Weather and generation: some hard open problems

D.A. Forsyth, UIUC

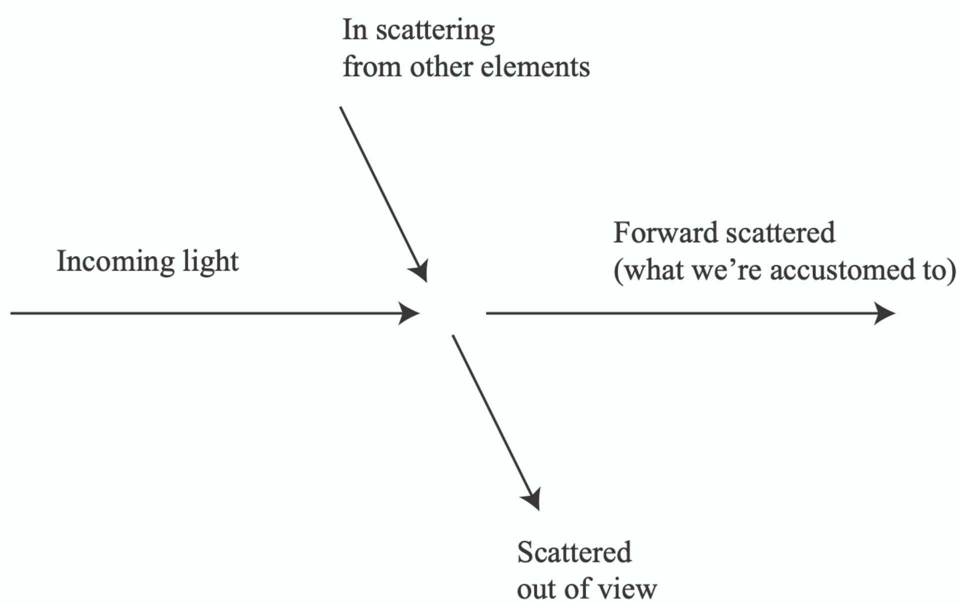
Theme

- There’s an awful lot to do…
- Two interesting, poorly understood problems
  - Weather is bad for vision
  - We are very good at making images, but don’t understand what we’re doing

Weather

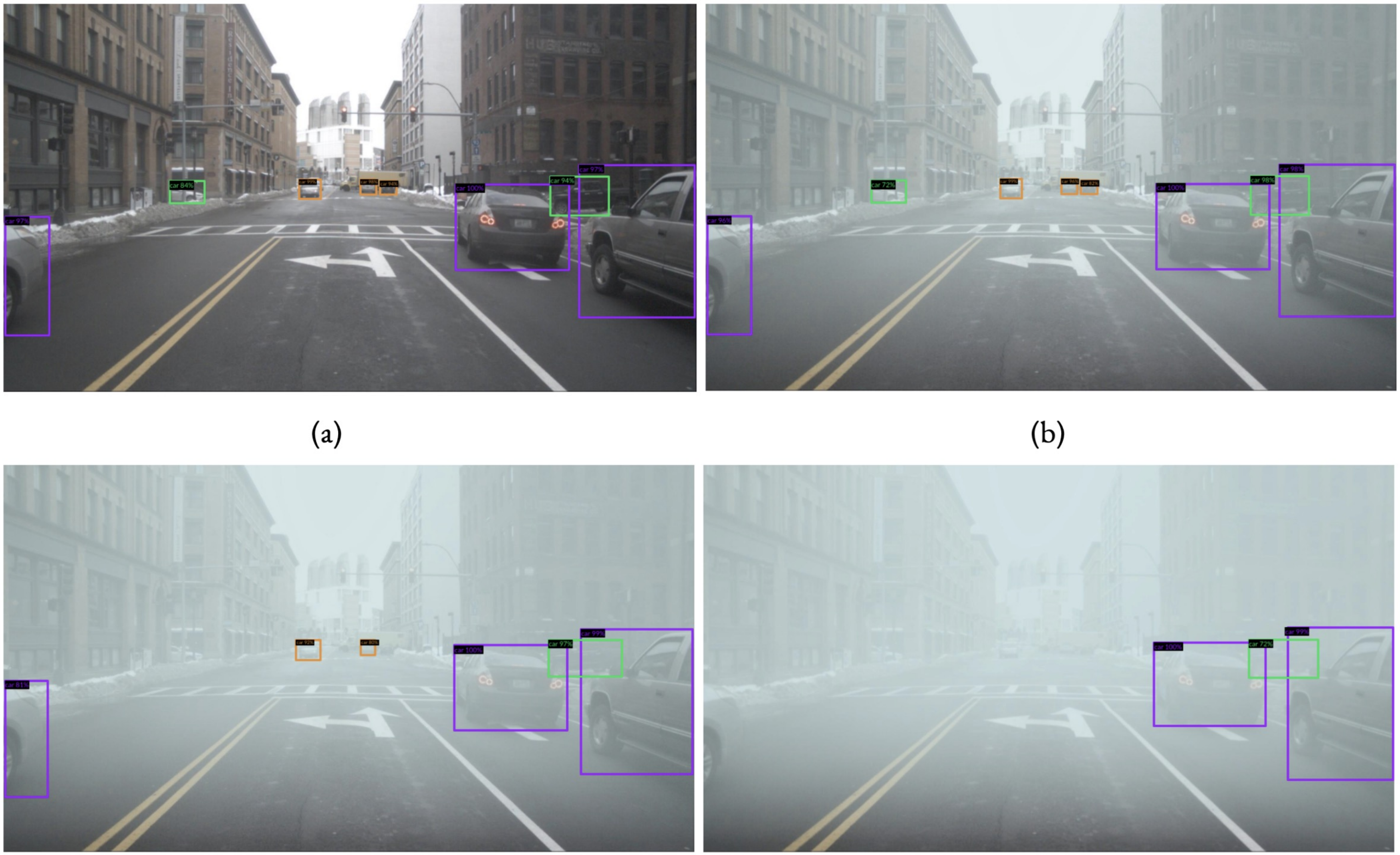
- Weather mangles the performance of all our methods
  - detectors, classifiers, interest point finders, stereo, etc. etc.
  - Fog reduces contrast, blurs images and changes colors
  - Rain is a bit like fog, but adds streaks, puddles, and more
- Computer vision procedures are being used for autonomous vehicles
  - And we don’t want them hurting people cause the weather is bad
- Current “solutions” are quite unconvincing

A ray passing through scattering material



From Lynch and Livingstone, Color and Light in Nature

From Lynch and Livingstone, Color and Light in Nature



(a) (b)

(c) (d)

Kore thesis, 2022

This sort of thing affects detectors, etc.

- What to do:
  - Train detectors on real weather images
    - hard - collect and mark them up; rich collection of effects
    - mostly, this won't work out
  - Remove weather effects, then apply detector
    - Q: Remove how?
      - Simple physics
      - Regression (next)
  - Take training images, synthesize weather on top
    - Q: How?
      - complicated mixture of physics and advanced regression tricks

Paired data

- Collect data on good days, bad days
  - along the same routes, w/ GPS
  - use dynamic programming, GPS to compute alignment at the image level
- Now label
  - annotator labels bad image round 1
  - compares to good image; fixes labelling round 2

Sakaridis et al, 21

(a) Input image I (b) Stage 1 annotation (draft) (c) Corresponding image I' (d) Stage 2 annotation (GT) (e) Invalid mask J

Figure 2. **Illustration of annotation protocol for ACDC.** The color coding of the semantic classes matches Fig. 1. All annotations in (b), (d) and (e) pertain to the input image I in (a). A white color in (b) and (d) denotes unlabeled pixels.

This sort of thing affects detectors, etc.

- Fairly clear (more later)
- What to do:
  - Train detectors on real weather images
    - hard - collect and mark them up; rich collection of effects
    - mostly, this won't work out
  - Remove weather effects, then apply detector
    - Q: Remove how?
      - Simple physics
      - Regression (next)
  - Take training images, synthesize weather on top
    - Q: How?
      - complicated mixture of physics and advanced regression tricks

Removing haze by physical reasoning

Airlight color at p

$$I(p) = J(p) \times T(p) + A(p) \times (1 - T(p))$$

Image color at p

Surface radiance color at p

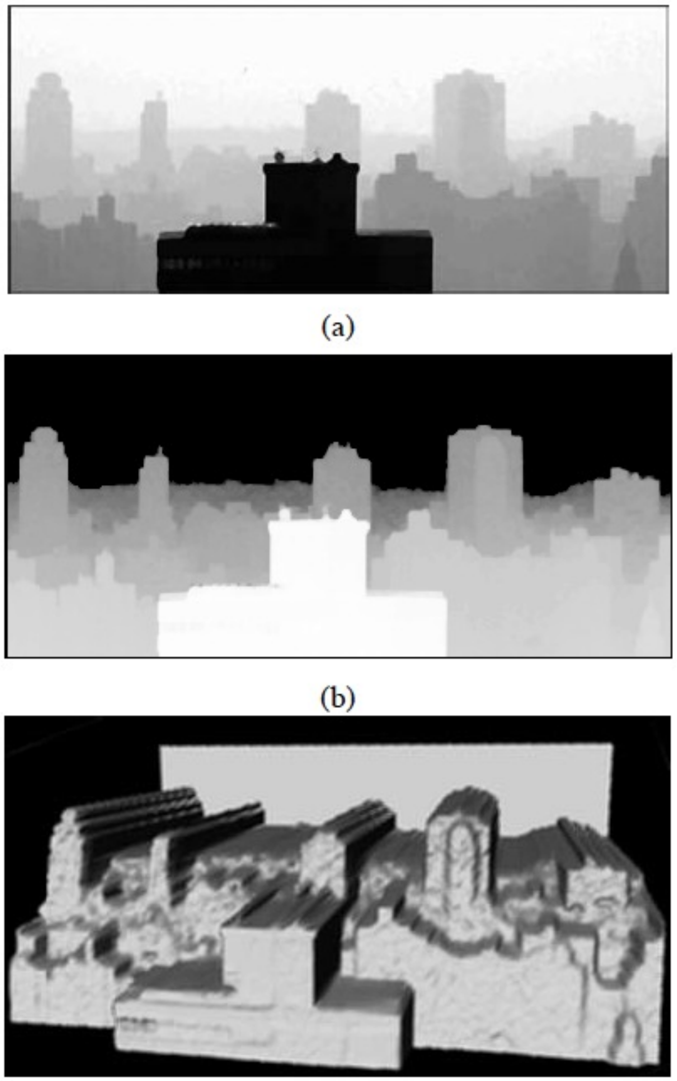
Absorption term, exponential in depth, at p

- Consequences
  - Brightness is a depth cue
  - Reasoning about airlight color yields dehazed image

Airlight yields a depth cue

$$I(p) = \cancel{J(p) \times T(p)} + A(p) \times (1 - T(p))$$

- Assume that airlight is dominant
  - (i.e. most of light arriving at camera is airlight)
  - then you can recover depth from a single image
- Disadvantages
  - requires significant fog (but not too much) or large scales



(a) (b) (c)

Nayar and Narasimhan, 1999

Model

Airlight color - same at all points

$$I(p) = J(p) \times T(p) + A(p) \times (1 - T(p))$$

Observed

Shading x albedo

Independent of shading

- With work, this yields
  - neighboring pixels with same albedo yield
    - constraints on shading and T
  - assume shading and T independent
    - estimate A to yield “most independent” shading and T
  - result: J(p)

Figure 1: Dehazing based on a single input image and the corresponding depth estimate.

Fattal, 08 - note depth map AND dehaze; note also slightly odd colors

Improved estimation by cleaner model

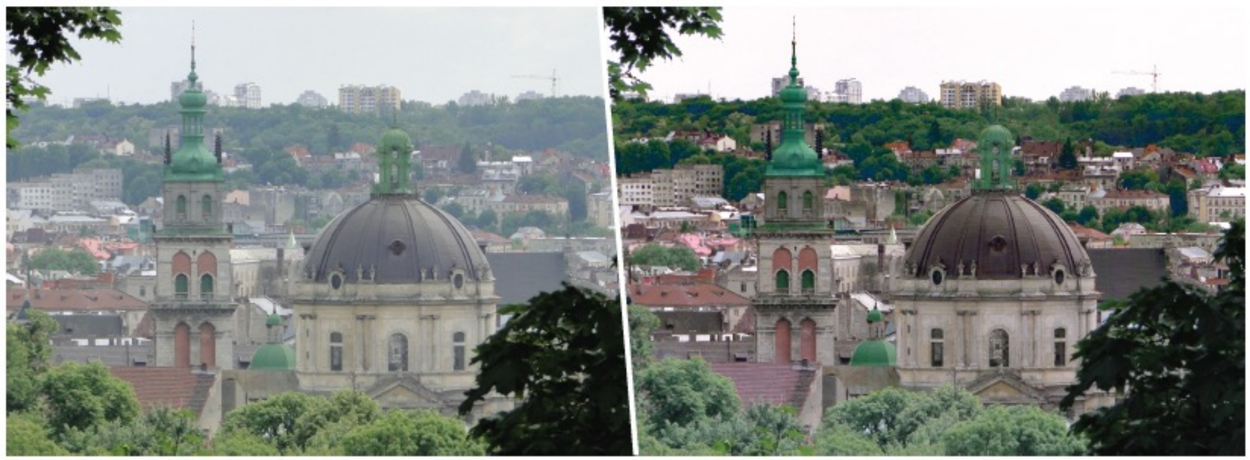


Fig. 1. Old Town of Lviv. Input image on the left, our result on the right.

Fattal, 08 - note depth map AND dehaze; note also slightly odd colors

This sort of thing affects detectors, etc.

- Fairly clear (more later)
- What to do:
  - Train detectors on real weather images
    - hard - collect and mark them up; rich collection of effects
    - mostly, this won't work out
  - Remove weather effects, then apply detector
    - Q: Remove how?
      - Simple physics
      - Regression
  - Take training images, synthesize weather on top
    - Q: How?
      - complicated mixture of physics and advanced regression tricks

Image regression

- Take an image, predict something “like” an image
  - Underlying technology is straightforward, significant tricks
- Cases
  - train with real paired data eg (image, foggy version of image)
  - train with fake paired data eg (image, simulated foggy version of image)
  - train with unpaired data; important, we’ll ignore
- Motivating problems
  - image -> depth
    - also, image pair -> optic flow; low res image-> high res image
  - image -> foggy image; image -> rainy image
- Mechanics sketched earlier

Paired datasets

- Obtain pairs (hazy image, clear image)
- Real data:
  - Take photos outdoors; introduce fog; repeat
    - NH-HAZE
      - https://data.vision.ee.ethz.ch/cvl/ntire20/nh-haze/
- Synthesized data:
  - Fake fog model on real image
    - Foggy cityscapes
      - https://people.ee.ethz.ch/~csakarid/SFSU_synthetic/
  - Render synthetic images fog/no-fog
    - RESIDE
      - https://arxiv.org/pdf/1712.04143.pdf

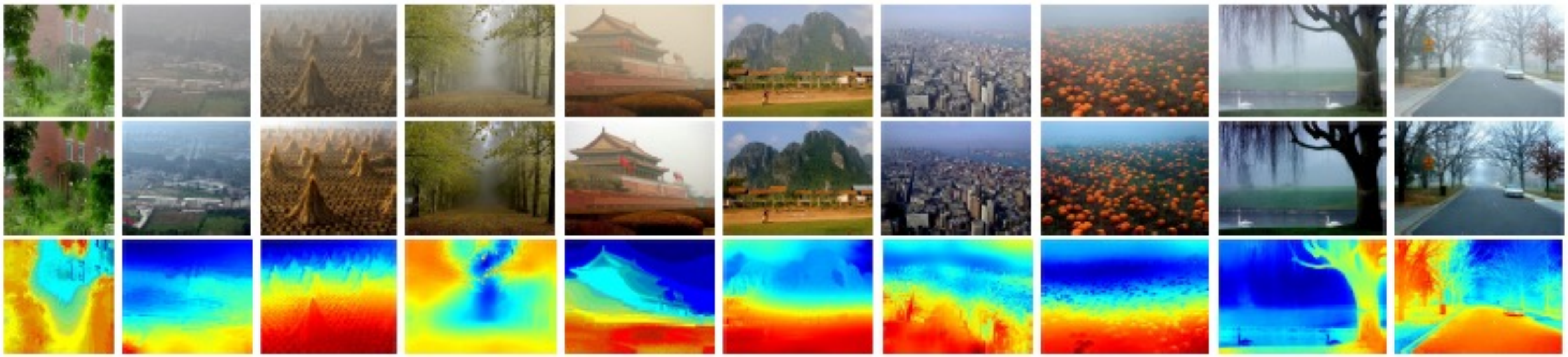


Fig. 11. The haze-free images and depth maps restored by DehazeNet

Cai et al 16 (DeHazeNet)

Single image dehazing

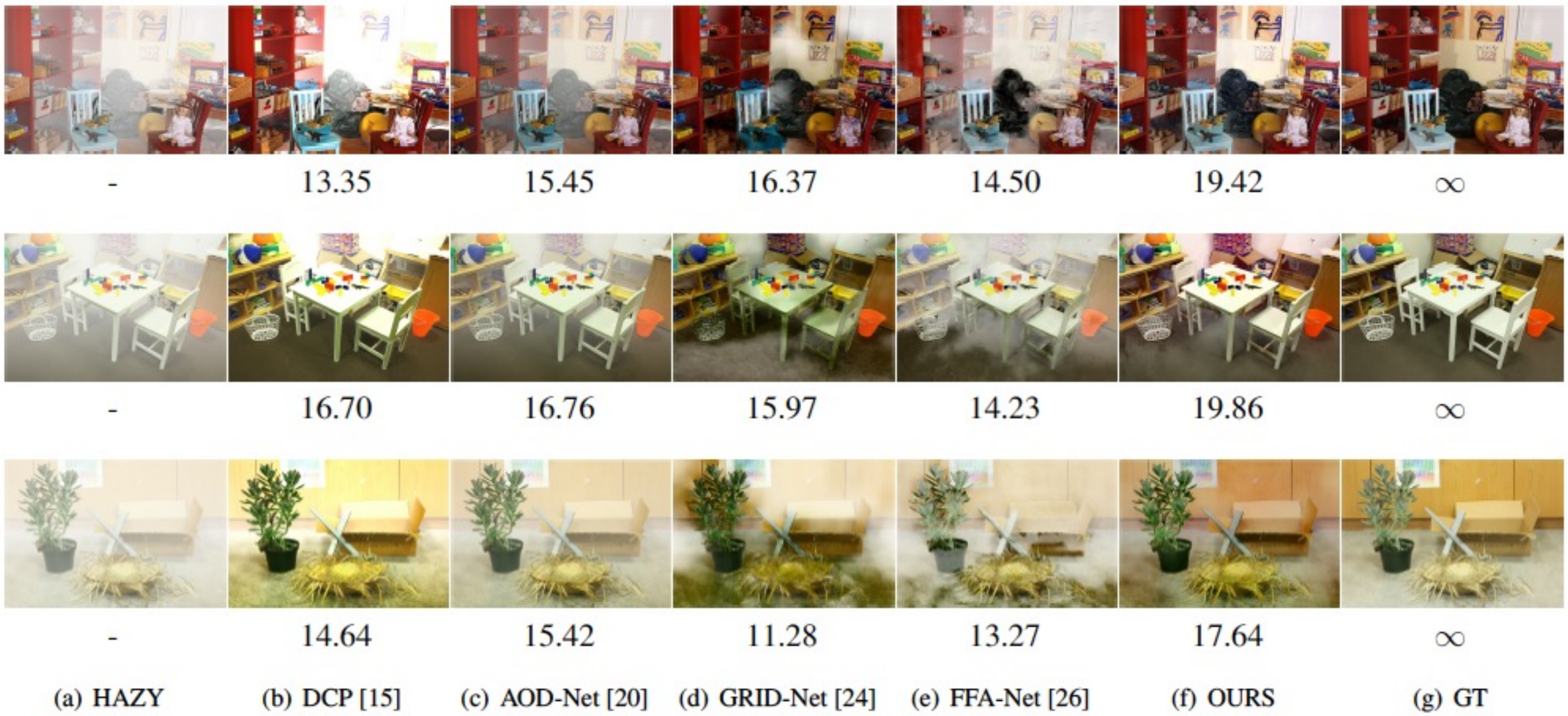
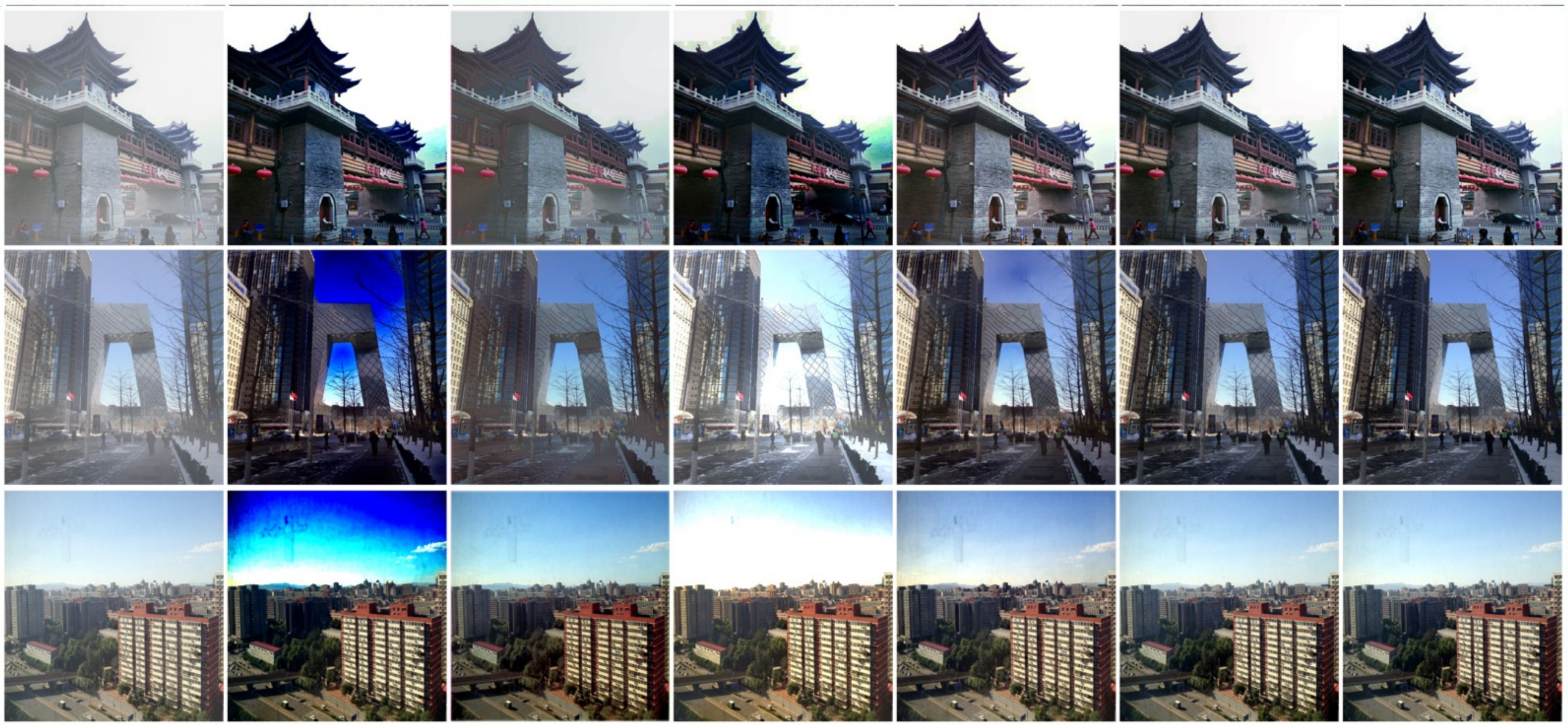


Figure 6. Qualitative comparisons with different state-of-the-art dehazing methods for indoor synthesis hazy images. The top two rows are from SOTS, the third row is from TestA dataset and the bottom three rows are from MiddleBury dehazing dataset. The numbers below image are PSNR (dB) value of each image.

Shen et al 19



(a)Hazy inputs (b)DCP (c)AOD-Net (d)DehazeNet (e)GCANet (f)Ours (g) GT

Qin et al 19 - Use feature attention

Asides

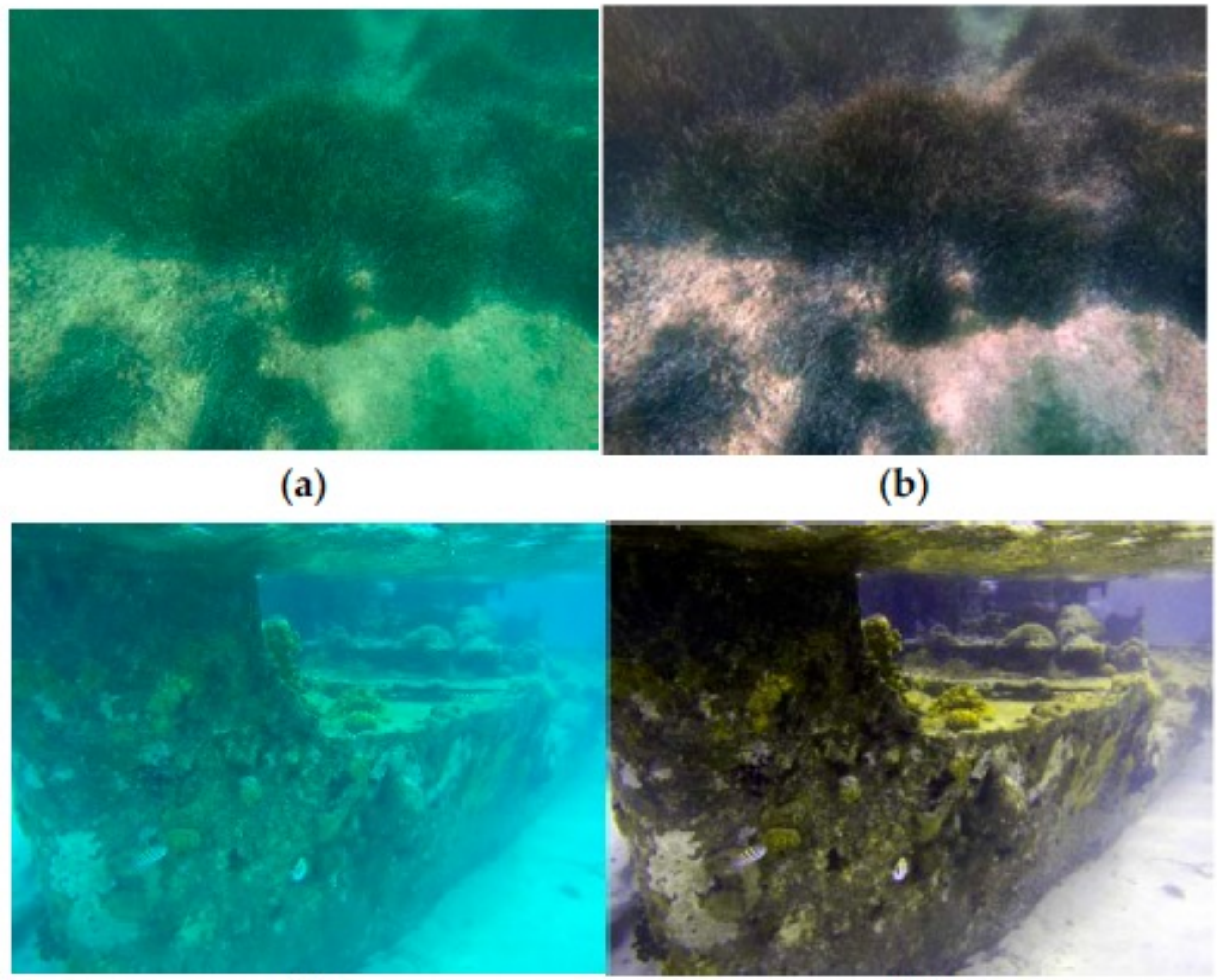
- Defoggers trained with simulated fog work well
  - Even if the depth map used to simulate the fog is wrong

Similar physics underwater

- Out scattering causes distant points to be darker and fuzzier
- Out and in scattering changes color
- Color changes depend on the water

(a)

(c)



(a) (b)

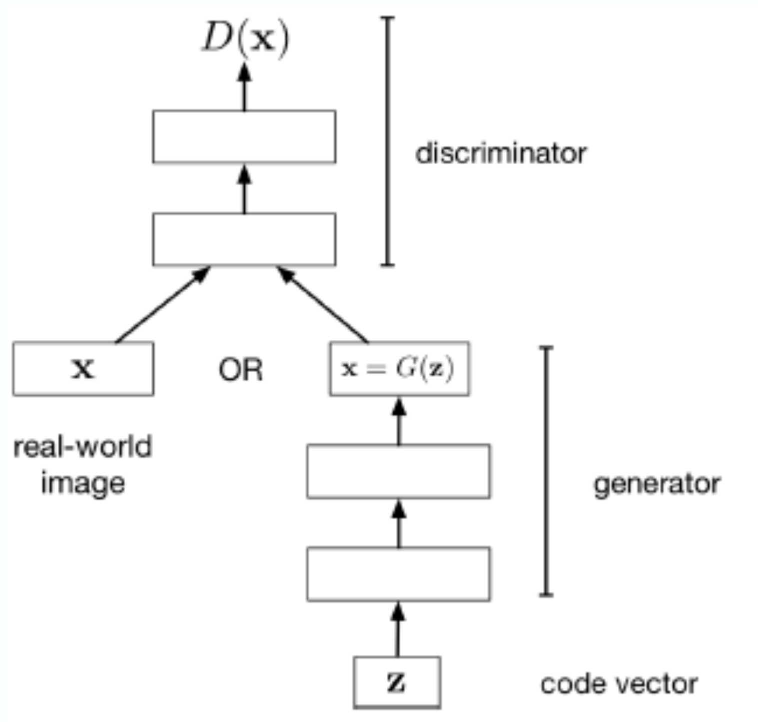
(c) (d)

Side topic - Adversarial losses

- Issue:
  - we are making pictures that should have a strong structure
    - eg - it should be “like” a true image
    - but we don’t know how to write a loss that imposes that structure
- Strategy:
  - build a classifier that tries to tell the difference between
    - true examples
    - examples we made
  - use that classifier as a loss

A GAN

Generative
Adversarial
Network



Grosse slides

- Let D denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

Notice: we want the discriminator to make a 1 for real data, 0 for fake data

- One possible cost function for the generator: the opposite of the discriminator's

$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\max_G \min_D \mathcal{J}_D$$

Solution (if exists, which is uncertain; and if can be found, ditto) is known as a saddle point. It has strong properties, but not much worth talking about, as we don't know if it is there or whether we have found it.

Grosse slides

Quote from the original paper on GANs:

> *"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles."*

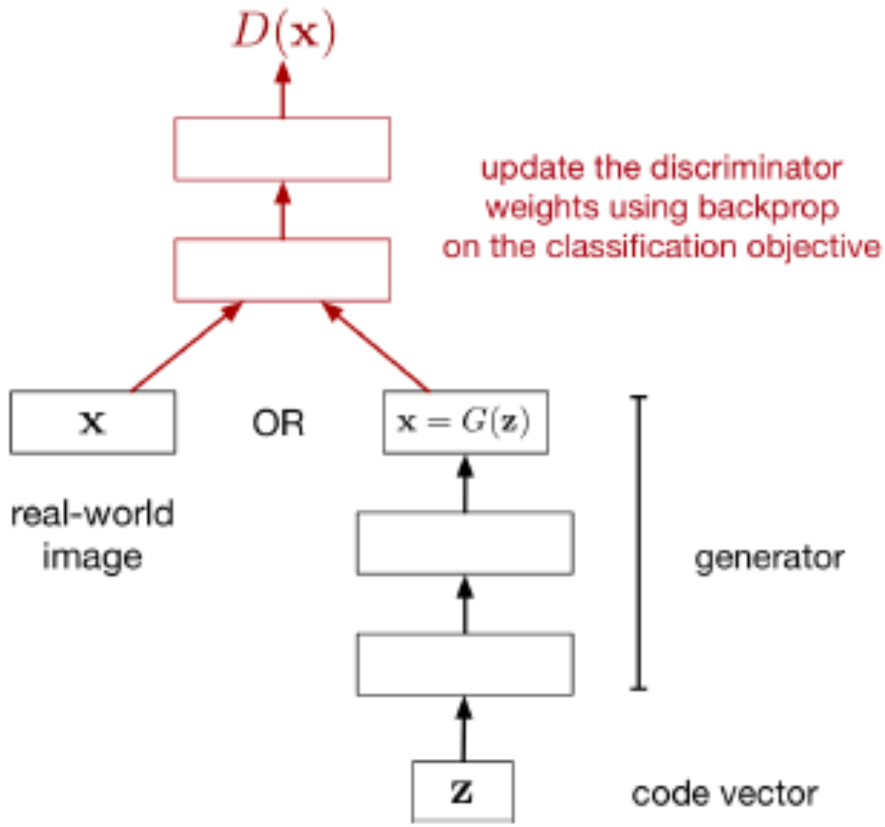
-Goodfellow et. al., "Generative Adversarial Networks" (2014)

Thakar slides

Important, general issue

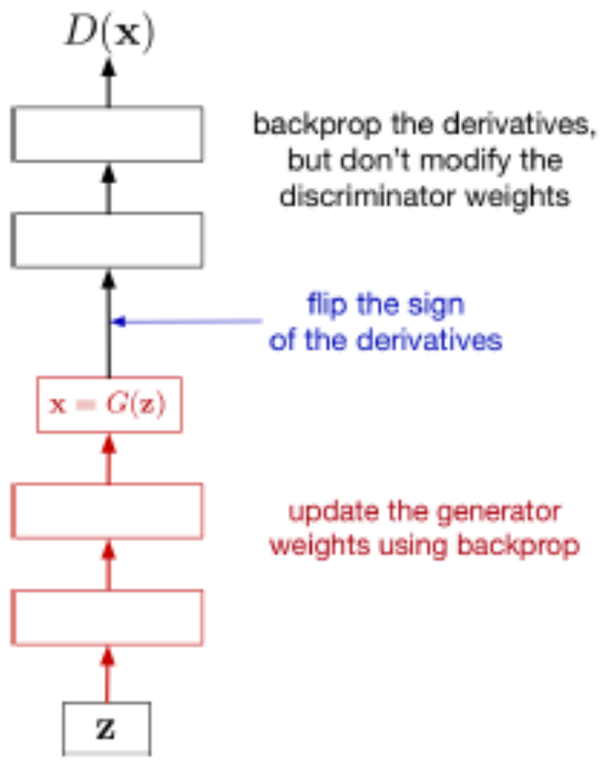
- If either generator or discriminator “wins” -> problem
- Discriminator “wins”
  - it may not be able to tell the generator how to fix examples
  - discriminators classify, rather than supply gradient
- Generator “wins”
  - likely the discriminator is too stupid to be useful
- Very little theory to guide on this point

Updating the discriminator:



Grosse slides

Updating the generator:



Grosse slides

One must be careful about losses…

- We introduced the minimax cost function for the generator:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem with this is saturation.
- Recall from our lecture on classification: when the prediction is really wrong,
  - "Logistic + squared error" gets a weak gradient signal
  - "Logistic + cross-entropy" gets a strong gradient signal
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat.

Grosse slides

One must be careful about losses…

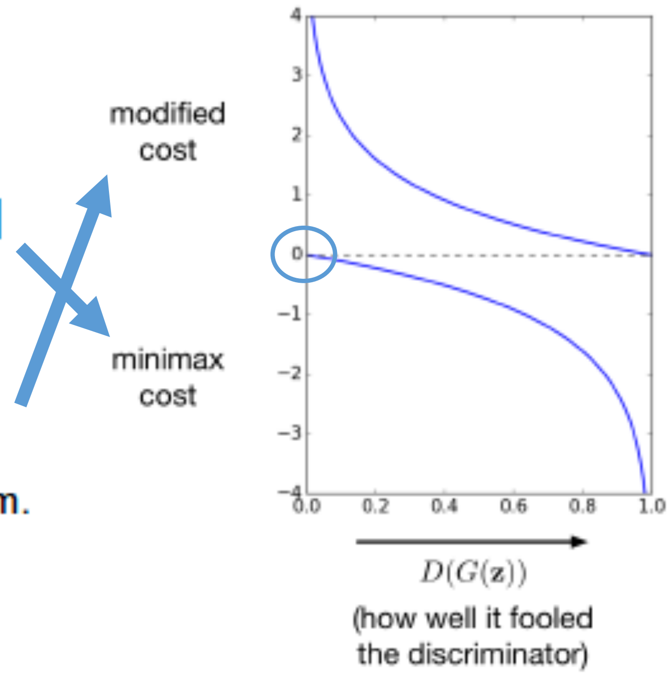
- Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

- This fixes the saturation problem.



Grosse slides

Alternative losses

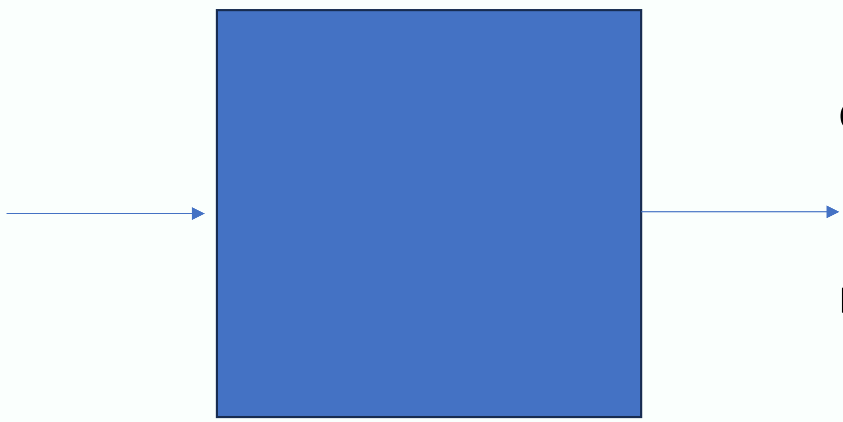
- Hinge:
  - Discriminator makes D(im)
    - want
      - real images -> -1
      - fake -> 1
  - Discriminator loss:

$$\sum_{\text{fakes and real}} \max(0, 1 - y_i D(I_i))$$

    - where y_i=-1 for real, y_i=1 for fake
  - Generator loss:

$$\sum_{\text{fakes}} D(I_i)$$

Simulated foggy image



Output

Check: is it like the original (non foggy) image?

Check: is it like an image?

Notice these checks are NOT the same

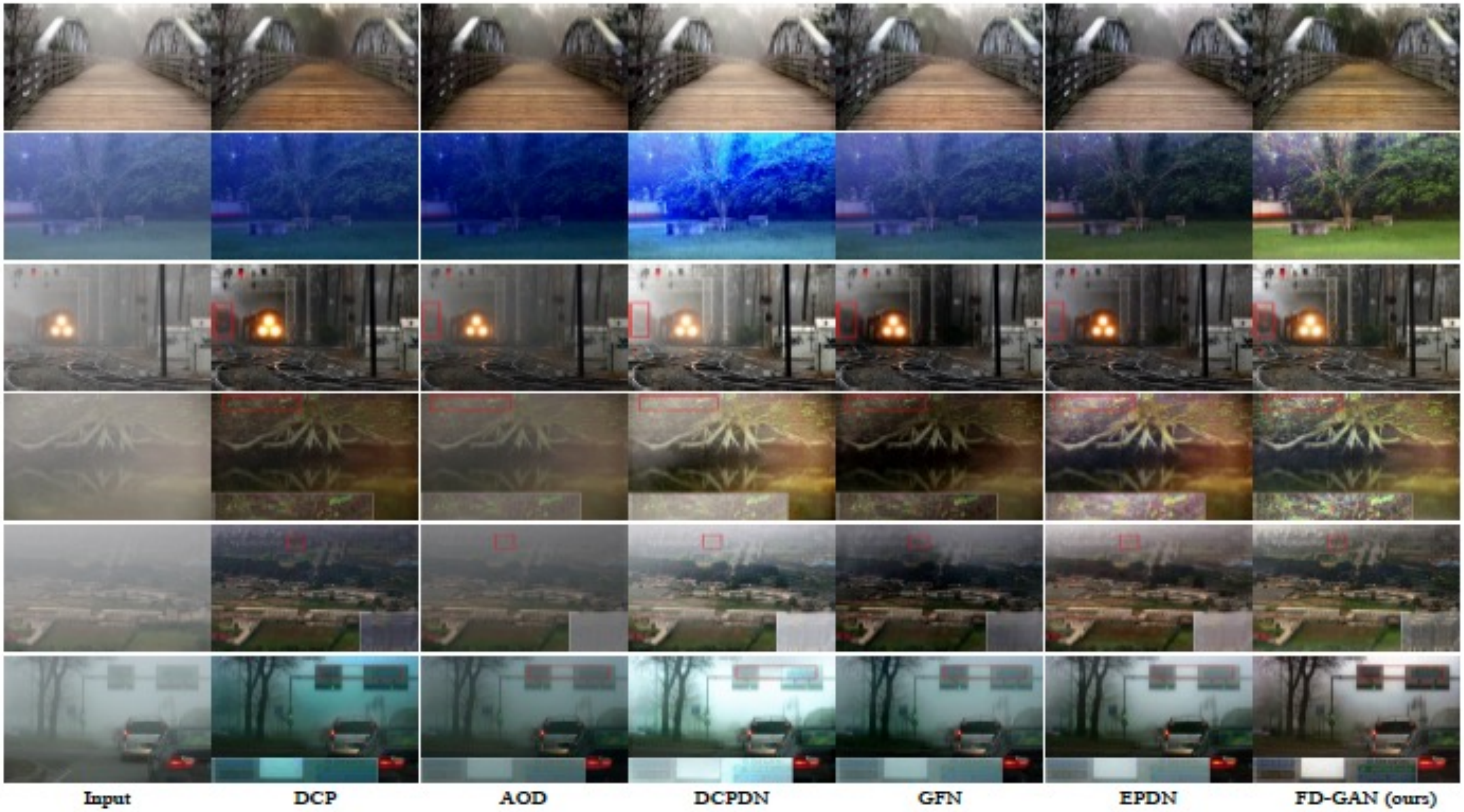
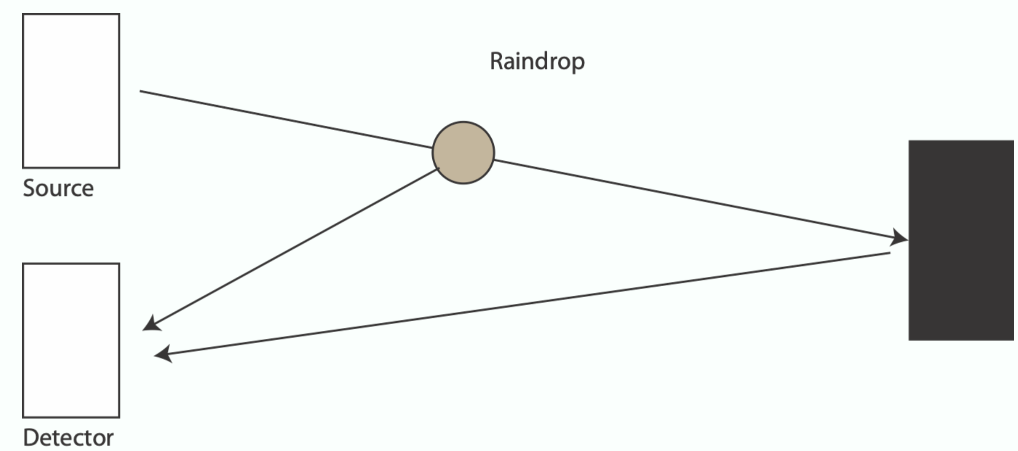


Figure 5: Visual comparisons on real-world hazy images. Our model can generate more natural and visual pleasing dehazed results with less color distortion. Please see the details in red rectangles. Zoom in for best view.

Dong et al 21 - Use an adversarial loss

Raindrop scatter



Backscatter

- Refraction in drops causes backscatter of headlight light
  - makes driving in rain at night harder
- Neat trick
  - (Tamburo et al 14)
  - Do not illuminate raindrops by
    - having headlights that are highly steerable (multiple micro mirrors)
    - very fast exposure with usual illumination identifies raindrops
      - too fast for driver to resolve
    - now direct light between drops

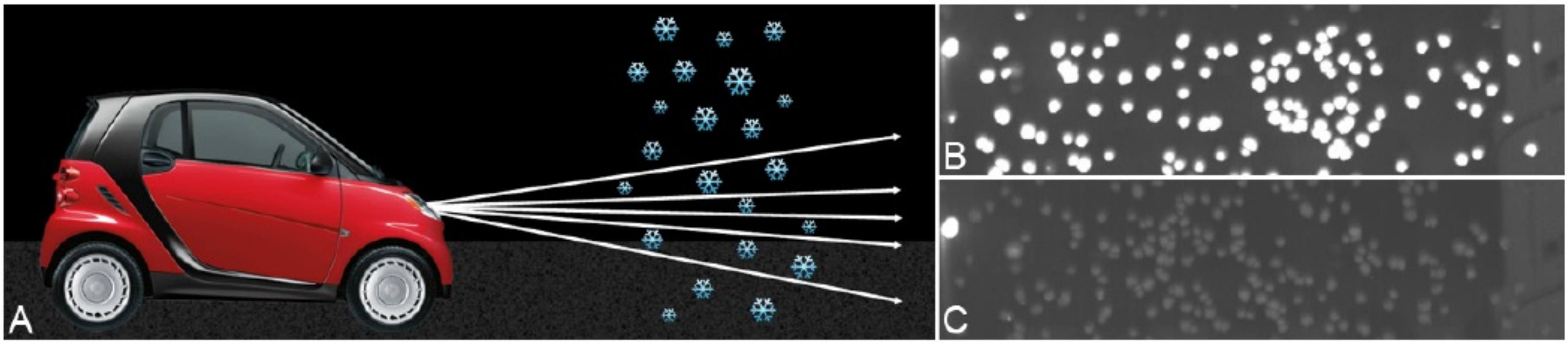


Fig. 7. A: Our headlight has unprecedented resolution over space and time so that beams of light may be sent in between the falling snow. Illustration adapted from [11]. B: Artificial snowflakes brightly illuminated by standard headlight. C: Our system avoids illuminating snowflakes making them much less visible.

Tamburo et al 14

Rain has multiple interesting effects

Blur from wet air

Puddles

Color shifts

Streaks

These are often quite strongly coupled to scene geometry

Rain mangles detection

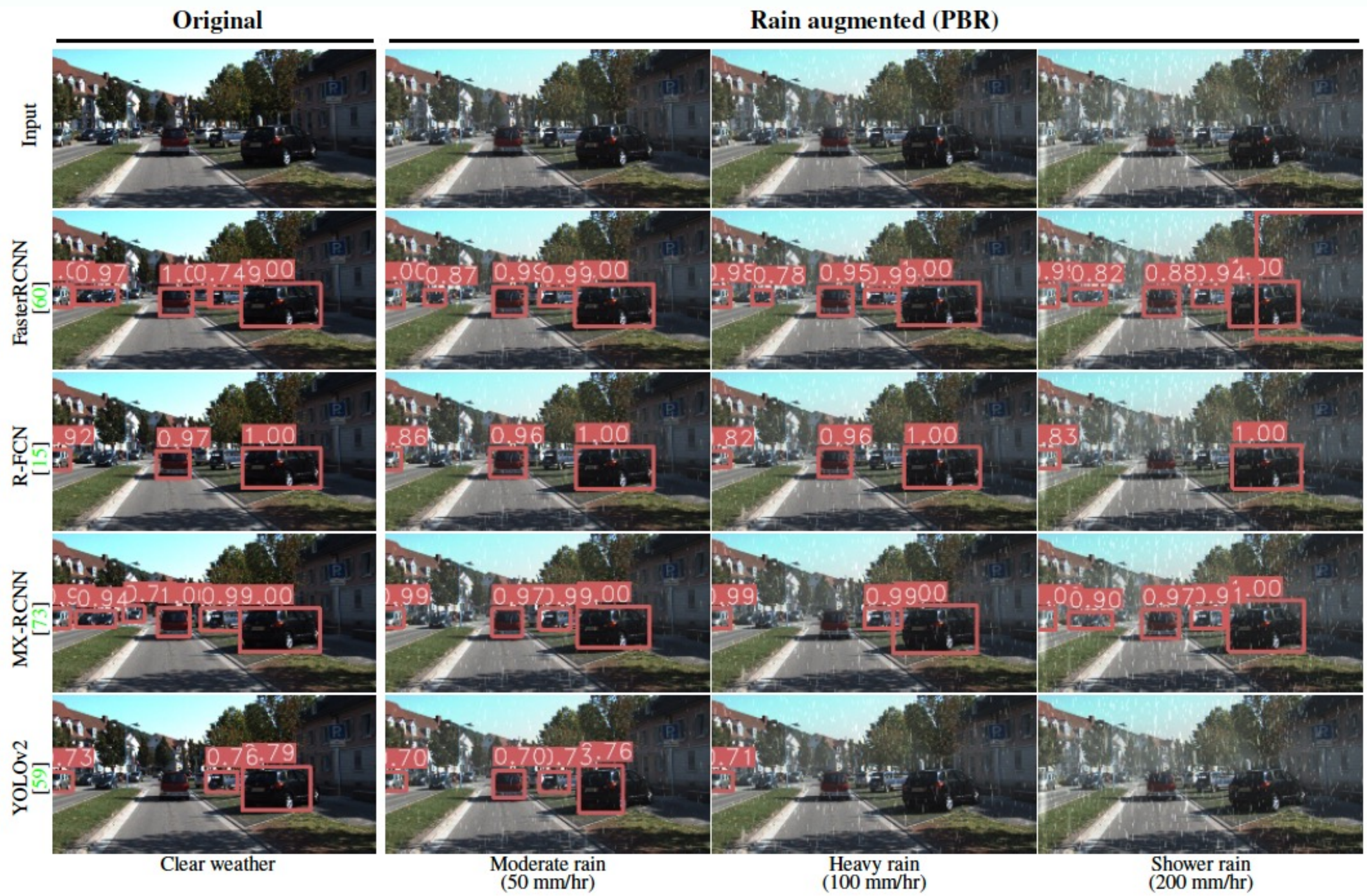


Fig. 11 Object detection on PBR rain augmentation of KITTI. From left to right, the original image (clear) and three PBR augmentations with varying rainfall rates. Images are cropped for visualization.

Tremblay et al 20

Simulating rain - issues

- Near field:
  - drops are bright, discrete, likely ballistic motion
    - how bright?
    - where?
    - how moving?
  - likely air is “wet”
    - so some fogging, depending on depth
- Far field:
  - fog like effects
- So we need to know
  - depth, environment map, falling drops, camera movement

Simulating rain

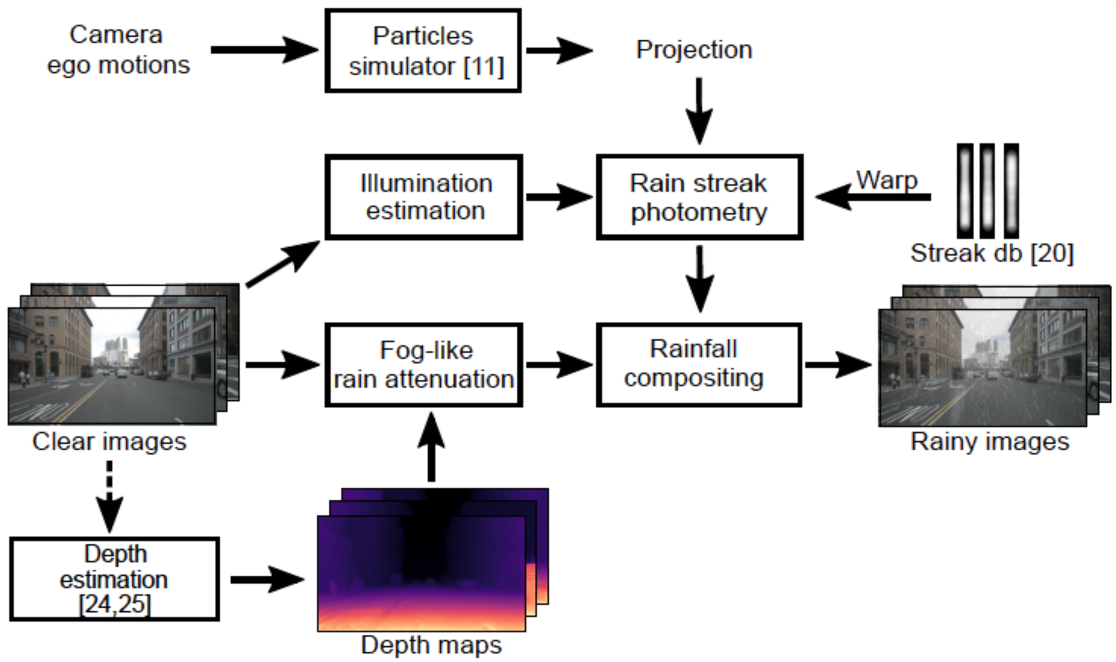


Fig. 2 Physics-Based Rendering for rain augmentation. We use particles simulation together with depth and illumination estimation to render arbitrarily controlled rainfall on clear images.

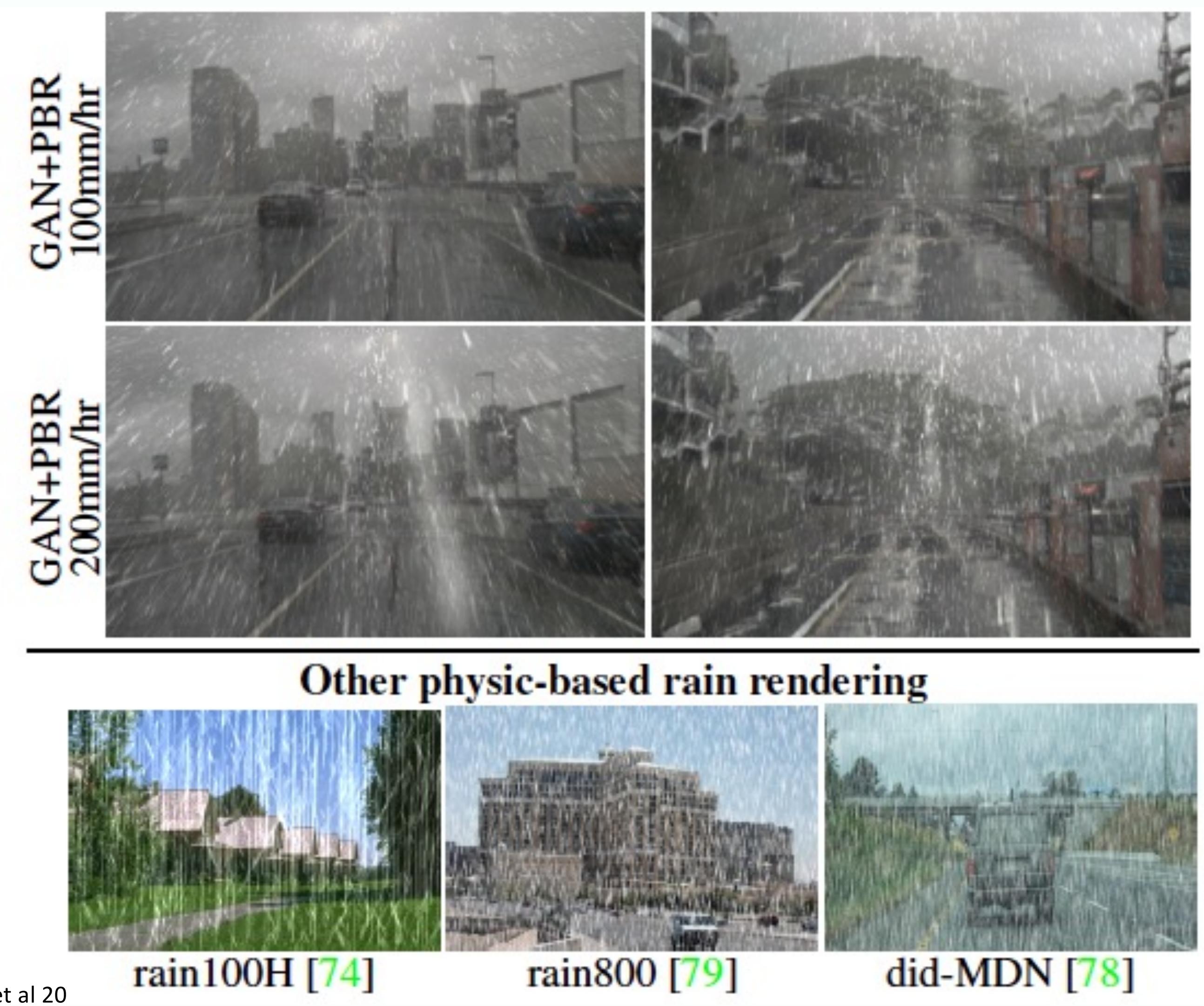
Tremblay et al 20

Simulating rain

- Trick:
  - rain causes color effects, specular effects etc.
    - CycleGAN is good at this, but bad at streaks
    - Physics based simulation is bad at this but good at streaks

Fig. 5 GAN+PBR rain-augmentation architecture. In this hybrid approach, clear images are first translated into rain with CycleGAN [83] and subsequently augmented with rain streaks with our PBR pipeline (see fig. 2).

Tremblay et al 20



Tremblay et al 20

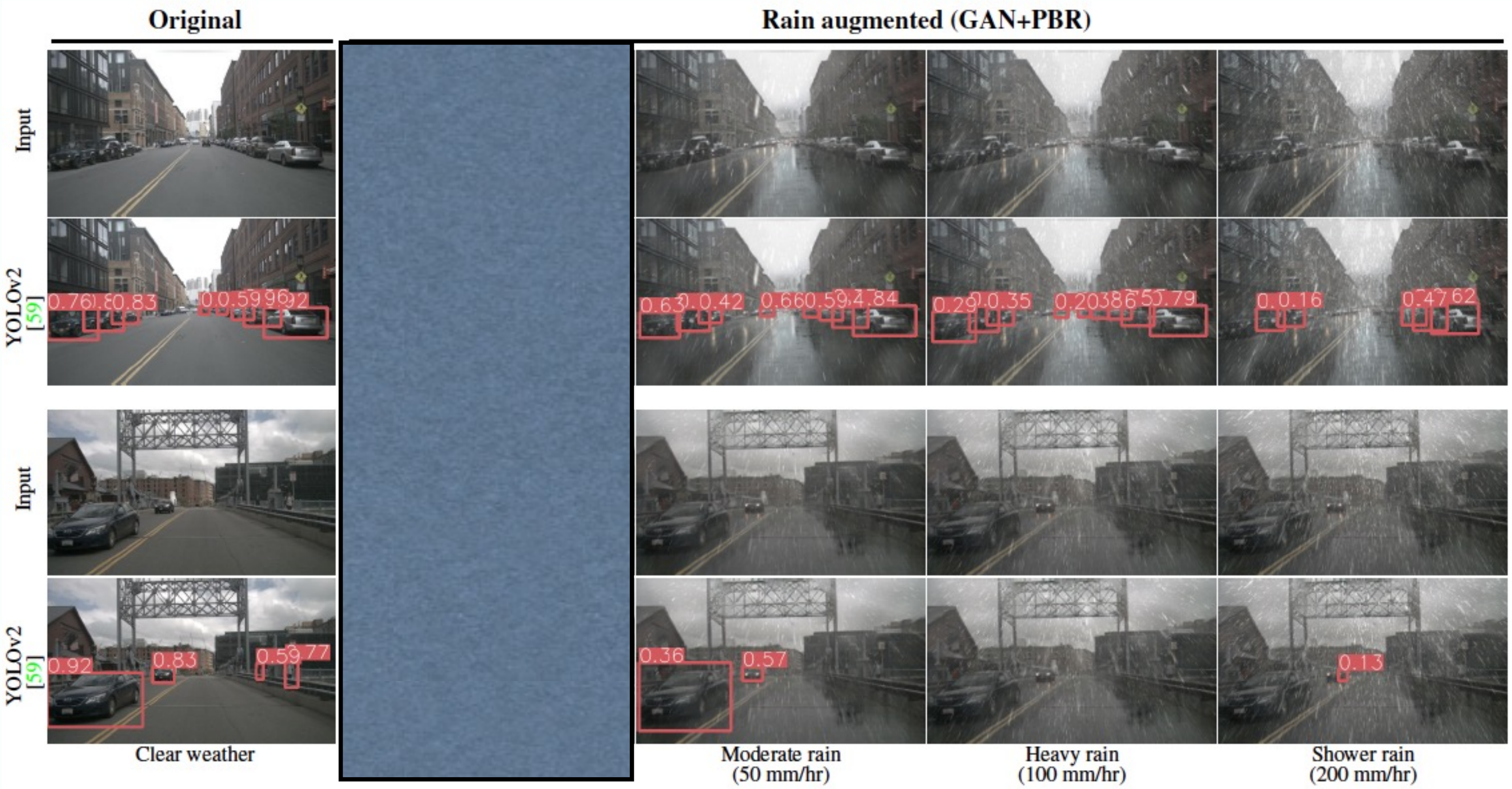


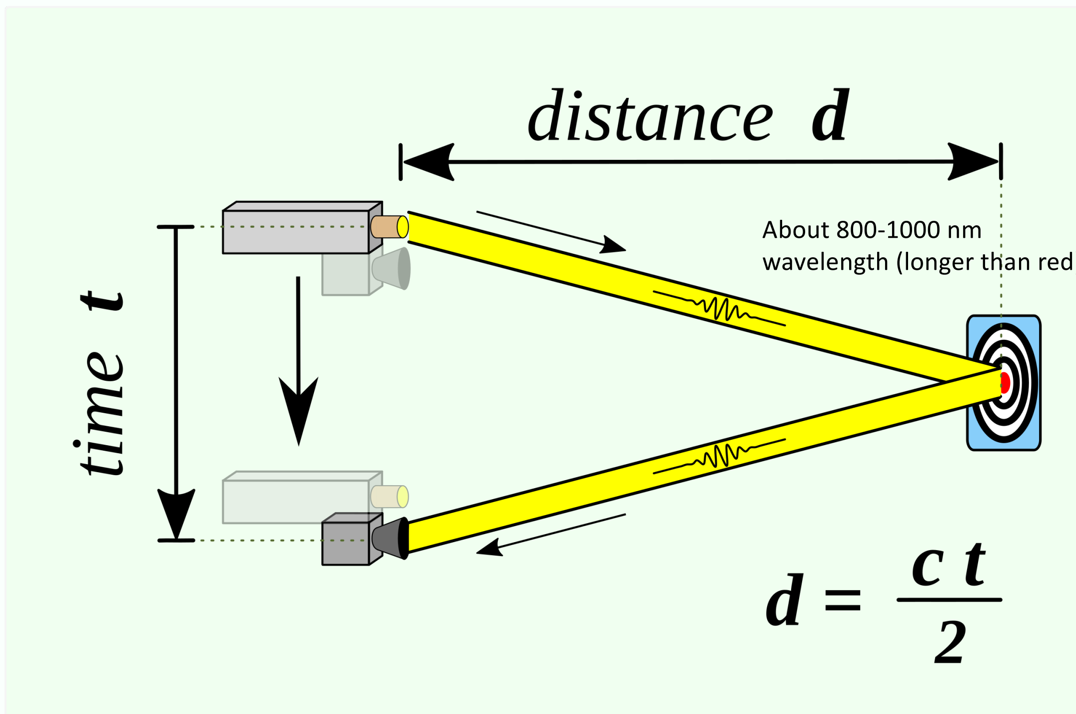
Fig. 15 Object detection on our GAN+PBR augmented nuScenes. From left to right, the original image (clear), the GAN augmented image and three GAN+PBR images.

Tremblay et al 20

Why not just use LIDAR?

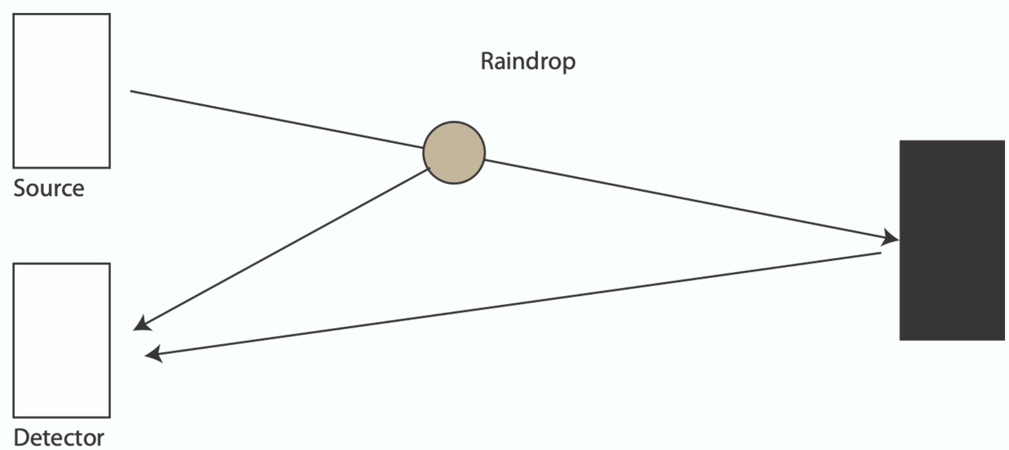
- Cause LIDAR suffers from weather problems, too

Fog and Lidar: Lidar



Wikipedia

Raindrop scatter



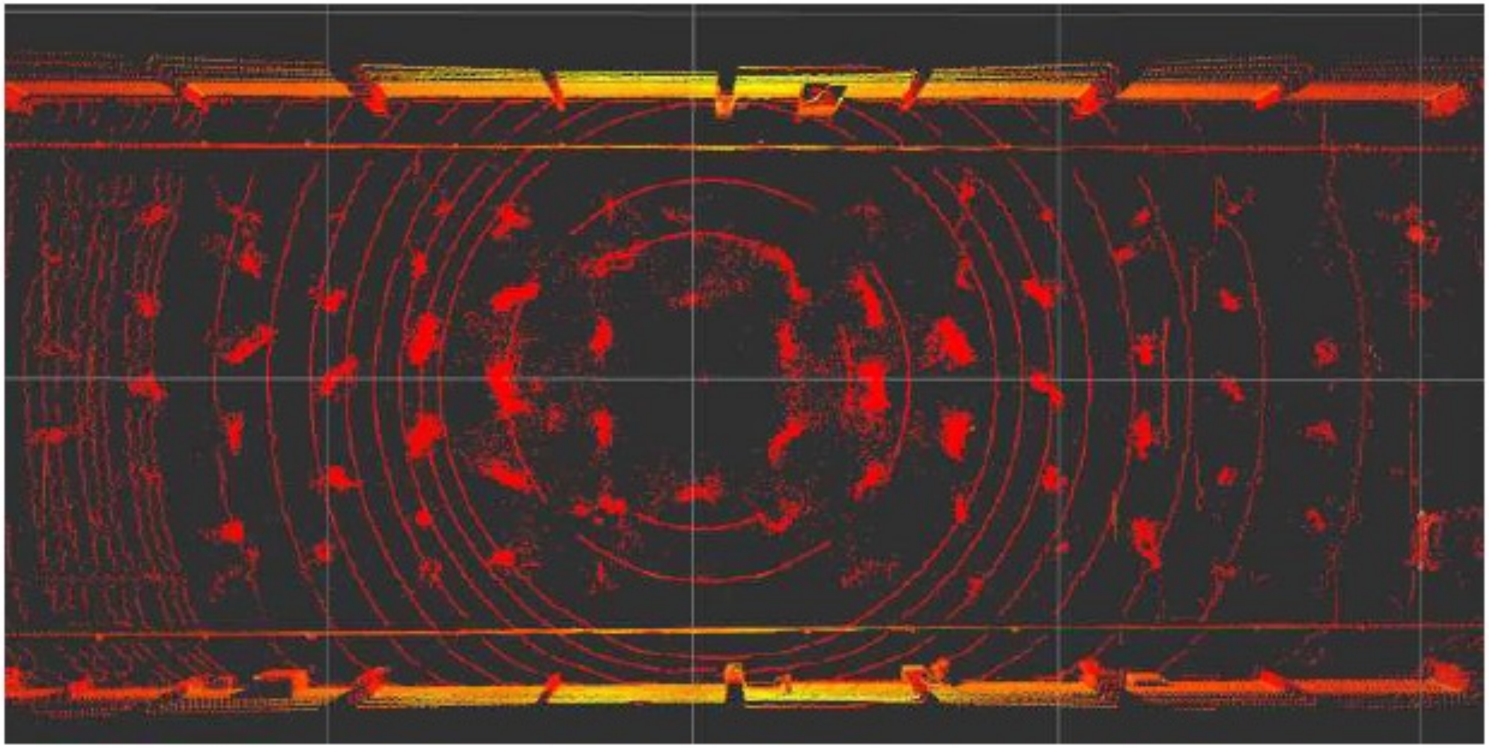
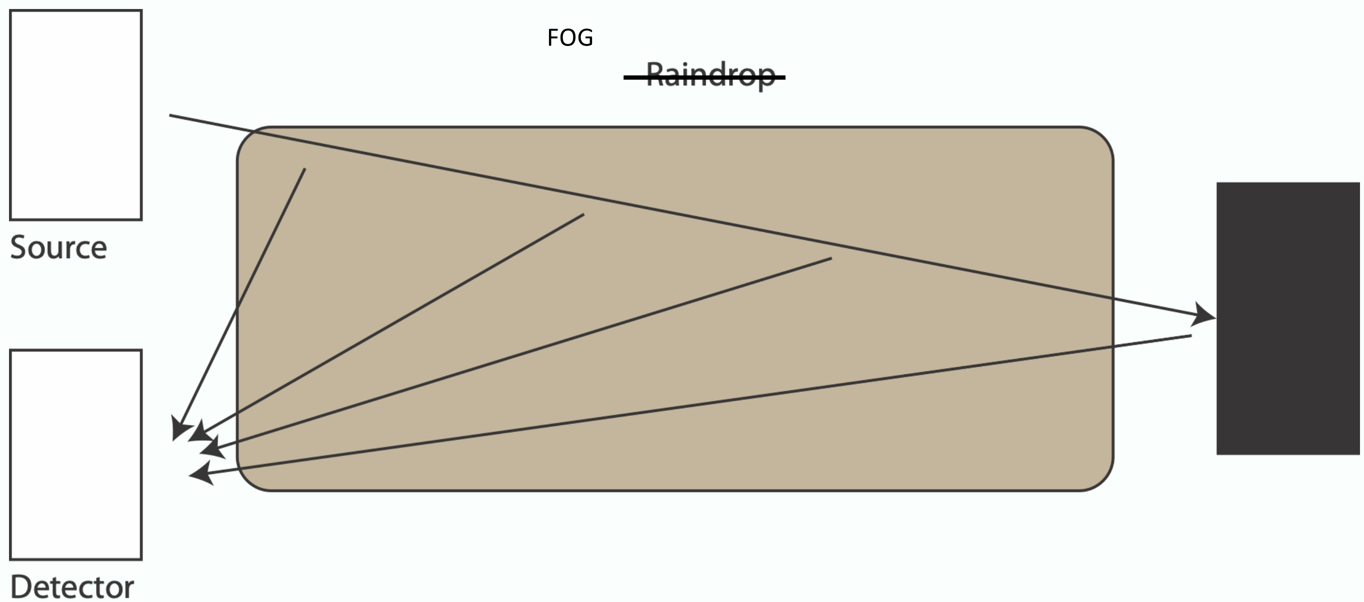


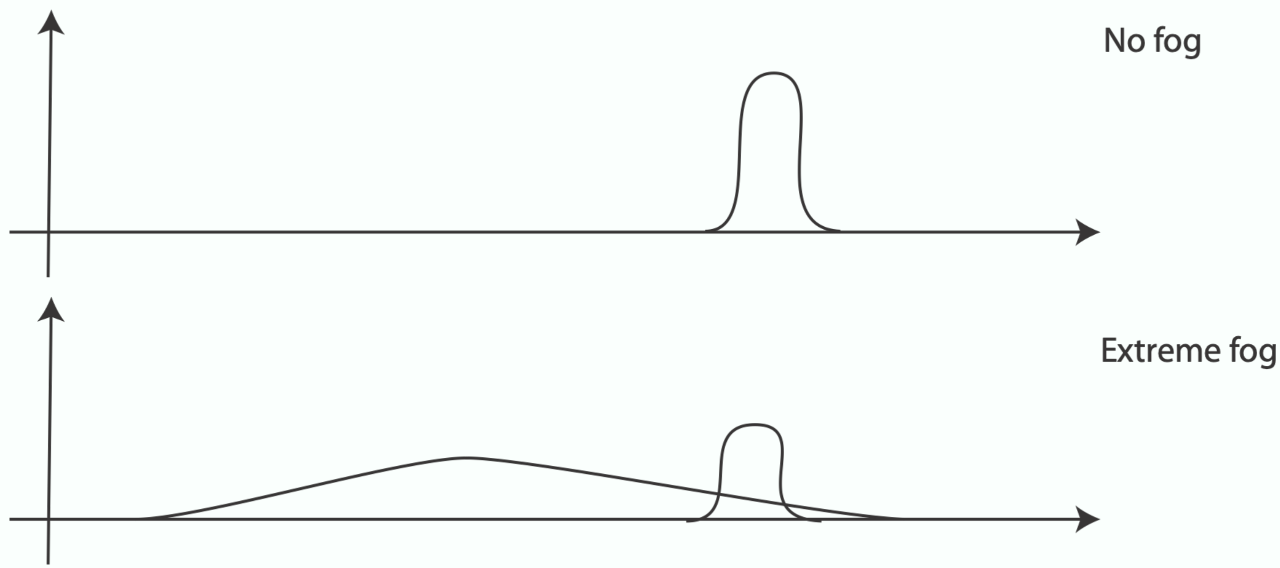
Fig. 9: "Rain pillars" as detected by a LiDAR.

Carballo, 20

Fog scattering



What the sensor sees…



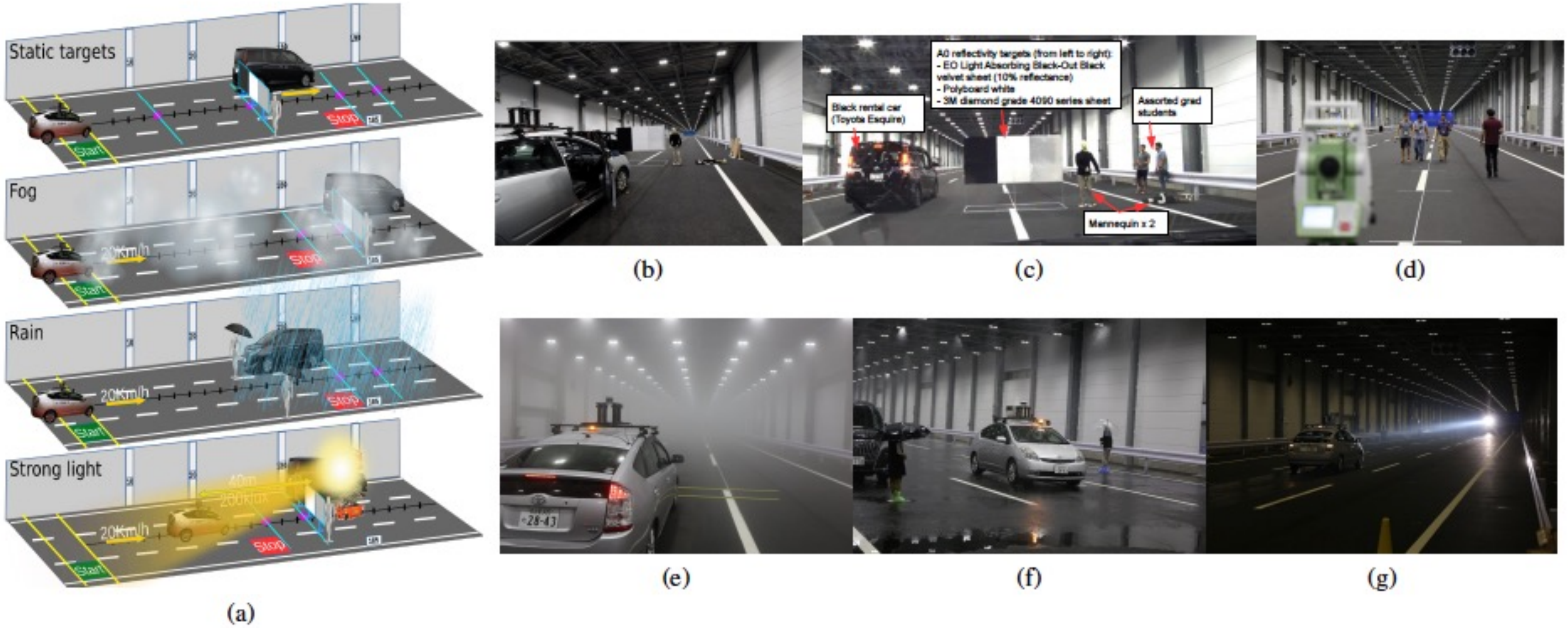


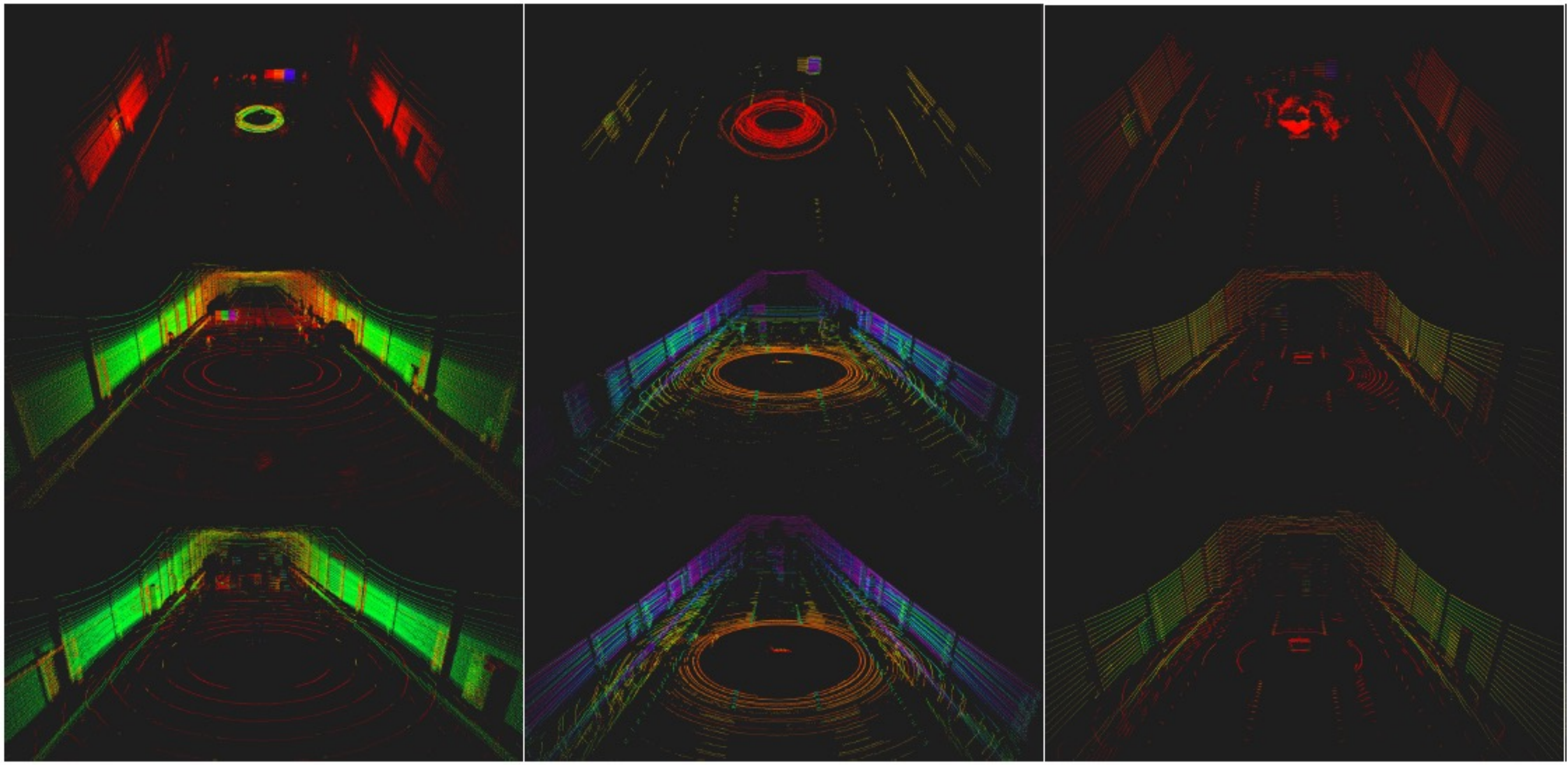
Fig. 5: Static targets and adverse weather experiments at JARI's weather chamber: (a) configuration of the different scenarios, (b) and (c) measurement, (e) to (g) sample adverse weather scenes, (d) setting up ground truth.

Carballo, 20

Fog

Rain

Very
bright
light



(a) VLS-128 (b) HDL-64S2 (c) HDL-32E

Carballo, 20

Radar is unaffected

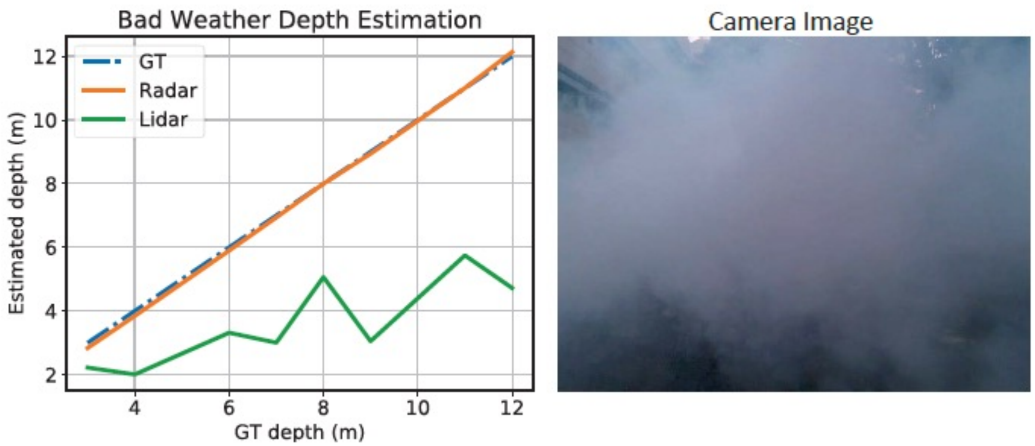


Figure 16: *Performance comparison of different sensors in the presence of adverse conditions. The left plot shows the depth estimation performance of Radar and LiDAR for an object directly in front of the sensor in the presence of fog. The right figure shows the camera image for the experiment.*

Bansal et al 20

Astonishing fact:

- You can generate images from random vectors
  - And they’re very good
- Questions:
  - How good are generators?
    - Extremely hard qn
      - How do you score “good”?
  - What do they get right?
    - Or wrong?
  - What do they “know” about images?
  - Can you control them?

Generative strategy

- StyleGAN is a network that
  - accepts random vectors
  - produces images

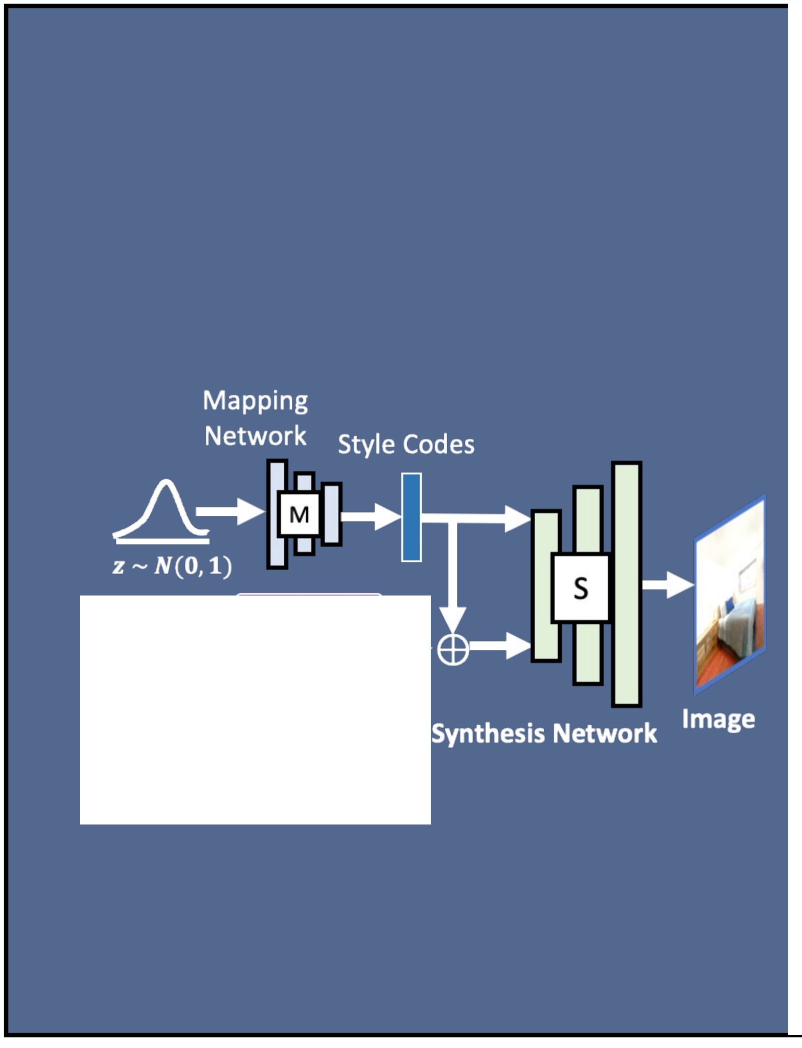
Kerras et al 20

A portrait of a human growing colorful flowers from her hair. Hyperrealistic oil painting. Intricate details.

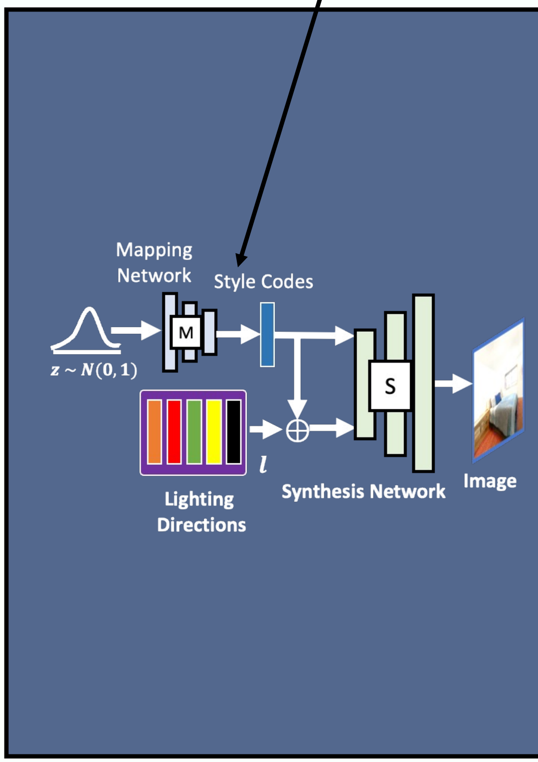
Kang et al., CVPR 2023

Yu et al. ICCV 2021
Yang et al. NeurIPS 2022

How StyleGAN works (ish)

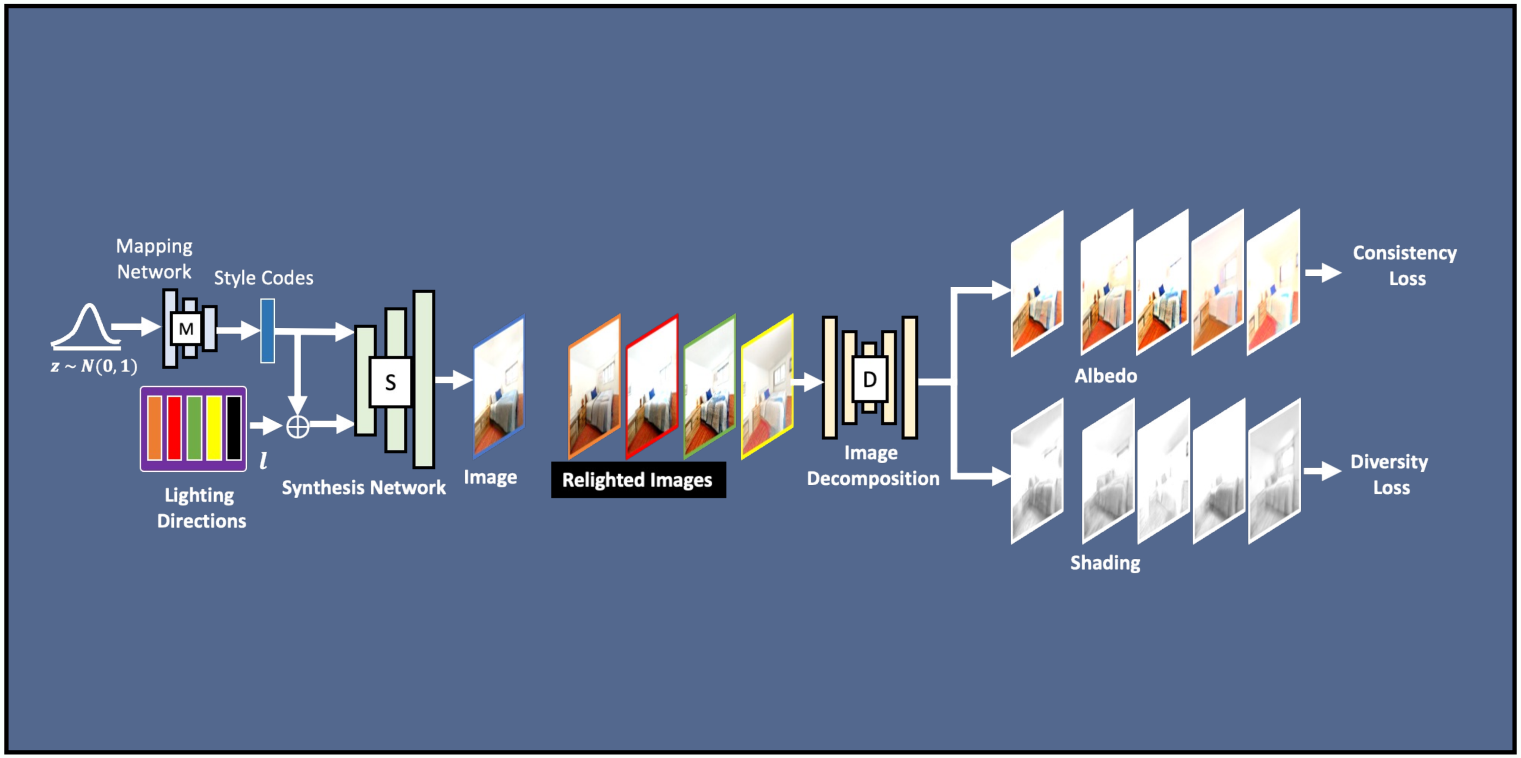


- Add offset to StyleGAN latents
  - various effects by choice of offset
- Q: how to get desired result?
  - A (ish): search offsets



Yang et al. IJCV 2020; Epstein et al. ECCV 2022; Shi et al. NeurIPS 2022

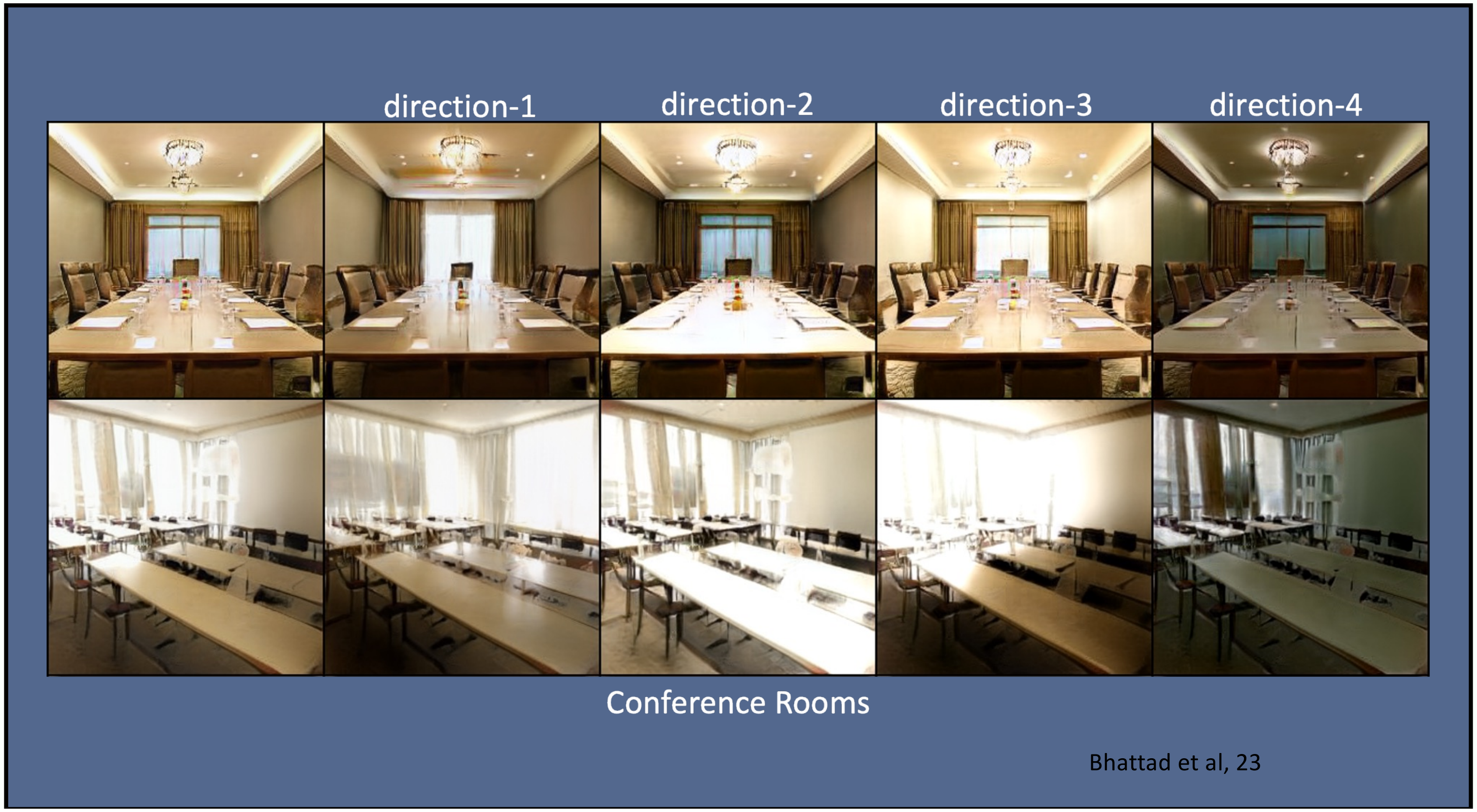
Find directions that fix albedo



Bhattad et al, 23

Relightings are realistic

Relightings are realistic

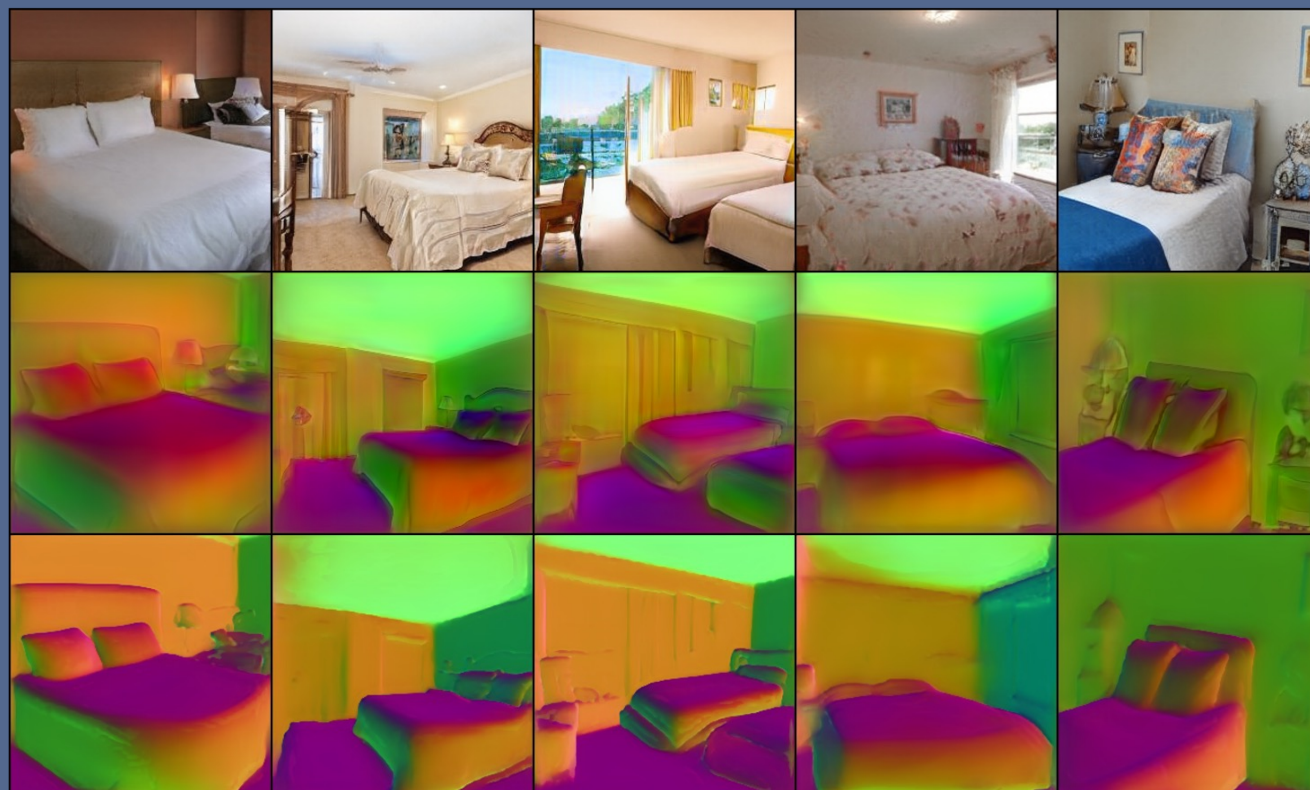


If you can relight images

- you must know stuff about
  - depth
  - normal
  - surface properties

- Q: Does StyleGAN
- A1:
  - It should (kind of obvious)
- A2:
  - It can be made to produce this information (astonishing)

… Normal



Bhattad et al, 23

… Depth

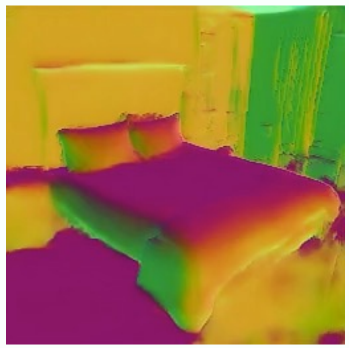
Bhattad et al, 23

… Segmentation

Bhattad et al, 23

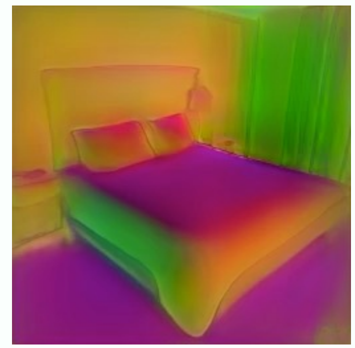
StyleGAN Normals behave well

Image +
Relighting



Recent Supervised SOTA
(XTC)

Current Supervised SOTA
(Omnidata v2)



StyleGAN Generated
(Ours)

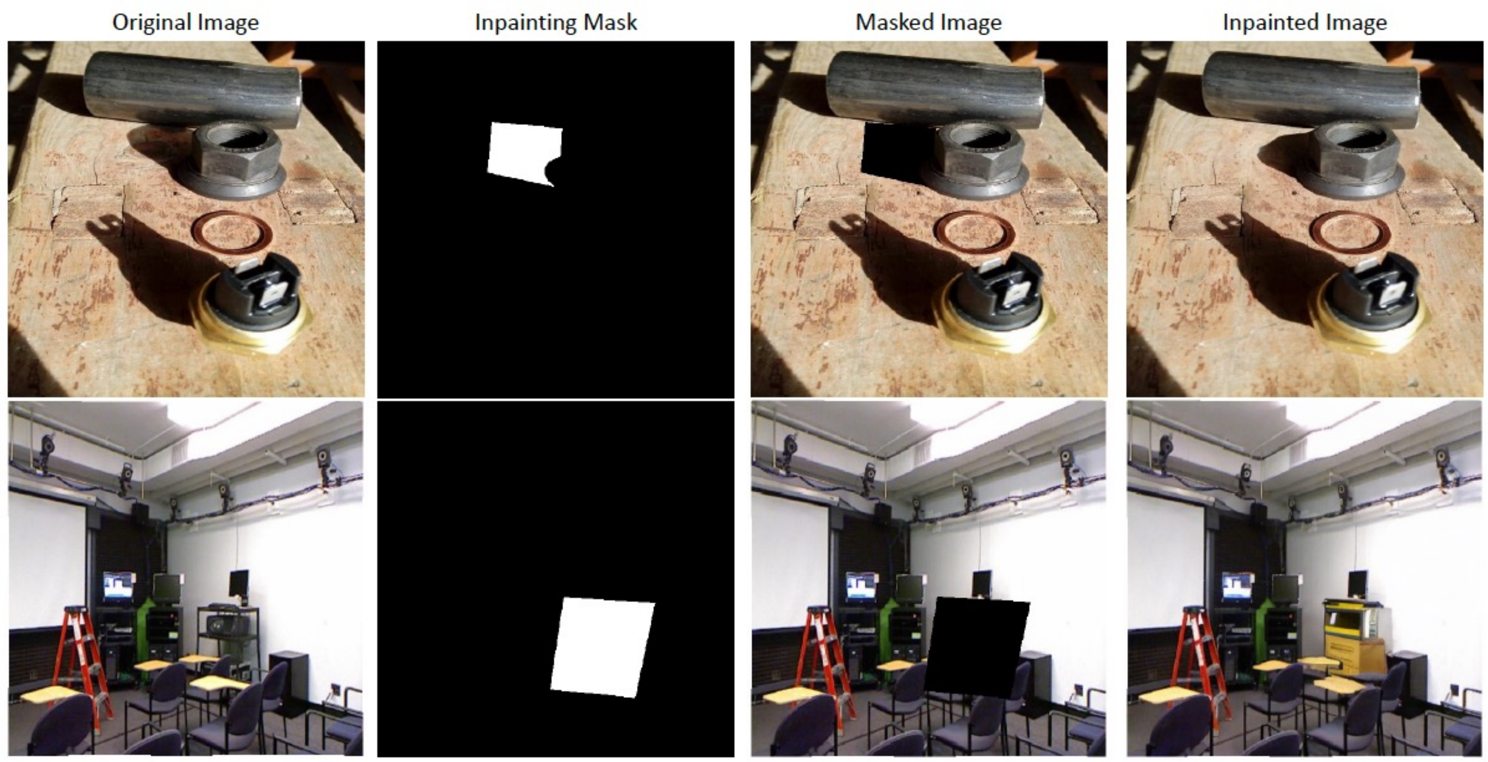
Zamir et al CVPR 2020
Kar et al CVPR 2022

Bhattad et al, 23

What Does Stable Diffusion Know about the 3D Scene?

Guanqi Zhan[1], **Chuanxia Zheng**[1], **Weidi Xie**[1,2], **Andrew Zisserman**[1]
Visual Geometry Group, University of Oxford[1]
Coop. Medianet Innovation Center, Shanghai Jiao Tong University[2]
{guanqi,cxzheng,weidi,az}@robots.ox.ac.uk

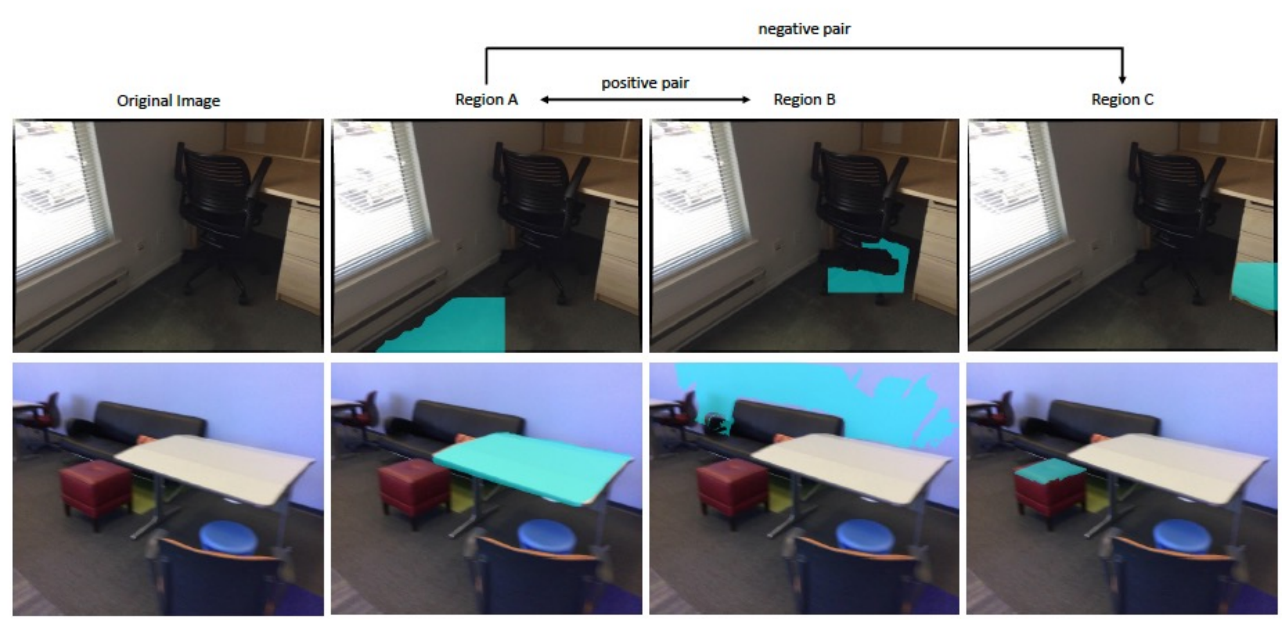
Stable diffusion “knows” geometry



Zhan et al, 23

Methodology: probing features

- Mark up standard dataset with geometric properties
- Q: can denoiser features predict this markup?



Zhan et al, 23

Stable diffusion "knows" some geometry

Table 4: **Performance of Stable Diffusion feature compared to state-of-the-art self-supervised features.** For each property, we use the best time step, layer and C found in the grid search in Table 2 for Stable Diffusion, and the final layer for other self-supervised features. The performance is the ROC AUC on the test set, and 'Random' means a random classifier.

Property	Random	OpenCLIP	DINOv1	DINOv2	Stable Diffusion
Same Plane	50	74.6	79.3	86.0	**95.0**
Perpendicular Plane	50	55.5	59.8	63.4	**83.9**
Material	50	60.4	62.1	63.8	**79.4**
Support Relation	50	84.7	84.3	88.3	**94.4**
Shadow	50	75.5	84.3	86.8	**94.5**
Occlusion	50	63.8	60.0	67.9	**75.6**
Depth	50	95.5	93.7	98.0	**99.3**

Zhan et al, 23

Generators make fascinating errors

Zhan et al, 23

Iffy projective geometry

https://huggingface.co/spaces/stabilityai/stable-diffusion/discussions/1593

Iffy projective geometry

https://huggingface.co/spaces/stabilityai/stable-diffusion/discussions/1593

Iffy projective geometry

https://huggingface.co/spaces/stabilityai/stable-diffusion/discussions/1593

Iffy lighting geometry

Weird errors in clothing

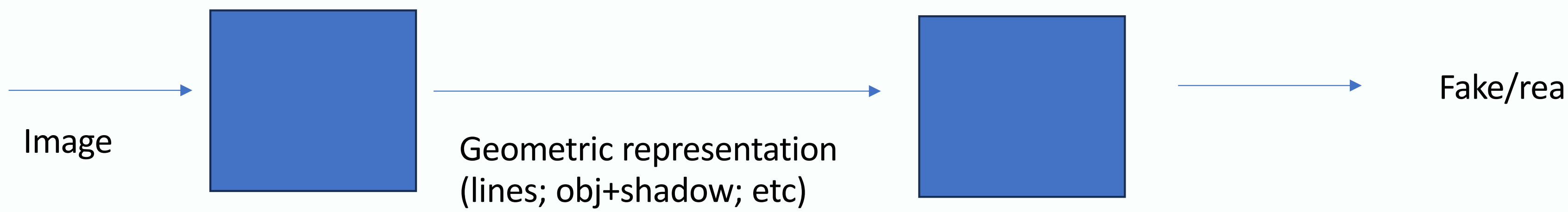
Zhu et al 23

Weird errors in clothing

Questions:

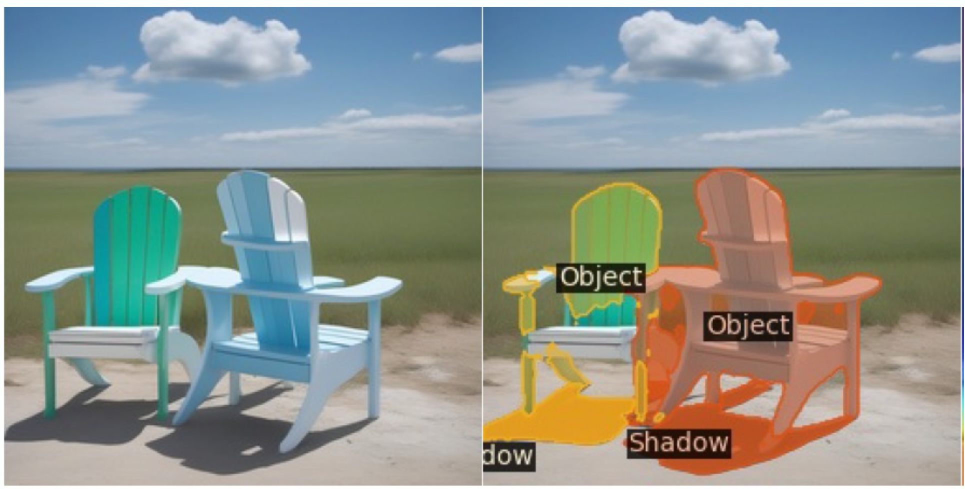
- Can you find these errors automatically?
  - A1: for some of them, yes
    - and it’s easy to use them to identify generated images very accurately

For some of them, yes



Sarkar et al, 23

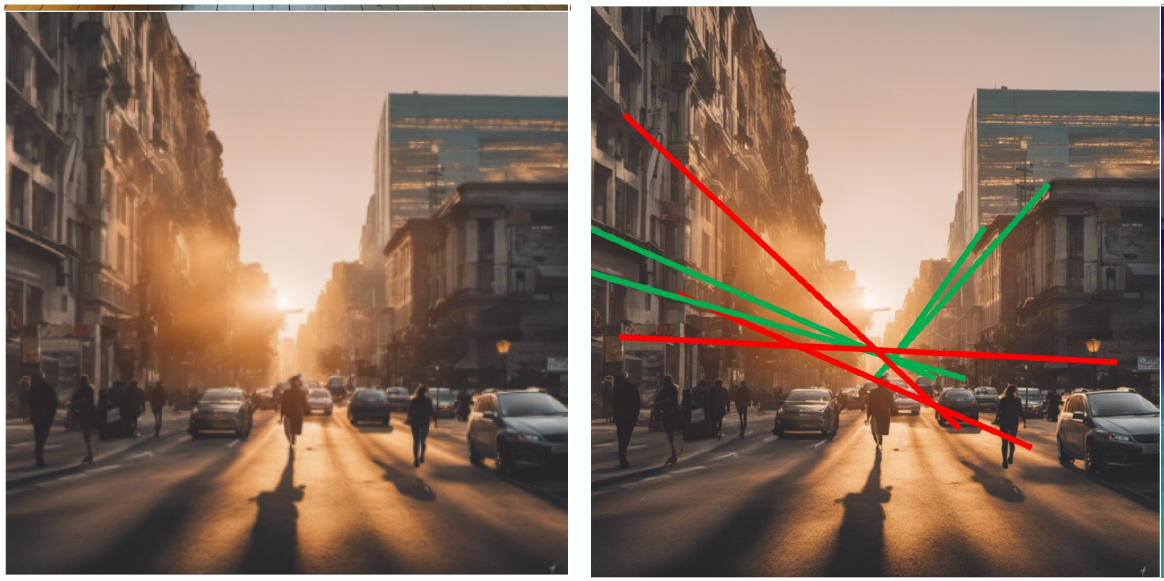
Geometric representations



Generated Image   Object-Shadow (OS)

Sarkar et al, 23

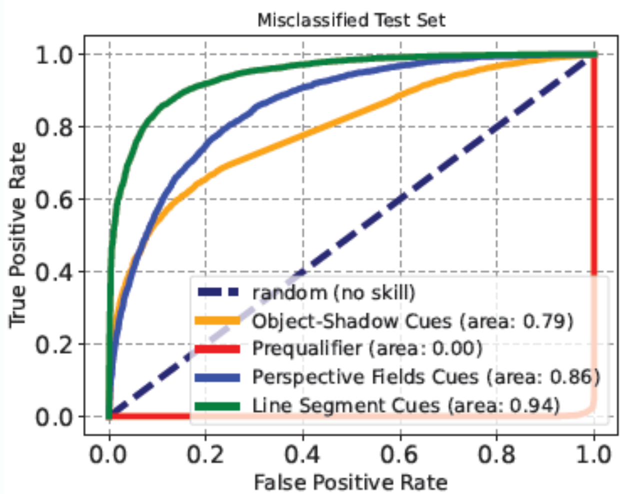
Geometric representations



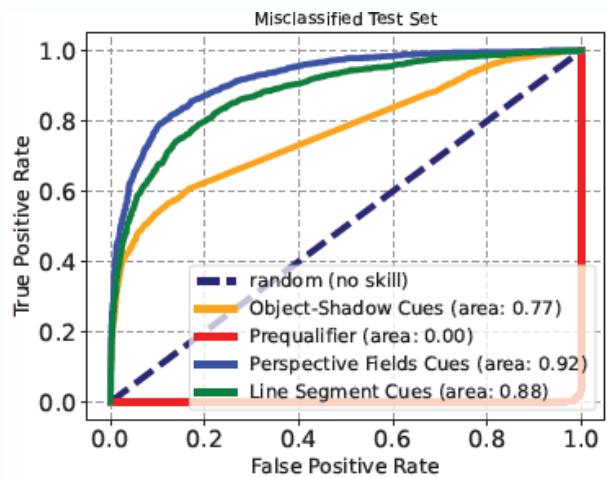
Generated Image   Vanishing Point Errors

Sarkar et al, 23

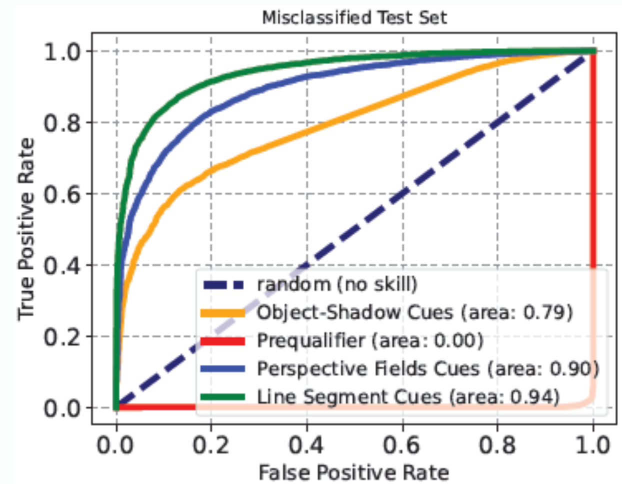
For some of them, yes



(c) Misclassified Test Set (Indoor)



(f) Misclassified Test Set ((Outdoor))



(i) Misclassified Test Set (Indoor + Outdoor)

Sarkar et al, 23

Questions:

- Can you find these errors automatically?
  - A1: for some of them, yes
    - and it’s easy to use them to identify generated images very accurately
  - A2: but for others, no
    - the garment example is fantastically hard, and important
- What causes them?
  - A: ?
- Can you make them go away?
  - A:?