Reshading and relighting

D.A. Forsyth

Goals and representations

- Generate a relighted/reshaded image
    - Goal cases:
        - Change lighting on a fixed scene
        - Insert an object into a scene and relight
    - Data cases:
        - Comprehensive scene/object data
        - Minimal scene/object data

Ramamoorthi + Hanrahan 01

Basic machinery

- Q1: Can one infer illumination from reflected light?
  - A1: Not much for diffuse surfaces (Ramamoorthi+Hanrahan, 01)
  - A2: Illumination and BRDF under some conditions if all reflected directions are observed (R+H, 04)
- Light transport
  - A representation of a scene in terms of what it does to light

Ramamoorthi + Hanrahan 01

Illumination from reflected light

- Assume
  - illumination depends only on angle
  - surface is diffuse, convex
  - can measure radiosity at all points on surface
-

Illumination (what we want, doesn't depend on x)

$$E(\mathbf{x}) = \int_{\Omega'} L(\mathbf{x}, \theta_i', \phi_i') \cos\theta_i' \, \mathrm{d}\Omega',$$

Radiosity (what we observe) ⟶ $B(\mathbf{x}) = \rho E(\mathbf{x}),$

Known

But actually…..

$$E(\mathbf{n}) = \int_{\Omega'} L(\theta_i, \phi_i) \cos\theta_i' \, \mathrm{d}\Omega'.$$

Ramamoorthi + Hanrahan 01

Illumination from reflected light - II

- Angles in integral are in the local frame of the point
- So

$$E(n) = \int \text{Rotation(fixed illumination field)} d\Omega$$

- Where
  - the rotation is given by n,
  - the integral is over the hemisphere
- Notice analogy with filtering

Ramamoorthi + Hanrahan 01

Illumination from reflected light - III

- Our problem:

$$E(n) = \int \text{Rotation(fixed illumination field)} d\Omega$$

Linear, rotation invariant operator, takes illumination field and makes E

- Filtering

$$G(u) = \int H(x-u) I(x) dx$$

Linear, translation invariant operator, takes I and makes G

Ramamoorthi + Hanrahan 01

Illumination from reflected light - III

- Filtering

  Linear, translation invariant operator, takes I and makes G

  - often convenient to represent in a different basis
  - Fourier transform - filtering (convolution)-> multiplication

- Our problem:

  Linear, rotation invariant operator, takes illumination field and makes E

  - actually is a form of convolution
  - basis is spherical harmonics
    - (analogous with fourier series, sphere is compact)
  - in this basis, our operation is apply a KNOWN linear operator

Ramamoorthi + Hanrahan 01

Illumination from reflected light - IV

- Spherical harmonics
    - multiple definitions
    - my favorite is
        - monomials restricted to the sphere factored by terms that vanish
        - sometimes normalized, etc.
        - eg
            - 1, x, y, z, x^2, xy, xz, y^, yz, z^2
                - on surface of sphere
                - but note on surface of sphere, x^2+y^+z^2=1
                    - so there are actually only 9 basis functions
                    - degree == order in some notations (paper)

Ramamoorthi + Hanrahan 01

Key points

- You can’t get odd degree SH’s greater than one
- You can’t really get more than the first 9 SH’s

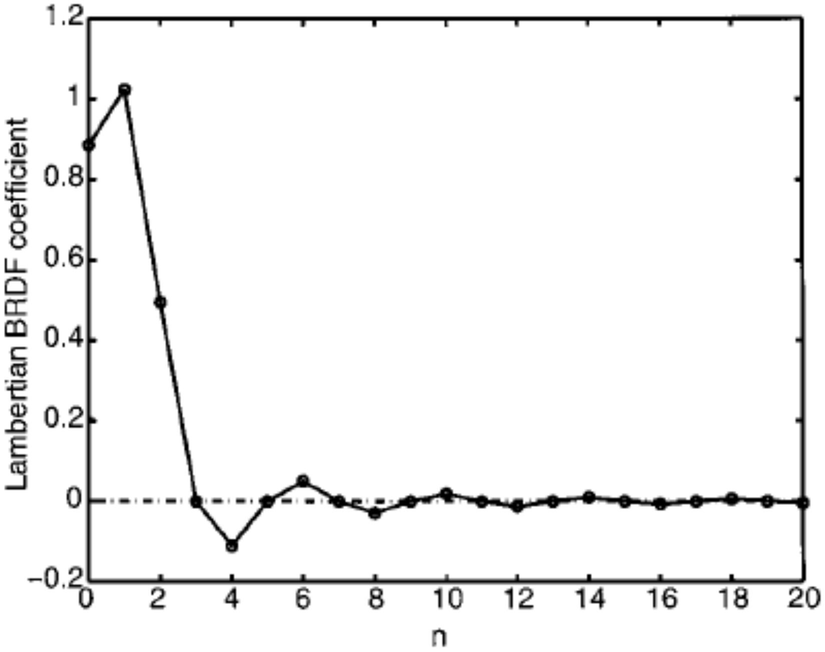


Fig. 3. The solid curve is a plot of A_n versus n. It can be seen that odd terms with $n > 1$ have $A_n = 0$. Also, as n increases, the coefficients rapidly decay.

Ramamoorthi + Hanrahan 01

Illumination and BRDF from reflection

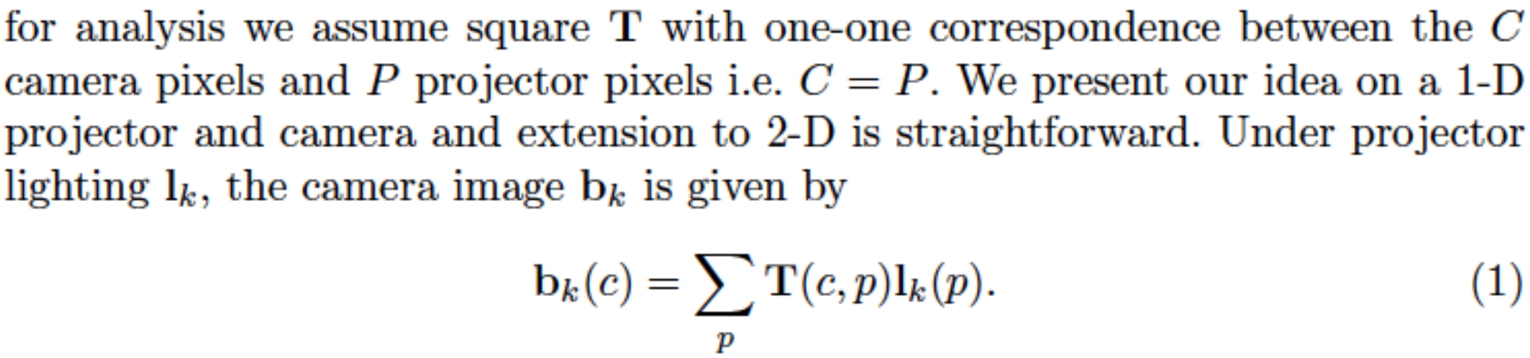
- Assume:
  - you can measure illumination along each direction at each point
  - object has unknown BRDF
  - object is immersed in angular illumination field
- You can:
  - recover both BRDF coefficients and illumination field
    - essentially, expand in spherical harmonics, and jockey terms
- Important, but difficult to use
  - see assumptions

Ramamoorthi+Hanrahan 04

Light transport operators

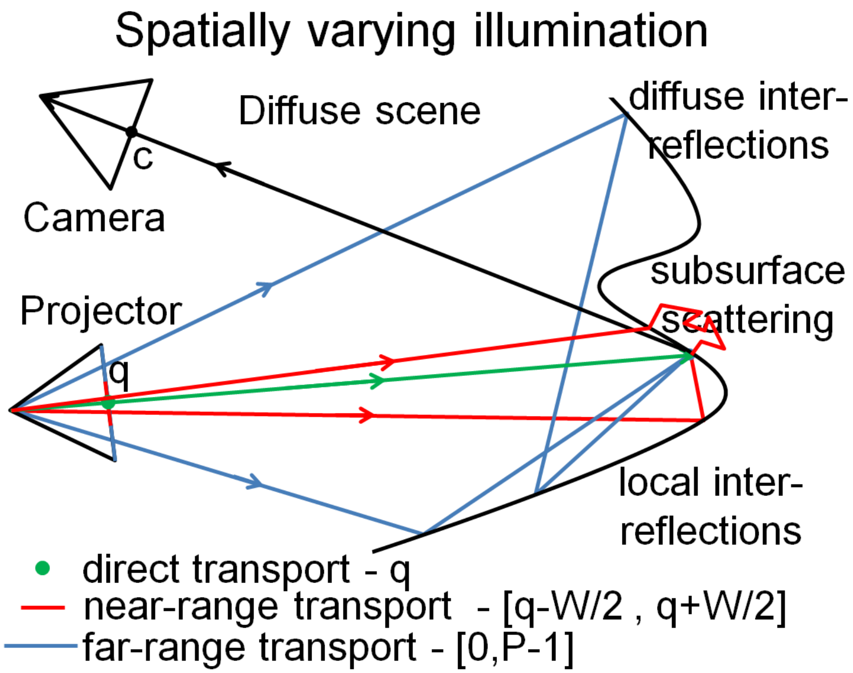
- Think of a scene as an operator that maps
  - Input illumination to camera intensities
  - This is linear
    - Q: why?
- Can we:
  - measure it?
  - interpolate it?
- Brute force measurement is hard
  - but do-able;
    - stick in tons of projectors, tons of cameras
    - record mapping from projector to camera
    - done

Light transport operators



for analysis we assume square $\mathbf{T}$ with one-one correspondence between the C camera pixels and P projector pixels i.e. $C = P$. We present our idea on a 1-D projector and camera and extension to 2-D is straightforward. Under projector lighting $\mathbf{l}_k$, the camera image $\mathbf{b}_k$ is given by

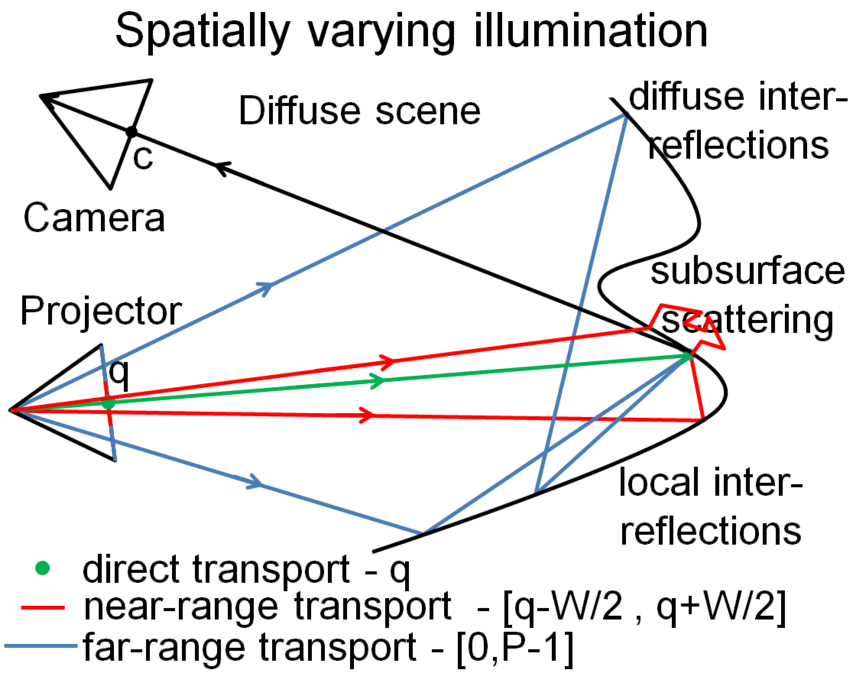
$$\mathbf{b}_k(c) = \sum_p \mathbf{T}(c,p)\mathbf{l}_k(p). \quad (1)$$


Spatially varying illumination
Diffuse scene
c
Camera
diffuse inter-reflections
subsurface scattering
Projector
q
local inter-reflections
● direct transport - q
near-range transport - [q-W/2 , q+W/2]
far-range transport - [0,P-1]

Reddy et al 12

Light transport operators

- BUT
  - the operator has a lot of structure
  - eg
    - decompose into
      - direct transport
        - “diagonal” in appropriate repn
      - near range transport
        - few non-zero “off-diagonal”
      - far range transport
        - low rank, smallish
- Consequence
  - reduced measurement by a factor of 15


Spatially varying illumination
Diffuse scene
diffuse inter-reflections
c
Camera
subsurface scattering
Projector
q
local inter-reflections
● direct transport - q
— near-range transport - [q-W/2 , q+W/2]
— far-range transport - [0,P-1]

Reddy et al 12

Reshading cases

- Inserting a CG object
  - we know shape+material of the thing to be inserted
  - Big issue:
    - what illumination field to recover, and how to recover it?
- Cut-and-paste
  - we know very little about the thing to be inserted
  - Big issue:
    - how to fake convincing illumination field
- Object recovery
  - we see multiple images and want a reshadable representation
  - Big issue:
    - how to recover transport operator efficiently

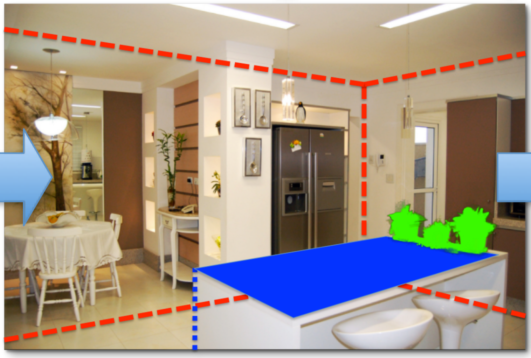
Inserting objects

- Debevec 98:
  - Image scene with HDR measurement object (reflective sphere)
  - Use sphere intensities to recover environment map
  - Render CG object using environment map, composite in

Inserting Computer Graphics

Input image

Estimate geometry



Using above, with manual help

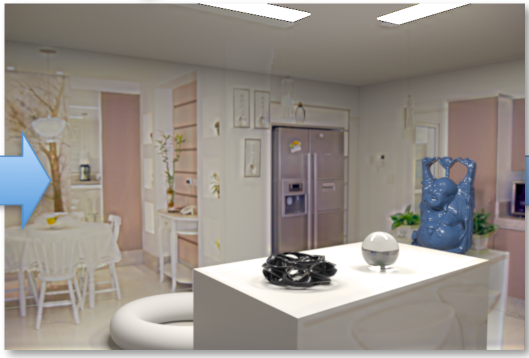
Estimate materials

Standard methods (Land 71)

Estimate lighting

Markup, for the moment
Secret sauce: Consistency
Secret sauce: Shafts

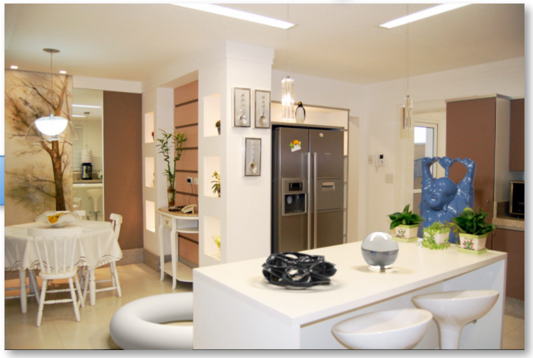
Compose & render



Secret sauce: Physical renderer

Karsch ea 11

Final composite

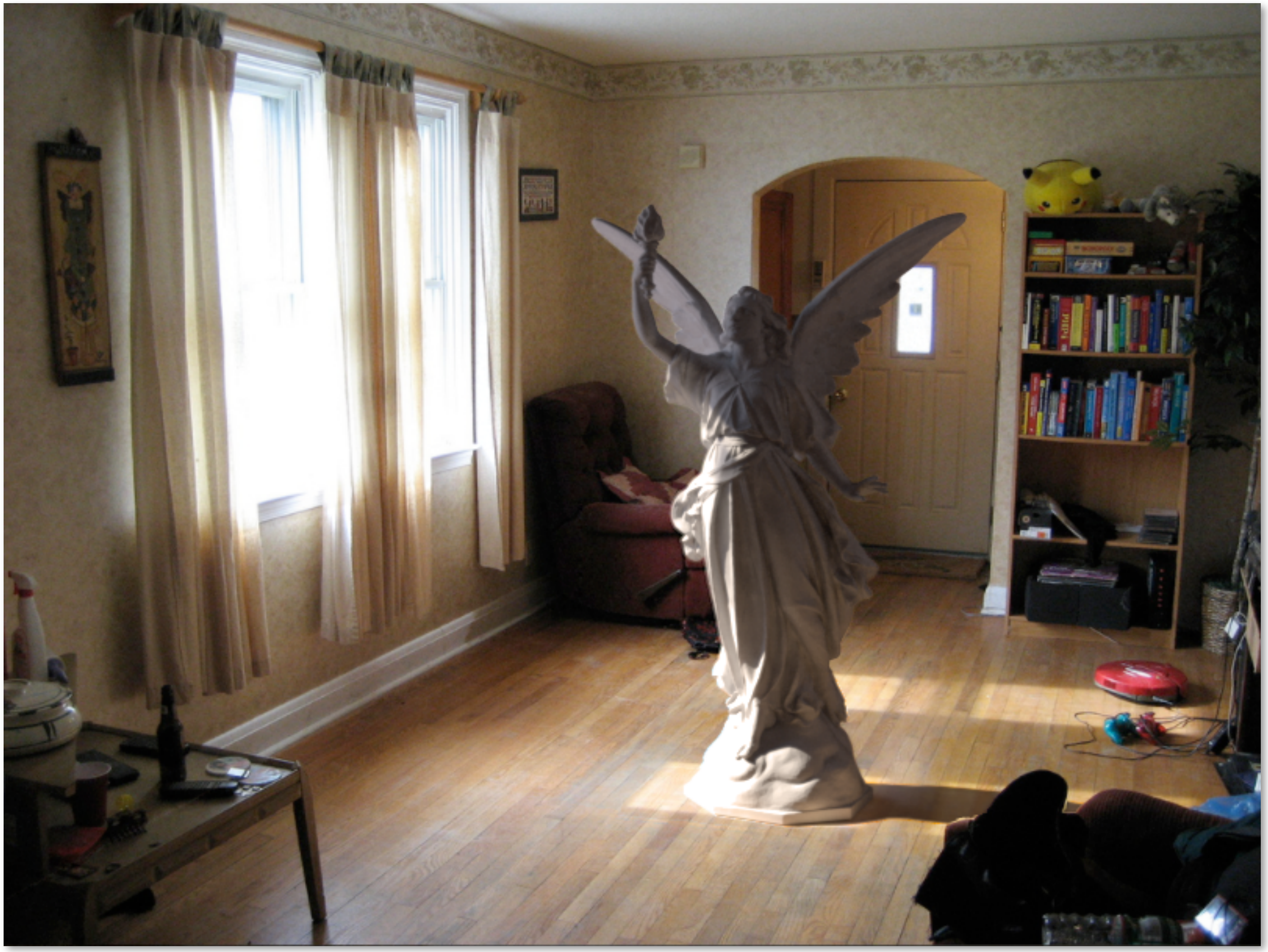


Compositing by standard
method Debevec 98

Results

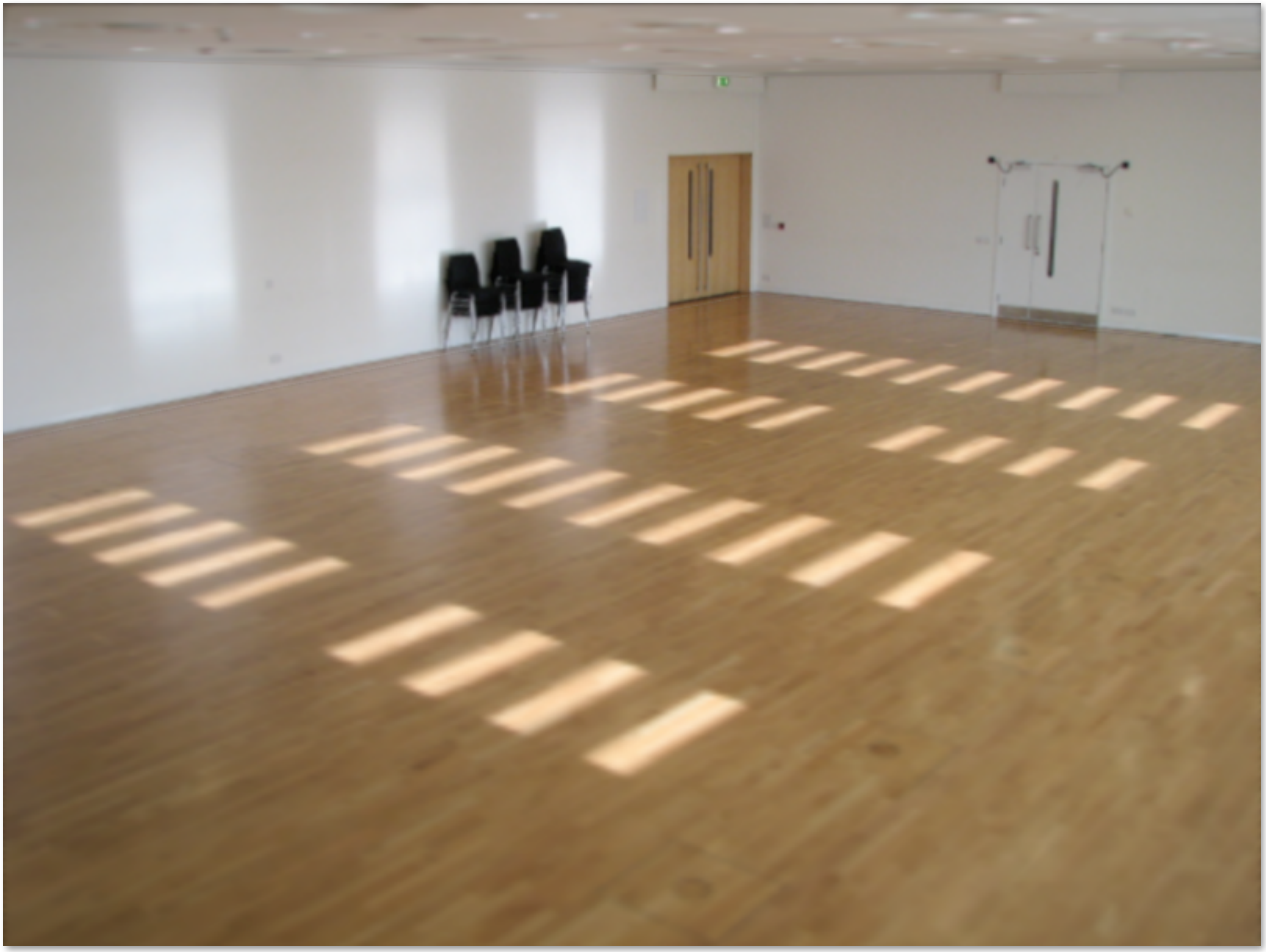
Karsch ea 11

Results



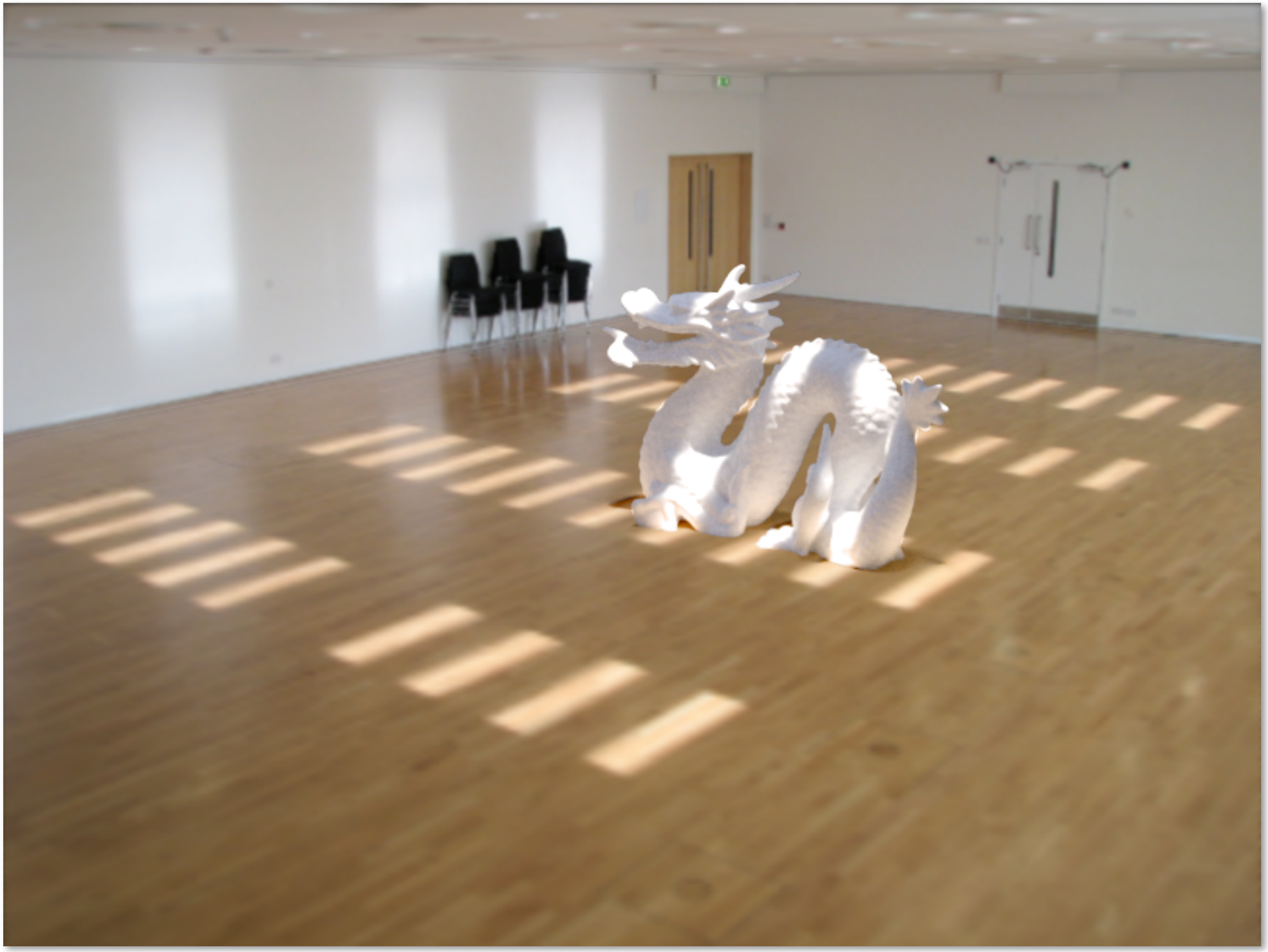
Karsch ea 11

Results



Karsch ea 11

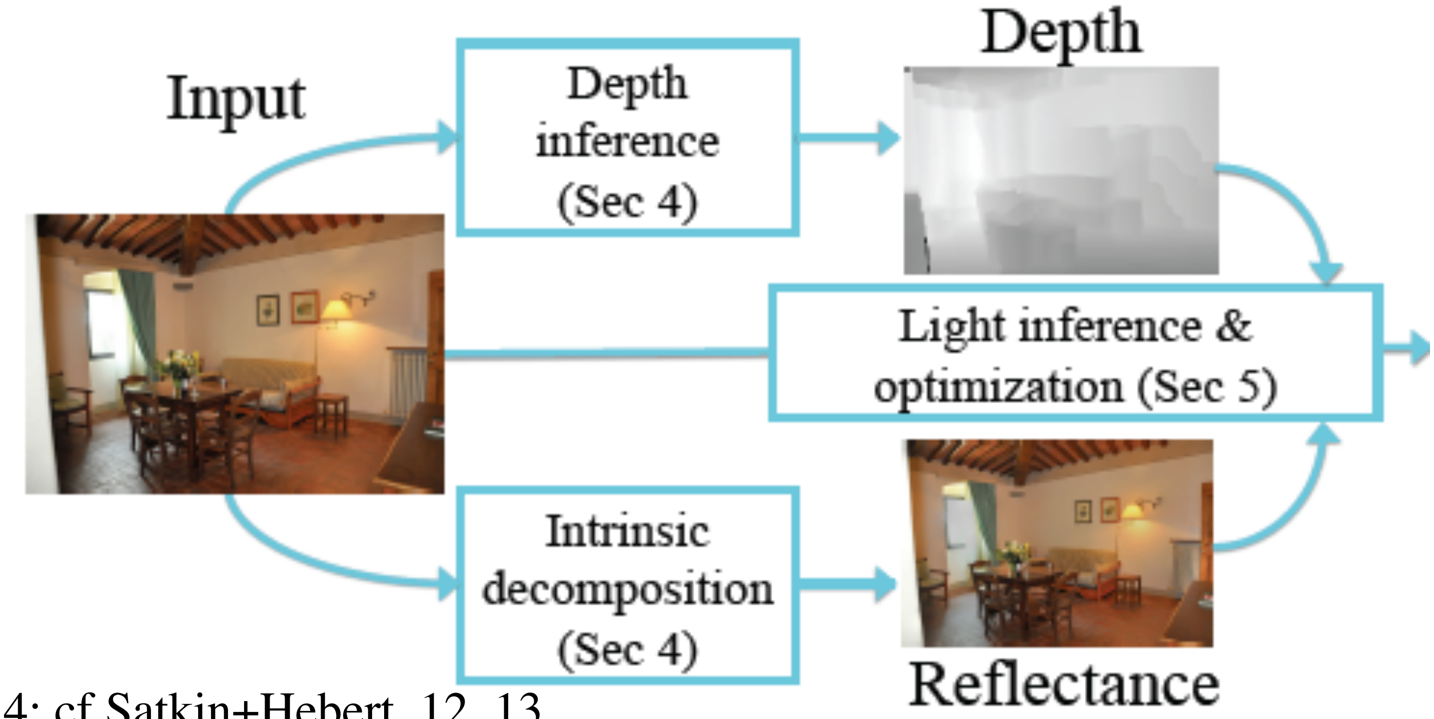
Results



Karsch ea 11

Making it automatic

- Depth map by matching to RGBD images
  - and various clever tricks to make the map render well, etc.
- Illumination by matching to images with known illum.
  - and various clever tricks to slide sources around, etc.


Input
Depth inference (Sec 4)
Depth
Light inference & optimization (Sec 5)
Intrinsic decomposition (Sec 4)
Reflectance

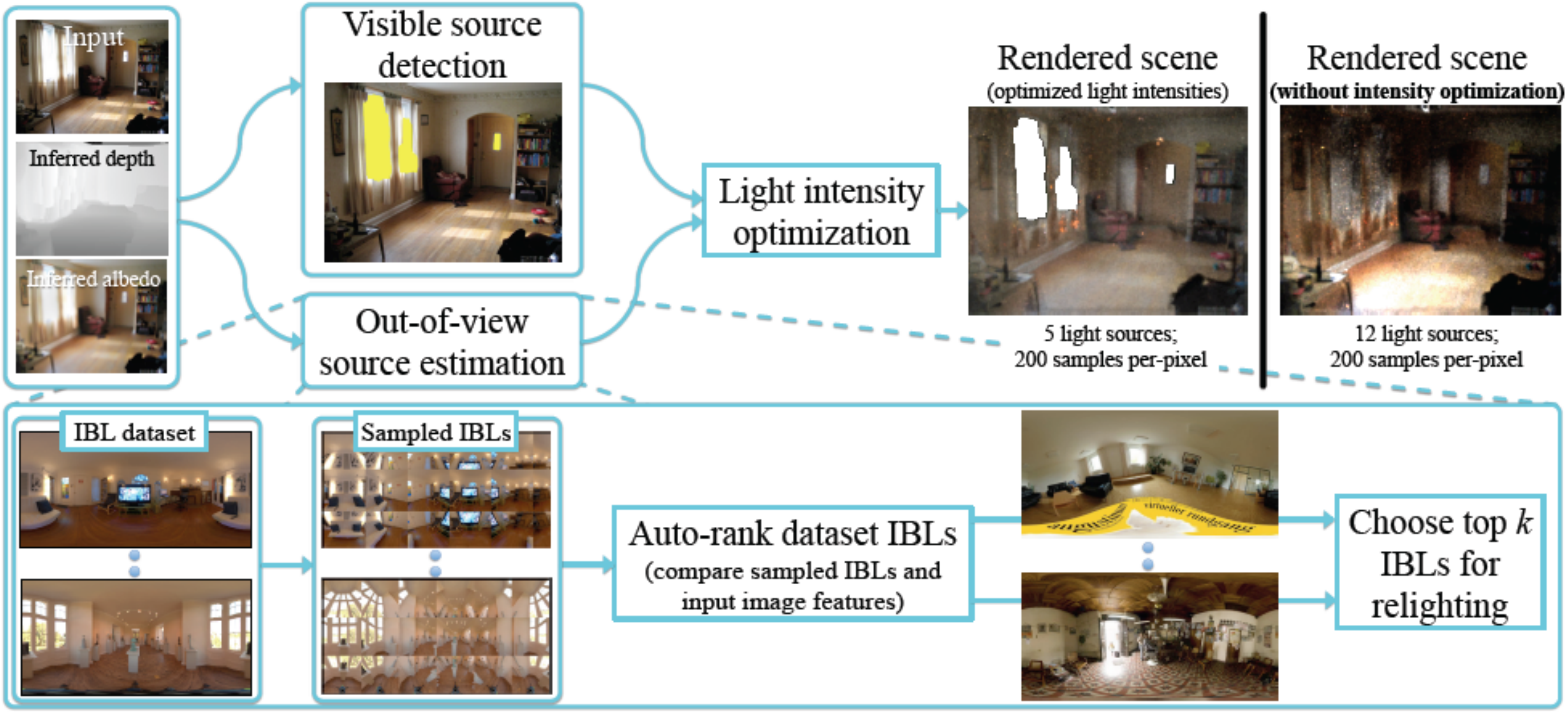
Karsch ea 14; cf Satkin+Hebert, 12, 13

Lighting by Matching

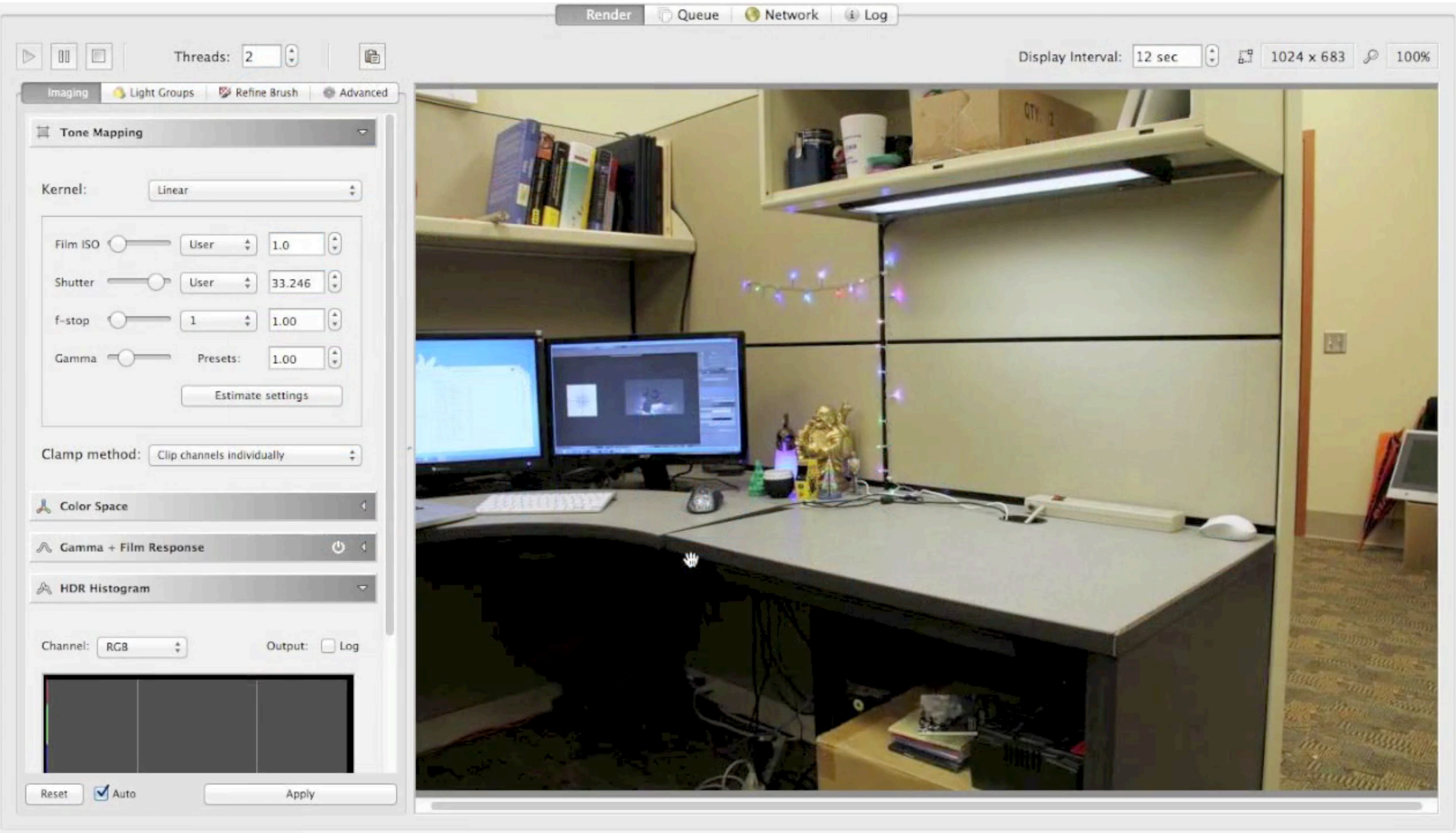
- What about invisible sources?
  - Lighting can be divided into:
    - sources:
      - high intensity luminaires
      - other stuff that is lower intensity
    - visibility:
      - can be seen
      - can't be seen
- Strategy:
  - Visible luminaires are easy
  - Invisible luminaires by matching
  - Q: what about invisible, low intensity sources?
    - A: ignore

Karsch ea 14

Lighting by Matching



Karsch ea 14



Threads: 2

Display Interval: 12 sec

1024 x 683

100%

Kernel: Linear

Film ISO User 1.0

Shutter User 33.246

f-stop 1 1.00

Gamma Presets: 1.00

Clamp method: Clip channels individually

Channel: RGB

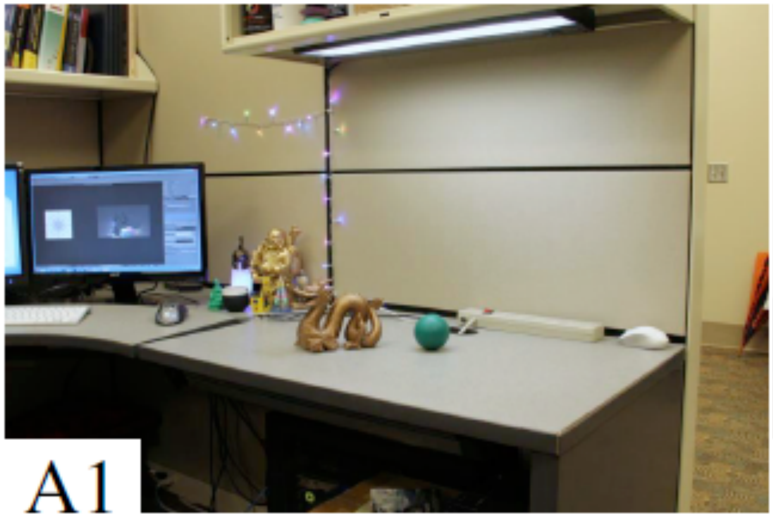
Output: □ Log

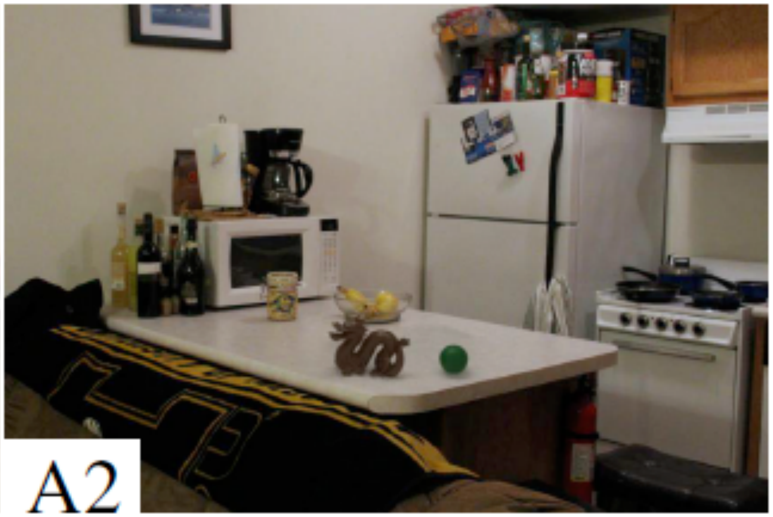
☑ Auto

Karsch ea 14

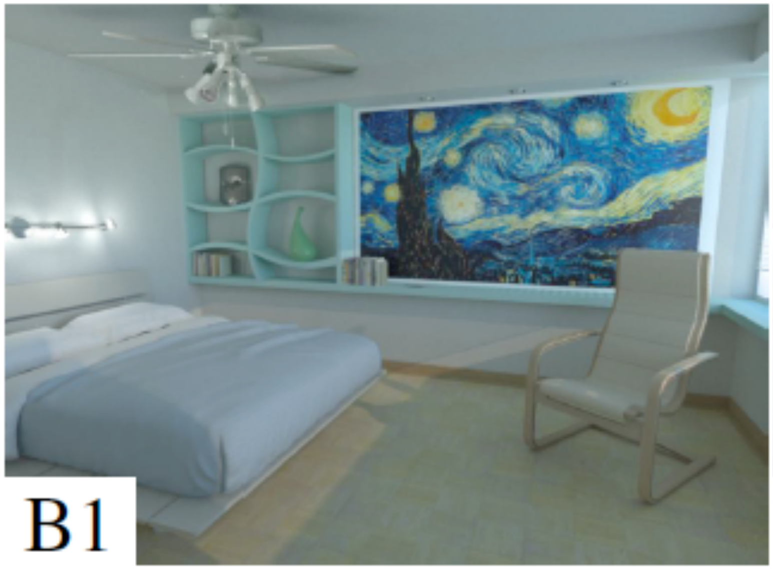
You can fool people

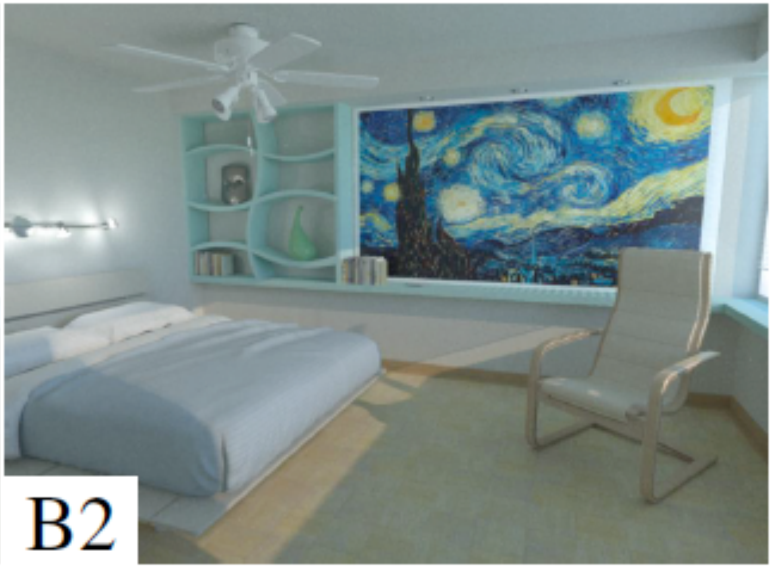
Real





Synthetic





Indoor lighting from LDR image

- Representation
  - Panorama
    - every surface is lit by gathering from one panorama
      - so panorama includes luminaires as well as reflectors
      - needs to be HDR - but image is LDR
    - equivalent to position independent lighting, as above

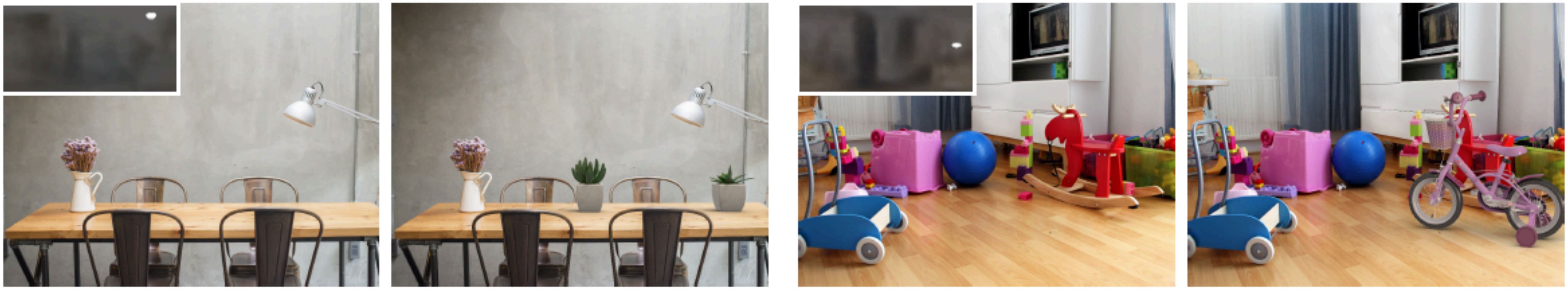


Fig. 1. Given a single LDR image of an indoor scene, our method automatically predicts HDR lighting (insets, tone-mapped for visualization). Our method learns a direct mapping from image appearance to scene lighting from large amounts of real image data; it does not require any additional scene information, and can even recover light sources that are not visible in the photograph, as shown in these examples. Using our lighting estimates, virtual objects can be realistically relit and composited into photographs.

Gardner et al 17

Gathering from a panorama needs care

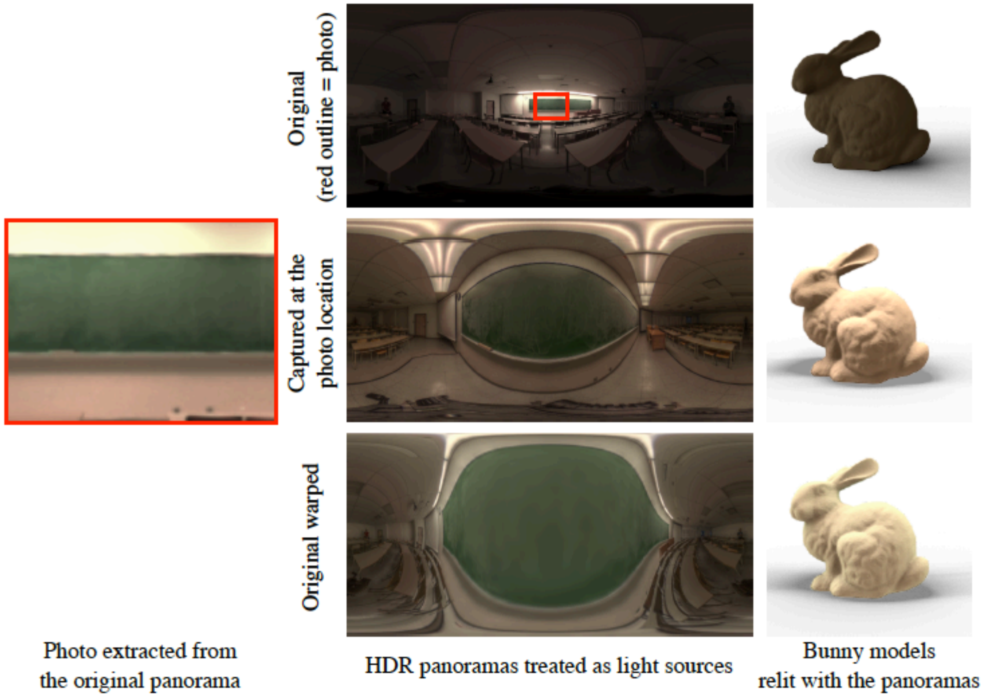


Fig. 6. The importance of light locality for indoor scenes. Left, a photo for which we want to estimate the lighting conditions. The photo was cropped from the "original" panorama (top row, middle). Treating this panorama as the light source for the photo is wrong; its center of projection is in front of the scene in the photo, and relighting a virtual bunny (top row, right) makes it appear to be backlit. The correct HDR panorama, captured with a light probe at the position of the cropped photo, is shown in the middle row, and captures the location of the lights on top of the scene. We introduce a warping operator that can be estimated with no scene information, and distorts the original panorama to approximate the location of the light sources on the top (bottom row). Relighting an object with the warped panorama yields results that are much closer to the ground truth.

Gardner et al 17

HDR panorama from LDR image

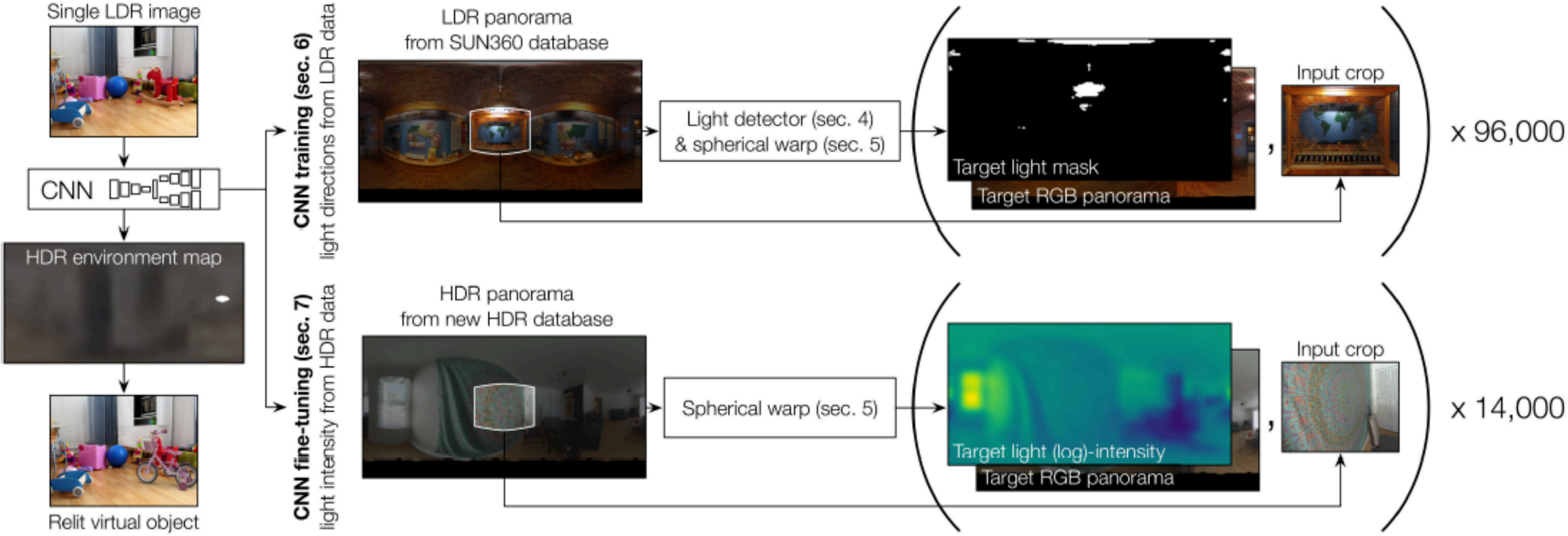


Fig. 3. Overview of the paper. Our method automatically predicts the HDR lighting conditions from a single photograph (left). To do so, it relies on a deep CNN that is trained in two stages. First, we rely on a database of LDR panoramas [Xiao et al. 2012]. To compensate for the low dynamic range, light sources are detected and the panoramas are warped to generate target light masks, which, combined with crops extracted from the panoramas, can be used to train the CNN to predict light directions. Second, the network is fine-tuned on a novel dataset of HDR panoramas, which allows it to learn to predict light intensities.

Gardner et al 17

Learned light detector is better…

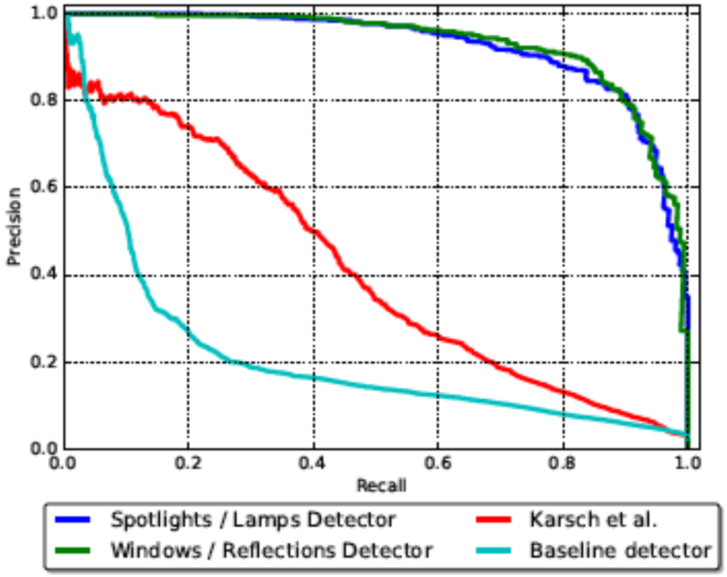


Fig. 4. Precision-recall curves for the light detector on the test set for our detectors and the one of Karsch et al. [2014]. In blue, the curve for the spotlights and lamps detection, and in green, the curve for the windows and light reflections. In red, the result for [Karsch et al. 2014]. In cyan, the curve for a baseline detector relying solely on the intensity of a pixel. Note that because of the inherent uncertainty of the importance of a light (including reflections) relative to the others (even for a human annotator), a perfect match between human and algorithm predictions is highly unlikely.

Gardner et al 17

This works….

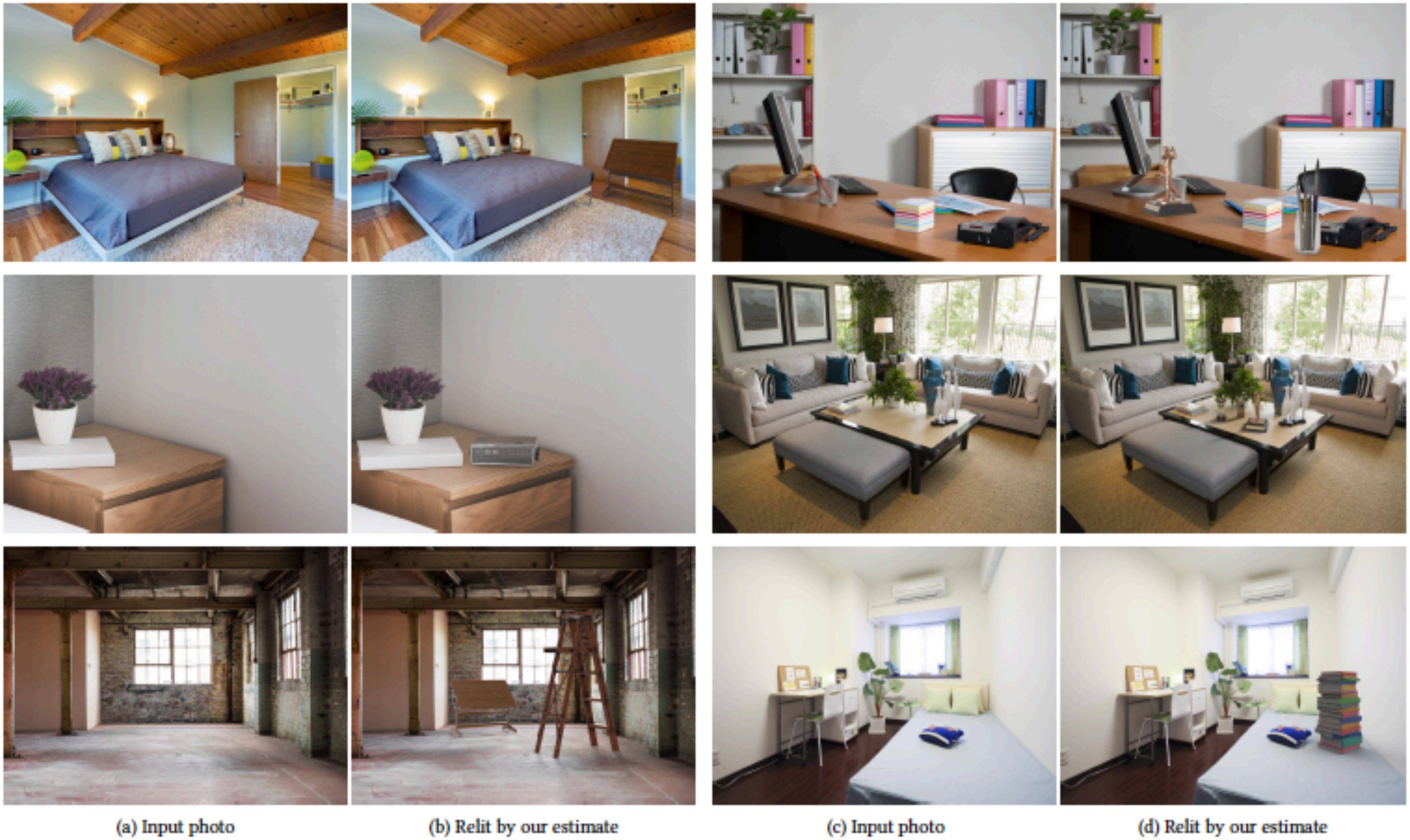


Fig. 16. Object relighting on a variety of generic stock photos downloaded from the Internet. In all cases, light estimation is performed completely automatically by our HDR network, the output of which is directly used by the rendering engine to relight the virtual objects.

Gardner et al 17

Evaluating realism is *REALLY HARD*

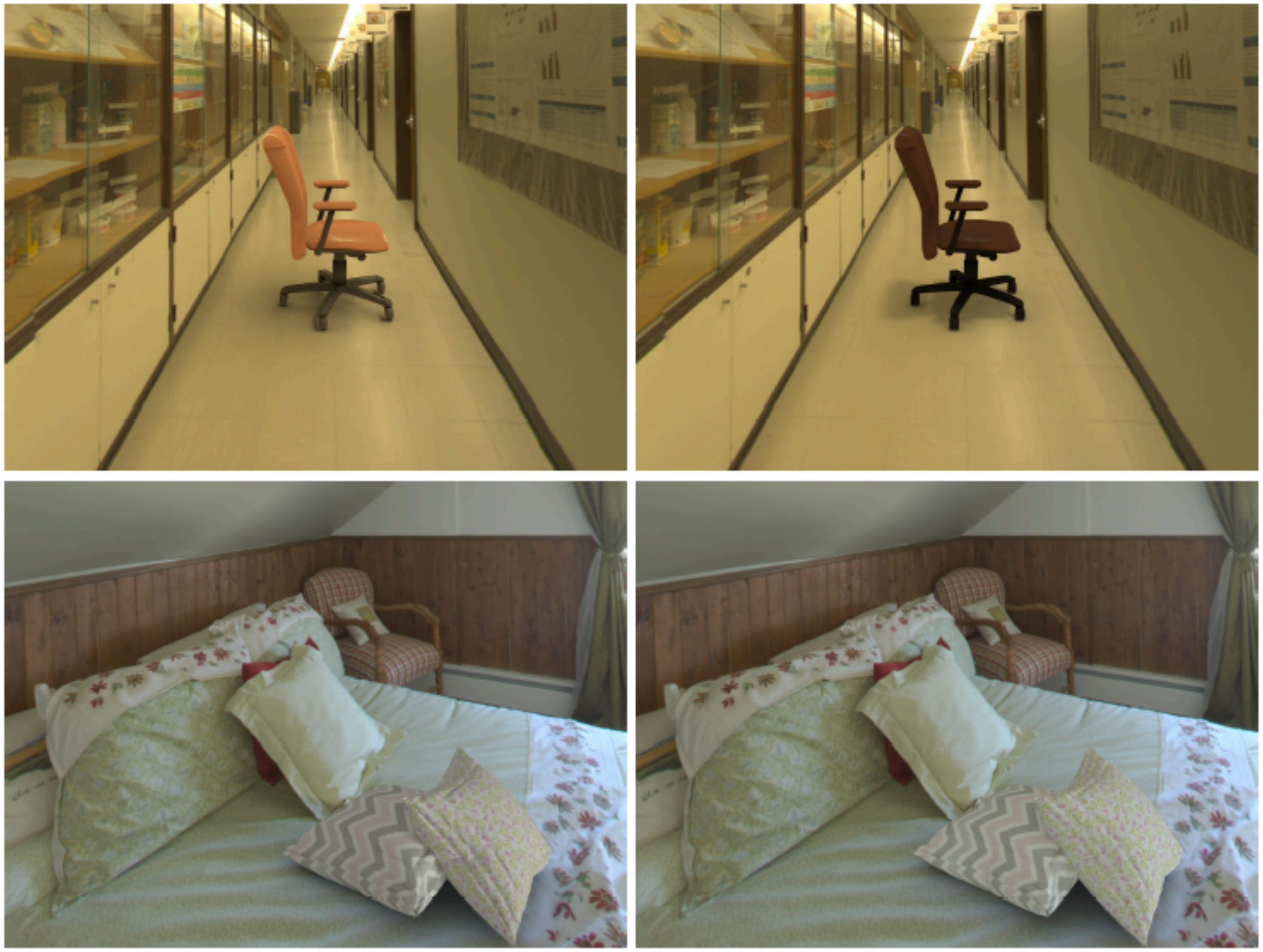


Fig. 18. For each row, the virtual objects in the image are either lit by the ground truth or by the output of our HDR network. Can you guess which is which? Answers below.

Gardner et al 17

Evaluating realism is *REALLY HARD*

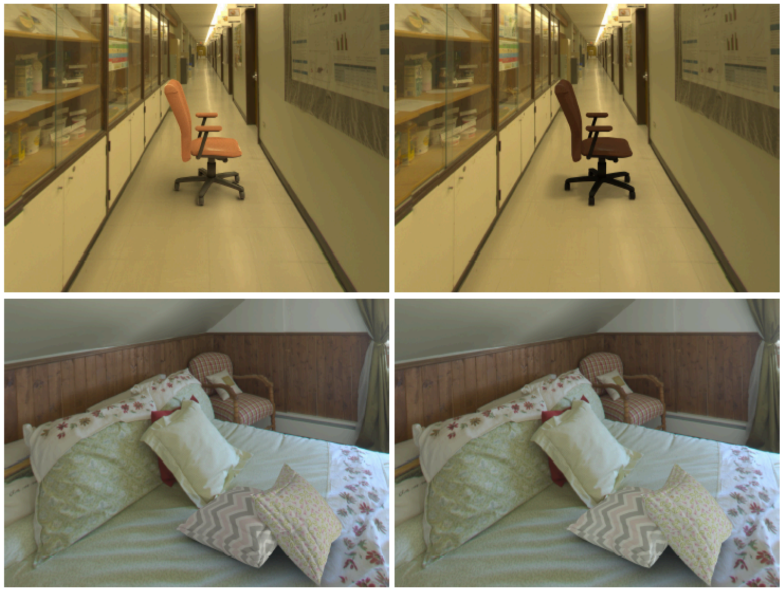
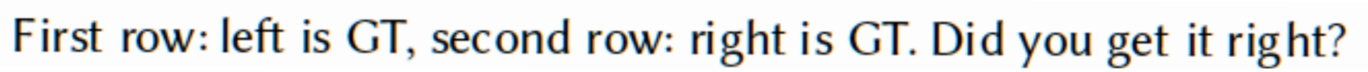


Fig. 18. For each row, the virtual objects in the image are either lit by the ground truth or by the output of our HDR network. Can you guess which is which? Answers below.



First row: left is GT, second row: right is GT. Did you get it right?

Gardner et al 17

Notes and queries

- Note:
  - there's a tension here w/ R+H
    - what is being inferred appears in considerable detail
    - note material parameters are NOT inferred
- Q: why does this work? and why does Karsch 14 work?
- Q: why use loss against ground truth panoramas?
  - rather than (say) illumination prediction on inserted objects?

Improvements

- Neural Illumination
  - Panorama depends on position
  - intermediate predictions
-

Neural Illumination

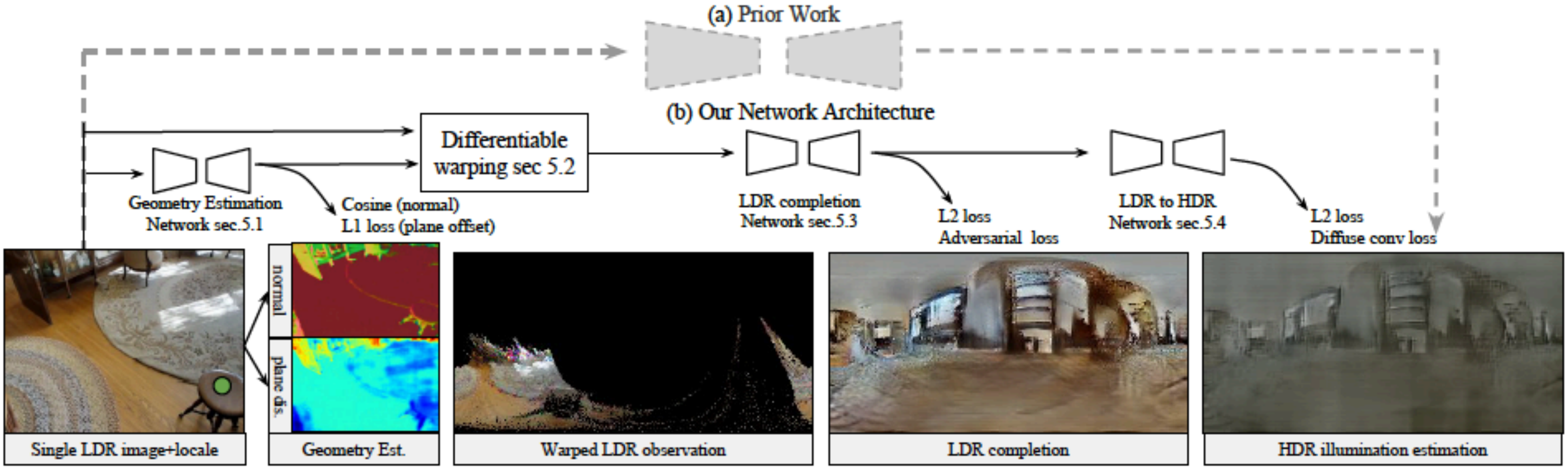


Figure 2. **Neural Illumination.** In contrast to prior work (a) [7] that directly trains a single network to learn the mapping from input images to output illumination maps, our network (b) decomposes the problem into three sub-modules: first the network takes a single LDR RGB image as input and estimate the 3D geometry of the observed scene. This geometry is used to warp pixels from the input image onto a spherical projection centered around an input locale. The warped image is then fed into LDR completion network to predict color information for the pixels in the unobserved regions. Finally, the completed image is passed through the LDR2HDR network to infer the HDR image. The entire network is differentiable and is trained with supervision end-to-end as well as for each intermediate sub-module.

Song+Funkhouser, 19

Neural Illumination

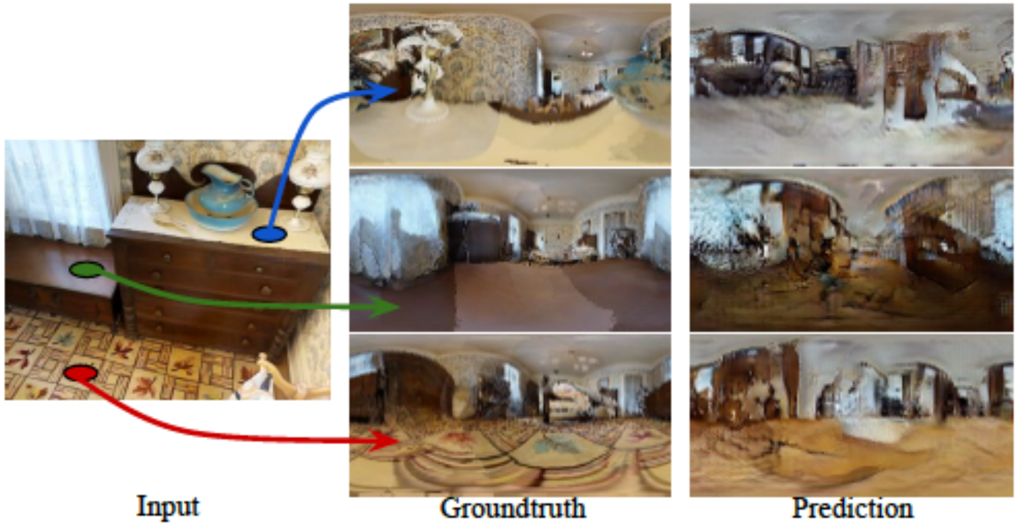
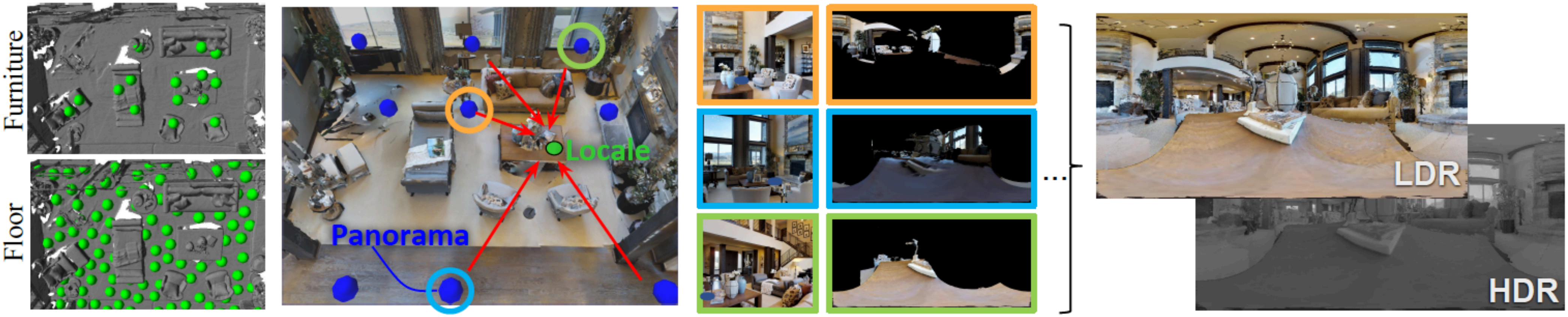


Figure 3. **Spatially varying illumination.** By using the 3D geometry, we can generate ground truth illumination for any target locale. As a result, our model is also able to infer spatially varying illumination conditioned on the target pixel location.

Song+Funkhouser, 19

Neural Illumination



(a) Target locales (b) Resample“nearby” panoramas (c) Observations (d) Combined pano as illumination groundtruth

Figure 4. **Ground truth illumination map generation.** We generate reconstructions of over 90 different building-scale indoor scenes using HDR RGB-D images from the Matterport3D dataset [2]. From these reconstructions, we sample target locales (a) on supporting surfaces (floors and flat horizontal surfaces on furniture). For each locale, we use HDR and 3D geometric information from nearby RGB-D images to generate ground truth panoramic illumination maps.

Song+Funkhouser, 19

Notes and queries

- All N+Q for Gardner apply
- Also, (ukase) this sort of thing has to stop
    - data collection exposes fundamentally absurd way of thinking about vision

Alternative spatially varying

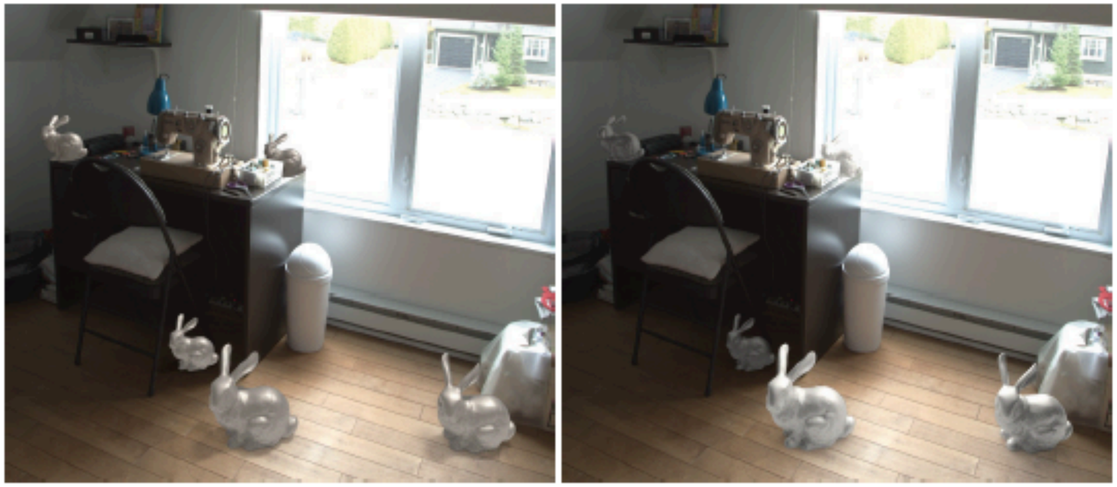


Figure 1. Indoor lighting is spatially-varying. Methods that estimate global lighting [8] (left) do not account for local lighting effects resulting in inconsistent renders when lighting virtual objects. In contrast, our method (right) produces spatially-varying lighting from a single RGB image, resulting in much more realistic results.

Garon et al 19

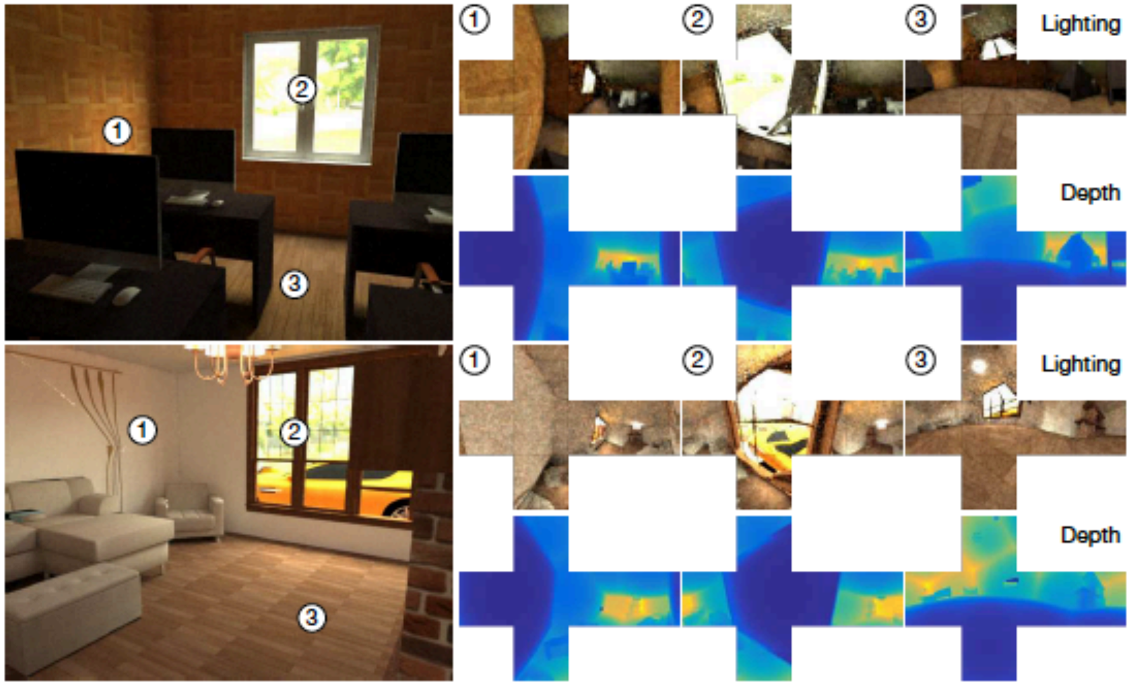


Figure 2. Example synthetic light probes sampled in our dataset. Locations on the image are randomly sampled (left). For each location, light probes (right, top) and their corresponding depth maps (right, bottom) are rendered into cube maps.

Garon et al 19

Adversarial smoother - important, and interesting

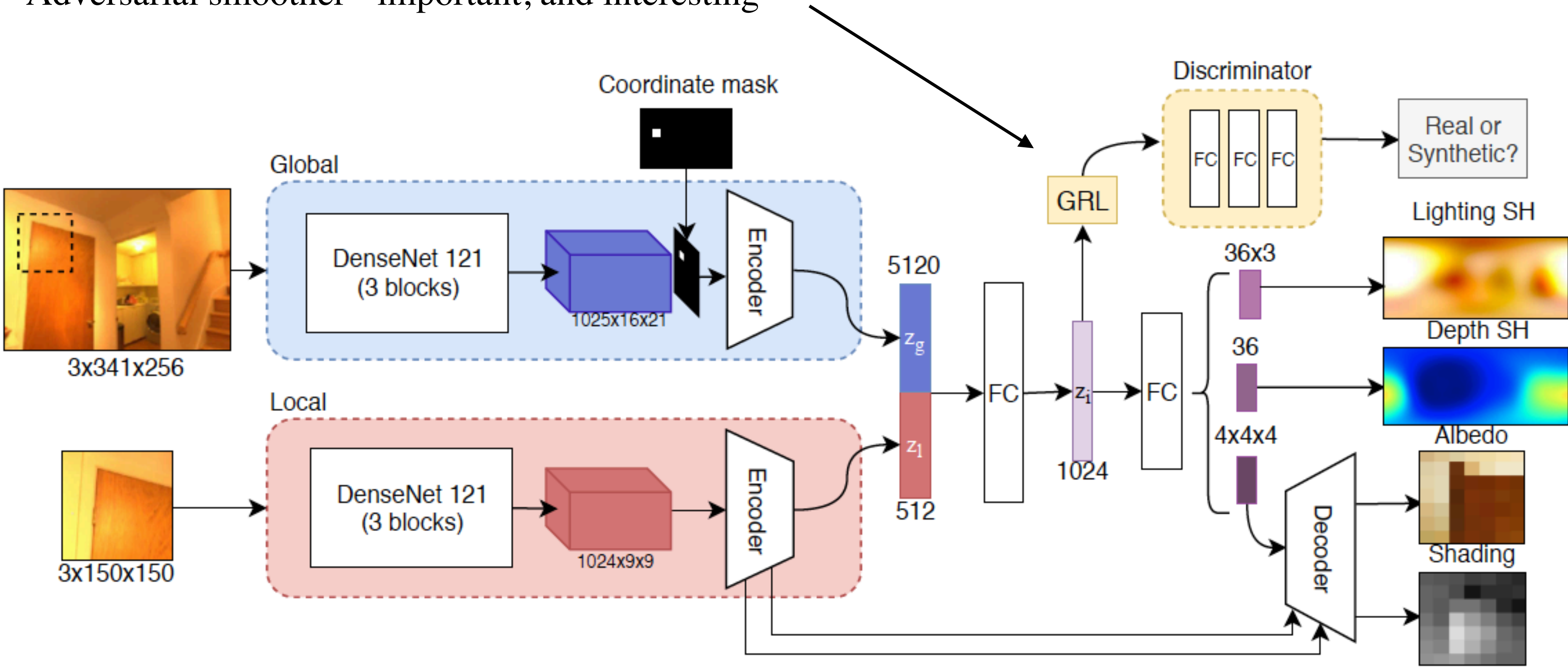


Figure 3. The architecture of our neural network. In blue, the *global path* where the full image is processed, and in red, the *local path* where a patch of the image centered at the coordinate where we want to estimate the light probe is provided. In both paths, three pre-trained blocks of DenseNet [15] and two Fire modules [16] trained from scratch are used to obtain the local and global features. The features are combined with two fully connected layers to output the RGB spherical harmonics coefficients of the lighting and spherical harmonics coefficients of the depth. A decoder is jointly trained to regress the albedo and the shading of the local patch. We apply domain adaptation [7] with a discriminator (yellow) and an adversarial loss on the latent vector to generalize to real images.

Garon et al 19

Albedo is accounted for

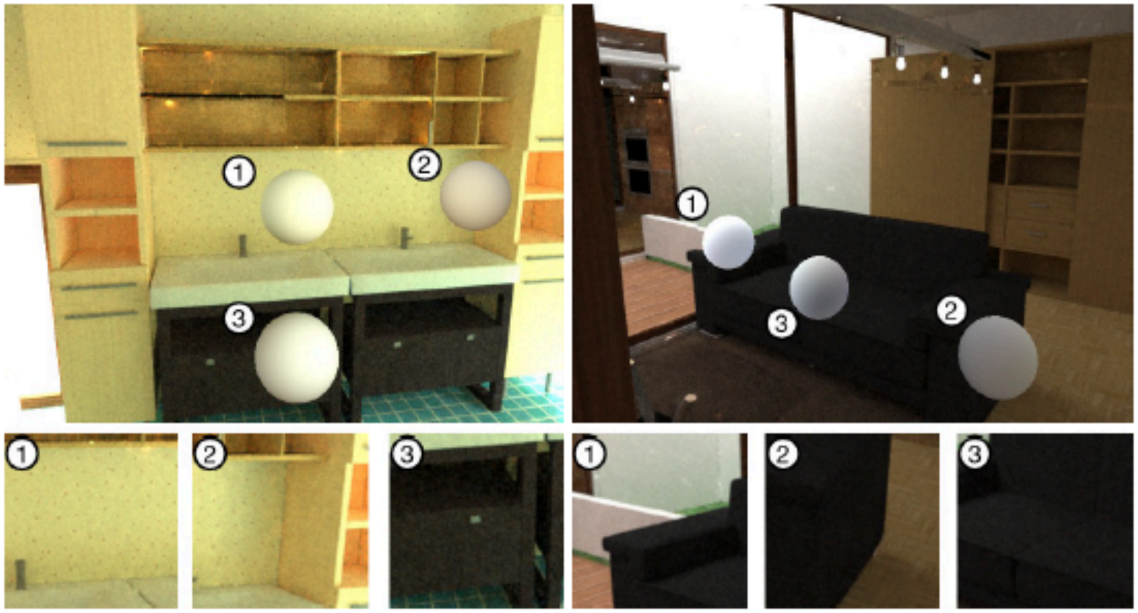
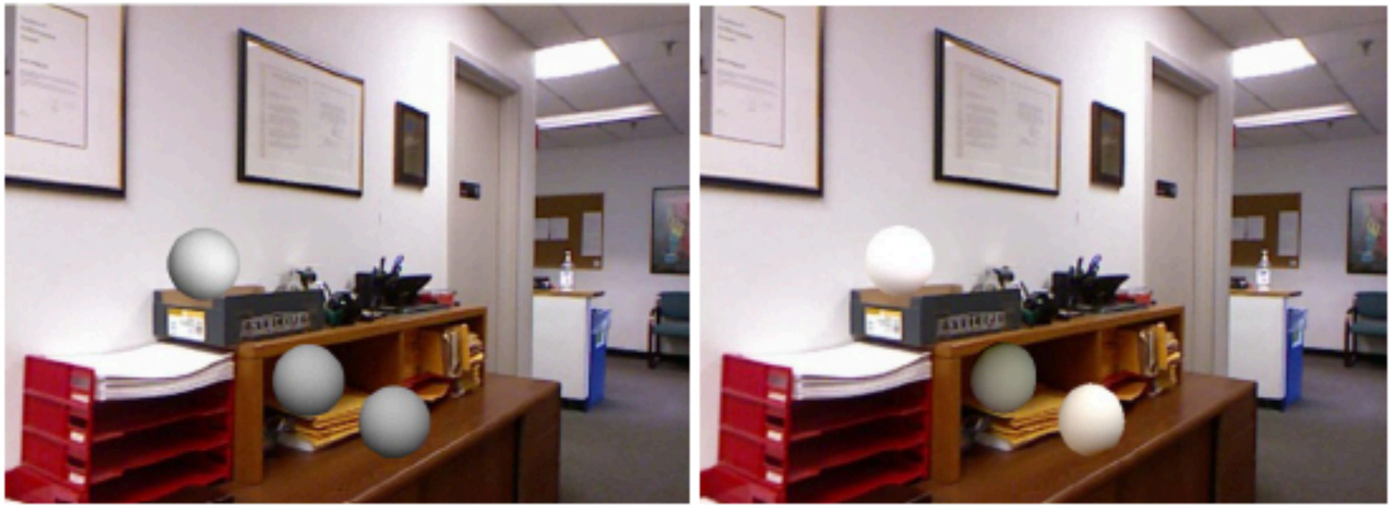


Figure 4. Qualitative examples of robustness to albedo changes. Our network adapts to the changes in albedo in the scene, and does not strictly rely on average patch brightness to estimate ambient lighting. We demonstrate three estimations: (1) a reference patch, (2) a patch that has similar brightness as the reference, but different lighting; and (3) a patch that has similar lighting as the reference, but different brightness. Notice how our network adapts to these changes and predicts coherent lighting estimates.

Garon et al 19

Better than spatially varying SH



(a) Barron and Malik [1]

(b) Ours

Figure 6. Qualitative comparison to Barron and Malik [1] on the NYU-v2 dataset [27]. While their approach yields spatially-varying SH lighting, it typically produces conservative estimates that do not capture the spatial variation in lighting accurately. In contrast, our method requires only RGB input, runs in real-time, and yields more realistic lighting estimates.

Garon et al 19

Notes and queries

- All NQ for Song above apply
- Adversarial smoother is interesting
- Q: why not directly score relights with adversary?

Taken to extremes…

Low res estimates

High res estimates

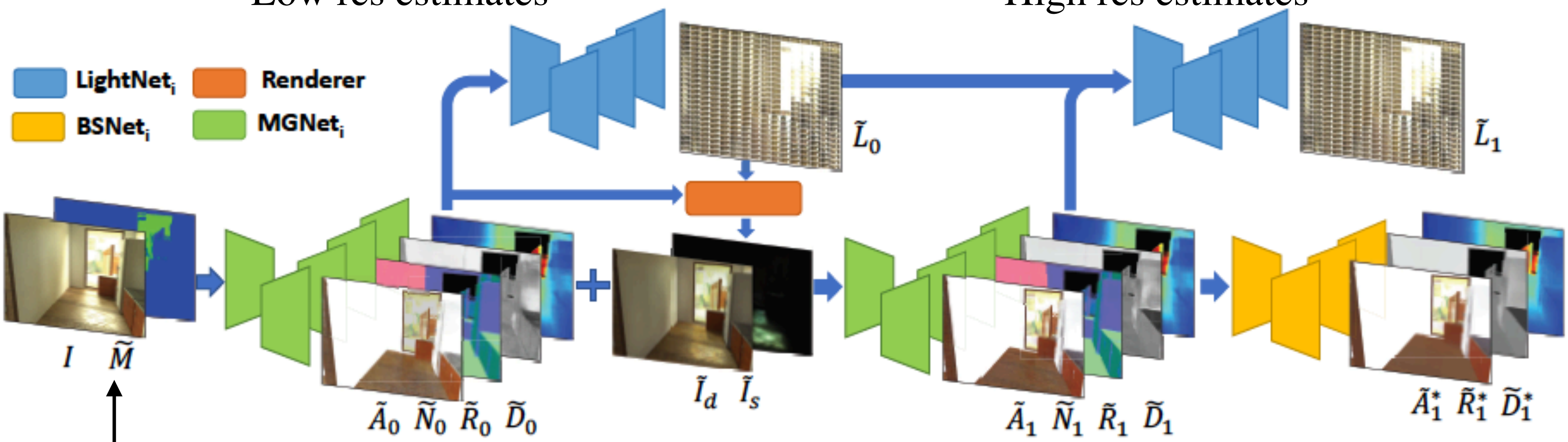

LightNet$_i$
Renderer
BSNet$_i$
MGNet$_i$
$\tilde{L}_0$
$\tilde{L}_1$
I $\tilde{M}$
$\tilde{A}_0$ $\tilde{N}_0$ $\tilde{R}_0$ $\tilde{D}_0$
$\tilde{I}_d$ $\tilde{I}_s$
$\tilde{A}_1$ $\tilde{N}_1$ $\tilde{R}_1$ $\tilde{D}_1$
$\tilde{A}_1^*$ $\tilde{R}_1^*$ $\tilde{D}_1^*$

Figure 7. Our network design consists of a cascade, with one encoder-decoder for material and geometry prediction and another one for spatially-varying lighting, along with a physically-based differentiable rendering layer and a bilateral solver for refinement.

Three estimated segmentation maps (object, area source, envmap)

Estimates of albedo normal, roughness, depth

Li et al, 20

Works very well

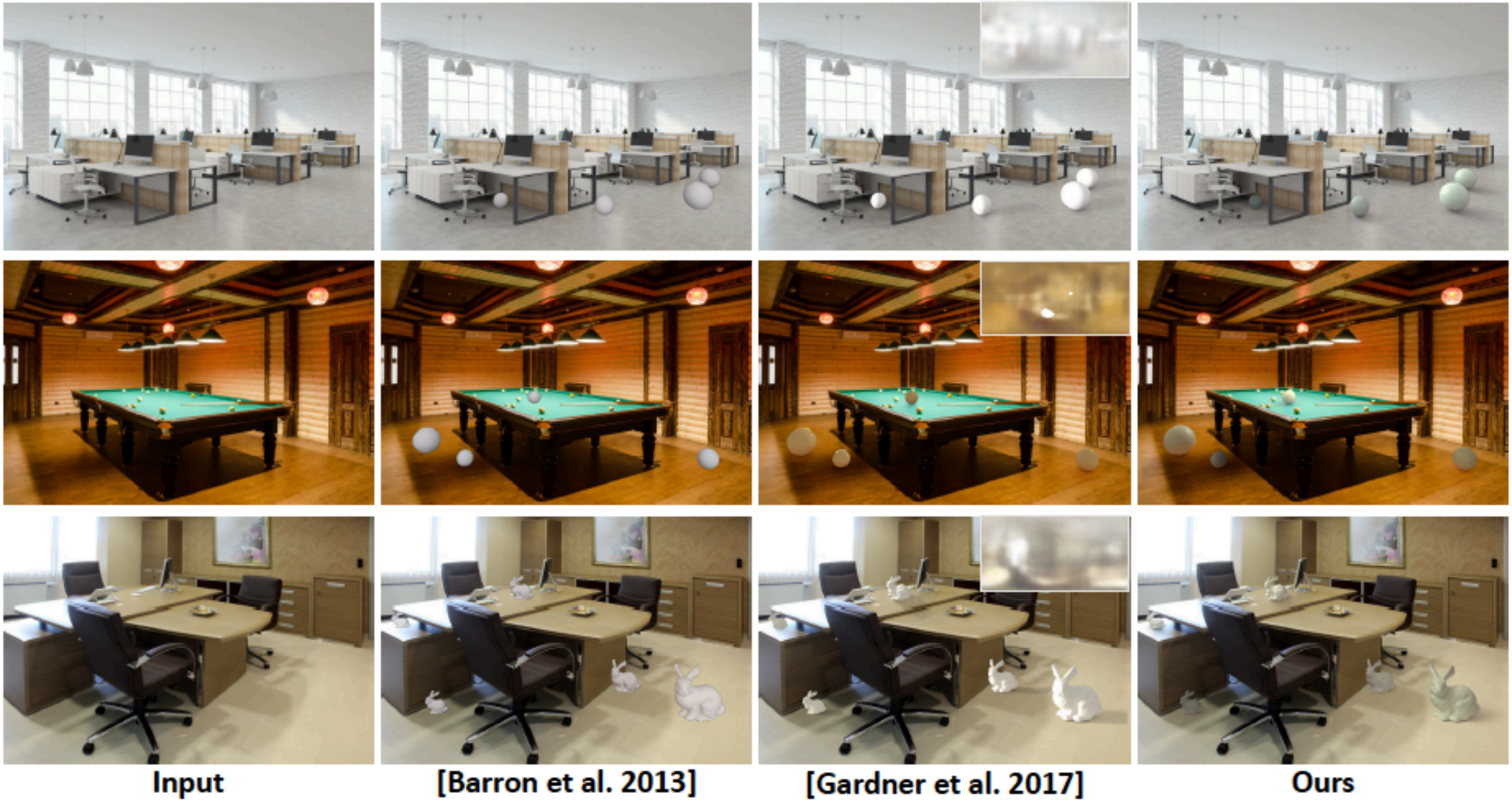


Figure 14. Object insertion examples and comparisons. Our proposed method estimates shape (depth and surface normals), spatially-varyin complex reflectance (based on a micro-facet SVBRDF model) and spatially-varying lighting from a single image of an indoor scene. Give these estimates, we can insert virtual 3D objects into these images and produce photo-realistic results where the objects look like they trul belong in the scene. Note the shading and specular highlights on the inserted spheres, the realistic shadows cast on the ground, the reflection from the ground onto the spheres and adaptation of the appearance of the spheres to the local shadows and shading in the scene. We als compare with previous works of Barron et al. [6] and Gardner et al. [20] on real images. Note that in the results of [20], the shadows of som objects might be truncated by the plane we segment from the scene.

Li et al, 20

Training

- Q: how did they train this?

Naughty…!

Li et al, 20

Notes and queries

- Sample from prior
- See also
    - Srinivasan et al 20
        - global spatial coherence of illumination, stereo input
    - Sengupta et al 19
        - alternative inverse renderer

Cut-and-paste

- Very little is known
- Hard:
  - we don’t have surface/material models of the thing being inserted
  - Q:
    - How should it interact with inferred light field?
    - Do we need an inferred light field?
- Promising:
  - cut-and-paste is a super rendering paradigm
    - easy to control
    - easy for use
    - IF we could do it

Inserting Image fragments, V1

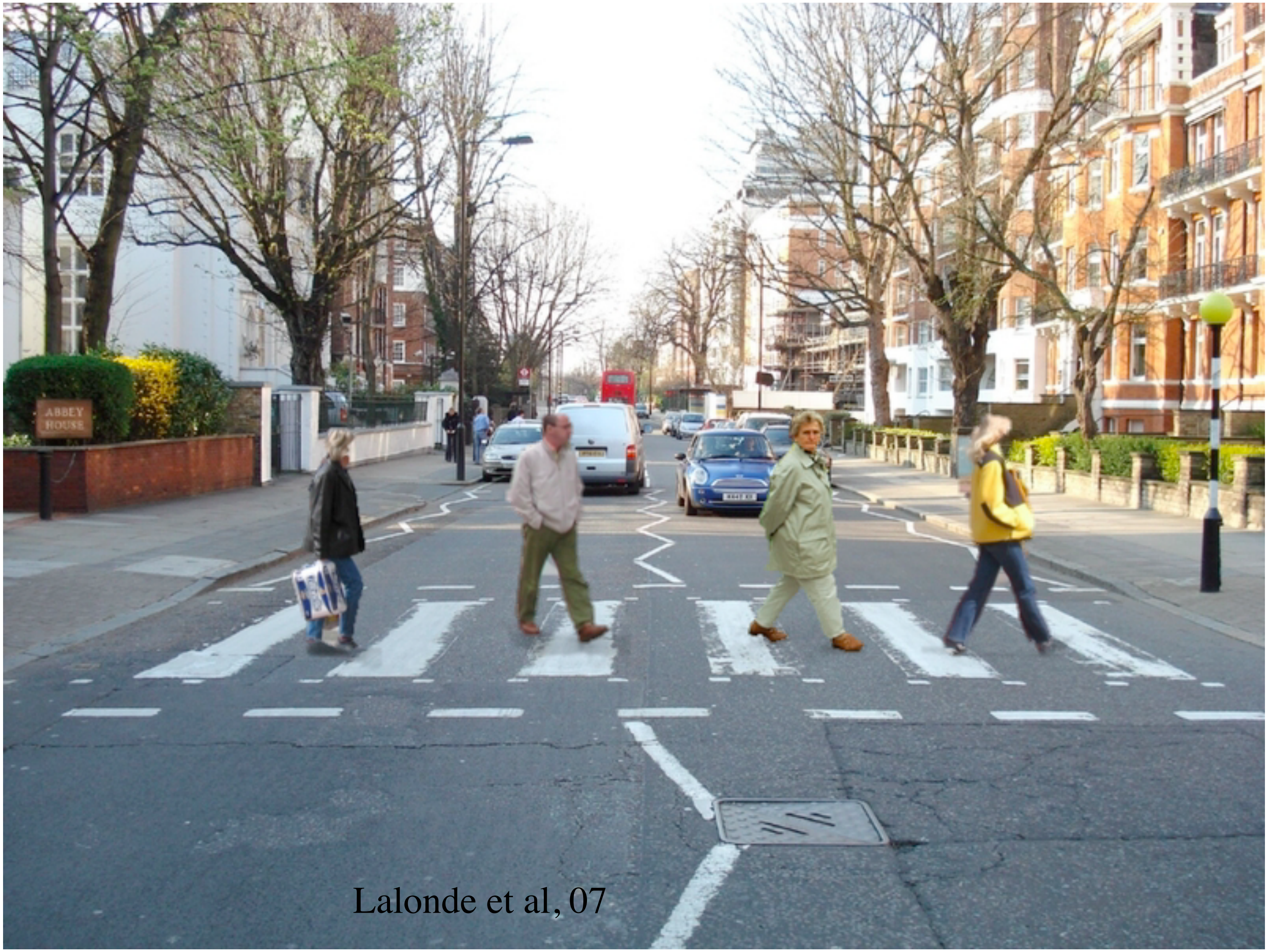
- Algorithm
  - build a dictionary of image fragments, ideally tagged
    - for these fragments, estimate height using ground plane
  - artist chooses image
    - system estimates horizon, ground plane
    - this gives foreshortening
  - artist searches with tag, chooses fragment
  - places on image

Lalonde et al, 07

Inserting fragments

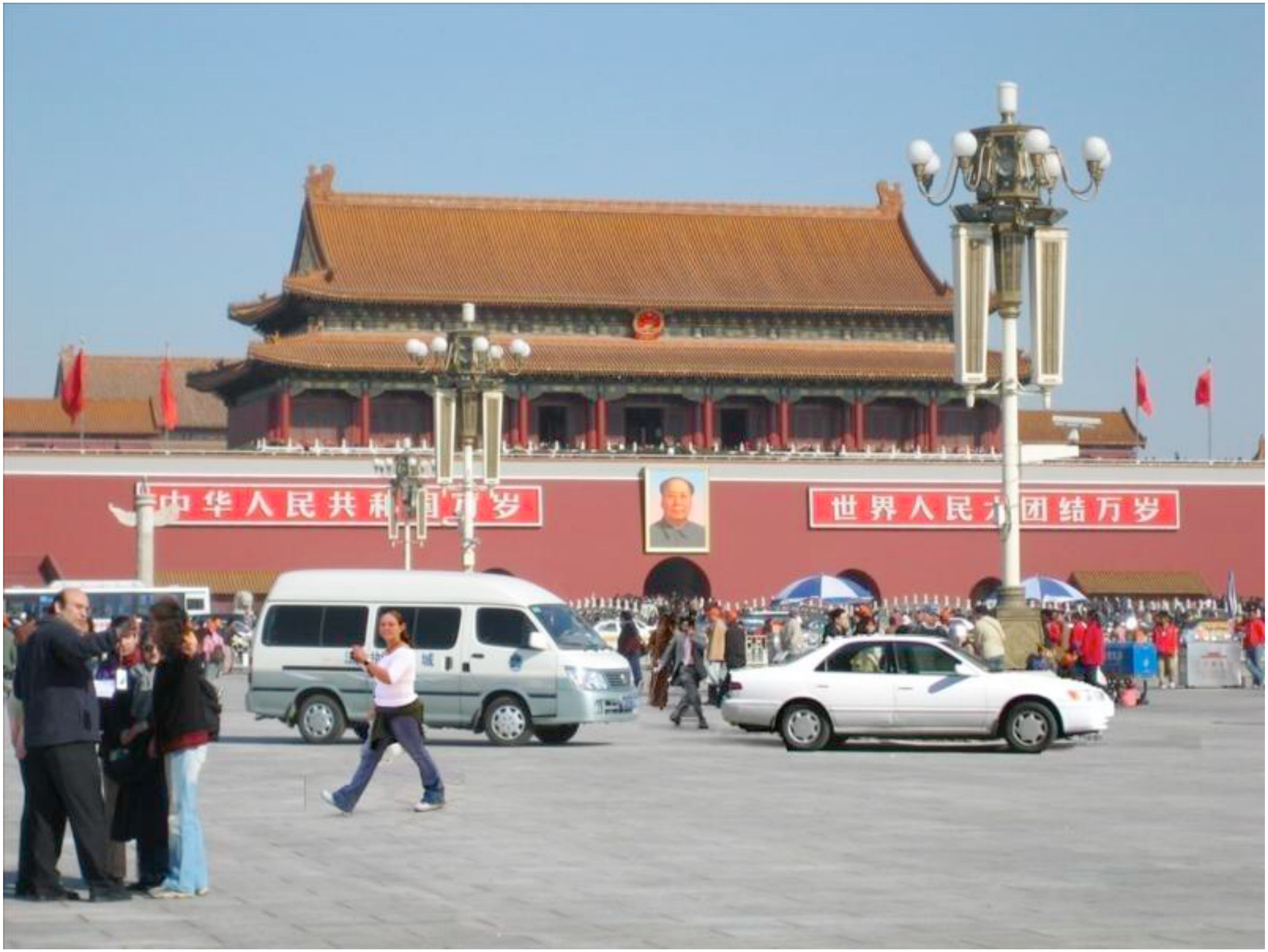
Lalonde et al, 07

Inserting fragments

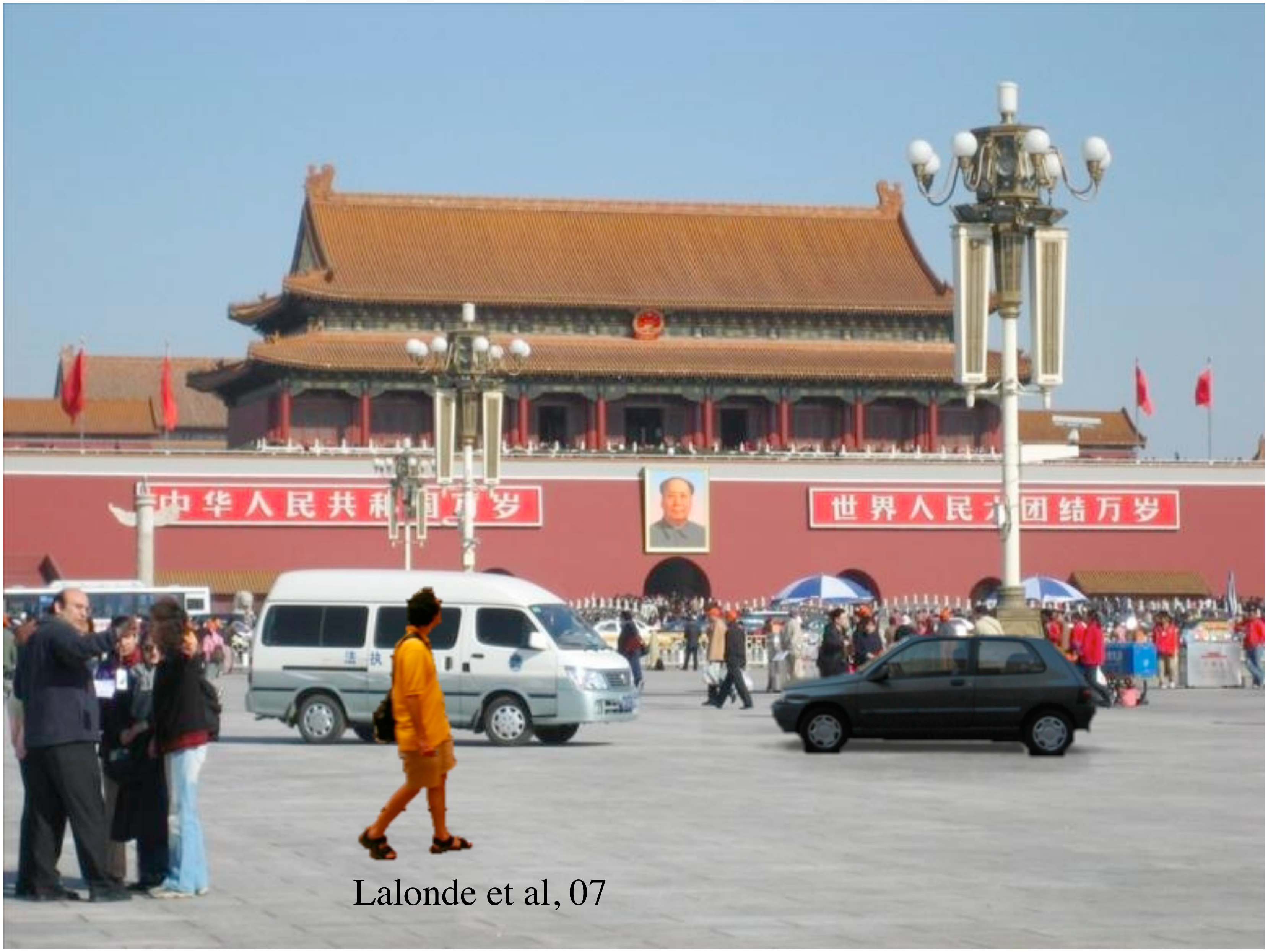


Lalonde et al, 07

Illumination issues: good match



Illumination issues: bad match



Lalonde et al, 07

Inserting Image fragments, V1

- Algorithm
  - build a dictionary of image fragments, ideally tagged
    - for these fragments, estimate height using ground plane
  - artist chooses image
    - system estimates horizon, ground plane
    - this gives foreshortening
  - artist searches with tag, chooses fragment
  - places on image

- Q: what if the light is wrong?
  - A: don’t use that fragment

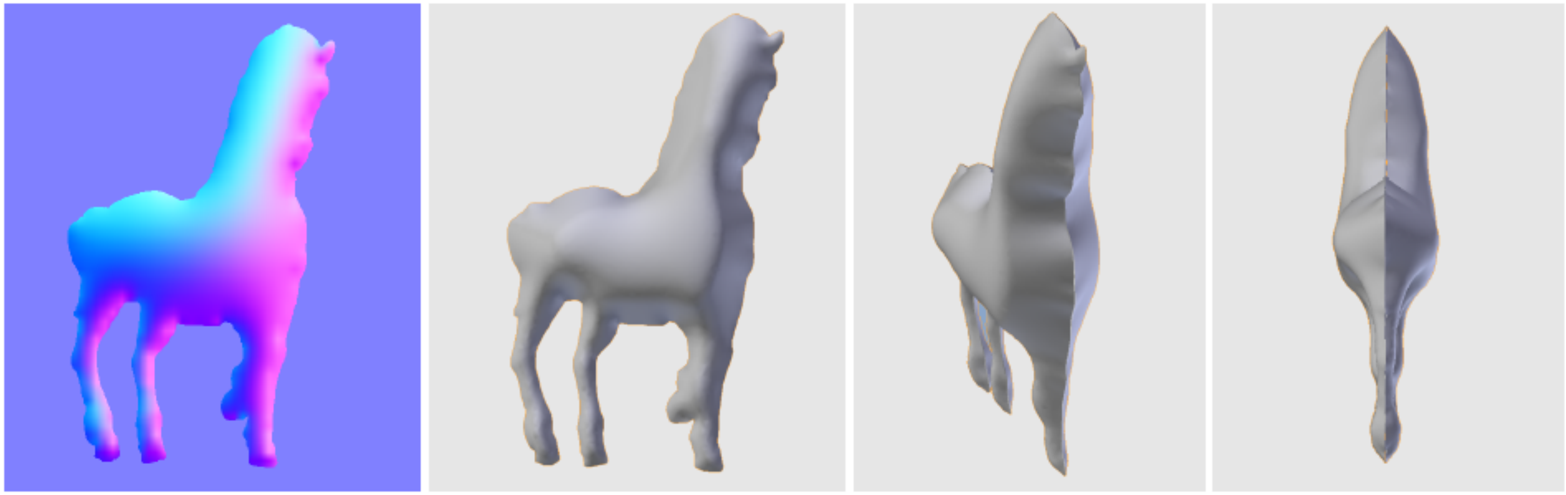
Lalonde et al, 07

Inserting Image fragments, V2

- Problem: need the shape to get shadows, shading
  - current technology cannot do this for complex surfaces
- Resolution:
  - People are very bad at shadow consistency
  - Weak shape approximations are good enough IF
    - you carefully preserve surface material properties
  - Decompose object shading into
    - Smooth component (from weak shape approximation)
    - Faster components
      - from materials

Weak shape approximations

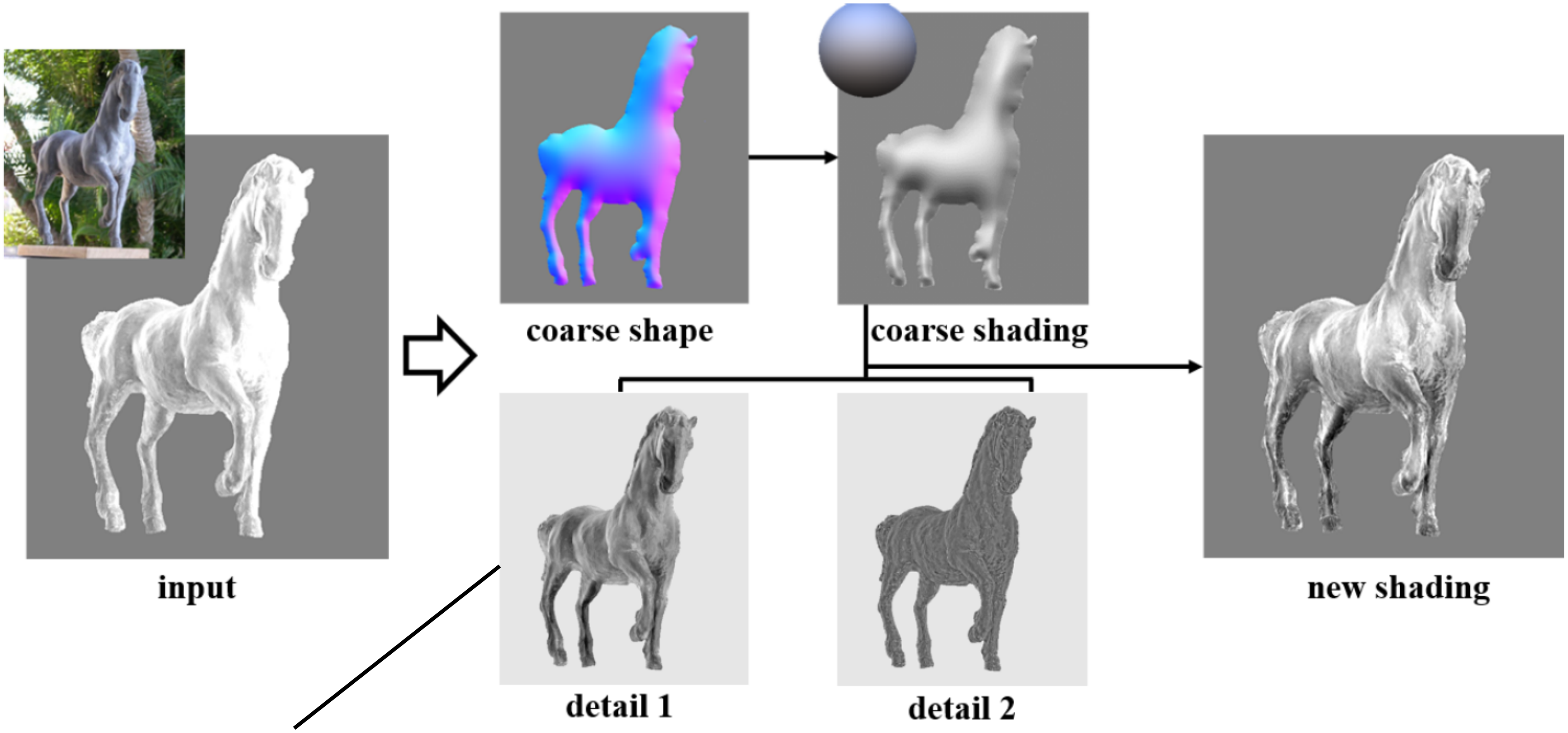
- Simple shape from contour
  - with small modifications to ensure crease at contour, planar contour
  - flip+glue to provide a back



(a) normal (b) 3D shape (c) view2 (d) view3

Liao et al 2015, 2019

Fast components are residuals

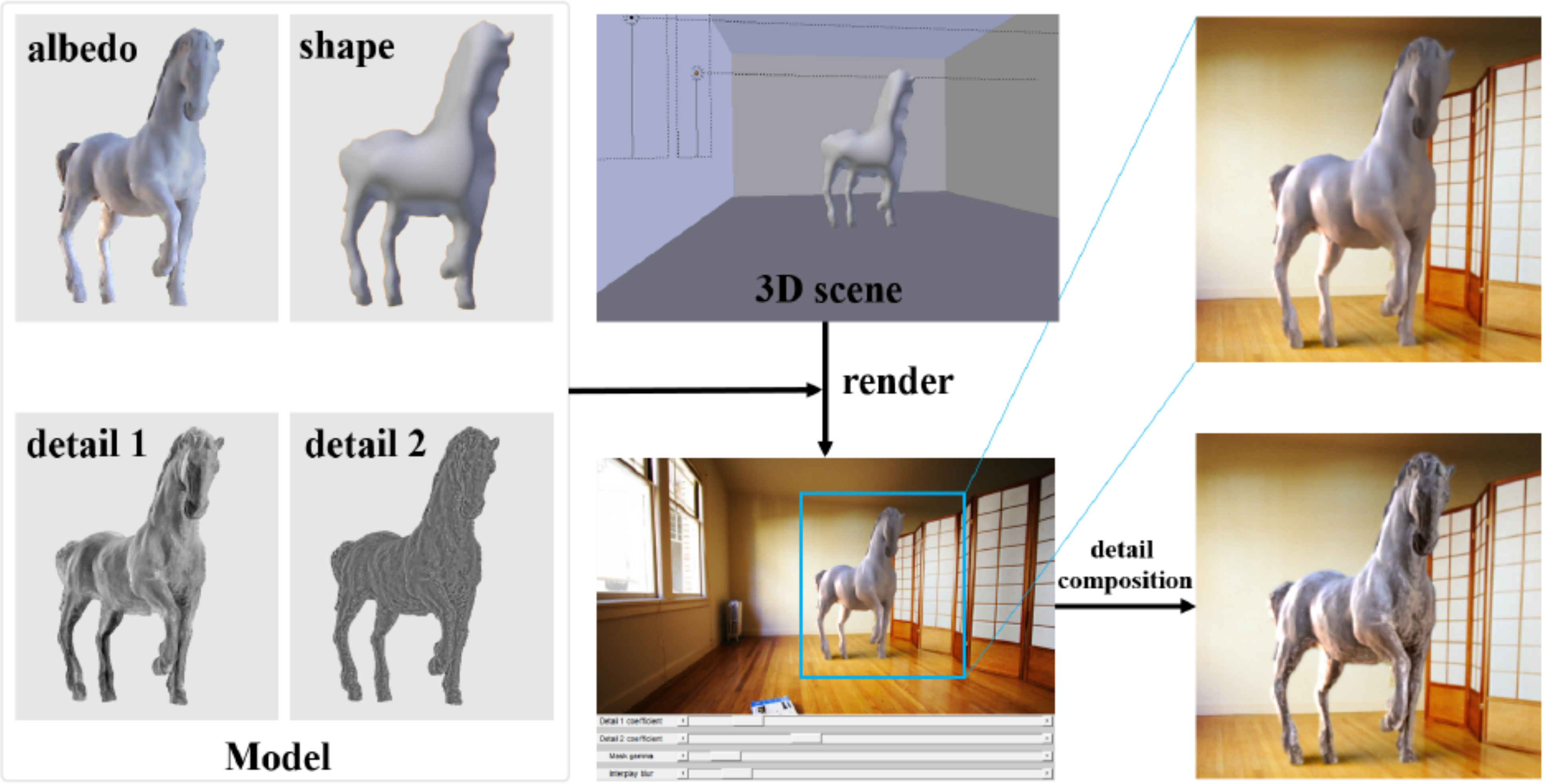


Detail 1= Image-Shade(coarse shape, estimated light)

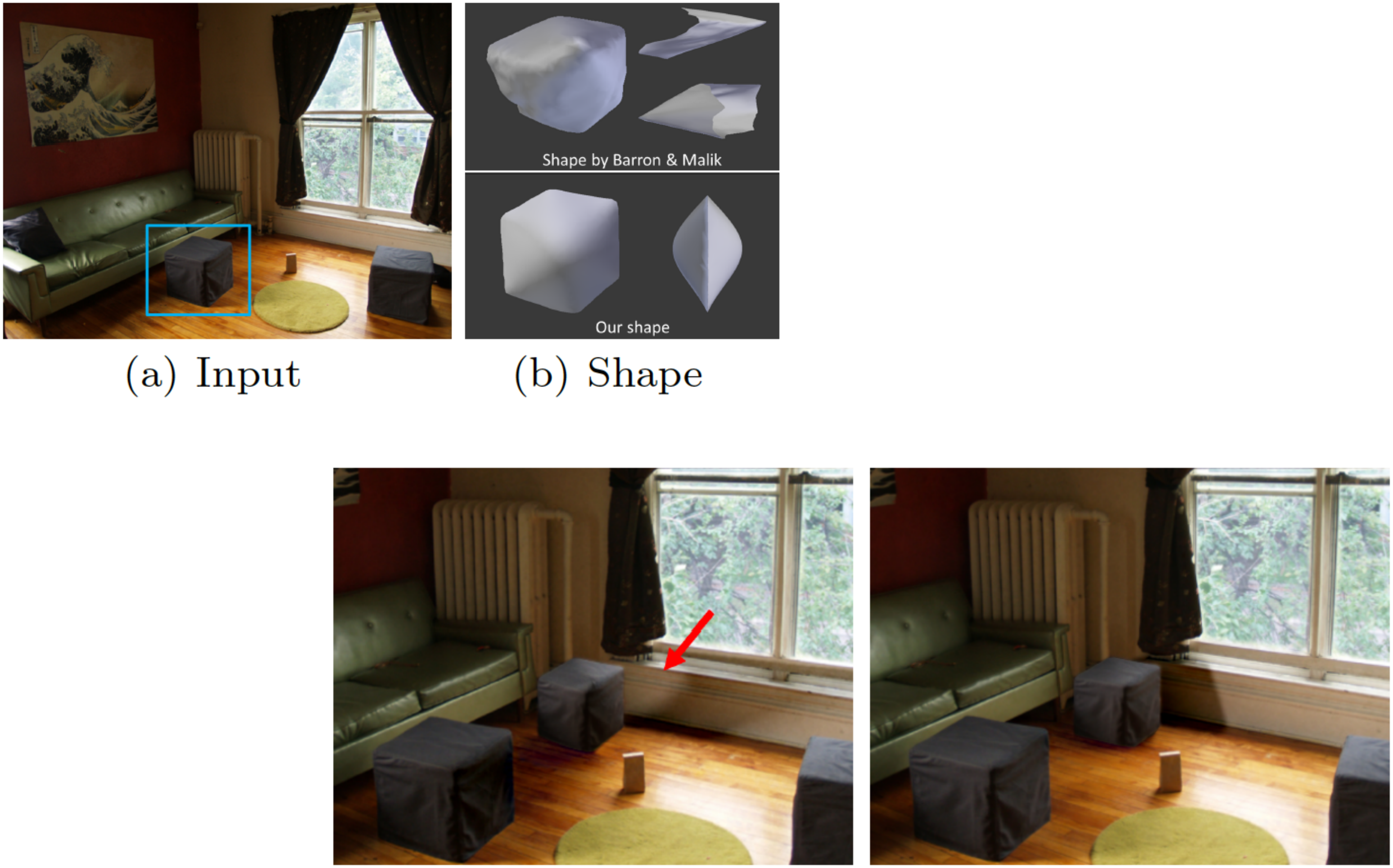
Detail 2= Shading detail(Image-Shade(coarse shape, estimated light))

Liao et al 2015, 2019

Placing and rendering objects



Liao et al 2015, 2019



(a) Input

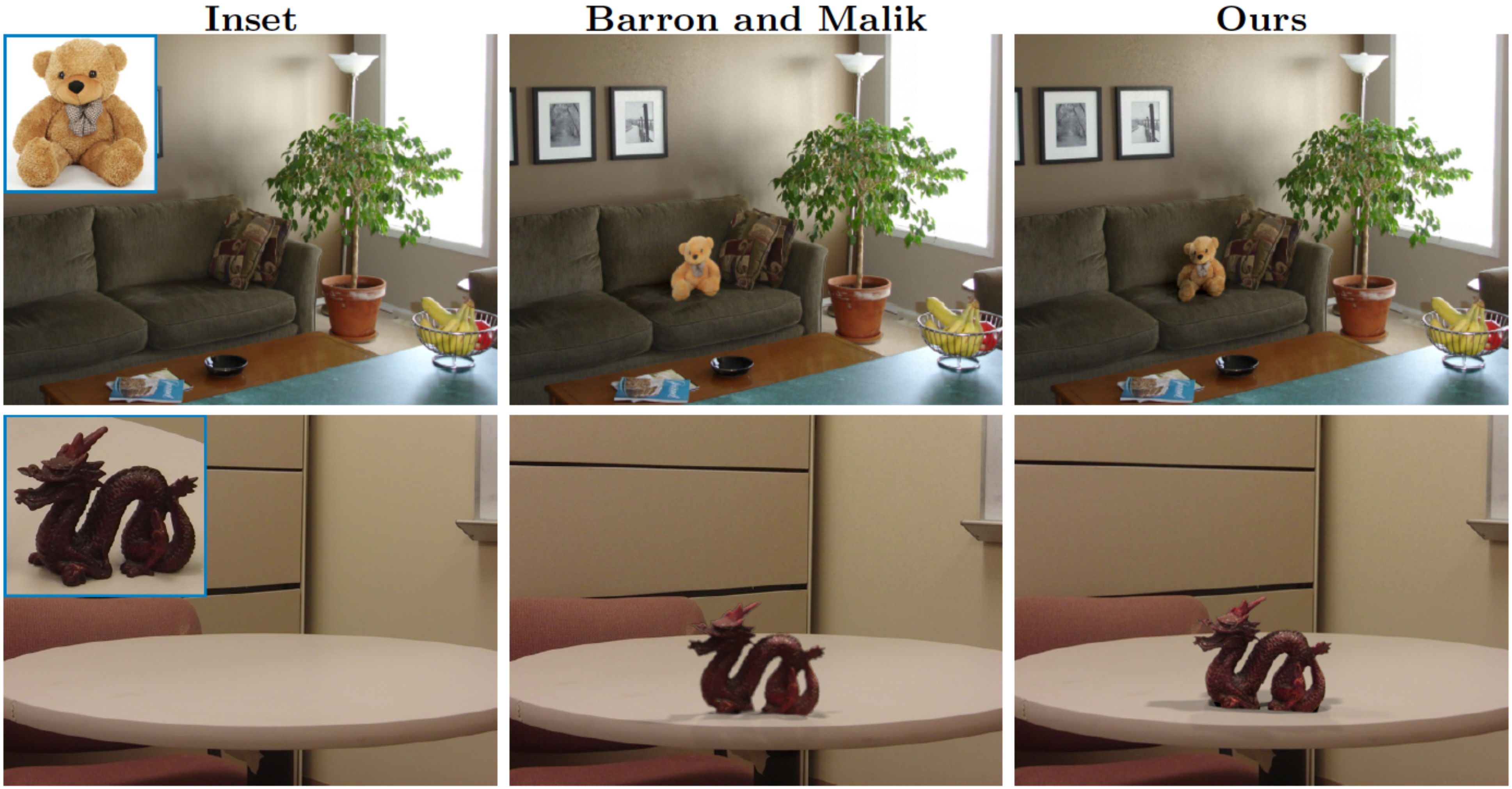
(b) Shape

(c) Result of B & M

(d) Our result

Liao et al 2015, 2019

Visual comparisons



Inset

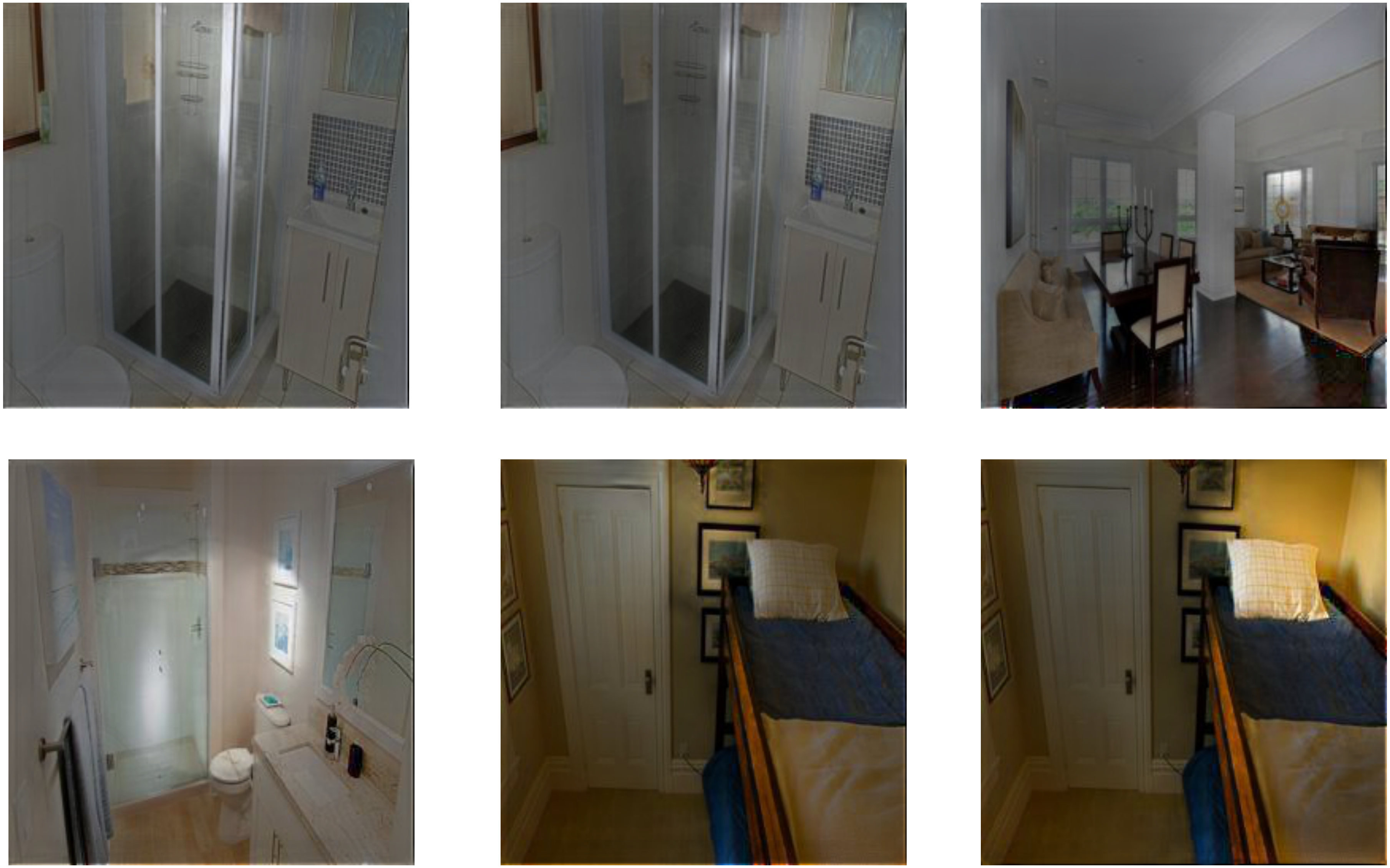
Barron and Malik

Ours

Liao et al 2015, 2019

Image based reshading for scenes

- Issues:
  - how to decompose images
  - how to generate multiple diverse shading fields from one image of a scene



Intrinsic images

- (Originally) Maps of an image that explain pixel values
    - Intrinsic properties:
        - independent of viewing; “object” or “world” properties
    - Extrinsic properties:
        - depend on viewing circumstances

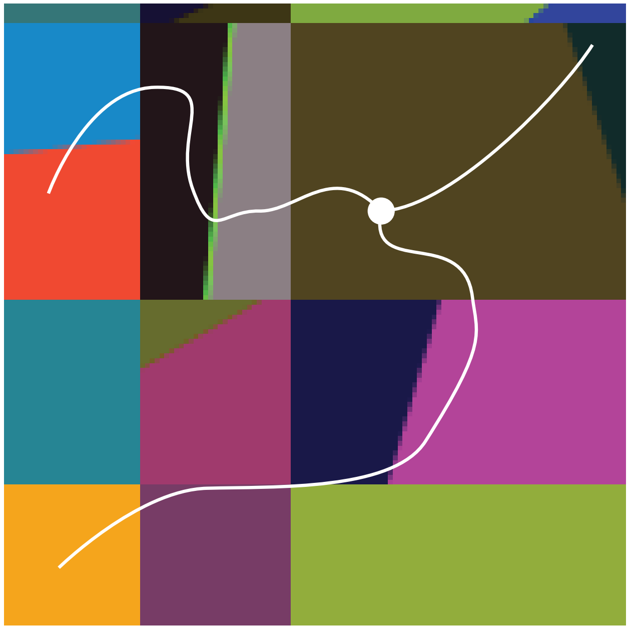
- (Later) Albedo/Shading maps
    - I=A x S
    - Albedo (A) is a natural intrinsic
    - Shading (S) is a natural extrinsic

No ground truth decompositions

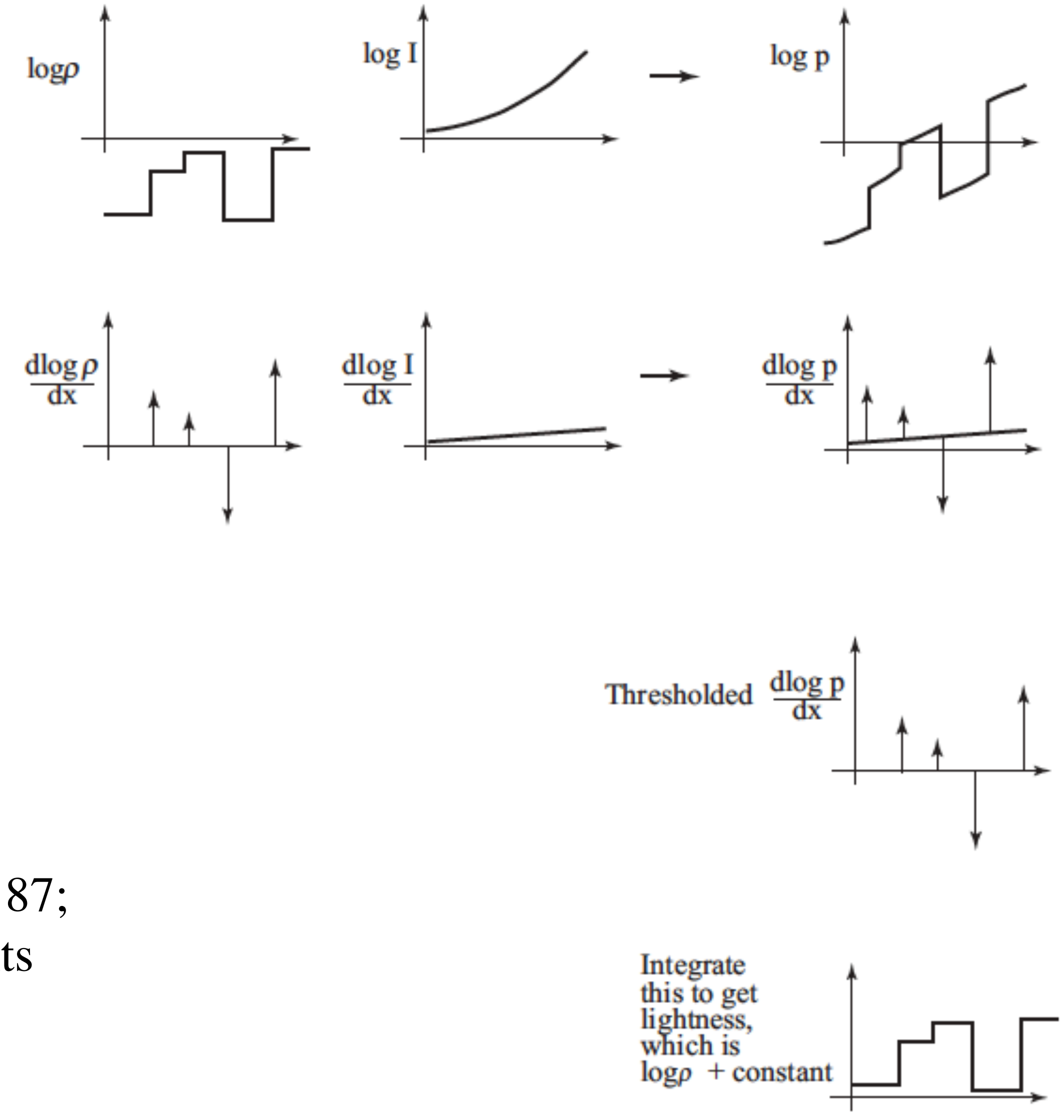
- And there never will be
  - rendering is do-able (but hard)
  - modelling is hopeless
- Q: how do you train an image decomposition method when you don’t know the right answer?
- Retinex provides clues - spatial statistics are the key

Albedo/shading and Retinex

- Spatial reasoning, Land (59, 59, 77); Land +McCann 71:
  - Surface color changes either quickly or not at all
  - Light color changes slowly
  - Retinex
    - large family of algorithms
    - quite hard to know what Retinex does (Brainard+Wandell, 86)



Computer vision versions of Retinex



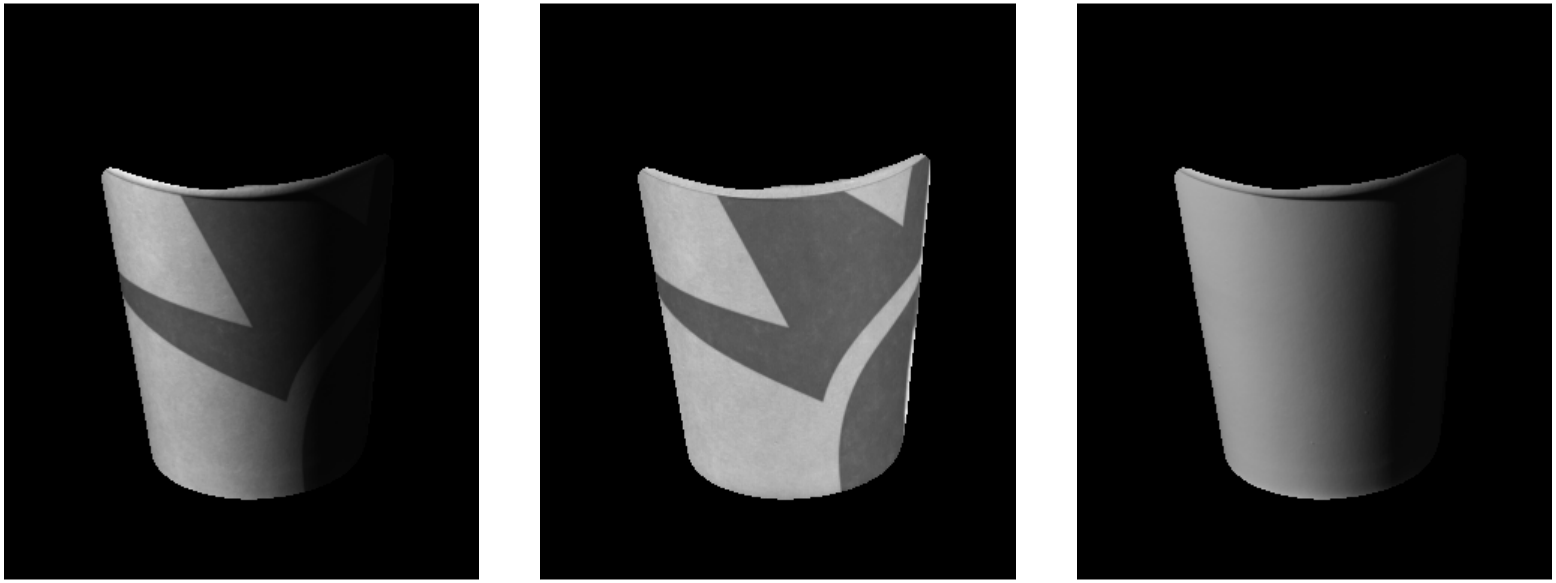
Integrate this to get lightness, which is logρ + constant

Horn, 73; 74
Brelstaff+Blake, 87;
multiple variants

Real data is hard to collect

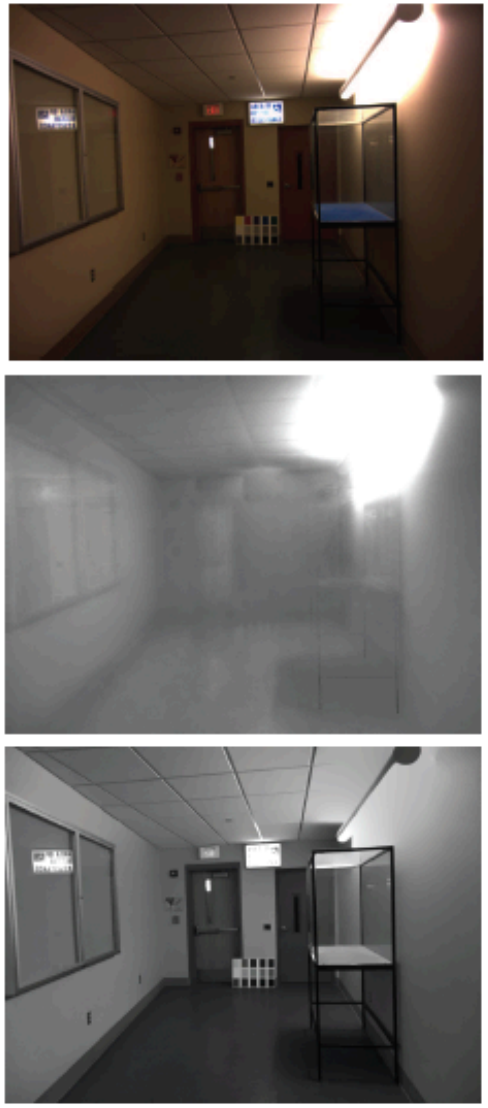
- spraypaint, multiple images, etc…

Images from dataset of Gosse et al. 09

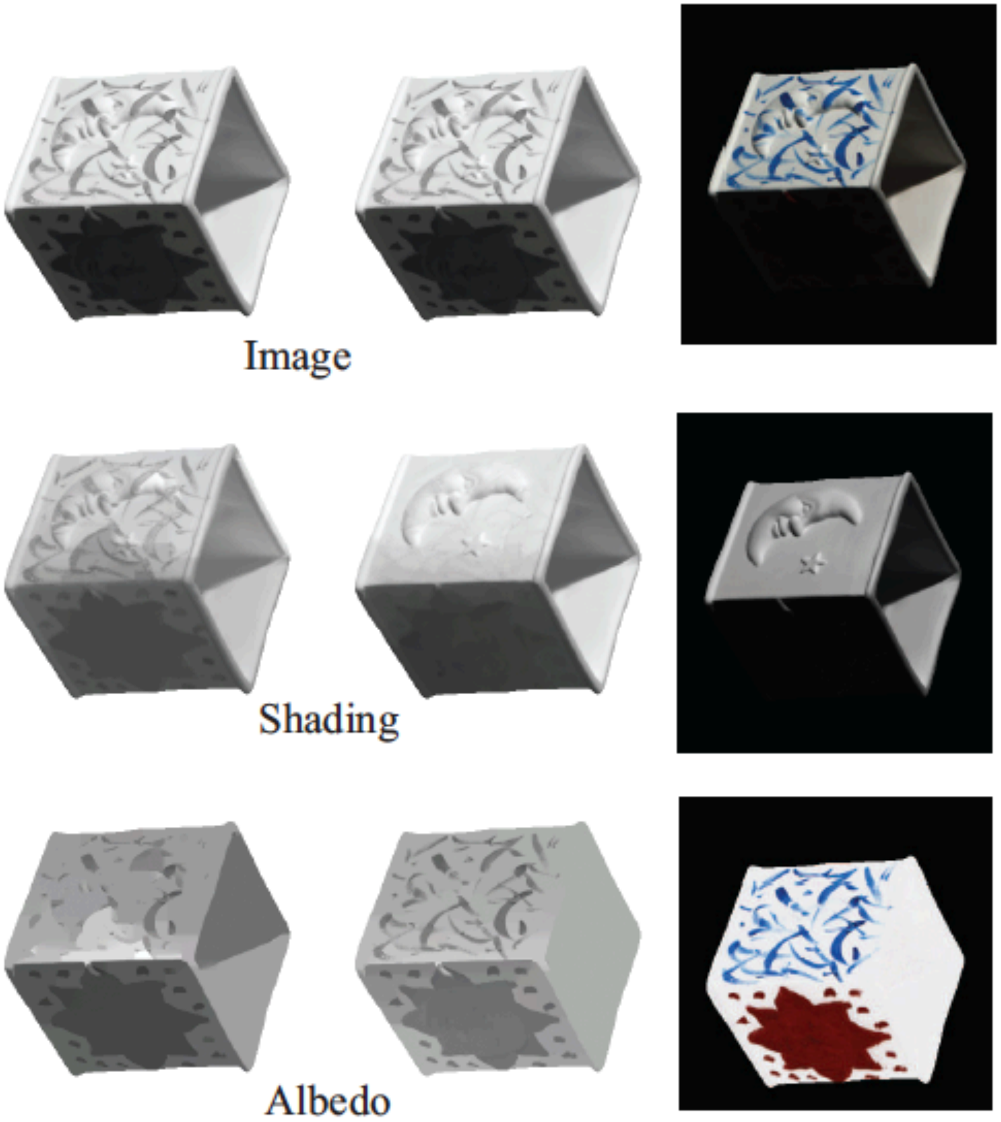


Retinex is really quite good

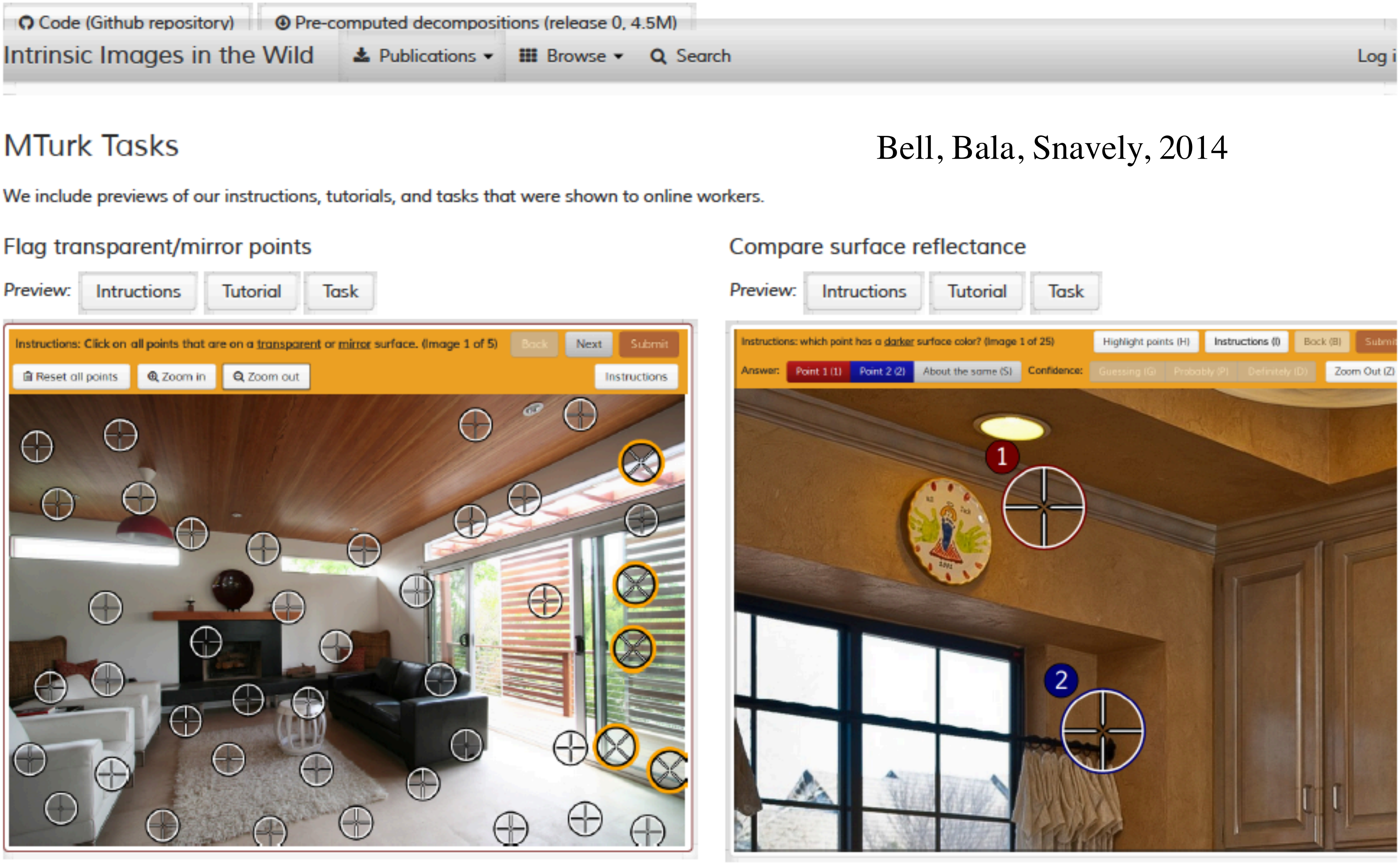
Implementation of Retinex
due to Kevin Karsch



Ground truth
images from dataset of Gosse et al. 09



Human judgements are easier



Code (Github repository)

Pre-computed decompositions (release 0, 4.5M)

Intrinsic Images in the Wild

Publications ▾ Browse ▾ Search Log i

MTurk Tasks

Bell, Bala, Snavely, 2014

We include previews of our instructions, tutorials, and tasks that were shown to online workers.

Flag transparent/mirror points

Preview: Intructions Tutorial Task

Instructions: Click on all points that are on a transparent or mirror surface. (Image 1 of 5)

Compare surface reflectance

Preview: Intructions Tutorial Task

Instructions: which point has a darker surface color? (Image 1 of 25)

Answer: Point 1 (1) Point 2 (2) About the same (S)

Confidence: Guessing (G) Probably (P) Definitely (D)

This gives an evaluation task

- WHDR=Weighted Human Disagreement Ratio
  - compute lightness from intrinsic image representation at points
  - predict
    - A lighter than B
    - B lighter than A
    - Lightness match
  - compute weighted estimate of accuracy
    - weights low where human judgements are uncertain, high otherwise
- There are issues, but allows evaluation
  - and competition

Modern strategies - Optimization

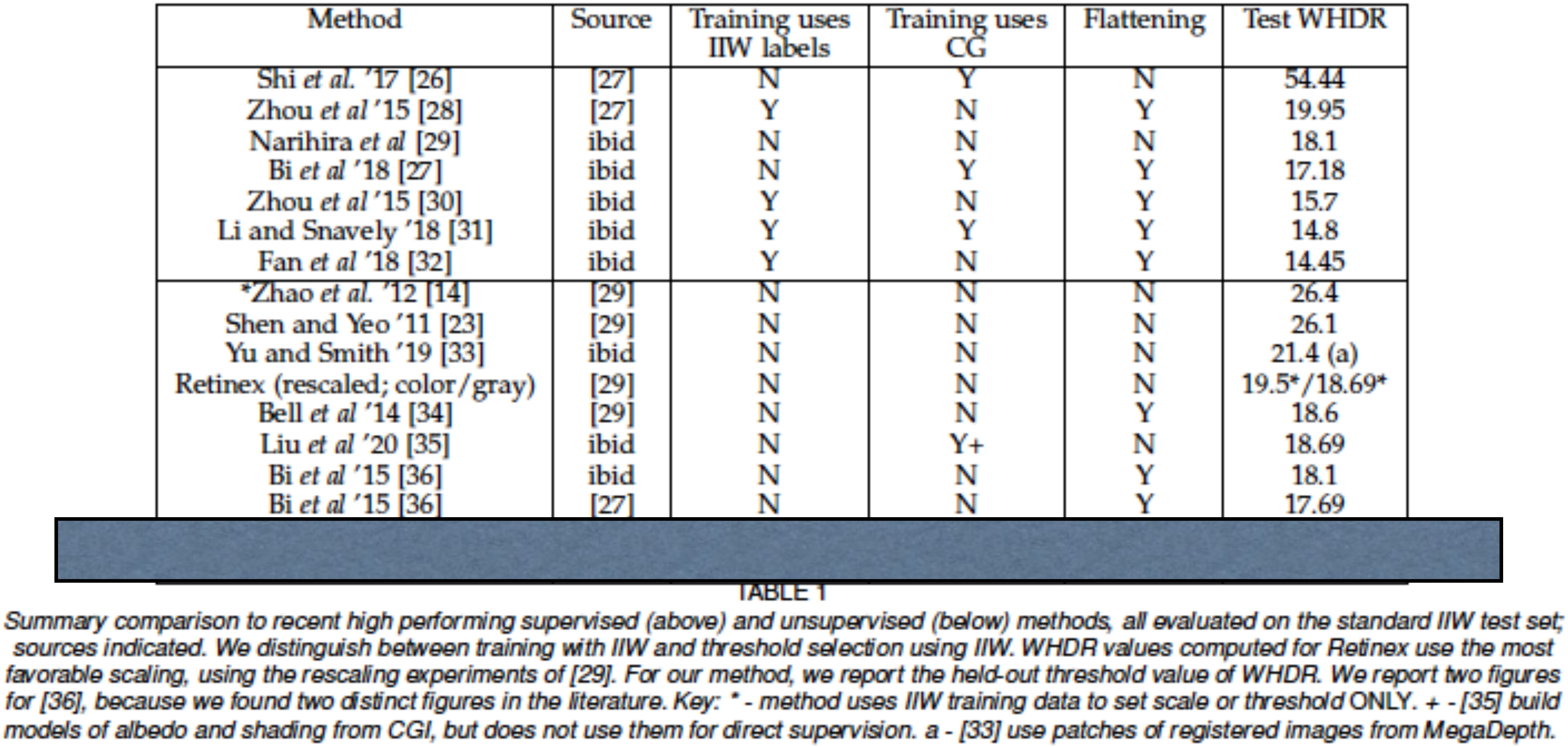
- Apply the priors that
  - albedo is piecewise constant
  - there are “few” albedo values
  - albedo and shading explain image

- Solve
  - eg Bell 14, Nestmeyer 17, Bi 15

Modern strategies - Regression

- Regression of ground truth against image
    - use training set from WHDR data (Narihira et al 2015)
        - and perhaps rendered data
    - surprisingly, rendered data is very helpful
        - Li et al 18; Bi et al 18; Fan et al 18; etc
- Surprising because
    - Albedo in renderings isn’t like albedo in the world
    - Illumination in renderings *really* isn’t like illumination in the world

Recent history



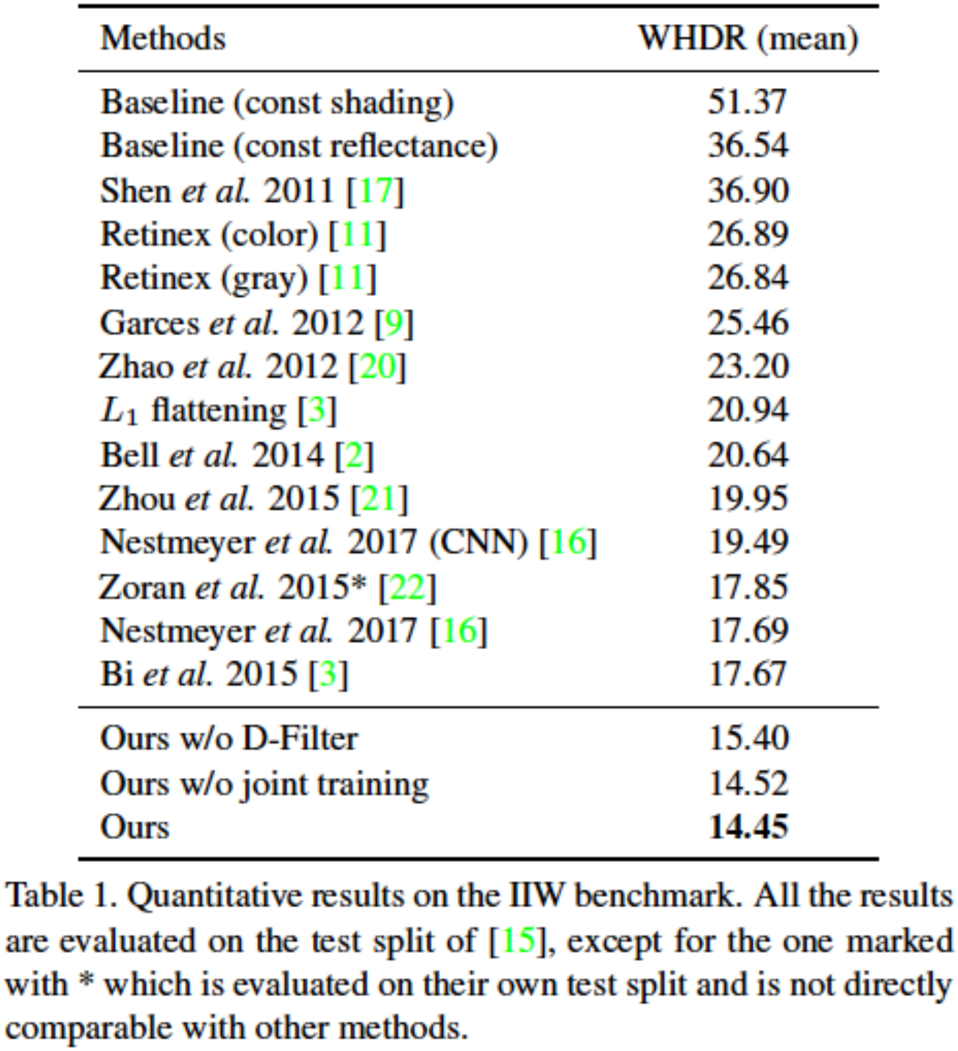
Method	Source	Training uses IIW labels	Training uses CG	Flattening	Test WHDR
Shi *et al.* '17 [26]	[27]	N	Y	N	54.44
Zhou *et al* '15 [28]	[27]	Y	N	Y	19.95
Narihira *et al* [29]	ibid	N	N	N	18.1
Bi *et al* '18 [27]	ibid	N	Y	Y	17.18
Zhou *et al* '15 [30]	ibid	Y	N	Y	15.7
Li and Snavely '18 [31]	ibid	Y	Y	Y	14.8
Fan *et al* '18 [32]	ibid	Y	N	Y	14.45
*Zhao *et al.* '12 [14]	[29]	N	N	N	26.4
Shen and Yeo '11 [23]	[29]	N	N	N	26.1
Yu and Smith '19 [33]	ibid	N	N	N	21.4 (a)
Retinex (rescaled; color/gray)	[29]	N	N	N	19.5*/18.69*
Bell *et al* '14 [34]	[29]	N	N	Y	18.6
Liu *et al* '20 [35]	ibid	N	Y+	N	18.69
Bi *et al* '15 [36]	ibid	N	N	Y	18.1
Bi *et al* '15 [36]	[27]	N	N	Y	17.69
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

TABLE 1

*Summary comparison to recent high performing supervised (above) and unsupervised (below) methods, all evaluated on the standard IIW test set; sources indicated. We distinguish between training with IIW and threshold selection using IIW. WHDR values computed for Retinex use the most favorable scaling, using the rescaling experiments of [29]. For our method, we report the held-out threshold value of WHDR. We report two figures for [36], because we found two distinct figures in the literature. Key: * - method uses IIW training data to set scale or threshold ONLY. + - [35] build models of albedo and shading from CGI, but does not use them for direct supervision. a - [33] use patches of registered images from MegaDepth.*

WHDR is tricky - I

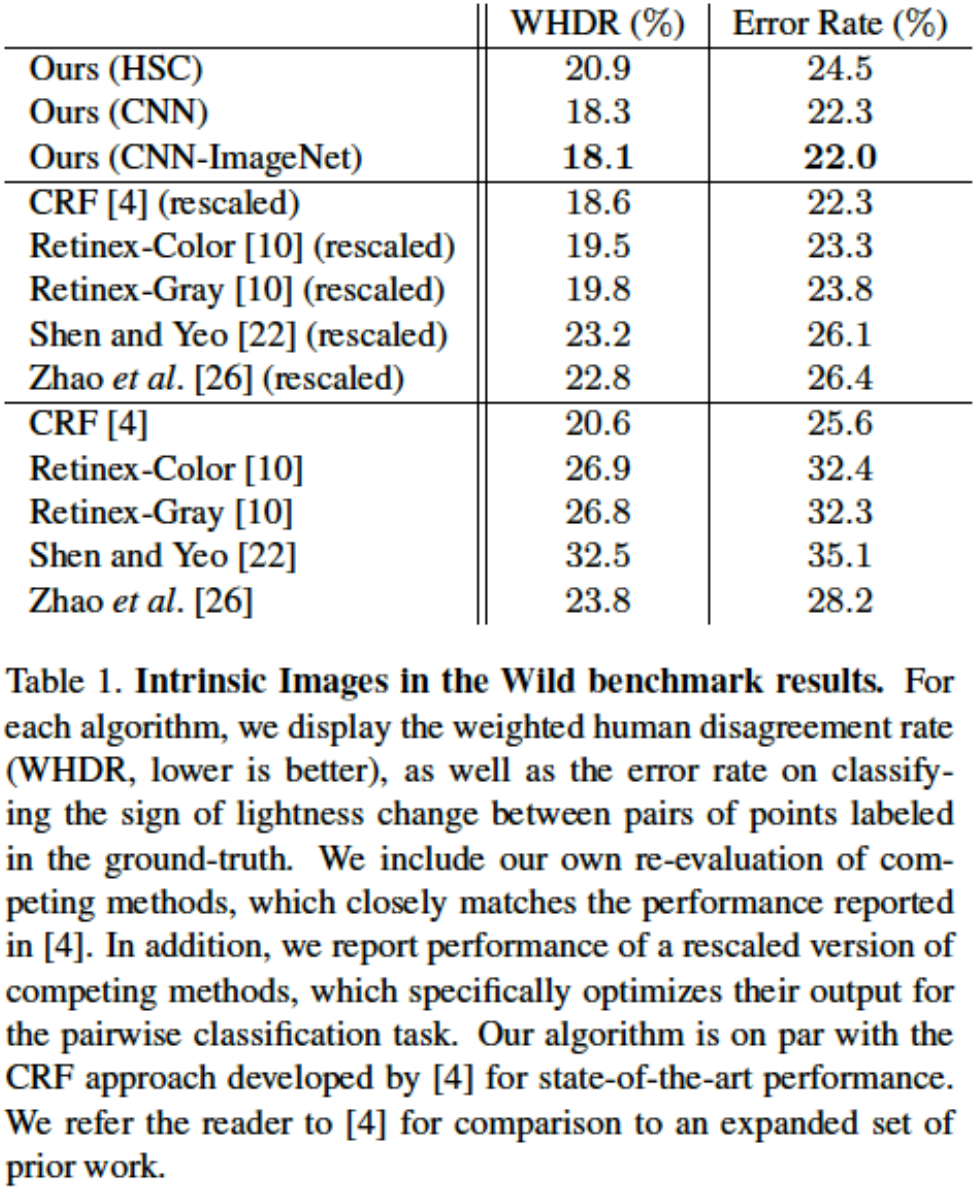
From Fan 18



Methods	WHDR (mean)
Baseline (const shading)	51.37
Baseline (const reflectance)	36.54
Shen *et al.* 2011 [17]	36.90
Retinex (color) [11]	26.89
Retinex (gray) [11]	26.84
Garces *et al.* 2012 [9]	25.46
Zhao *et al.* 2012 [20]	23.20
L_1 flattening [3]	20.94
Bell *et al.* 2014 [2]	20.64
Zhou *et al.* 2015 [21]	19.95
Nestmeyer *et al.* 2017 (CNN) [16]	19.49
Zoran *et al.* 2015* [22]	17.85
Nestmeyer *et al.* 2017 [16]	17.69
Bi *et al.* 2015 [3]	17.67
Ours w/o D-Filter	15.40
Ours w/o joint training	14.52
Ours	**14.45**

Table 1. Quantitative results on the IIW benchmark. All the results are evaluated on the test split of [15], except for the one marked with * which is evaluated on their own test split and is not directly comparable with other methods.

Narihira et al 15



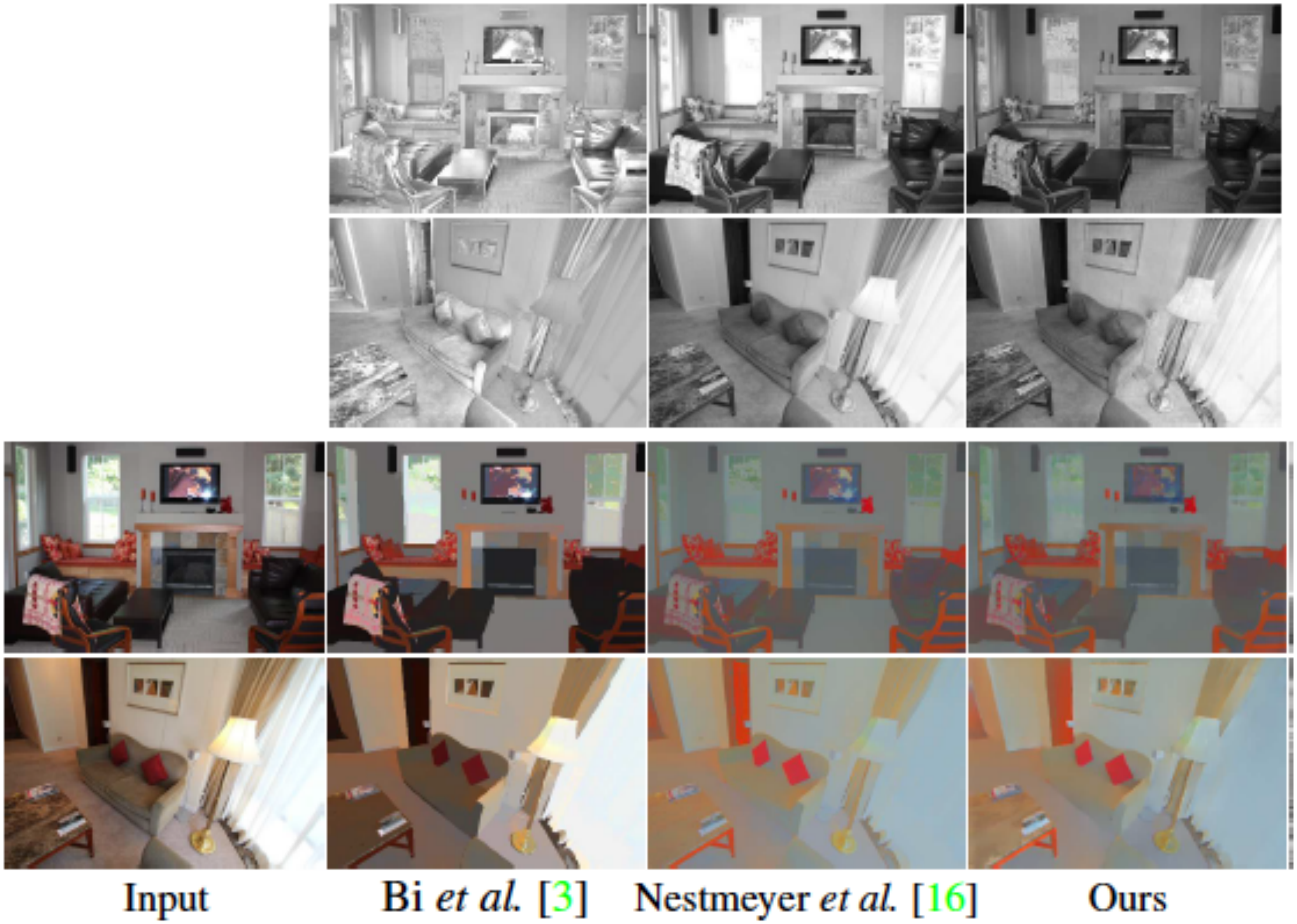
	WHDR (%)	Error Rate (%)
Ours (HSC)	20.9	24.5
Ours (CNN)	18.3	22.3
Ours (CNN-ImageNet)	**18.1**	**22.0**
CRF [4] (rescaled)	18.6	22.3
Retinex-Color [10] (rescaled)	19.5	23.3
Retinex-Gray [10] (rescaled)	19.8	23.8
Shen and Yeo [22] (rescaled)	23.2	26.1
Zhao *et al.* [26] (rescaled)	22.8	26.4
CRF [4]	20.6	25.6
Retinex-Color [10]	26.9	32.4
Retinex-Gray [10]	26.8	32.3
Shen and Yeo [22]	32.5	35.1
Zhao *et al.* [26]	23.8	28.2

Table 1. **Intrinsic Images in the Wild benchmark results.** For each algorithm, we display the weighted human disagreement rate (WHDR, lower is better), as well as the error rate on classifying the sign of lightness change between pairs of points labeled in the ground-truth. We include our own re-evaluation of competing methods, which closely matches the performance reported in [4]. In addition, we report performance of a rescaled version of competing methods, which specifically optimizes their output for the pairwise classification task. Our algorithm is on par with the CRF approach developed by [4] for state-of-the-art performance. We refer the reader to [4] for comparison to an expanded set of prior work.

WHDR is tricky - II

- Predict by
  - f(m1, m2) >t -> 1 is lighter
  - -t<f(m1, m2)<t -> same
  - f(m1, m2)<-t -> 2 is lighter
- Issues:
  - choice of f
    - m1 - m2
    - log(m1/m2)-1
  - choice of m
    - lightness potential
    - predicted albedo
  - choice of threshold
    - interacts with scale

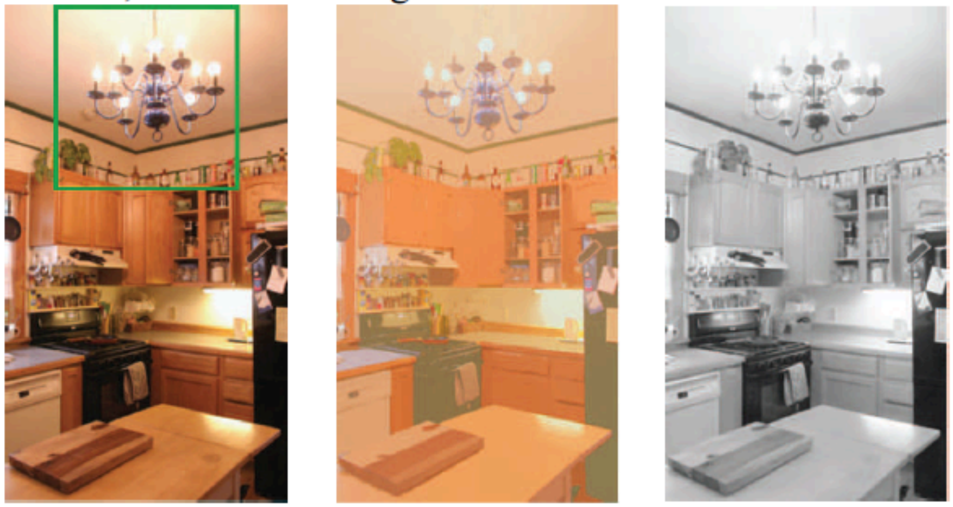
WHDR is tricky - III

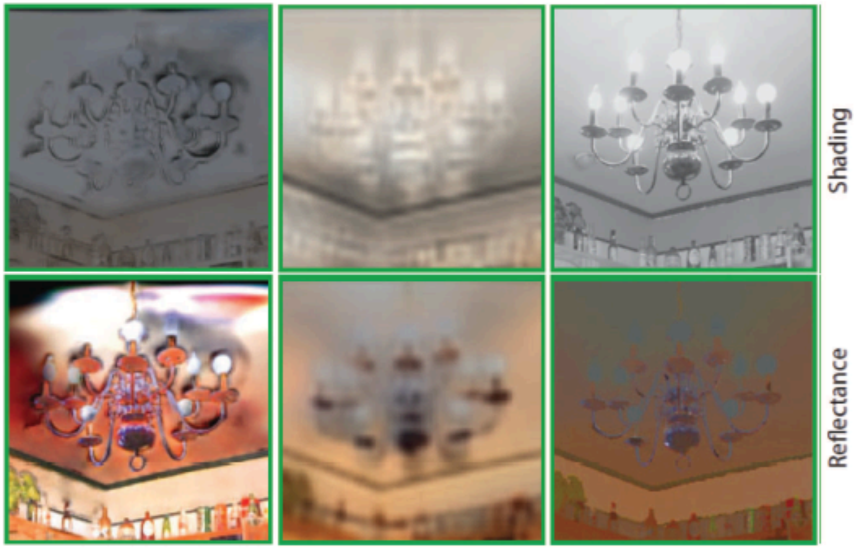


Fan 18 - current SOTA WHDR of 14.45%

WHDR is tricky - IV

Bi et al, 2018 - this image WHDR 6.61%

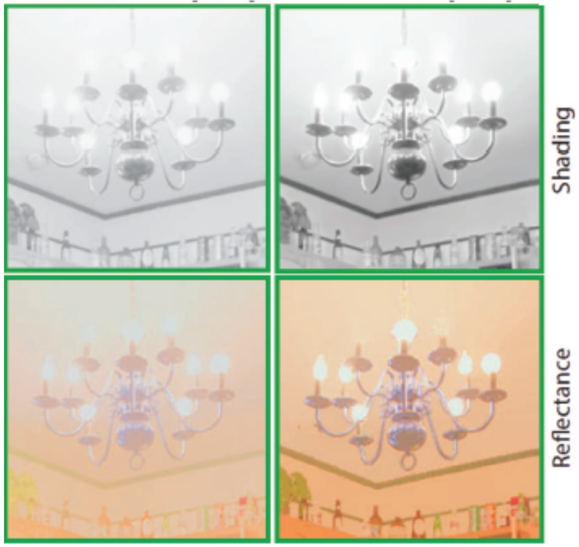




WHDR: 75.70%
Shi et al. [2017]

WHDR: 36.03%
Narihira et al. [2015]

WHDR: 11.48%
Zhou et al. [2015]

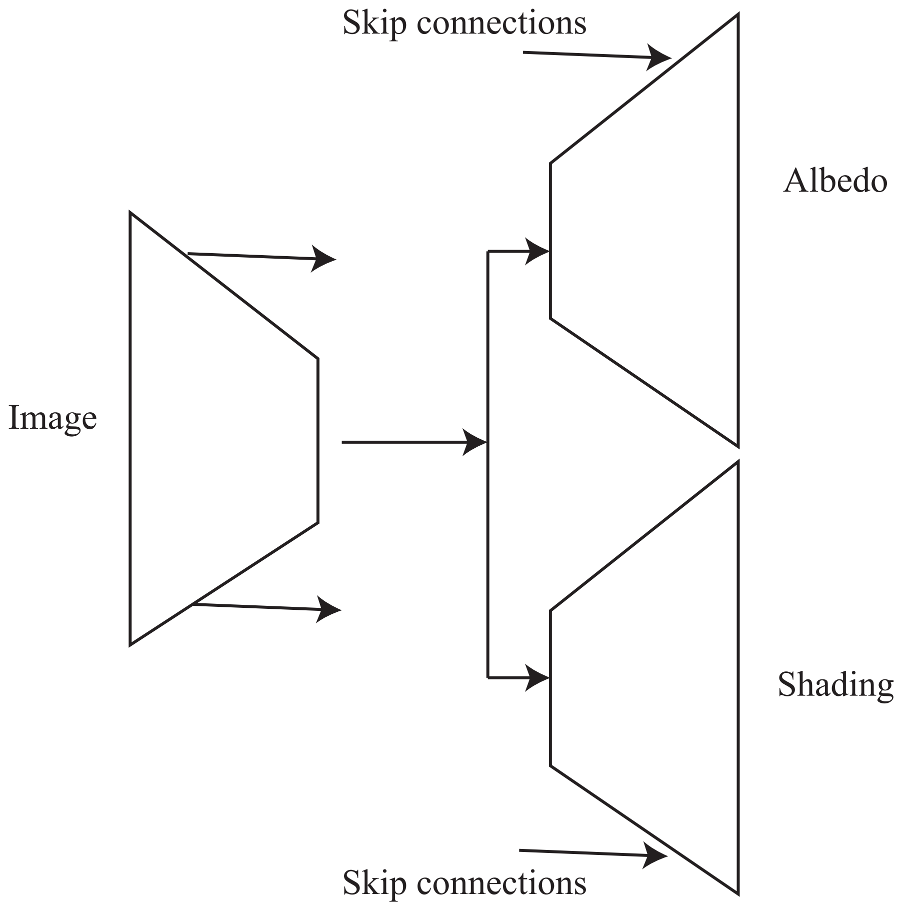


WHDR: 7.35%
Nestmeyer et al. [2017]

WHDR: 6.61%
Bi et al 2018

- Note:
  - odd colors
  - “colored paper” effect
  - “indecision”

One approach (local!)



Skip connections

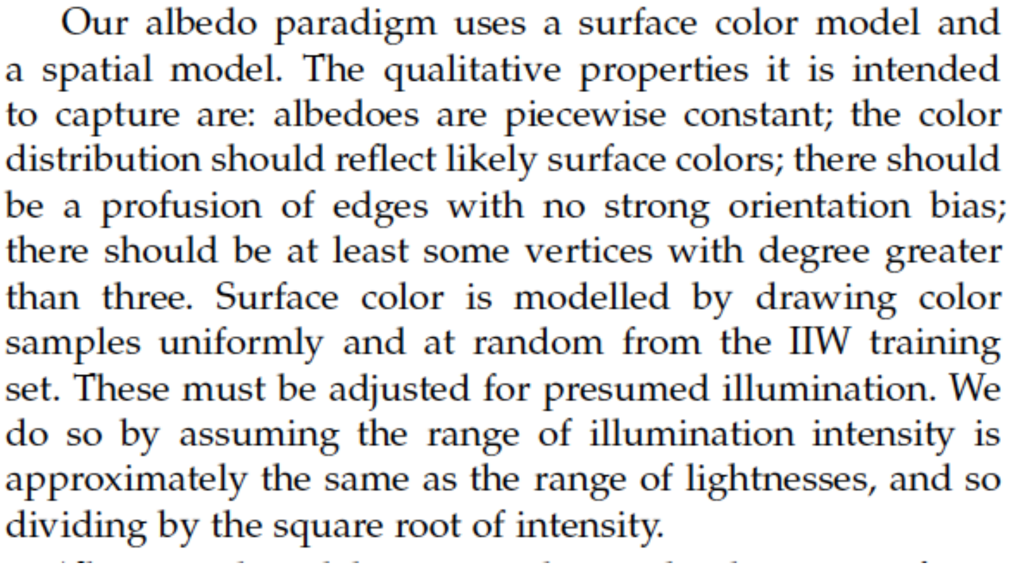
Albedo

Image

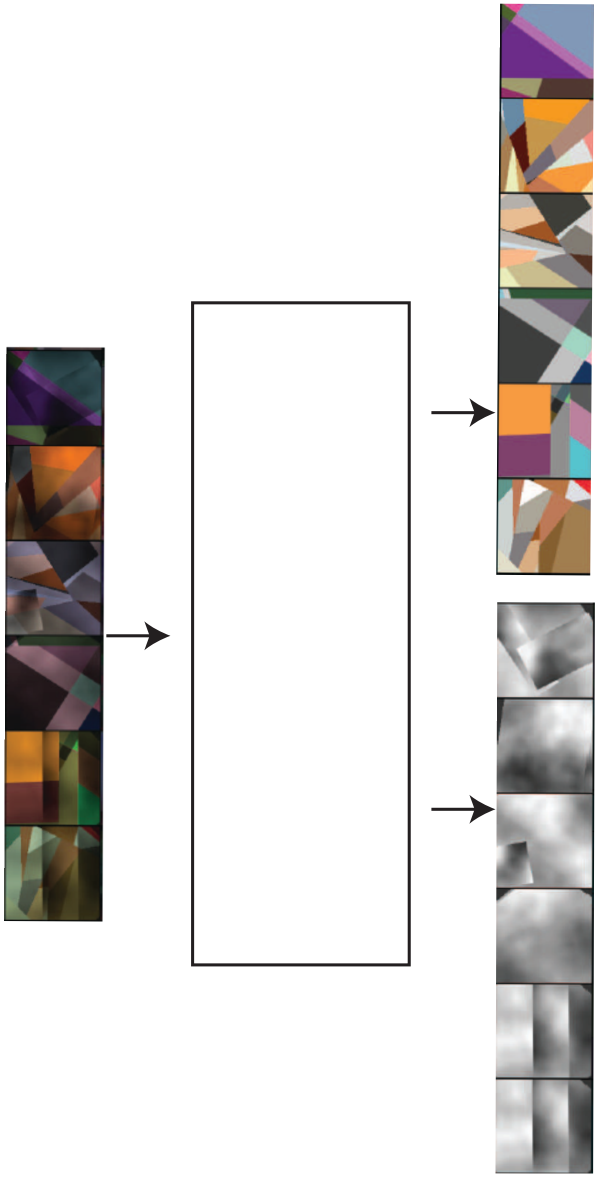
Shading

Skip connections

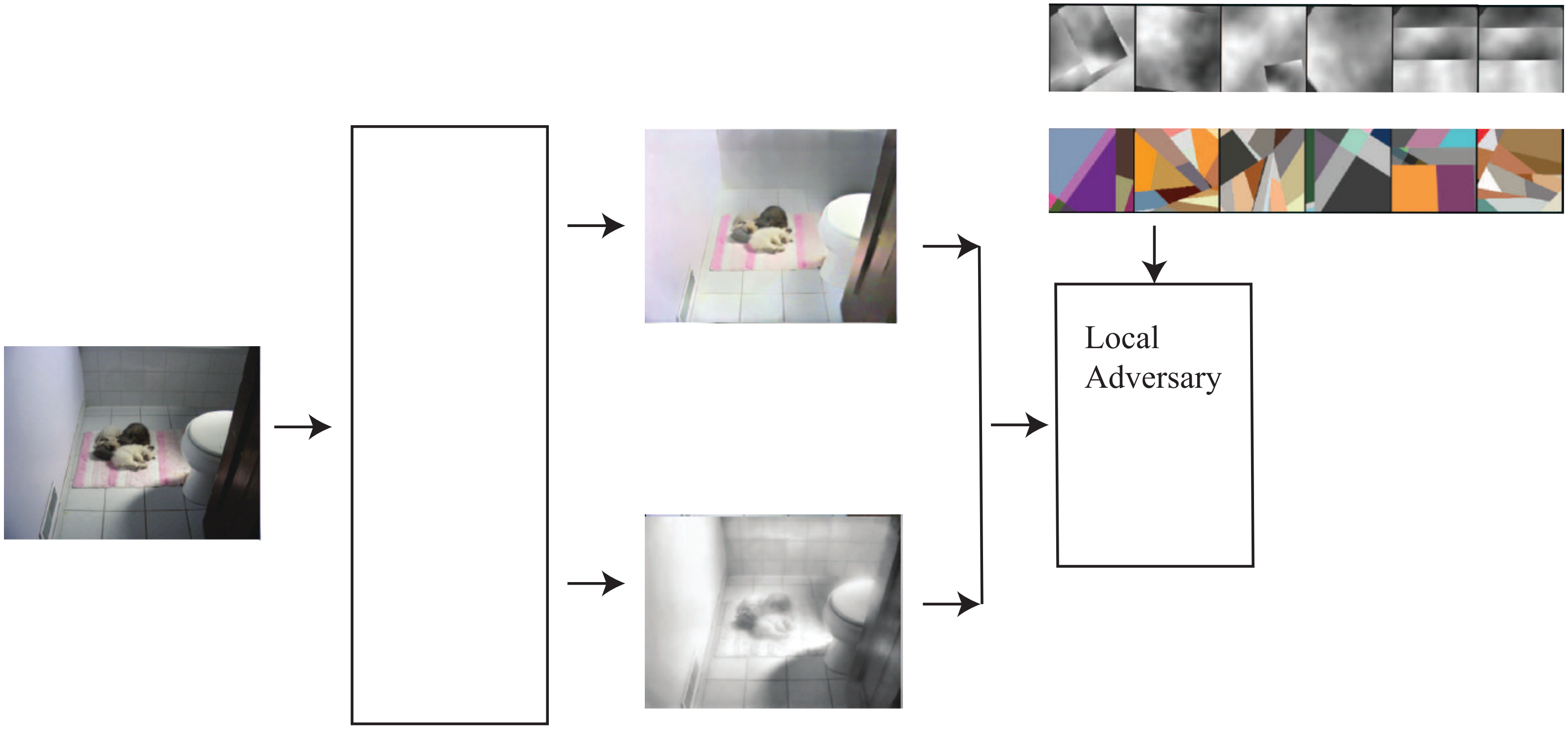
Training - I



Our albedo paradigm uses a surface color model and a spatial model. The qualitative properties it is intended to capture are: albedoes are piecewise constant; the color distribution should reflect likely surface colors; there should be a profusion of edges with no strong orientation bias; there should be at least some vertices with degree greater than three. Surface color is modelled by drawing color samples uniformly and at random from the IIW training set. These must be adjusted for presumed illumination. We do so by assuming the range of illumination intensity is approximately the same as the range of lightnesses, and so dividing by the square root of intensity.



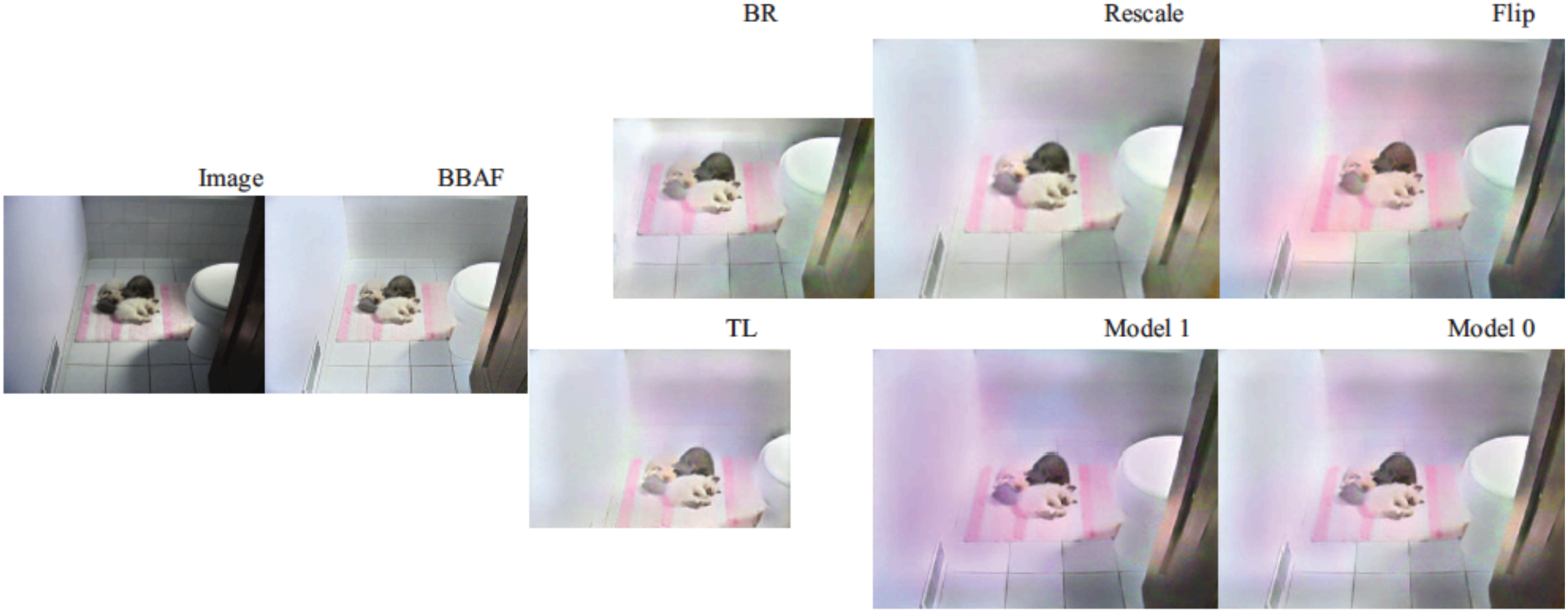
Training - II



Inference

- Network is trained on 128 x 128 tiles of image
- We want equivariance properties from albedo, shading
  - eg translate, rotate, scale image
    - albedo for translated (etc) image should be translated albedo
    - shading for translated (etc) image should be translated shading
- This doesn’t come naturally

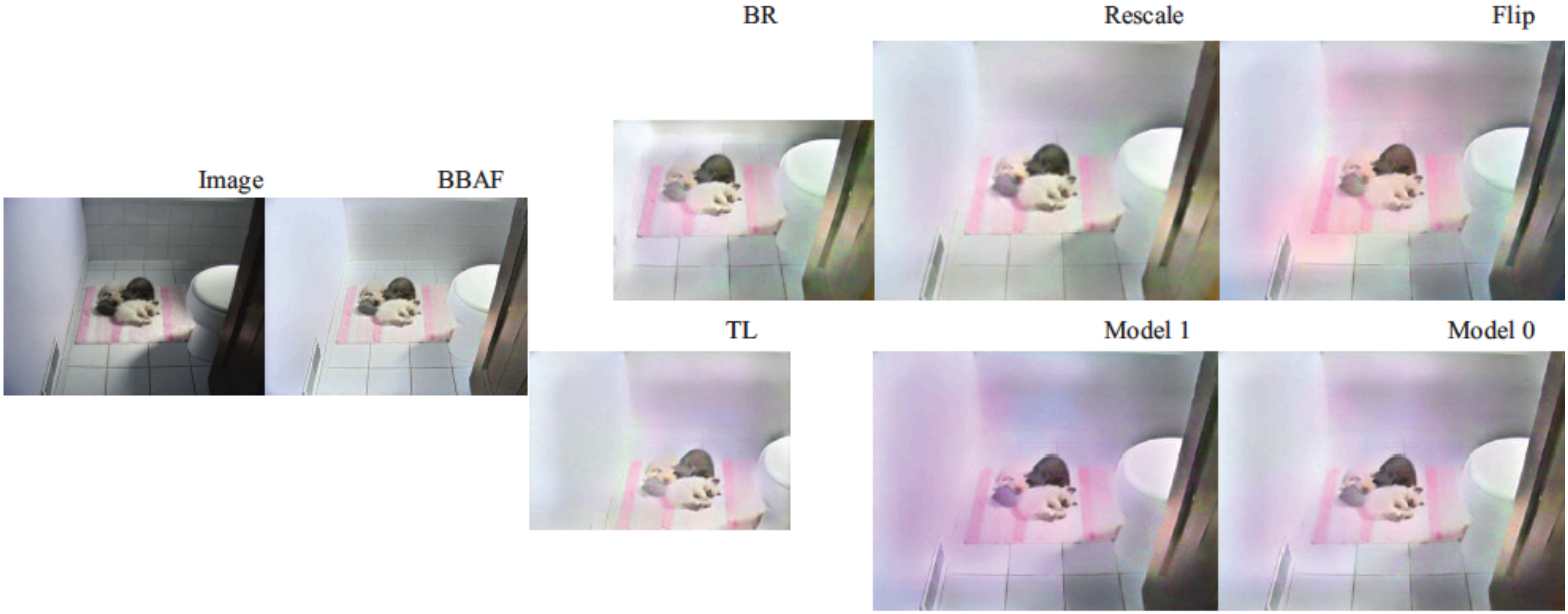
Equivariance must be imposed



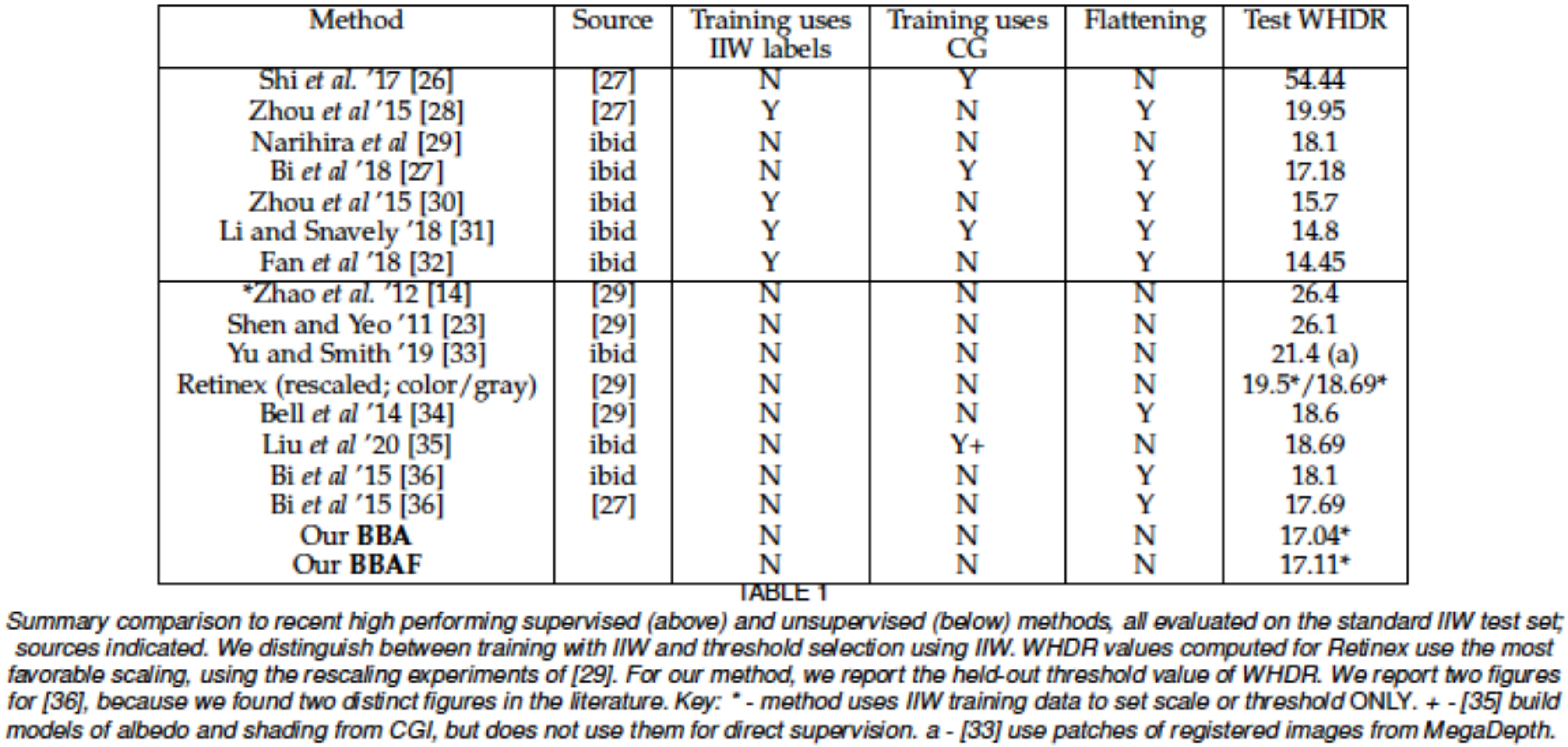
Imposing equivariance

- Translation:
  - cover image with many, shifted, overlapping tiles
  - for each, recover albedo, shading
    - albedo at pixel is weighted average of all overlapping tiles
- Scale:
  - rescale image up, down
    - for each, recover albedo/shading using translation averaging
    - then rescale back
  - average results
- Rotation
  - average estimates from above over 8 flips

Averaging very strongly suppresses error



Results



Method	Source	Training uses IIW labels	Training uses CG	Flattening	Test WHDR
Shi *et al.* '17 [26]	[27]	N	Y	N	54.44
Zhou *et al* '15 [28]	[27]	Y	N	Y	19.95
Narihira *et al* [29]	ibid	N	N	N	18.1
Bi *et al* '18 [27]	ibid	N	Y	Y	17.18
Zhou *et al* '15 [30]	ibid	Y	N	Y	15.7
Li and Snavely '18 [31]	ibid	Y	Y	Y	14.8
Fan *et al* '18 [32]	ibid	Y	N	Y	14.45
*Zhao *et al.* '12 [14]	[29]	N	N	N	26.4
Shen and Yeo '11 [23]	[29]	N	N	N	26.1
Yu and Smith '19 [33]	ibid	N	N	N	21.4 (a)
Retinex (rescaled; color/gray)	[29]	N	N	N	19.5*/18.69*
Bell *et al* '14 [34]	[29]	N	N	Y	18.6
Liu *et al* '20 [35]	ibid	N	Y+	N	18.69
Bi *et al* '15 [36]	ibid	N	N	Y	18.1
Bi *et al* '15 [36]	[27]	N	N	Y	17.69
Our **BBA**		N	N	N	17.04*
Our **BBAF**		N	N	N	17.11*

TABLE 1

*Summary comparison to recent high performing supervised (above) and unsupervised (below) methods, all evaluated on the standard IIW test set; sources indicated. We distinguish between training with IIW and threshold selection using IIW. WHDR values computed for Retinex use the most favorable scaling, using the rescaling experiments of [29]. For our method, we report the held-out threshold value of WHDR. We report two figures for [36], because we found two distinct figures in the literature. Key: * - method uses IIW training data to set scale or threshold ONLY. + - [35] build models of albedo and shading from CGI, but does not use them for direct supervision. a - [33] use patches of registered images from MegaDepth.*

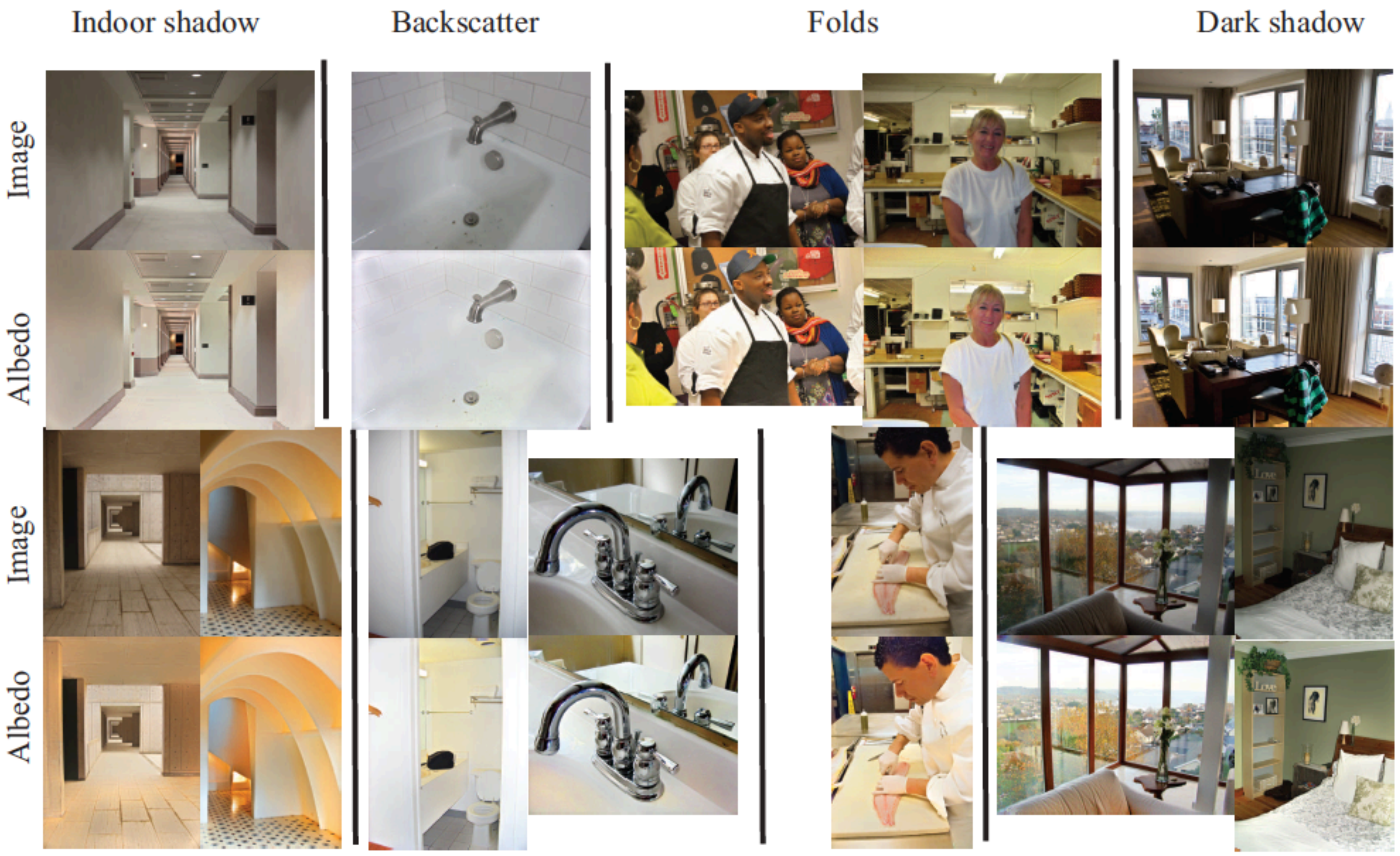


Fig. 2. *Qualitative examples, from our best model (BBAF), showing (L to R): suppression of indoor shadows; suppression of backscatter from shiny bathroom fittings; suppression of fast shading effects from clothing folds; correctly handled dark shadow (couch back).*

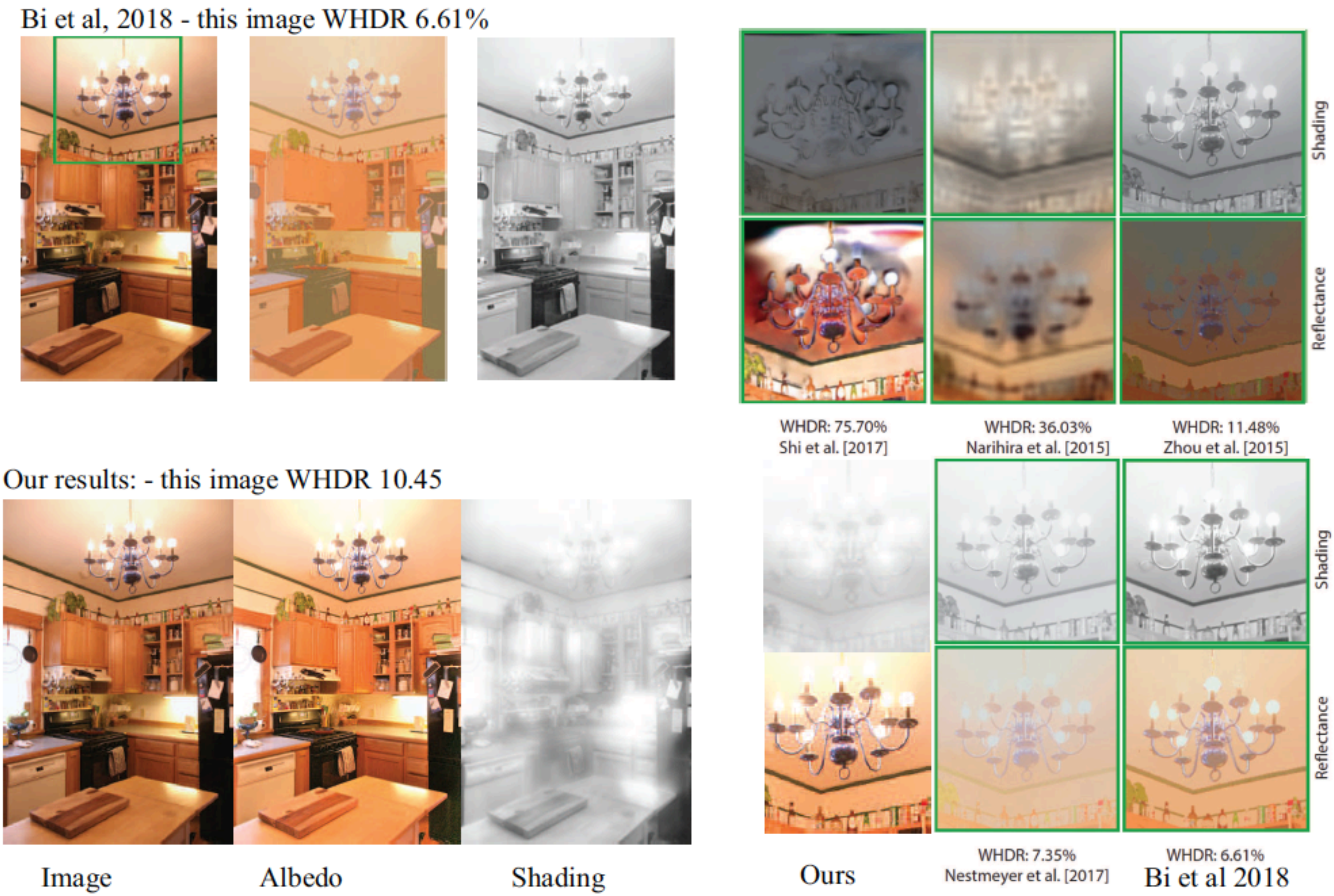


Fig. 6. *Qualitative comparison to [27], [26], [48], [45] and [62], using parts of Figure 1 of [27]. As [27] remark, the methods of [26] and [48] are trained on rendered data alone, and face difficulties due to the difference between rendered data and real images. As [27] remark, the methods of [48] and [45] face difficulties due to the deep shadows in the scene. The albedo produced by our method does not show the "colored paper" effect seen in other methods and does not produce odd colors; this is an advantage (text). Our method reports albedo and shading up to image boundaries, that of [27] appears not to (the crop of the figures is as in the original paper; for our method, we show the whole image).*

Smoothing is important

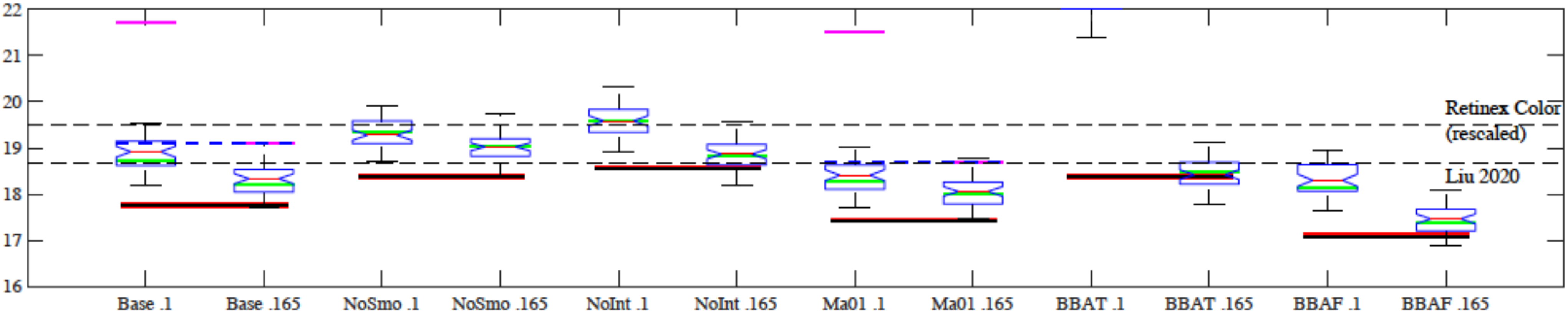


Fig. 8. *Smoothing, averaging and postprocessing are important. Without adversarial smoothing (***NoSmo***), performance is comparable to Retinex. Adversarial smoothing* alone *(***NoInt***) is surprisingly well behaved. Averaging makes a very significant difference (compare blue/black bars and purple/green bars) and averaging over a larger number of tiles is better (cf.* **BBA** *and* **Base***). Discrete image averaging results in improvements (cf.* **BBA** *and* **BBAF***), and is clearly better than discrete tile averaging (cf.* **BBAF** *and* **BBAT***).* **Key:** Fixed thresholds: *shown in boxplots of WHDR values for 50 simulated test sets for the two fixed thresholds, and green bars are the value for the standard test set.* Oracle thresholds: *heavy black bar.* Held out threshold: *heavy red bar.* Oracle threshold without smoothing: *heavy blue dashed bar.* Fixed threshold without smoothing: *heavy purple bar.* Boxplots: *horizontal bar = median; notch = fraction of interquartile range outside which a difference in medians is significant; bottom and top of the box = 25 and 75 percentiles resp.; whiskers extend to the most extreme data points that are not outliers; outliers – greater than 1.5 times the interquartile range outside top and bottom – are '+'.* Best viewed in color.

Paradigms beat graphics

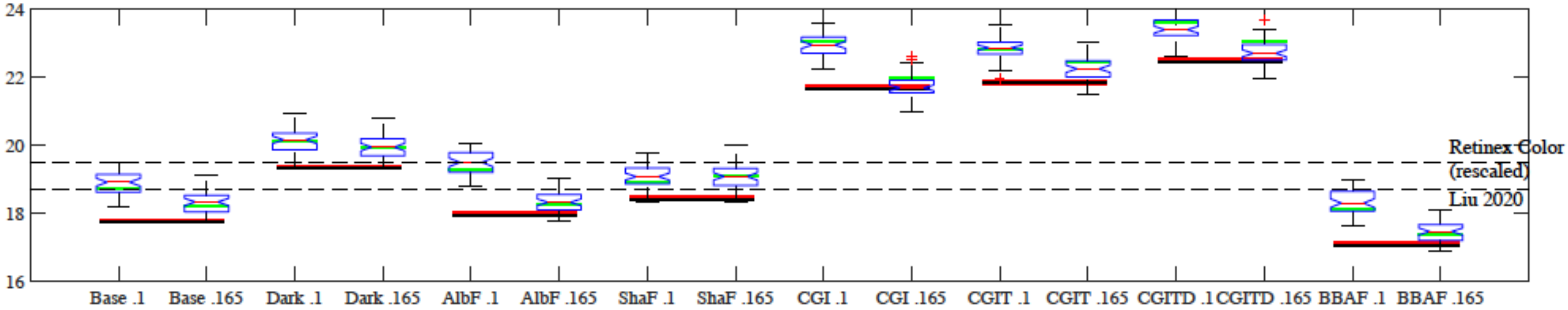


Fig. 9. *Varying the details of the paradigm has some effect; a* Dark *shading paradigm creates notable difficulties, but varying the size of shading (*ShaF*) and albedo (*ShaF*) fragments seems to have only minor effects. Using tiles excerpted from CGIntrinsics [47] leads to significant fall off in performance (*CGI *– tiles extracted from CGIntrinsics at original scale;* CGIT *– extracted from images shrunk so that tiles contain more detaile;* CGITD *– dependency between shading and albedo preserved). Graphical conventions as in Figure 5.* Best viewed in color.

Scale matters

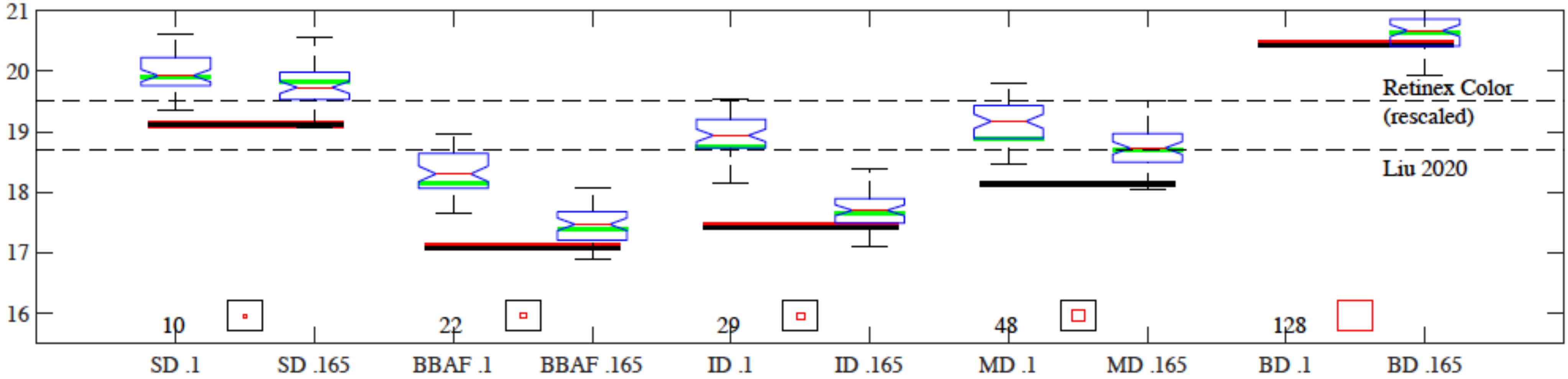


Fig. 10. *Varying the scale of the discriminator has an important effect on performance.* **SD** *the discriminator sees* 10×10 *patches;* **BBAF** *as in other figures our best model,* 22×22*;* **ID** 29×29*;* **MD** 48×48*; and* **BD** 128×128*. The scale of* **ID** *was chosen by interpolating oracle WHDR for the others, then choosing the scale that produced the best predicted WHDR. The red boxes show the scale of the discriminator patches with respect to the tile (black boxes) for each model. Graphical conventions as in Figure 5.* Best viewed in color.

Indecisiveness remains (aargh!)

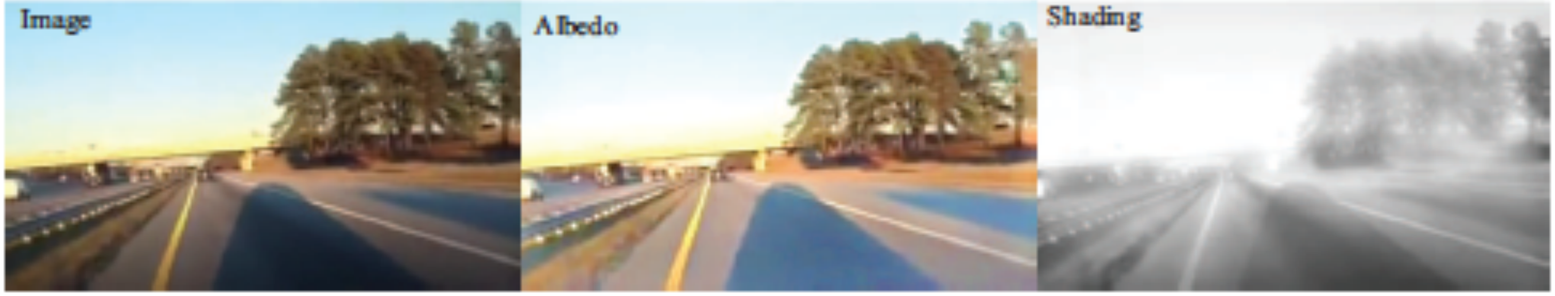


Fig. 13. *Our method suffers indecisiveness, as do others; this is a persistent problem in intrinsic image methods. Figures show a decomposition of an outdoor image, using our method. Note the pronounced shadow leaves effects in both albedo and shading fields; versions of this effect for other methods can be seen in Figure 6.* Best viewed in color.

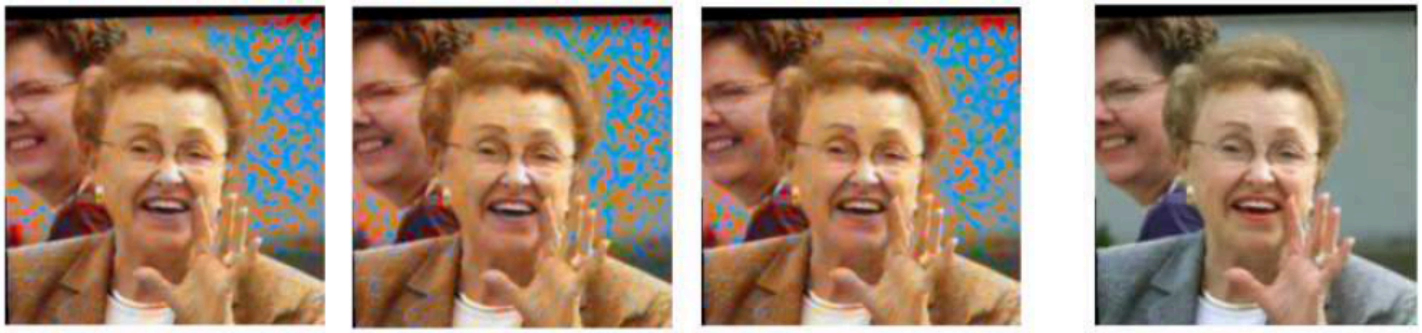
Other Possible Intrinsics

- Surface relief and material properties
  - and perhaps many of them
- Surface mechanical properties
- Surface glossiness
- Texture flow

Learning Conditional Models

- Learn P(Y|X) from examples
  - P(Y|X) is wildly multimodal
  - usually, Y is strongly variable and has high spatial correlations
- Model problems:
  - SPADE:
    - X is semantic labels, Y is image
  - Colorization
    - X is a grey level image, Y a color field
  - Reshading
    - X is an albedo image of a scene, Y a shading field
  - Motion
    - X is image, Y is optic flow

Spatial correlations are a real problem



(a) Sampling per-pixel distribution of [30]

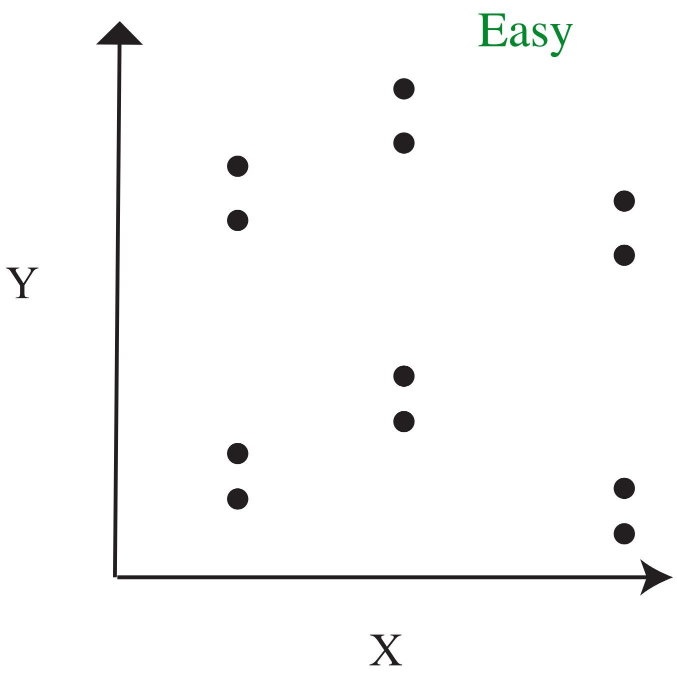
(b) Ground truth

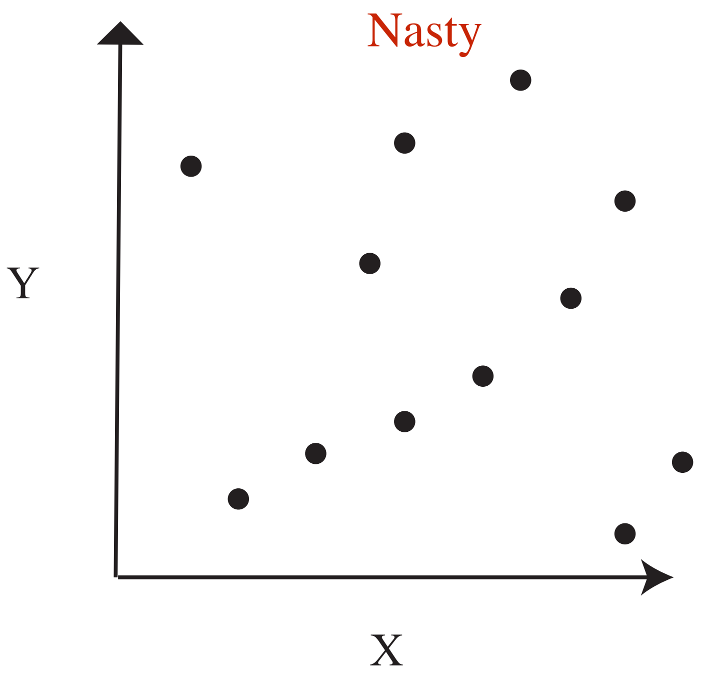
Figure 2: Zhang et al. [30] predict a per-pixel probability distribution over colors. First three images are diverse colorizations obtained by sampling the per-pixel distributions independently. The last image is the ground-truth color image. These images demonstrate the speckled noise and lack of spatial co-ordination resulting from independent sampling of pixel colors.

Deshpande et al 17

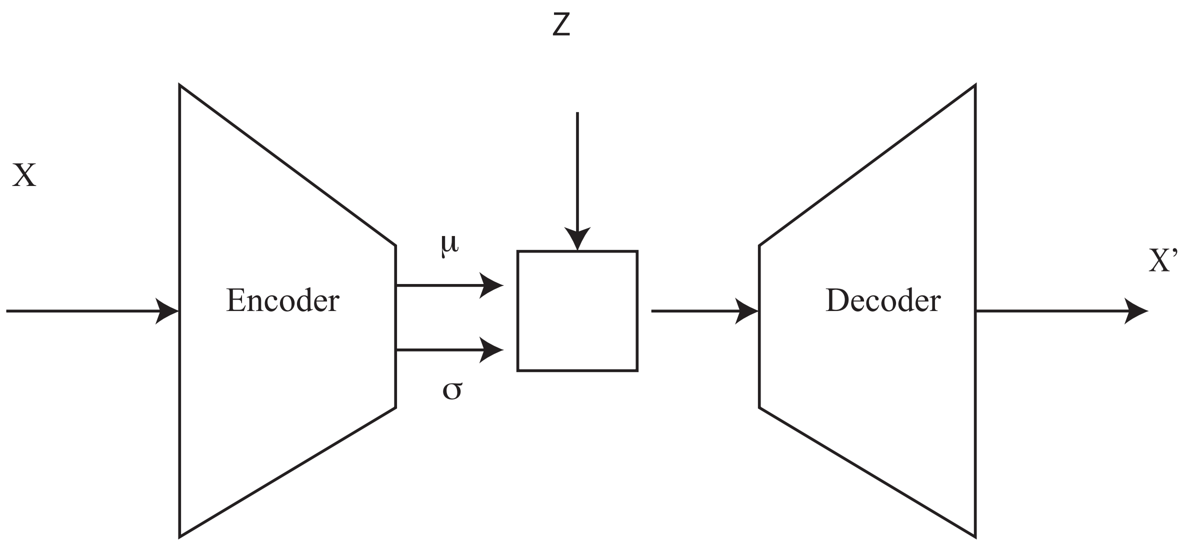
Learning Conditional Models

- Learn P(Y|X) from examples
  - (Y, X) pairs
  - cases:
    - many pairs share an X: easy, very uncommon
    - pairs have different X's: very hard, confusing, common

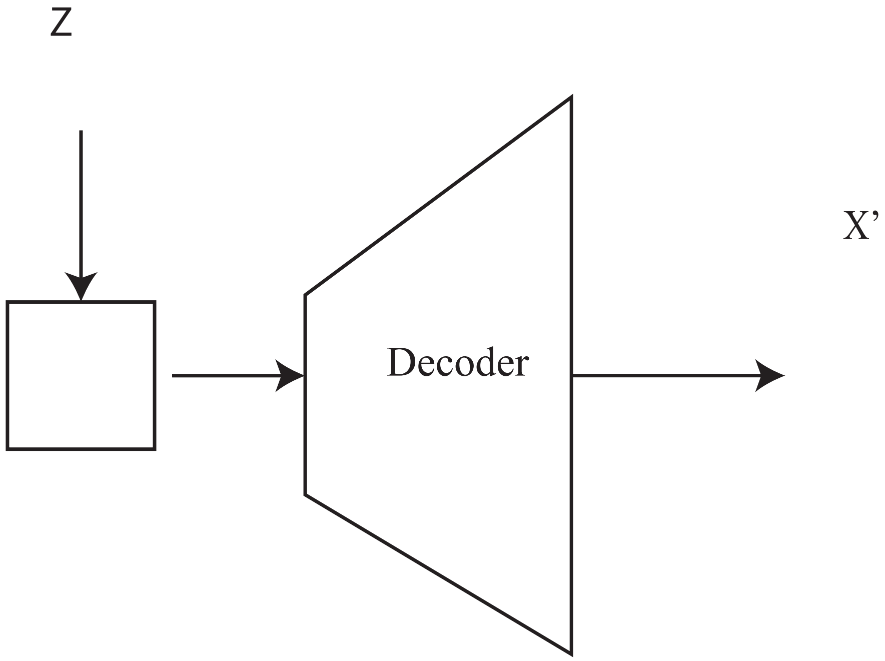




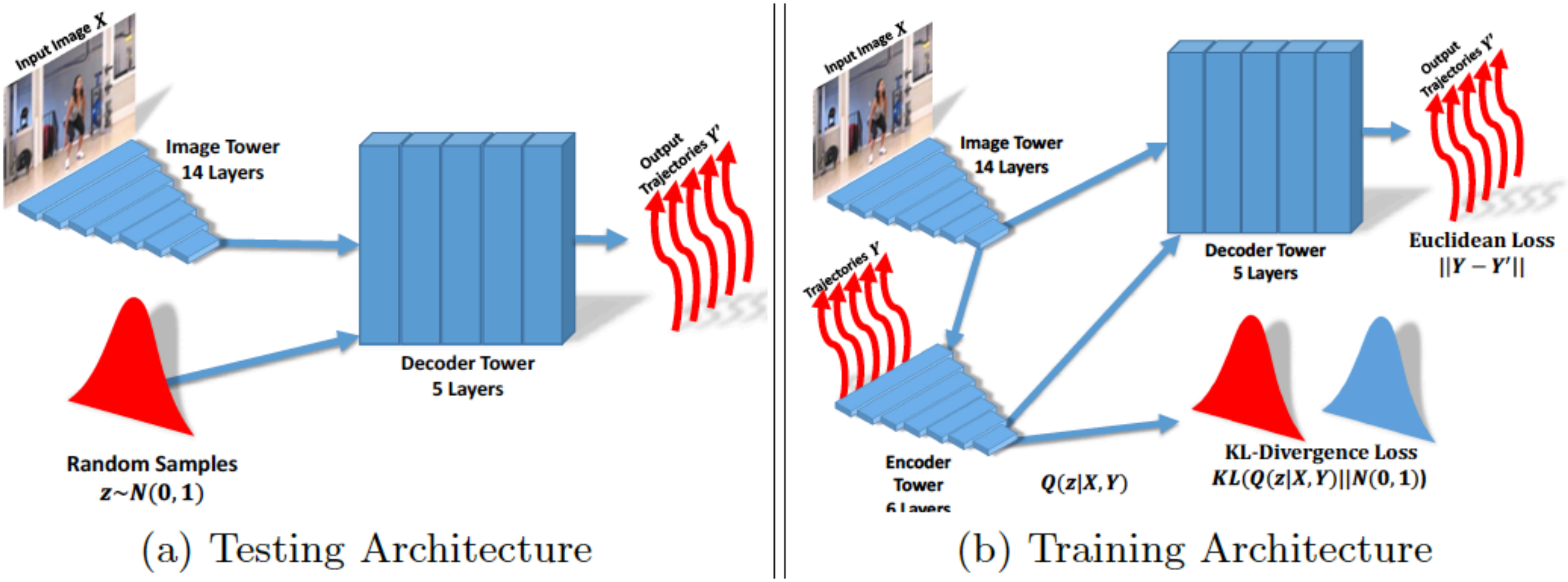
Quick and dirty background: VAE



Quick and dirty background: VAE



Conditioning a VAE (CVAE)



(a) Testing Architecture

(b) Training Architecture

Fig. 2. Overview of the architecture. During training, the inputs to the network include both the image and the ground truth trajectories. A variational autoencoder encodes the joint image and trajectory space, while the decoder produces trajectories depending both on the image information as well as output from the encoder. During test time, the only inputs to the decoder are the image and latent variables sampled from a normal distribution.

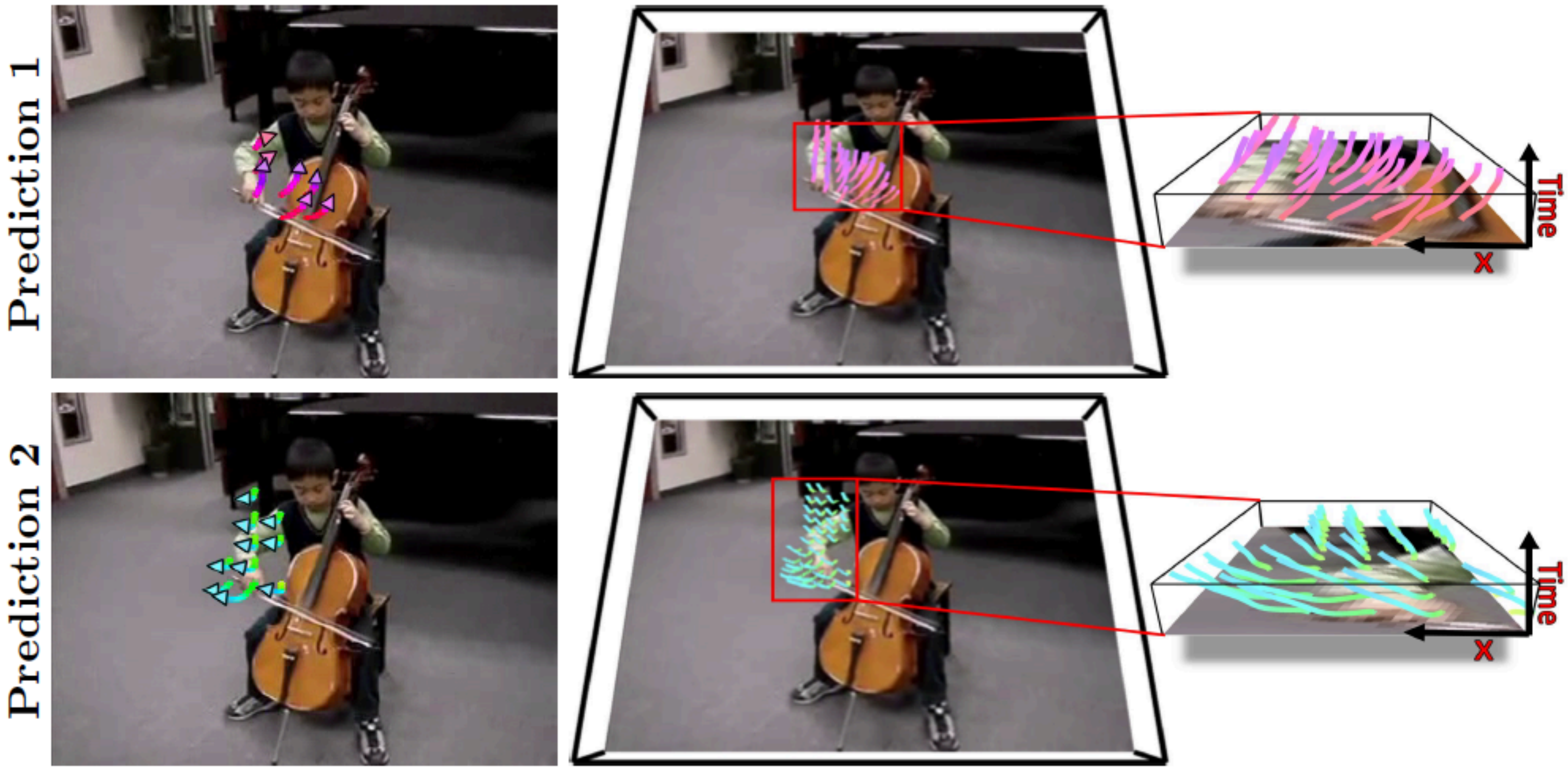
Walker et al 16

Procedure

- Want a model of P(Y|X)
  - but must smooth
    - learn c(X) (a code)
      - such that “similar” images have “similar” codes
    - and build model of P(Y|c(X))
- Draw samples by
  - y=F(z; c(X))
- The loss you use is very important

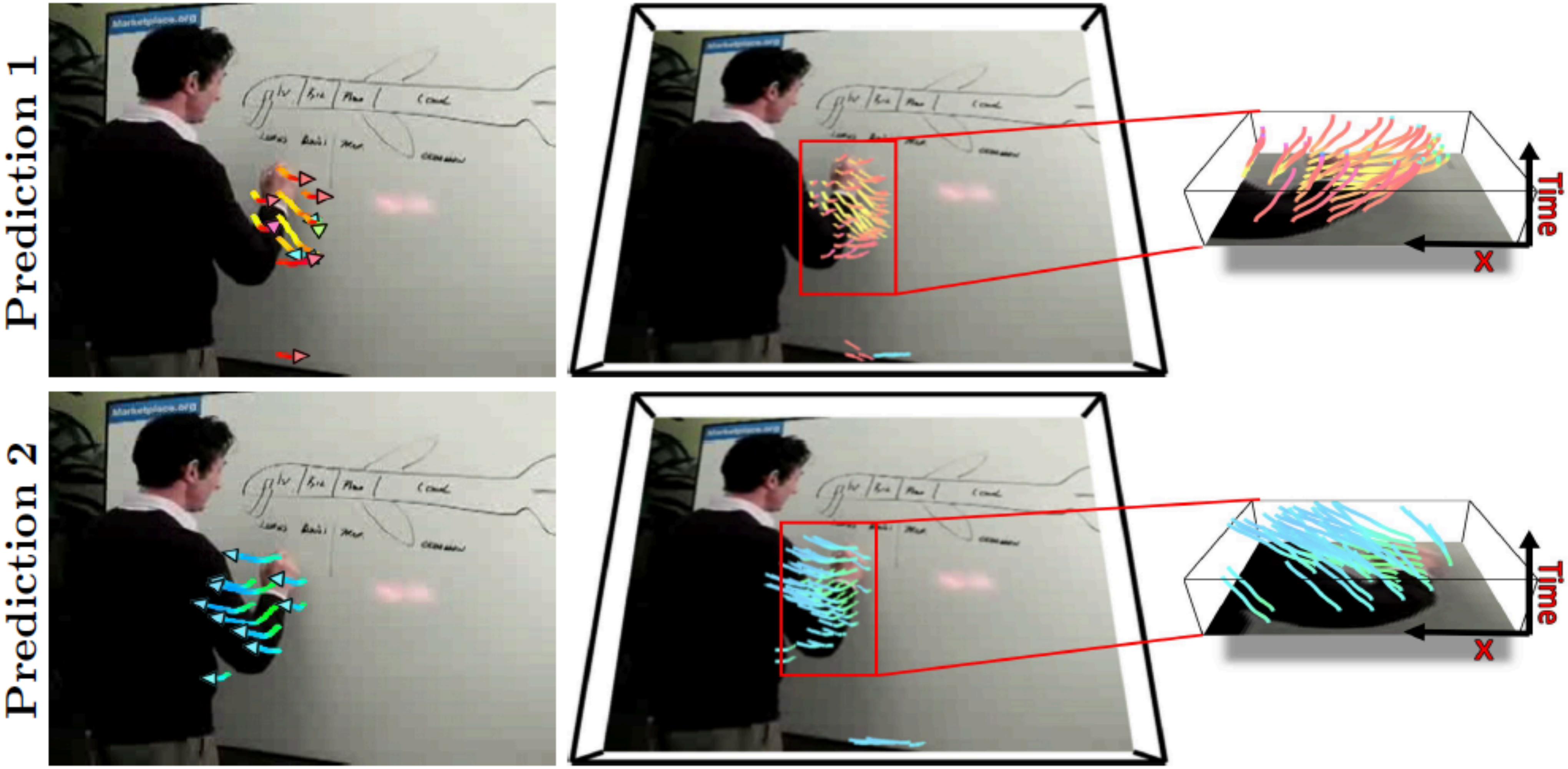
- Walker et al use (essentially)
  - - conditional log-likelihood of Y_i | X_i

Motion prediction



Walker et al 16

Motion prediction

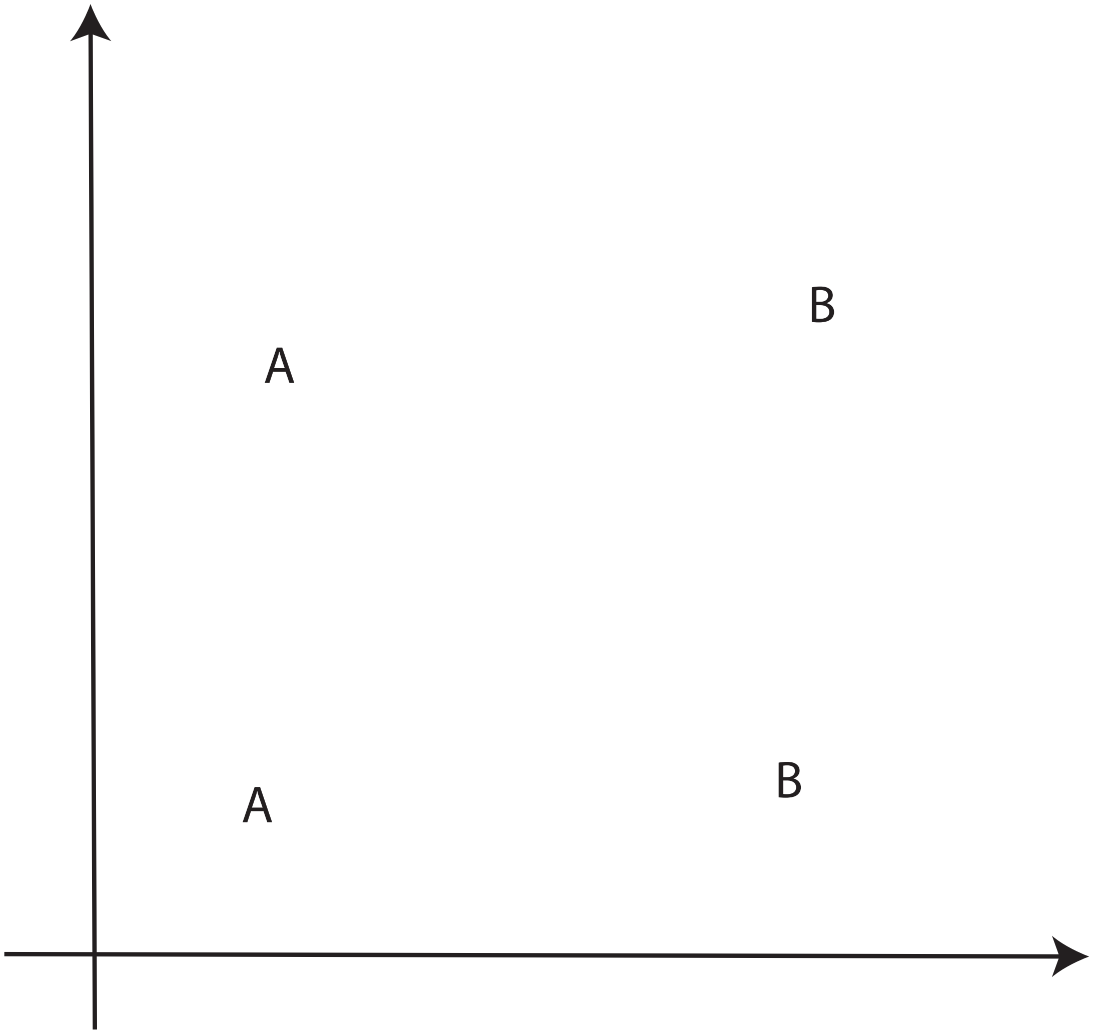


Walker et al 16

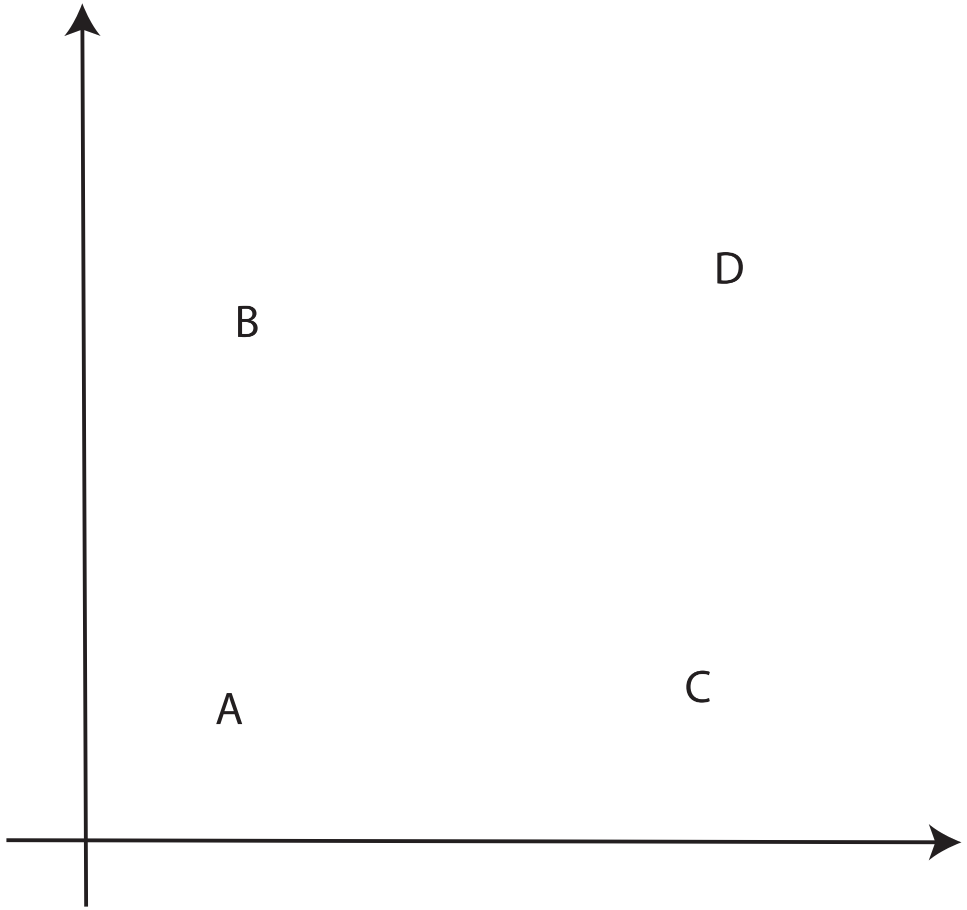
Giant issue: code collapse

- Draw samples by
  - y=F(z; c(X))
- But how was c(X) chosen?
  - Imagine a hostile player chose c(X) malignantly
  - We could get very odd p(Y|X)

Code collapse - I



Code collapse - II

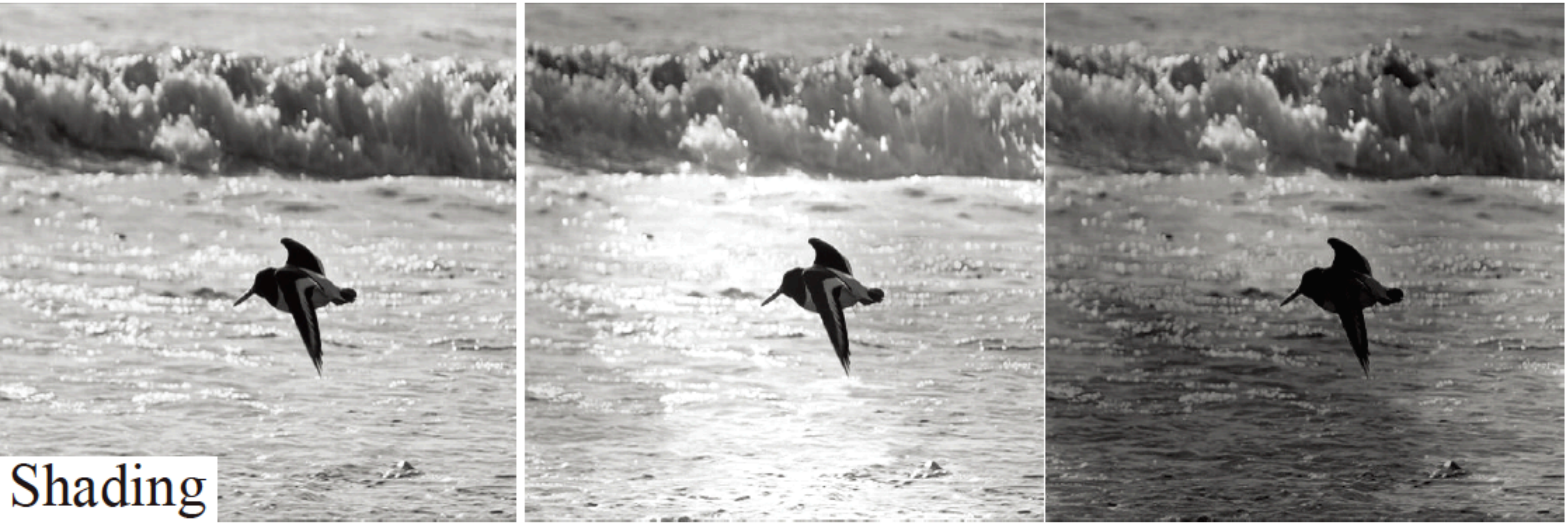


Giant issue - how do we evaluate?

- Model is trained so that Y_i | X_i has high log-likelihood
    - good loss (=high likelihood) on held out Y, X pairs is a good sign
    - BUT
        - we want to be sure that
            - the model gives diverse Y for a given X_i
            - and these are all right
        - high likelihood on held out Y, X pairs might come from code collapse

Containing this problem

- Build codes for Y and for X
  - that can be decoded to produce output/input
  - then build explicit models of P(c(Y)|c(X))
    - e.g. mixture density network



Lu et al , ND

Diverse colorization

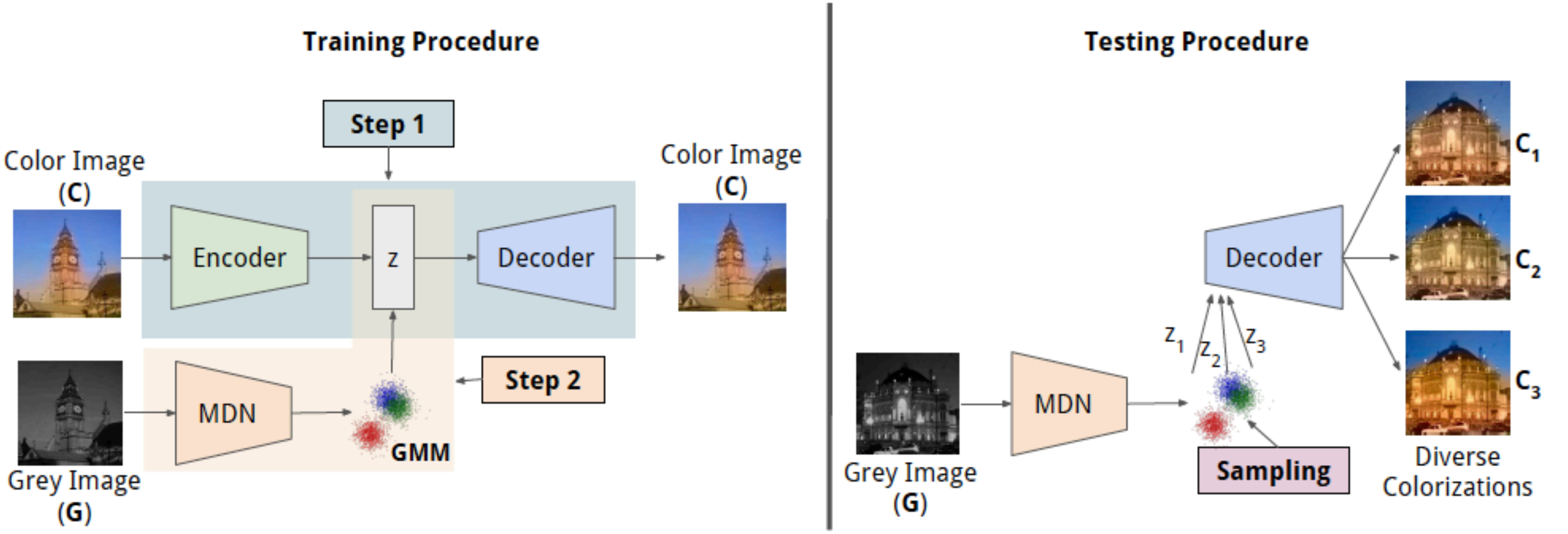
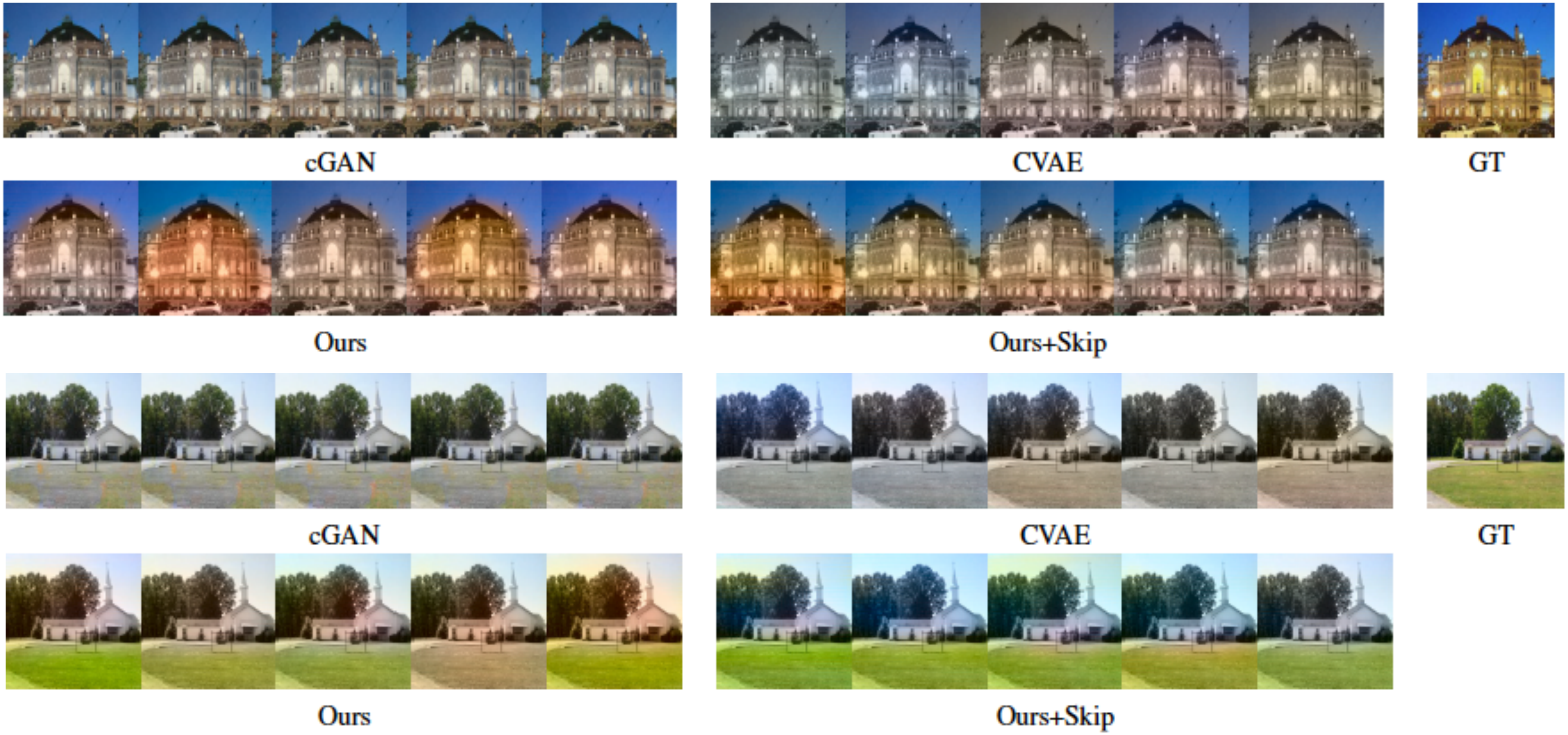


Figure 1: Step 1, we learn a low-dimensional embedding $\mathbf{z}$ for a color field $\mathbf{C}$. Step 2, we train a multi-modal conditional model $P(\mathbf{z}|\mathbf{G})$ that generates the low-dimensional embedding from grey-level features $\mathbf{G}$. At test time, we can sample the conditional model $\{\mathbf{z}_k\}_{k=1}^N \sim P(\mathbf{z}|\mathbf{G})$ and use the VAE decoder to generate the corresponding diverse color fields $\{\mathbf{C}_k\}_{k=1}^N$.

Deshpande et al 17

Colorization



cGAN

CVAE

GT

Ours

Ours+Skip

cGAN

CVAE

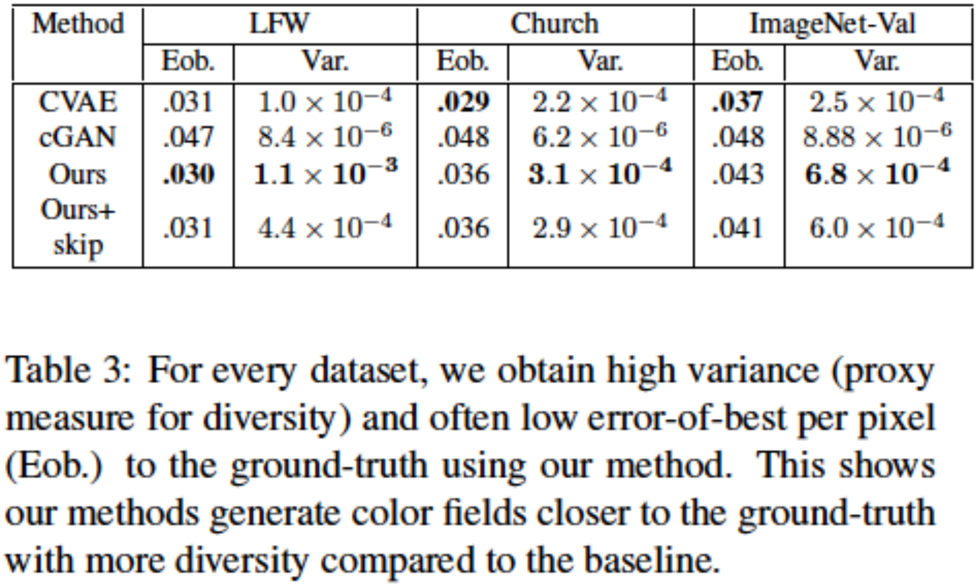
GT

Ours

Ours+Skip

Deshpande et al 17

Evaluation



Method	LFW		Church		ImageNet-Val	
	Eob.	Var.	Eob.	Var.	Eob.	Var.
CVAE	.031	1.0×10^{-4}	**.029**	2.2×10^{-4}	**.037**	2.5×10^{-4}
cGAN	.047	8.4×10^{-6}	.048	6.2×10^{-6}	.048	8.88×10^{-6}
Ours	**.030**	$\mathbf{1.1 \times 10^{-3}}$	.036	$\mathbf{3.1 \times 10^{-4}}$	.043	$\mathbf{6.8 \times 10^{-4}}$
Ours+ skip	.031	4.4×10^{-4}	.036	2.9×10^{-4}	.041	6.0×10^{-4}

Table 3: For every dataset, we obtain high variance (proxy measure for diversity) and often low error-of-best per pixel (Eob.) to the ground-truth using our method. This shows our methods generate color fields closer to the ground-truth with more diversity compared to the baseline.

Deshpande et al 17

Evaluation by joint distribution

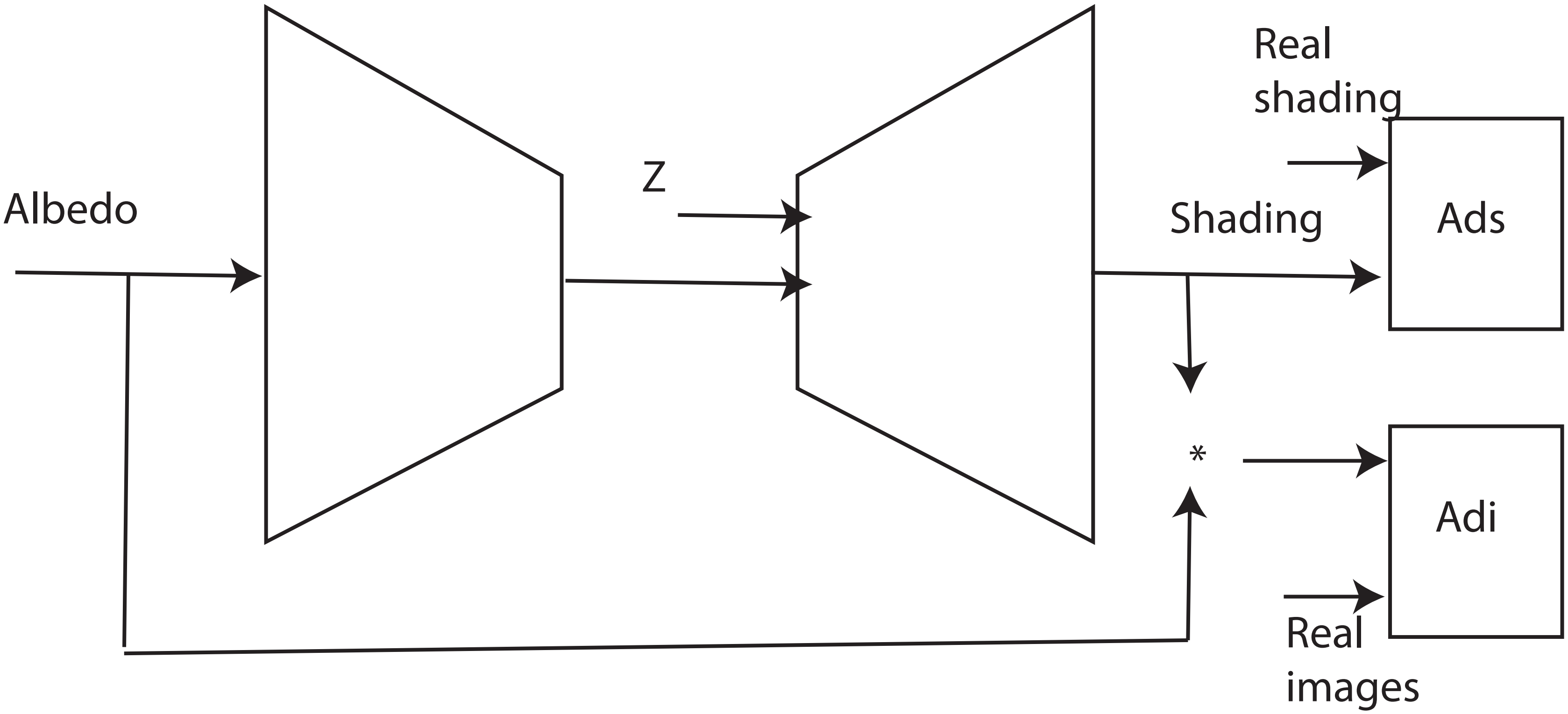
- Assume X_i ~ P(X)
  - write Y_j (X_i) for a sample drawn from model conditioned on X_i
  - then IF model is correct
    - (Y_j(X_i), X_i) ~ P(Y, X)=P(Y|X)P(X)
  - we can evaluate this using FID
- Check
  - FID[{(Y_j(X_i), X_i)}, {(Y_i, X_i)}]
  - AND
  - Y_j far from Y_i

This suggests a training strategy….

- We train a CGAN to produce Ys from Xs, requiring
  - the Y’s are “like” real Y’s
    - i.e. P_m(Y) close to P(Y)
  - the (X, Y) pairs are “like” real pairs
    - i.e. P_m(Y, X) close to P(Y, X)

- We already have X_i ~ P(X)
  - this, together with conditions above is necessary, but not sufficient
    - for P_m(Y|X) to be right

What if…?



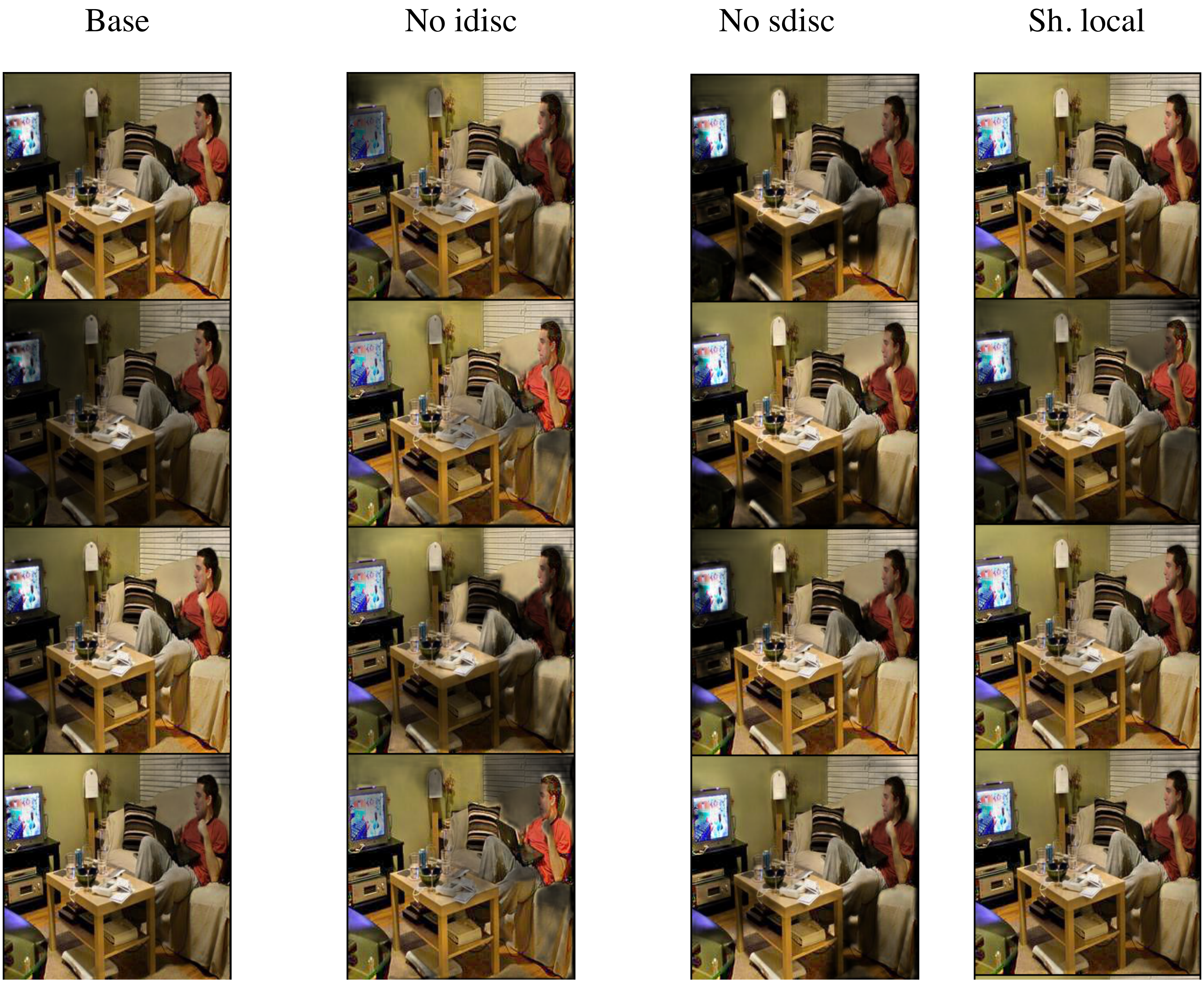
+ Diversity loss

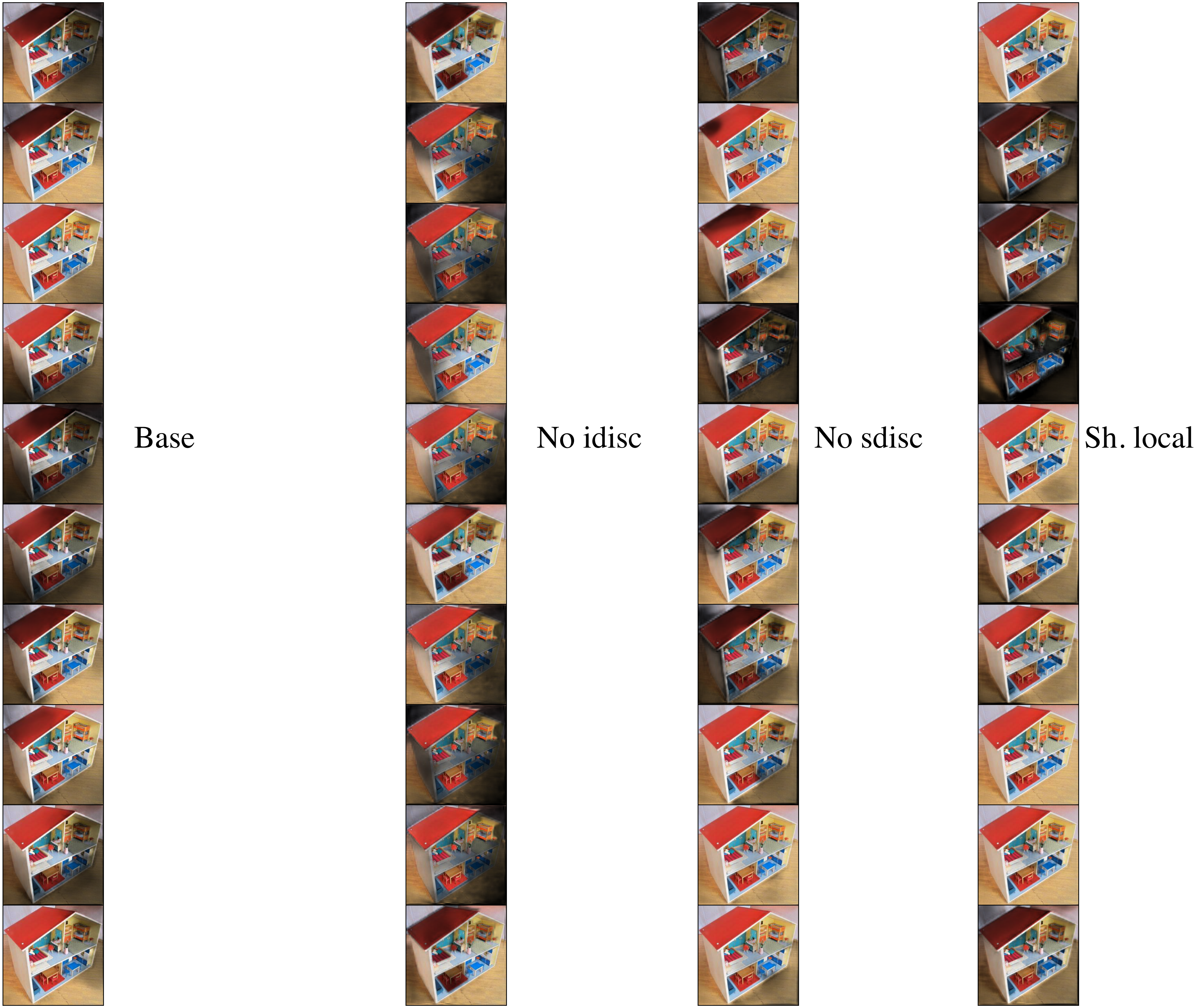
Base

No idisc

No sdisc

Sh. local


Base
No idisc
No sdisc
Sh. local


Base
No idisc
No sdisc
Sh. local

Base

No idisc

No sdisc

Sh. local

Evaluating reshading…

A possible alternative to NERF…

- From NOPC (next)
  - pinch the idea of feature enhanced point clouds
- From point-based DR (after that)
  - make features differentiable
- Cut-and-shut with
  - image norms
  - adversary
- Avoids clunky bits
  - of NeRF - volume rendering
  - of PixelNeRF - no world coord representation

NOPC - neural opacity point clouds

- Aim:
  - recover and represent fuzzy objects

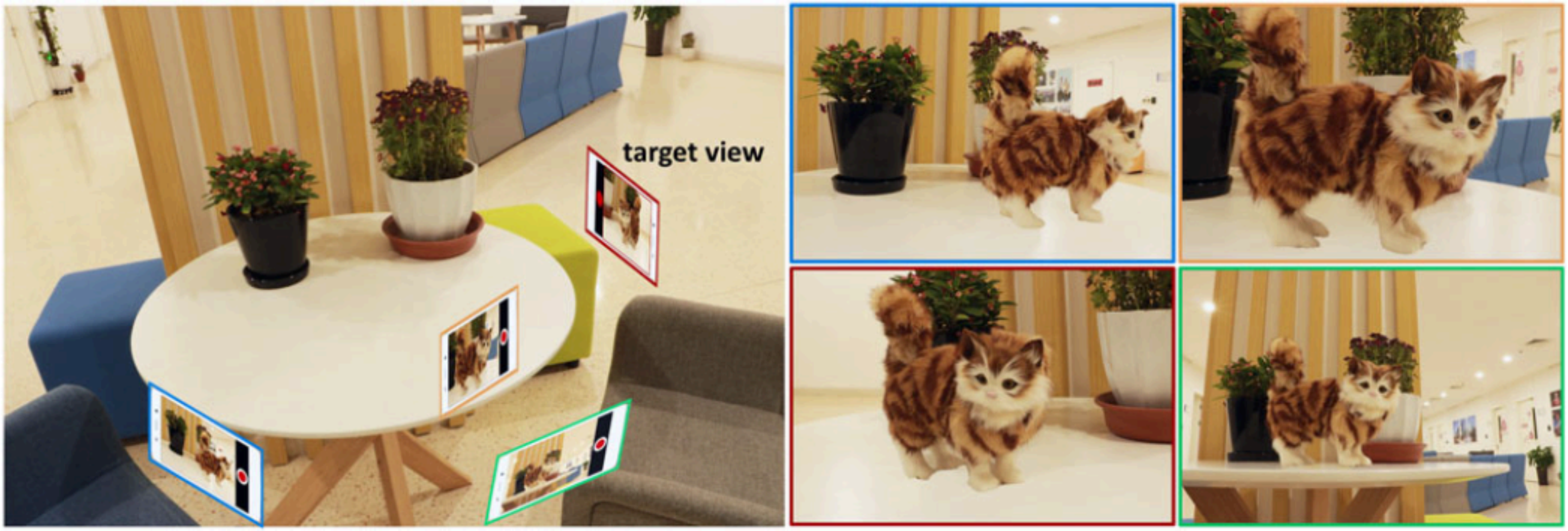


Fig. 1. Our Neural Opacity Point Cloud (NOPC) renderer produces photorealistic, free viewpoint rendering of fuzzy objects from a sparsely sampled images (Fig. 3). By rendering high quality RGB and alpha at an arbitrary viewpoint, NOPC can insert fuzzy virtual objects into real environment.

Wang et al 20

NOPC - representation

Key idea: an entirely conventional point based renderer renders FEATURES attached to points
Then a network adjusts this rendered feature map into an image

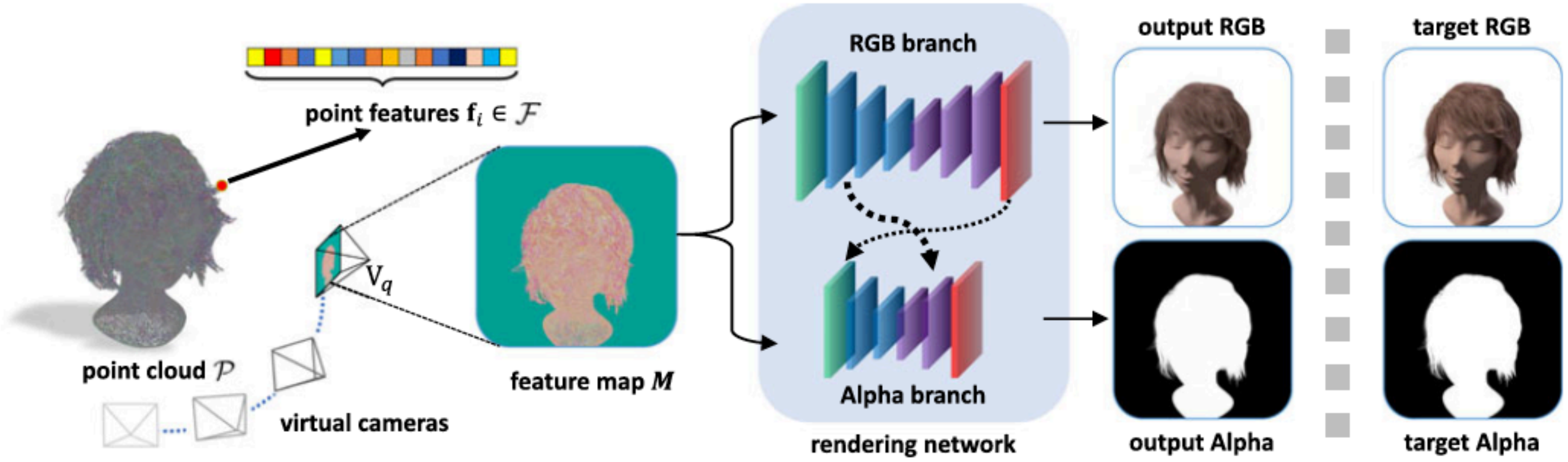
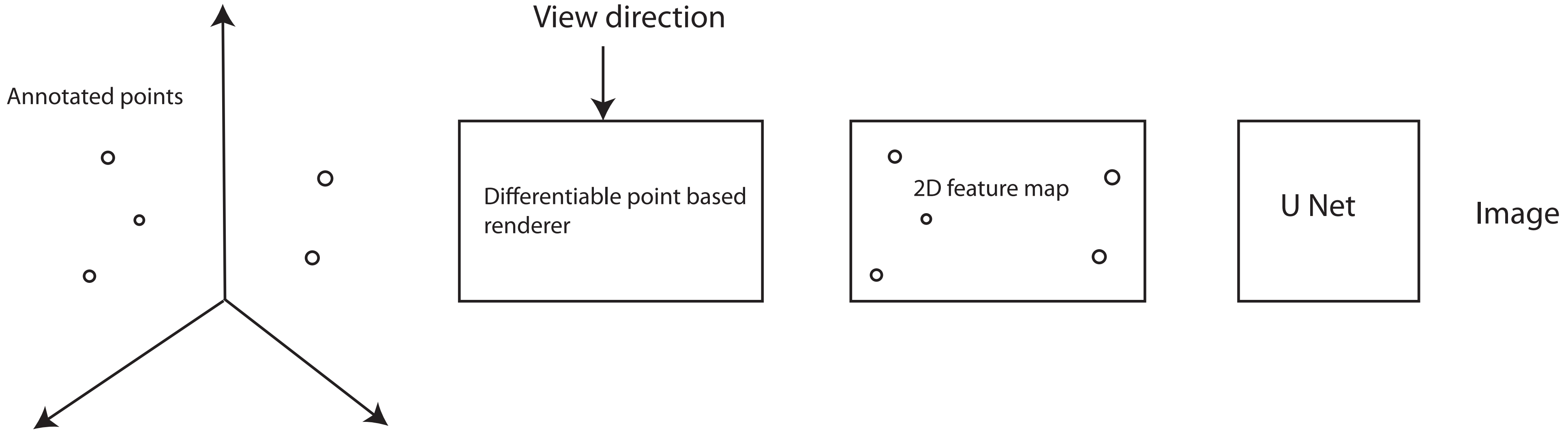


Fig. 2. Our NOPC framework: From a 3D point cloud $\mathcal{P}$, we first learn its corresponding cloud $\mathcal{F}$. To render virtual new view V, we project $\mathcal{P}$ and $\mathcal{F}$ onto V to form a view-dependent feature map M. Our multiple-branch Encoder-Decoder network maps M to an RGB image and an alpha matte at V. The network can be trained using the ground truth RGB images and alpha mattes in an end-to-end manner in Section 3.

Wang et al 20

Idea: attach to differentiable PBR



You could now use losses on images, etc.
to shape features on the point cloud

Differentiable Point Based Rendering

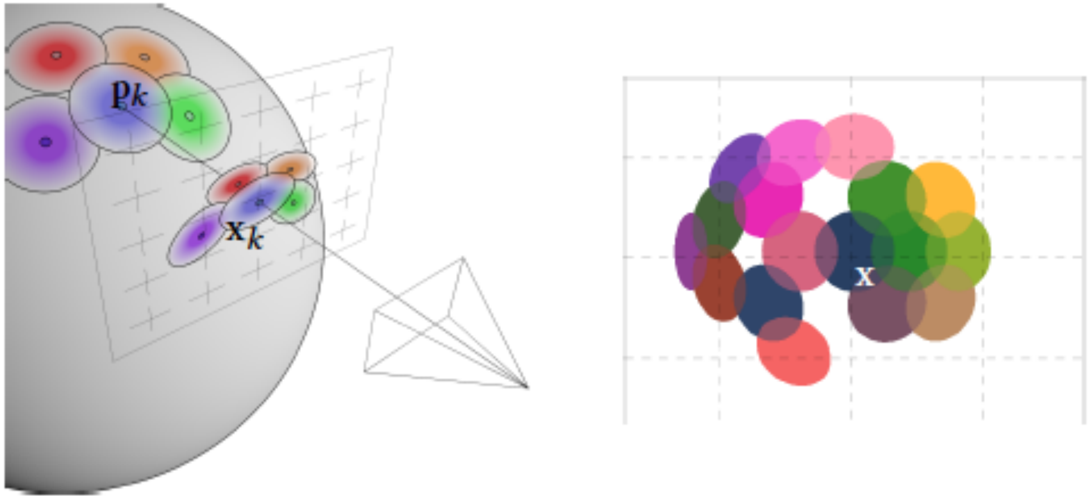


Fig. 2. Illustration of forward splatting using EWA [Zwicker et al. 2001]. A point in space $\mathbf{p}_k$ is rendered as an anisotropic ellipse centered at the projection point $\mathbf{x}_k$. The final pixel value $\mathbb{I}_\mathbf{x}$ at a pixel $\mathbf{x}$ in the image (shown on the right) is the normalized sum of all such ellipses overlapping at $\mathbf{x}$.

- Needs some adaptation
- Surface = point cloud
  - each point is a flat circle centered at point location, with normal
  - project these
    - colored ellipses
      - our case: featured ellipses
    - render=weighted sum of overlapping ellipses

Yifan et al 19

You can clearly render feature maps…

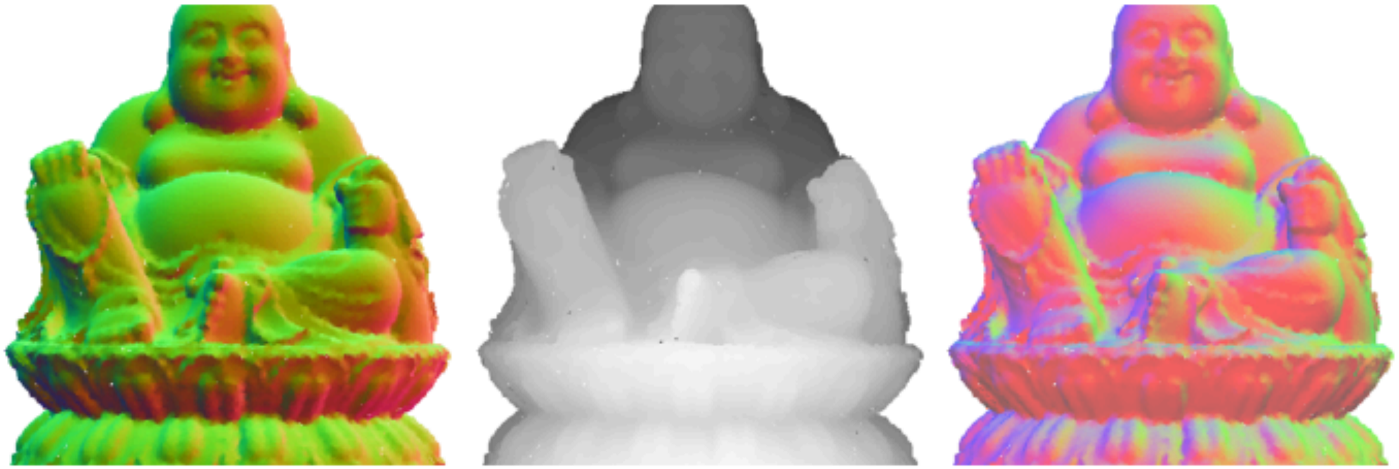


Fig. 3. Examples of images rendered using DSS. From left to right, we render the normals, inverse depth values and diffuse shading with three RGB-colored sun light sources.

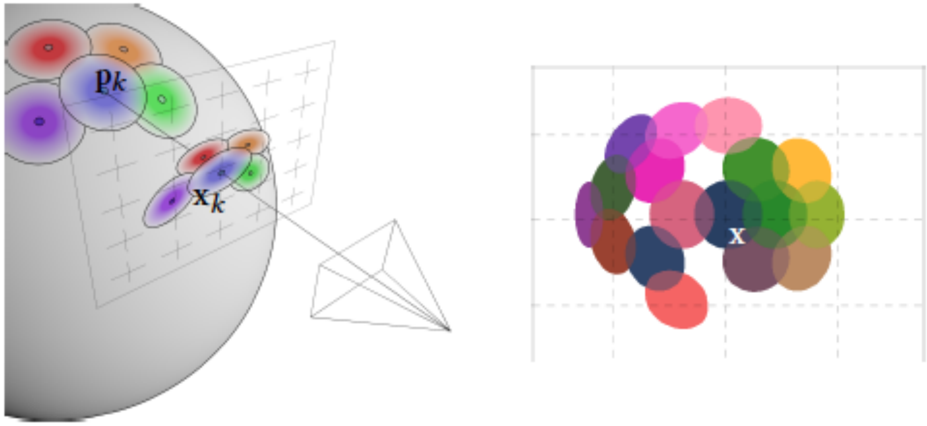
Yifan et al 19

Render

- Circles on tangent plane around p

$$\mathcal{G}_{\mathbf{p}_k,\mathbf{V}_k}(\mathbf{p}) = \frac{1}{2\pi|\mathbf{V}_k|^{\frac{1}{2}}} e^{(\mathbf{p}-\mathbf{p}_k)^\top \mathbf{V}_k^{-1}(\mathbf{p}-\mathbf{p}_k)}, \quad \mathbf{V}_k = \sigma_k^2 \mathbf{I},$$

- Project to ellipse in image, low pass filter



$$\bar{\rho}_k(\mathbf{x}) = \frac{1}{\left|\mathbf{J}_k^{-1}\right|} \mathcal{G}_{\mathbf{J}_k \mathbf{V}_k \mathbf{J}_k^\top + \mathbf{I}}(\mathbf{x} - \mathbf{x}_k) \, .$$

From low pass filter

Jacobian of projection to image

Fig. 2. Illustration of forward splatting using EWA [Zwicker et al. 2001]. A point in space $\mathbf{p}_k$ is rendered as an anisotropic ellipse centered at the projection point $\mathbf{x}_k$. The final pixel value $\mathbb{I}_\mathbf{x}$ at a pixel $\mathbf{x}$ in the image (shown on the right) is the normalized sum of all such ellipses overlapping at $\mathbf{x}$.

Render - II

- Occlusion
  - keep the 5 points closest to pixel

- Weighting

$$\rho_k(\mathbf{x}) = \begin{cases} 0, & \text{if } \frac{1}{2}\mathbf{x}^\top \left(\mathbf{J}\mathbf{V}_k\mathbf{J}^\top + \mathbf{I}\right)\mathbf{x} > C, \\ 0, & \text{if } \mathbf{p}_k \text{ is occluded}, \\ \bar{\rho}_k, & \text{otherwise.} \end{cases}$$

- Final render

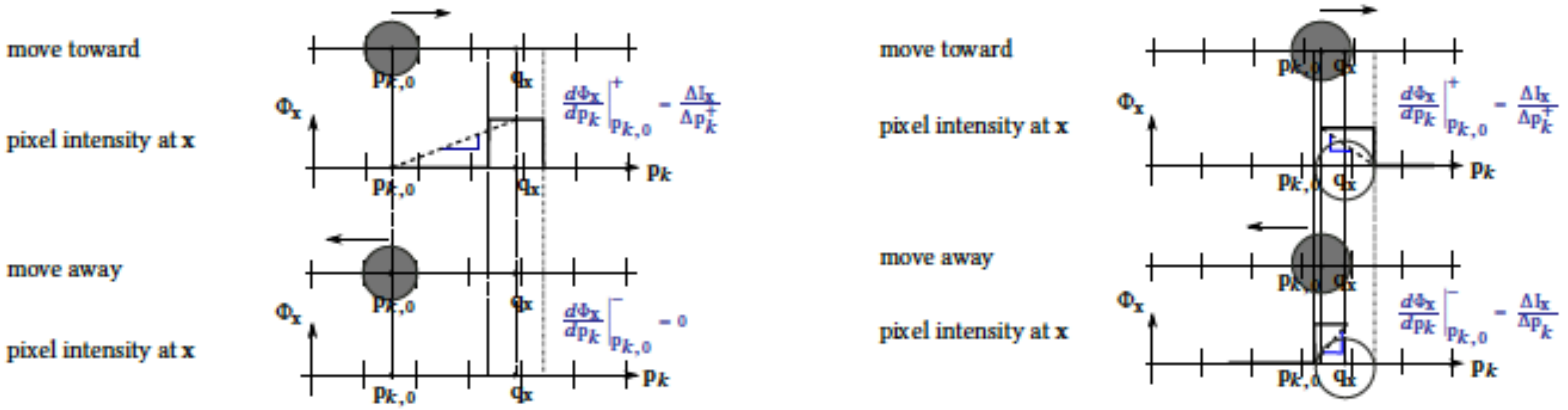
This is whatever is living in point (color; feature vector; etc)

$$\mathbb{I}_{\mathbf{x}} = \frac{\sum_{k=0}^{N-1} \rho_k(\mathbf{x})\ \mathbf{w}_k}{\sum_{k=0}^{N-1} \rho_k(\mathbf{x})}.$$

Differentiating

- Two hard bits:
  - occlusion
  - weight truncation
- “Smooth” by
  - averaging left and right differences
  - odd, but seems to work

Differentiating - 1D example

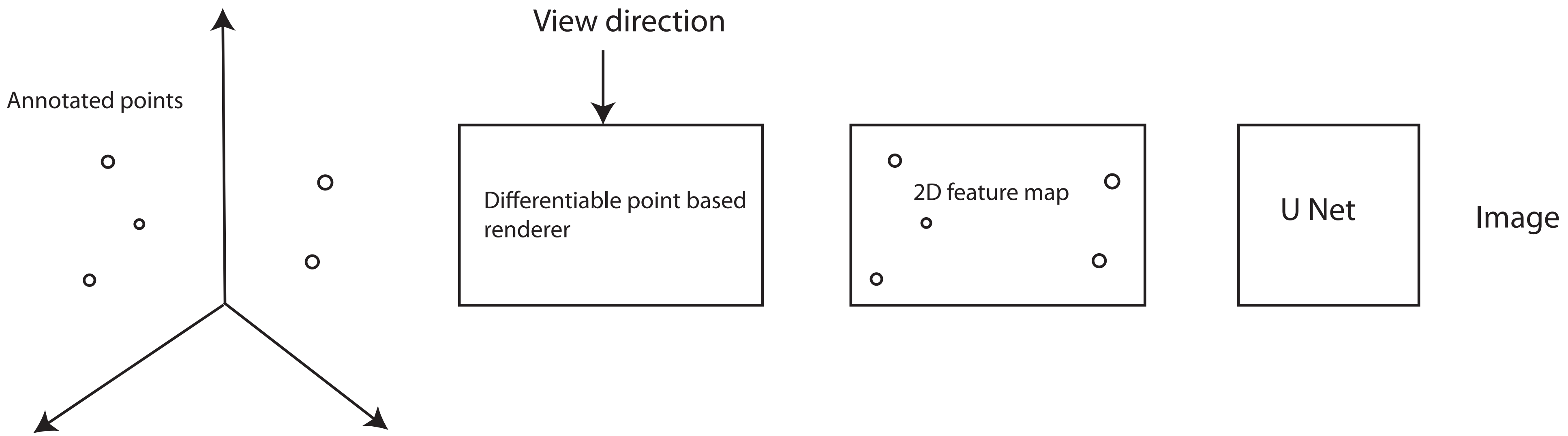

move toward
pixel intensity at x
Φ_x
$p_{k,0}$
q_x
p_k
$\frac{d\Phi_x}{dp_k}\Big|^{+}_{p_{k,0}} = \frac{\Delta I_x}{\Delta p_k^+}$
move away
pixel intensity at x
$\frac{d\Phi_x}{dp_k}\Big|^{-}_{p_{k,0}} = 0$
move toward
pixel intensity at x
$\frac{d\Phi_x}{dp_k}\Big|^{+}_{p_{k,0}} = \frac{\Delta I_x}{\Delta p_k^+}$
move away
pixel intensity at x
$\frac{d\Phi_x}{dp_k}\Big|^{-}_{p_{k,0}} = \frac{\Delta I_x}{\Delta p_k^-}$

(a) The ellipse centered at $\mathbf{p}_{k,0}$ is not visible at $\mathbf{x}$.

(b) The ellipse centered at $\mathbf{p}_{k,0}$ is visible at $\mathbf{x}$.

Fig. 4. An illustration of the artificial gradient in two 1D scenarios: the ellipse centered at $\mathbf{p}_{k,0}$ is invisible (Fig. 4a) and visible (Fig. 4b) at pixel $\mathbf{x}$. $\Phi_{\mathbf{x},k}$ is the pixel intensity $\mathbf{I}_\mathbf{x}$ as a function of point position $\mathbf{p}_k$, $\mathbf{q}_\mathbf{x}$ is the coordinates of the pixel $\mathbf{x}$ back-projected to world coordinates. Notice the ellipse has constant pixel intensity after normalization (Eq. (6)). We approximate the discontinuous $\Phi_{\mathbf{x},k}$ as a linear function defined by the change of pixel intensity $\Delta\mathbf{I}_\mathbf{x}$ and the movement of the $\Delta\mathbf{p}_k$ during a visibility switch. As $\mathbf{p}_k$ moves toward ($\Delta\mathbf{p}_k^+$) or away ($\Delta\mathbf{p}_k^-$) from the pixel, we obtain two different gradient values. We define the final gradient as their sum.

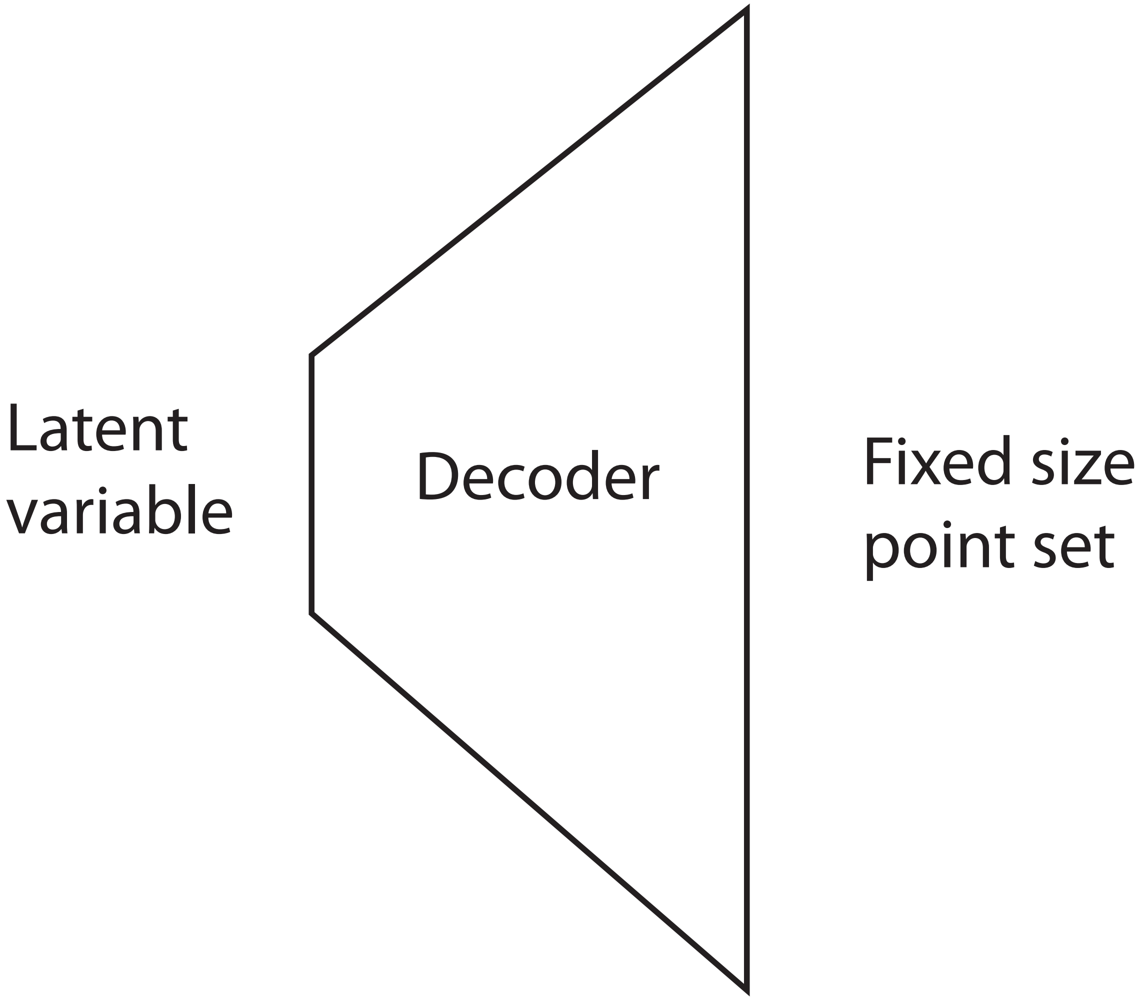
Idea: attach to differentiable PBR



You could now use losses on images, etc.
to shape features on the point cloud

But where does the point cloud come from?

What about …



Latent variable

Decoder

Fixed size point set

Advantages

- No volume rendering
- Could use an adversarial loss
  - perhaps locally?
- Perhaps could build an “objectGAN”?
- Perhaps could build a “conditional object GAN”?
  - which is roughly what this is