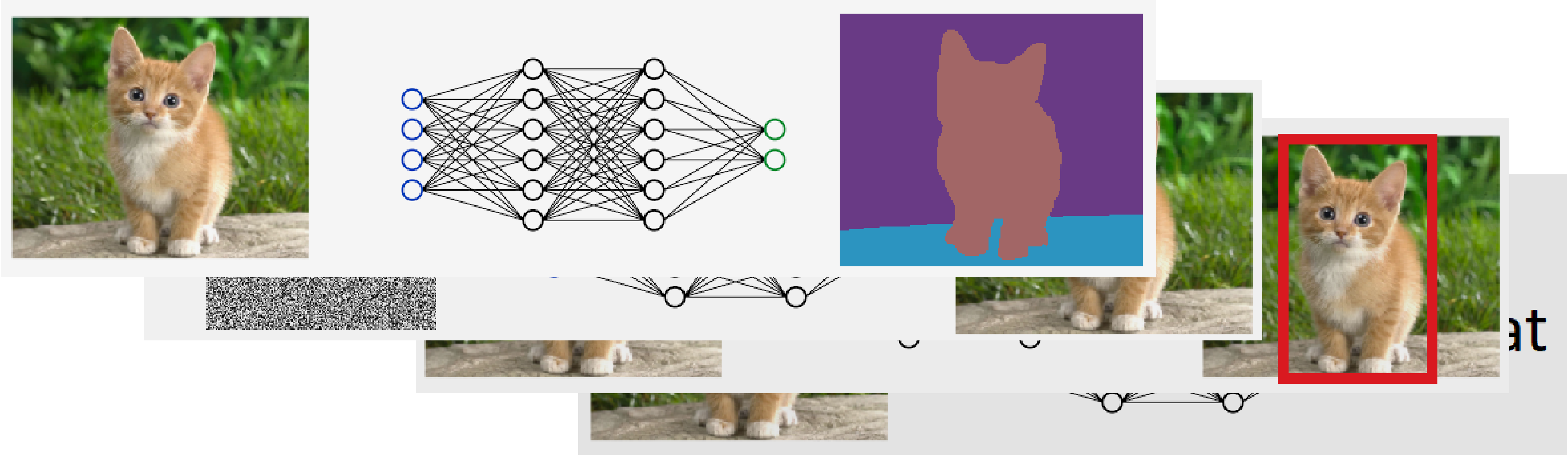
# Lecture 19
# Computer Vision and Inverse problems
# Computer Graphics and Rendering

January 11st 2023

Meirav Galun

# Computer vision

We are familiar already with a variety of neural net architectures, dedicated for tasks such as  classification, detection, generation, segmentation...



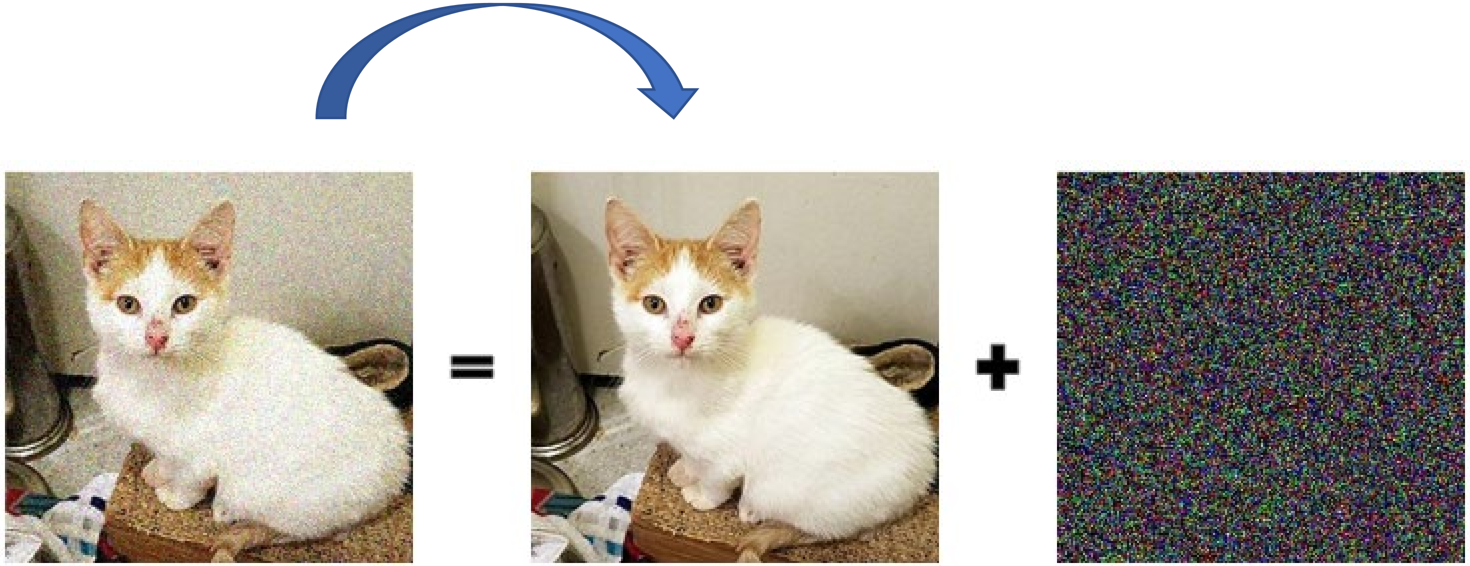
# Inverse problems in computer vision

- Restoration tasks: denoising, super-resolution and inpainting

# Inverse problems in computer vision

- Restoration tasks: denoising, super-resolution and inpainting

Denoising



degraded image    clean image

# Inverse problems in computer vision

Super-resolution

Inpainting

# Inverse problems in computer vision

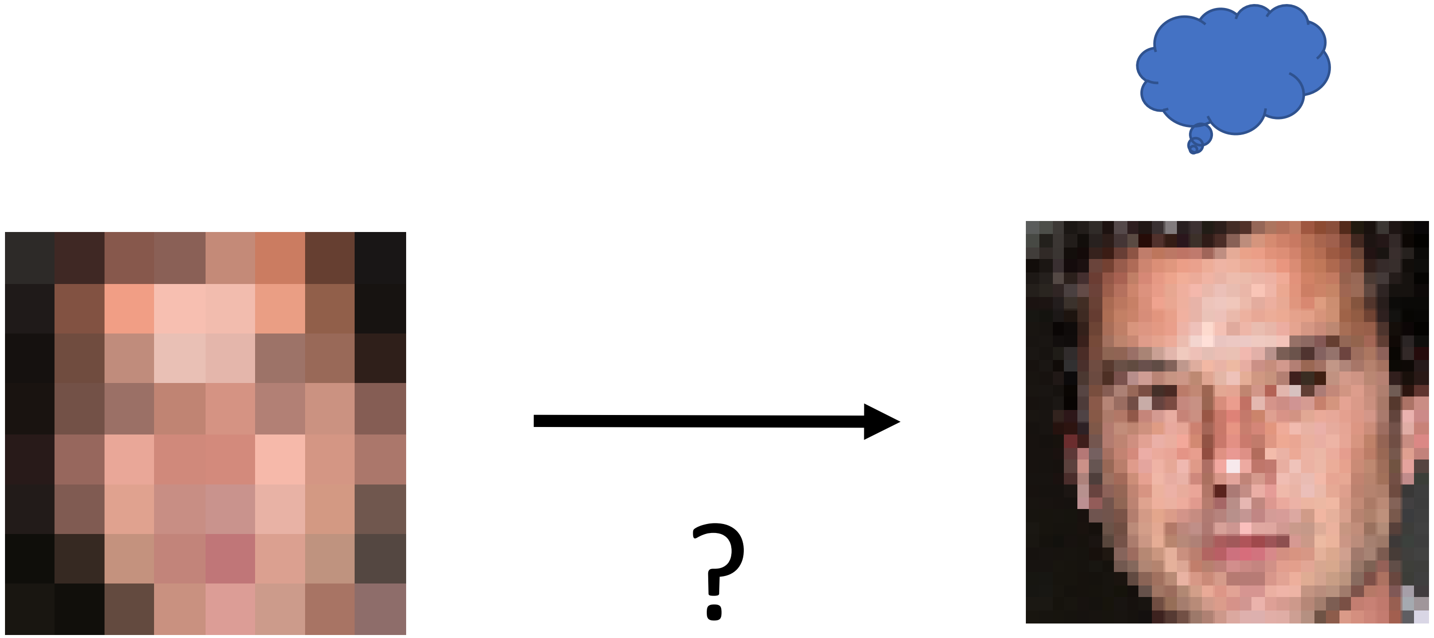
- Supervised approach for solving inverse methods performs well, when utilizing deep convolutional networks

- CNNs are trained over a large number of pairs of degraded images and their corresponding clean images

- The excellent performance of the CNNs is attributed to their ability to learn ***realistic image priors*** from a ***large training dataset*** of images

- Is the common approach of supervised training of a large dataset indeed the best / possible way to learn image priors?

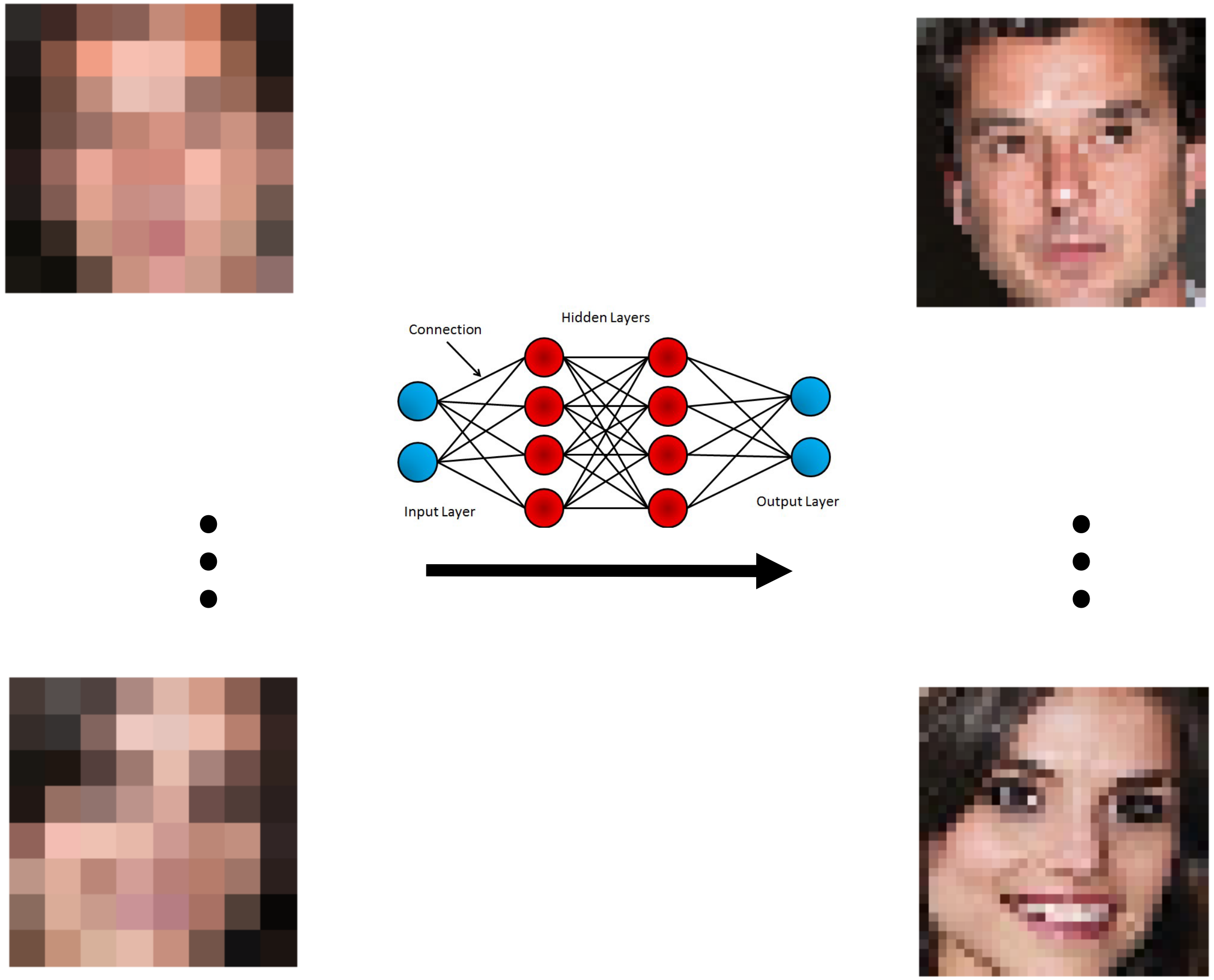
# What is a prior?




**Prior = our knowledge about the visual world**

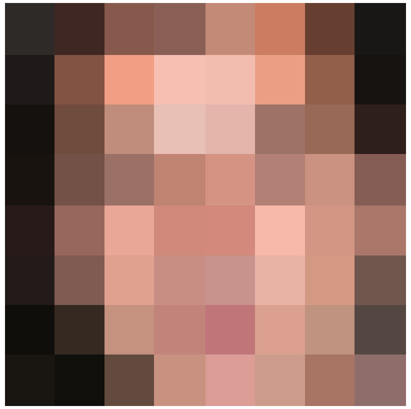
Images from Dahl et al, 2017

# Learned priors




Images from Dahl et al, 2017

$\hat{x}$



# Explicit priors

down-sampling

- By introducing constraints

$$\min_x ||d(x) - \hat{x}||$$

s.t. $x$ is a face, natural image, etc.

- $R(x)$ expresses constraints

$$\min_x ||d(x) - \hat{x}|| + \lambda R(x)$$

- <u>Example:</u> Total Variation (TV) $R(x) = \sum_{i,j} |x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|$

  encourages images to contain uniform regions

- In general, it is difficult to express “natural” constraints mathematically

# Deep image (implicit) prior

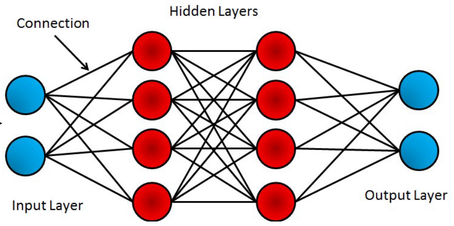
- Constructing an <u>implicit prior</u> by neural network

$$\min_{x} \|d(x) - \hat{x}\|$$

s.t. $x$ is an output of CNN

The network weights parametrize the restored image

$z \rightarrow$

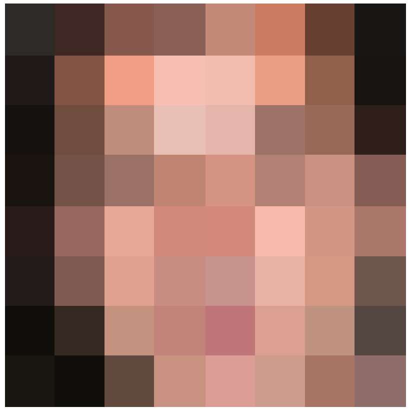



Noise

$\longrightarrow$ $x$

$\rightarrow \|d(x) - \hat{x}\|$

$\hat{x}$



Ulyanov, D., Vedaldi, A., & Lempitsky, V., Deep image prior, CVPR, 2018

# Deep image prior

- Pretrained network or large image datasets are not required
- Eliminating the data and the learning process still yields good results for image restoration
- The structure of the network is sufficient and imposes a strong prior to restore the original image while taking into consideration the degraded image only

The structure of the CNN imposes a strong prior
Why?



Ulyanov, D., Vedaldi, A., & Lempitsky, V., Deep image prior, CVPR, 2018

# Deep image prior

## Why do the structure of the CNN impose a strong prior?

- The network captures low-level statistics of natural images
- The structure of the network imposes self-similarity (within the same scale) at multiple scales, making the corresponding priors suitable for the restoration of natural images
  - The translation equivariance and locality of the convolution operator
  - The hierarchy of such convolutions captures the statistics of pixel neighborhood at multiple scales



Ulyanov, D., Vedaldi, A., & Lempitsky, V., Deep image prior, CVPR, 2018

# Deep image prior
# → Computer vision, Computer Graphics

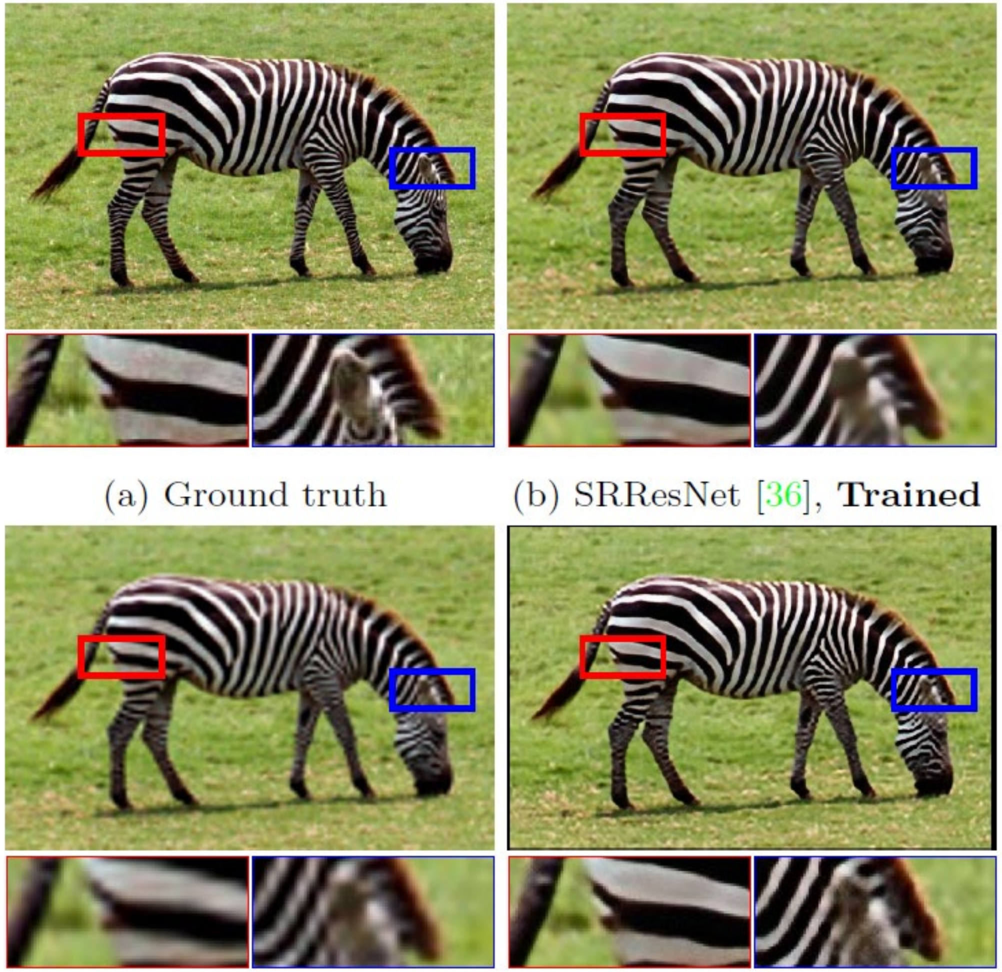
The structure of the network allows

Parametrizing signals by the net weights
1. 2D images
2. 3D volumes
3. Continuous functions

How and why is it useful?

Diving into “deep image prior”

# Deep image prior, super-resolution results



(a) Ground truth (b) SRResNet [36], **Trained**

(c) Bicubic, **Not trained** (d) Deep prior, **Not trained**

# Image restoration

## Problem setting

$x$ - clean image
$\hat{x}$ - degraded image (observed)
$x^*$ - restored image

$x$ → degradation → $\hat{x}$ → restoration → $x^*$

# Image restoration

## Problem setting

$x$ - clean image

$\hat{x}$ - degraded image (observed)

$x^*$ - restored image

$x$ → degradation → $\hat{x}$ → restoration → $x^*$

(maximum a posterior probability) **MAP**: $x^* = \arg\max_x p(x|\hat{x})$

$$p(x|\hat{x}) = \frac{p(\hat{x}|x)p(x)}{p(\hat{x})} \propto \underbrace{p(\hat{x}|x)}_{\text{Likelihood}}\underbrace{p(x)}_{\text{Prior}}$$

# Image restoration

## Example: Likelihood for denoising

clean image $x$

corrupted image $\hat{x}$

restored image $x^*$

degradation

$$\hat{x} = x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$p(\hat{x}|x) = \mathcal{N}(\hat{x}; x, \sigma^2)$$

restoration

$$x^* = \arg\max_x p(x|\hat{x})$$
$$= \arg\max_x p(\hat{x}|x)p(x)$$

# Image restoration

## The significant role of the prior

$$x^* = \arg\max_x p(x|\hat{x}) = \arg\max_x p(\hat{x}|x)\cancel{p(x)} = \arg\max_x p(\hat{x}|x) = \arg\max_x \mathcal{N}(\hat{x}; x, \sigma^2) = \hat{x}$$

clean image $x$

corrupted image $\hat{x}$

restored image $x^* = \hat{x}$

# Image restoration

## Alternative notation

clean image $x$
corrupted image $\hat{x}$
restored image $x^*$

data term $E(x;\hat{x})$

image prior term
(regularization) $R(x)$

$$
\begin{aligned}
x^* &= \arg\max_x p(x|\hat{x}) \\
&= \arg\max_x p(\hat{x}|x)p(x) \\
&= \arg\min_x -\log p(\hat{x}|x) - \log p(x) \\
&= \arg\min_x E(x;\hat{x}) + R(x)
\end{aligned}
$$

Example:

$$p(\hat{x}|x) = \mathcal{N}\left(\hat{x}; x, \sigma^2\right) \Rightarrow E(x;\hat{x}) = \|x - \hat{x}\|^2$$

# Image restoration

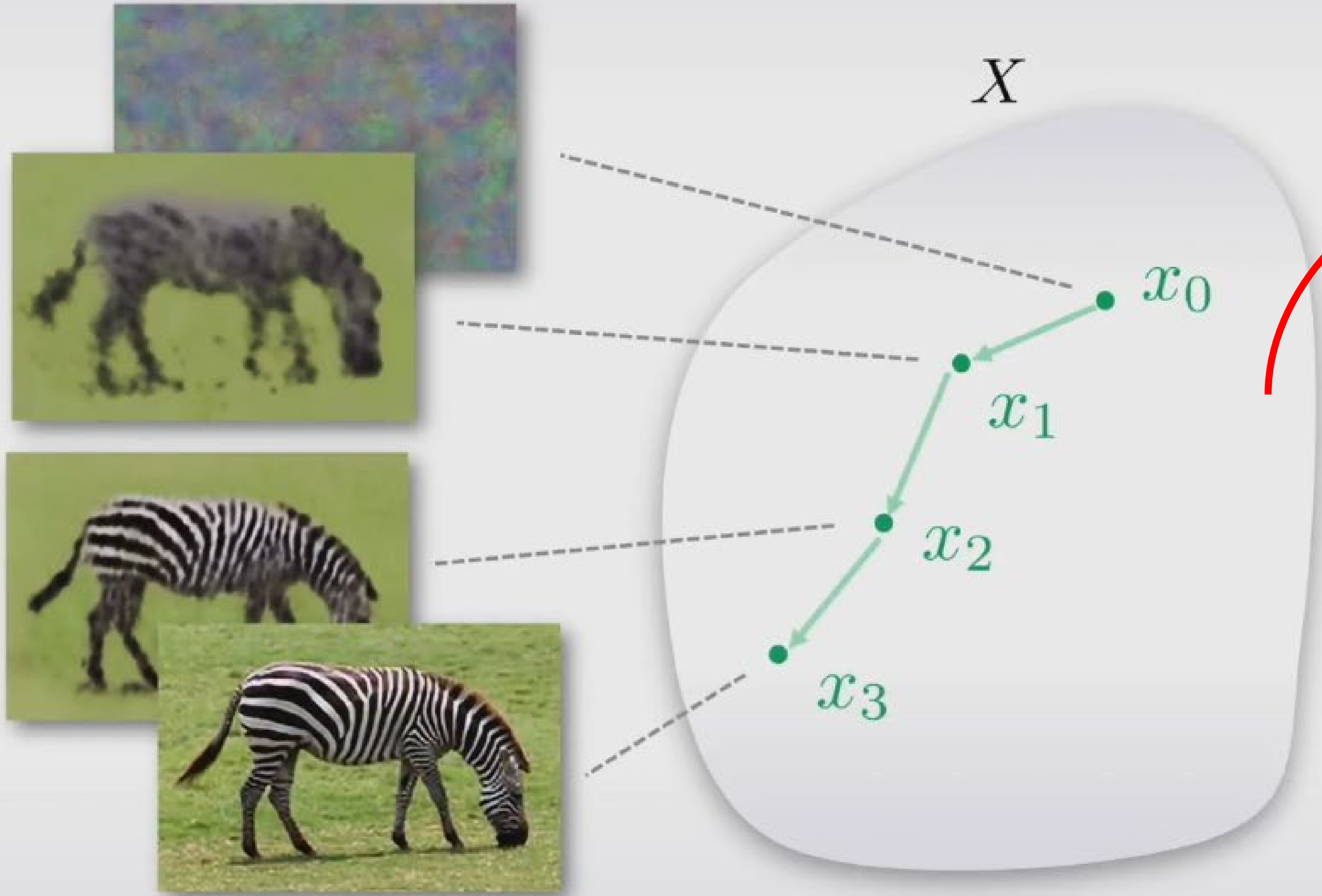
## Optimization task

clean image $x$
corrupted image $\hat{x}$
restored image $x^*$

$$x^* = \arg\min_x E(x; \hat{x}) + R(x)$$

# Regular

$$x^* = \arg\min_x E(x;\hat{x}) + R(x)$$

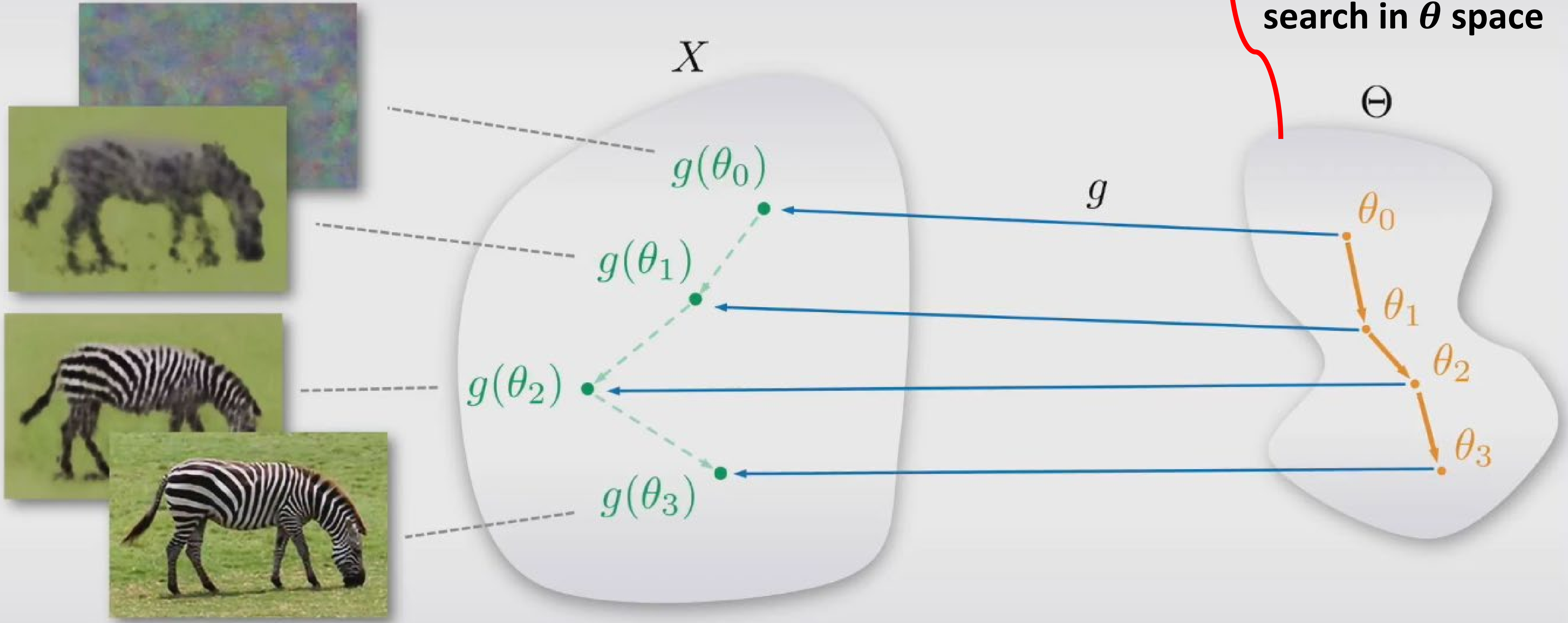



**search in the image space**

# Parametrized

$$x^* = \arg\min_\theta E(g(\theta); \hat{x}) + R(g(\theta))$$

search in $\theta$ space



# Image restoration

## Optimization by parametrization

clean image $x$
corrupted image $\hat{x}$

**Regular optimization:** $$\arg\min_{x} E(x;\hat{x}) + R(x)$$

**By parametrization:** $$\arg\min_{\theta} E(g(\theta);\hat{x}) + R(g(\theta))$$

If $\boldsymbol{g}$ is surjective (i.e., for each $x$ $\exists\theta$ s.t. $g(\theta) = x$) then the two problems shares the same minima (i.e., equivalent)

*So, why to switch between the problems?*

- In practice, as we cannot guarantee global minima, the solutions will be different (we search over different spaces)
- While it is not easy to express mathematically the explicit prior $R(x)$, we can gain from the expressivity of $g(\theta)$

# Image restoration

## Optimization by parametrization

clean image $x$
corrupted image $\hat{x}$

$$\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$$

- We can consider $g$ as a prior by itself, e.g. $g$ expresses the world of natural images
- $g(\theta)$ acts as a prior, helps in selecting a good mapping which gives a desired output image, and prevents getting the wrong images
- It is sufficient to optimize over the data term only

$$\arg\min_{\theta} E(g(\theta); \hat{x})$$

# Deep image prior

$\hat{x}$ corrupted image

$$\arg\min_{\theta} E(g(\theta); \hat{x})$$

$$\boldsymbol{g}(\boldsymbol{\theta}) \equiv \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{z})$$

- $\boldsymbol{f}_{\boldsymbol{\theta}}$ is a convolutional neural network with parameters $\boldsymbol{\theta}$
- Drop the explicit regularization $R(x)$ and use instead the implicit prior captured by the neural network parametrization

- How do we map the parameters of the neural network to the image?
- Fix the input $z$ (e.g. noisy image)
- Unlike the common practice, i.e., fixing the weights and varying the input
- Here, we fix the input and vary the weights $\theta$, to get different outputs
- The convolutional neural network learns a generator $x = f_\theta(z)$ which maps random code $z$ to an image $x$

# Deep image prior

$\hat{x}$ corrupted image

1. **Initialize** $z$: Fill the input $z$ by uniform noise, or any other random image.
2. **Solve** by gradient descent

$$\arg\min_{\theta} E(f_\theta(z); \hat{x})$$

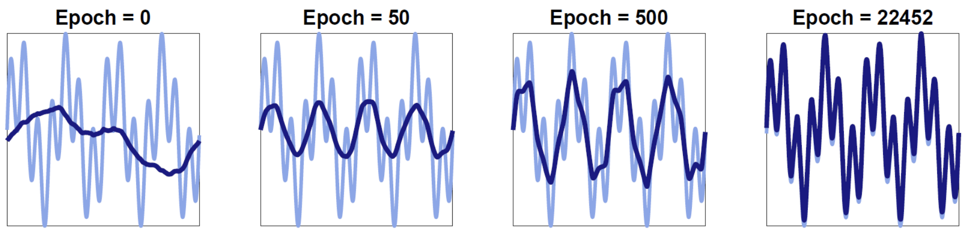
$$\theta^{k+1} = \theta^k - \alpha \frac{\partial E(f_\theta(z); \hat{x})}{\partial \theta}$$

**3. Get** the reconstructed image by forward passing

$$x^* = f_{\theta^*}(z)$$

# Deep image prior

## Inductive bias



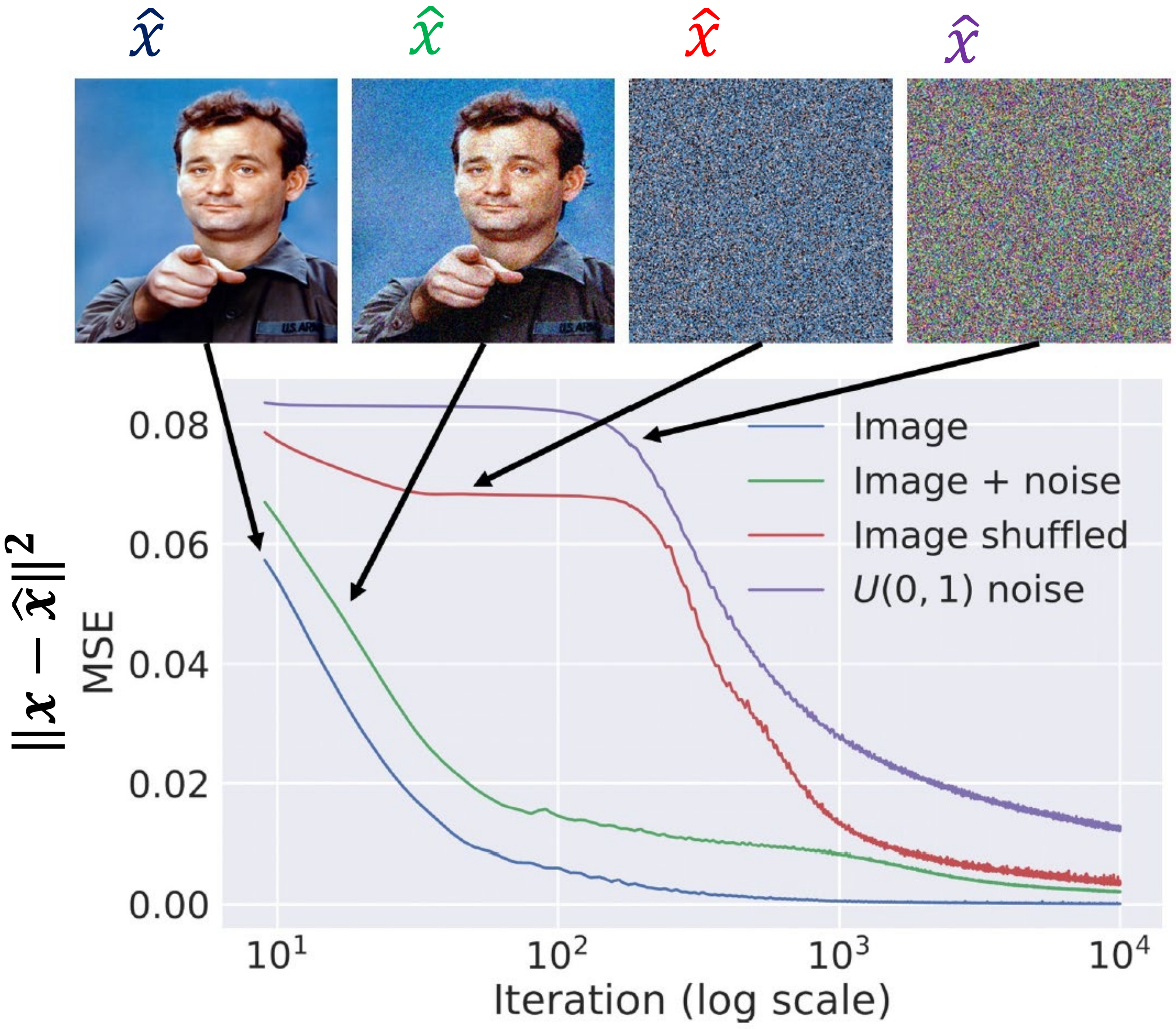

Ronen Basri, David Jacobs, Yoni Kasten, Shira Kritchman, NeurIPS 2019

### Spectral Bias

FC network fits the lower frequency component of the target function faster than the higher frequencies

Denoising $E(x, \hat{x}) = \|x - \hat{x}\|^2$

$$\arg\min_{\theta} E(f_\theta(z); \hat{x})$$



# Deep image prior

## Recap

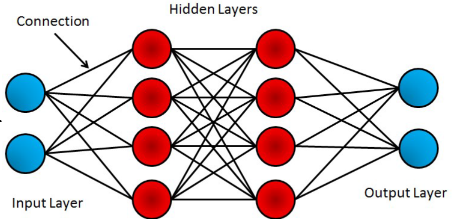
$x \rightarrow$ degradation $\rightarrow \hat{x} \rightarrow$ restoration $\rightarrow x^*$

$$x^* = \arg\max_x p(x|\hat{x}) = \arg\max_x p(\hat{x}|x)p(x) = \arg\min_x E(x;\hat{x}) + R(x)$$

$$\arg\min_\theta E(f_\theta(z);\hat{x})$$

Implicit regularization by parametrizing the restored image using CNN with parameters $\theta$

$z \rightarrow$



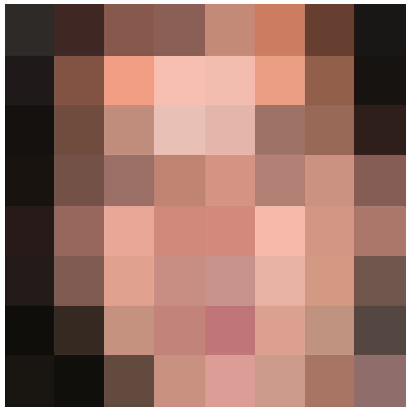

$\longrightarrow$

$x$

$f_\theta(z)$

$\rightarrow \min_x E(x, \hat{x})$

$\hat{x}$



# Deep image prior

Image restoration objectives

$$\arg\min_{\theta} E(f_\theta(z); \hat{x})$$

Denoising $E(x,\hat{x}) = \|x - \hat{x}\|^2$

clean image $x$

corrupted image $\hat{x}$

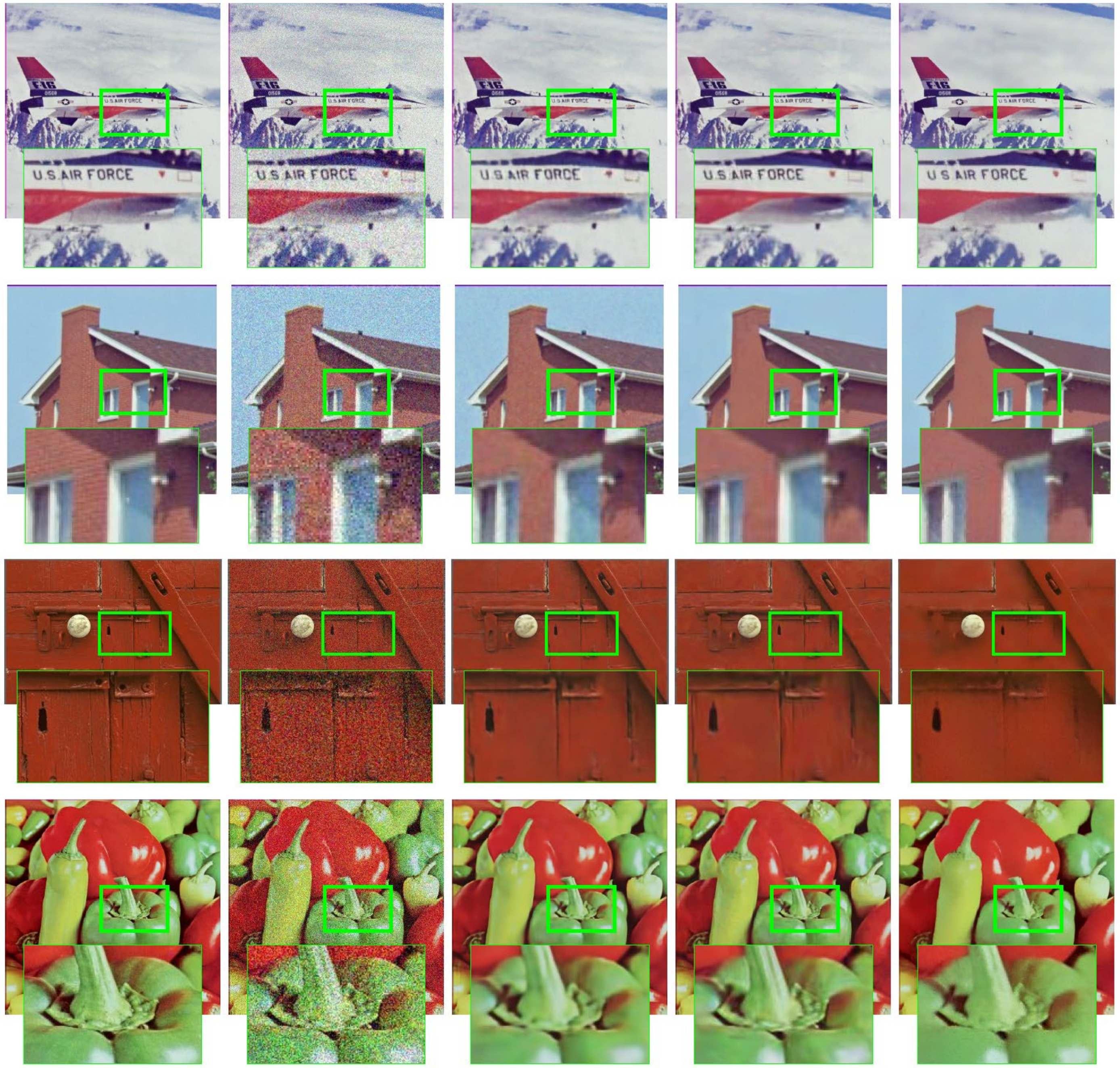
binary mask $m$

Inpainting $E(x,\hat{x}) = \|(x - \hat{x}) \odot m\|^2$

Super-resolution $E(x,\hat{x}) = \|d(x) - \hat{x}\|^2$

# Deep image prior

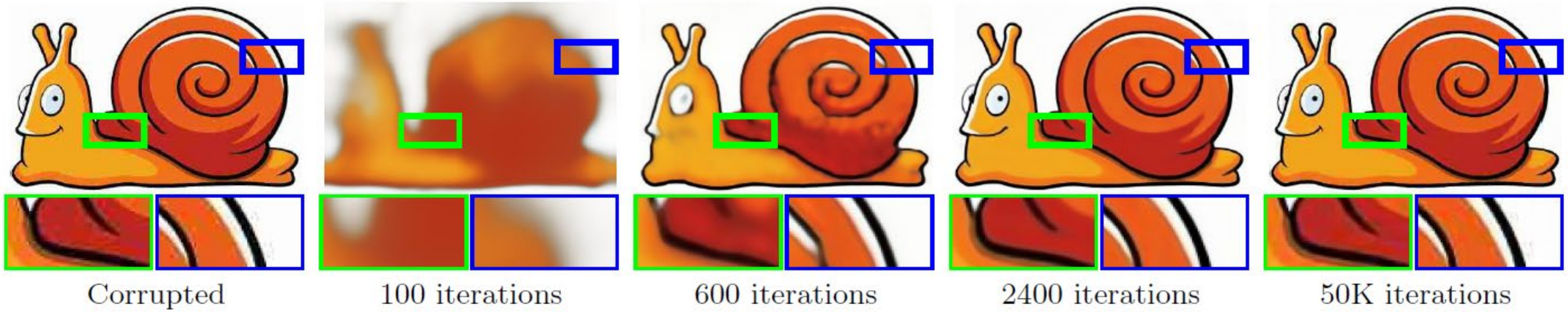
Denoising




(a) GT (b) Input (c) Ours (d) CBM3D (e) NLM

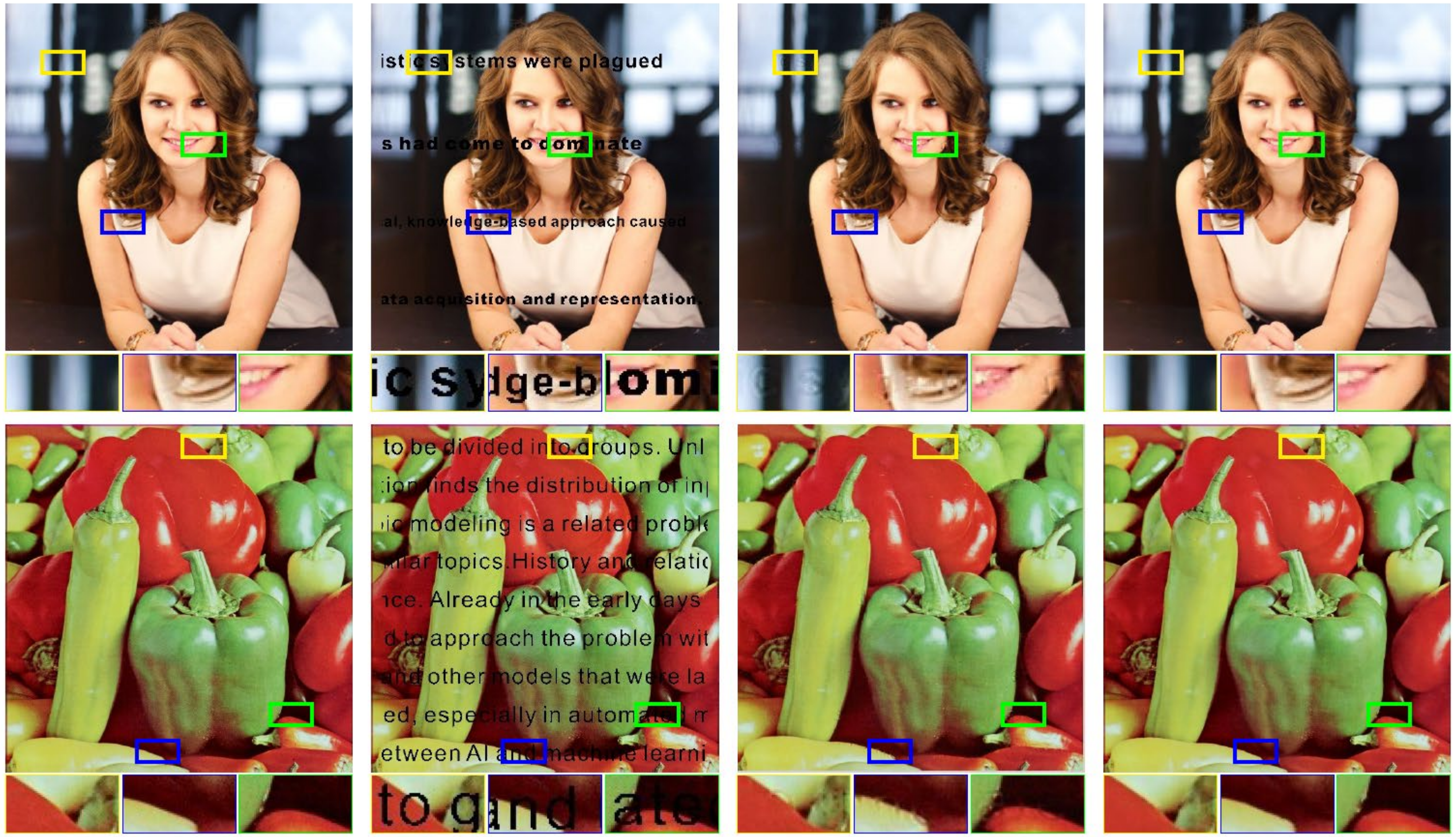
# Deep image prior

Jpeg artifact removal



# Deep image prior

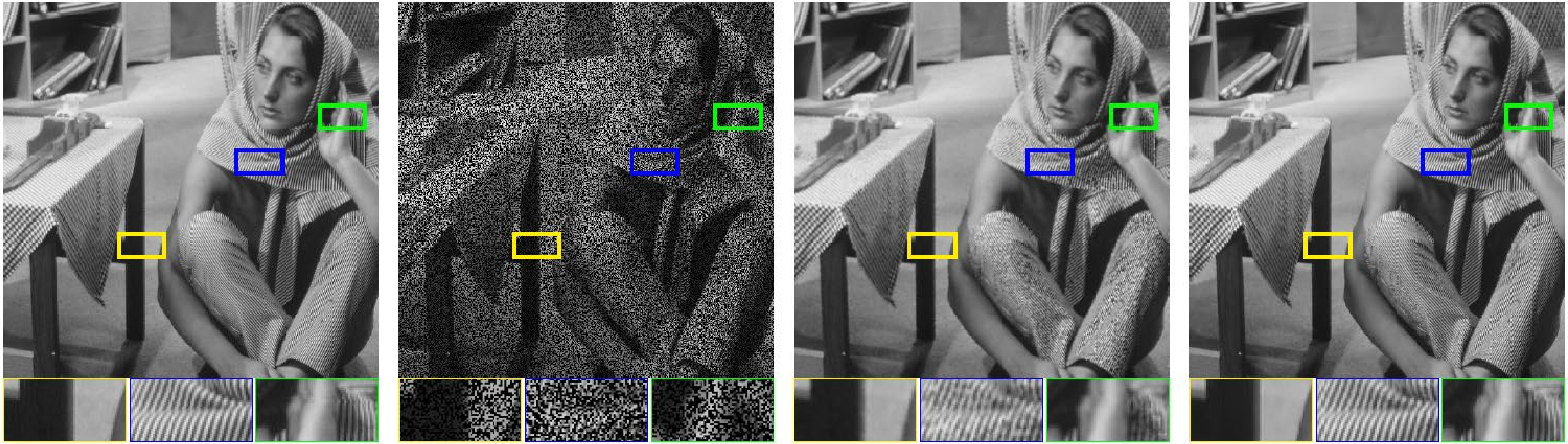
## Inpainting



(a) Original image (b) Corrupted image (c) Shepard networks [47] (d) Deep Image Prior

# Deep image prior

## Inpainting



(a) Original image (b) Corrupted image (c) CSC [44] (d) Deep image prior

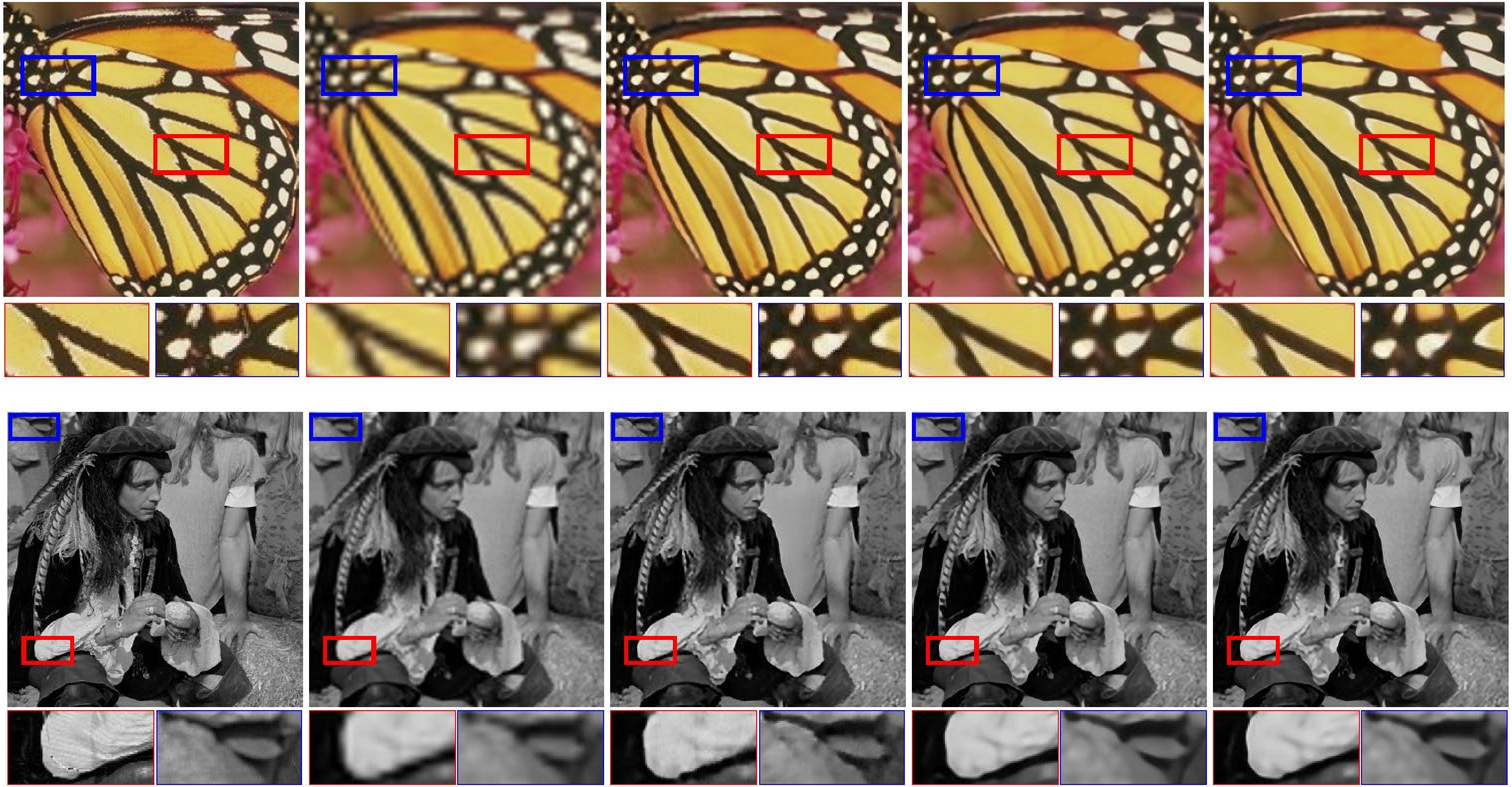
# Deep image prior

## Super-resolution

Deep Image Prior 9

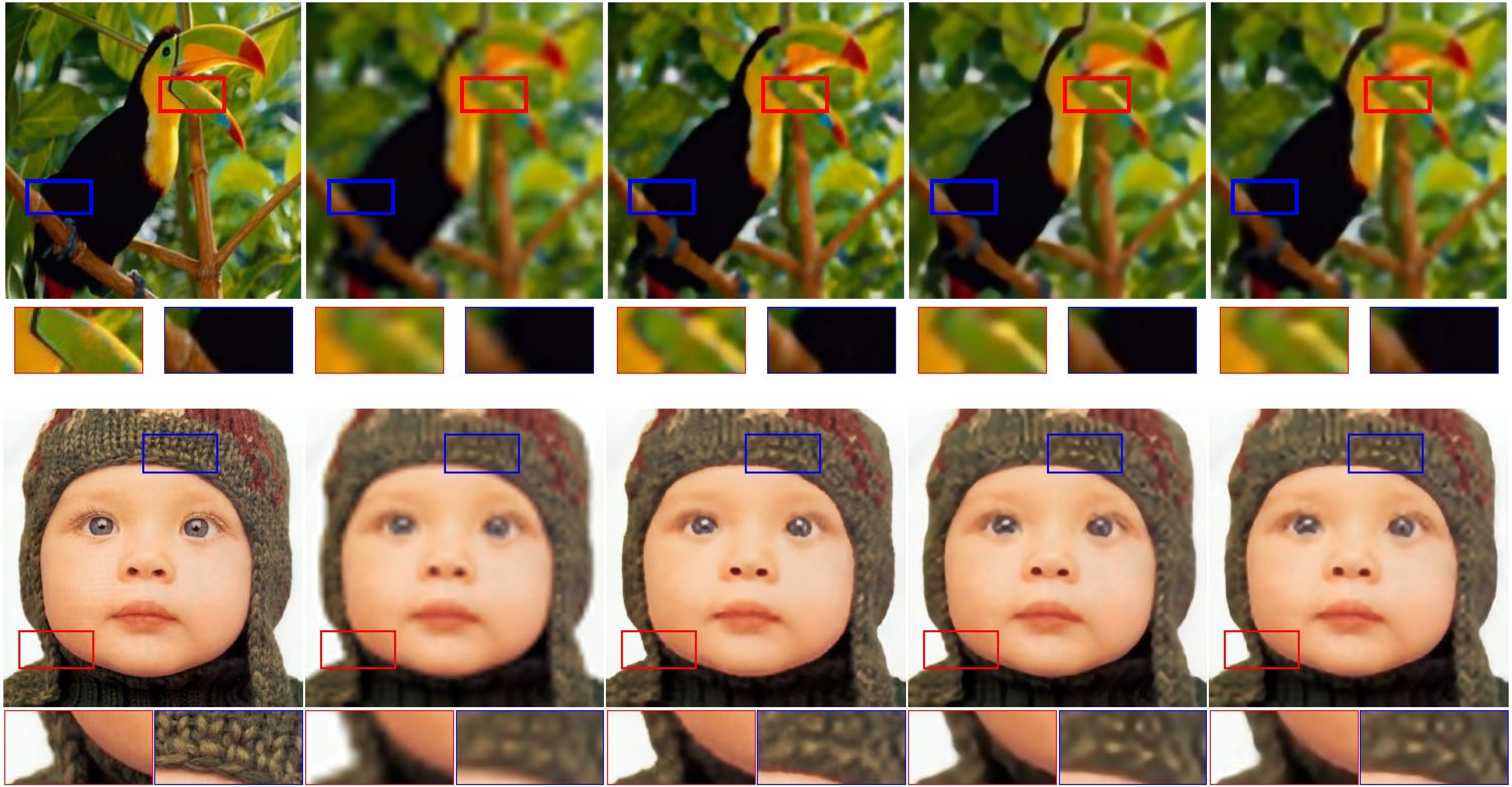
(a) HR image (b) Bicubic upsampling (c) No prior (d) TV prior (e) Deep image prior

4× super-resolution



(a) Original image | (b) Bicubic, **Not trained** | (c) Ours, **Not trained** | (d) LapSRN, **Trained** | (e) SRResNet, **Trained**

## 8× super-resolution



(f) Original image

(g) Bicubic, **Not trained**

(h) Ours, **Not trained**

(i) LapSRN, **Trained**

(j) VDSR, **Trained**

# Deep image prior

## Super-resolution

**4× super-resolution**

| | Baboon | Barbara | Bridge | Coastguard | Comic | Face | Flowers | Foreman | Lenna | Man | Monarch | Pepper | Ppt3 | Zebra | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No prior | 22.24 | 24.89 | 23.94 | 24.62 | 21.06 | 29.99 | 23.75 | 29.01 | 28.23 | 24.84 | 25.76 | 28.74 | 20.26 | 21.69 | 24.93 |
| Bicubic | **22.44** | 25.15 | 24.47 | 25.53 | 21.59 | **31.34** | 25.33 | 29.45 | 29.84 | 25.7 | 27.45 | 30.63 | 21.78 | 24.01 | 26.05 |
| TV prior | 22.34 | 24.78 | 24.46 | 25.78 | 21.95 | **31.34** | 25.91 | 30.63 | 29.76 | 25.94 | 28.46 | 31.32 | 22.75 | 24.52 | 26.42 |
| Glasner et al. | **22.44** | 25.38 | **24.73** | 25.38 | 21.98 | 31.09 | 25.54 | 30.40 | 30.48 | **26.33** | 28.22 | 32.02 | 22.16 | 24.34 | 26.46 |
| Ours | 22.29 | **25.53** | 24.38 | **25.81** | **22.18** | 31.02 | **26.14** | **31.66** | **30.83** | 26.09 | **29.98** | **32.08** | **24.38** | **25.71** | **27.00** |
| SRResNet-MSE | 23.0 | 26.08 | 25.52 | 26.31 | 23.44 | 32.71 | 28.13 | 33.8 | 32.42 | 27.43 | 32.85 | 34.28 | 26.56 | 26.95 | 28.53 |
| LapSRN | 22.83 | 25.69 | 25.36 | 26.21 | 22.9 | 32.62 | 27.54 | 33.59 | 31.98 | 27.27 | 31.62 | 33.88 | 25.36 | 26.98 | 28.13 |

**8× super-resolution**

| | Baboon | Barbara | Bridge | Coastguard | Comic | Face | Flowers | Foreman | Lenna | Man | Monarch | Pepper | Ppt3 | Zebra | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No prior | 21.09 | 23.04 | 21.78 | 23.63 | 18.65 | 27.84 | 21.05 | 25.62 | 25.42 | 22.54 | 22.91 | 25.34 | 18.15 | 18.85 | 22.56 |
| Bicubic | 21.28 | 23.44 | 22.24 | 23.65 | 19.25 | 28.79 | 22.06 | 25.37 | 26.27 | 23.06 | 23.18 | 26.55 | 18.62 | 19.59 | 23.09 |
| TV prior | 21.30 | 23.72 | **22.30** | 23.82 | 19.50 | 28.84 | 22.50 | 26.07 | 26.74 | 23.53 | 23.71 | 27.56 | 19.34 | 19.89 | 23.48 |
| SelfExSR | 21.37 | 23.90 | 22.28 | 24.17 | 19.79 | 29.48 | **22.93** | 27.01 | 27.72 | **23.83** | 24.02 | 28.63 | 20.09 | 20.25 | 23.96 |
| Ours | **21.38** | **23.94** | 22.20 | **24.21** | **19.86** | **29.52** | 22.86 | **27.87** | **27.93** | 23.57 | **24.86** | **29.18** | **20.12** | **20.62** | **24.15** |
| LapSRN | 21.51 | 24.21 | 22.77 | 24.10 | 20.06 | 29.85 | 23.31 | 28.13 | 28.22 | 24.20 | 24.97 | 29.22 | 20.13 | 20.28 | 24.35 |

Table 1: Detailed super-resolution PSNR comparison on the Set14 dataset with different scaling factors.

# Deep image prior

## Super-resolution

**4× super-resolution**

| | Baby | Bird | Butterfly | Head | Woman | **Avg.** |
|---|---|---|---|---|---|---|
| No prior | 30.16 | 27.67 | 19.82 | 29.98 | 25.18 | 26.56 |
| Bicubic | 31.78 | 30.2 | 22.13 | 31.34 | 26.75 | 28.44 |
| TV prior | 31.21 | 30.43 | 24.38 | 31.34 | 26.93 | 28.85 |
| Glasner et al. | **32.24** | 31.10 | 22.36 | **31.69** | 26.85 | 28.84 |
| Ours | 31.49 | **31.80** | **26.23** | 31.04 | **28.93** | **29.89** |
| LapSRN | 33.55 | 33.76 | 27.28 | 32.62 | 30.72 | 31.58 |
| SRResNet-MSE | 33.66 | 35.10 | 28.41 | 32.73 | 30.6 | 32.10 |

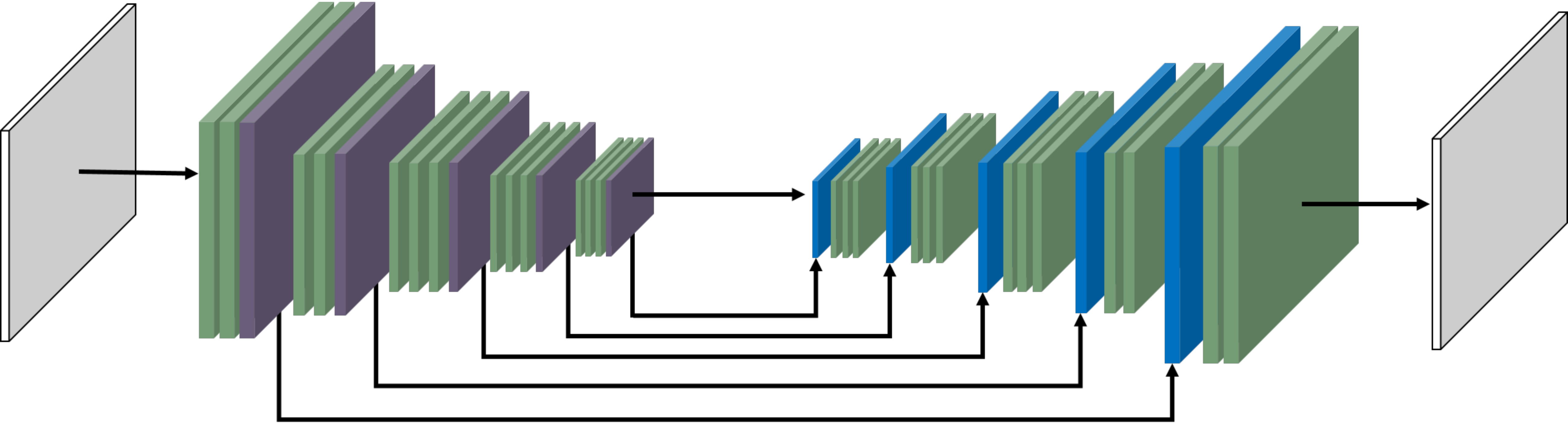
**8× super-resolution**

| | Baby | Bird | Butterfly | Head | Woman | **Avg.** |
|---|---|---|---|---|---|---|
| No prior | 26.28 | 24.03 | 17.64 | 27.94 | 21.37 | 23.45 |
| Bicubic | 27.28 | 25.28 | 17.74 | 28.82 | 22.74 | 24.37 |
| TV prior | 27.93 | 25.82 | 18.40 | 28.87 | 23.36 | 24.87 |
| SelfExSR | **28.45** | 26.48 | 18.80 | 29.36 | 24.05 | 25.42 |
| Ours | 28.28 | **27.09** | **20.02** | **29.55** | **24.50** | **25.88** |
| LapSRN | 28.88 | 27.10 | 19.97 | 29.76 | 24.79 | 26.10 |

Table 2: Detailed super-resolution PSNR comparison on the Set5 dataset with different scaling factors.

# Deep image prior

Encode-decoder architecture



# Deep image prior

## Depths and architectures

(a) Input (white=masked)
(b) Encoder-decoder, depth=6
(c) Encoder-decoder, depth=4
(d) Encoder-decoder, depth=2
(e) ResNet, depth=8
(f) U-net, depth=5

# Conclusions

- The success of deep neural networks is often attributed to ability to learn image prior using large databases

- In “Deep image prior”   it is shown that the structure of the generator network is sufficient to capture low-level image statistics prior, for image restoration tasks, without any learning

- The structure of the network imposes a strong prior

- Limitations: slowness, generalization (not necessarily SOTA)

# Deep image prior
# → Computer vision, Computer Graphics

The structure of the network allows

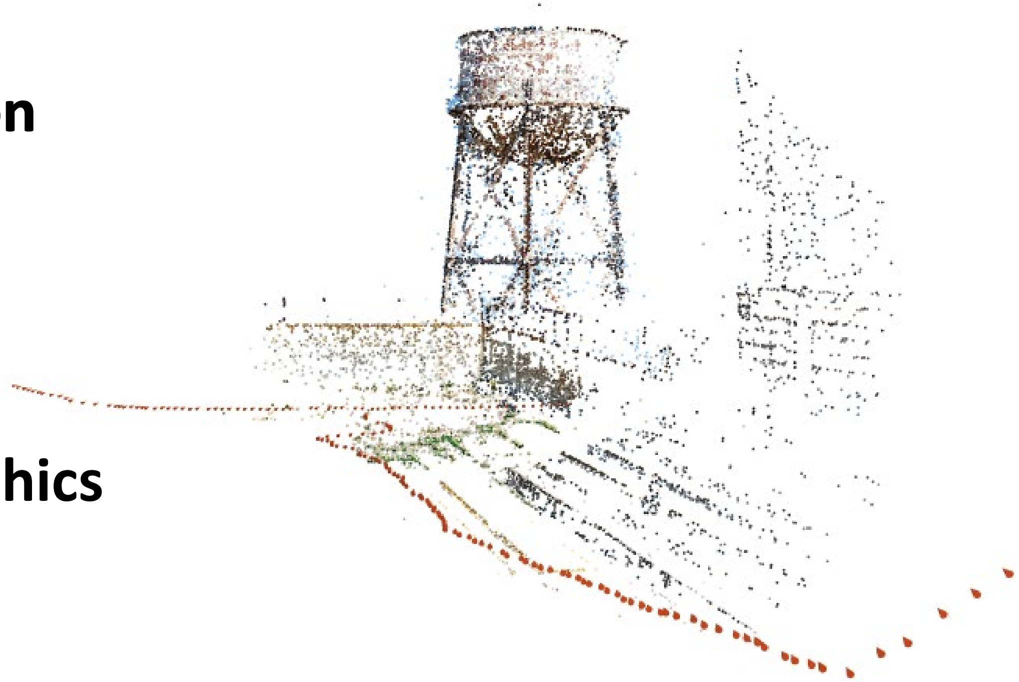
Parametrizing signals by the net weights

1. 2D images
2. 3D volumes
3. Continuous functions

# Geometry in computer vision

**Computer vision**

**Computer graphics**



**Geometry**
**3D**

**Images**
**2D**

**Structure from motion problem**

Moran*,Koslowsky*, Kasten, Maron, Galun, Basri, ICCV, 2021

# Computer graphics
# and
# Rendering

Based on

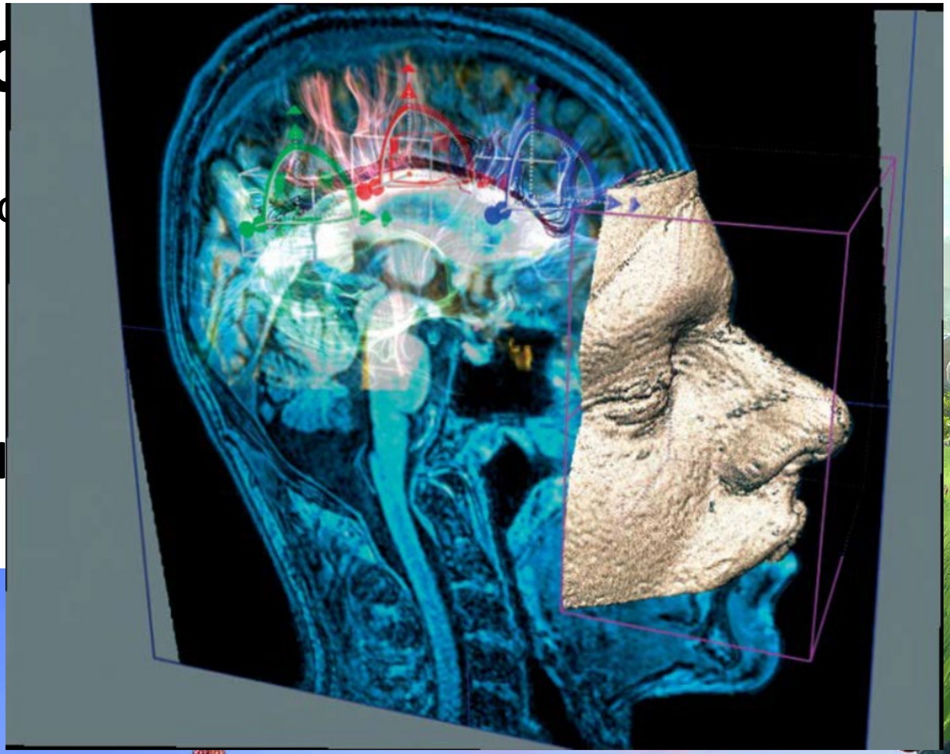
1. Keenan Crane’s course on Computer Graphics, CMU 15-462/662
2. The book Computer Graphics: principles and practice by Foley
3. The ECCV 2022 Tutorial Neural Volumetric Rendering for Computer Vision

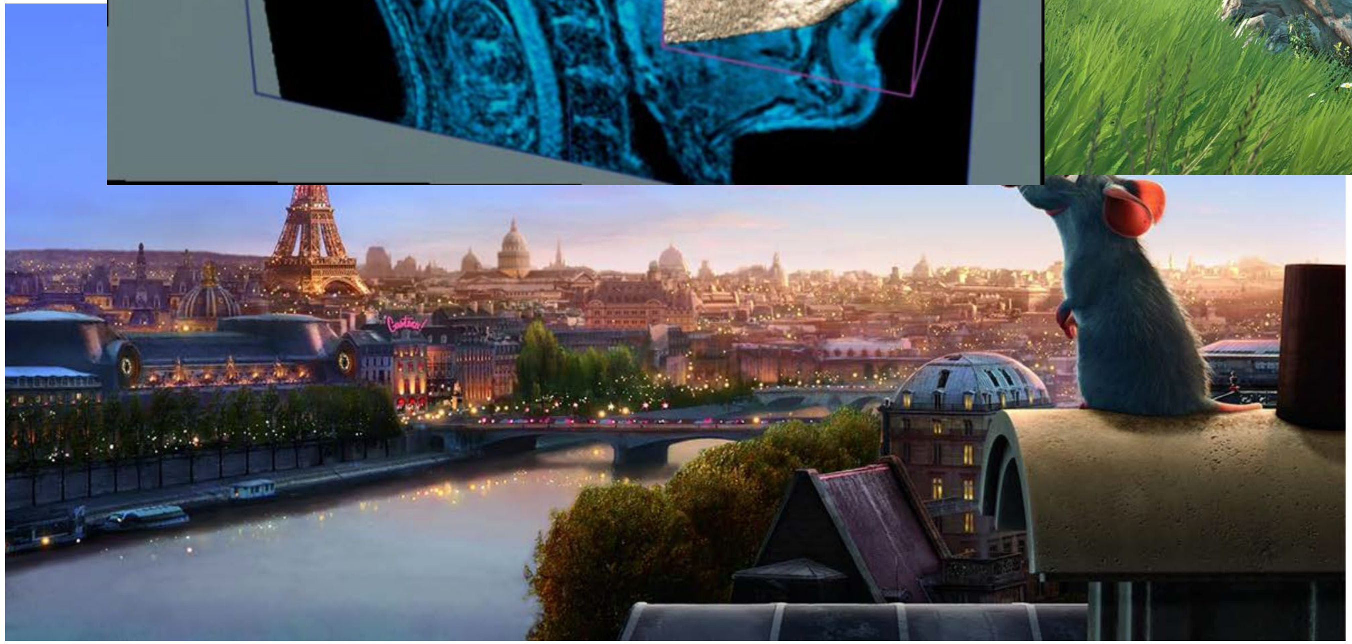
WAIC

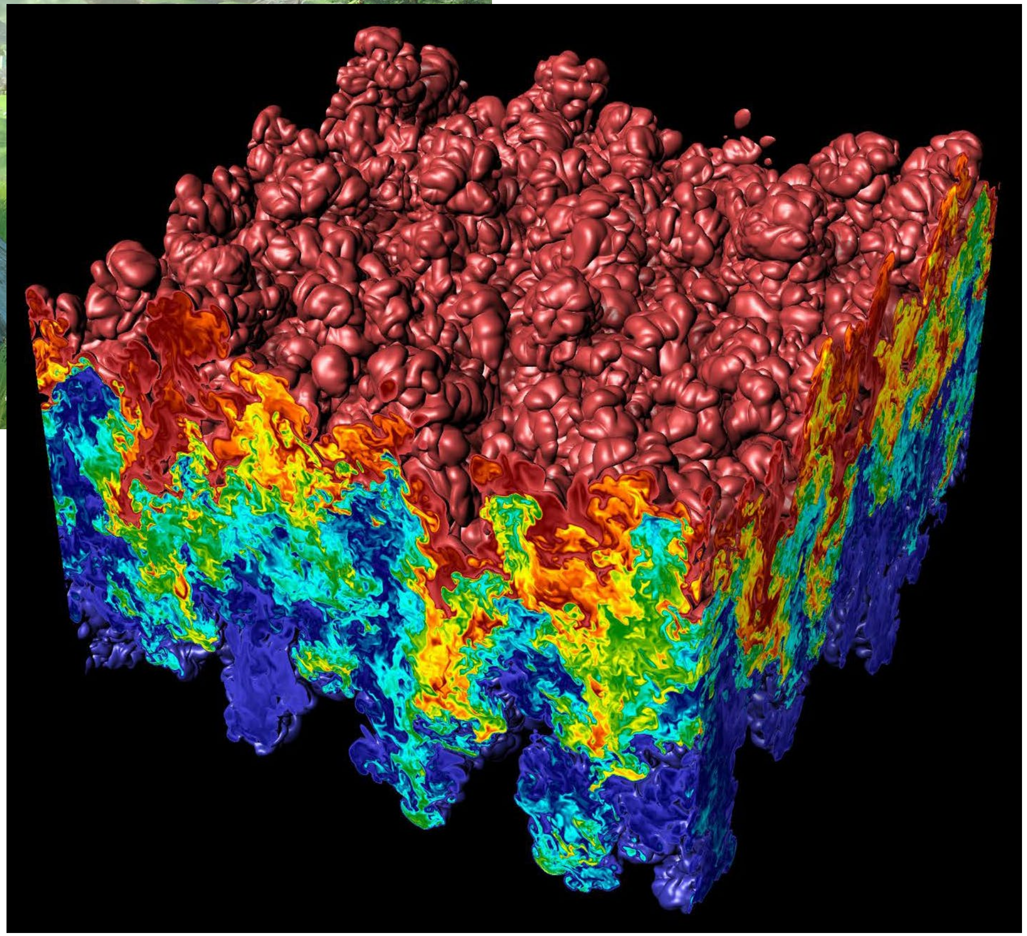
# Co

Co

**Cl**

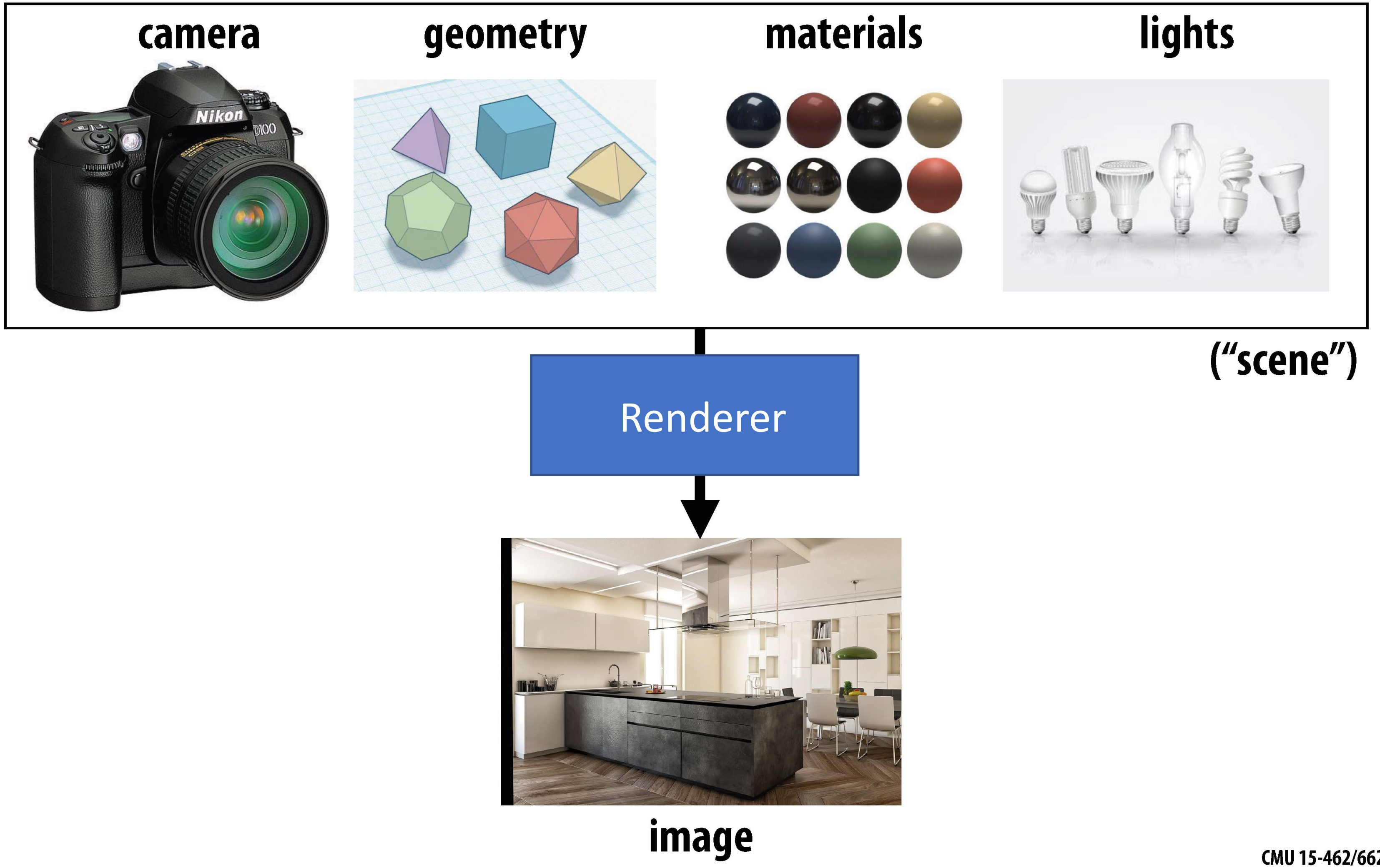




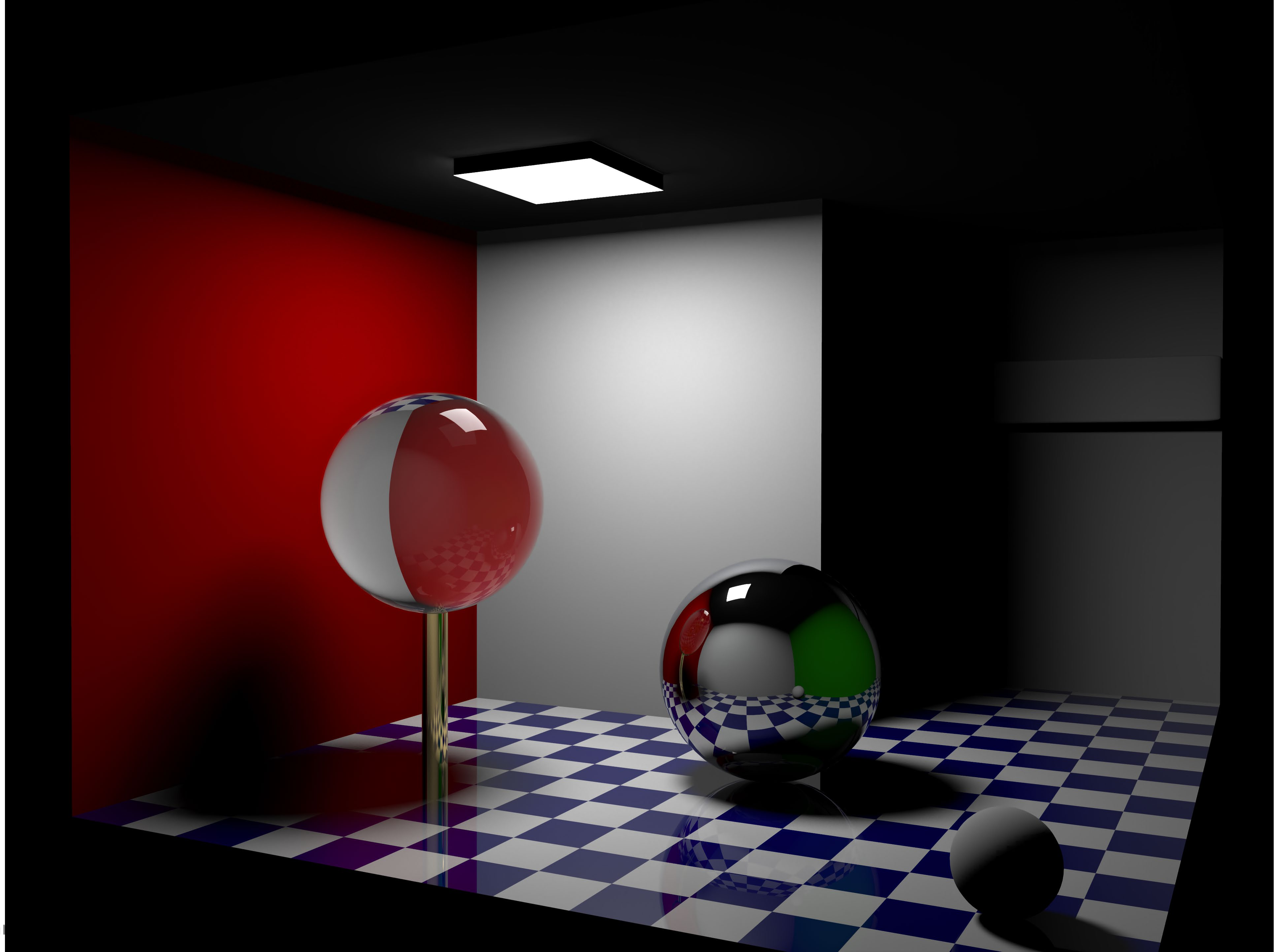


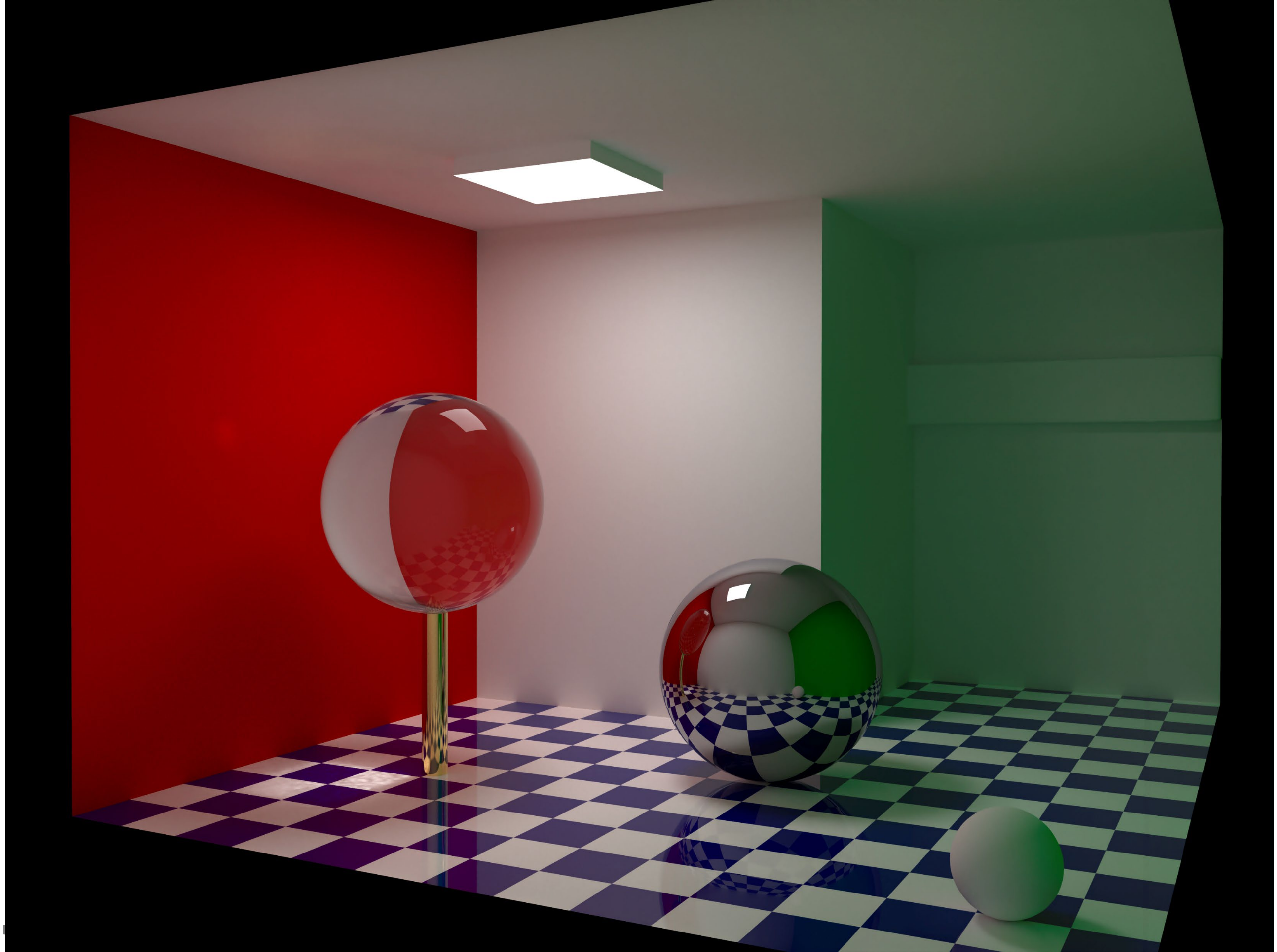
# Rendering

The process of generating a photorealistic image from a 3D model




CMU 15-462/662



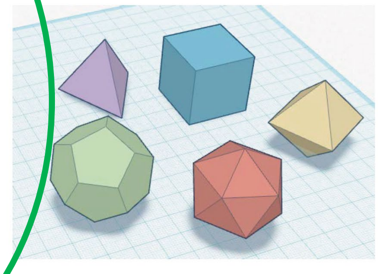


# Photorealistic Rendering—Basic Goal

## What are the INPUTS and OUTPUTS?

camera


geometry



materials

lights

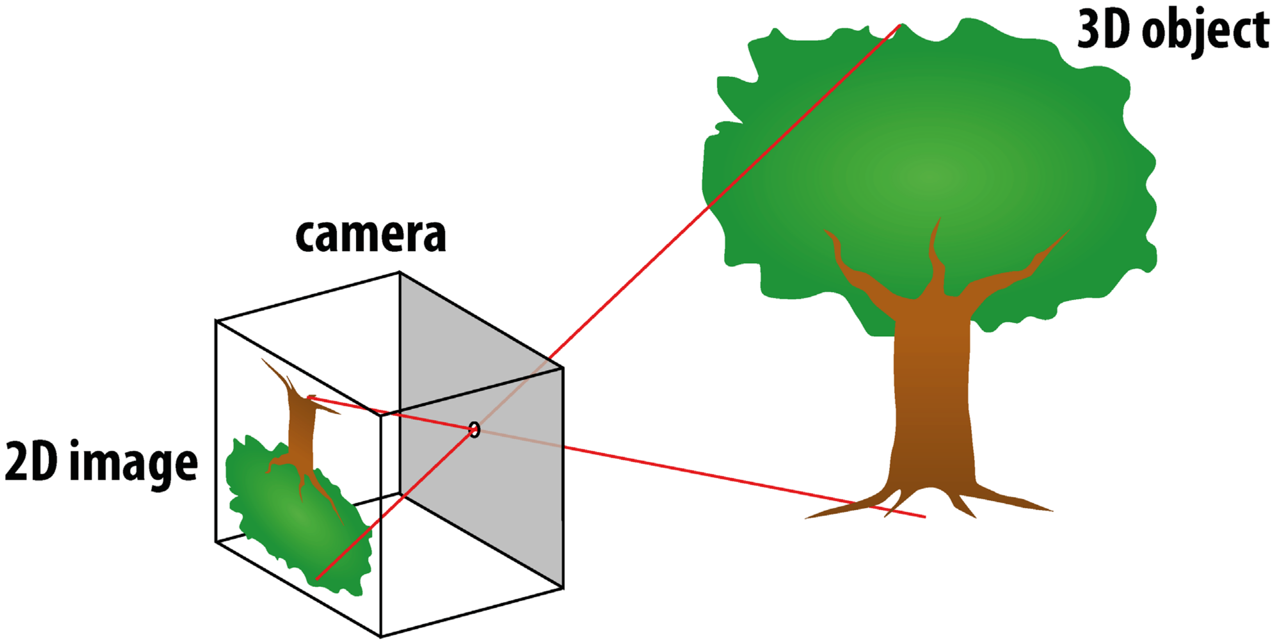
("scene")

Rendering

image

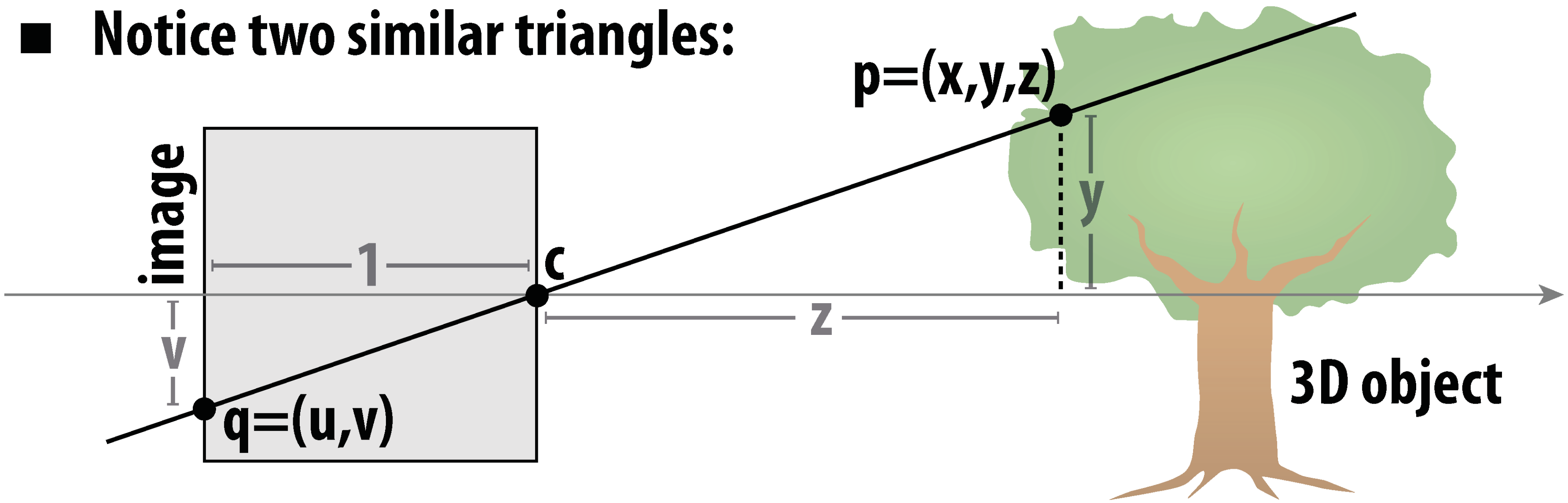
# **Perspective projection**
# Pinhole camera model

- Objects look smaller as they get further away
- Parallel lines “meet”  at infinity



# Perspective projection

## Pinhole camera model



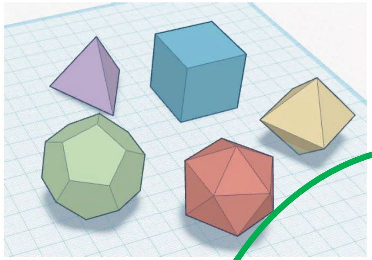

- Camera pinhole at $c = (0,0,0)$
- The image plane located $z = -1$
- Using similar triangles $v = \frac{y}{z}$ and $u = \frac{x}{z}$

# Photorealistic Rendering—Basic Goal

## What are the INPUTS and OUTPUTS?

**camera**

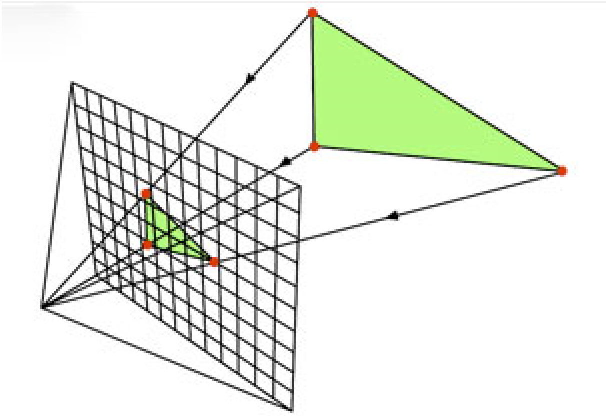

**geometry**



**materials**

**lights**

**(“scene”)**



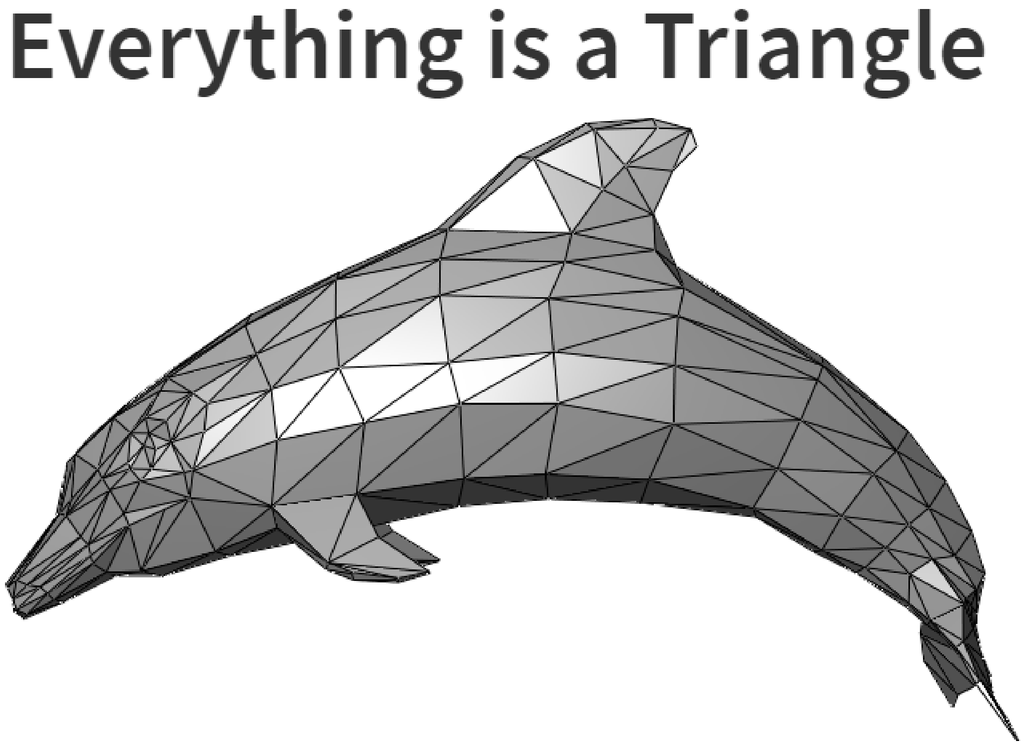
Rendering

**image**

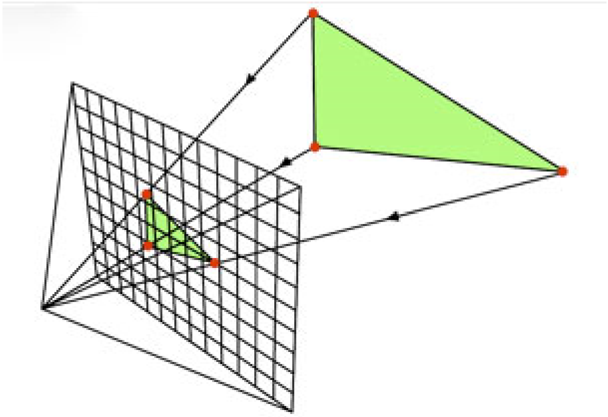
# Rendering

## Drawing on the screen (3D→2D)

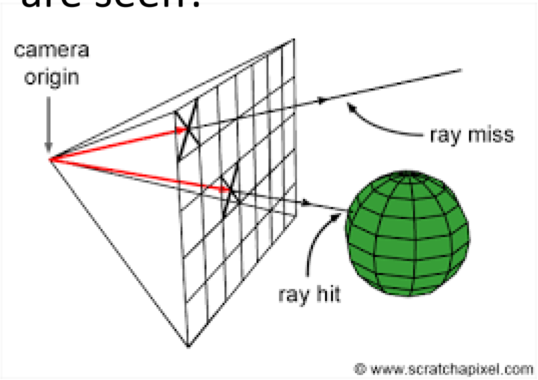
Two ways of turning triangles into image
- Rasterization
- Ray tracing




**Rasterization**
for each primitive (triangle),
which pixels are covered?



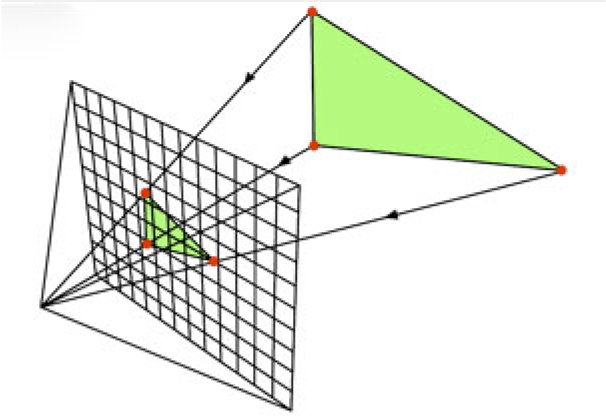
**Ray tracing**
for each pixel,
which primitives (triangles)
are seen?



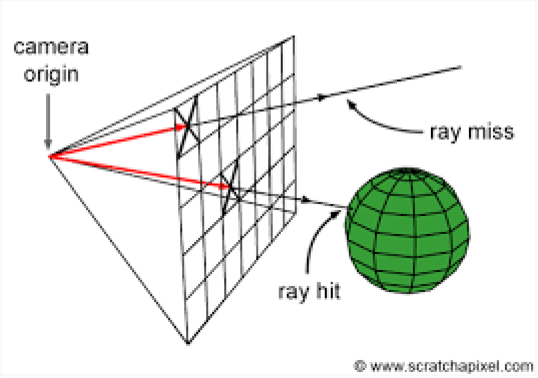
# Rendering

## Drawing on the screen (3D→2D)

Rasterization
- for each primitive (triangle), which pixels are covered?
- extremely fast (Billions of triangles per second on GPU)
- harder (but possible) to achieve photorealism
- games and real-time applications



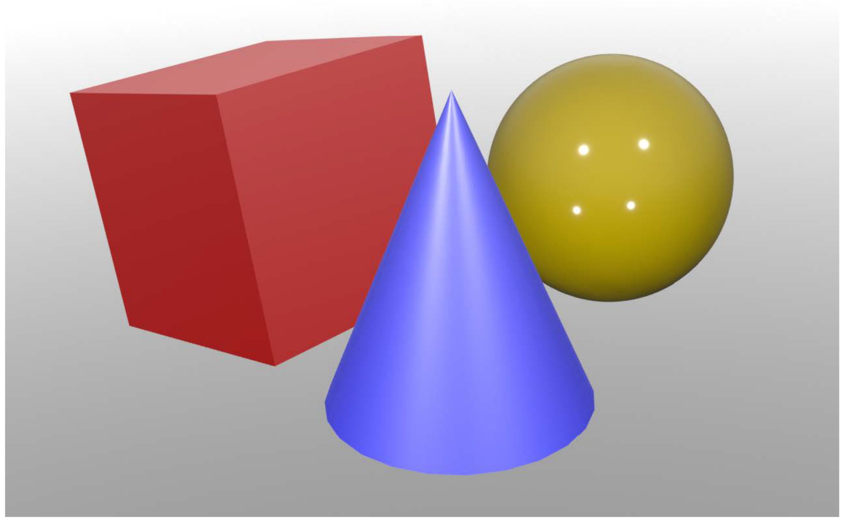
Ray tracing
- for each pixel, which primitives (triangles) are seen?
- generally slower
- easier to get photorealism
- movies and video clips



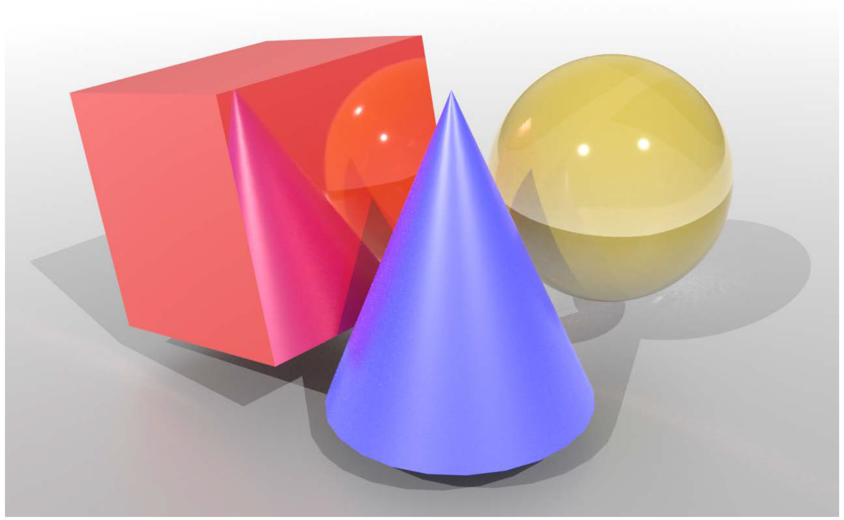
# Ray Tracing vs. Rasterization—Illumination

- More major difference: sophistication of illumination model
  - [LOCAL] rasterizer processes one *primitive* at a time; hard* to determine things like "A is in the shadow of B"
  - [GLOBAL] ray tracer processes on *ray* at a time; ray knows about everything it intersects, easy to talk about shadows & other "global" illumination effects

RASTERIZATION



RAY TRACING

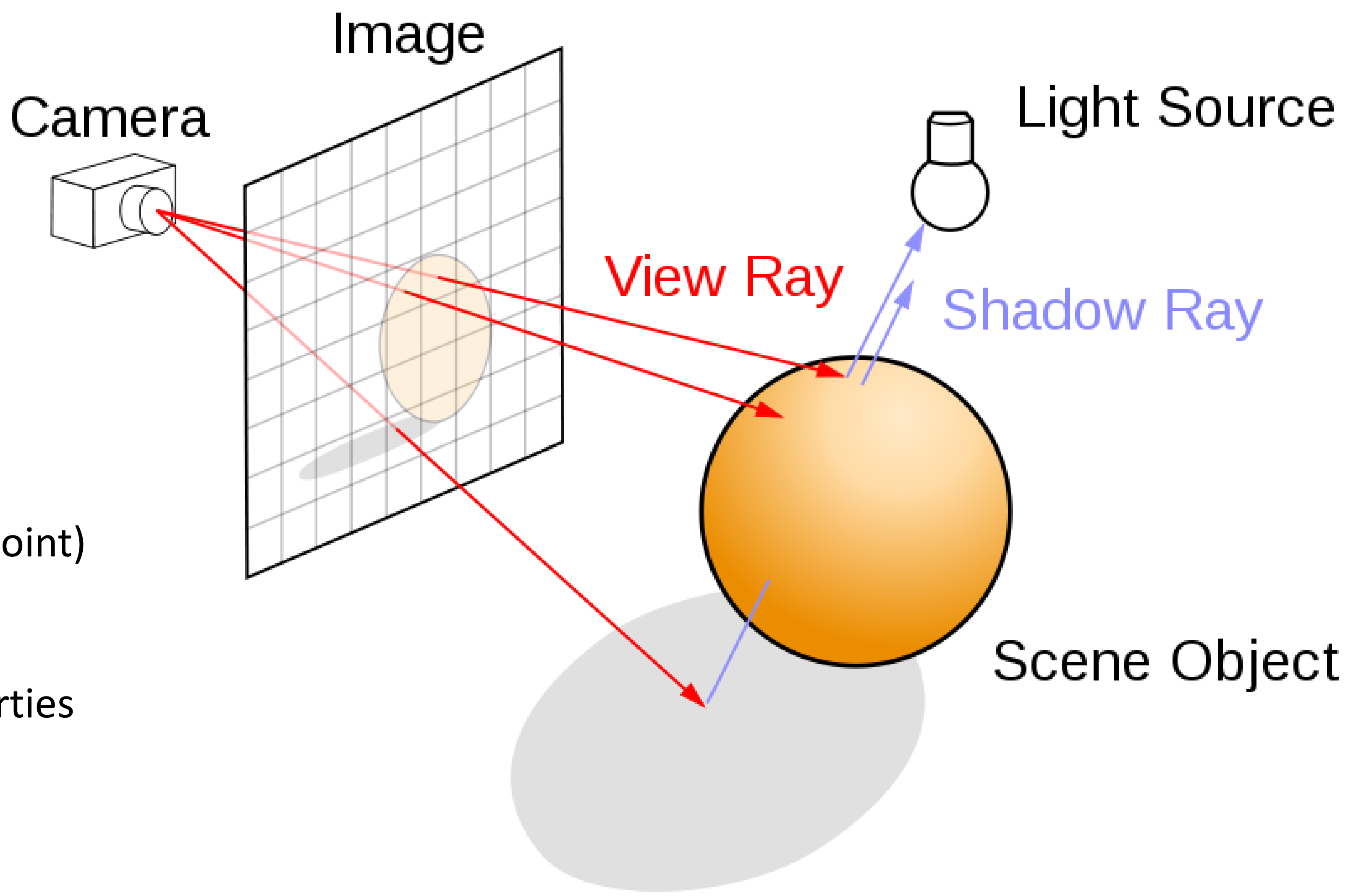


**Q: What illumination effects are missing from the image on the left?**

*But not *impossible* to do some things with rasterization (e.g., shadow maps)… just results in more complexity

# Rendering

## Drawing on the screen



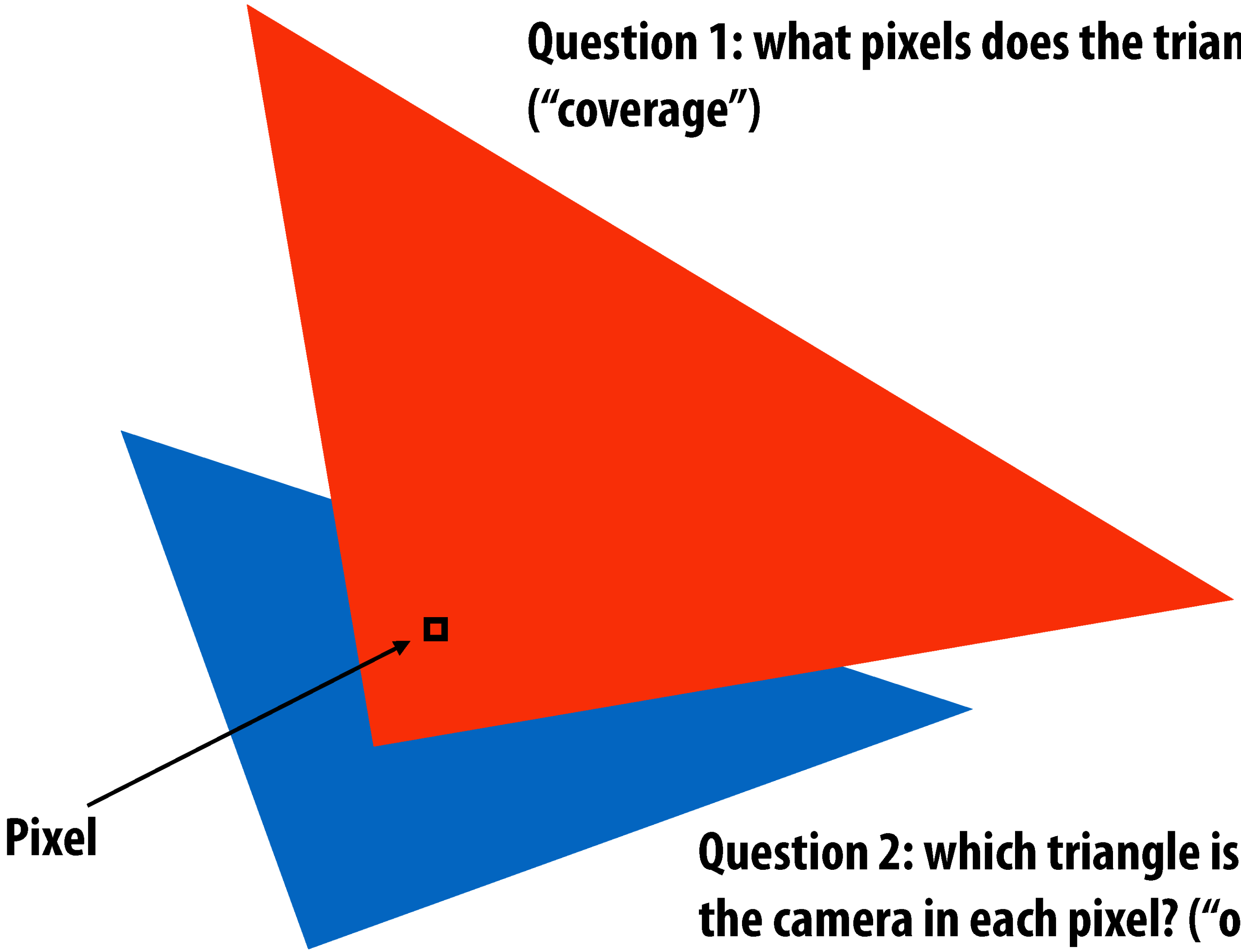

- camera (view point)
- light sources
- geometry
- material properties

# Rendering

## The visibility problem

## Rasterization

**Question 1: what pixels does the triangle overlap? ("coverage")**



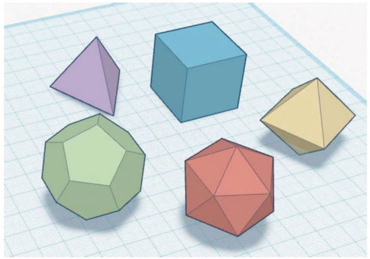

**Question 2: which triangle is closest to the camera in each pixel? ("occlusion")**

# Photorealistic Rendering—Basic Goal

## What are the INPUTS and OUTPUTS?

camera


geometry



materials

lights

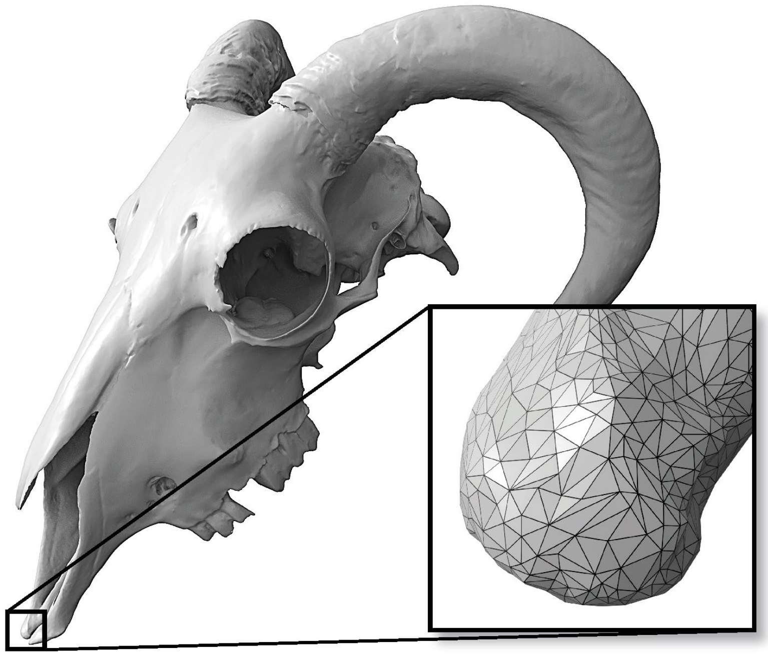
("scene")

Rendering

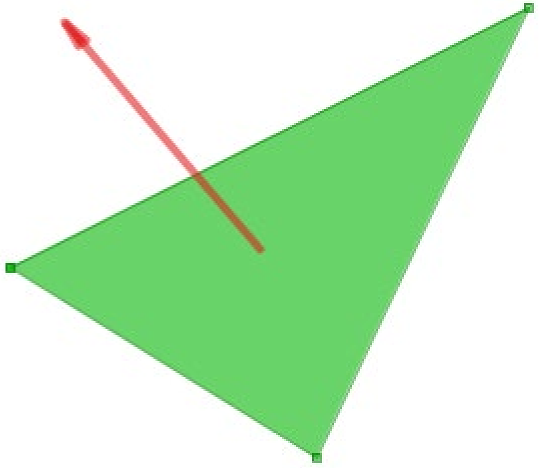
image

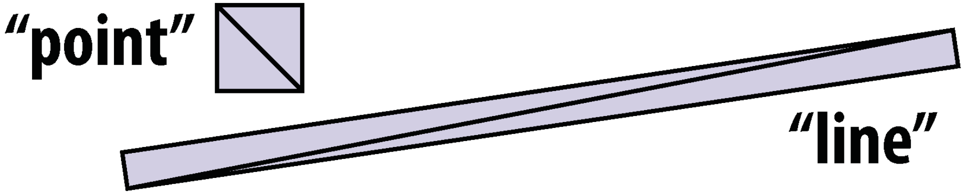
CMU 15-462/662

# Geometry
## Why triangles?



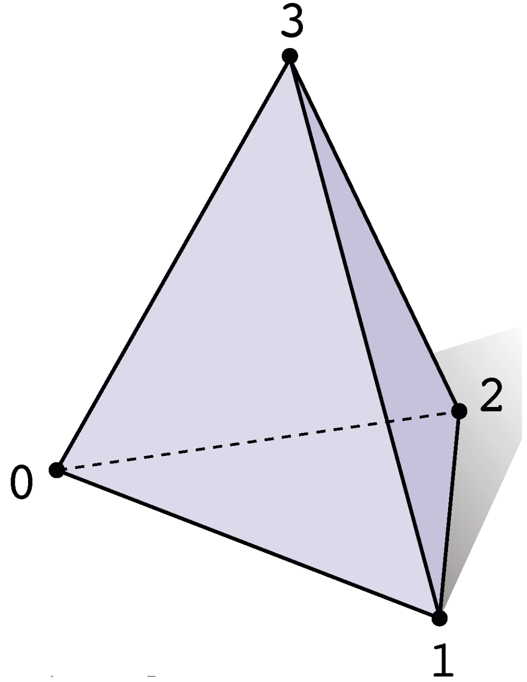
- can approximate any shape
- always planar, well-defined normal
- easy to interpolate data, using “barycentric coordinates”
- optimized and uniform drawing pipeline





# Geometry
## Triangle mesh (explicit)

- store vertices as triplets of coordinates $(x, y, z)$
- store triangles as triplets of indices $(i, j, k)$




- **E.g., tetrahedron:**

**VERTICES**

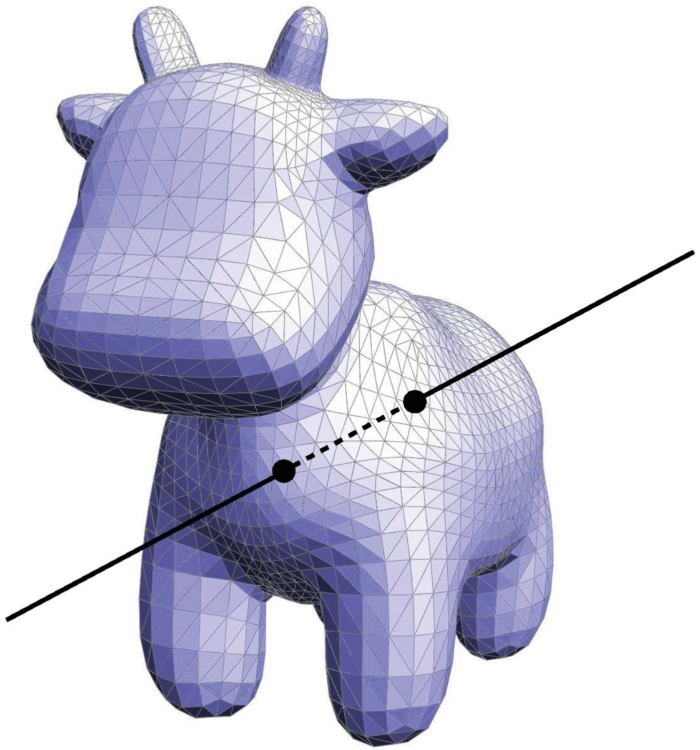
| | x | y | z |
|---|---|---|---|
| 0: | –1 | –1 | –1 |
| 1: | 1 | –1 | 1 |
| 2: | 1 | 1 | –1 |
| 3: | –1 | 1 | 1 |

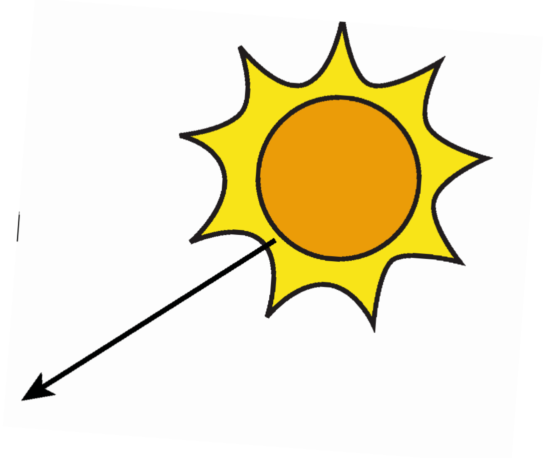
**TRIANGLES**

| i | j | k |
|---|---|---|
| 0 | 2 | 1 |
| 0 | 3 | 2 |
| 3 | 0 | 1 |
| 3 | 1 | 2 |

# **Geometry**

# Ray-mesh intersection





- Think about a ray of light traveling from the sun
- Want to know where a ray pierces a surface
- Might pierce surface in many places
- A significant step towards visibility and ray tracing

# Geometry

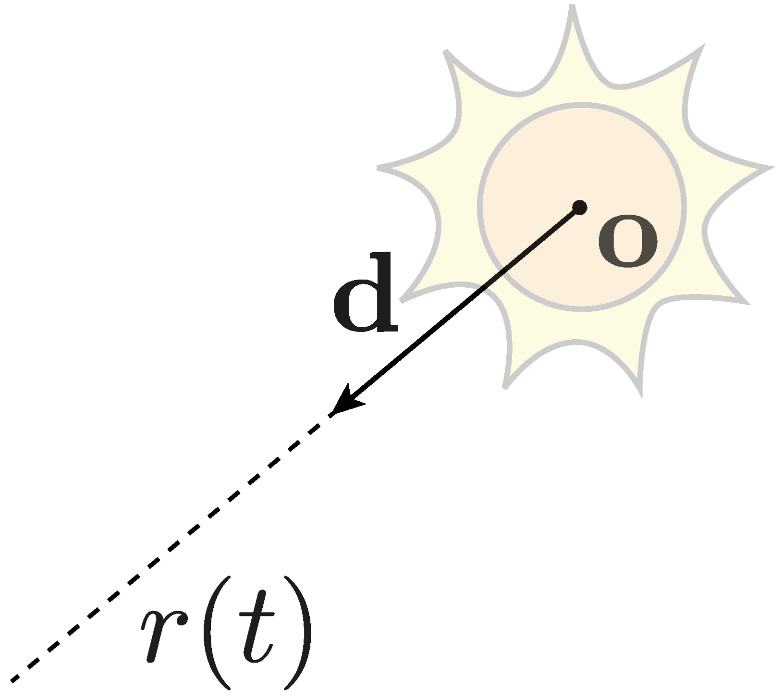
## Ray equation

Ray source

Unit direction

$$r(t) = o + td$$

Point along ray parametrized by $t$



# Geometry

## Ray-plane intersection

Intersection between

plane $N^T x = c$

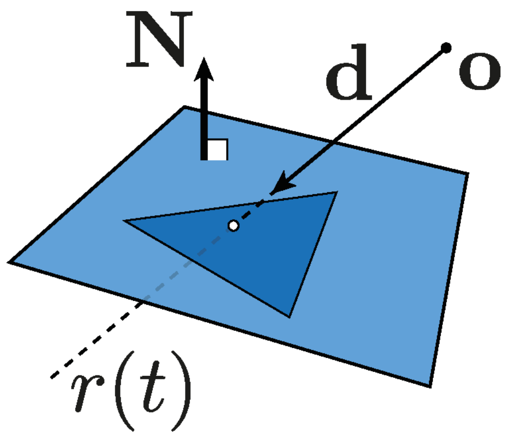
and

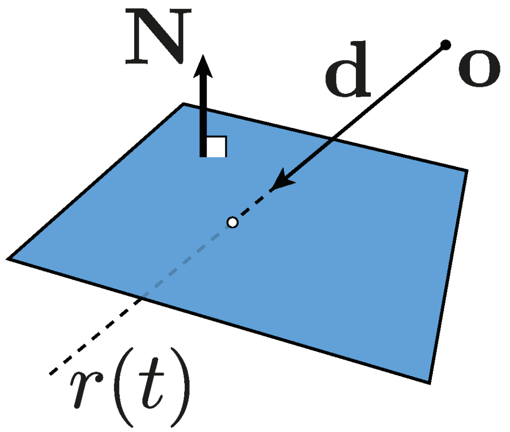
ray $r(t) = o + td$

$$N^T r(t) = c$$

$$N^T(o + td) = c \Rightarrow t = \frac{c - N^T o}{N^T d}$$

$$r(t) = o + \frac{c - N^T o}{N^T d} d$$



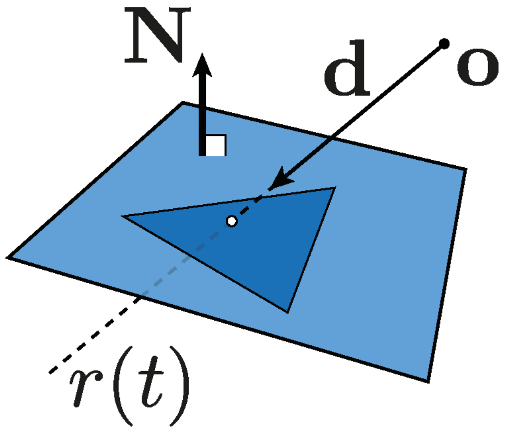




$N$ unit normal

$C$ offset

# Geometry

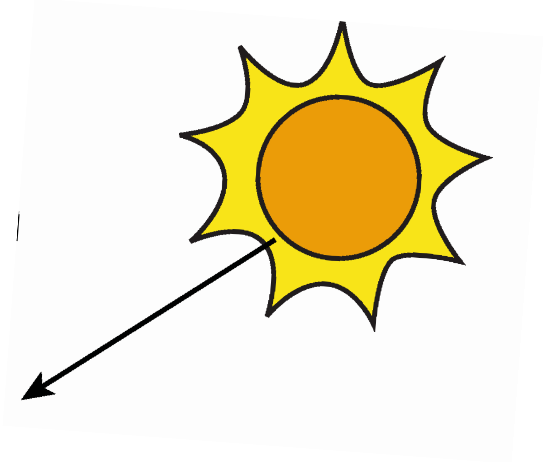
## Ray-triangle intersection

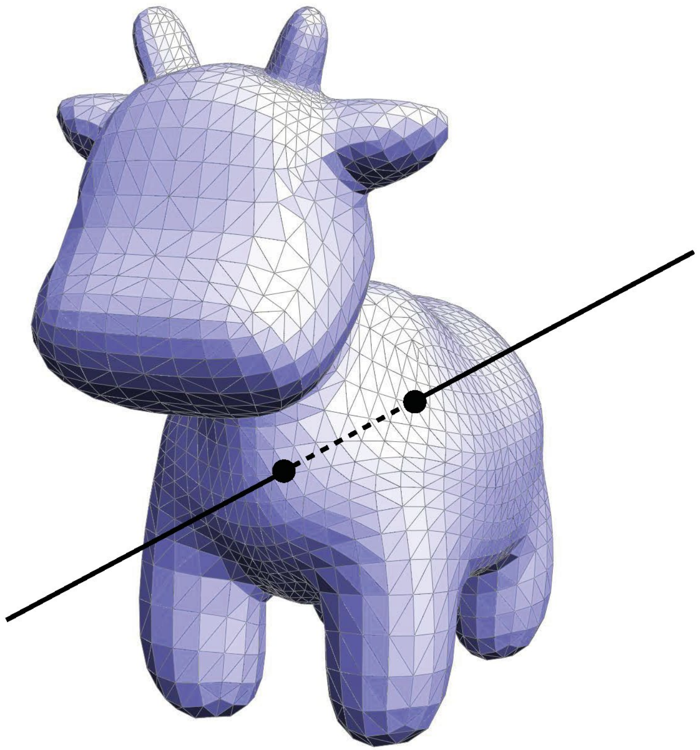



- need to determine if  point of intersection is within the triangle
- compute ray-plane intersection
- compute barycentric coordinates of hit point
- if all barycentric coordinates are positive, point in triangle

# Geometry

## Ray-mesh intersection





Challenges in performance

- How to accelerate the naïve algorithm, given a ray, scan all triangles
- There are a lot of triangles and a lot of rays
- By hierarchical approach and dedicated hardware

# Why care about performance?

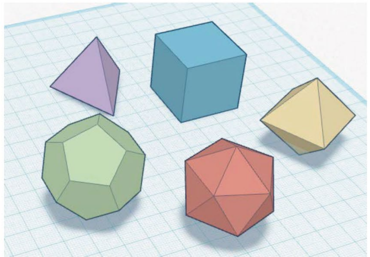
**Pixar's "Coco" — about 50 <u>hours</u> per frame (@24 frames/sec)**

# Photorealistic Rendering—Basic Goal

## What are the INPUTS and OUTPUTS?

camera


geometry



materials

lights

("scene")

Rendering

image

# Radiometry = measuring light

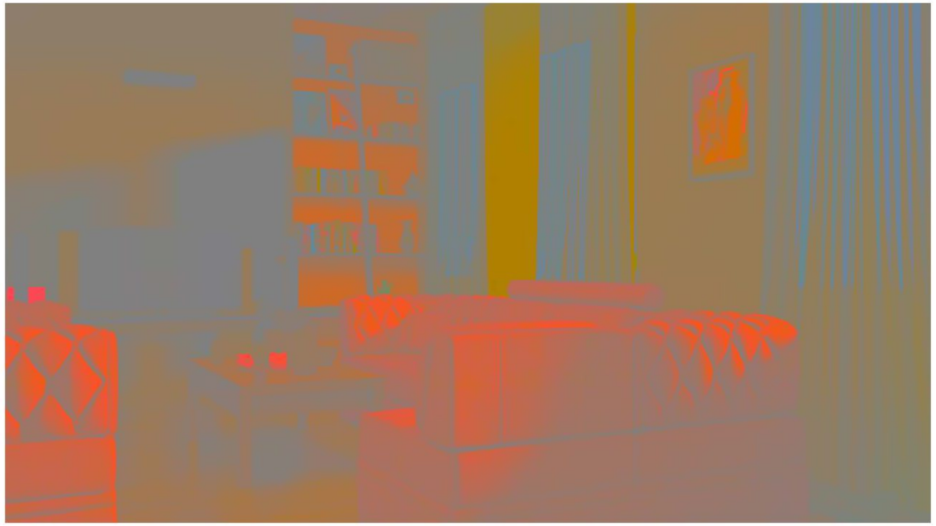
Aim: Photo realistic images

- Which color at each pixel?
- How much light (illumination) at each pixel?
- Why some parts of the surface look lighter or darker?
- Final image = at every point, what color and how intense or bright it is

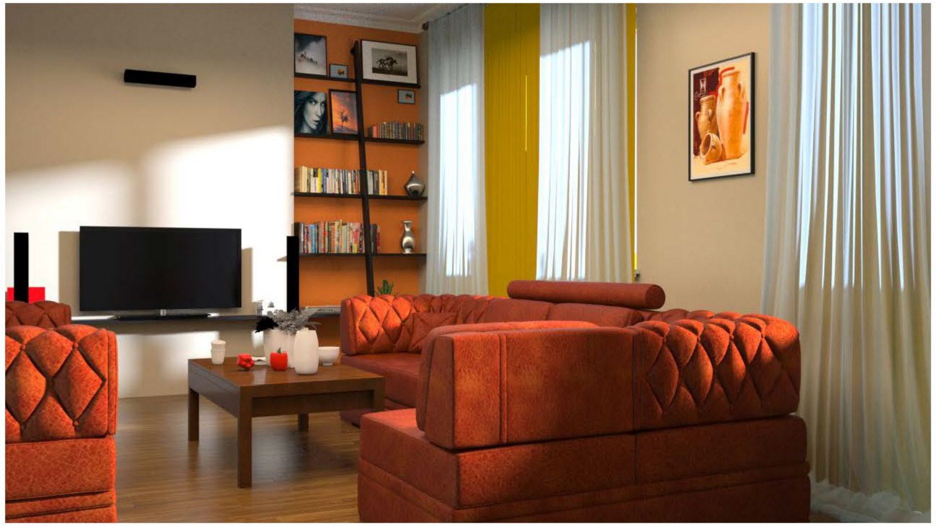
# Rendering is more than just color!

- Also need to know ***how much*** light hits each pixel:

color

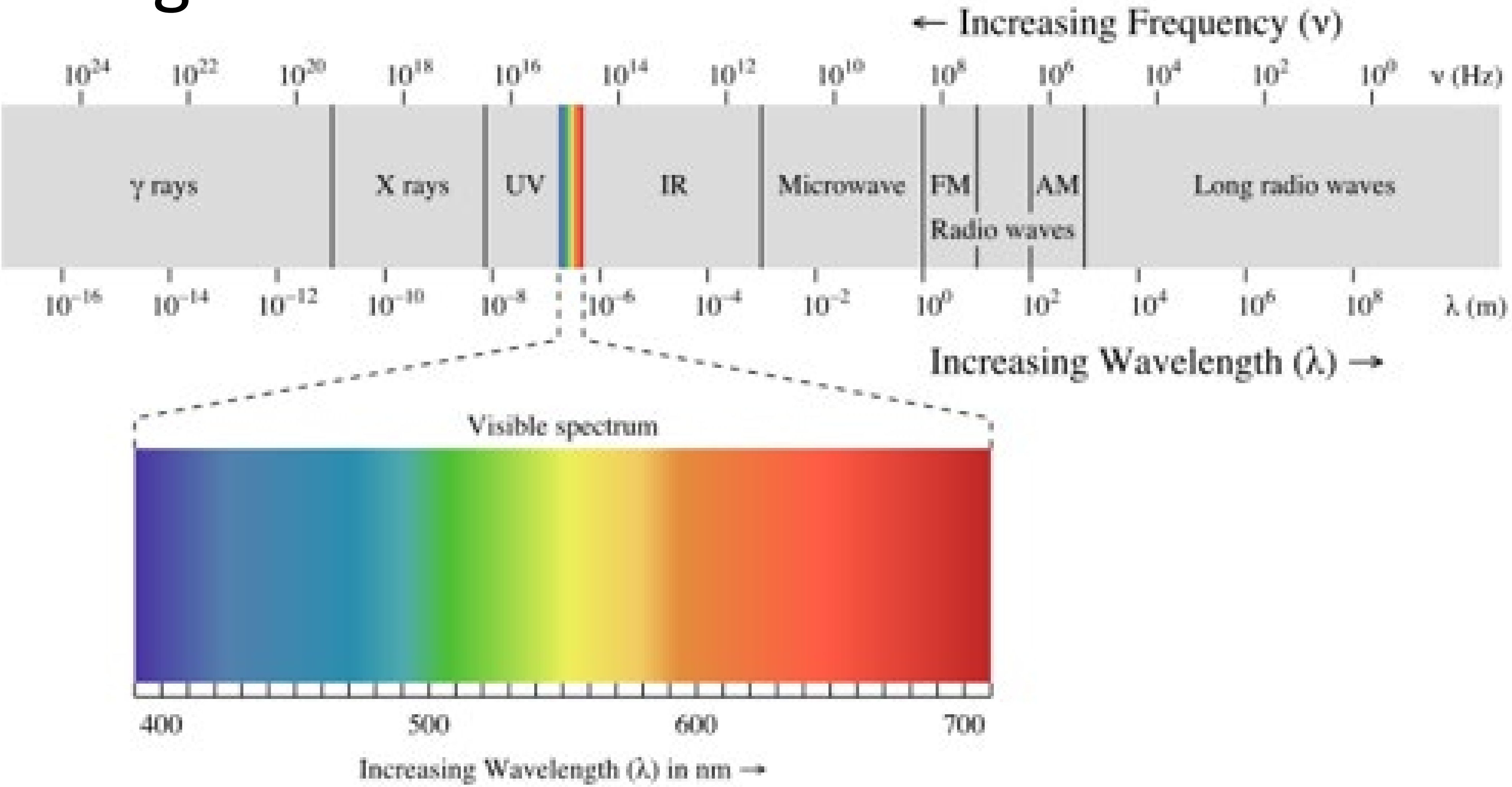


intensity



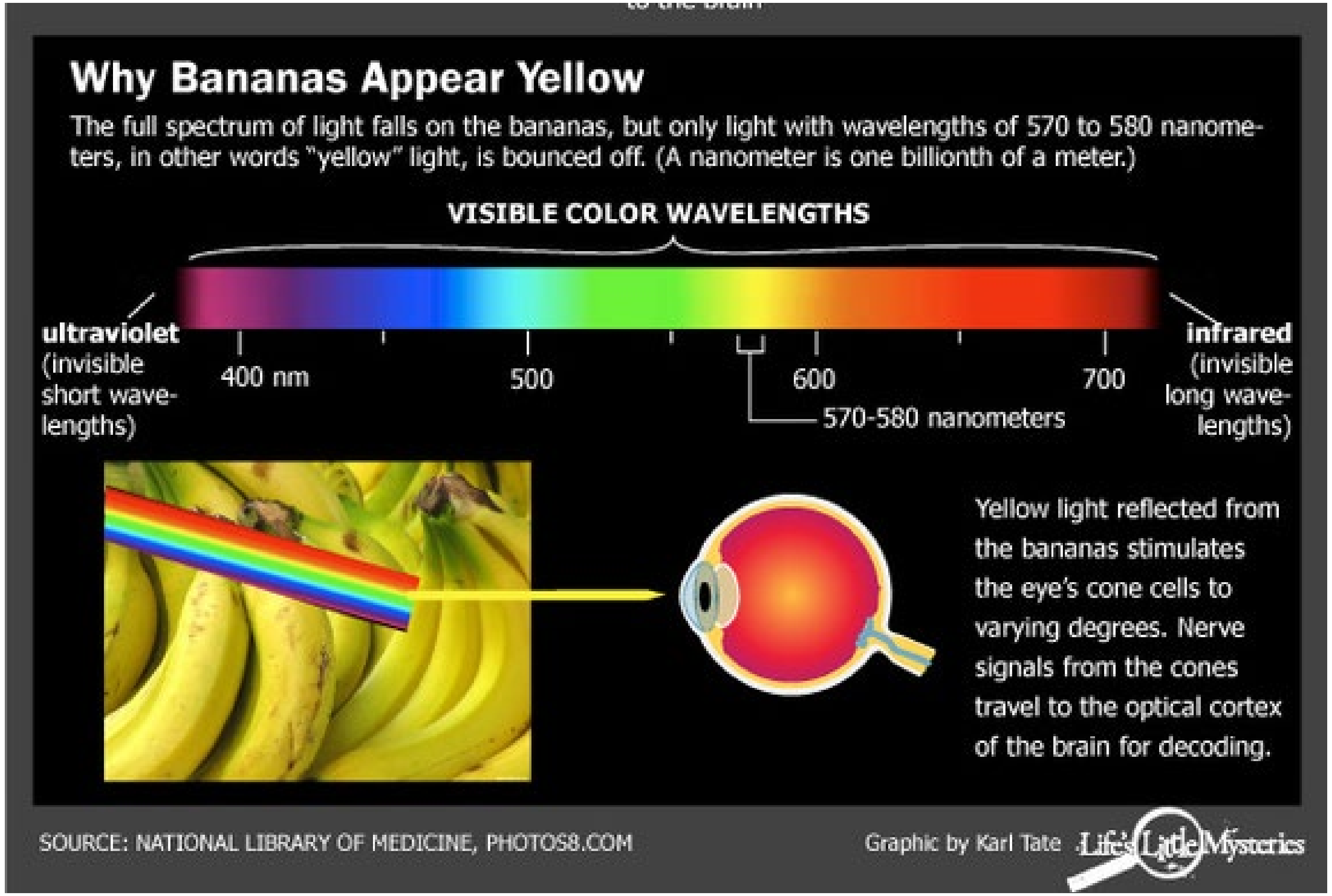
image

# Radiometry
## Electromagnetic radiation

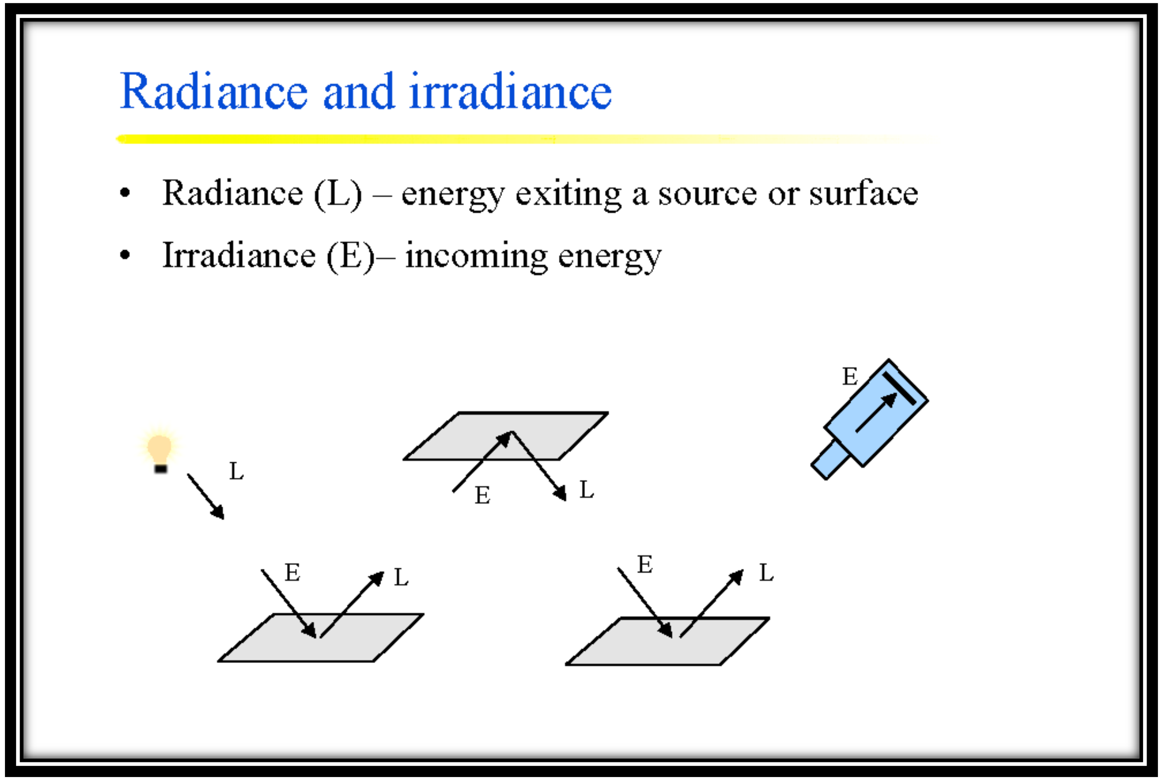



**Light is electromagnetic radiation that is visible to eye**

# Radiometry
## Electromagnetic radiation



# Radiometry
## Radiance and irradiance






Slide by Matthew Turk

# Radiometry
## Irradiance

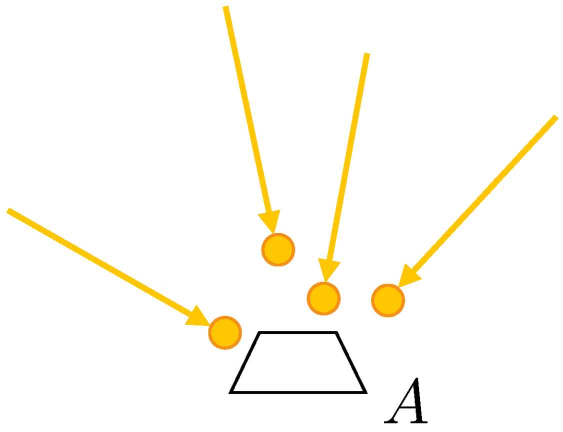
**Irradiance:** area density of radiant flux

Given a sensor with area $A$, we consider the average flux over the entire sensor area

$$\frac{\Phi}{A}$$

Irradiance $(E)$ = flux density, i.e., the incident flux per unit surface area

$$[\frac{\text{Watts}}{\text{m}^2}]$$



# Given what we now know about radiant energy…

# Why do some parts of a surface look lighter or darker?

# Radiometry

## Lambert's cosine law

Consider beam with flux $\Phi$ incident on surface with area $A$

radiant flux

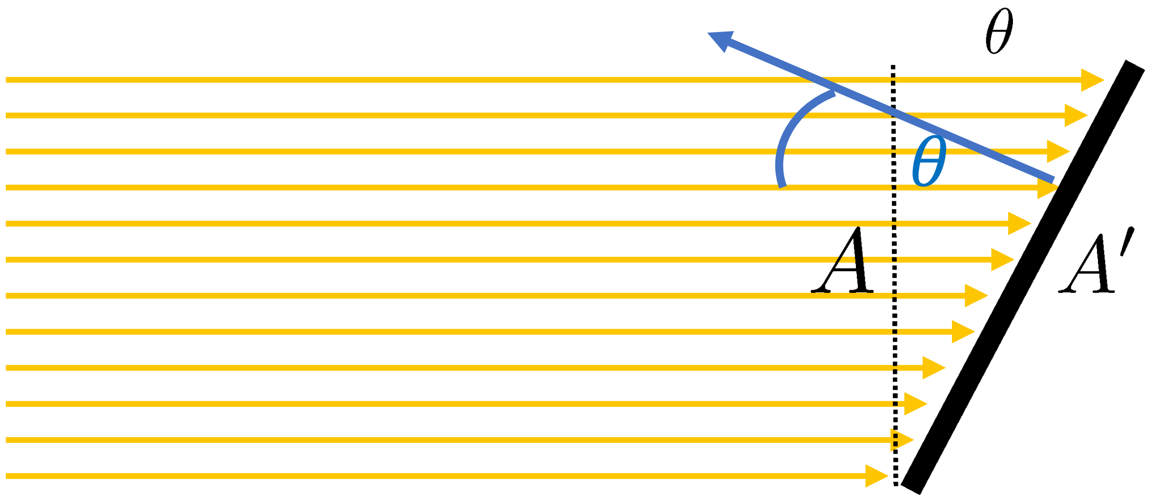
irradiance

$$E = \frac{\Phi}{A}$$

$$\Phi = EA$$

# Radiometry
## Lambert's cosine law

- Consider beam with flux $\Phi$ incident on tilted surface with area $A'$
- Irradiance at surface is proportional to cosine of the angle between light direction and surface normal



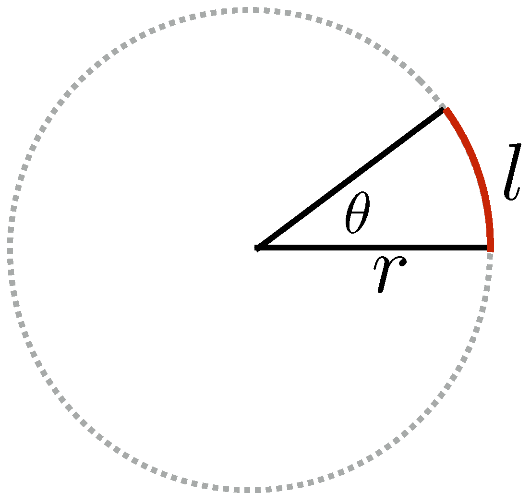

$$A = A' \cos\theta$$

$$E = \frac{\Phi}{A'} = \frac{\Phi\cos\theta}{A}$$

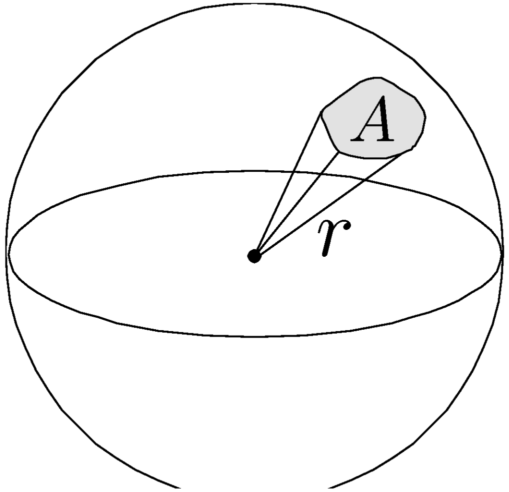
# Radiometry
## Solid angle

We need to break the energy over direction (angles), not just over space

- **Angle: ratio of subtended arc length on circle to radius**
  - $\theta = \frac{l}{r}$
  - **Circle has $2\pi$ radians**

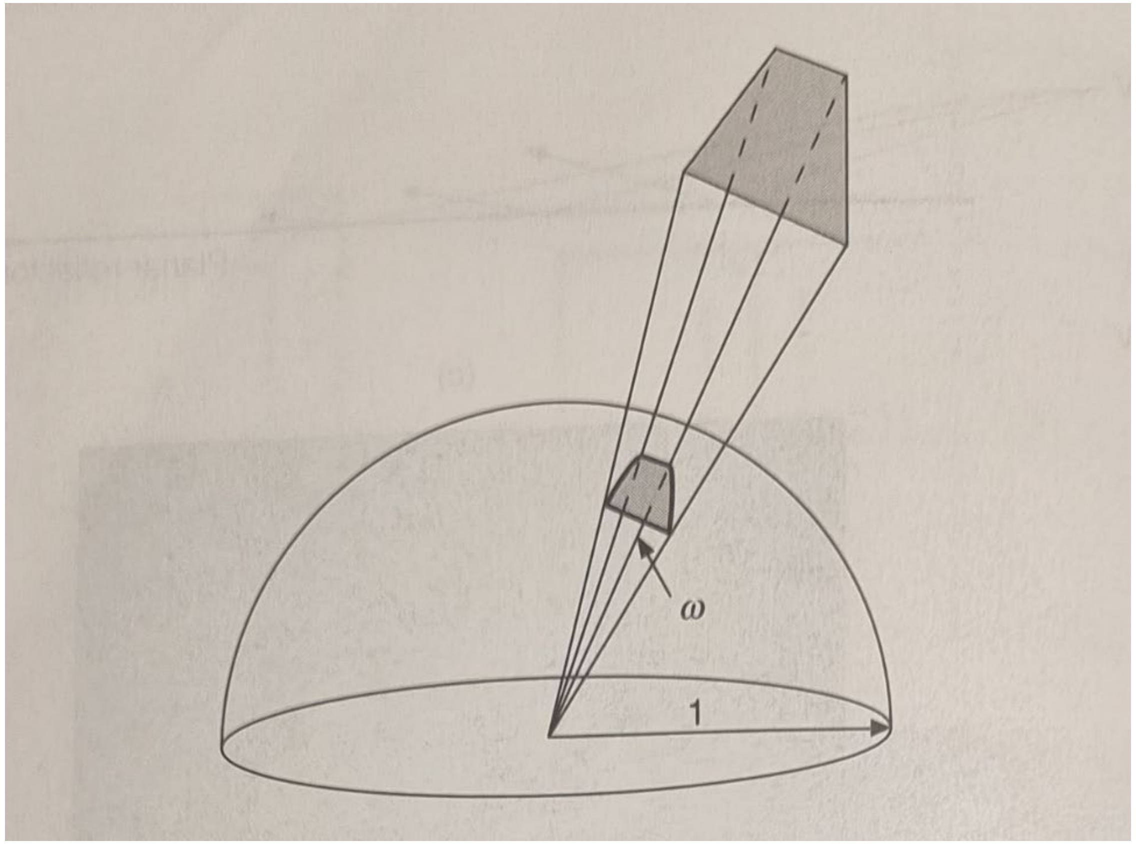



- **Solid angle: ratio of subtended area on sphere to radius squared**
  - $\Omega = \frac{A}{r^2}$
  - **Sphere has $4\pi$ steradians**



# Radiometry
Solid angle

The solid angle subtended by an object from a point on a surface =
The area covered by the object's projection onto a unit hemisphere above the point



# Radiometry
## Radiance

**Radiance** is the solid angle density of irradiance

$$L(\boldsymbol{p}, w)$$

Radiance is energy along a ray defined by origin point $\boldsymbol{p}$ and direction $w$

Radiant energy per unit time per unit area per unit solid angle

$$[\frac{\mathrm{W}}{\mathrm{m}^2\ \mathrm{sr}}]$$

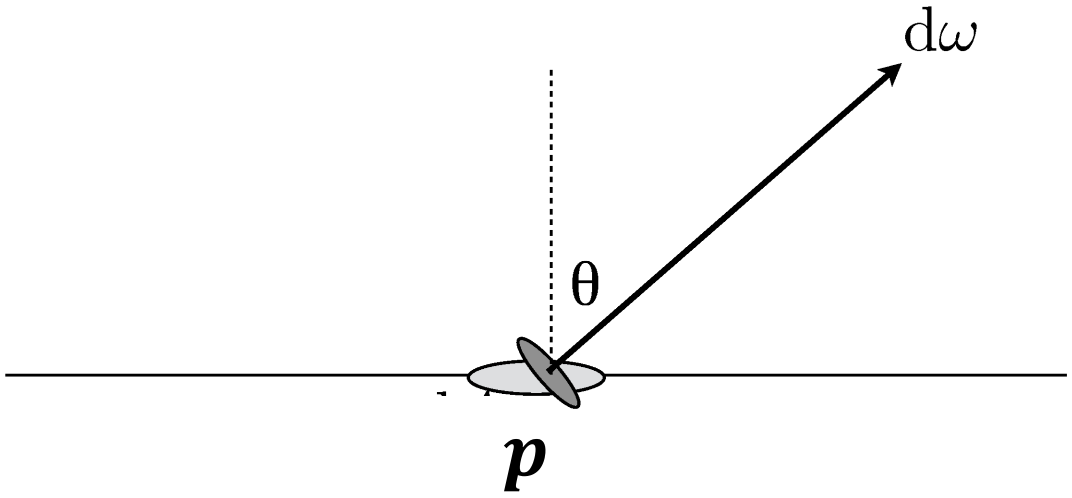
# Radiometry

## Radiance

A surface experiencing radiance $L(p, w)$, coming in from solid angle $dw$, experiences irradiance

$$dE(\boldsymbol{p}) = \quad L(\boldsymbol{p}, w) \quad \cos(\theta) \quad dw$$

Radiance | Lambert's law | Solid angle



# Radiometry
## Radiance (Fields) properties

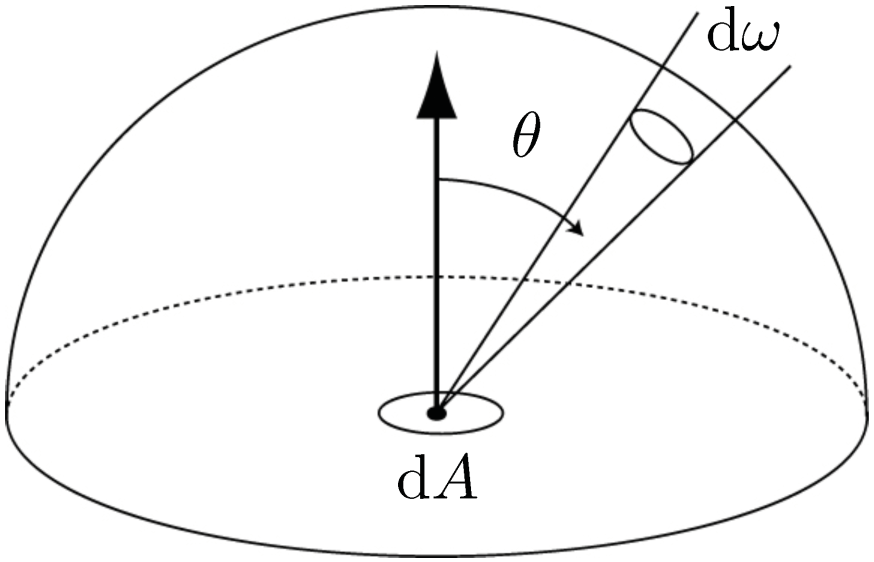
- Radiance is a fundamental quantity that characterizes the distribution of light an environment
- Radiance is the quantity associated with a ray (constant a long a ray)
- Rendering is all about computing radiance
- A pinhole camera measures radiance

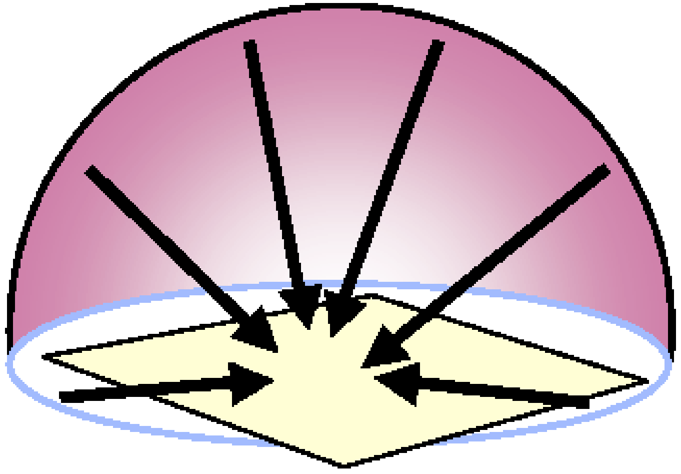
# Radiometry

## Irradiance from the environment

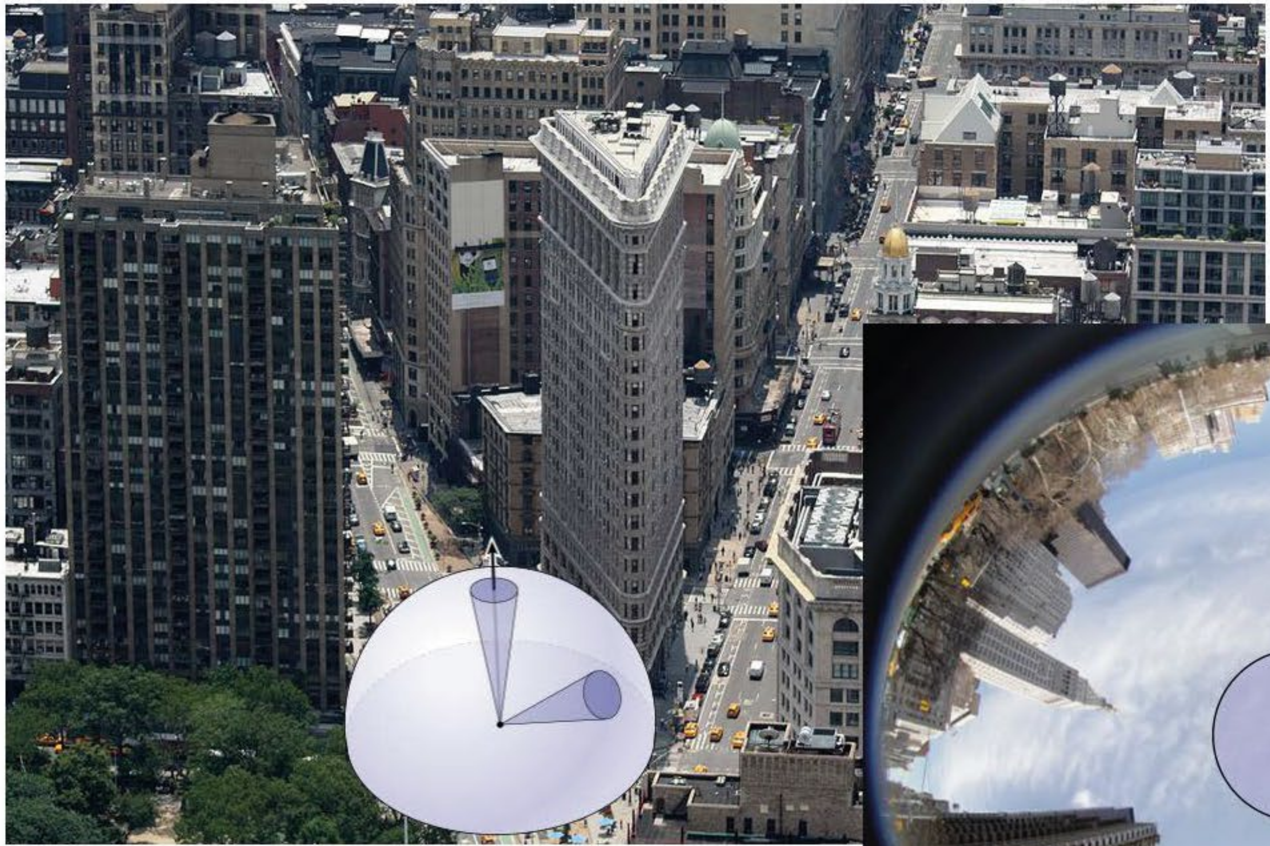
Computing flux per unit area on surface, due to incoming light from all directions

$$E(\mathsf{p}) = \int_{H^2} L_i(\mathsf{p}, \omega) \cos\theta d\omega$$



# Recap: Radiance and Irradiance



irradiance

radiance in direction ω

$$E = \int_{H^2} L(\omega) \cos\theta \, d\omega$$

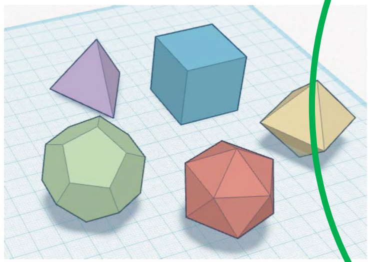
angle between ω and normal

# Photorealistic Rendering—Basic Goal

## What are the INPUTS and OUTPUTS?

camera

geometry



materials

lights

("scene")

Rendering

image

# Radiometry
## Bidirectional reflectance distribution function

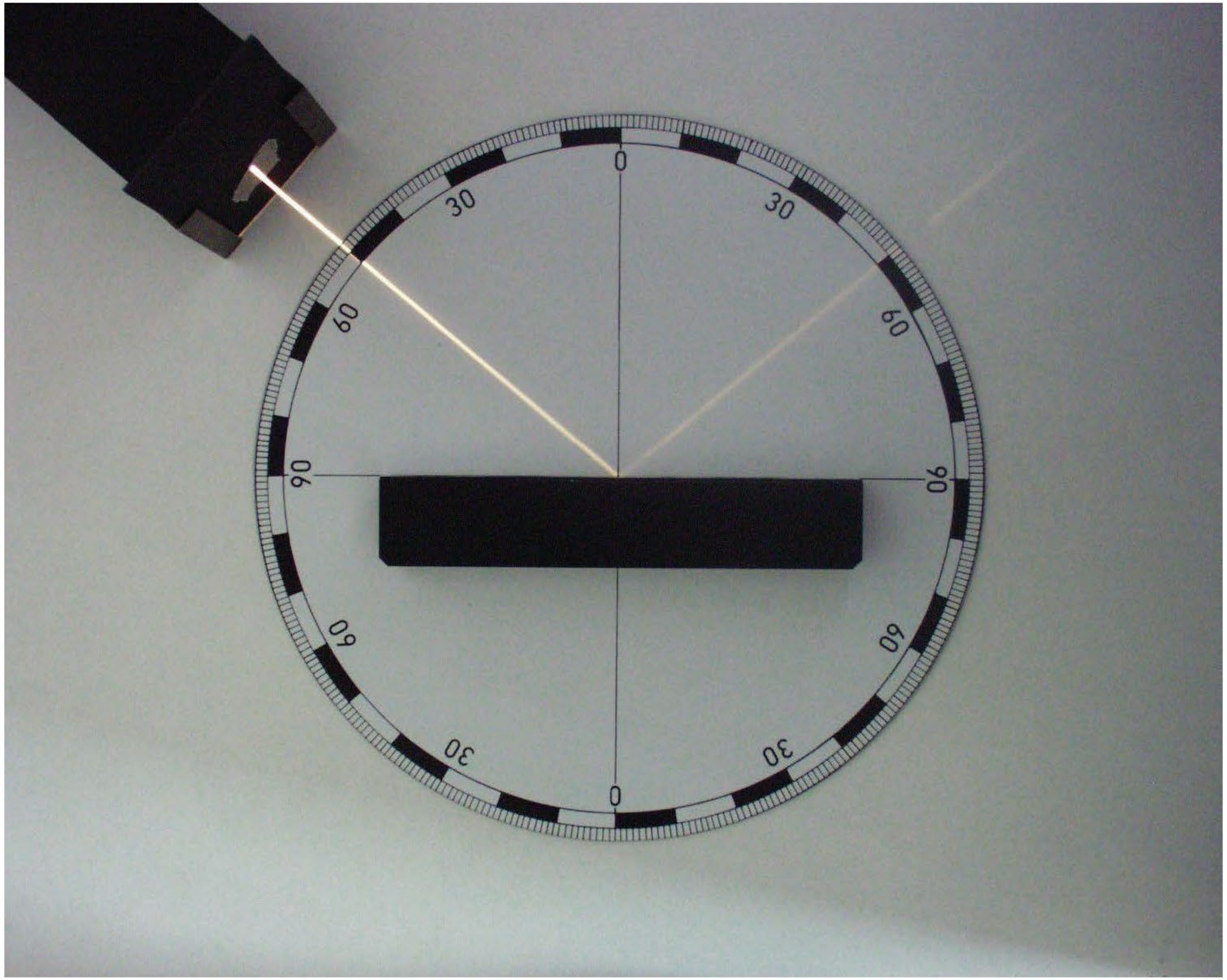
- When light hits a surface, the way it is reflected (scattered off the surface), depends on the surface material properties

- This is encoded by the "Bidirectional reflectance distribution function" (BRDF)

- Given incoming direction $w_i$, how much light gets scattered in any given outgoing direction $w_o$?

- The BRDF tells us how bright a surface appears when viewed from one direction while light falls from another one

# Example: perfect specular reflection




[Zátonyi Sándor]

# Radiometry
## Reflected radiance and incident irradiance

The incident radiance $L_i$

The incident irradiance $E_i = L_i \cos\theta_i dw_i$

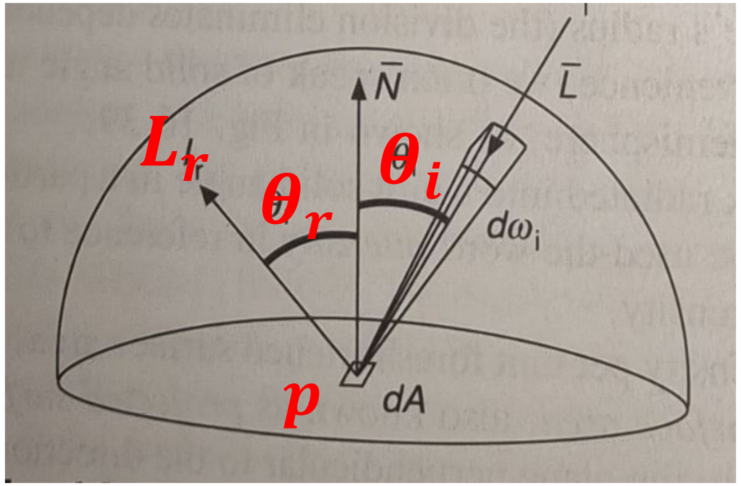
The reflected radiance $L_r$

BRDF = $f(p, w_i, w_r) = \dfrac{reflected\ energy}{incident\ energy} = \dfrac{L_r}{E_i}$

$$[\frac{1}{\text{sr}}]$$

$L_i$

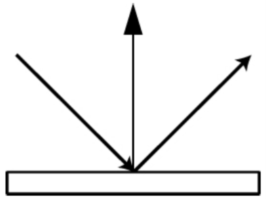
$E_i = L_i \cos\theta_i dw_i$



# Some basic reflection functions

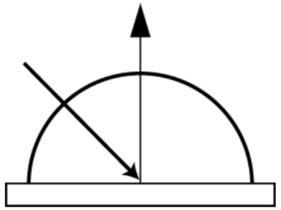
- **Ideal specular**

  Perfect mirror



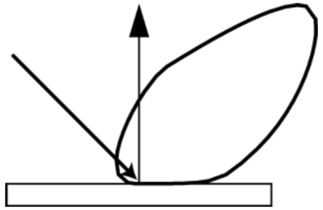
- **Ideal diffuse**

  Uniform reflection in all directions



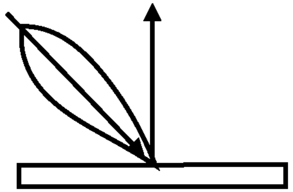

- **Glossy specular**

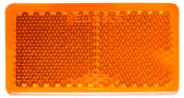
  Majority of light distributed in reflection direction



- **Retro-reflective**

  Reflects light back toward source



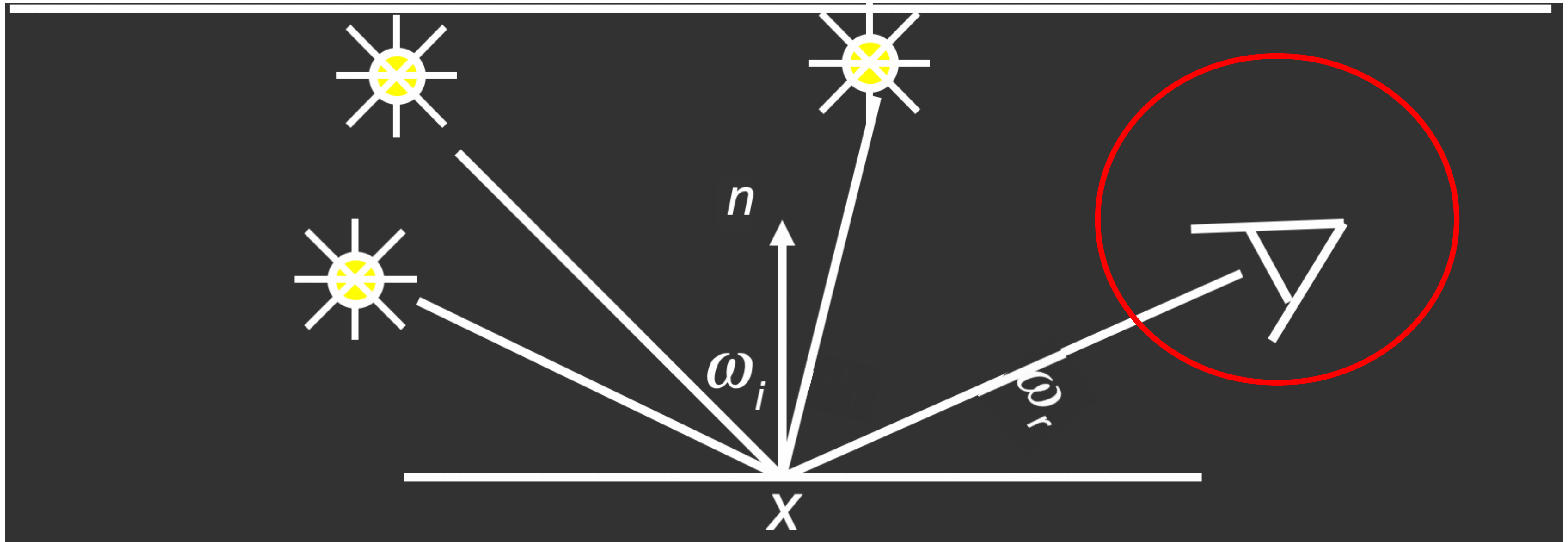



Diagrams illustrate how incoming light energy from given direction is reflected in various directions.

# Materials: diffuse

# Reflection equation

## Multiple light sources



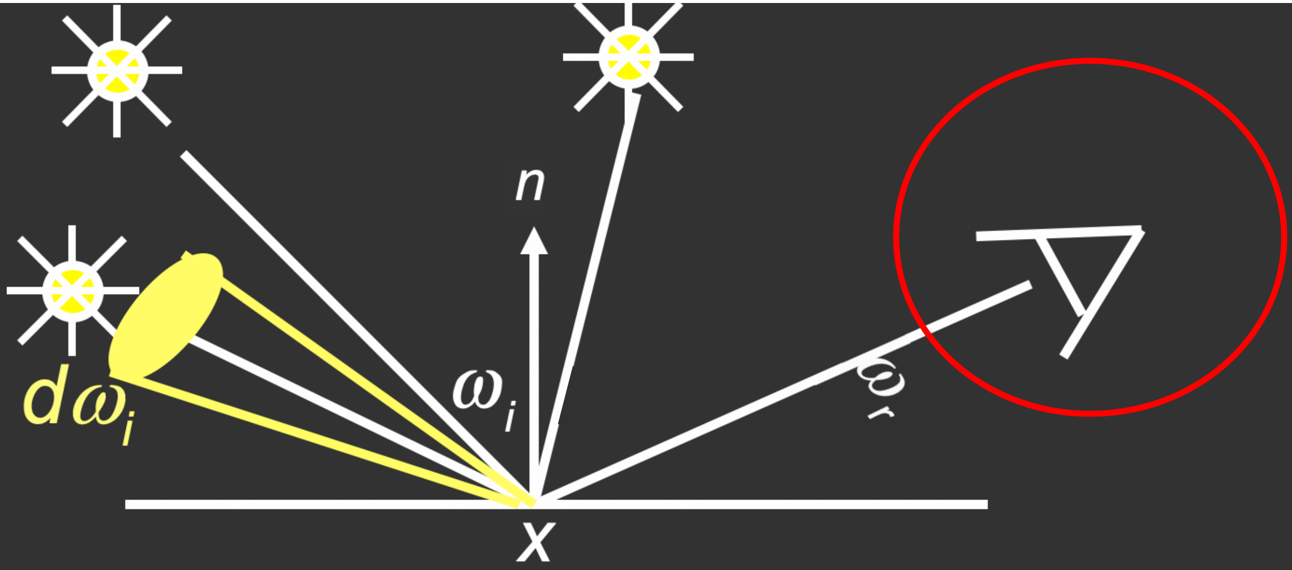

Sum over all light sources

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \sum L_i(x,\omega_i) f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

Reflected Light (Output Image) | Emission | Incident Light (from light source) | BRDF | Cosine of Incident angle

# Reflection equation

Environment of light sources




Replace sum with integral

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

Reflected Light (Output Image) | Emission | Incident Light (from light source) | BRDF | Cosine of Incident angle



Slide by Lior Yariv

# Reflection equation (local illumination)
## Recap

- The image of a three dimensional object depends on its shape, its reflectance properties, and the distribution of the light sources
- The interactions of light with scene surfaces depend on the material properties of the surfaces. Materials may be represented as bidirectional reflectance distribution functions (BRDF)
- The BRDF leads to the reflection equation
- The reflection equation considers only local illumination (direct light), i.e., light directly from light sources to surfaces

# Rendering equation (global illumination)

- Core functionality of photorealistic renderer is to estimate radiance at a given point, in a given direction

- To get photorealism we need to consider global illumination, multiple bounces (indirect light), called  interreflections.

- In real scenarios,  light reflected from an object strikes other objects in the surrounding area, illuminating them

# Rendering equation (global illumination)

James Kajiya, 1986

- Computing reflection equation requires knowing the incoming radiance from surfaces

- But determining incoming radiance requires knowing reflected radiance from surfaces

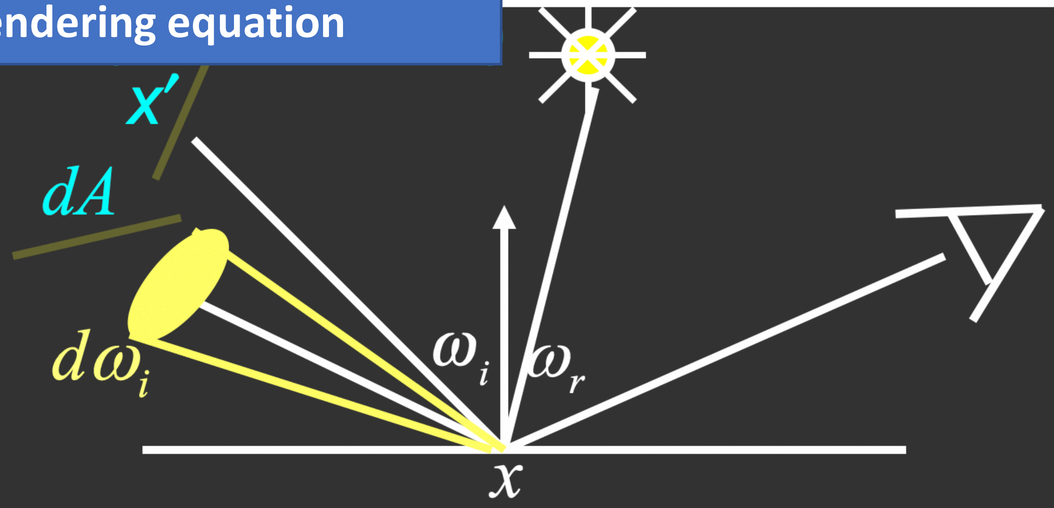
- So we have to compute another integral, we have exactly the same equation

- Rendering equation is recursive

**Reflection equation**

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_\Omega L_i(x,\omega_i) f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

**Rendering equation**

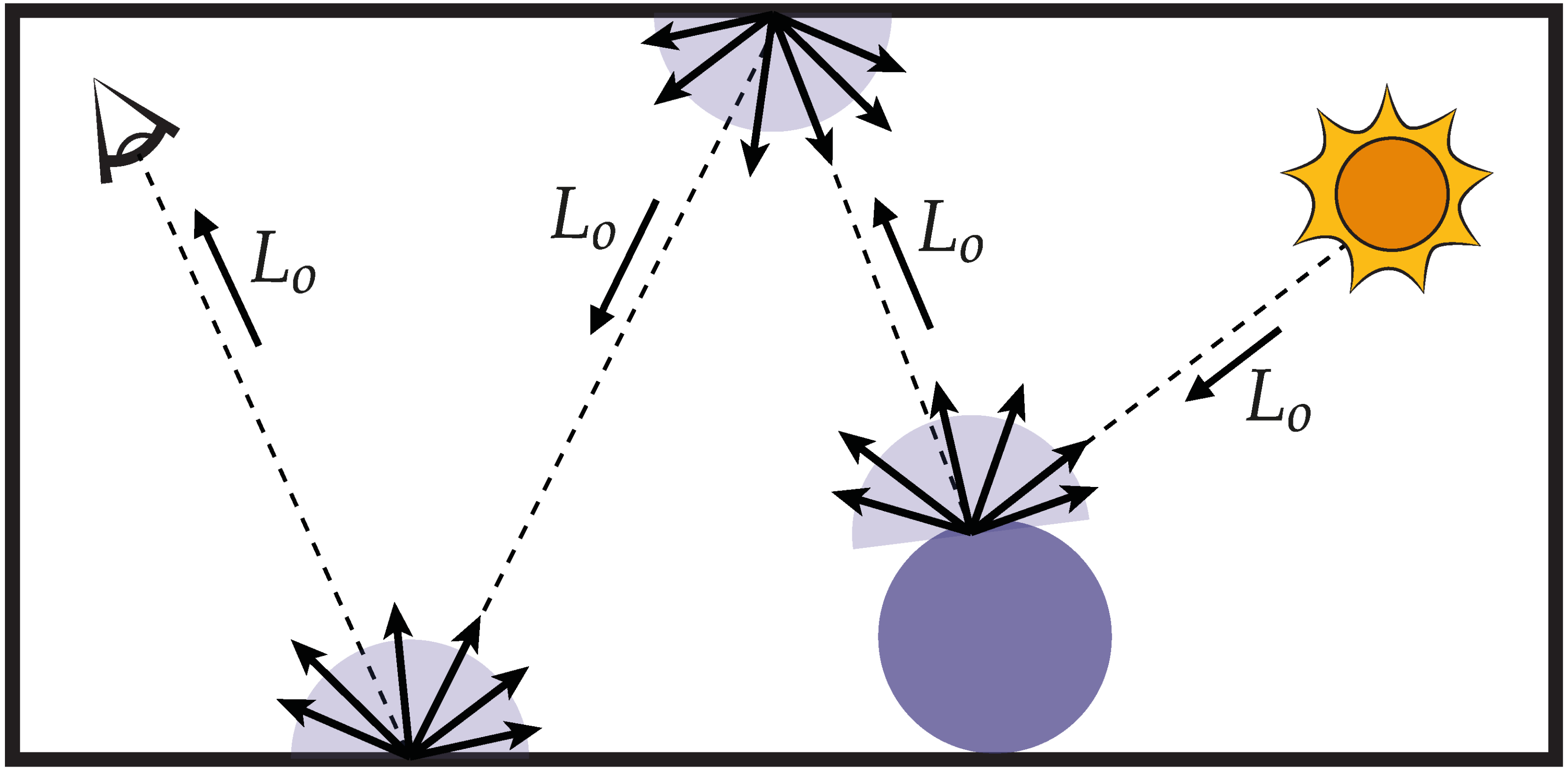



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_\Omega L_r(x',-\omega_i) f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

| Reflected Light (Output Image) | Emission | Reflected Light | BRDF | Cosine of Incident angle |
|---|---|---|---|---|
| UNKNOWN | KNOWN | UNKNOWN | KNOWN | KNOWN |

# Recursive Raytracing

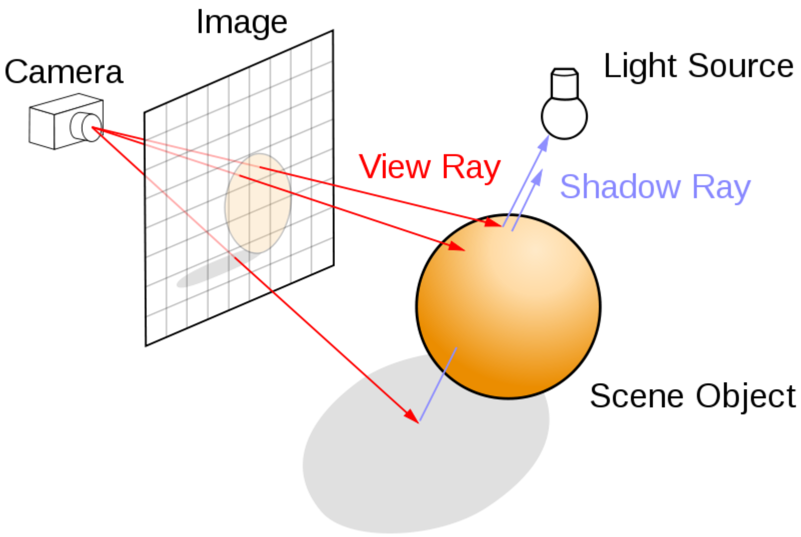
- Basic strategy: recursively evaluate rendering equation!



# Rendering equation

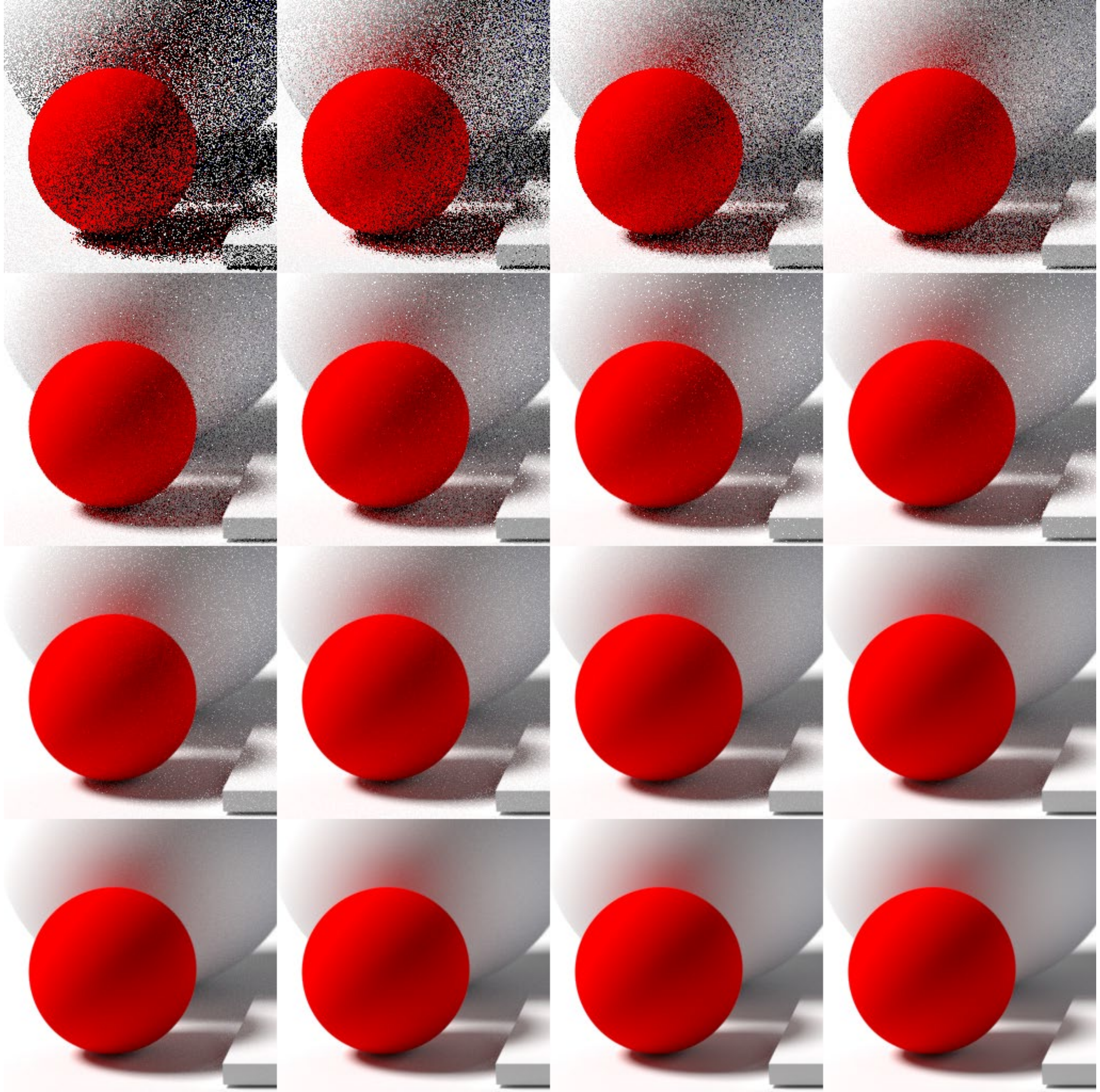
## How to solve?

- Too hard for analytic solution
- Very challenging to apply directly recursive ray tracing
- Monte-Carlo rendering
- Ray tracing is crucial here
- Little control in rasterization, which rays we evaluate?



Noise decreases as the number of samples per pixel increases.

The top left shows 1 sample per pixel, and doubles from left to right each square.



p
Direct illumination


•p
One-bounce global illumination


p
Two-bounce global illumination


p
Four-bounce global illumination


p
Eight-bounce global illumination


•p
Sixteen-bounce global illumination

# Summary

- Computer graphics, in particular classical rendering: **ray tracing** and **rasterization**
- Geometry representation, specifically explicit representation by **triangular mesh**
- **Radiometry**, including radiance and irradiance
- **Materials** properties are encoded by **BRDF** (Bidirectional reflectance distribution function)
- Illumination models
  - **local model** -> reflection equation
  - **global model** -> rendering equation
- Very challenging to solve the **rendering equation**
- Simplifications by **Monte-Carlo sampling**
- **Neural rendering and implicit representation** (next time)