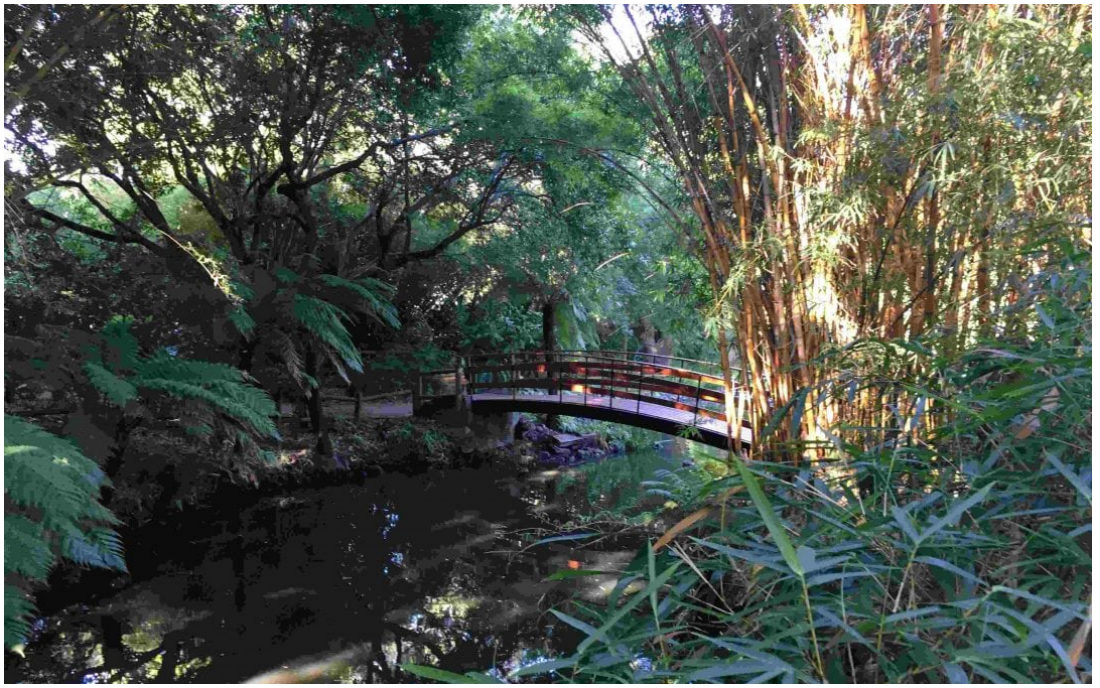
Relief - intrinsic, because
small local shadows do not
move with illumination
(at least Koenderink+Van Doorn, 77)



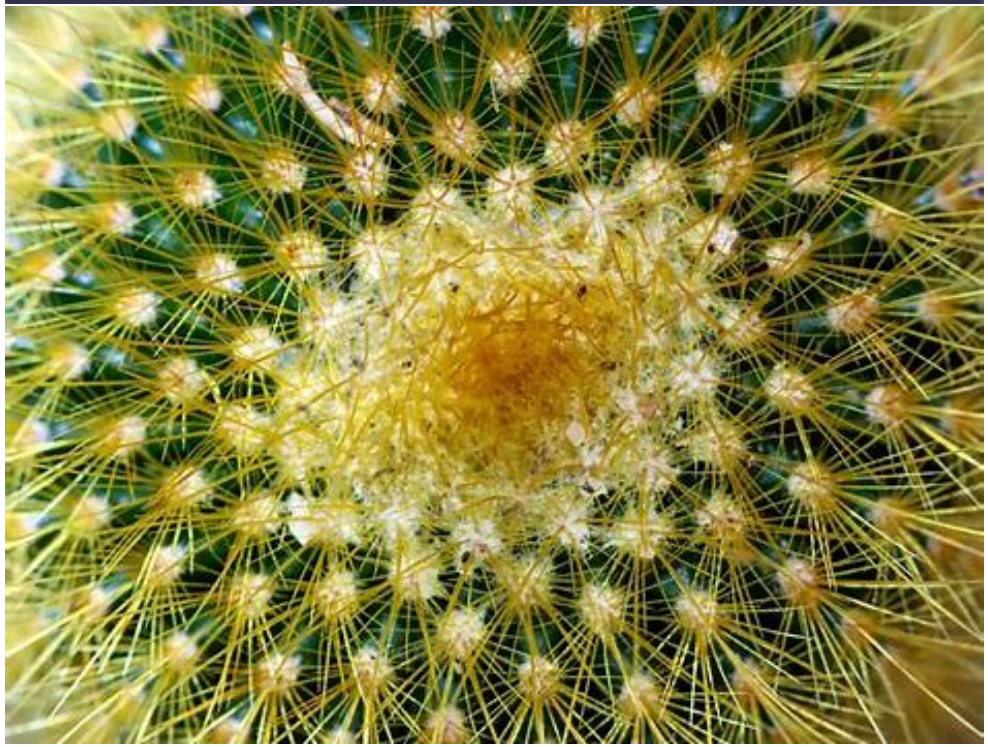


Fur - intrinsic, because
small local shadows do not
move with illumination
(at least Koenderink+Van Doorn, 77)

Relief - intrinsic (at least at this scale),
because small local shadows do not
move with illumination
(at least Koenderink+Van Doorn, 77)

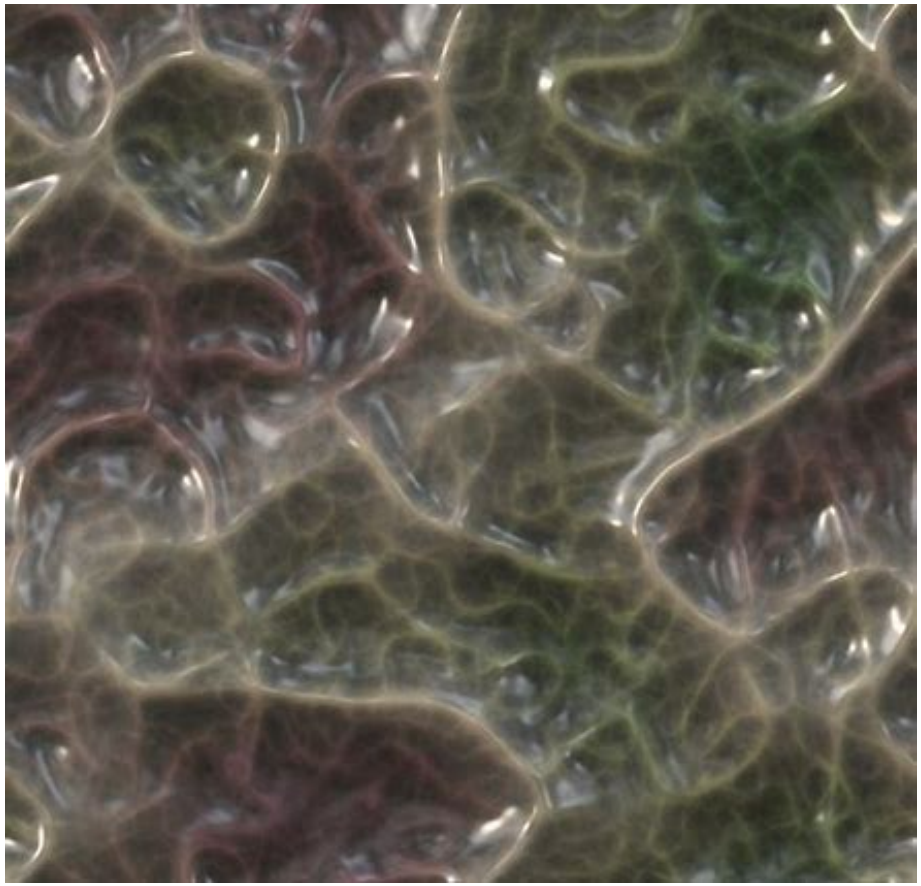


??? - intrinsic, because
mostly not a property of viewing
circumstances (?)

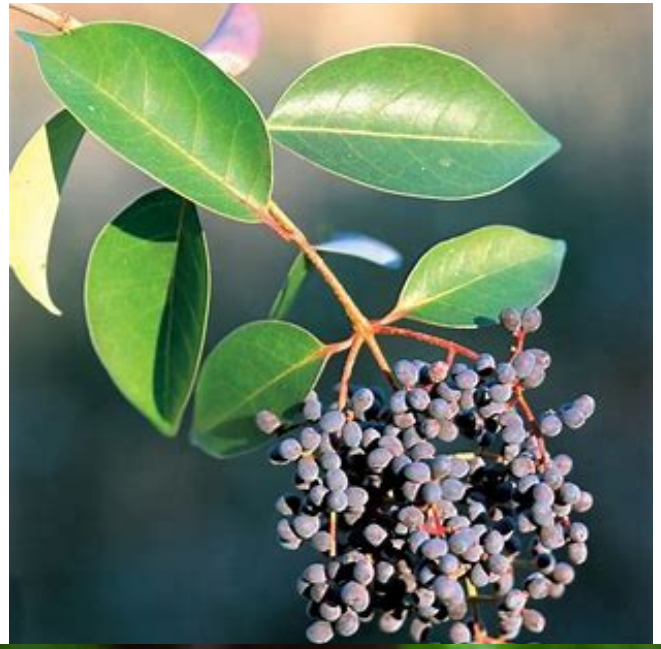


Iridescence
creating intrinsic gloss effects
intrinsic because the color effects will be
there for almost all illumination

??? - intrinsic, the specularities move but are always there



??? - intrinsic, the specularities move but are always there



SPADE - Images from label maps

Figure 1: Our model allows user control over both semantic and style as synthesizing an image. The semantic (e.g., the existence of a tree) is controlled via a label map (the top row), while the style is controlled via the reference style image (the leftmost column). Please visit our website for interactive image synthesis demos.

Question time

- Can pix2pix do this?
- Can cyclegan do this?

Pix2Pix HD - improving pix2pix

- pix2pix doesn't like high resolution
  - training tends to be unstable
- Fix 1
  - wrap an HR generator around an LR generator

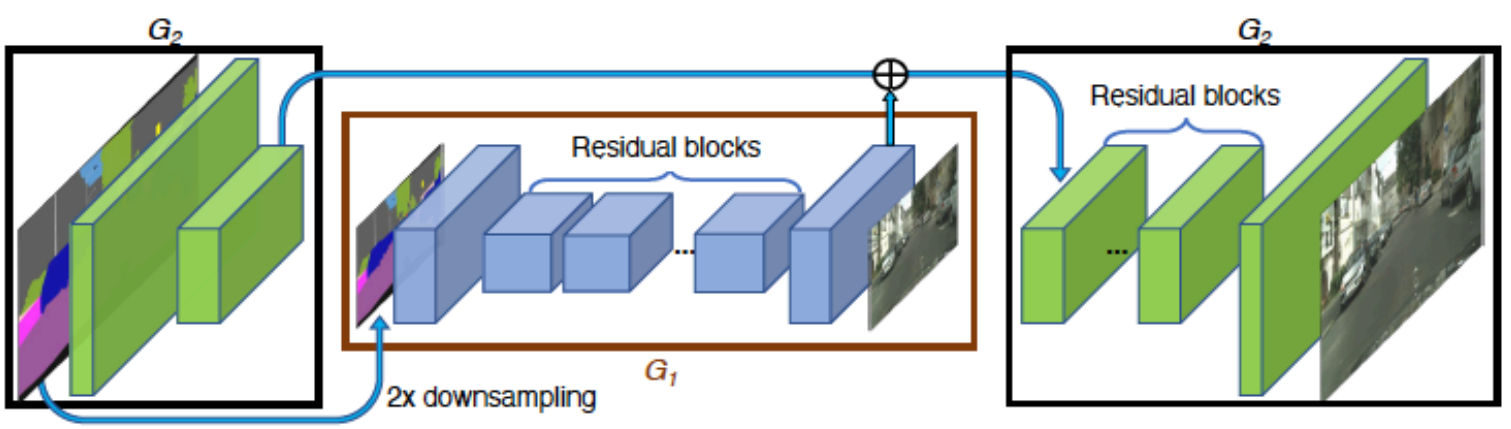


Figure 3: Network architecture of our generator. We first train a residual network G_1 on lower resolution images. Then, another residual network G_2 is appended to G_1 and the two networks are trained jointly on high resolution images. Specifically, the input to the residual blocks in G_2 is the element-wise sum of the feature map from G_2 and the last feature map from G_1.

Wang et al 18

Pix2Pix HD - improving pix2pix

- pix2pix doesn't like high resolution
  - training tends to be unstable
- Fix 2
  - match feature statistics in discriminator layers with extra loss

Improved adversarial loss We improve the GAN loss in Eq. (2) by incorporating a feature matching loss based on the discriminator. This loss stabilizes the training as the generator has to produce natural statistics at multiple scales. Specifically, we extract features from multiple layers of the discriminator and learn to match these intermediate representations from the real and the synthesized image. For ease of presentation, we denote the ith-layer feature extractor of discriminator D_k as $D_k^{(i)}$ (from input to the ith layer of D_k). The feature matching loss $\mathcal{L}_{\text{FM}}(G, D_k)$ is then calculated as:

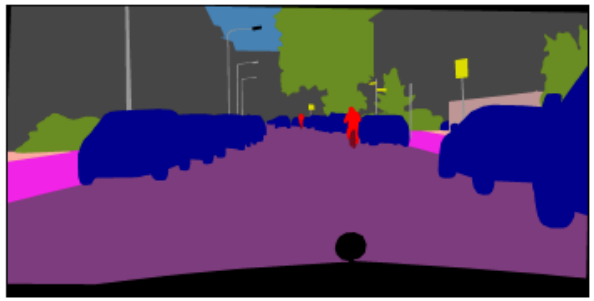
$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{(\mathbf{s},\mathbf{x})} \sum_{i=1}^{T} \frac{1}{N_i} [||D_k^{(i)}(\mathbf{s}, \mathbf{x}) - D_k^{(i)}(\mathbf{s}, G(\mathbf{s}))||_1], \tag{4}$$

where T is the total number of layers and N_i denotes the number of elements in each layer. Our GAN discriminator

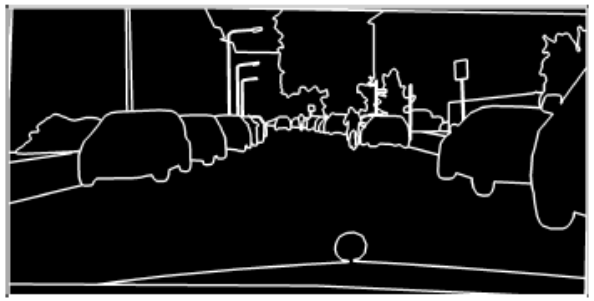
Wang et al 18

Pix2Pix HD - improving pix2pix

- pix2pix localization from label maps can be poor
  - because label maps don't delineate individual objects well
- Fix
  - use instance maps



(a) Semantic labels



(b) Boundary map

Figure 4: Using instance maps: (a) a typical semantic label map. Note that all connected cars have the same label, which makes it hard to tell them apart. (b) The extracted instance boundary map. With this information, separating different objects becomes much easier.

Wang et al 18

Images from label maps - Control

- General problem
  - Examples:
    - Label is sky, but what kind of sky? cloudy? overcast? etc
    - Label is car, but in what way is car 1 different from car 2? etc
  - Underlying issue:
    - label map has less information than image, so generator must create
      - possible solutions:
        - supply random numbers (but what about control)
        - supply input from some other feature process

Control in pix2pixHD

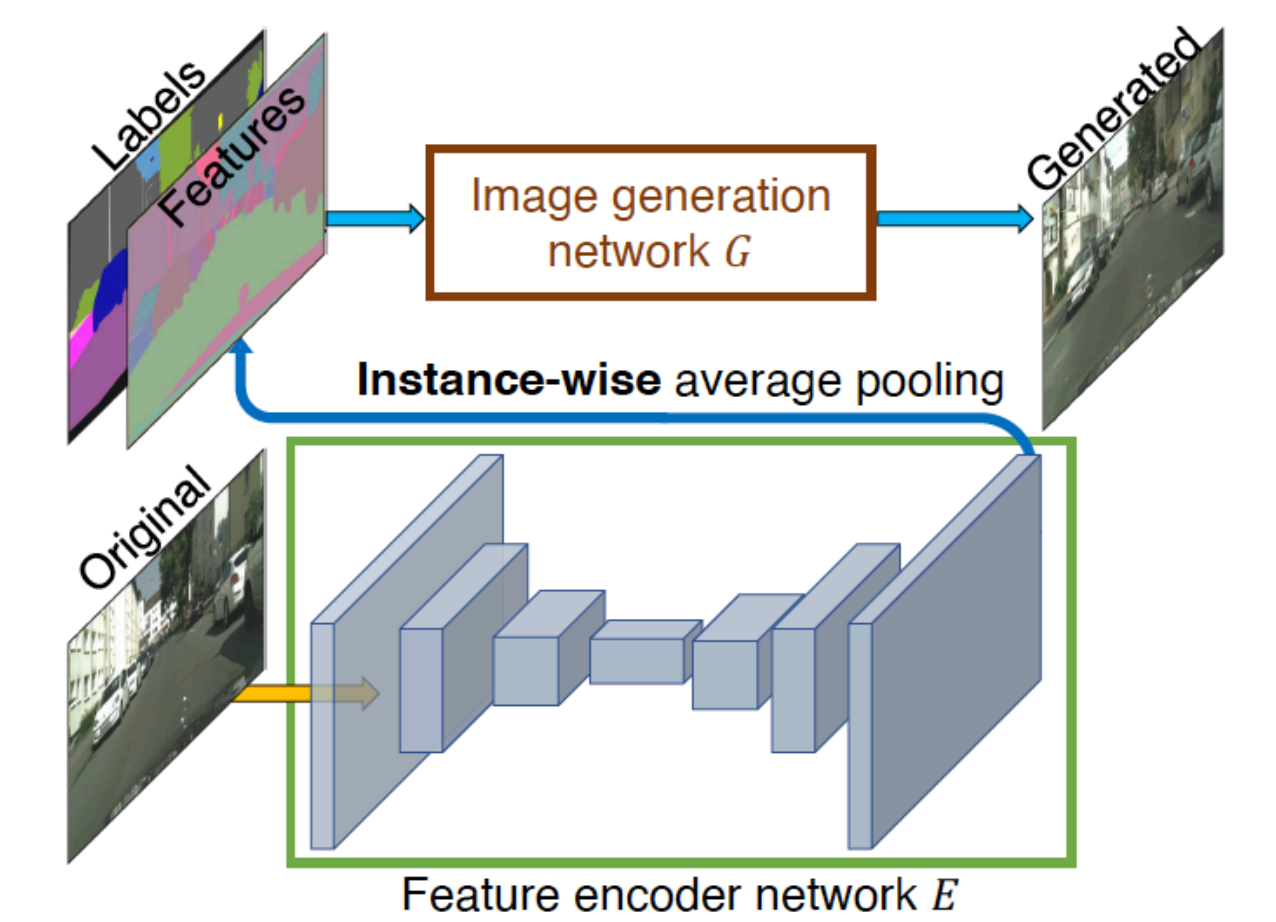


Figure 6: Using instance-wise features in addition to labels for generating images.

Wang et al 18

Evaluation

- pix2pix HD
  - check - when you apply a semantic segmenter to output, do you get input?
  - This is clearly not enough
    - Human study
      - tough to iterate
- Generators in general
  - Are generated images “like” real images?
    - FID:
      - compute an embedding of generated/real images
      - compare embeddings
    - Inception score
      - compare label predictions for some classifier for generated/real images
  - BOTH ARE HELPFUL, BUT DUBIOUS
    - among other things, they’re biased!

Bias in FID

2.1. Fréchet Inception Distance

To compute Fréchet Inception Distance, we pass generated and true data through an ImageNet [9] pretrained Inception V3 [36] model to obtain visually relevant features. Let (M_t, C_t) and (M_g, C_g) represent the mean and covariance of the true and generated features respectively, then compute

$$\text{FID} = ||M_t - M_g||_2^2 + Tr(C_t + C_g - 2(C_t C_g)^{\frac{1}{2}}) \quad (1)$$

FID seems to correspond well with human judgement of image quality and diversity [38].

This is an estimate of the squared difference of first and second moments

Embedding

$$M_t \approx \mathbb{E}_t(\mathbf{g}(\mathbf{x}))$$

So this term estimates the difference in means

Chong et al 20

Bias in FID

Remember, the average of a set of points in an embedding space is an estimate of an integral

$$\frac{1}{N}\sum_i \mathbf{g}(\mathbf{x}_i) \approx \mathbb{E}_p(\mathbf{g}(\mathbf{x})) = \int \mathbf{g}(\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

The mean and covariance used in estimating FID_N are Monte Carlo estimates of integrals (the relevant expectations). The terms M_t and C_t computed on true images are not random, as proper comparisons fix the set of true images used. However, the terms M_g and C_g are random – if one uses different samples to evaluate these terms, one gets different values. A Monte Carlo (MC) estimate of an integral $\int h(x)p(x)dx$ whose true value is $I(h)$, made using N IID samples, yields $\hat{I} = I + \xi$, where

$$\mathbb{E}[\xi] = 0 \text{ and } \text{var}(\xi) = \frac{C(h)}{N} \quad (3)$$

where $C(h) \geq 0$ is $\int (h(x) - I(h))^2 p(x)dx$ [6]. Note the value of C is usually very hard to estimate directly, but C is non-negative and depends strongly on the function being integrated. A key algorithmic question is to identify pro-

Chong et al 20

Bias in FID

Now consider some function G of a Monte Carlo integral $I(h)$, where G is sufficiently smooth. We have

$$G(\hat{I}) = G(I+\xi) \approx G(I) + \xi G^{'}(I) + \xi^2 \frac{G^{''}(I)}{2} + O(\xi^3) \tag{5}$$

so that

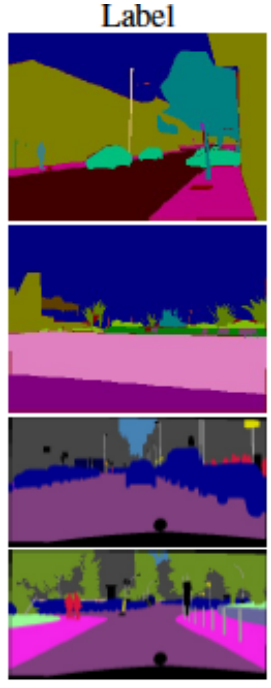
$$\mathbb{E}\left[G(\hat{I})\right] = G(I) + \frac{K}{N} + O(1/N^2) \tag{6}$$

where $K = C(h)\frac{G^{''}(I)}{2}$ and $\frac{K}{N}$ is bias.

Consider an estimate of FID, estimated with N samples. The terms M_g and C_g are estimated with an MC integrator, so the estimate must have a bias of $\frac{C_F}{N} + O(1/N^2)$. Note that C_F must *depend on the generator g* (section 2.3).

Chong et al 20

Pix2PixHD can map labels to images



But notice a certain squashiness in “stuff”.

Two issues here:

- “stuff” is missing information as well, like instances
- normalization in the network makes this worse

Park et al 19

Normalization can suppress detail

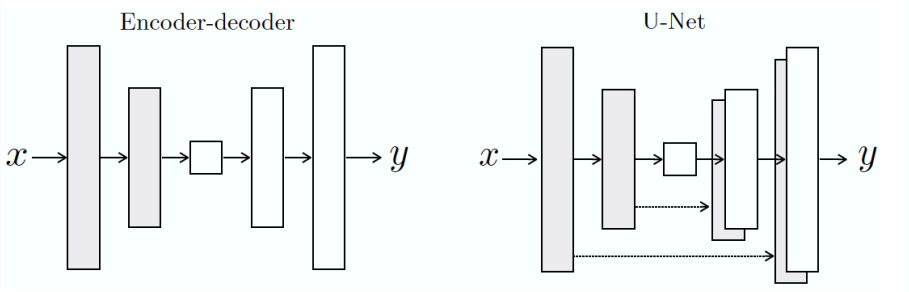


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Isola et al, 16 (MUST READ)

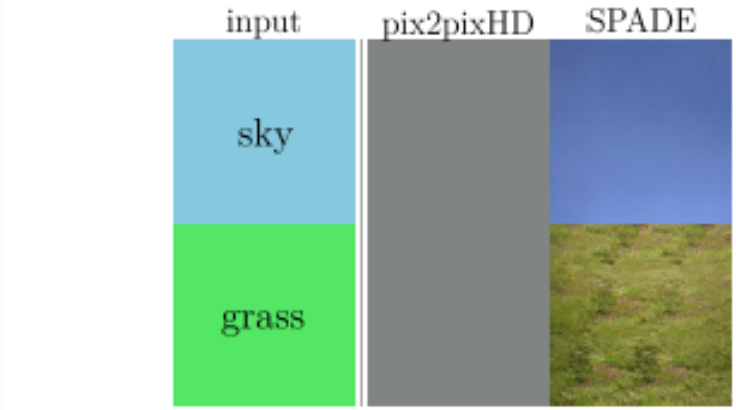


Figure 3: Comparing results given uniform segmentation maps: while the SPADE generator produces plausible textures, the pix2pixHD generator [48] produces two identical outputs due to the loss of the semantic information after the normalization layer.

- Simple example:
  - instance norm in layers
  - single label in input
  - instance norm makes input 0
    - we've lost everything!

Park et al 19

Pix2PixHD is weirdly inefficient

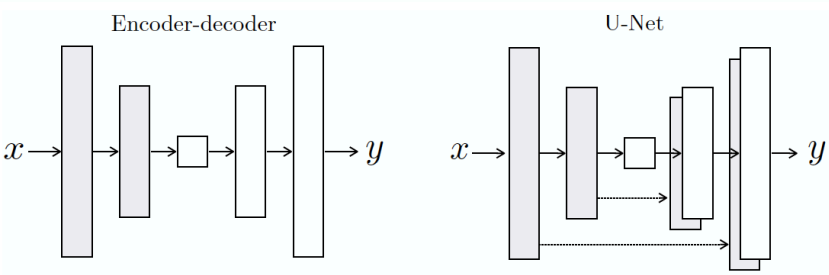


Figure 3: Two choices for the architecture of the generator. The "U-Net" [50] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Isola et al, 16 (MUST READ)

- Long scale embedding of a label map doesn't make sense
  - and is expensive

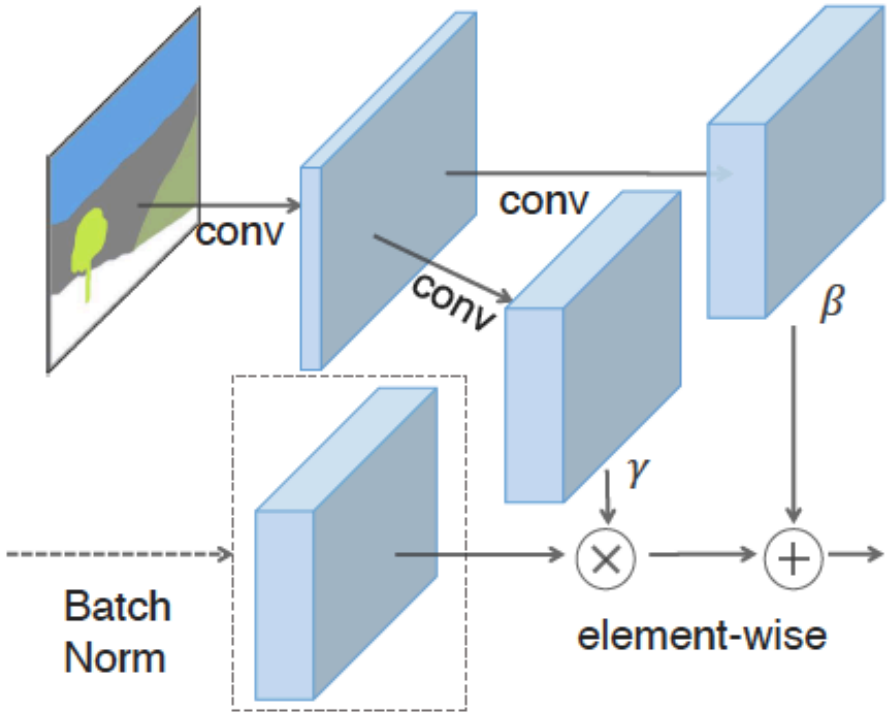


Figure 2: In the SPADE, the mask is first projected onto an embedding space and then convolved to produce the modulation parameters γ and β. Unlike prior conditional normalization methods, γ and β are not vectors, but tensors with spatial dimensions. The produced γ and β are multiplied and added to the normalized activation element-wise.

Park et al 19

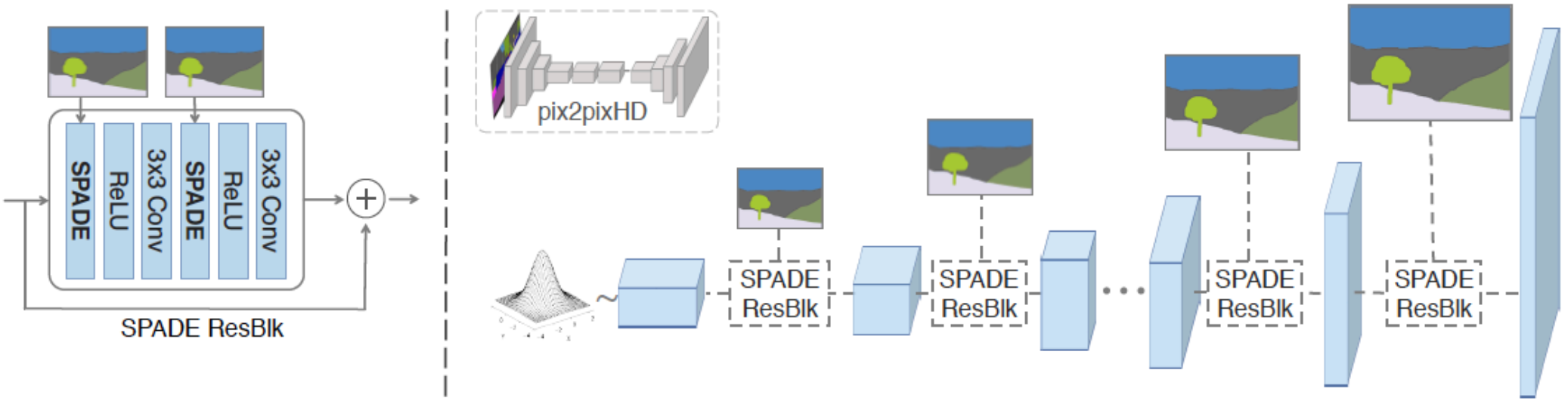


Figure 4: In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations. *(left)* Structure of one residual block with the SPADE. *(right)* The generator contains a series of the SPADE residual blocks with upsampling layers. Our architecture achieves better performance with a smaller number of parameters by removing the downsampling layers of leading image-to-image translation networks such as the pix2pixHD model [48].

Park et al 19

Figure 8: Semantic image synthesis results on COCO-Stuff. Our method successfully generates realistic images in diverse scenes ranging from animals to sports activities.

Park et al 19

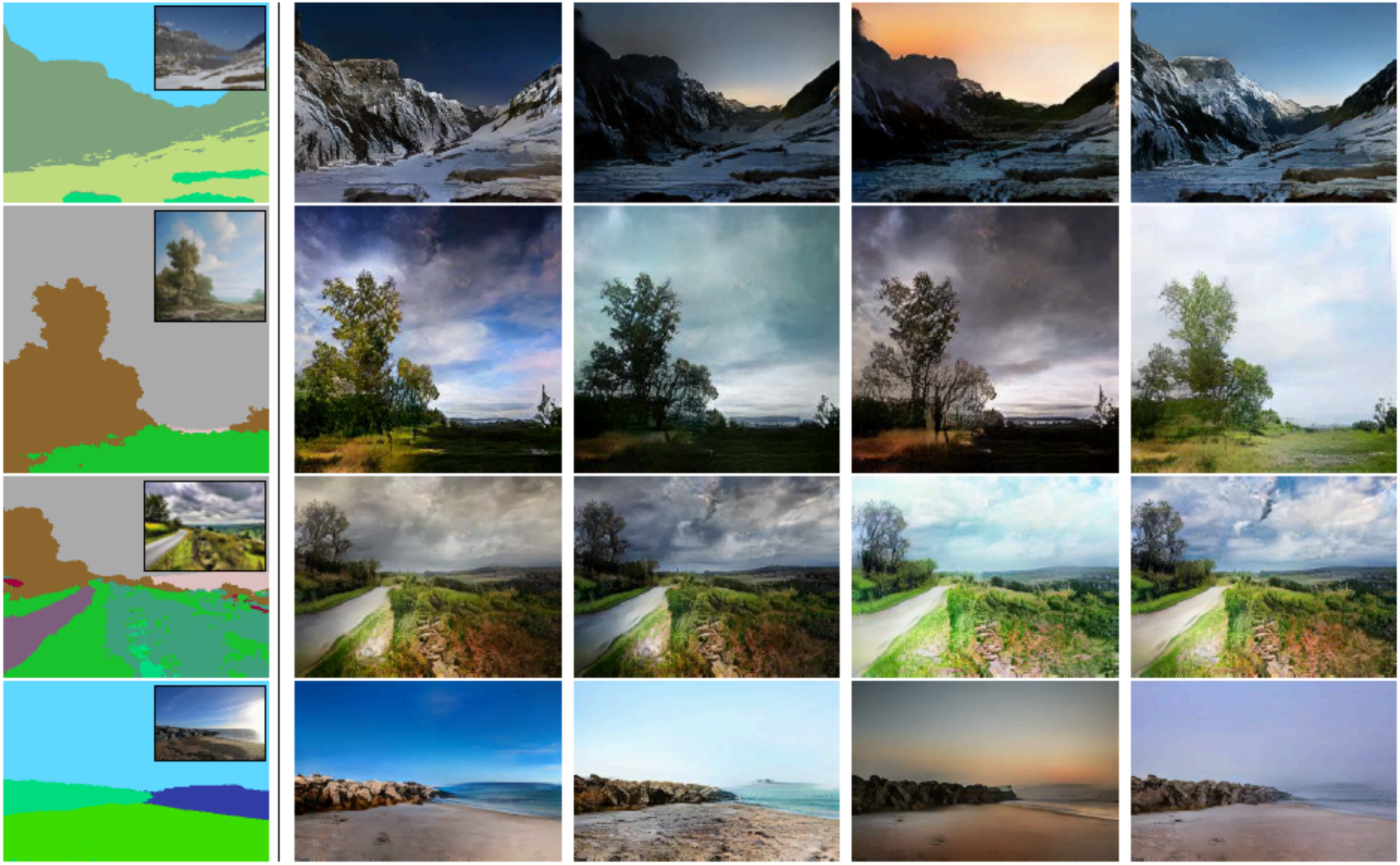


Figure 9: Our model attains multimodal synthesis capability when trained with the image encoder. During deployment, by using different random noise, our model synthesizes outputs with diverse appearances but all having the same semantic layouts depicted in the input mask. For reference, the ground truth image is shown inside the input segmentation mask.

Park et al 19

Control

- SPADE offers
  - segment label based generation
  - variability
- But how do we control?
  - Very little seems known
- Some strategies:
  - create appearance abstractions (time of day, etc) and use rejection sampling
  - generate conditioned on appearance abstractions
  - require pictures be “like this” (but how? and what this?
- Important obstacle
  - object appearance is fantastically complex in detail
  - and there isn’t a good “vocabulary” for describing it

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
- Reshade and relight
- Exaggerate effects
  - eg motion fields

We've seen a bunch of this

- NeRF
- IBR
- One more idea - PixelNeRF

PixelNeRF

- Key Problem with NeRF (that isn't integration related!)
  - No sharing - each NeRF is its own thing
  - This is genuinely weird
    - 3D models of things should be "like one another"
      - eg densities should be locally either low or high
      - eg some spatial densities should be more common than others
  - Rendering and model building are profoundly integrated
    - but why?

PixelNeRF

This mapping is independent of image, viewpoint

This mapping is independent of image, viewpoint

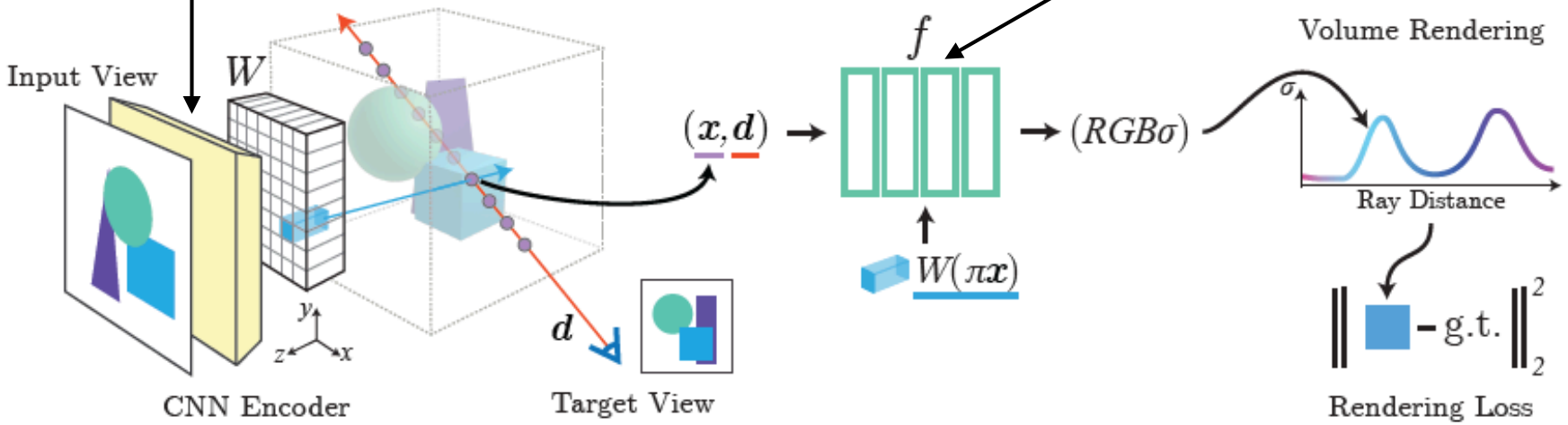


Figure 2: **Proposed architecture in the single-view case.** For a query point x along a target camera ray with view direction d, a corresponding image feature is extracted from the feature volume W via projection and interpolation. This feature is then passed into the NeRF network f along with the spatial coordinates. The output RGB and density value is volume-rendered and compared with the target pixel value. The coordinates x and d are in the camera coordinate system of the input view.

Yu et al 20

PixelNeRF - a second view

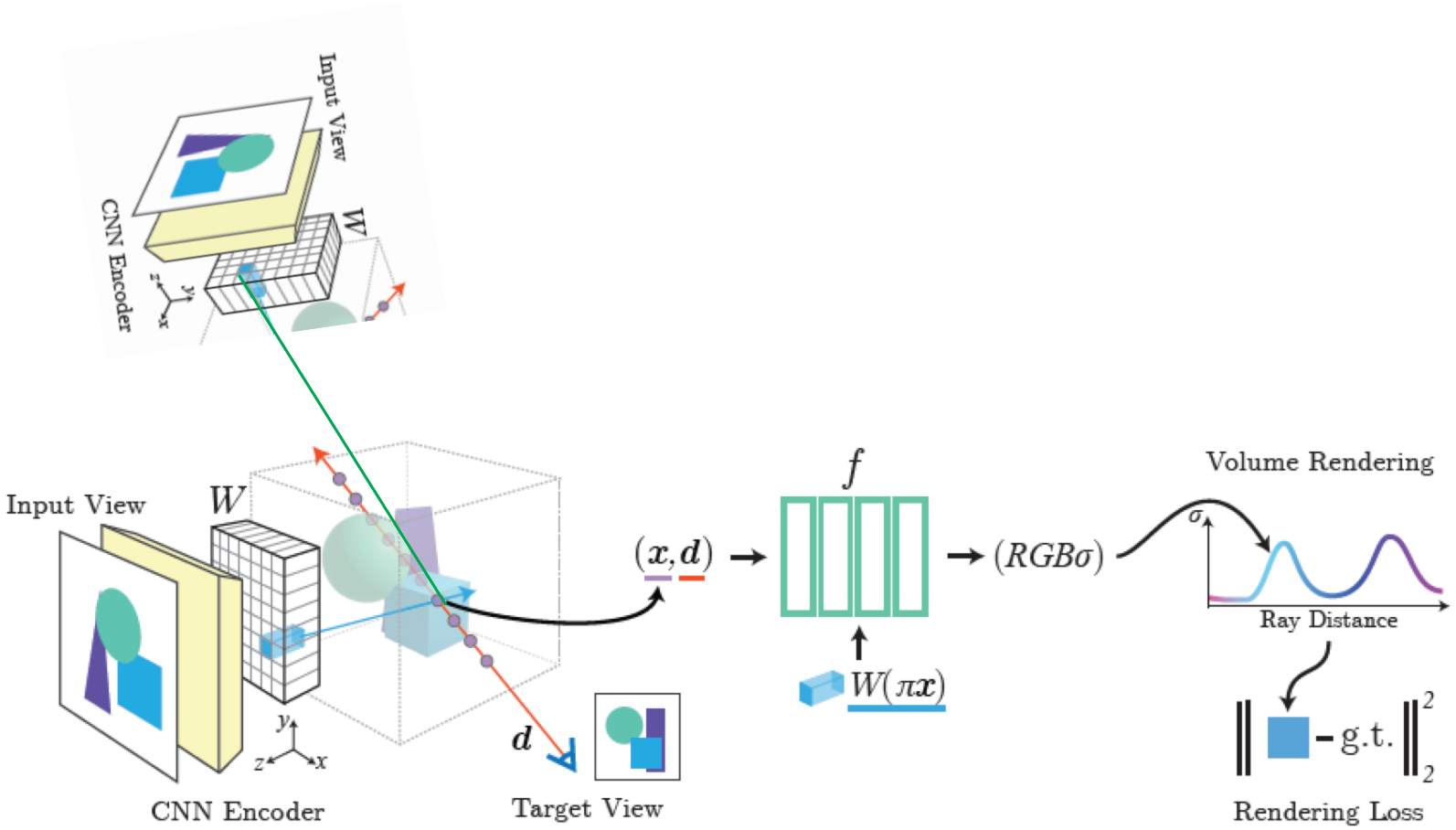


Figure 2: **Proposed architecture in the single-view case.** For a query point x along a target camera ray with view direction d, a corresponding image feature is extracted from the feature volume W via projection and interpolation. This feature is then passed into the NeRF network f along with the spatial coordinates. The output RGB and density value is volume-rendered and compared with the target pixel value. The coordinates x and d are in the camera coordinate system of the input view.

Yu et al 20

PixelNeRF

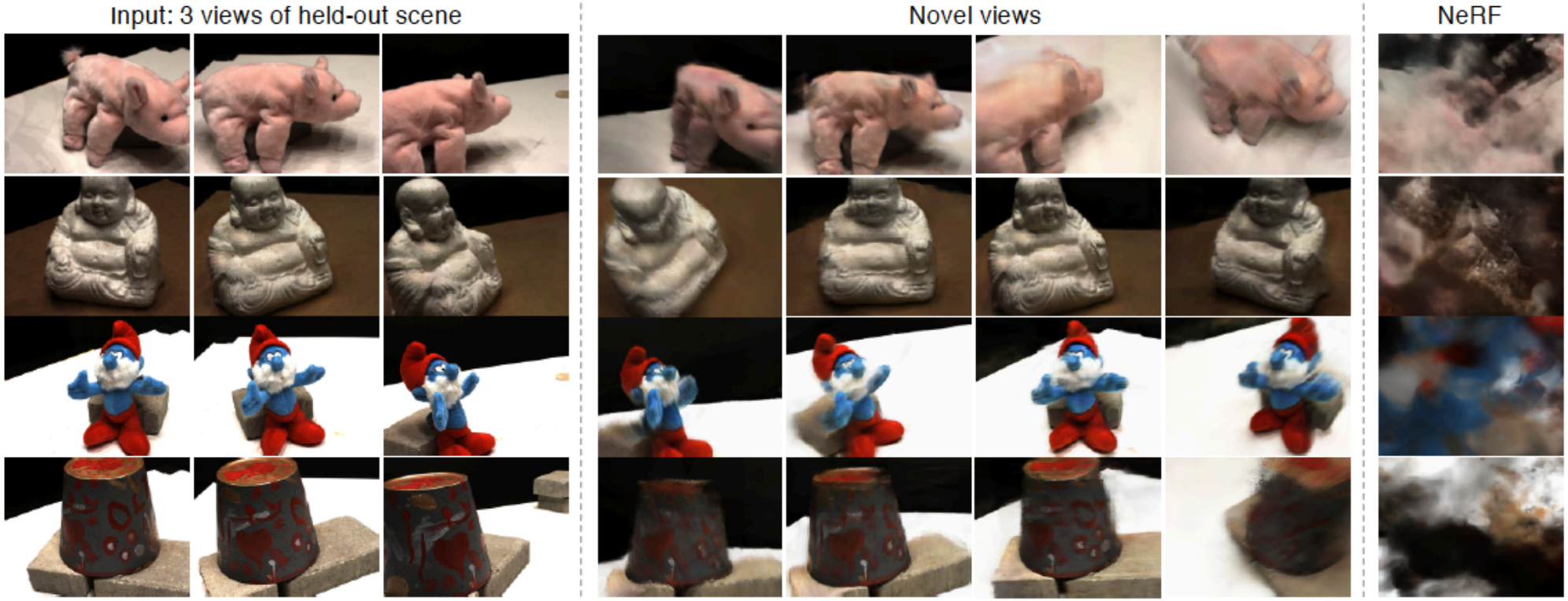


Figure 9: **Wide baseline novel-view synthesis on a real image dataset.** We train our model to distinct scenes in the DTU MVS dataset [12]. Perhaps surprisingly, even in this case, our model is able to infer novel views with reasonable quality for held-out scenes without further test-time optimization, all from only three views. Note the train/test sets share no overlapping scenes.

Yu et al 20

Still missing some features ….

- Something like an image prior is missing
  - local constraint on reconstruction fields
  - Q: how to supply?
- Something like adversarial training is missing
  - rendering too slow
  - Q: how to supply?

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
- Reshade and relight
- Exaggerate effects
  - eg motion fields