Lecture 17
Computer Vision and Inverse problems
Computer Graphics and Rendering

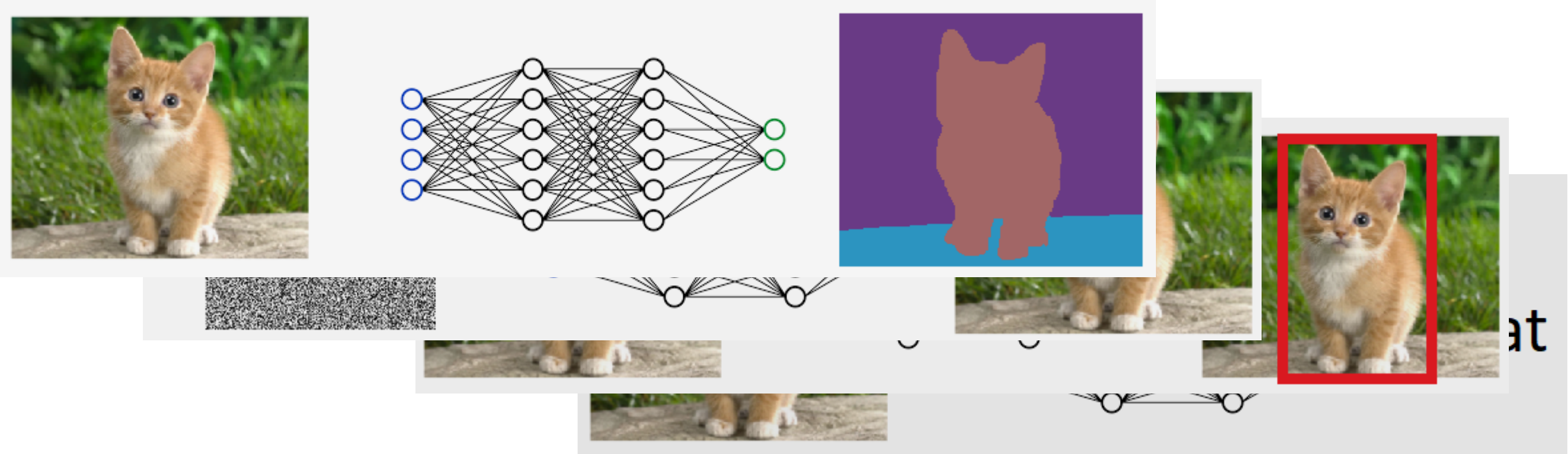February 7th 2024

Meirav Galun

WAIC

Material covered

- Inverse problems in computer vision

- Computer graphics and image rendering

<mark>Implicit representation of signals</mark>

Computer vision

We are familiar already with a variety of neural net architectures, dedicated for tasks such as classification, detection, generation, segmentation…
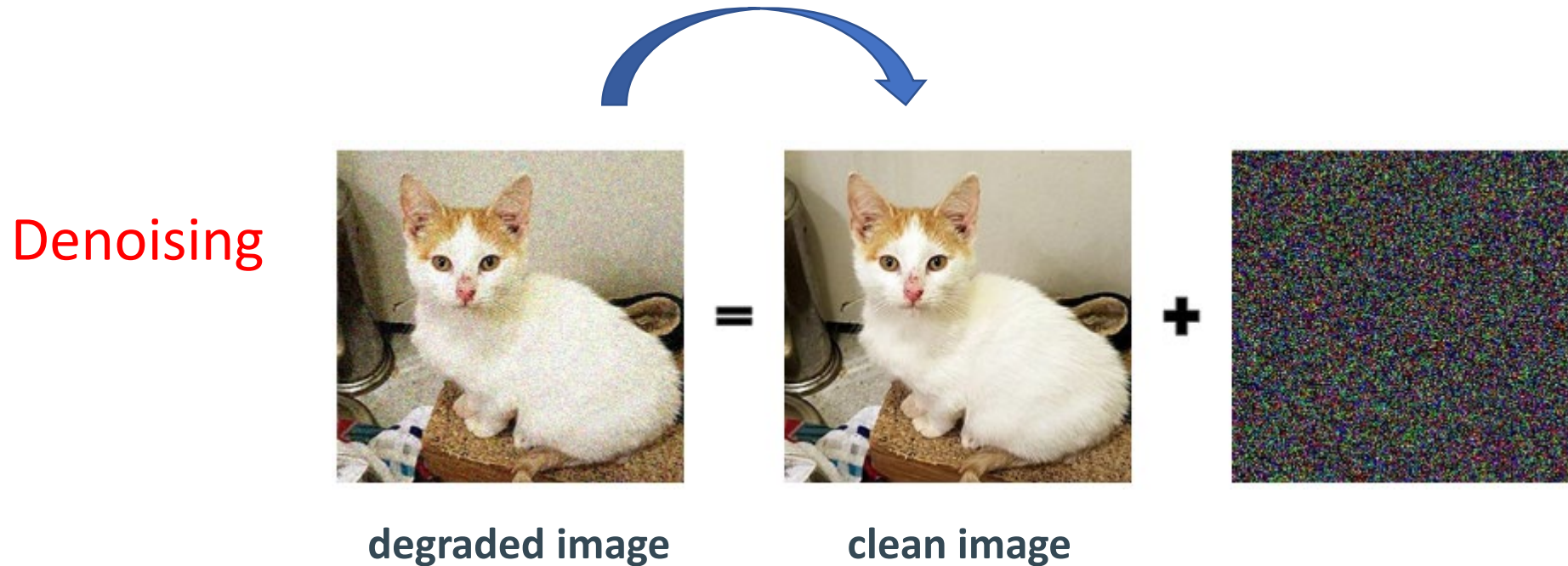
Inverse problems in computer vision

The input is a <u>degraded image,</u> and the aim is to restore the original clean image

- Restoration tasks
  - denoising
  - super-resolution
  - 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

- <u>Supervised</u> 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?
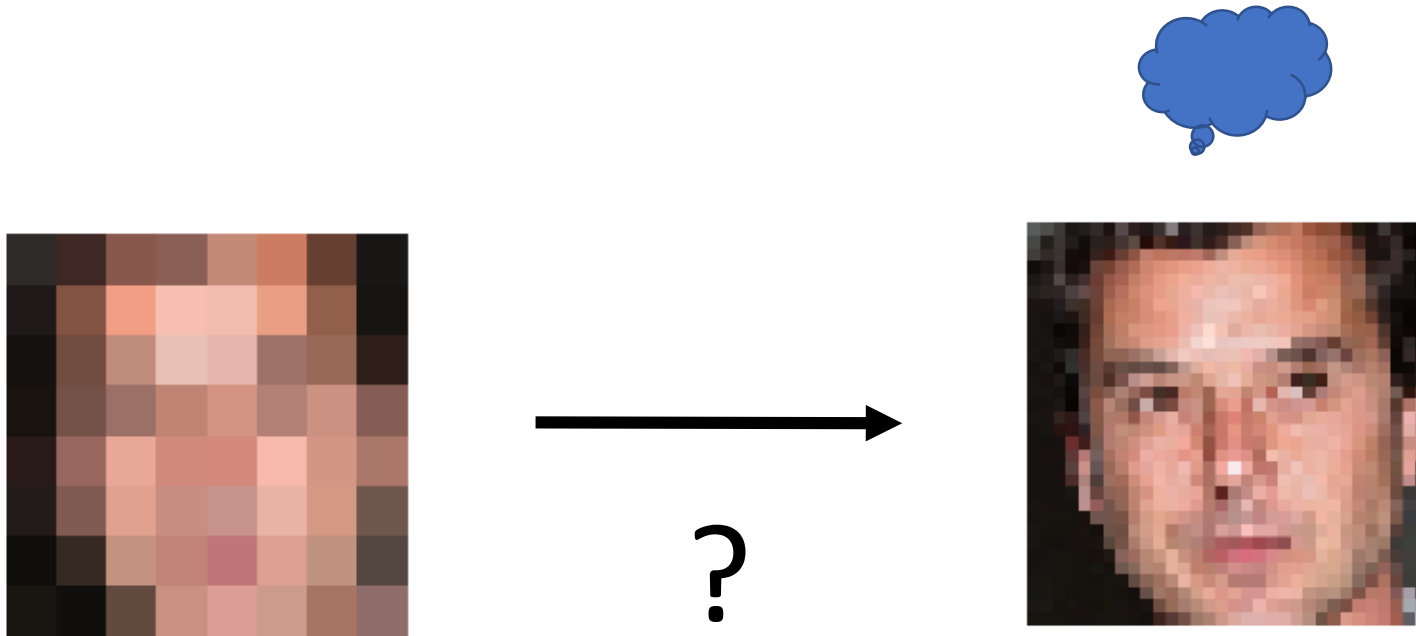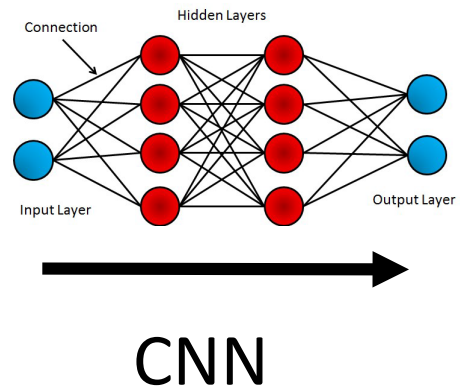

Prior = our knowledge about the visual world

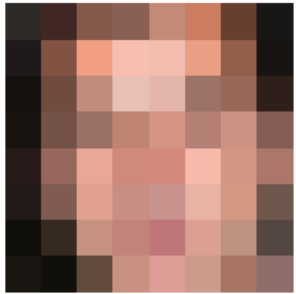Images from Dahl et al, 2017

What is a prior?

How to construct a prior?

Learned priors

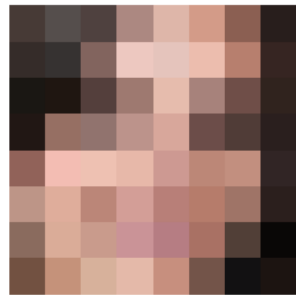by collecting large databases

Explicit priors

by introducing constraints

Learned priors
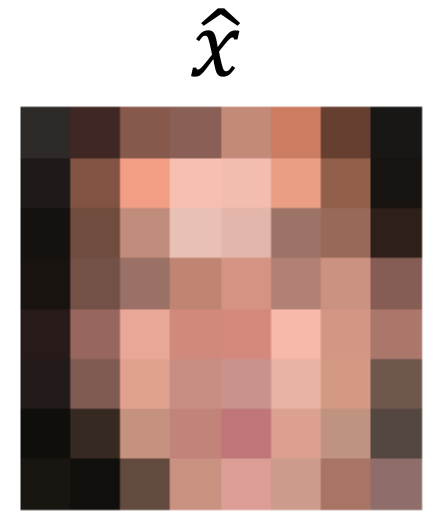

CNN

Explicit priors

$\hat{x}$

- By introducing constraints

down-sampling

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

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

What is a prior?

How to construct a prior?

Learned priors

by collecting large databases

Explicit priors

by introducing constraints

<mark>Implicit priors</mark>

w/o Large databases
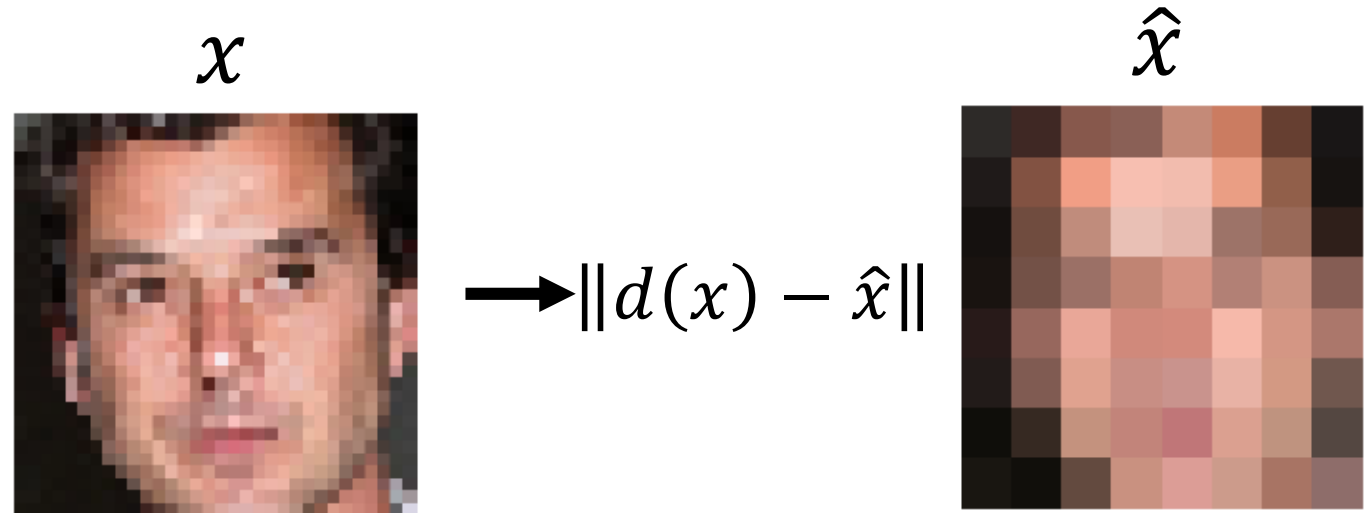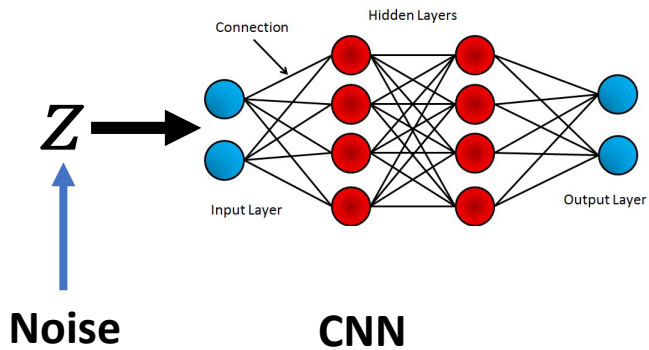
w/o Introducing constraints

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

x

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

$\hat{x}$

Noise

CNN

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. How?

- 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 <u>translation equivariance and locality</u> of the convolution operator

  - The hierarchy of such convolutions captures the statistics of pixel neighborhood at <u>multiple scales</u>

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

Deep image prior
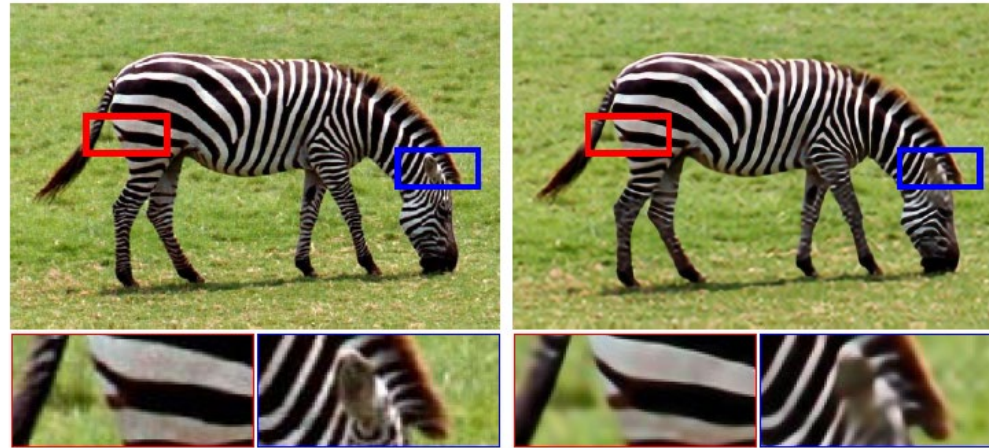# → Computer vision, Computer Graphics

The structure of the network allows

Parametrizing signals by the net weights
1. 2D images
2. 3D volumes
3. Continuous functions
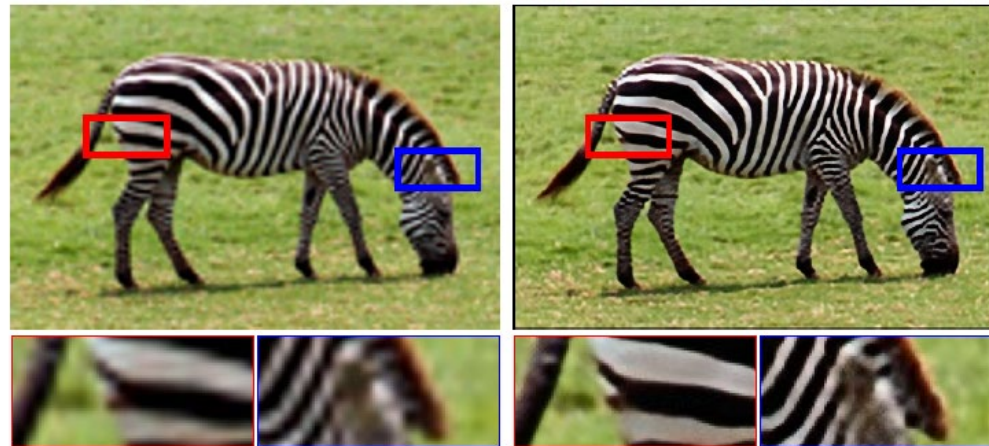
How and why is it useful?

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

$$x \quad \rightarrow \quad \boxed{\text{degradation}} \quad \rightarrow \hat{x} \quad \rightarrow \quad \boxed{\text{restoration}} \quad \rightarrow x^*$$

Image restoration

Problem setting

x - clean image

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

x^* - restored image

$$x \quad \rightarrow \quad \boxed{\text{degradation}} \quad \rightarrow \hat{x} \quad \rightarrow \quad \boxed{\text{restoration}} \quad \rightarrow x^*$$
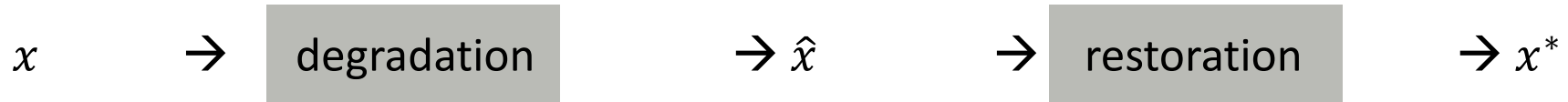
(maximum a posterior probability) $\textbf{MAP}: x^* = \underset{x}{\arg\max}\, p(x|\hat{x})$

$$p(x|\hat{x}) = \frac{p(\hat{x}|x)p(x)}{p(\hat{x})} \propto \textcolor{red}{p(\hat{x}|x)}\textcolor{blue}{p(x)}$$

Likelihood   Prior

Image restoration

The significant role of the prior

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

degradation

restoration

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

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

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

Image restoration
The significant role of the prior

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

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

Ignoring the prior $\Rightarrow$ The best estimation of clean image is the degraded image

Image restoration

Alternative notation

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

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

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

$$x^* = \arg \max_x p(x|\hat{x})$$
$$= \arg \max_x p(\hat{x}|x)p(x)$$
$$= \arg \min_x \left(-\log p(\hat{x}|x) - \log p(x) \right)$$
$$= {\color{red}\arg \min_x E(x; \hat{x}) + R(x)}$$

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

Image restoration

Optimization task

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

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

How are we going to optimize?
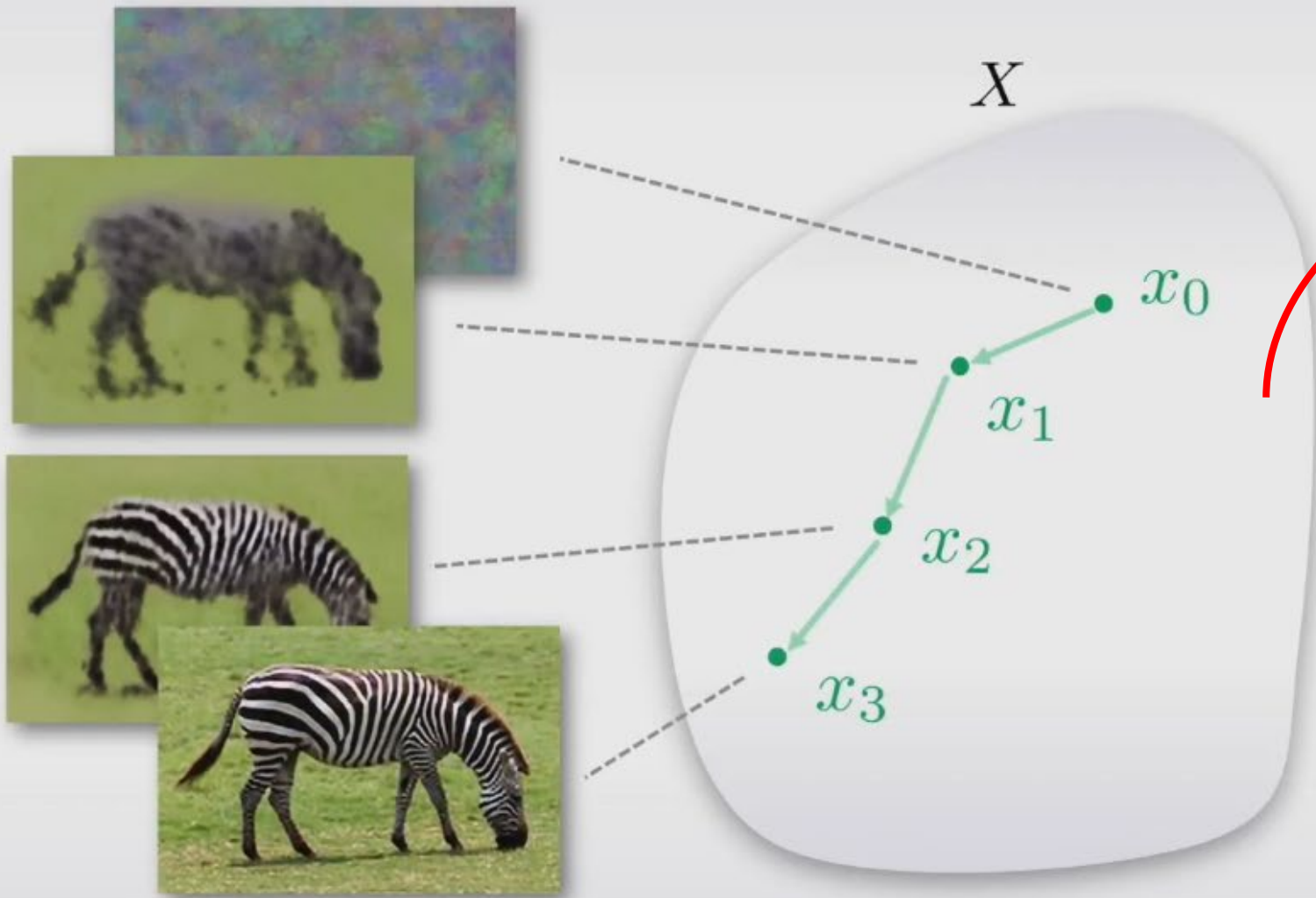
Parametrized

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

search in θ space

X

$g(\theta_0)$

$g(\theta_1)$

$g(\theta_2)$

$g(\theta_3)$

g

Θ

θ_0

θ_1

θ_2

θ_3
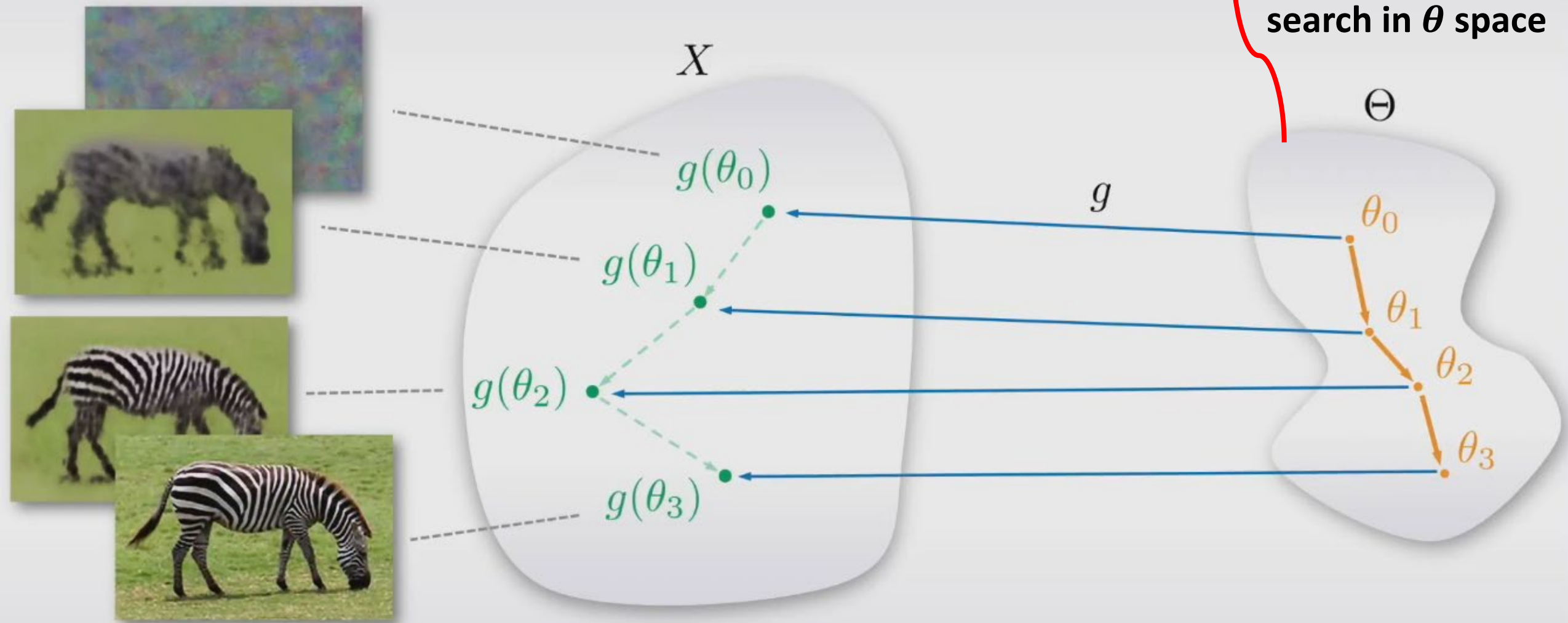
Image restoration
Optimization by parametrization

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 $g(\theta)$

Deep image prior

$\hat{x}$ corrupted image

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

$$g(\theta) \equiv f_{\theta}(z)$$

- f_{θ} is a convolutional neural network with parameters θ
- 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 θ, 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

Epoch = 0   Epoch = 50   Epoch = 500   Epoch = 22452
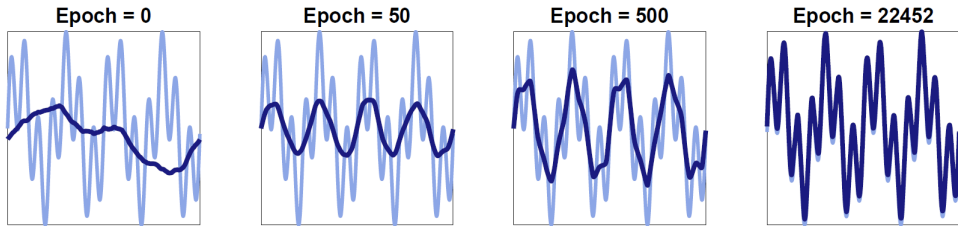
Ronen Basri, David Jacobs, Yoni Kasten, Shira Kritchman, NeurIPS 2019

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

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

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

$\hat{x} \qquad \hat{x} \qquad \hat{x} \qquad \hat{x}$

- Image
- Image + noise
- Image shuffled
- $U(0, 1)$ noise

The CNN is reluctant to noisy images and descends much more quickly and easily towards naturally-looking images
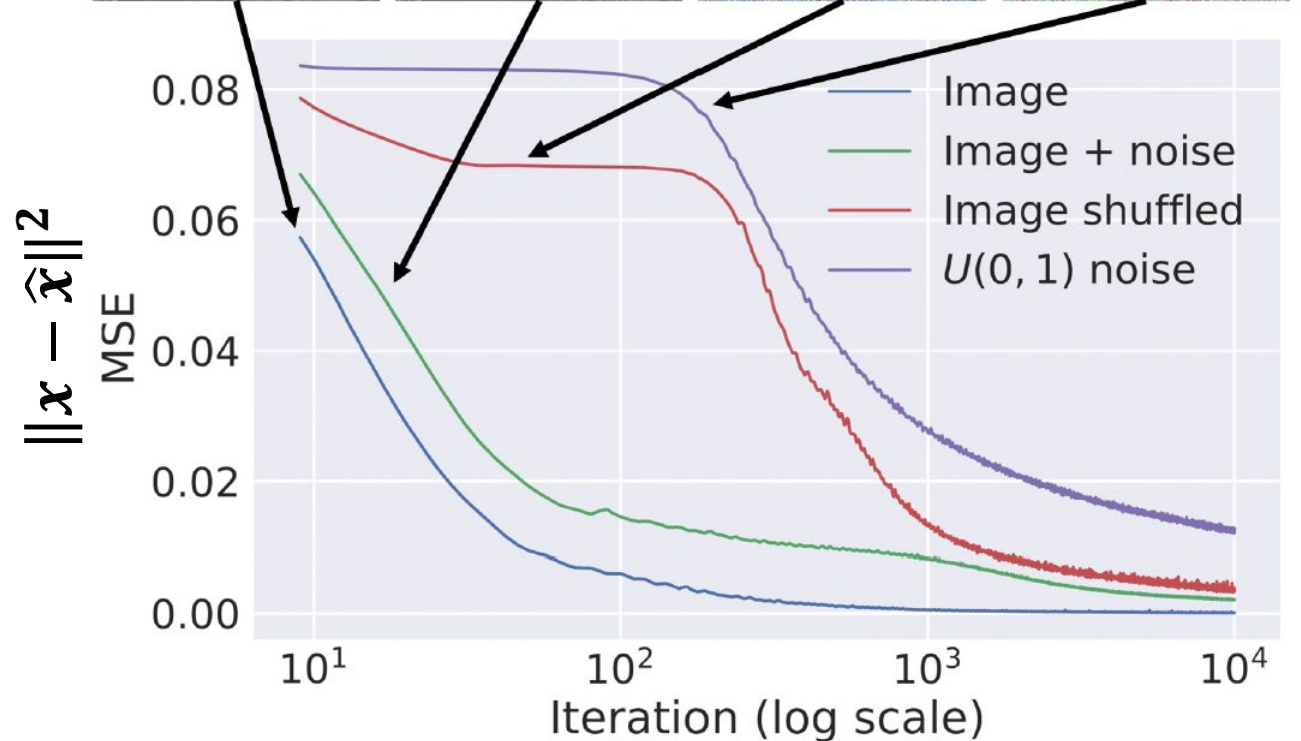
Deep image (implicit) prior

Recap

$$x \quad \rightarrow \quad \boxed{\text{degradation}} \quad \rightarrow \hat{x} \quad \rightarrow \quad \boxed{\text{restoration}} \quad \rightarrow x^*$$
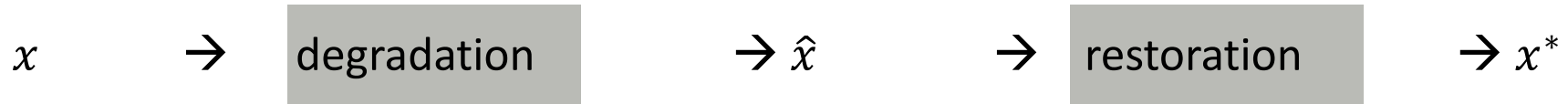
$$x^* = \arg\max_x p(x|\hat{x}) = \arg\max_x p(\hat{x}|x)p(x) = \arg\min_x E(x;\hat{x}) + R(x)$$

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

Implicit regularization by parametrizing the restored image using CNN with parameters θ
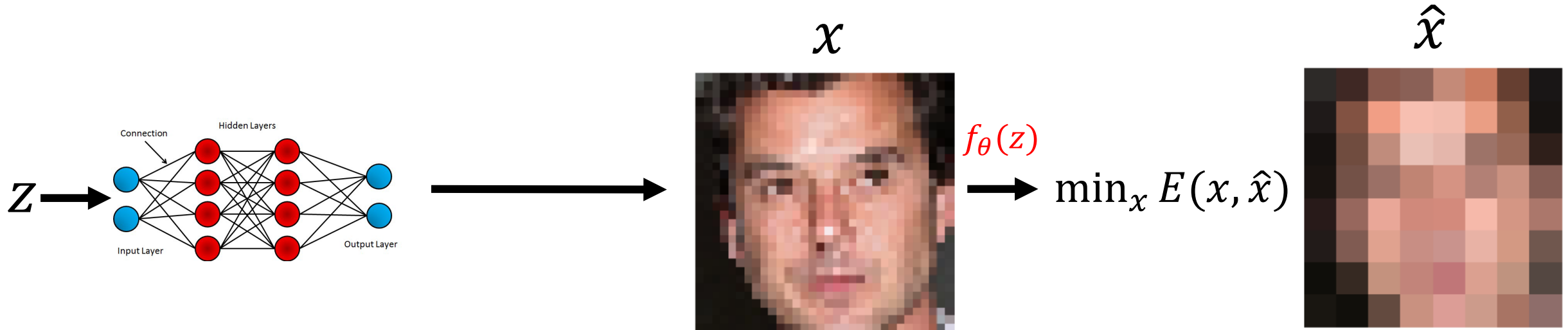
Deep image prior

Image restoration objectives

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

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

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

clean image x
corrupted image $\hat{x}$
binary mask m

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

Deep image prior

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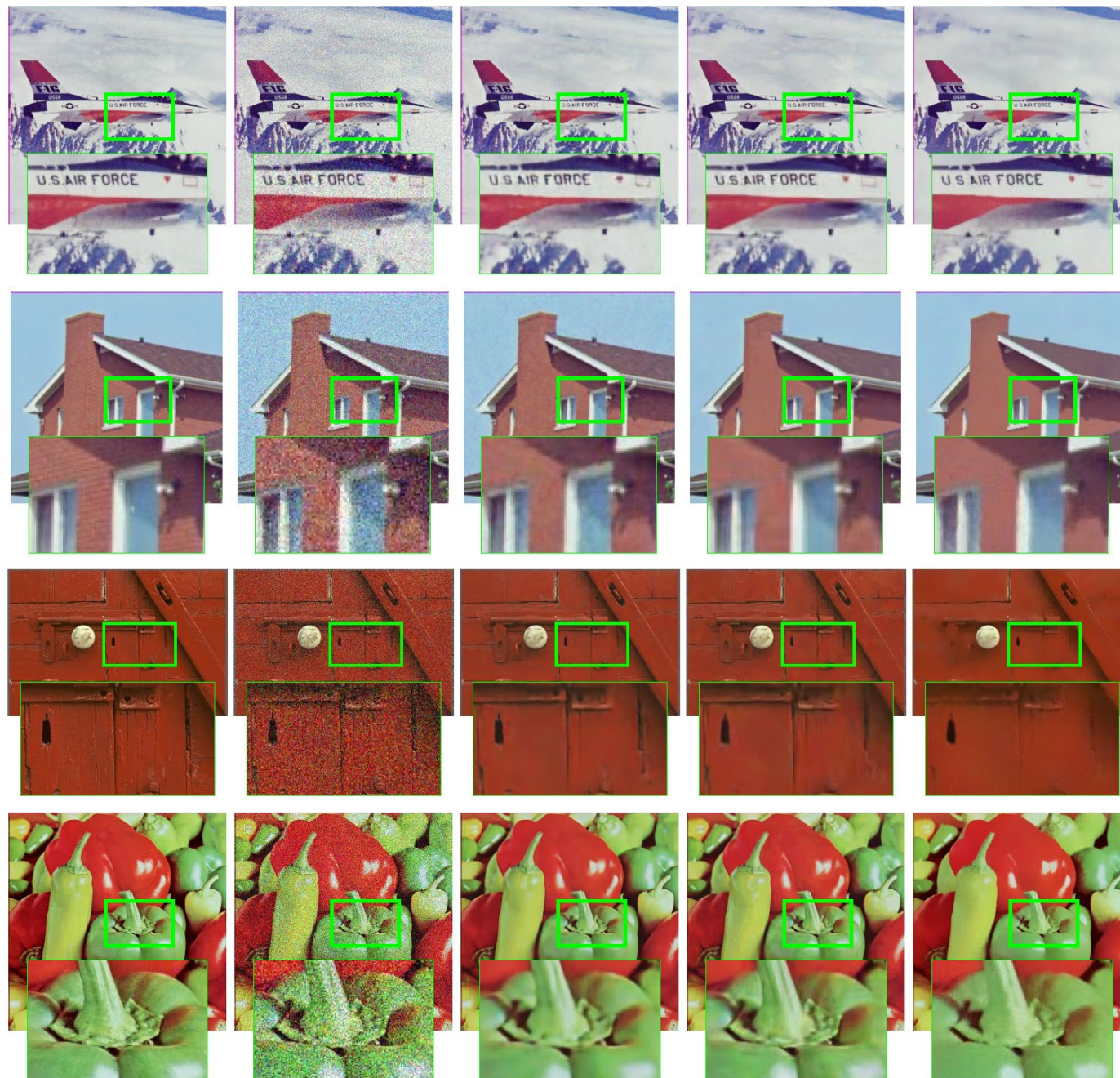
(a) GT   (b) Input   (c) Ours   (d) CBM3D   (e) NLM

Deep image prior

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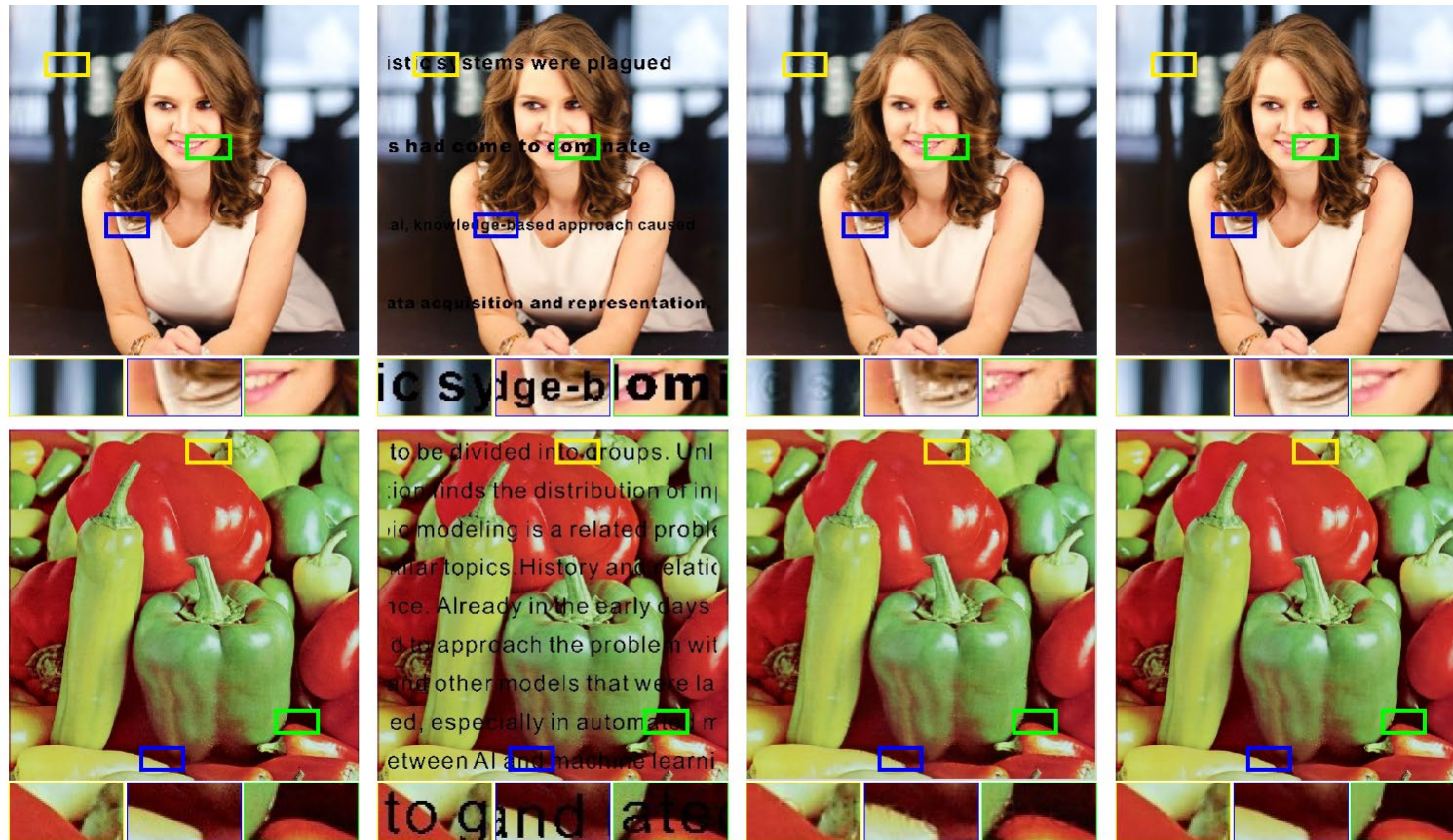
(a) Original image   (b) Corrupted image   (c) Shepard networks [47]   (d) Deep Image Prior

Deep image prior

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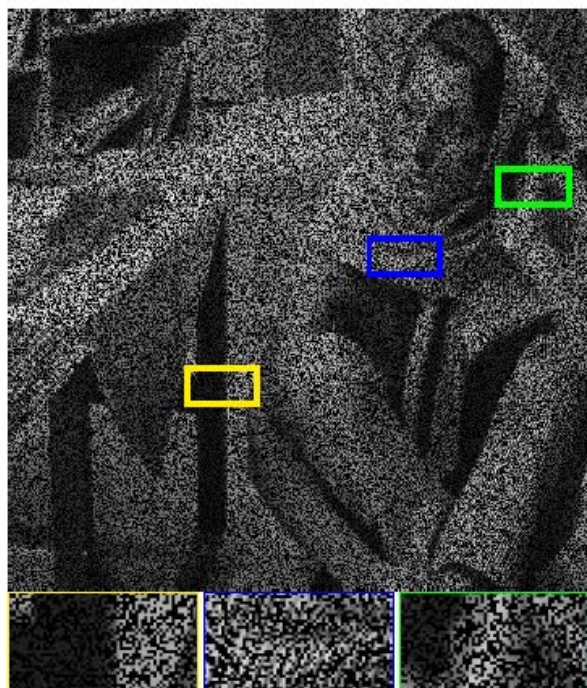
(a) Original image   (b) Corrupted image   (c) CSC [44]   (d) Deep image prior

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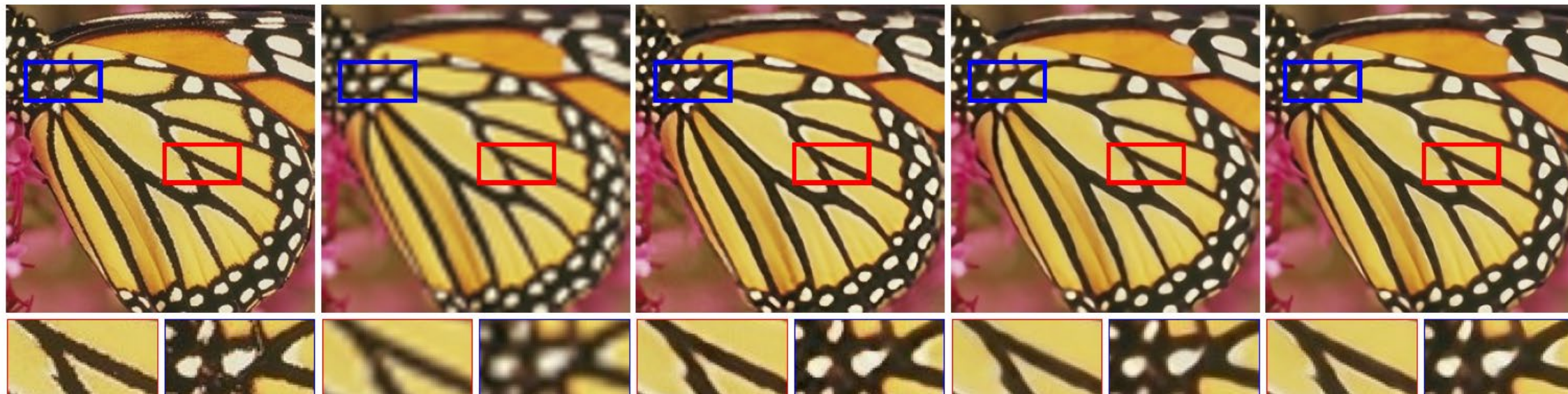
(a) Original image (b) Bicubic, **Not trained** (c) Ours, **Not trained** (d) LapSRN, **Trained** (e) SRResNet, **Trained**

8× super-resolution
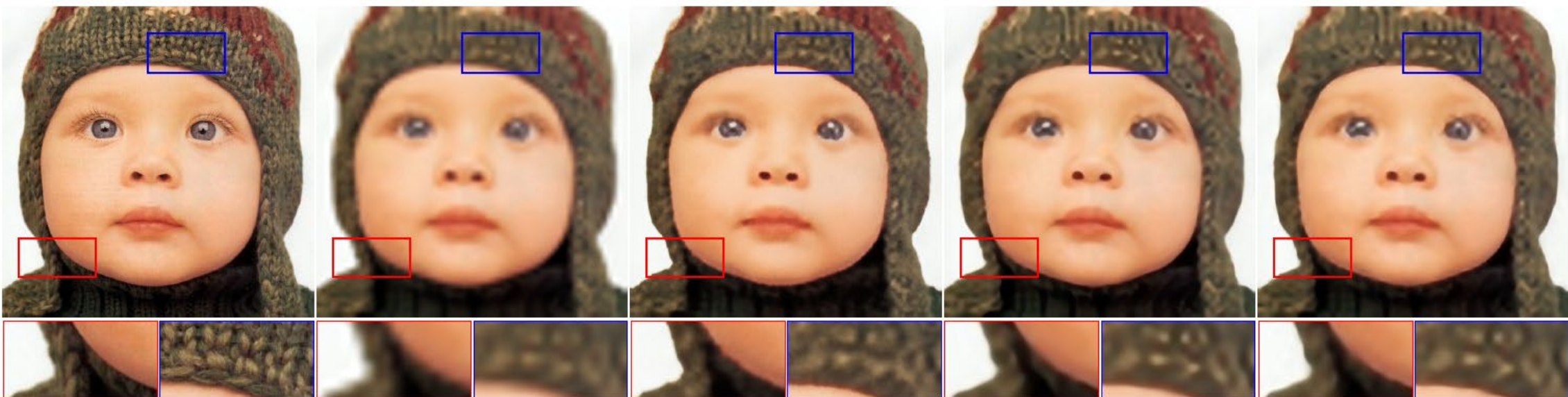
(f) Original image   (g) Bicubic, **Not trained**   (h) Ours, **Not trained**   (i) LapSRN, **Trained**   (j) VDSR, **Trained**

Deep image prior

Encode-decoder architecture
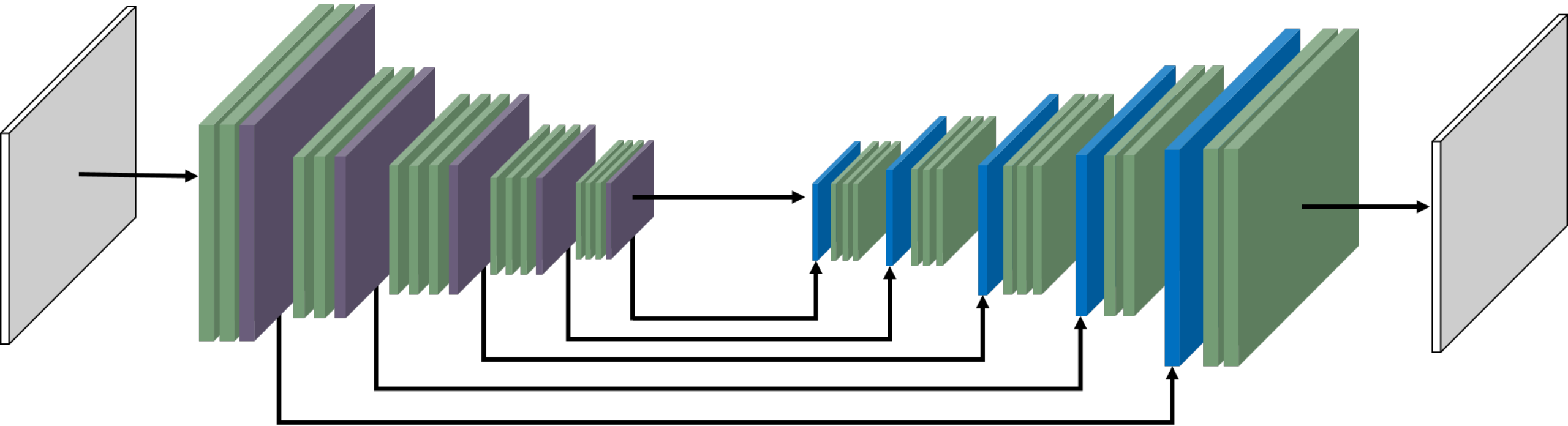
Deep image prior
Depths and architectures

(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

**Images
2D**

**Geometry
3D**

Structure from motion problem
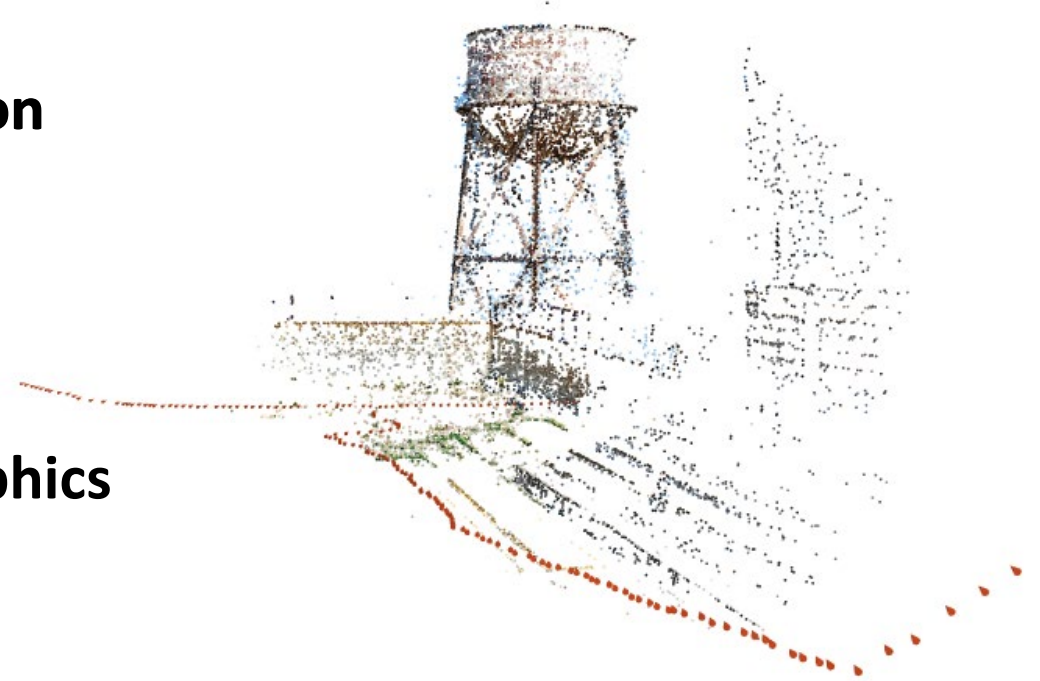
Moran*,Koslowsky*, Kasten,
Maron, Galun, Basri, ICCV, 2021

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

Co

Co

cl
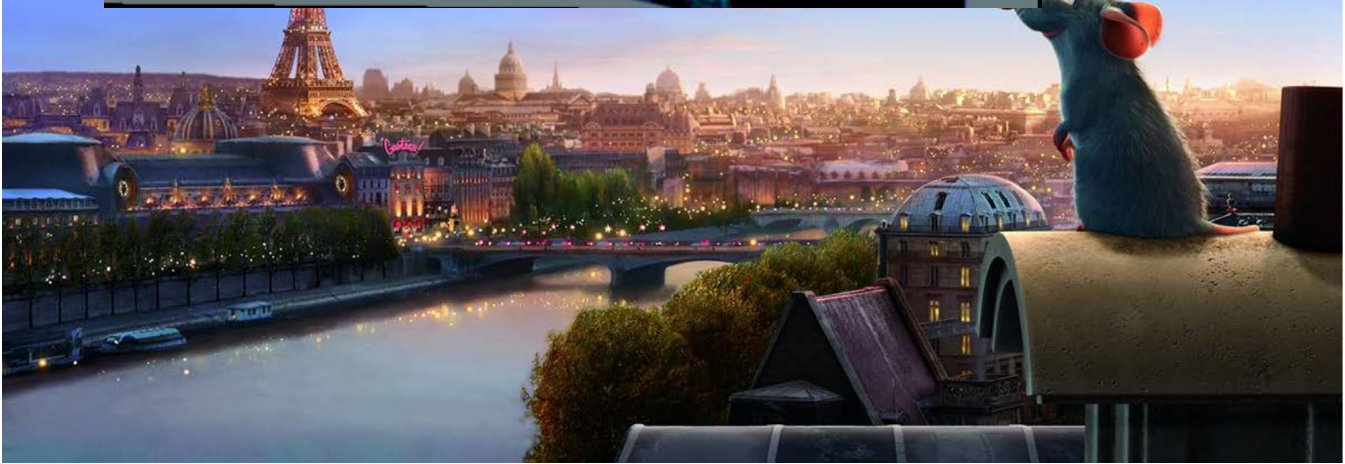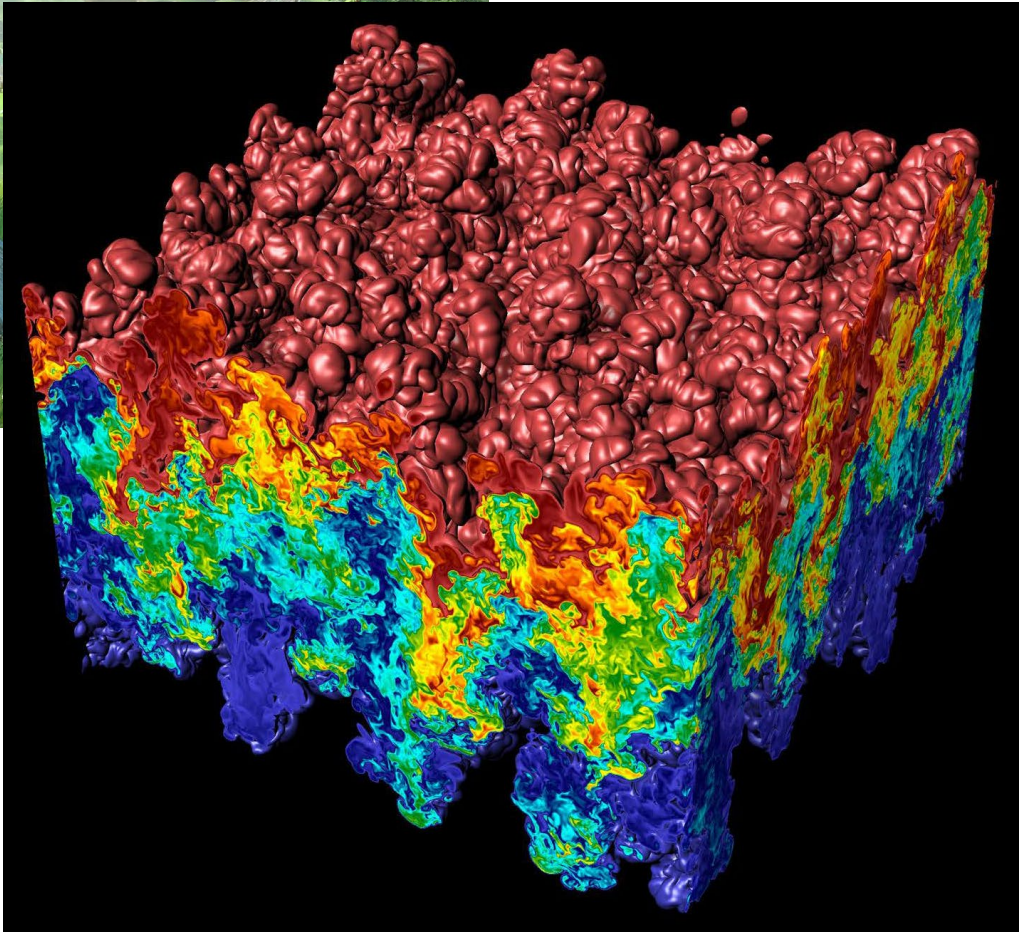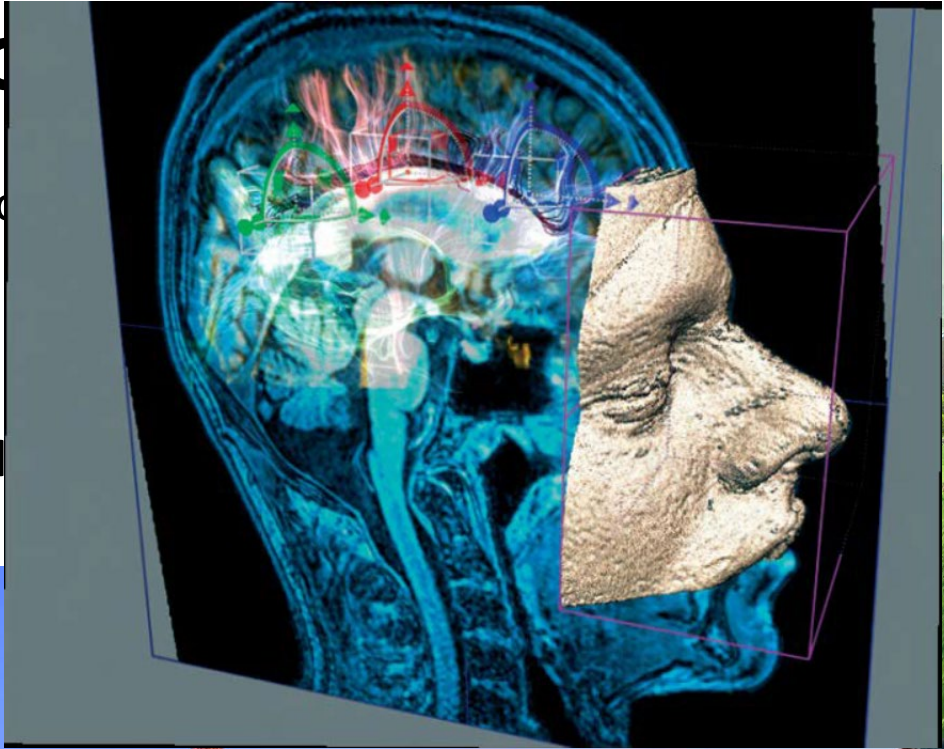
Rendering

The process of generating a photorealistic image from a 3D model
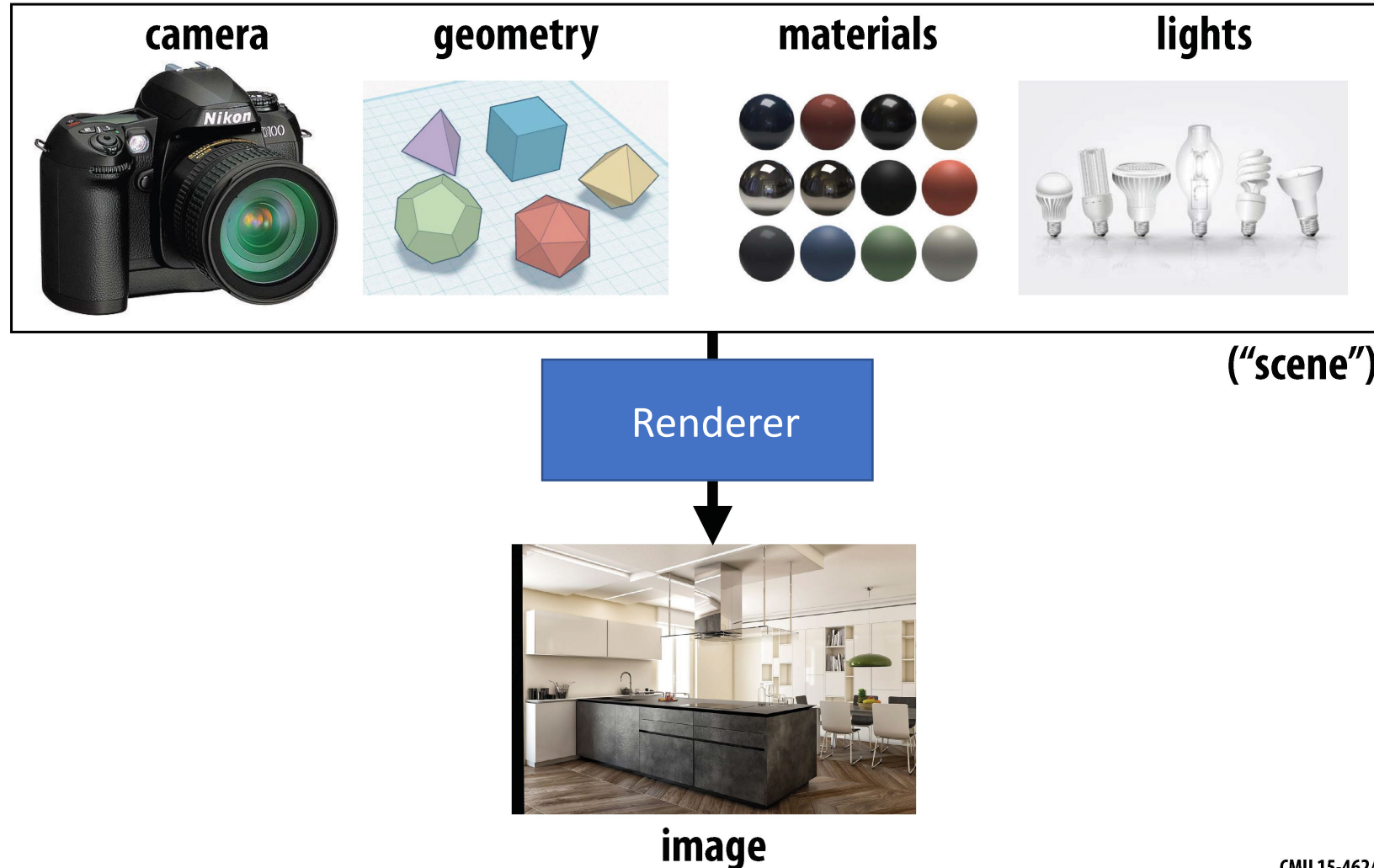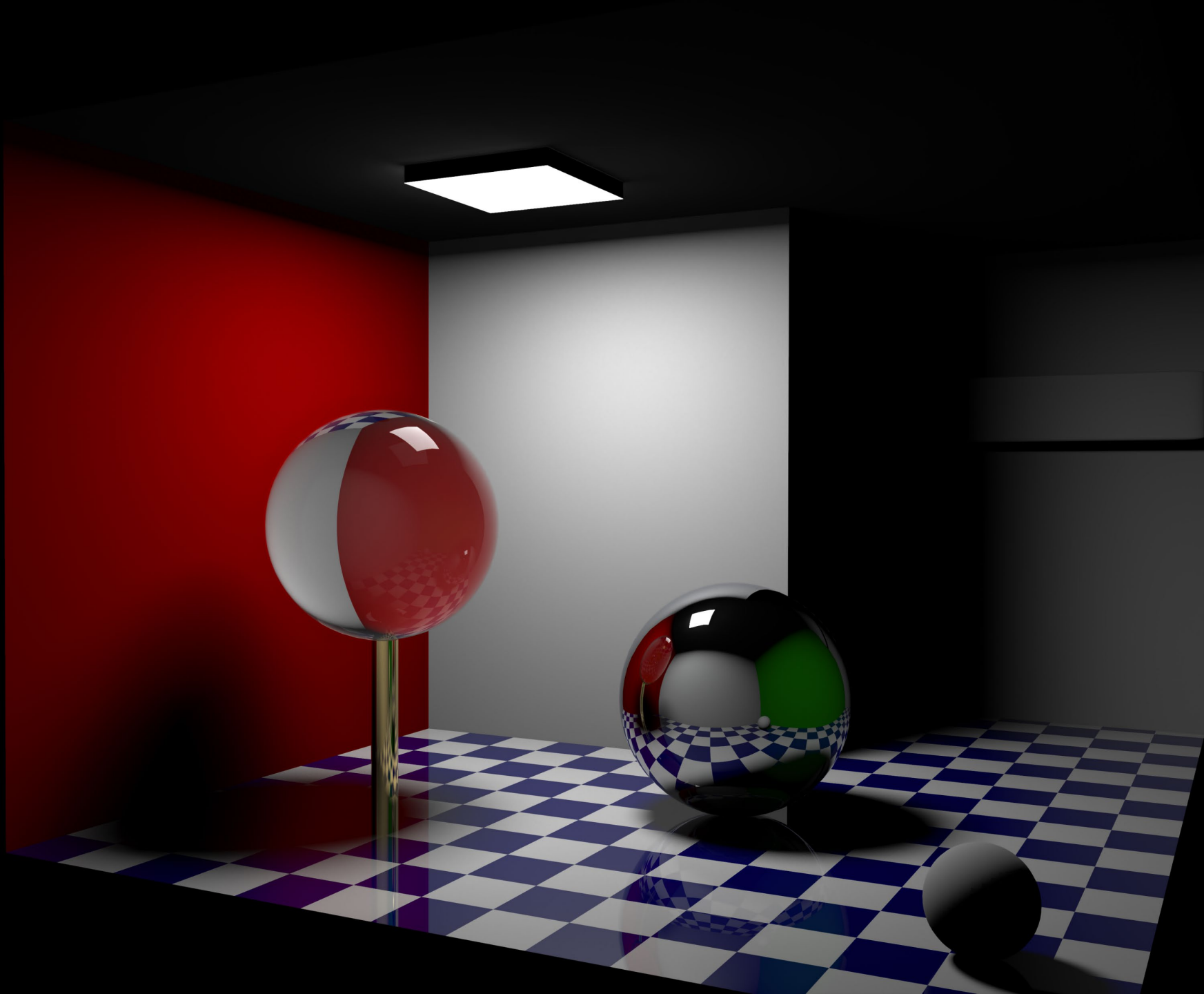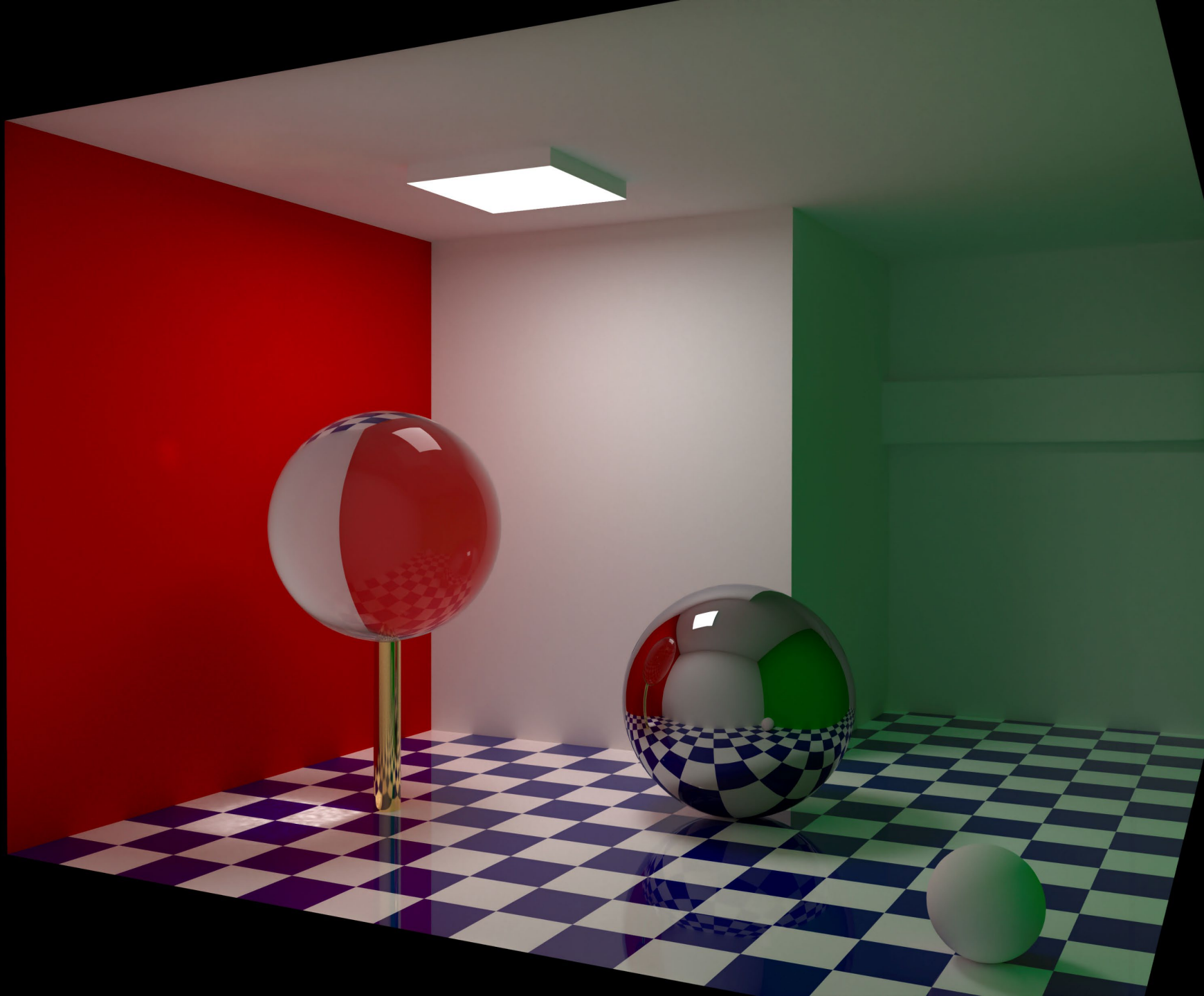
camera   geometry   materials   lights

("scene")

Renderer

image

CMU 15-462/662

WAIC

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera | geometry | materials | lights

("scene")

Rendering
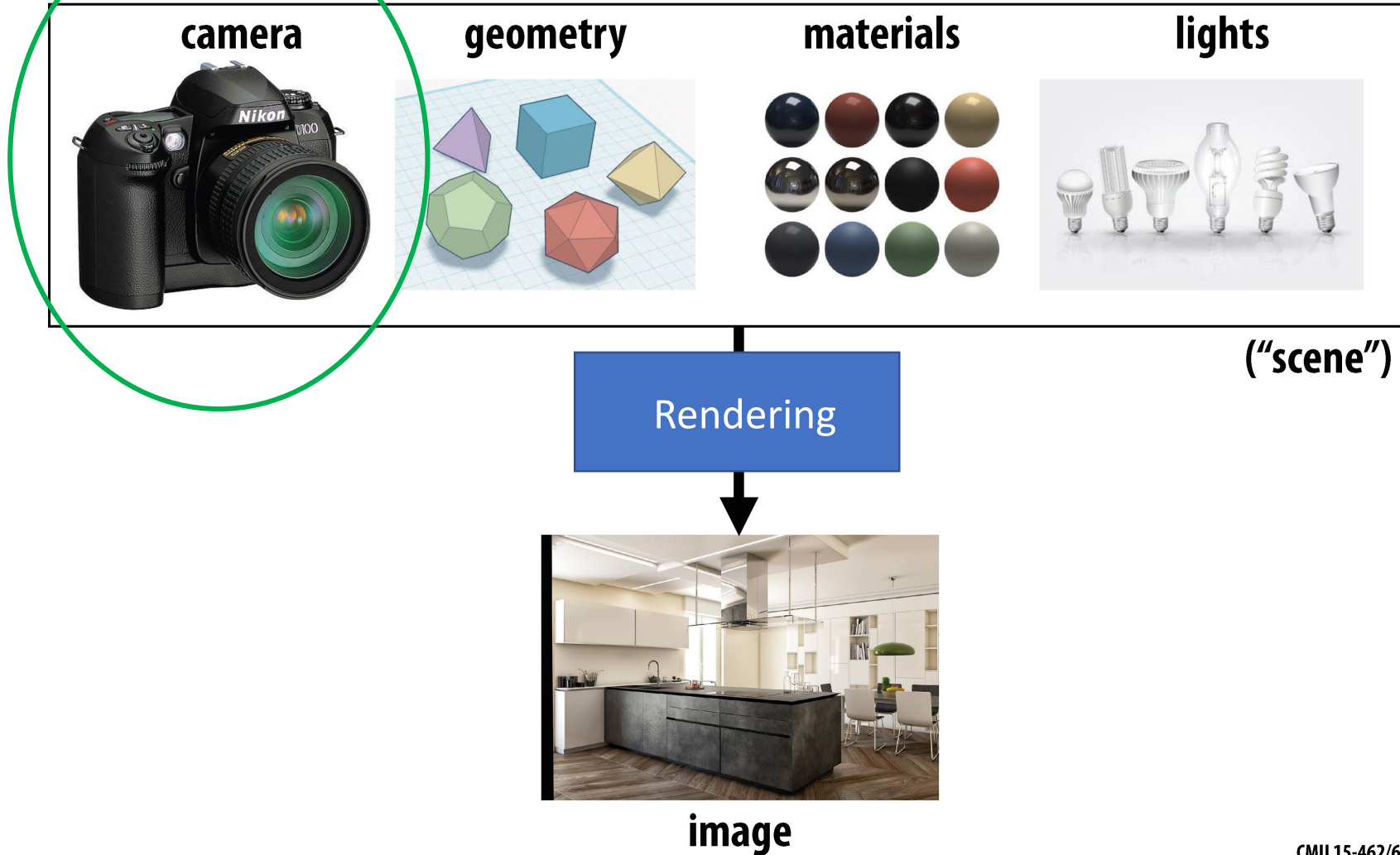
image

Perspective projection
Pinhole camera model

- Objects look smaller as they get further away
- Parallel lines "meet" at infinity
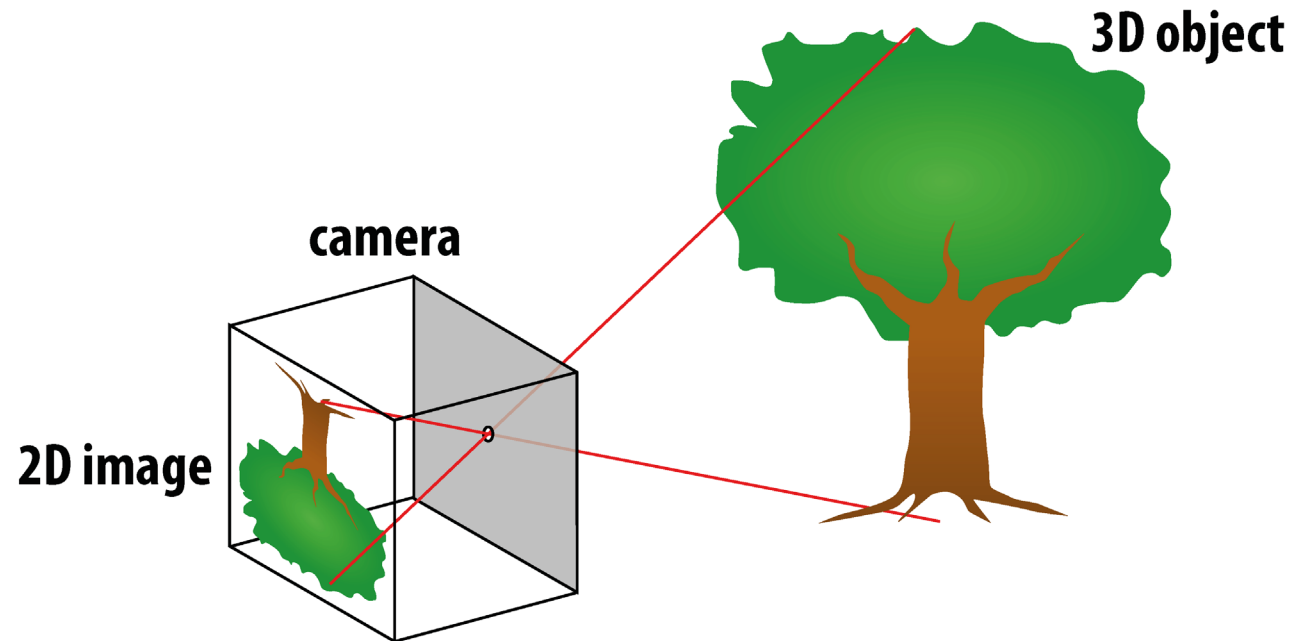
Perspective projection
Pinhole camera model

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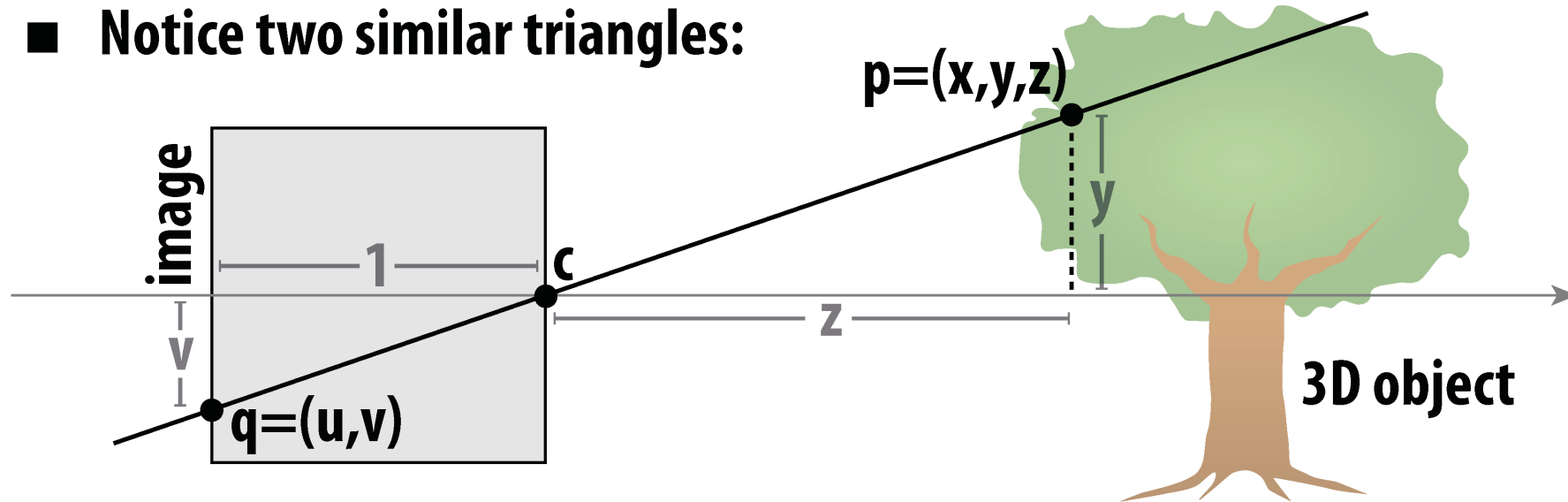
- Camera pinhole at $c = (0,0,0)$
- The image plane located $z = -1$
- Using similar triangles $v = \dfrac{y}{z}$ and $u = \dfrac{x}{z}$

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera   geometry   materials   lights
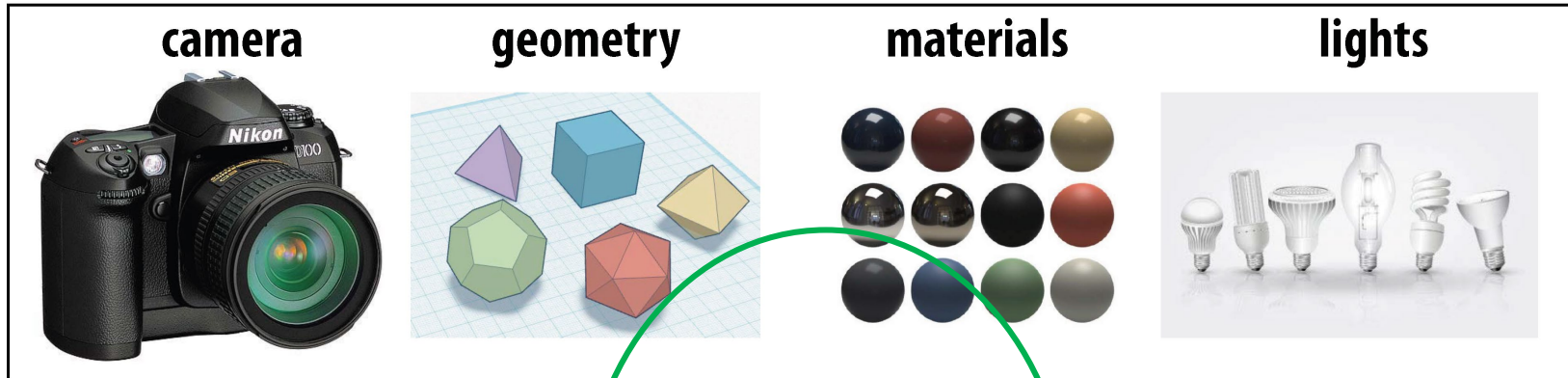
("scene")
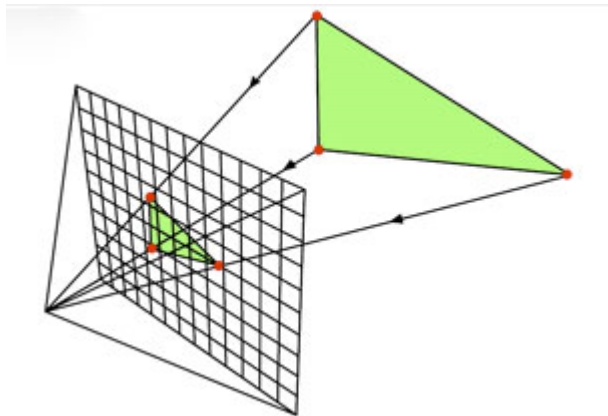
Rendering

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?

camera
origin

ray miss

ray hit

© www.scratchapixel.com
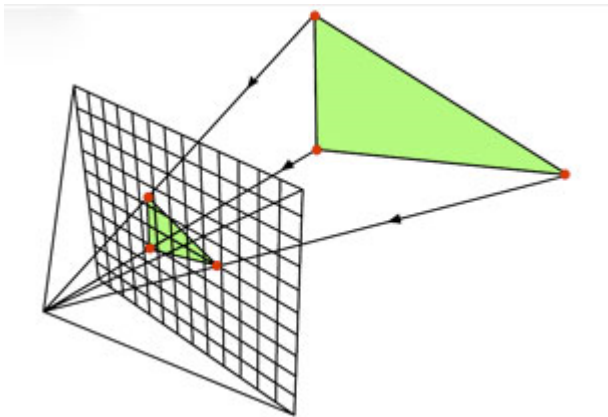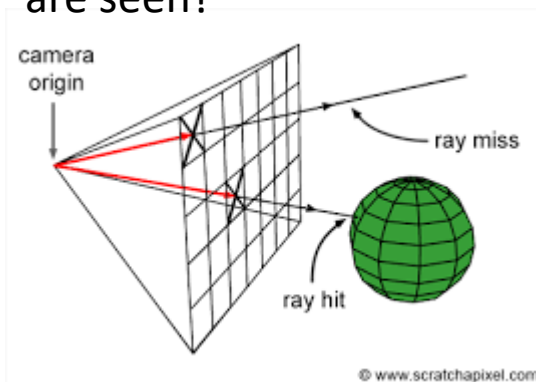
Everything is a Triangle
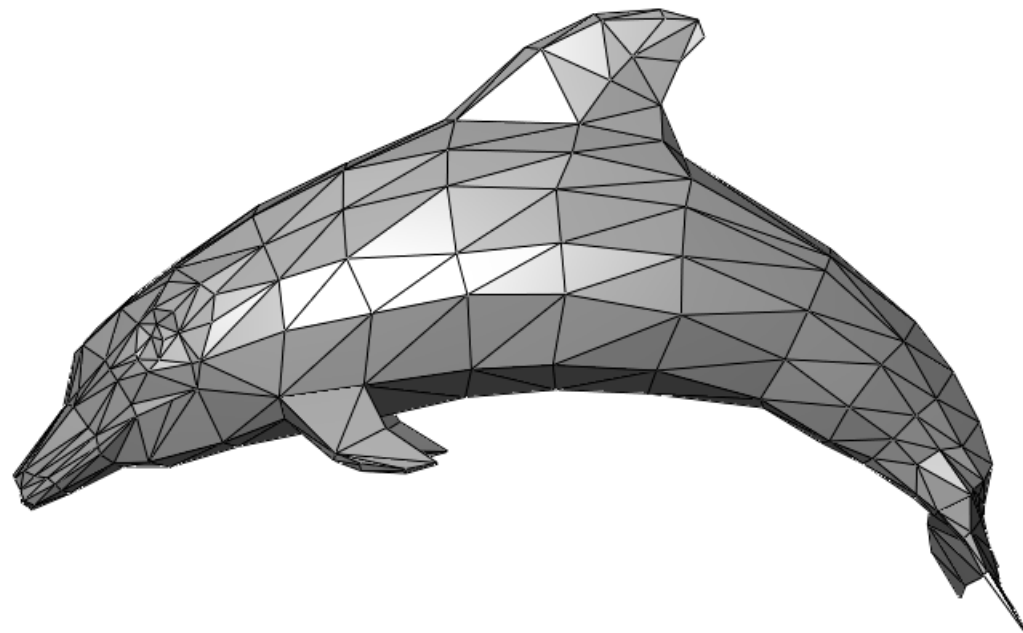
Rendering
Drawing on the screen (3D→2D)

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
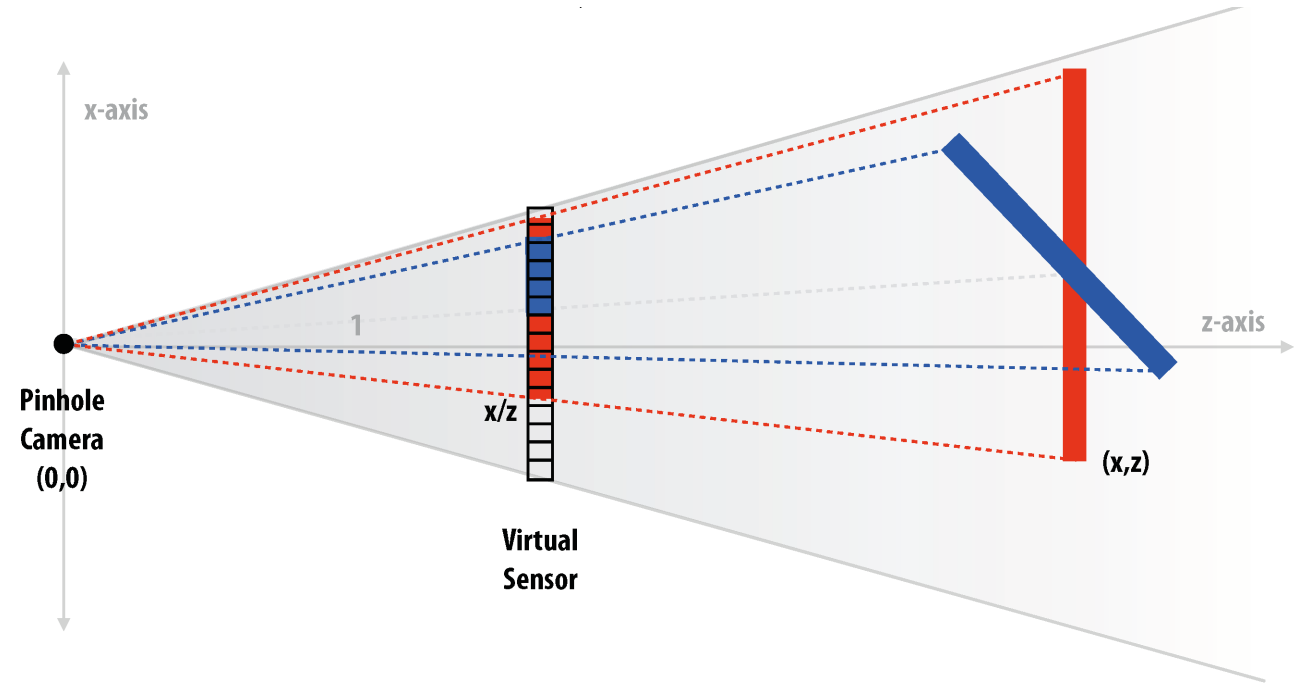- movies and video clips
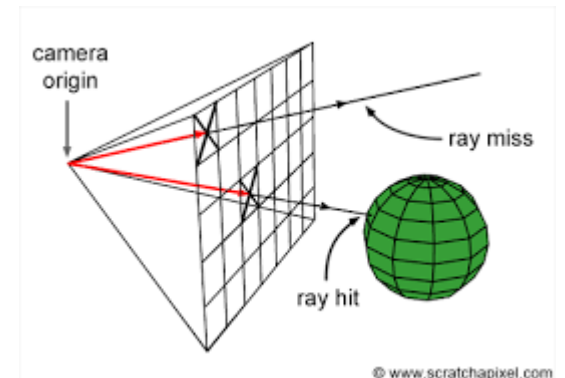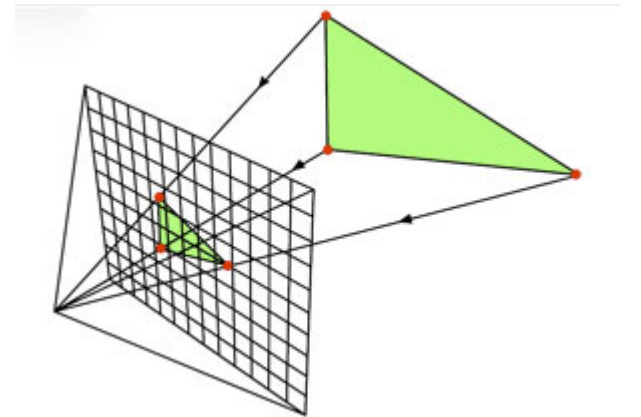
Rendering
Drawing on the screen (3D→2D)
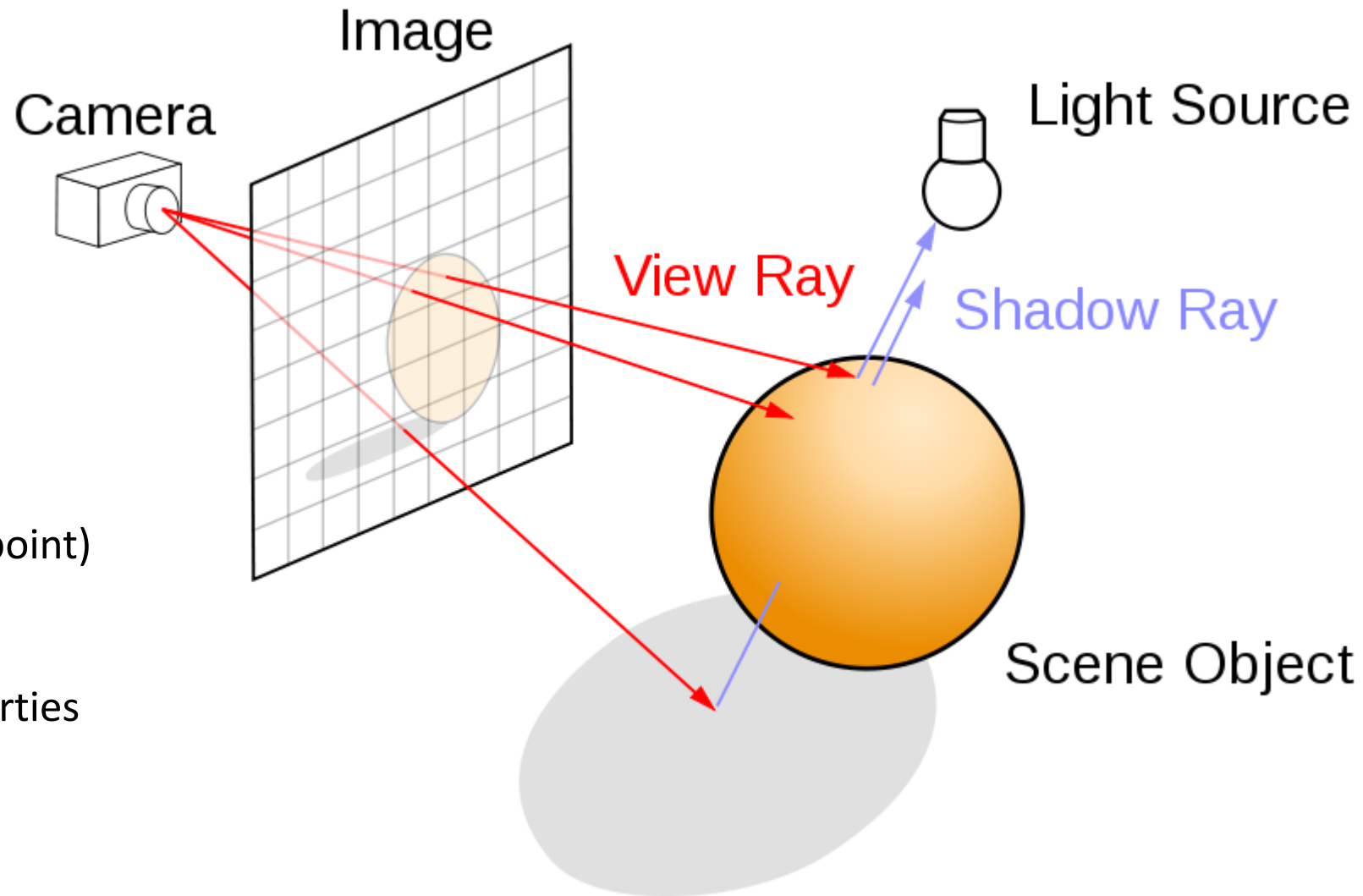
- camera (view point)
- light sources
- geometry
- material properties

Rendering
The visibility problem
ray tracing

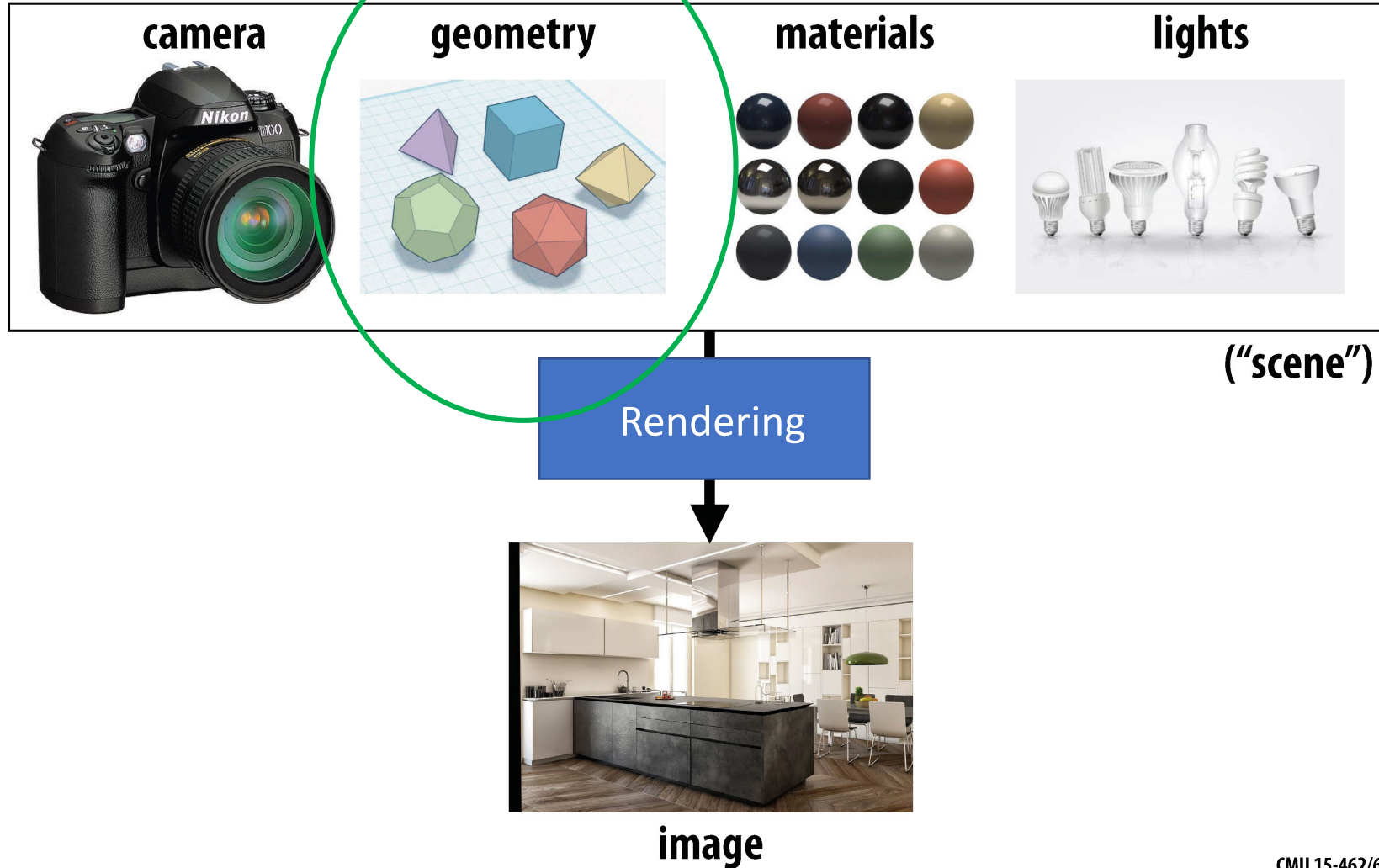- **Visibility problem in terms of rays:**

  - COVERAGE: What scene geometry is hit by a ray from a pixel through the pinhole?

  - OCCLUSION: Which object is the <u>first</u> hit along that ray?
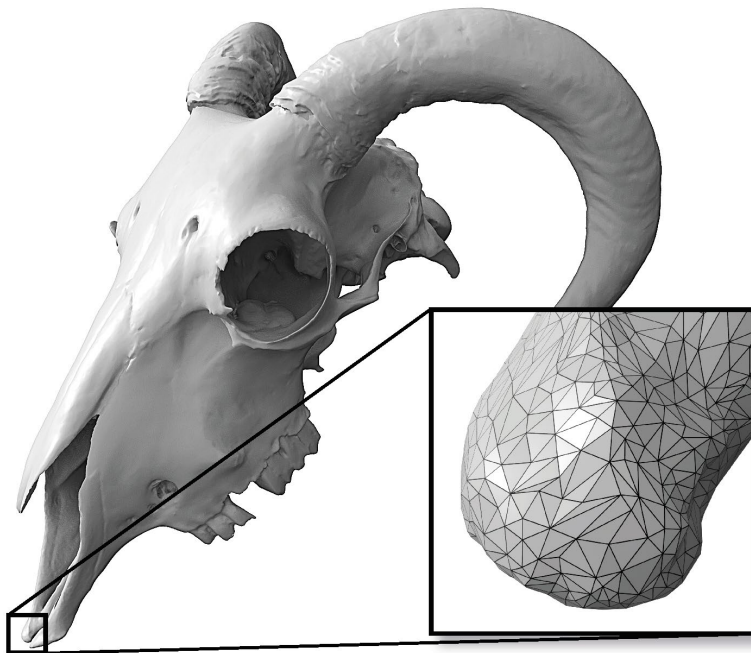
Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?
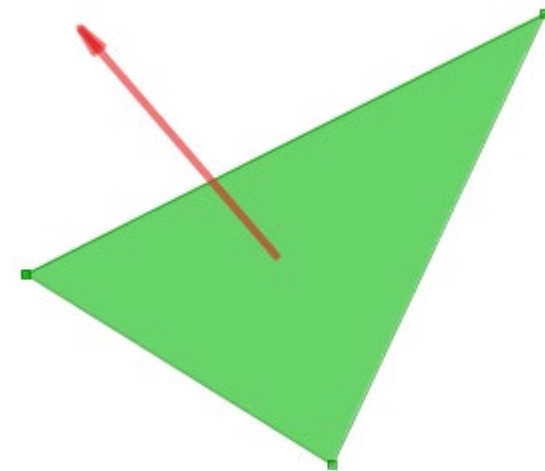

camera   geometry   materials   lights
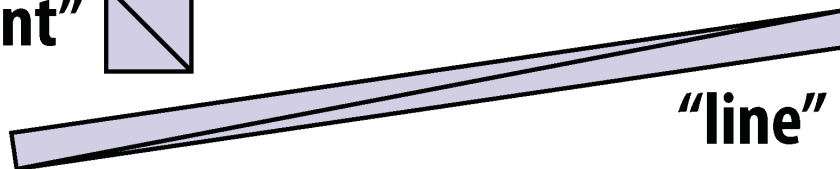
("scene")

Rendering

image

Geometry
Why triangles?

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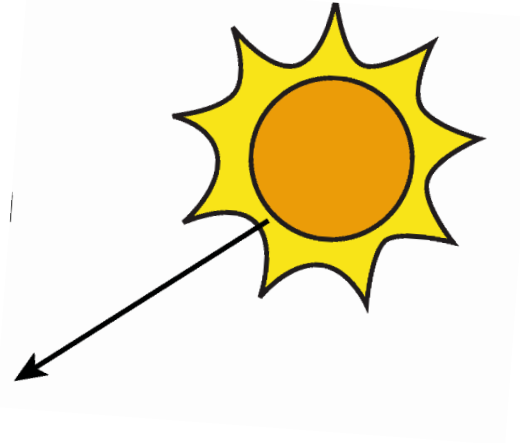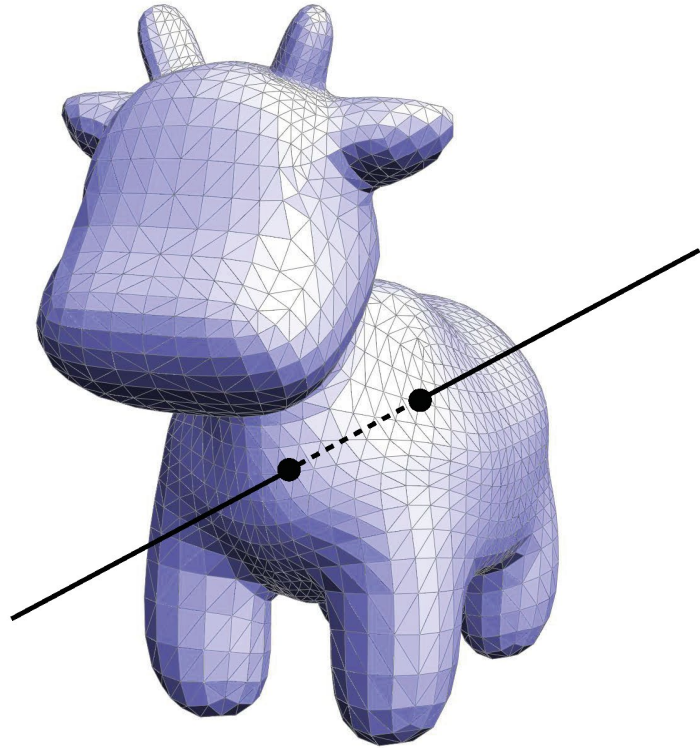- optimized and uniform drawing pipeline

"point"

"line"

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

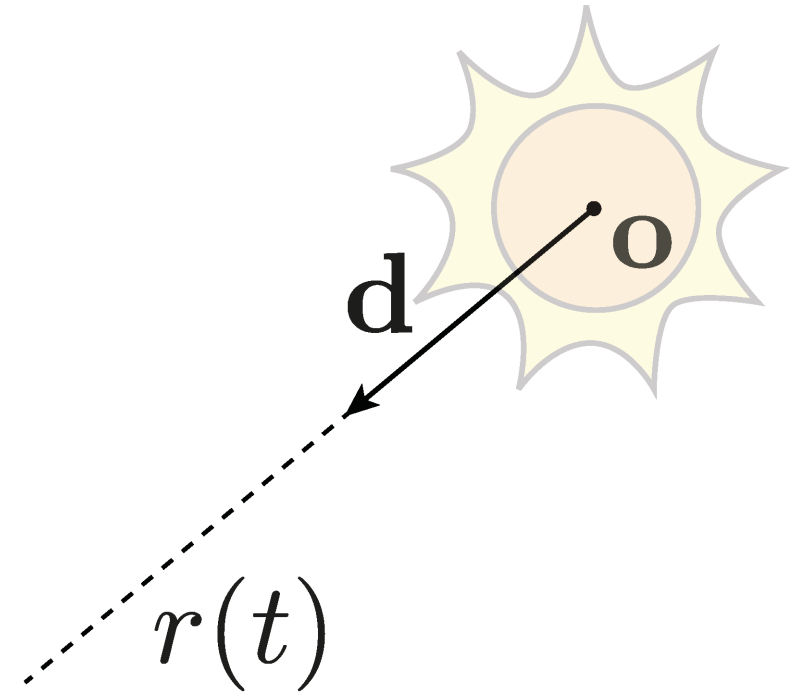- A significant step towards visibility and ray tracing

Geometry
Ray equation
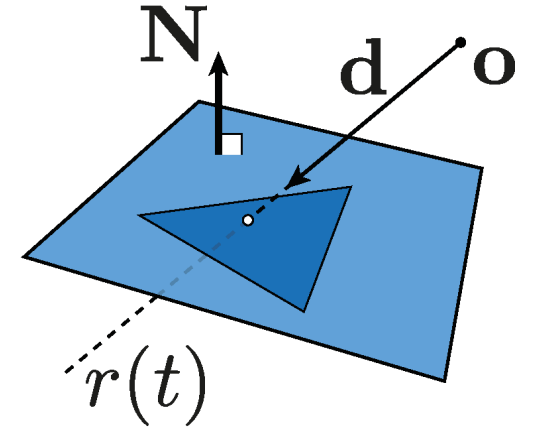
$$r(t) = o + td$$

Ray source

Unit
direction

Point along ray
parametrized by t

d

o
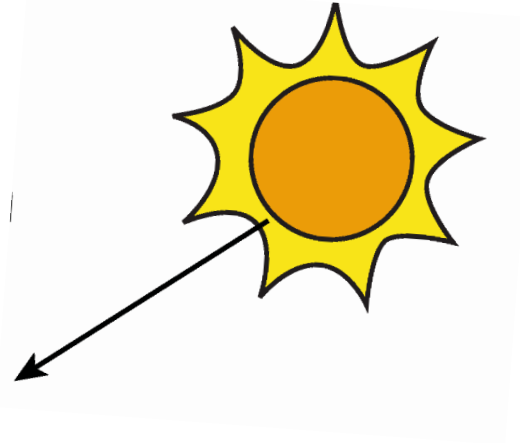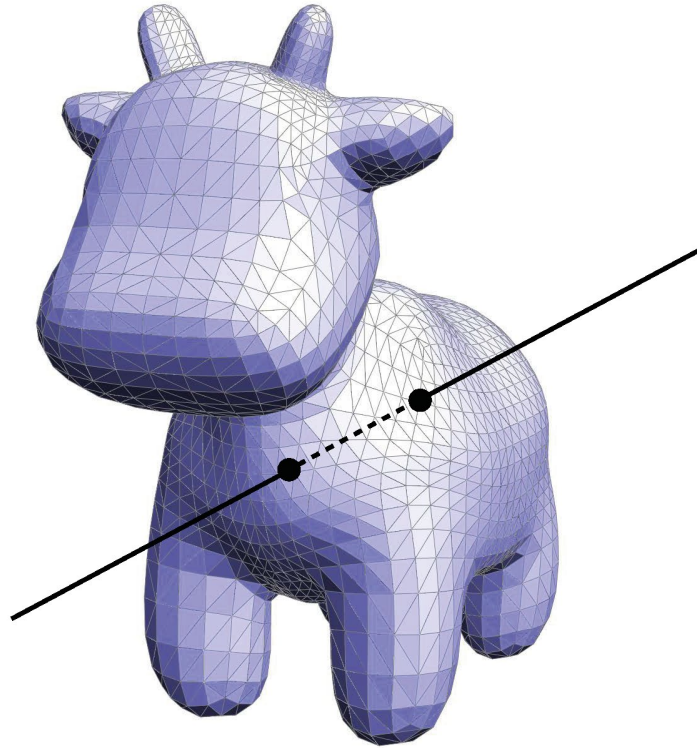
$r(t)$

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

Intersection between
plane $N^T x = c$
and
ray $r(t) = o + td$

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
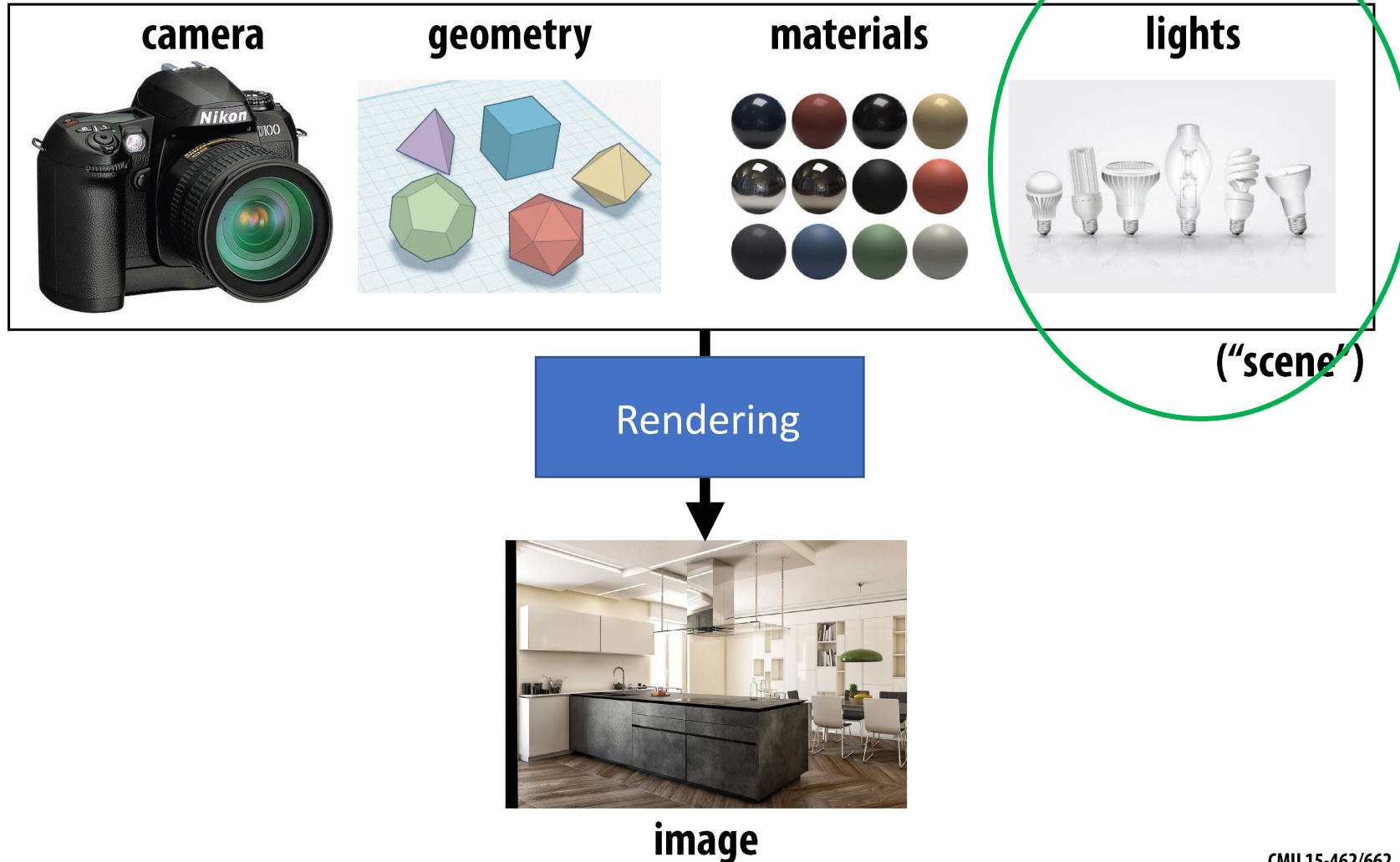- By hierarchical approach and dedicated hardware

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

CMU 15-462/662

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?

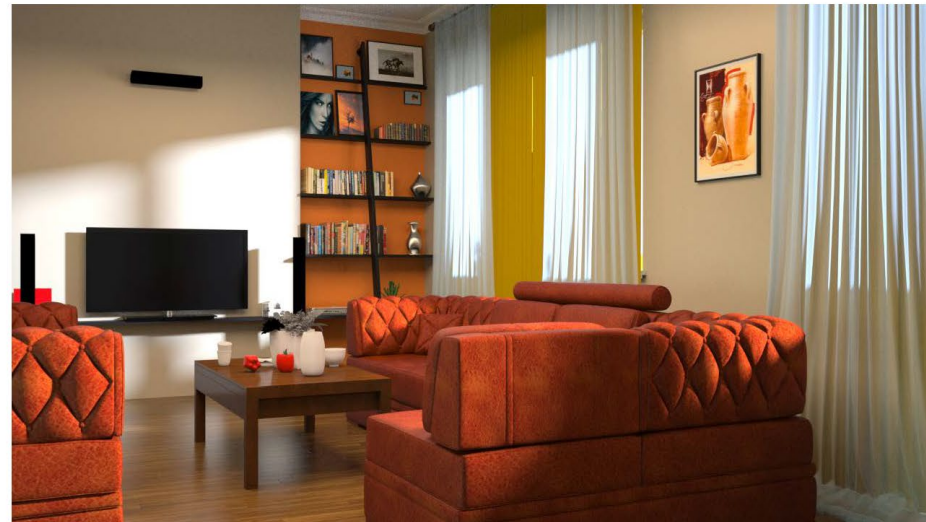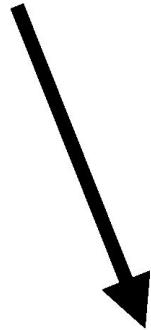- Final image = at every point, what color and how intense or bright it is

Rendering is more than just color!
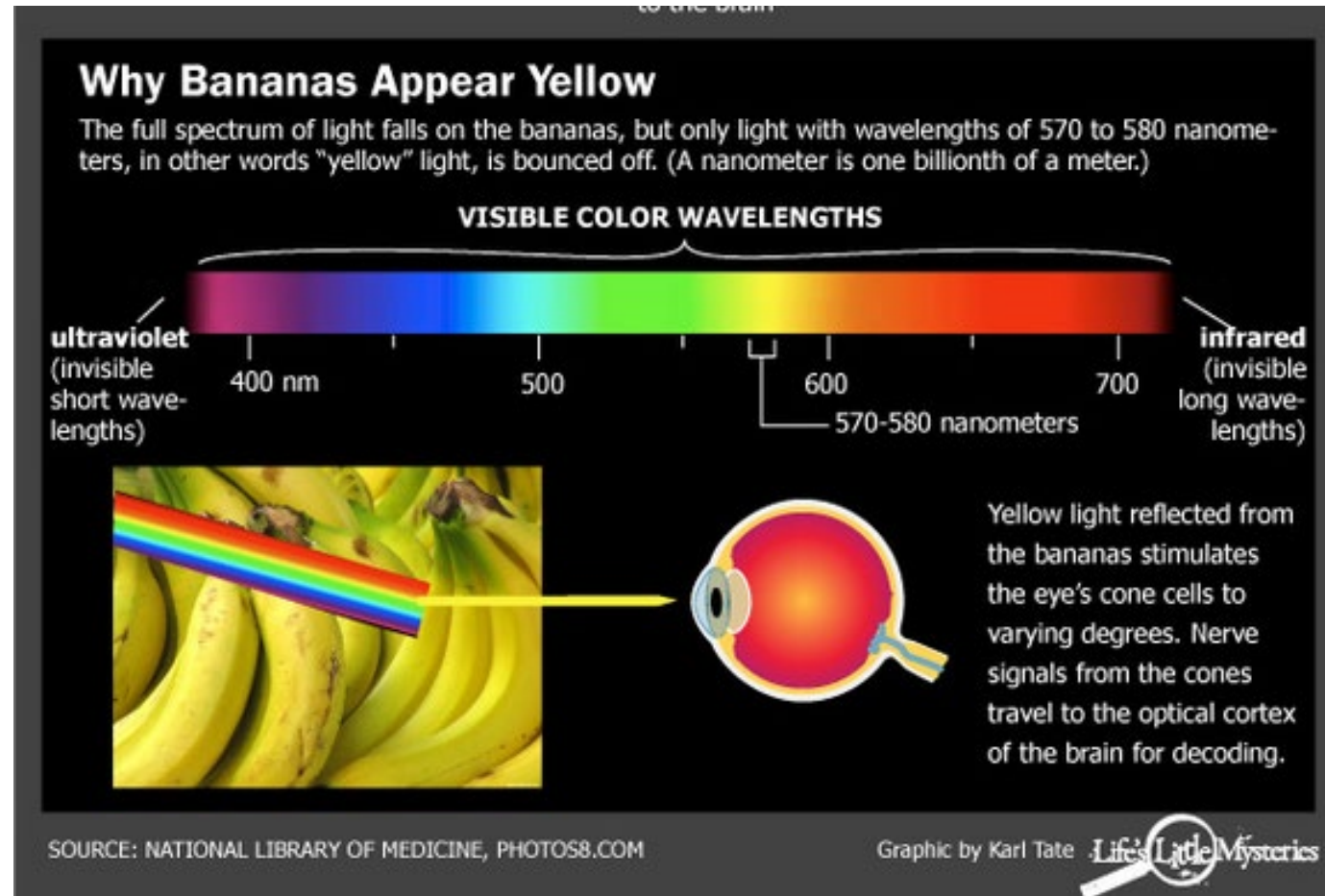
- **Also need to know _how much_ light hits each pixel:**

color

intensity
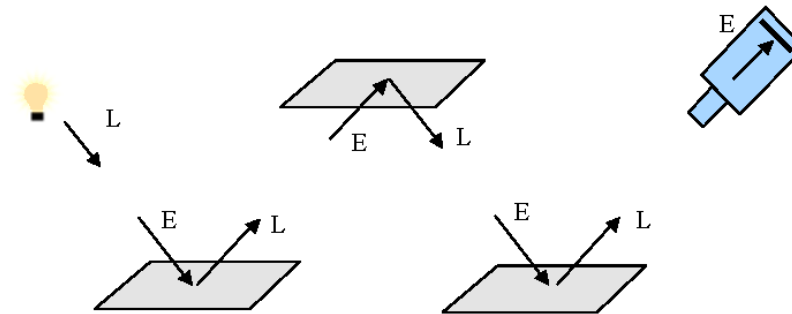

image

Radiometry
Electromagnetic radiation

Light is electromagnetic radiation that is visible to eye

Radiometry
Radiance and irradiance

Radiometry
Radiance and irradiance

Irradiance (E)
The measure of the radiant power incident on a surface per unit area

Radiance (L)
Radiance is the measure of the amount of radiant energy per unit solid angle <u>leaving or arriving at a point in space in a given direction</u>

Radiometry
Radiance and irradiance

<u>Key differences</u>

Directionality

<u>Radiance</u> includes information about the directional distribution of light while <u>Irradiance</u> represents the total power received by a surface

Units

- Irradiance is expressed in units of watts per square meter
- Radiance is expressed in units of watts per square meter per steradian
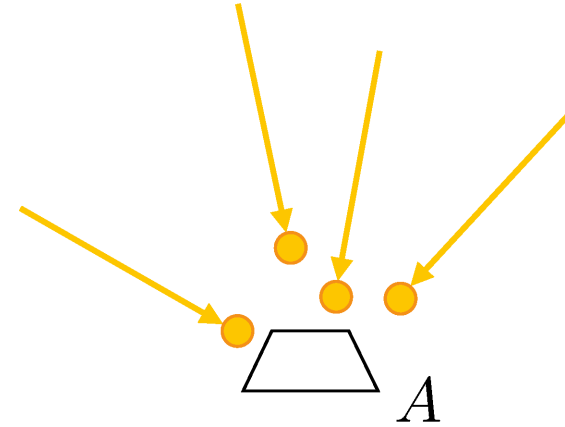
Radiometry
Irradiance

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

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

A

Given what we now know about radiant energy…

Why do some parts of a surface look lighter or darker?
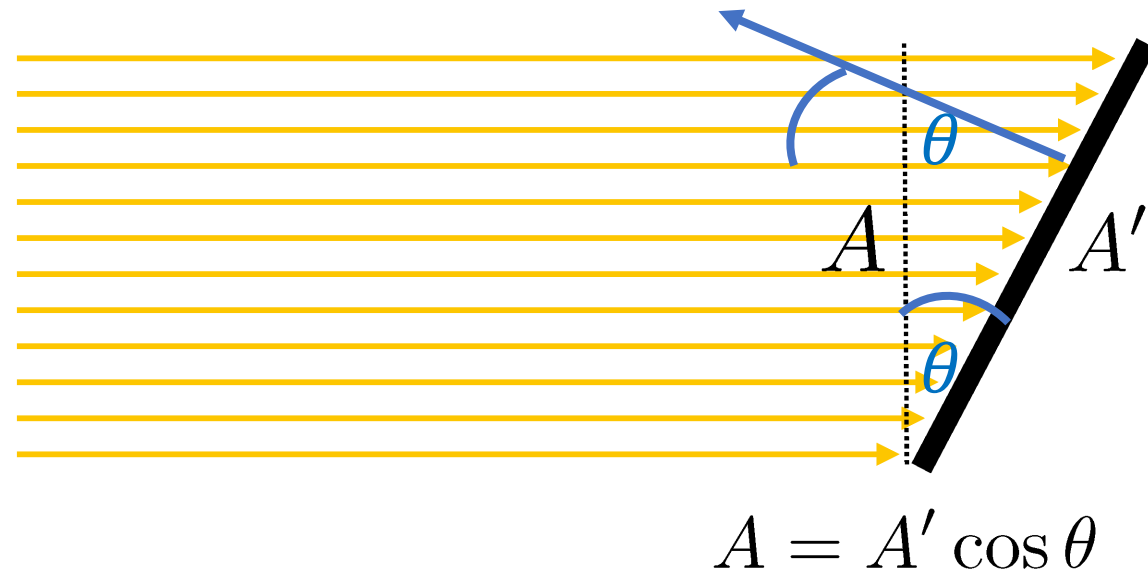
Radiometry
Lambert's cosine law

Consider beam with flux Φ incident on surface with area A

irradiance

radiant flux

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

$$\Phi = EA$$

A

Radiometry
Lambert's cosine law

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

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

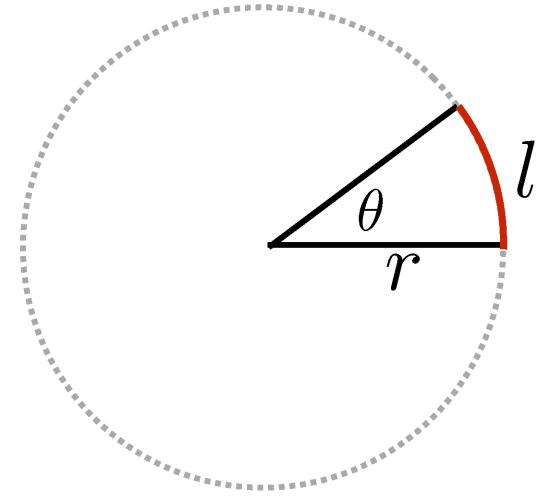$$E = \frac{\Phi}{A'} = \frac{\Phi \cos \theta}{A}$$

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**
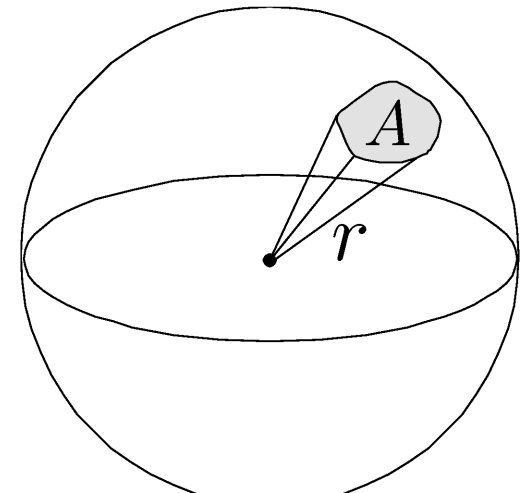
- $\theta = \dfrac{l}{r}$

- **Circle has 2π radians**

■ **Solid angle: ratio of subtended area on sphere to radius squared**

- $\Omega = \dfrac{A}{r^2}$

- **Sphere has 4π steradians**

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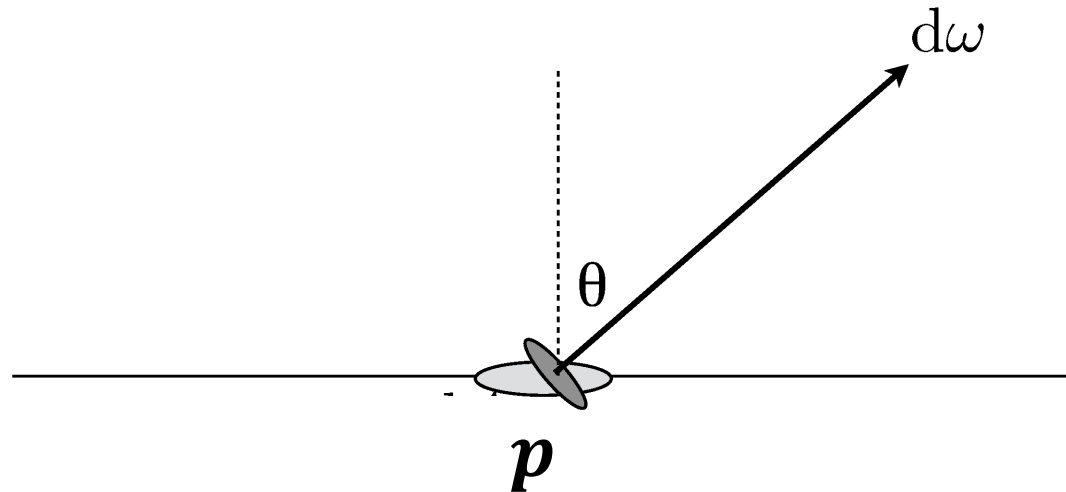
$$\left[\frac{\mathrm{W}}{\mathrm{m}^2\ \mathrm{sr}}\right]$$

Radiometry
Radiance

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

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

Radiance   Lambert's law   Solid angle
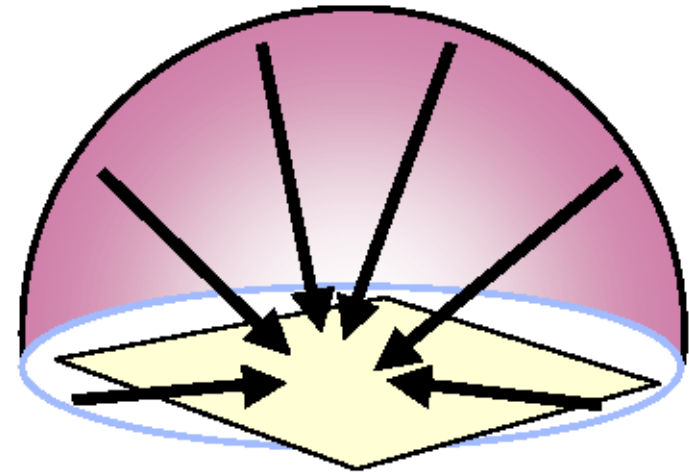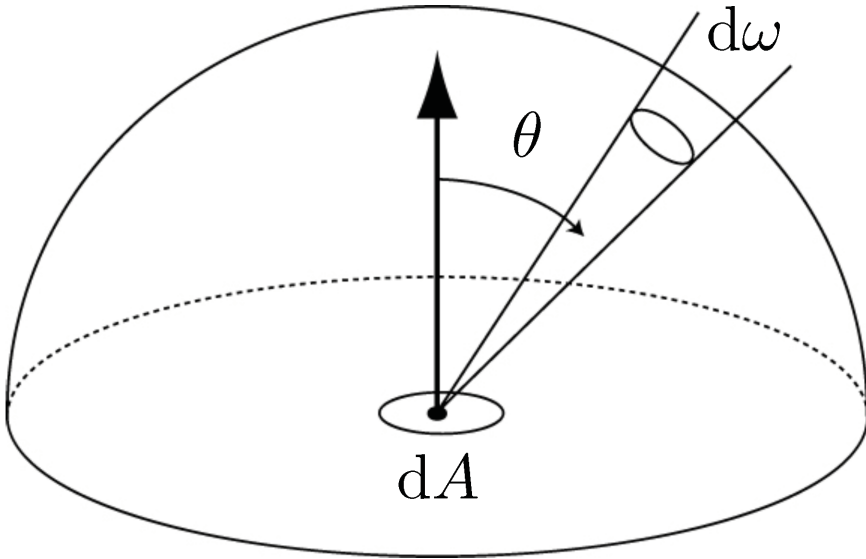
Radiometry
Radiance (Fields) properties

- 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
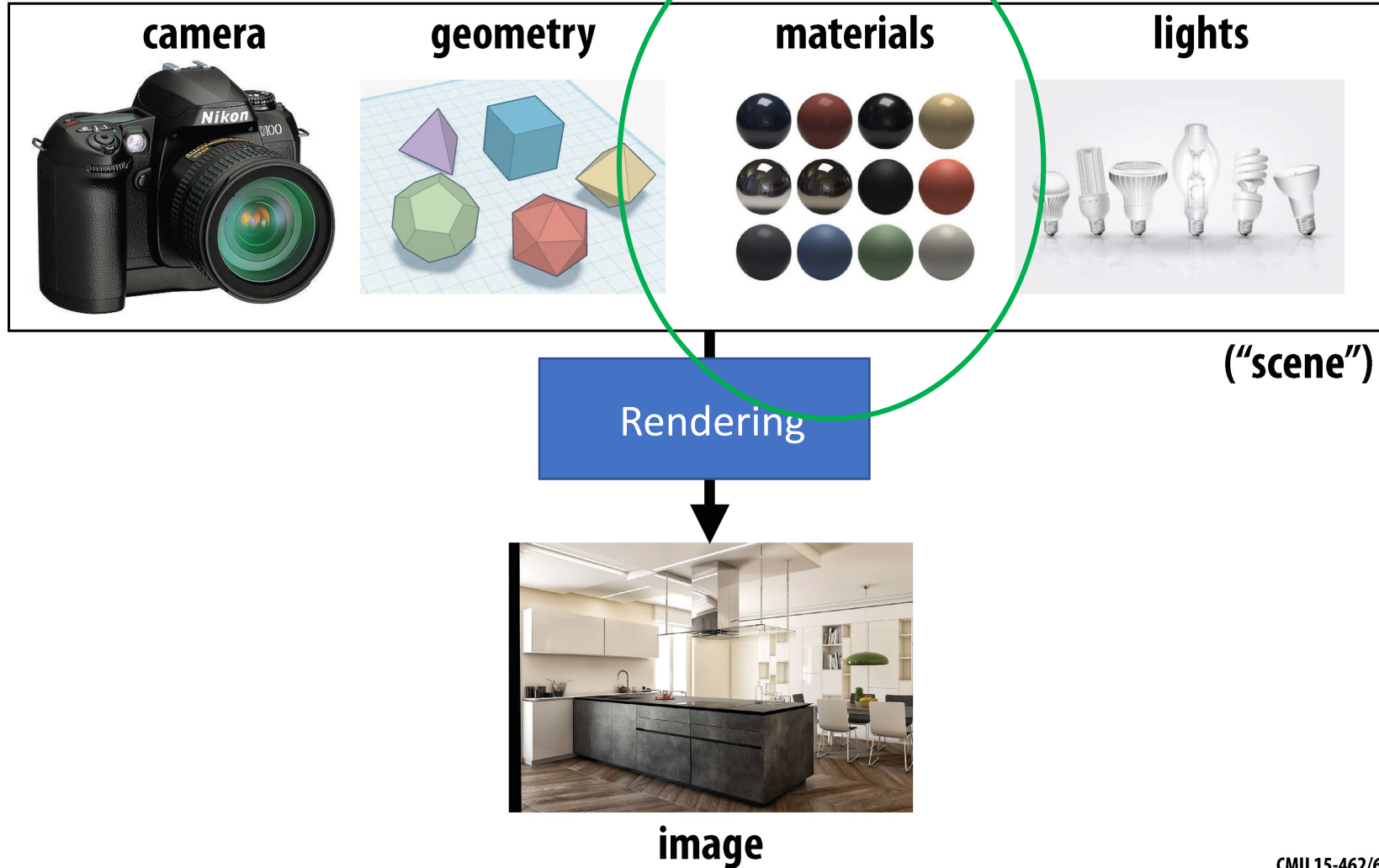## Irradiance from the environment

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

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

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]

WAIC
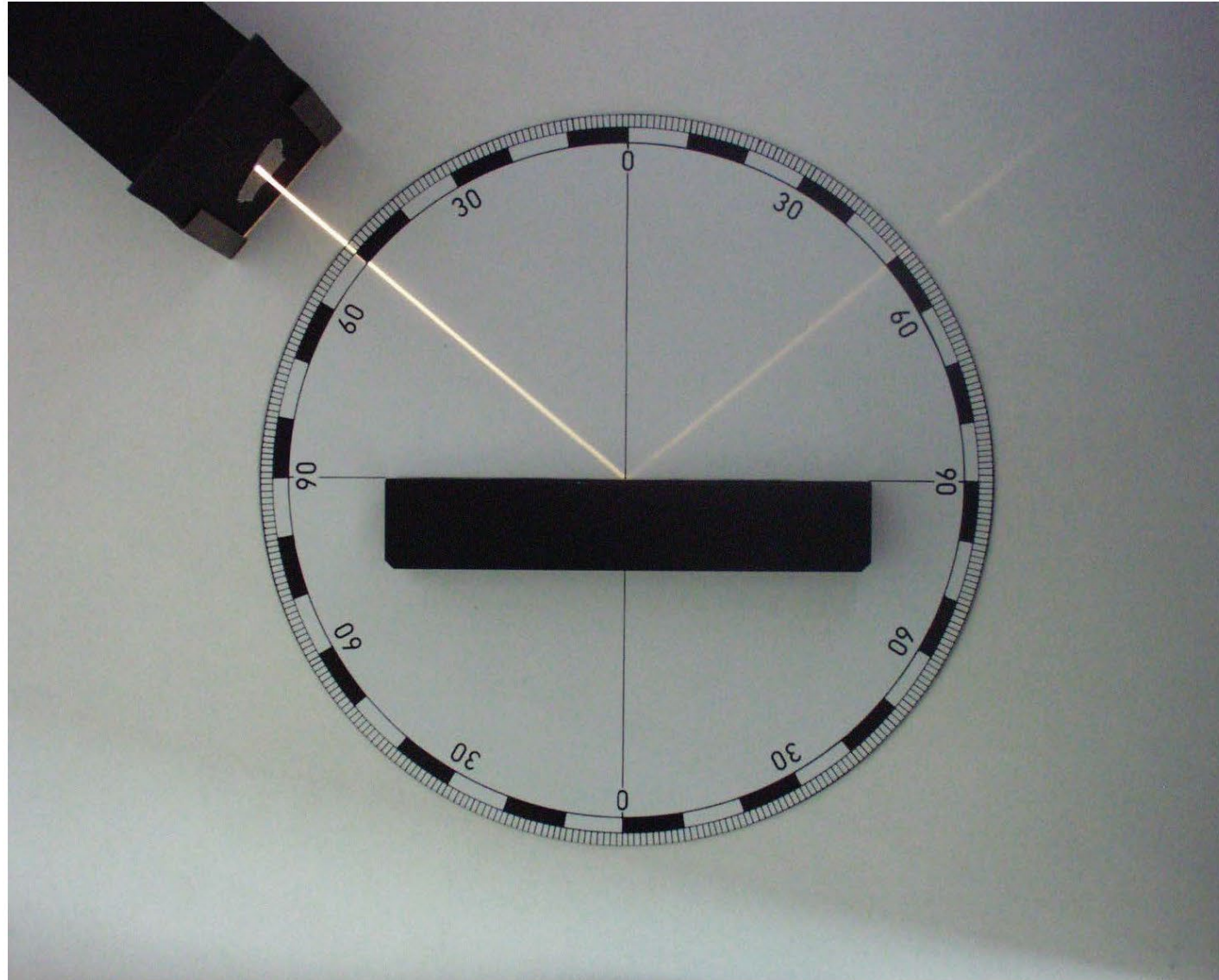
Radiometry
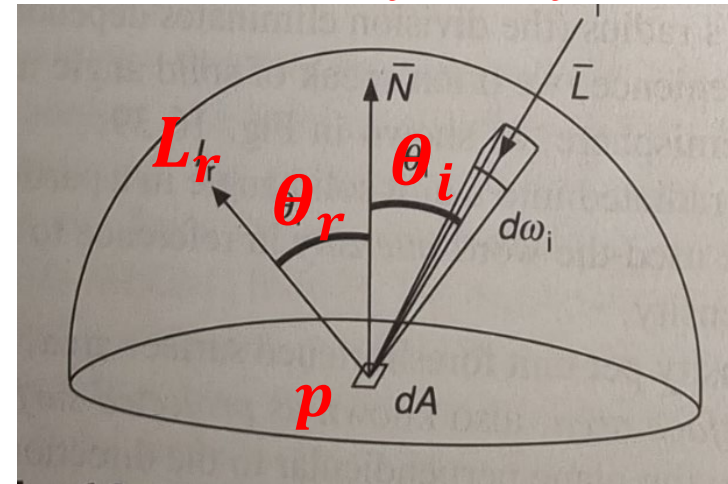## Reflected radiance and incident irradiance

The incident radiance $\quad L_i$

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

The reflected radiance $\quad L_r$

BRDF $= f(p, w_i, w_r) = \dfrac{reflected\ energy}{incident\ energy} = \dfrac{L_r}{E_i}$
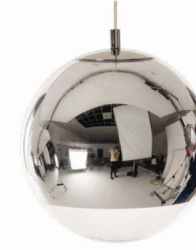
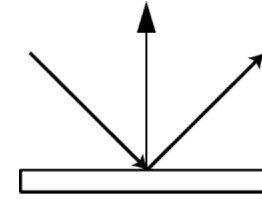$\left[\dfrac{1}{\text{sr}}\right]$

L_i

$E_i = L_i \cos \theta_i dw_i$
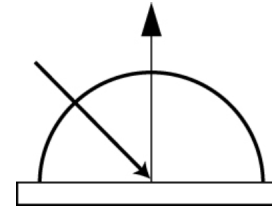
Some basic reflection functions
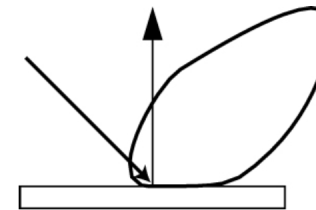
■ **Ideal specular**

    Perfect mirror

■ **Ideal diffuse**

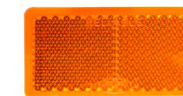   **Uniform reflection in all directions**

■ **Glossy specular**

    Majority of light distributed in reflection direction

■ **Retro-reflective**

    Reflects light back toward source
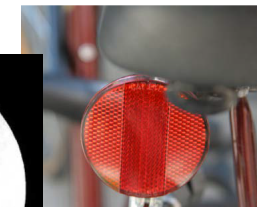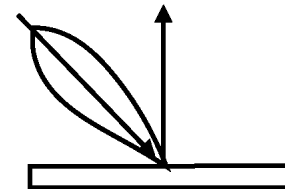
Diagrams illustrate how incoming light energy from given direction is reflected in various directions.

DL4CV Weizmann

Reflection equation
Multiple light sources
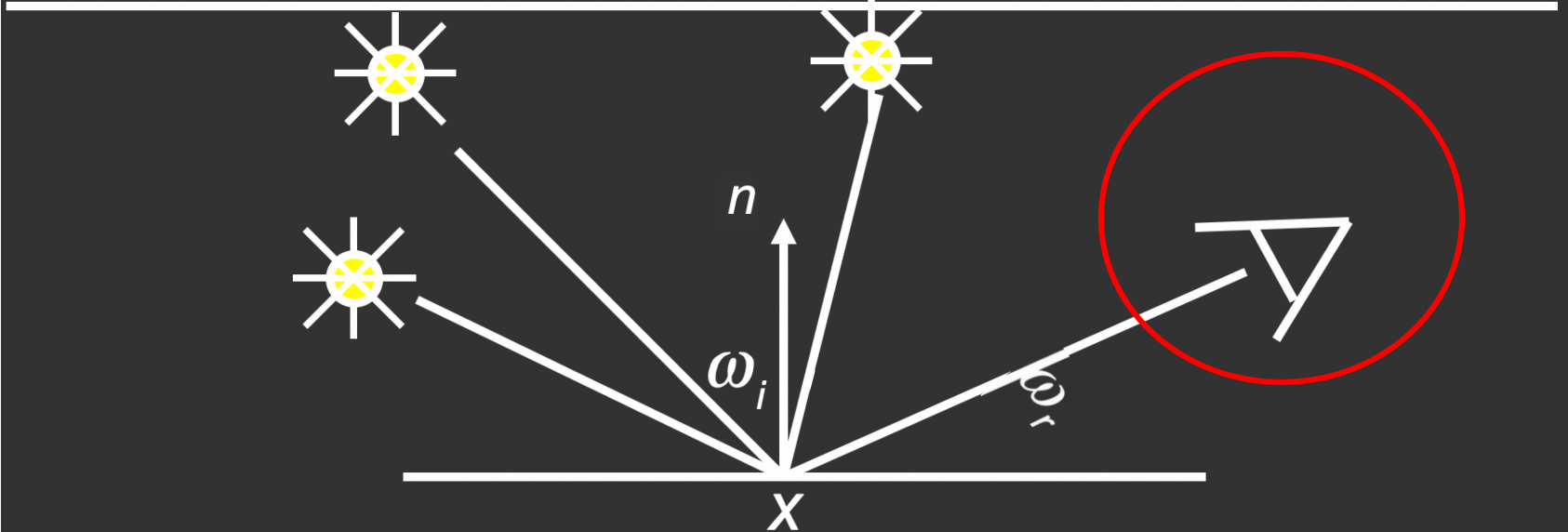
Sum over all light sources

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

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

Reflection equation
Environment of light sources

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

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

Reflection equation (local illumination)
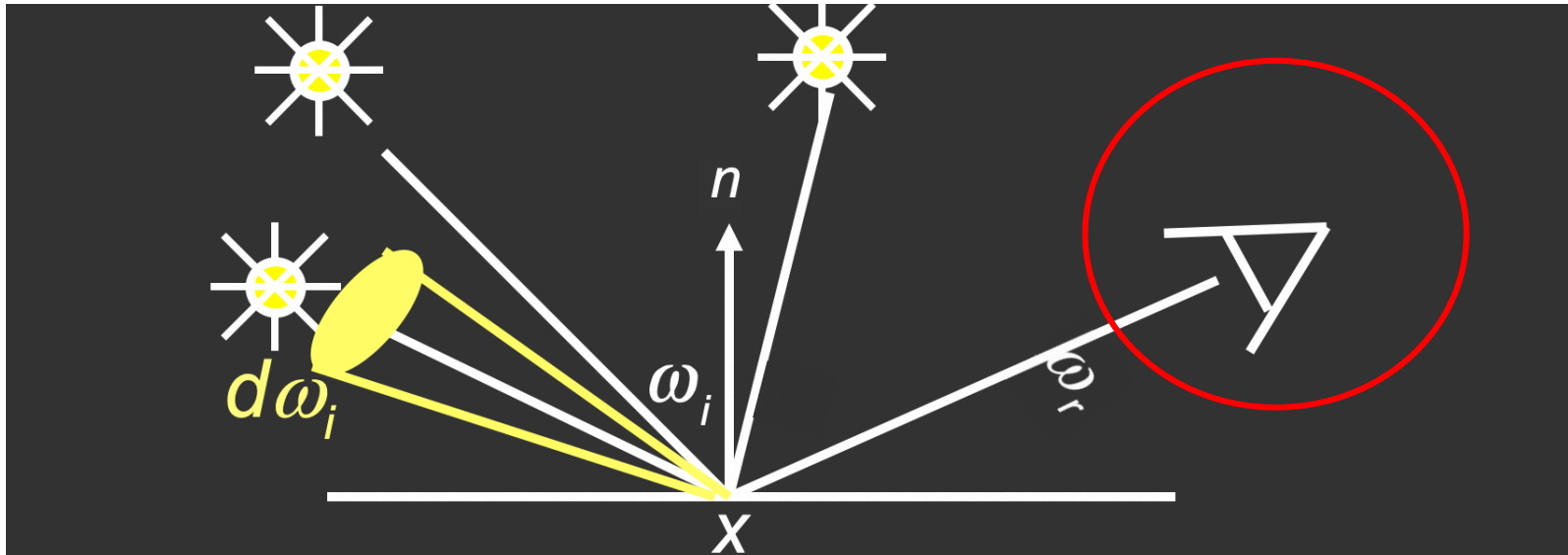## Recap

- 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)
Principles, James Kajiya, 1986

- For a given indoor scene, every object in the room must contribute illumination to every other object

- There is no distinction to be made between illumination emitted from a light source and illumination reflected from a surface

- The illumination coming from surfaces must scatter in a particular direction that is some function of the incoming direction of the arriving illumination, and the outgoing direction being sampled

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

Slide by Lior Yariv

Reflection equation

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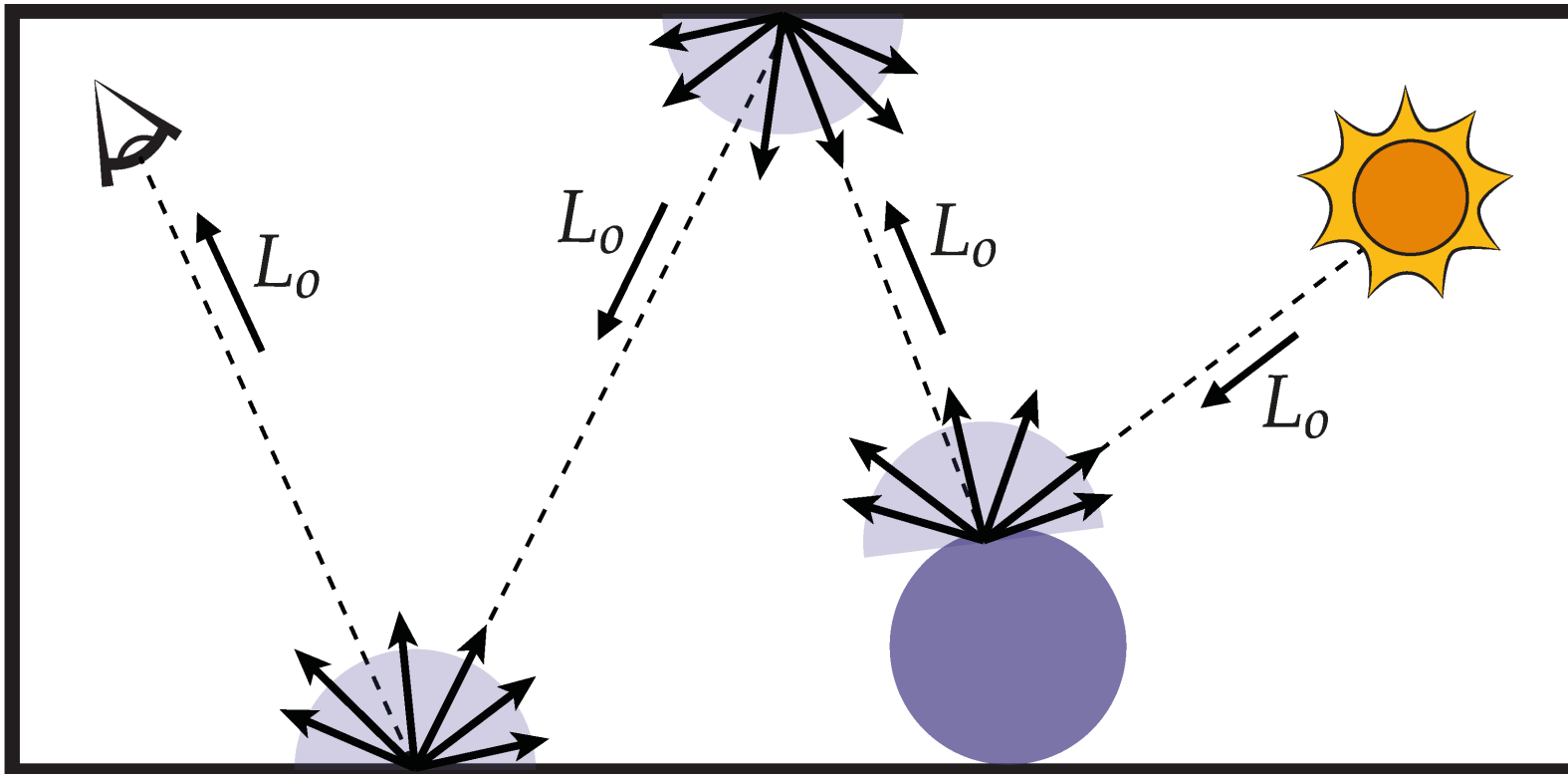
Rendering equation
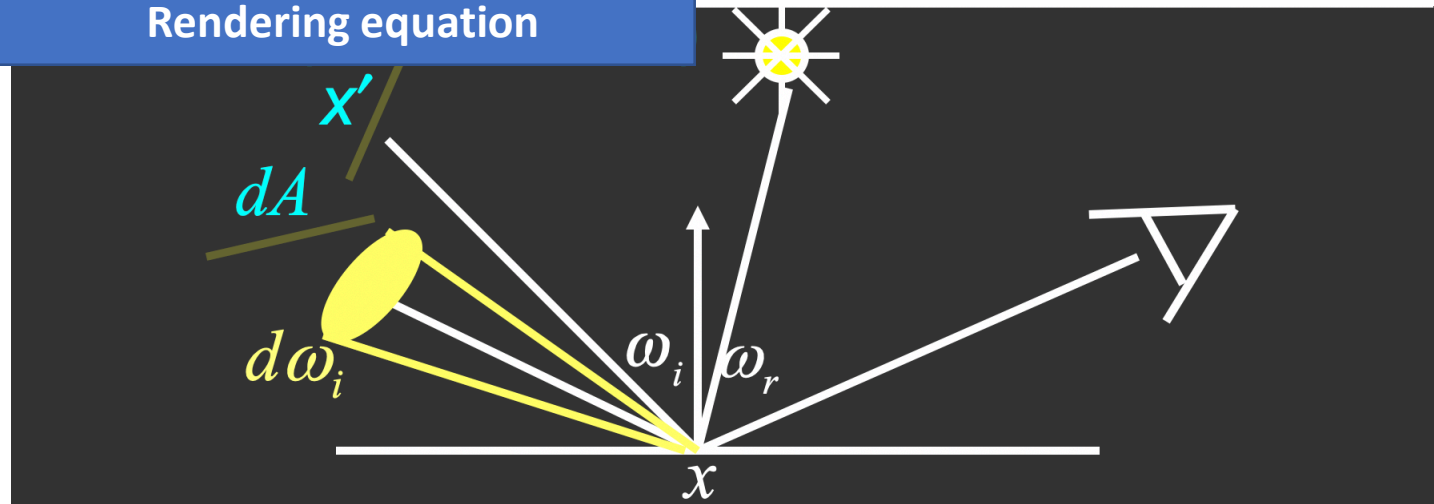
$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_\Omega L_r(x', -\omega_i) f(x, \omega_i, \omega_r) \cos\theta_i \, d\omega_i$$

| 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?
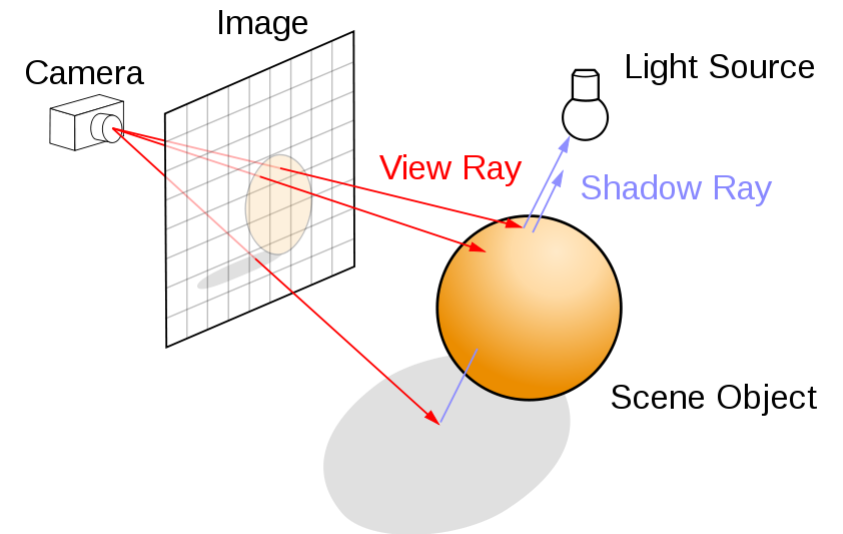
Noise decreases as the number of samples per pixel increases.

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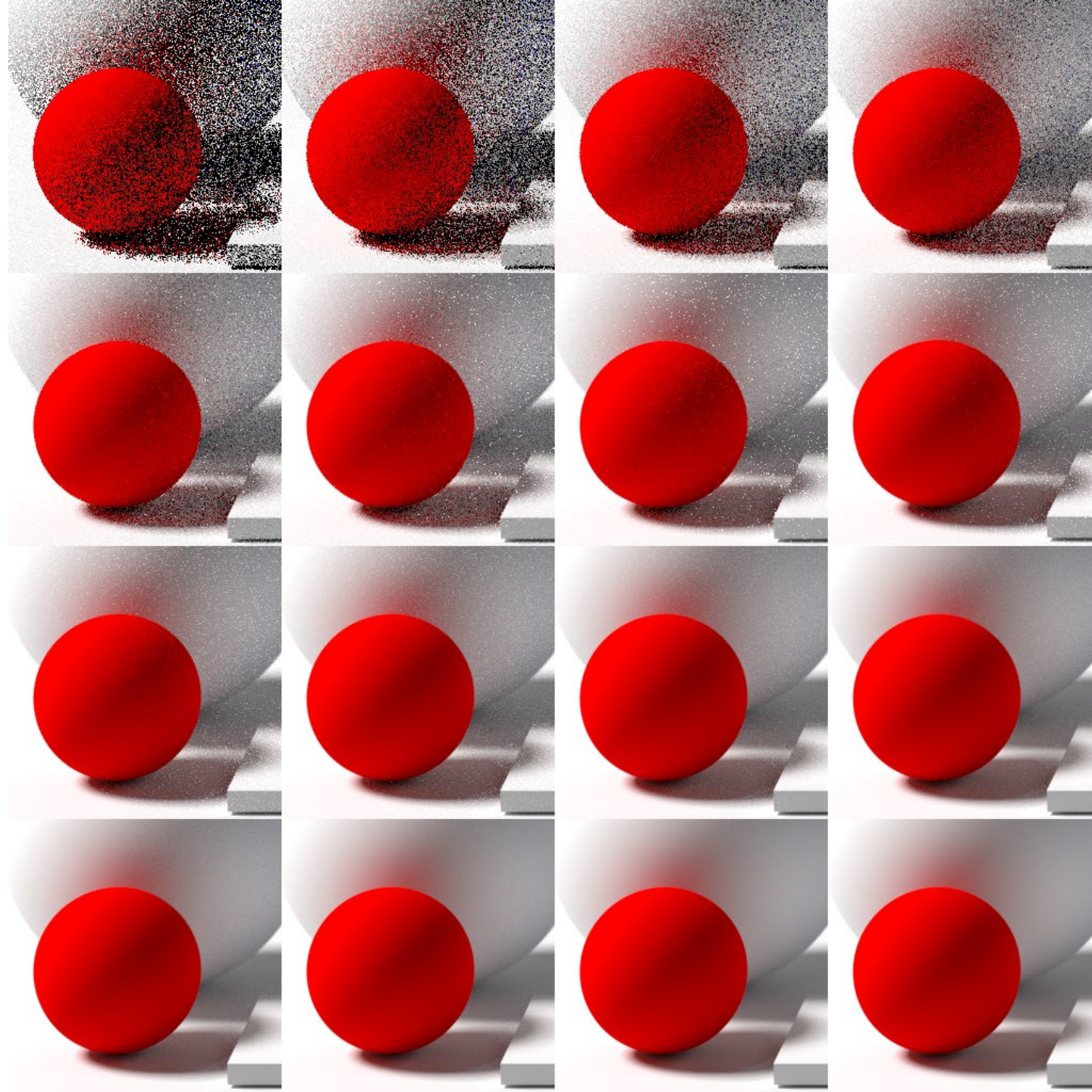
Direct illumination

$\bullet p$

One-bounce global illumination

Two-bounce global illumination

$\bullet p$

$\bullet p$

Four-bounce global illumination

Eight-bounce global illumination

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)