

Image Formation

Cameras and Color



Antonio Carta

2026-09-02

University of Pisa

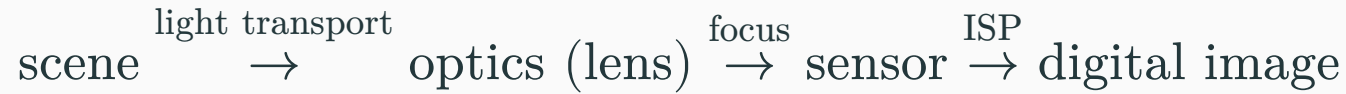
antonio.cart@unipi.it

- High-level understanding of image formation from **light leaving a surface** to **a pixel value in the image**: surfaces, lenses, sensors, color
- Mathematical modeling:
 - **linear-algebra view** of a camera, $l_s = Al_w$, and see why recovering the world from sensor data is an inverse problem
 - **geometric view** of a camera, perspective projection, and the simplified pinhole camera model (camera coordinates, no lenses)
- Discuss **Ambiguities** that arise from image formation:
 - perspective confounds size and depth
 - cameras discard information that must be recovered by (regularized) inversion
 - color confounds illuminant and material (plus our ambiguous perception of color)

- Light and surfaces
- Perspective projection and the pinhole camera model
- Cameras as linear systems
- Color

- Lecture 1 framed vision as an **inverse problem**: image $\rightarrow$ scene, the inverse of graphics (scene $\rightarrow$ image)
- Before solving the inverse problem, we need to understand the **forward model** precisely: how does a 3D scene actually become a 2D array of numbers?
- **CV tries to invert image formation**:
 - geometry: recovery of 3D structure from 2D images
 - filtering: removing noise, sharpening, deblurring
 - learned representations: disentangling material from illumination, finding invariant features, etc.

A photon's journey from the world to a pixel value:



- **Scene**: light sources, surfaces, materials, geometry
- **Optics**: lens focuses light onto the sensor plane (pinhole/perspective model, lenses)
- **Sensor**: converts photons to electrons to digital numbers, records one linear measurement per pixel
- **ISP** (image signal processor, not covered in depth today): demosaicing, white balance, gamma, ... turns the raw sensor reading into the RGB image you actually see

Today we focus on the geometry and physics that determine what information ends up in the final pixels. The main goal is understanding what information is lost in the final image.

Light and Surfaces

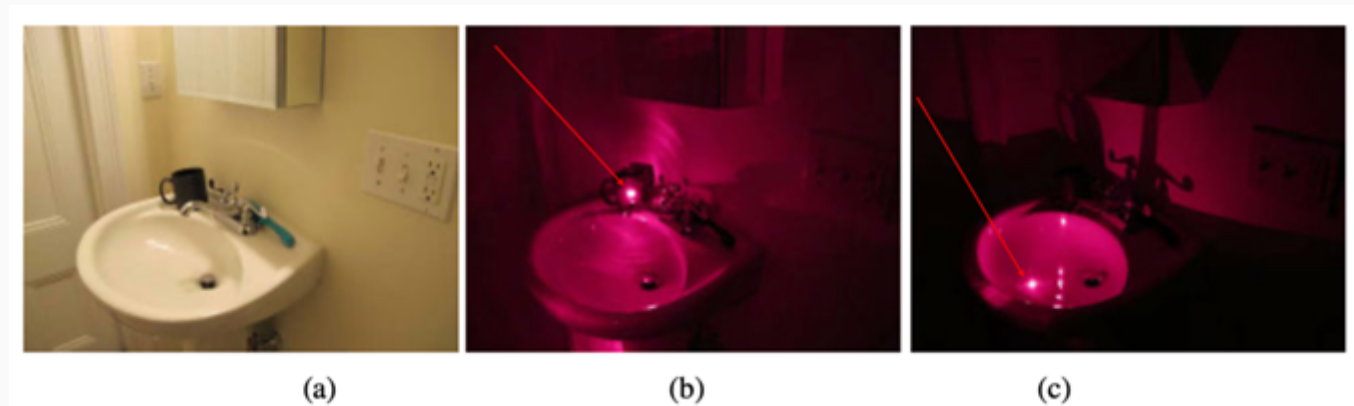


Figure 1: source: visionbook

- A light ray strikes a surface from direction $\mathbf{p}$ with intensity l_{in}
- The surface generates an **outgoing ray** in direction $\mathbf{q}$ with intensity l_{out}
- The relationship between l_{in} and l_{out} depends on:
 - the surface material
 - the surface normal $\mathbf{n}$
 - the directions $\mathbf{p}$ (in) and $\mathbf{q}$ (out)
 - the wavelength λ

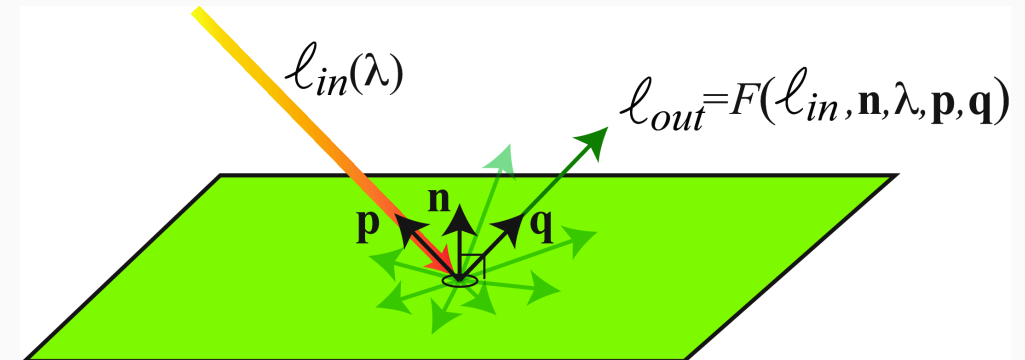


Figure 2: A light ray striking a surface.
source: visionbook.

The BRDF: General Formulation

The **bidirectional reflectance distribution function (BRDF)** F relates outgoing to incoming light:

$$l_{\text{out}} = F(l_{\text{in}}, \mathbf{n}, \mathbf{p}, \mathbf{q}, \lambda)$$

- l_{in} : intensity of the incoming ray
- l_{out} : intensity of the outgoing ray
- $\mathbf{n}$: surface normal
- $\mathbf{p}$: direction of the incoming ray
- $\mathbf{q}$: direction of the outgoing ray
- λ : wavelength (spectral component)

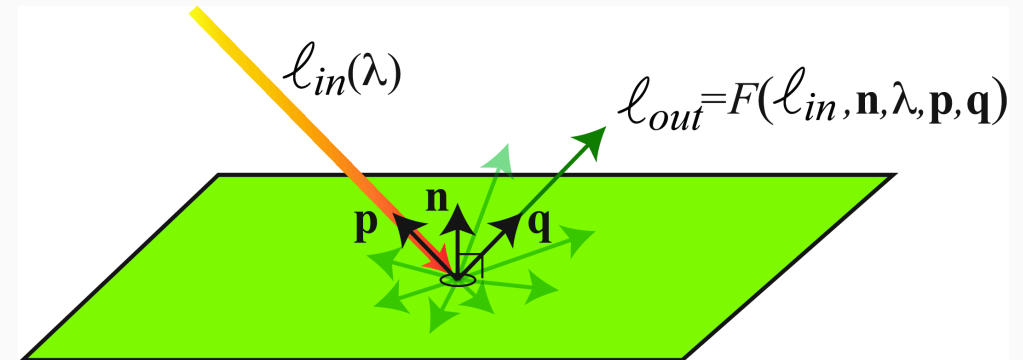


Figure 3: Lambertian shading depends only on the angle θ between the surface normal $\mathbf{n}$ and the incoming light direction $\mathbf{p}$. source: visionbook.

- Light behaves **linearly**: if multiple light sources illuminate a surface, the outgoing intensity is the **sum** of the contributions from each source
- Linearity justifies decomposing complex illumination sources into simple components (point lights, ambient light, ...) and summing their effects

The **Lambertian** model is the simplest BRDF: a perfectly diffuse surface that reflects light equally in all outgoing directions.

$$l_{\text{out}} = F_L(l_{\text{in}}(\lambda), \mathbf{n}, \mathbf{p}) = al_{\text{in}}(\lambda)(\mathbf{n} \cdot \mathbf{p})$$

- a : **albedo**, the surface's reflectance coefficient
- $\mathbf{n} \cdot \mathbf{p}$: the cosine of the angle between the surface normal and the incoming ray, $(\mathbf{n} \cdot \mathbf{p}) = \cos \theta$
- Notice F_L does **not** depend on the outgoing direction $\mathbf{q}$: the surface looks equally bright from every viewing angle, only the amount of incoming light and its angle matter

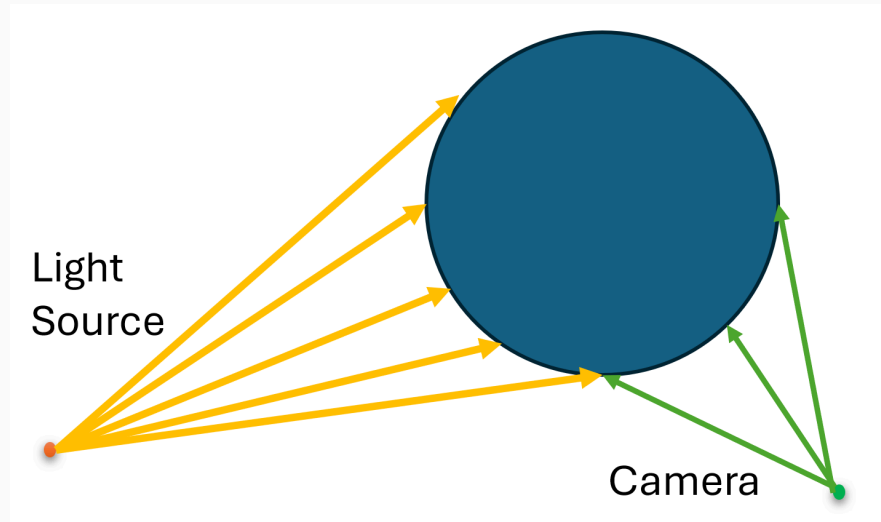


Figure 4: Point light source.

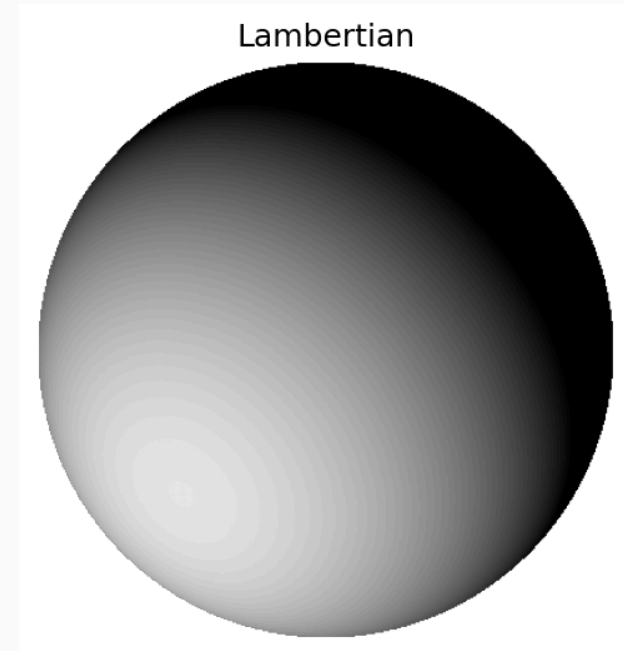


Figure 5: Lambertian sphere.

- Note the smooth gradient from bright (facing the light) to dark (facing away)

Real surfaces also reflect light **preferentially** around a mirror-like direction. The **Phong model** adds three terms:

- **Ambient**: a constant term, light scattered indirectly from everywhere. A crude approximation of global illumination
- **Diffuse**: the Lambertian term from before
- **Specular**: a highlight concentrated around the mirror reflection direction r

linearity of light means we can simply add these terms together.

Specular Surfaces and the Phong Model

Phong model:

$$l_{\text{out}} = \underset{\text{ambient}}{\color{red}k_a} + \underset{\text{diffuse (Lambertian)}}{a l_{\text{in}} (\color{green}\mathbf{n} \cdot \color{green}\mathbf{p})} + \underset{\text{specular}}{k_s (\color{blue}\mathbf{r} \cdot \color{blue}\mathbf{q})^\alpha l_{\text{in}}}$$

- k_a : ambient light coefficient
- a comes from Lambertian diffuse term

specular term:

- k_s : strength of the specular reflection
- α : “shininess”, controls how tightly the highlight is concentrated
- $\mathbf{r}$: direction of perfect mirror reflection of $\mathbf{p}$ about $\mathbf{n}$
- $(\mathbf{r} \cdot \mathbf{q})^\alpha$ is large only when the viewing direction $\mathbf{q}$ is close to $\mathbf{r}$

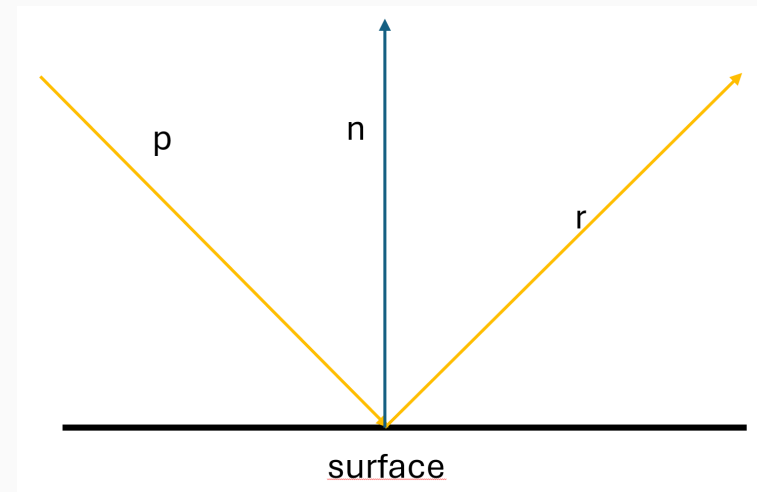


Figure 6: $\mathbf{r}$ in Phong model: the mirror reflection of the incoming light direction $\mathbf{p}$ about the surface normal $\mathbf{n}$.

Lambertian vs. Phong

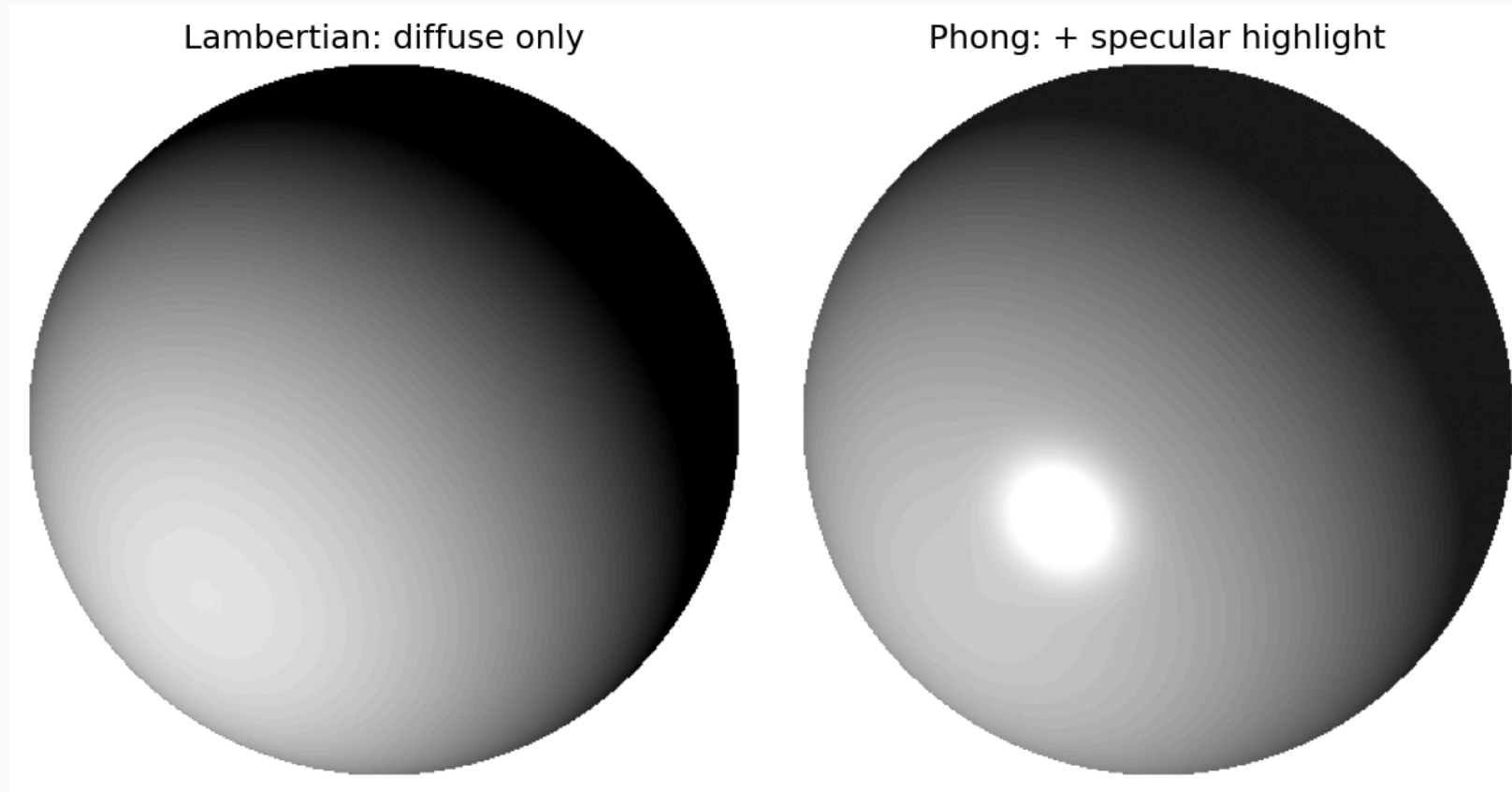


Figure 7: Same sphere, same light direction: the Lambertian render (left) is purely diffuse; the Phong render (right) adds a specular highlight where the viewing direction is close to the mirror-reflection direction $\mathbf{r}$.

$$l_{\text{out}} = k_a + \alpha l_{\text{in}}(\mathbf{n} \cdot \mathbf{p}) + k_s (\mathbf{r} \cdot \mathbf{q})^\alpha l_{\text{in}}$$

- Increasing α shrinks and sharpens the highlight (polished metal vs. matte plastic)
- Increasing k_s brightens the highlight without changing its size

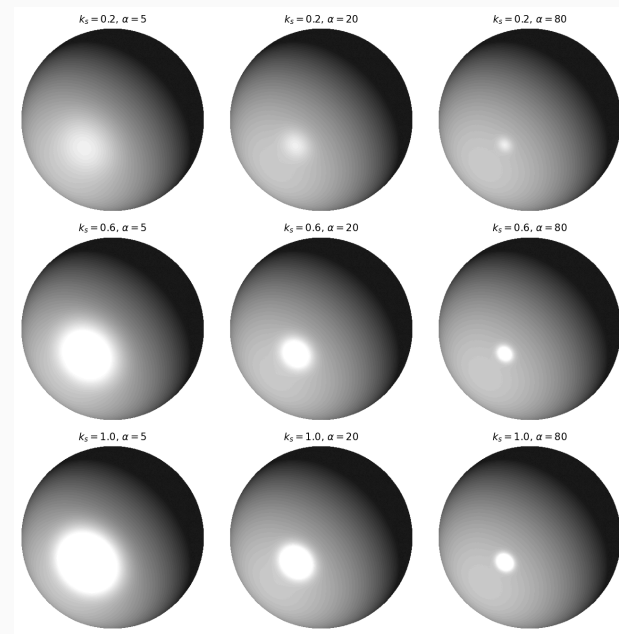
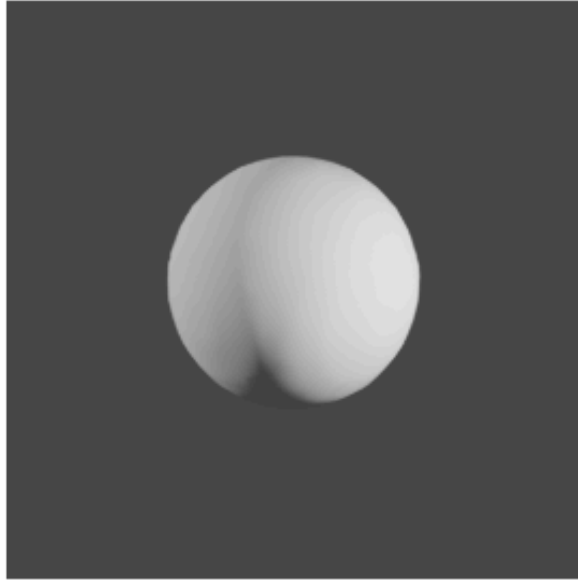
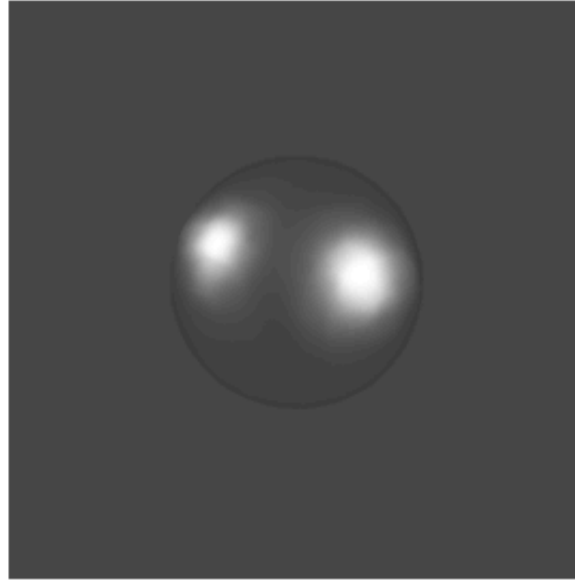


Figure 8: Same sphere and light direction, varying the specular strength k_s (rows) and shininess α (columns): larger α sharpens the highlight, larger k_s brightens it.



(a) Lambertian



(b) Phong



(c) Photograph

Figure 9: source: visionbook.

- the BRDF determines the image's appearance. Geometry, lighting, and material affect the way light interacts with surfaces, a recurring ambiguity in vision
- **advantage**: material properties can be inferred from images, and used for recognition and understanding
- **challenge**: material properties can be confused with geometry and lighting, making inference difficult

Example



Figure 10: Same objects, different light.

Pinhole Cameras



Why Don't We See Images on a Wall?

- Every point on a wall is illuminated by light arriving from **multiple directions**
- For a Lambertian wall, the outgoing radiance at a point is the **integral** over all incoming directions p :

$$l_{\text{out}} = \int_p a l_{\text{in}}(p) \cos(n \cdot p) dp$$

- This integral **averages away** all the spatial structure of the scene: the wall just looks like a uniformly lit surface, with no image of the scene visible on it

Restricting Incoming Rays

- To form an **image**, we must prevent each point on the imaging surface from seeing the entire scene
- Solution: place an opaque barrier with a single small hole (a **pinhole**) between the scene and the imaging surface
- Now each point on the imaging surface receives light from (approximately) **one direction only**, i.e. from one scene point
- This breaks the integral (previous slide) into a one-to-one (point-to-point) correspondence between scene points and image points

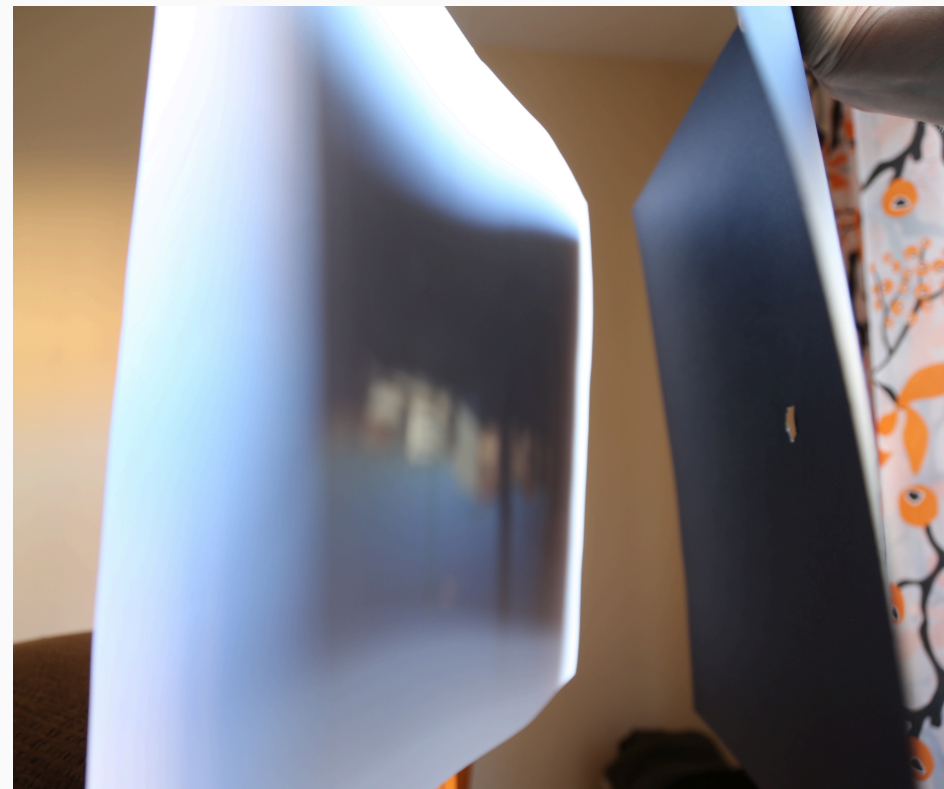


Figure 11: A simple pinhole camera setup.

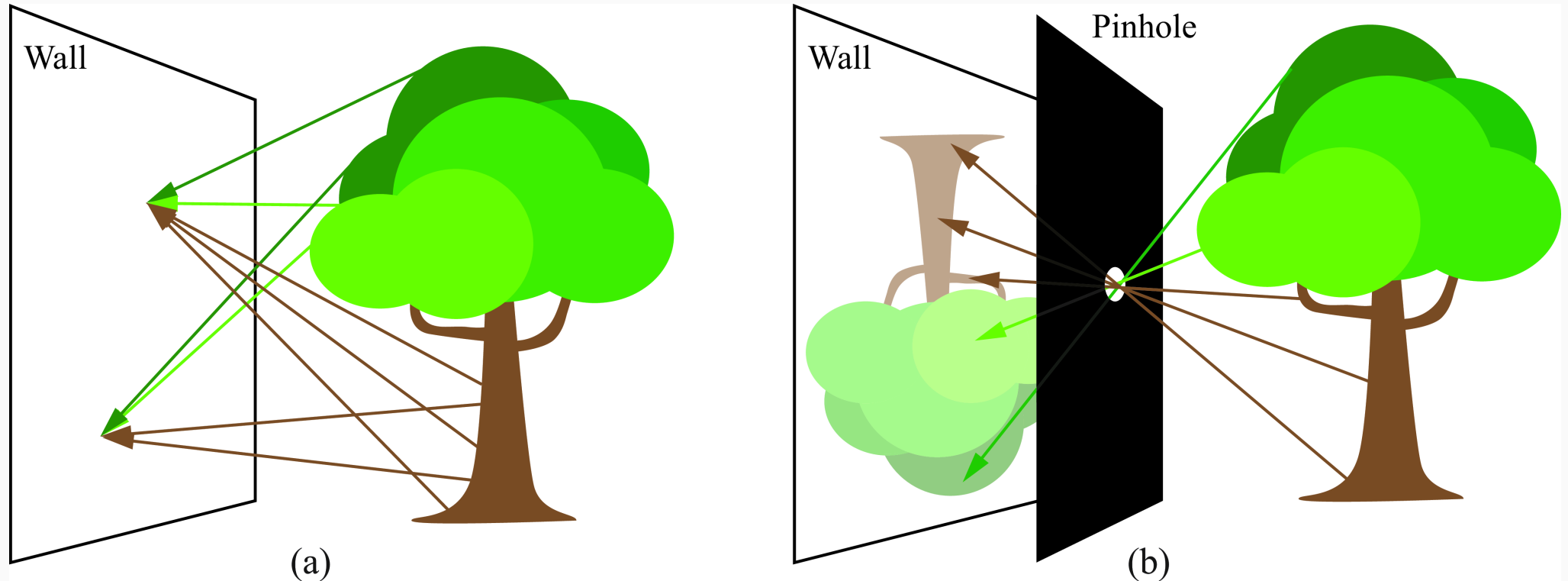


Figure 12: Left: without restriction, every point of the object reaches every point of the wall. Right: with a pinhole, each object point maps to exactly one wall point. source: visionbook.

The pinhole effect can occur naturally when light passes through a small gap (leaves in a tree, a small hole in a curtain).

Notice that the image is:

- inverted compared to the scene (upside down, left-right flipped)
- dimmer than the scene (because only a tiny fraction of the light passes through the hole)
- blurrier than the scene (because the hole is not an ideal point, and diffraction spreads the rays)



Figure 13: A simple pinhole camera setup.

- A pinhole camera consists of:
 - an opaque plane (the barrier) with a single small hole (the **pinhole** / **camera center** / **optical center**)
 - an imaging plane behind the barrier, where the image forms
- Each scene point emits/reflects light in all directions, but each image point receives light from a single ray, the one passing through the pinhole
- With a pinhole camera, we have (approximately) **one ray per image point**
- An ideal pinhole camera results in a perspective projection, the basis for all geometric models of cameras

Pinhole Limitations and Lenses

- problems with real-world pinhole cameras:
 - **Blur**: a real pinhole has finite size, so rays from a single scene point pass through different parts of the hole and arrive at different image points, blurring the image
 - **Light gathering**: a small pinhole admits little light, requiring long exposure times
- Solution**: a **lens** can have a large aperture (more light) while still focusing rays from each scene point to (approximately) a single image point

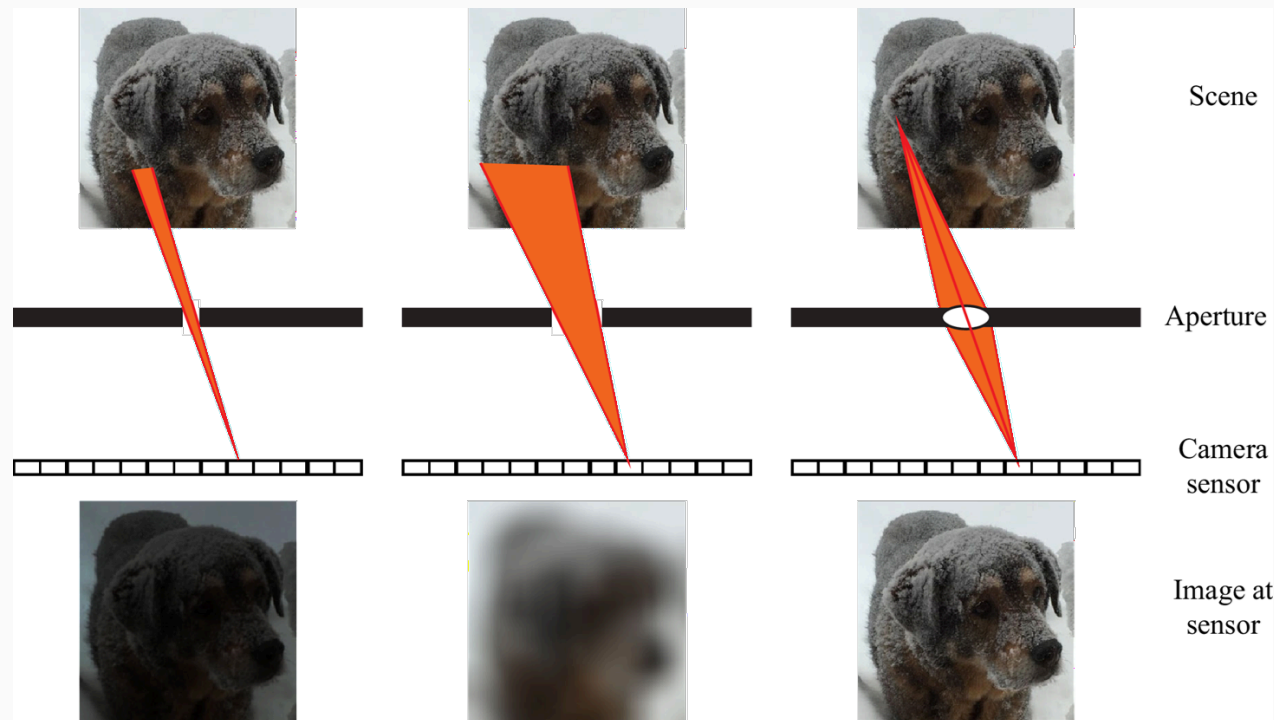


Figure 14: source: visionbook

Perspective Projection

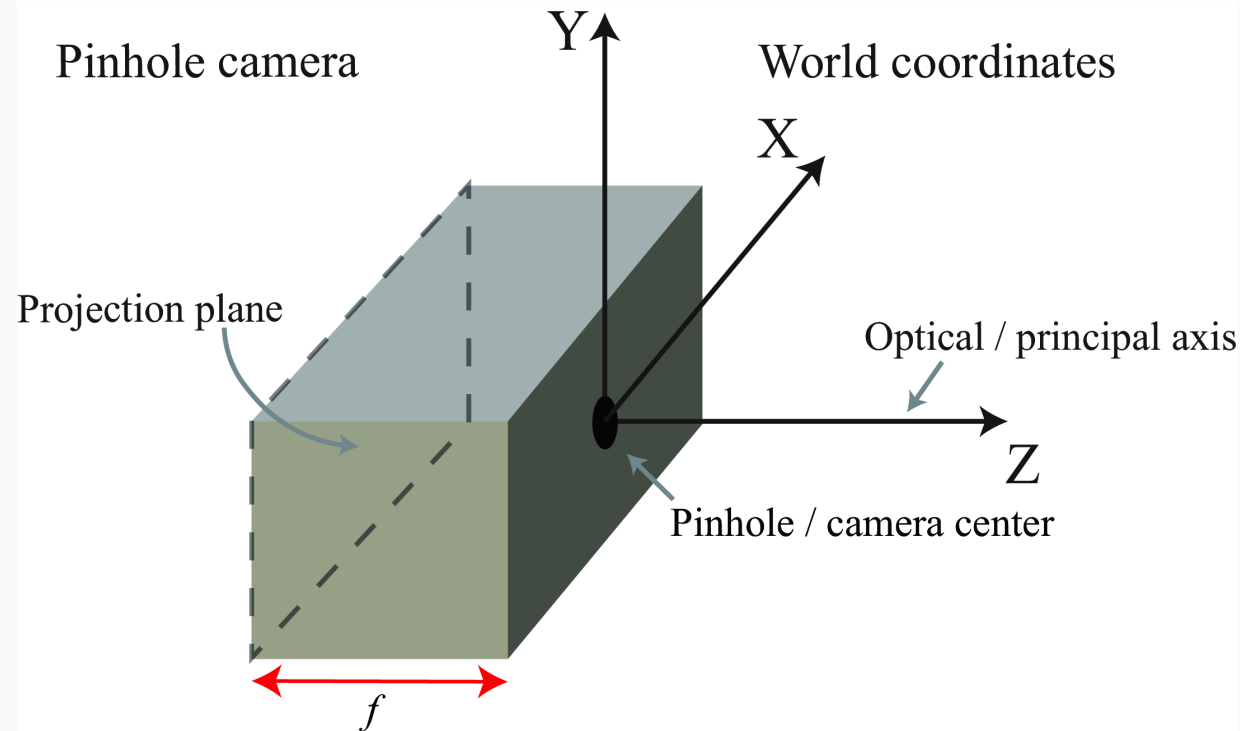


Figure 15: How do we convert coordinates from a 3D scene point P to the corresponding 2D image point p on the image plane?

Perspective Projection: World to Camera Coordinates

- **World coordinates**: a 3D point in the scene $P = (X, Y, Z)$
- **Camera coordinates**: the corresponding 2D position $p = (x, y)$ on the image plane
- The pinhole (camera center) is the origin $(0, 0, 0)$; the optical axis is the Z axis, pointing into the scene
 - we will remove this simplification later in the course
- f : the distance between the pinhole and the image plane (the **focal length**)

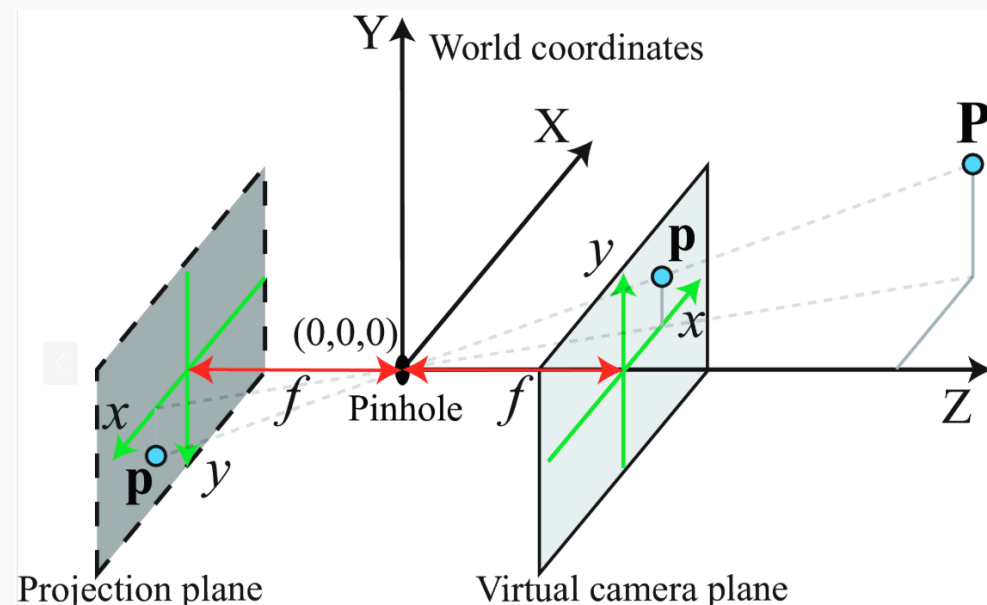


Figure 16: source: visionbook.

The Virtual Image Plane Trick

- the image plane is **behind** the pinhole at distance f , and the projected image is **inverted** (upside down, left-right flipped)
- Trick**: instead, consider a **virtual image plane** placed at distance f **in front of** the pinhole (between the pinhole and the scene)
- This virtual-plane convention is what we use from now on and the standard in most software

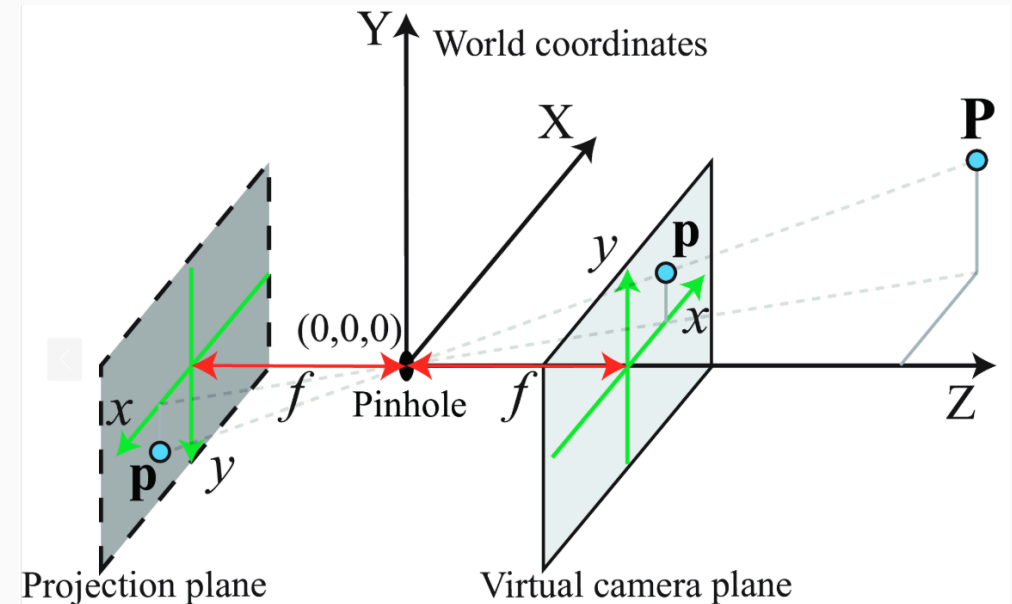


Figure 17: source: visionbook.

Coordinate Mapping: The Perspective Projection Equations

perspective projection:

focal length
↓
 $x = f \frac{X}{Z}, y = f \frac{Y}{Z}$
↑
depth

- Coordinates on the virtual plane are **not** inverted
- image coordinates (x, y) are proportional to the world coordinates (X, Y) , scaled by $\frac{f}{Z}$
- derived from similar triangles formed by the scene point, the pinhole, and the image point

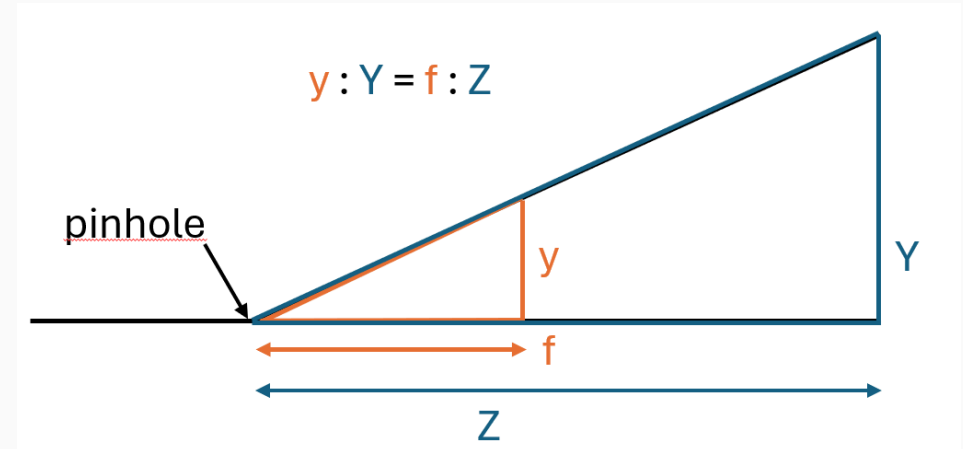


Figure 18: Similar triangles as used in the derivation of the perspective projection equations.

Distant Objects Appear Smaller

- given that $x = f \frac{X}{Z}$, for a fixed object size X , increasing the depth Z **decreases** the projected size x proportionally to $\frac{1}{Z}$
- Two objects of the same physical size at different depths project to different image sizes
- Parallel lines receding in depth converge toward a single point (a **vanishing point**) as $Z \rightarrow \infty$, since $x = f \frac{X}{Z} \rightarrow 0$ regardless of X
- This is the geometric basis for depth cues from a single image

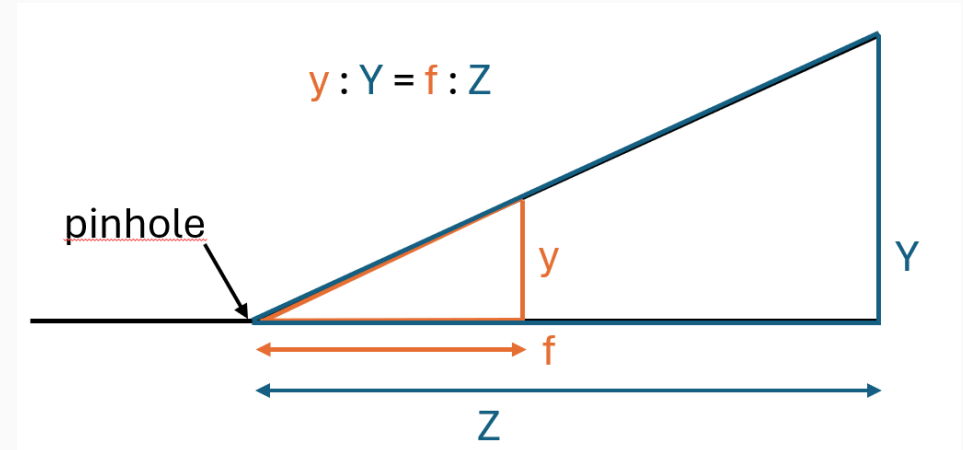


Figure 19: Similar triangles as used in the derivation of the perspective projection equations.

- $x = f \frac{X}{Z}$ depends on X and Z only through their **ratio** $\frac{X}{Z}$
- an object of size X at depth Z projects to exactly the same image point as an object of size kX at depth kZ , for any k
 - Example: object A ($X = 1, Z = 2$) and object B ($X = 2, Z = 4$) both project to $x = 0.5$
- Perspective projection is a **many-to-one** map: a single image is consistent with infinitely many (size, depth) pairs

consequences:

- we lose information about size and depth
- we need scale-invariance

From Camera Coordinates to Image (Pixel) Coordinates

- **Camera coordinates** (x, y) are **continuous**, measured in physical units, with origin on the optical axis
- **Image coordinates** (n, m) are **discrete** pixel indices, with origin conventionally at the top-left corner of the image

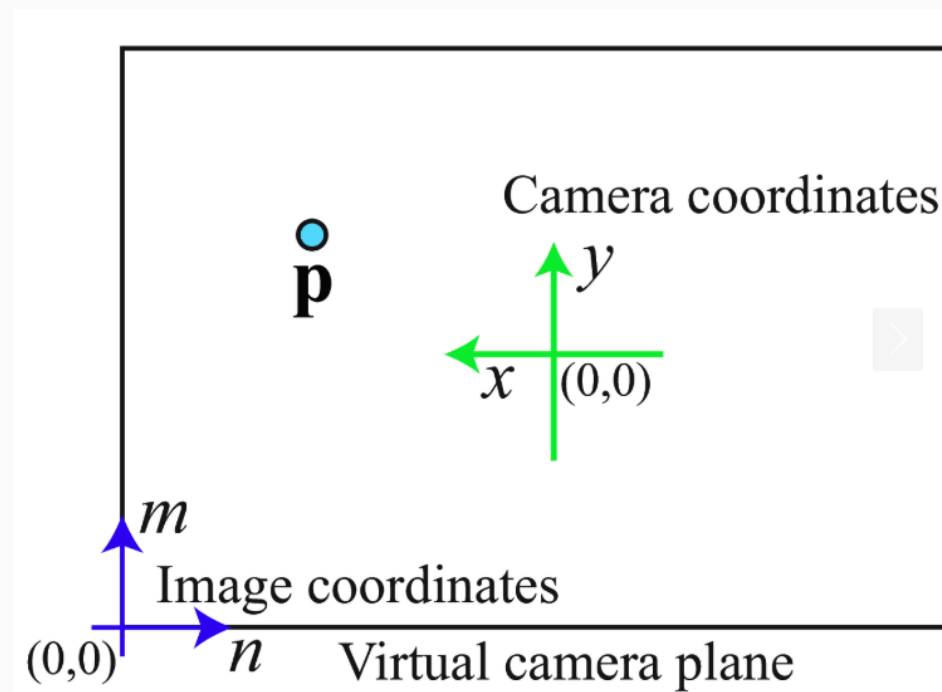


Figure 20: Pixel grid with image-coordinate axes (n, m) overlaid with the camera-coordinate axes (x, y) , related by the principal point (n_0, m_0) .

Image Coordinates

$$n = -\overset{\text{scale factor}}{a}x + \overset{\text{principal point}}{n_0}, m = -ay + m_0$$

- a : a **scale factor** converting physical units to pixels (related to pixel size and focal length)
- (n_0, m_0) : the **principal point**, the pixel coordinates of the optical axis
- the minus sign accounts for the **flip** between the camera-coordinate axes and the image-coordinate axes (e.g. y increasing upward vs. m increasing downward)

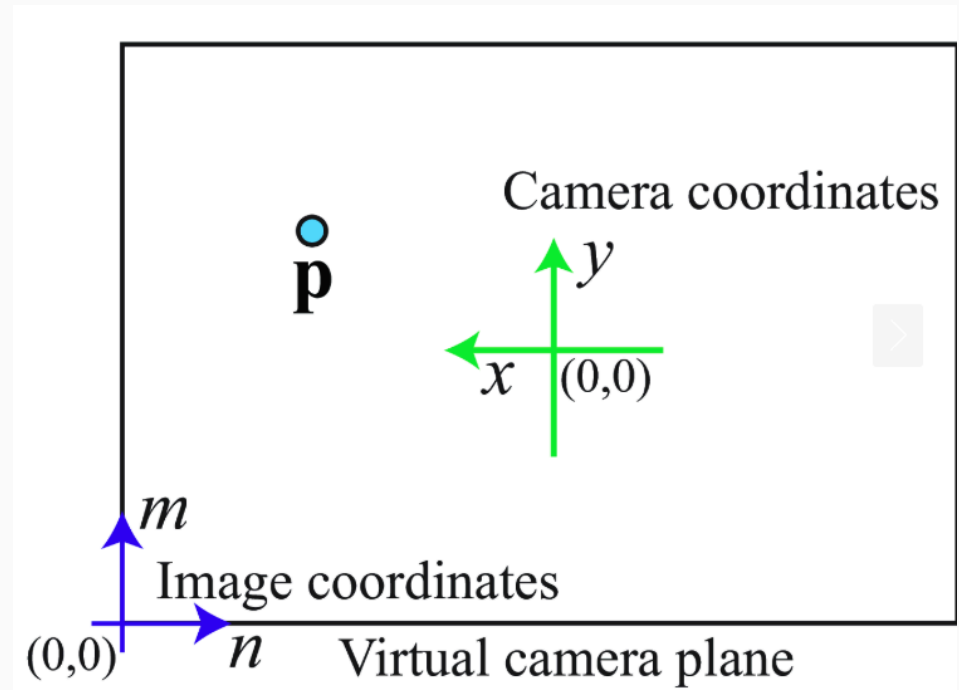


Figure 21: Pixel grid with image-coordinate axes (n, m) overlaid with the camera-coordinate axes (x, y) , related by the principal point (n_0, m_0) .

- perspective projection guides geometric reasoning about cameras and images
- perspective projection is a **many-to-one** mapping, losing information about absolute size and depth
- we have several coordinate systems: world, camera, images (with different and not always consistent conventions)
- we need to be able to translate between all these coordinates in many CV tasks (e.g. 3D reconstruction, camera calibration)

Orthographic Projection

Perspective Isn't the Only Choice

- Perspective projection is the natural consequence of a pinhole: rays **converge** to a single point, so projected size depends on depth ($x = f \frac{X}{Z}$)
- But that convergence is a **choice** of geometry, not a law of nature. What if we instead only accept rays that arrive **perpendicular** to the image plane?
- Then every scene point maps to the image via **parallel** rays, and the imaging geometry is fundamentally different

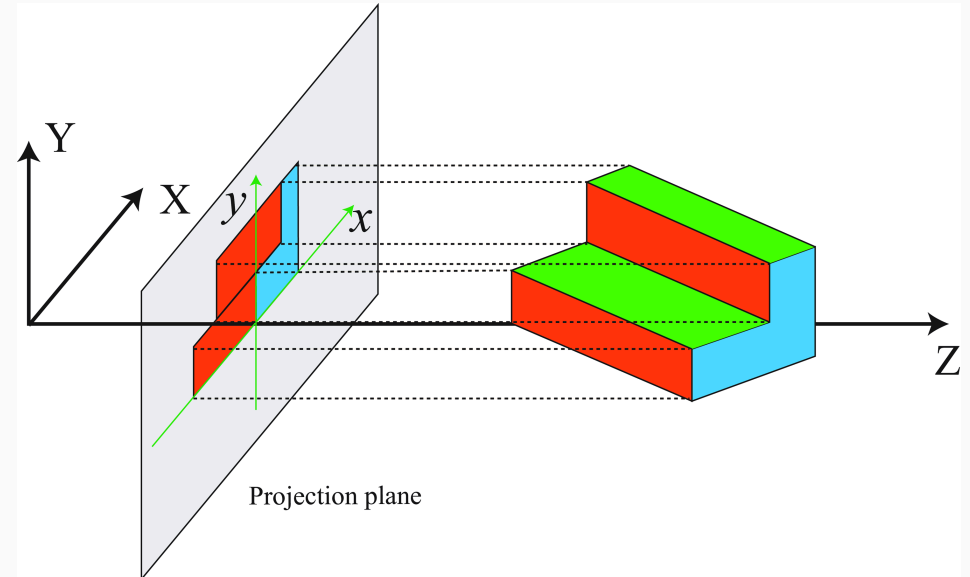


Figure 22: source: visionbook

orthographic projection:

$$x = kX, \quad y = kY$$

- k : a fixed, global scale factor (no dependence on depth Z at all)
- Object size in the image is **independent of depth**: a nearby and a far object of the same physical size project to the **same** image size
- This removes the size-depth ambiguity of perspective projection, but at the cost of removing depth information from the image **entirely**, there is no vanishing point, no perspective foreshortening

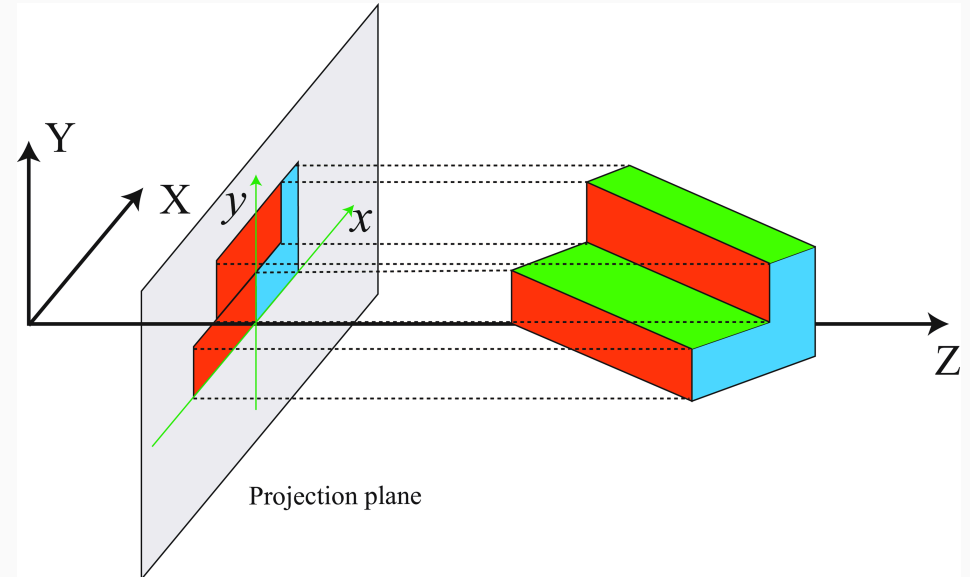
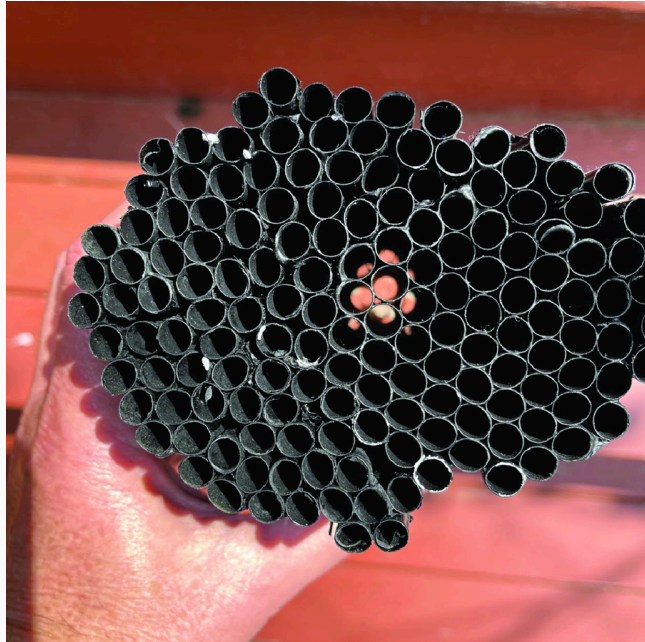


Figure 23: source: visionbook

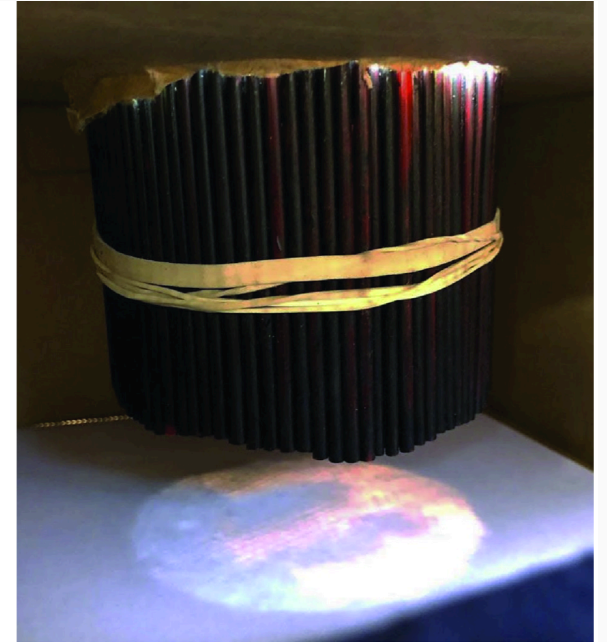
Soda Straw Camera



(a)



(b)



(c)

Figure 24: source: visionbook

- **Perspective**: keeps a depth cue (size shrinks with distance) but conflates it with object scale, an ambiguity that must be resolved by other means (stereo, priors, learned depth estimation)
- **Orthographic**: removes the scale-depth ambiguity by construction, but at the price of discarding depth information altogether
- Neither is “more correct”, they are different deliberate choices of what information an imaging system preserves, a theme that will recur when we look at cameras as linear systems

Cameras as Linear Systems

- So far, “camera” has meant an ideal pinhole or lens focusing rays onto a sensor
- But **any** device that turns a scene into measurements can be described the same way: light is linear, so the scene to sensor map is always some linear map $l_s = Al_w$
 - This abstraction lets us reason about any generic imaging setup
 - blurry real pinholes
 - corner cameras
- **Objective**: once we know A , recovering the scene l_w from measurements l_s becomes a general problem (inverting, or regularizing, a linear system)
 - we can apply this method to any camera

The Camera as a Linear System

Consider a simplified **1D world**:

- a vector $\mathbf{l}_w \in \mathbb{R}^N$ of scene light intensities (one entry per scene “position” or direction)
- The sensor records a vector $\mathbf{l}_s \in \mathbb{R}^M$ of measurements

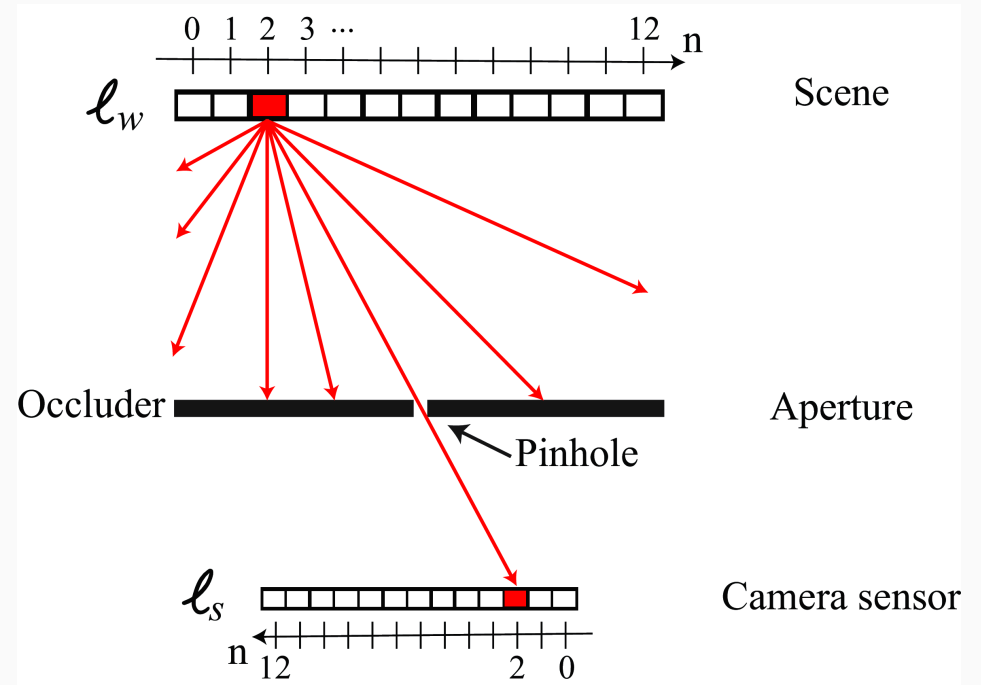


Figure 25: 1D world intensities $\mathbf{l}_w$, the camera as a linear operator A , and the resulting sensor measurements $\mathbf{l}_s$. source: visionbook.

The Camera as a Linear System

- Because light is **linear**, the relationship between scene and sensor is a **linear map**:

imaging operator

$$l_s = A l_w, A \in \mathbb{R}^{M \times N}$$

- Each row of A describes how one sensor measurement is a (weighted) combination of scene intensities, i.e. the camera's response/sensitivity pattern

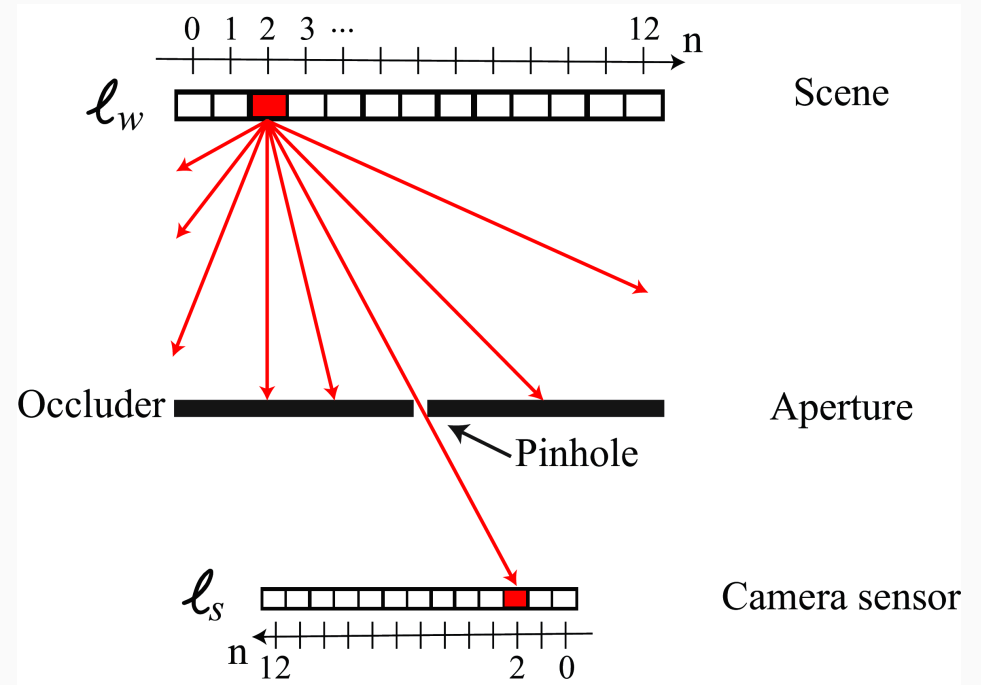


Figure 26: 1D world intensities l_w , the camera as a linear operator A , and the resulting sensor measurements l_s . source: visionbook.

The Ideal Pinhole as $A = I$

$$\mathbf{l}_s = \mathbf{A} \mathbf{l}_w, \mathbf{A} \in \mathbb{R}^{M \times N}$$

- For an **ideal pinhole camera**, each sensor pixel receives light from exactly one scene direction,
- using the virtual image plane convention, so the mapping is direct (no flip), we have $A = I$, the identity matrix

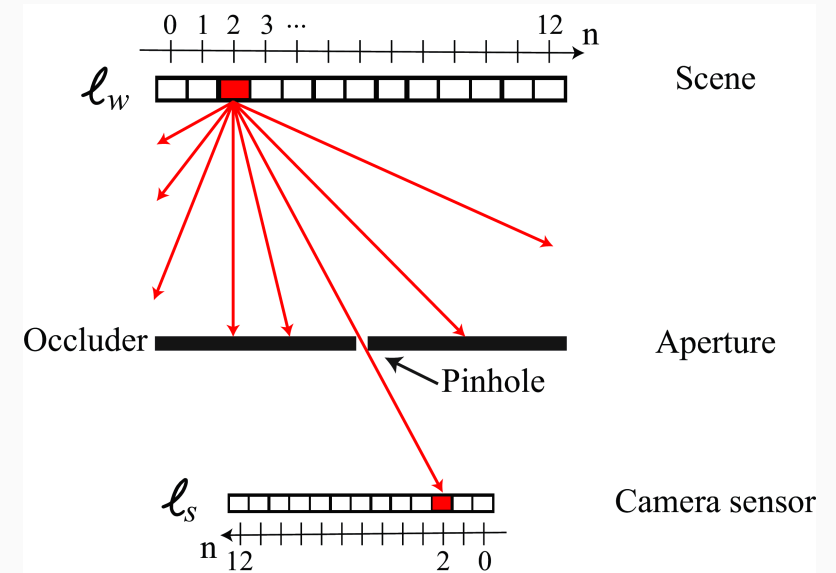


Figure 27: 1D world intensities $\mathbf{l}_w$, the camera as a linear operator $\mathbf{A}$, and the resulting sensor measurements $\mathbf{l}_s$. source: visionbook.

- A “less than ideal” pinhole (finite hole size, diffraction) makes A a matrix close to I but with some spread (each row has most of its weight on one entry, with small contributions from neighbors), this is a **blur**, i.e. a convolution

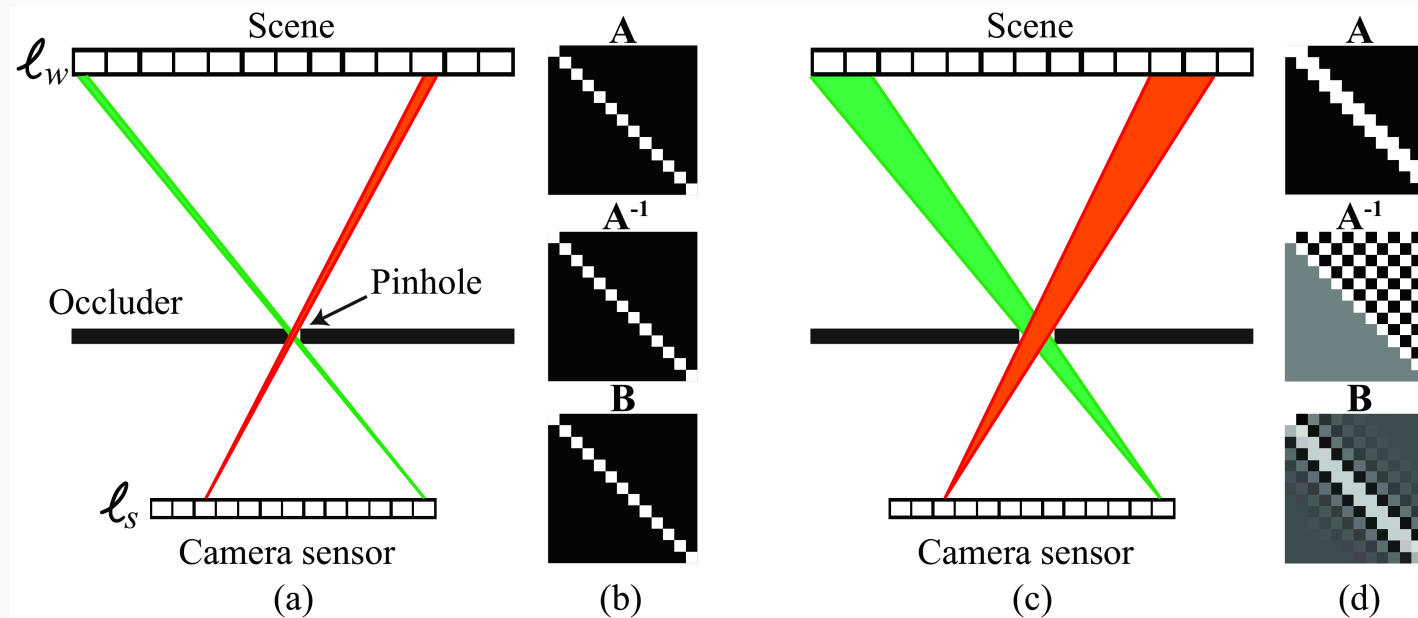


Figure 28: A noisy pinhole camera. source: visionbook.

- Once we accept that **any** linear A defines a valid imaging system, we can design cameras that look nothing like a pinhole, as long as we can still recover l_w from l_s

Examples:

- edge camera
- pinspeck camera
- corner camera

In each case, the imaging matrix A mixes scene values far more than a pinhole

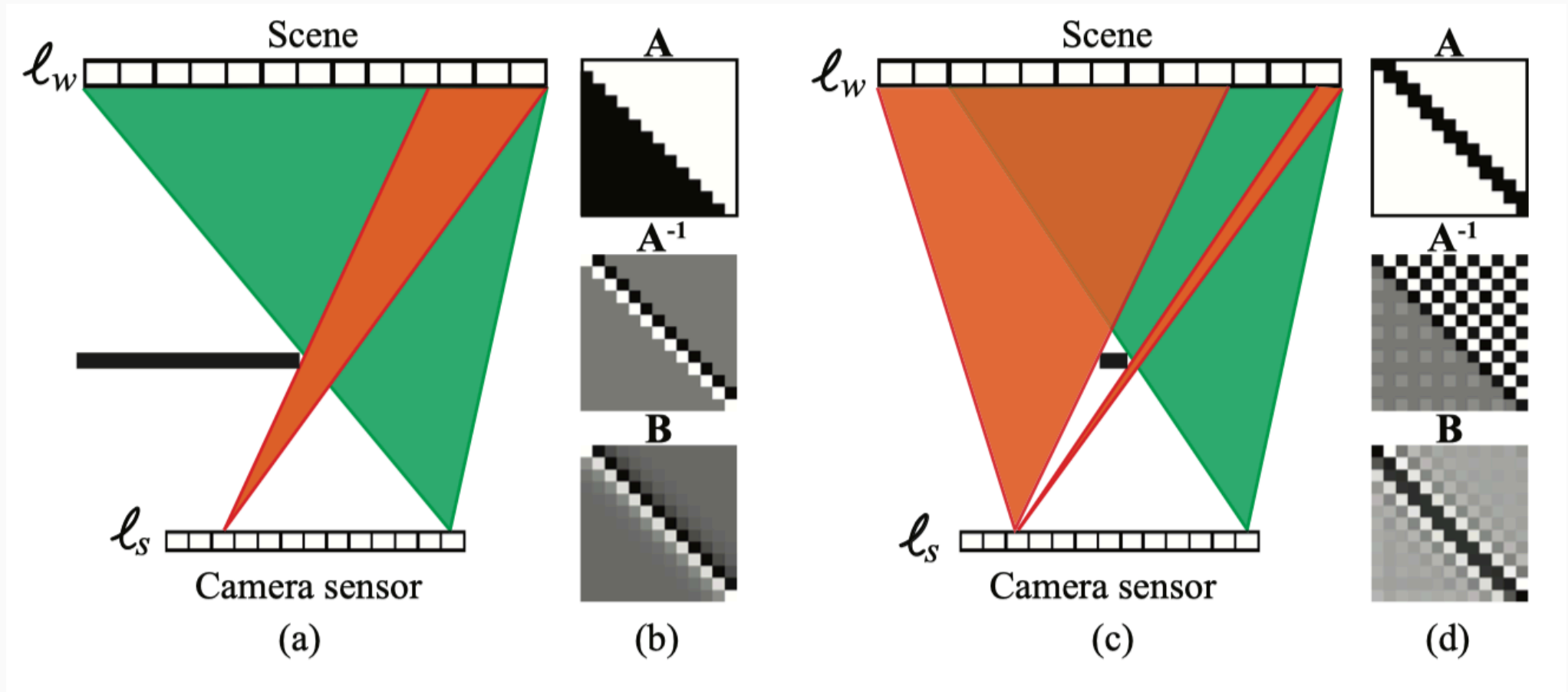


Figure 29: Edge (left) and pinspeck (right) camera. source: visionbook

- Given a measurement $l_s = Al_w$, can we recover l_w ?
- If A is square and invertible: $l_w = A^{-1}l_s$, done
- In practice, A is often **not invertible**:
 - **ill-conditioned**: small measurement noise causes huge errors in l_w
 - **rank-deficient** or non-square
- It's an **ill-posed** problem, where many different scenes l_w can produce nearly the same measurement l_s
 - the corner camera is an extreme example where the useful signal is a tiny fraction of the measurement

- To get a well-posed problem, add a **regularization** term that penalizes implausible solutions (e.g. very large or noisy l_w):

$$E(l_w) = \underbrace{\|l_s - Al_w\|^2}_{\text{data fidelity}} + \underbrace{\lambda \|l_w\|^2}_{\text{regularizer}}$$

- First term: data fidelity, how well does l_w explain the measurement?
- Second term: regularizer, prefer solutions with small norm (a simple prior), weighted by $\lambda > 0$
- λ controls the trade-off: $\lambda \rightarrow 0$ recovers the (possibly unstable) least-squares solution; large λ biases strongly toward $l_w \approx 0$

- Expand the objective:

$$E(\mathbf{l}_w) = (\mathbf{l}_s - A\mathbf{l}_w)^T (\mathbf{l}_s - A\mathbf{l}_w) + \lambda \mathbf{l}_w^T \mathbf{l}_w$$

- Take the gradient with respect to $\mathbf{l}_w$ and set it to zero:

$$\nabla_{\mathbf{l}_w} E = -2A^T (\mathbf{l}_s - A\mathbf{l}_w) + 2\lambda \mathbf{l}_w = 0$$

- Rearranging:

$$A^T A \mathbf{l}_w + \lambda \mathbf{l}_w = A^T \mathbf{l}_s \quad \Rightarrow \quad (A^T A + \lambda I) \mathbf{l}_w = A^T \mathbf{l}_s$$

- Solving for $\mathbf{l}_w$ gives the closed-form regularized inverse (next slide)

Minimizing $E(l_w)$ (setting the gradient to zero) gives a closed-form solution:

$$l_w = (A^T A + \lambda I)^{-1} A^T l_s = \underset{\substack{\uparrow \\ \text{regularized pseudo-inverse}}}{B} l_s$$

- $B = (A^T A + \lambda I)^{-1} A^T$ is the **regularized (pseudo-)inverse** of A
- Adding λI makes $A^T A + \lambda I$ invertible even when $A^T A$ alone is singular or ill-conditioned (it shifts all eigenvalues up by λ , away from zero)
- This is the classic **Tikhonov regularization** / **ridge regression**

The Regularized Inverse (2)

- Notice how, in the noisy pinhole, A^{-1} shows a “checkerboard” pattern, while the regularized inverse B is smoother.
- The checkerboard pattern makes it very sensitive to noise: small measurement errors get amplified into large, high-frequency errors in l_w

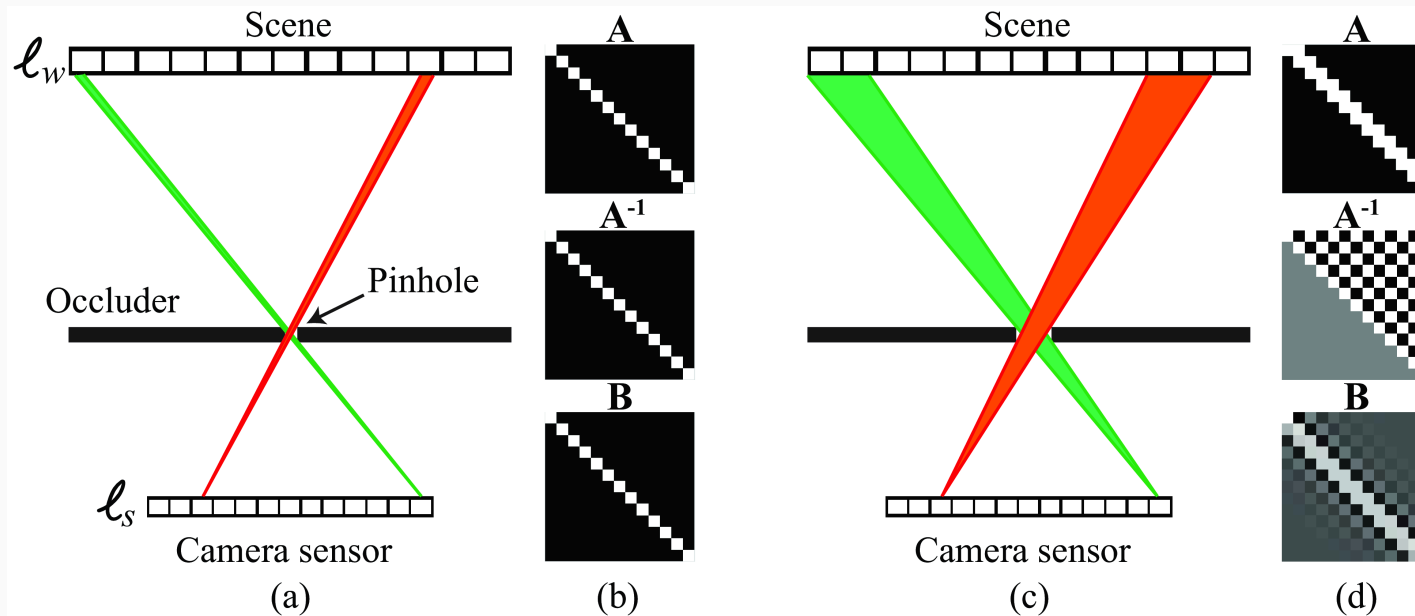


Figure 30: Edge and pinspeck cameras. source: visionbook.

The Regularized Inverse (3)

Notice how the regularized inverse B differs from the true inverse A^{-1} .

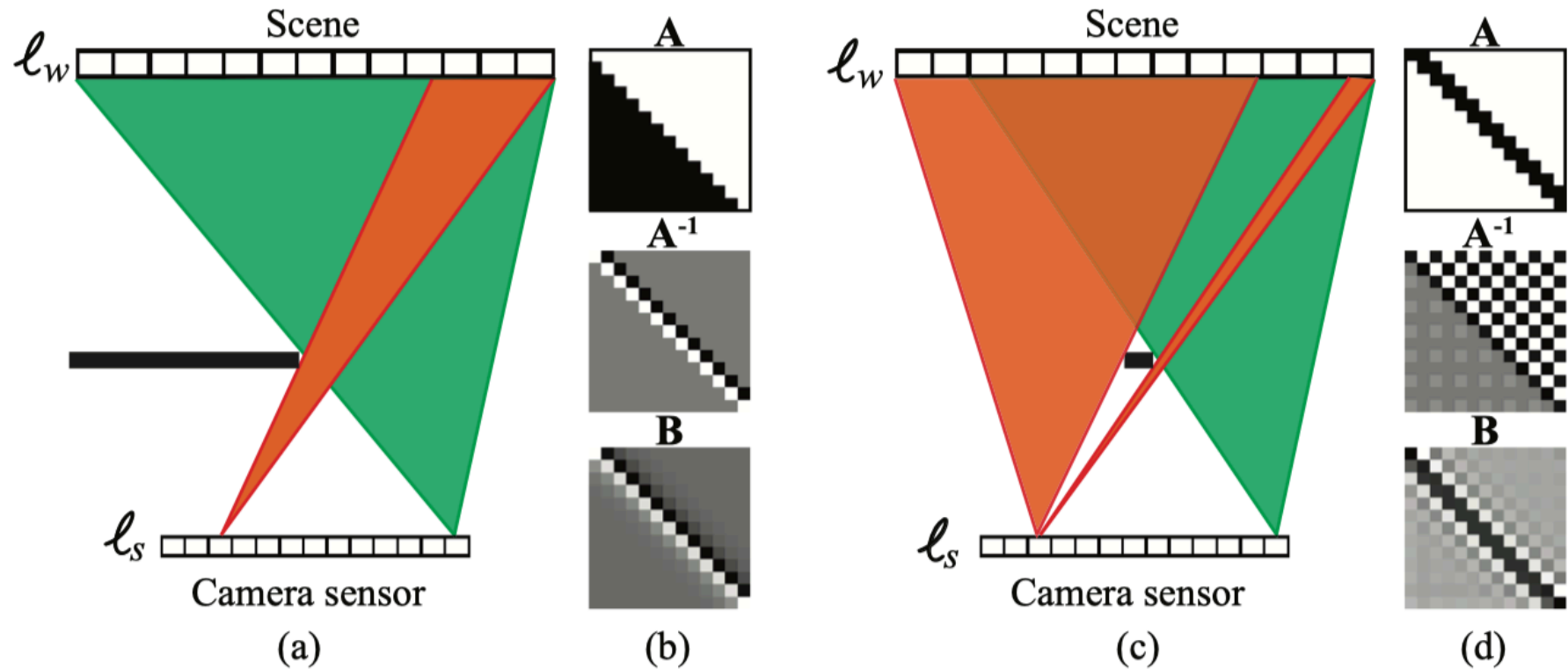


Figure 31: Edge and pinspeck cameras. source: visionbook.

- The simple linear forward model solved via regularized inverse is a common pattern in computer vision
- **Convolution/deconvolution**: a blur is a linear operator A (a convolution matrix); deblurring is solved via regularized inversion
 - the blurry pinhole blurs the image (convolution). Inverting the blurry pinhole is a deconvolution operation

Color



Color Is Ambiguous

- Light carries a full **spectrum** of power across wavelength
- Our color vision **compresses** that spectrum down to a handful of numbers
- Compression is lossy and **ambiguous**:
 - the same three numbers can arise from many different physical spectra
 - the same surface can produce very different numbers under different lighting
- Colors in images need to match our lossy perception

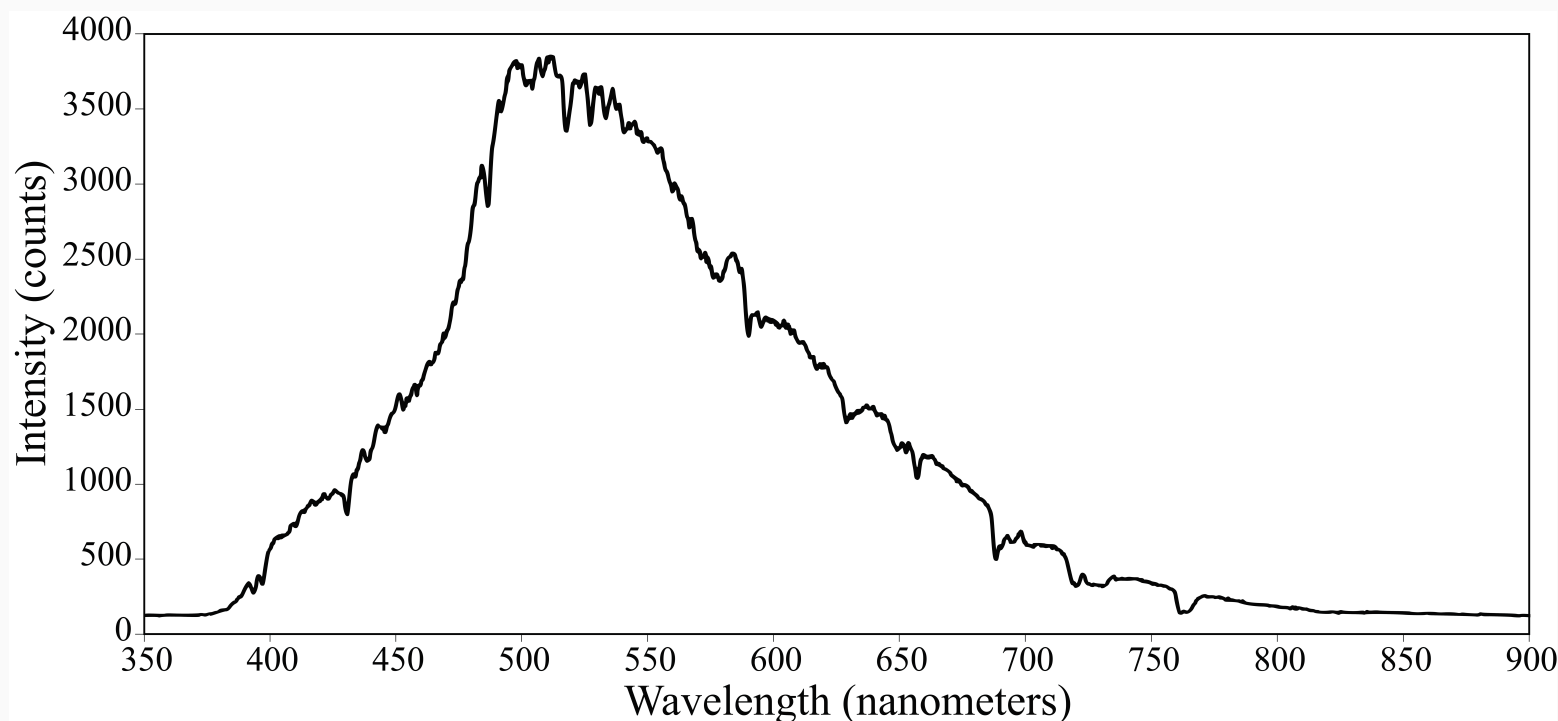


Figure 32: Light spectrum of a blue sky. source: visionbook

Light Power Spectra

- Visible light spans roughly 400-700 nm (nanometers) in wavelength, from blue/violet to deep red
- A light source or reflected ray is fully described by its **power spectrum**: the intensity at each wavelength, $I(\lambda)$
- A prism separates a beam by wavelength, revealing that “white” light is really a broad mixture of all visible wavelengths

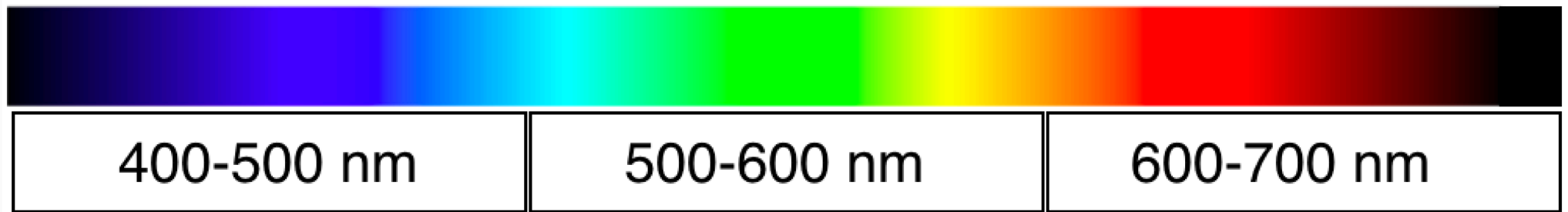


Figure 33: source: visionbook

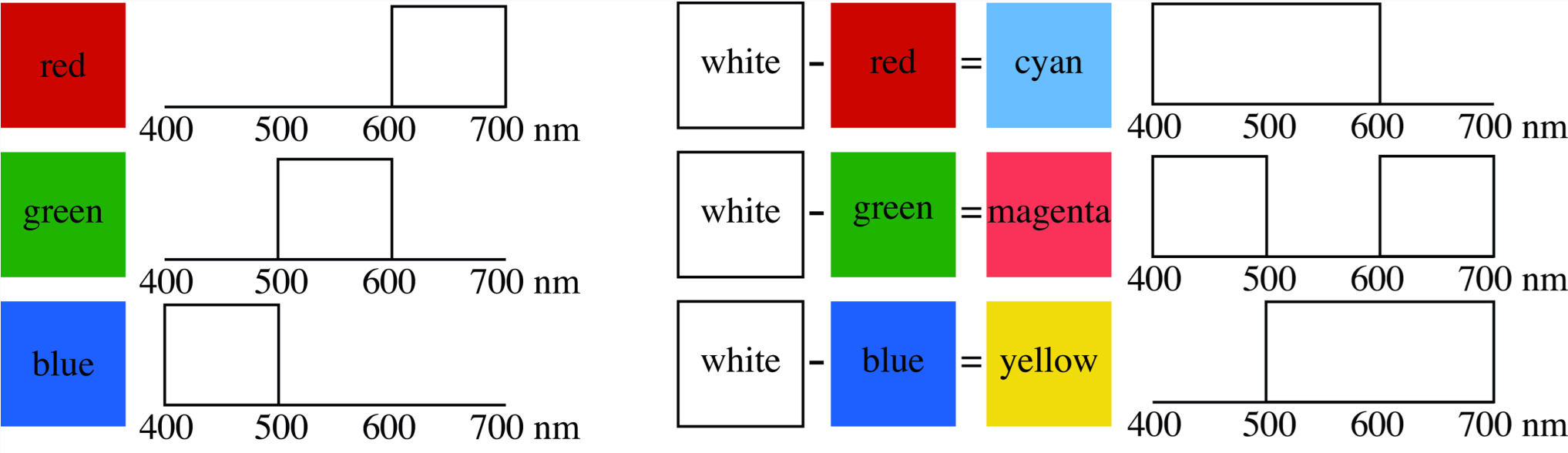


Figure 34: source: visionbook

- Recall the Lambertian BRDF from earlier today: $l_{\text{out}} = al_{\text{in}}(\mathbf{n} \cdot \mathbf{p})$, with a single scalar albedo a
- Making the wavelength-dependence explicit, a surface has a **reflectance spectrum** $s(\lambda) \in [0, 1]$ (how much of each wavelength it reflects), and the observed (reflected) spectrum is

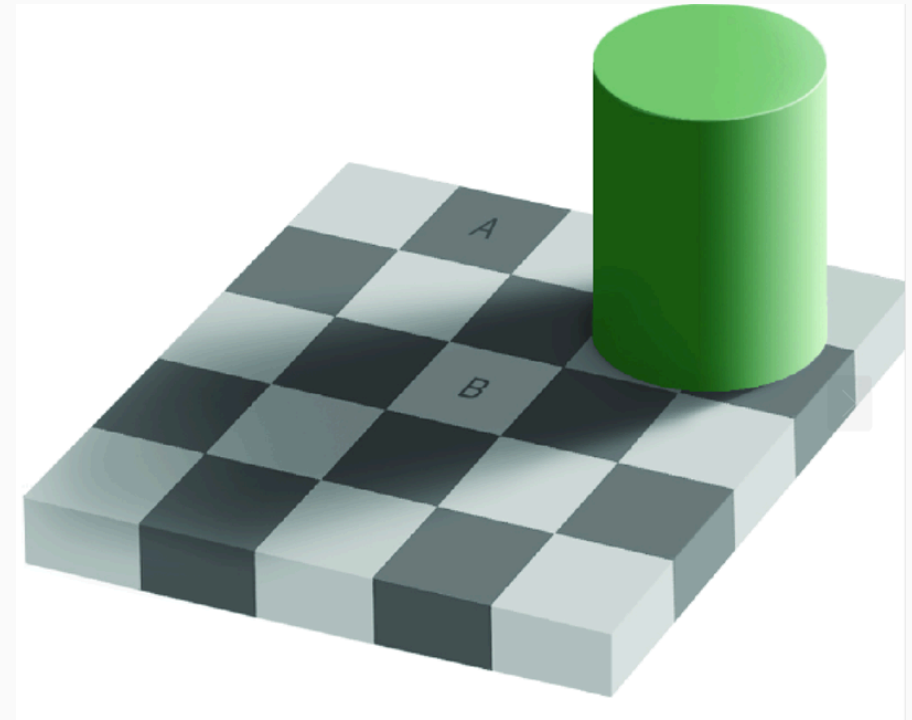
$$r(\lambda) = kl_{\text{in}}(\lambda)s(\lambda)$$

- $l_{\text{in}}(\lambda)$: the illuminant's spectrum, $s(\lambda)$: the material's reflectance, k : a geometry-dependent constant (the $\mathbf{n} \cdot \mathbf{p}$ factor from before)
- The reflectance spectrum $s(\lambda)$ is the material property we usually **want** (it correlates with ripeness, material identity, health, ...), but what we **observe**, $r(\lambda)$, is a **product** of $s(\lambda)$ with the unknown illuminant $l_{\text{in}}(\lambda)$

- Since $r(\lambda) = kl_{\text{in}}(\lambda)s(\lambda)$ is a product of two unknowns, observing $r(\lambda)$ alone cannot uniquely determine $s(\lambda)$: a reddish object under white light and a gray object under reddish light can produce the **same** observed spectrum
- **Color constancy**: the (ill-posed) problem of estimating the reflectance $s(\lambda)$ given only the observed $r(\lambda)$, i.e. discounting the illuminant
- The human visual system solves this approximately and automatically, cameras must do so with explicit algorithms (heuristics, statistics, or learned models)

Worked Example: The Same Pixel Value, Two Perceived Colors

- Squares A and B have **identical** pixel values (verify it), yet appear as different shades because the visual system discounts the apparent shadow, a form of automatic (partial) color constancy
- Pixel values are **not** a direct readout of surface color, they are surface color entangled with illumination

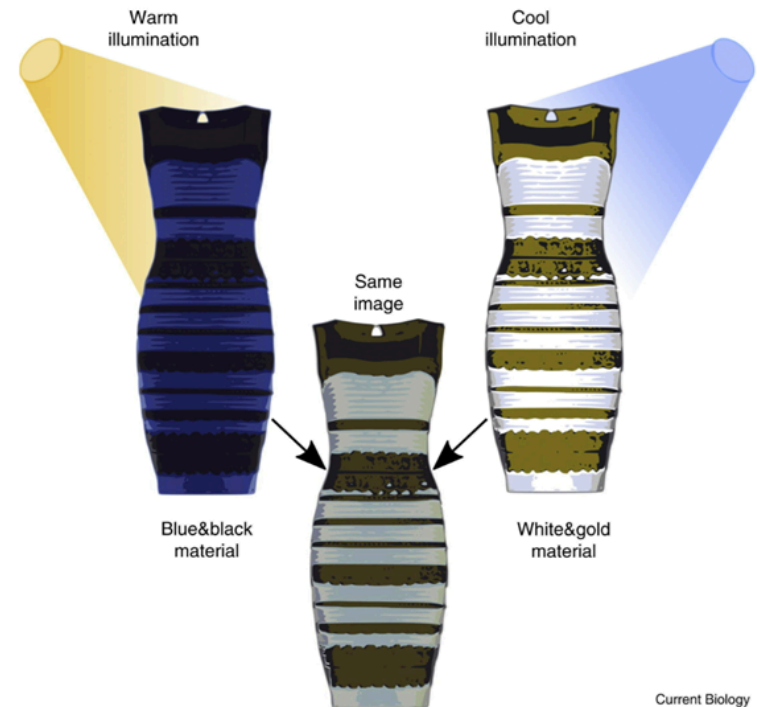


The Dress

- The 2015 “#TheDress” photograph is a famous example:
 - if you assume warm light, the dress is black and blue
 - if you assume cool light, the dress is white and yellow
 - this is an implicit assumption about the illuminant, often made unconsciously



(a)



(b)

Figure 36: source: visionbook

The Machinery of the Eye: Cones and Trichromacy

- The human retina has **rods** (monochromatic, active in low light) and **cones** (color vision, active in normal light)
- There are three cone types, **S**, **M**, **L** (short/medium/long wavelength peak sensitivity), each integrating incoming light across the full spectrum, weighted by its own sensitivity curve
- **Trichromacy**: human color perception is (to good approximation) determined by just these **three numbers**, regardless of the full (continuous, infinite-dimensional) spectrum of the incoming light
- This is **why** RGB representations are sufficient when targeting human vision

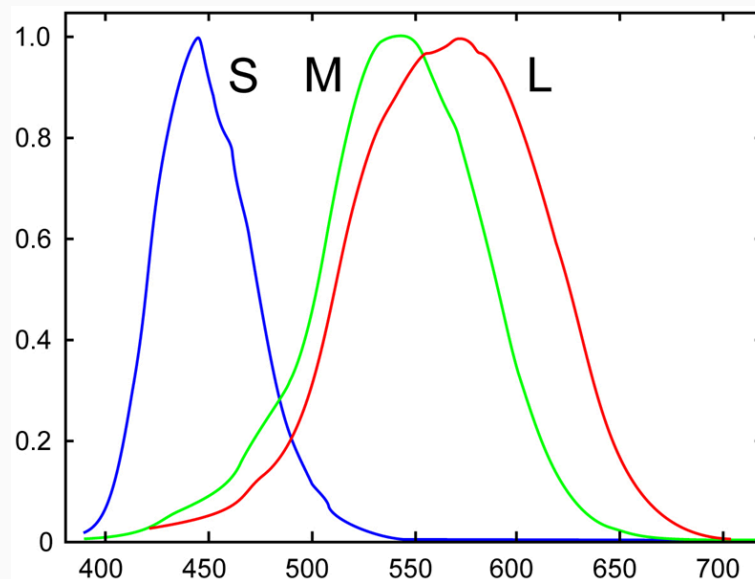


Figure 37: source: visionbook

Cone Response as a Linear Map

- Each cone's response is a weighted integral of the incoming spectrum $l(\lambda)$ against its sensitivity curve
- Stacking the three cone sensitivities as rows of $C_{\text{eye}} \in \mathbb{R}^{3 \times N}$, the response is the **linear map**

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = C_{\text{eye}} l(\lambda)$$

- Again, we use linear systems as we did for cameras, but applied to the eye: a 3-row linear sensor over an N -dimensional spectral input

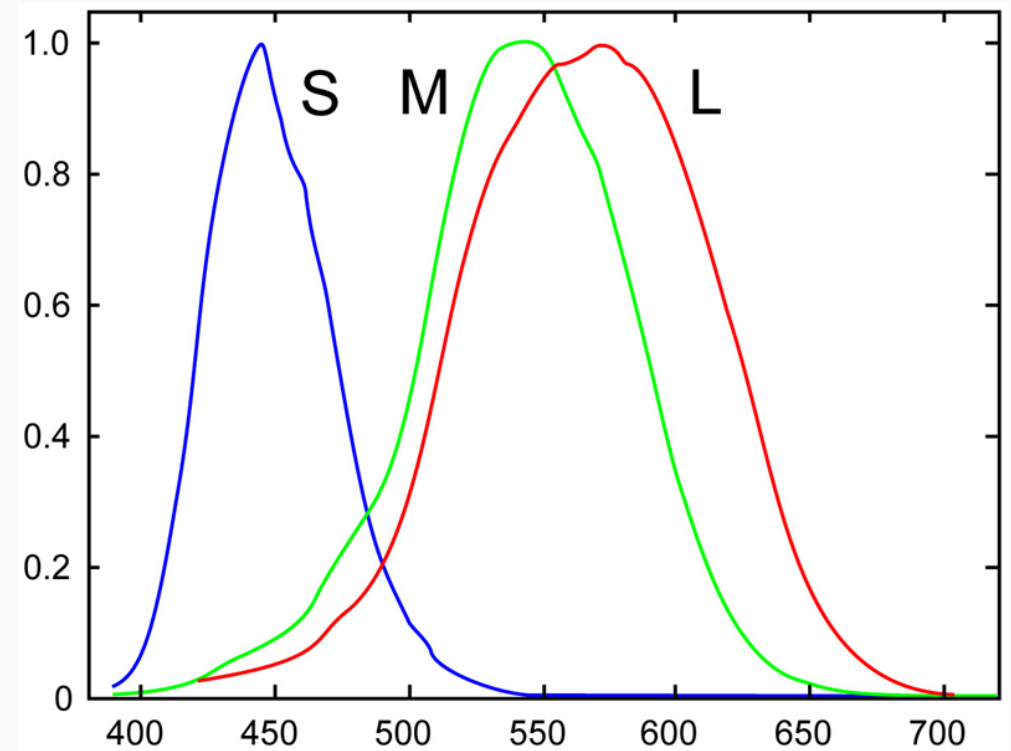


Figure 38: source: visionbook

- **Metamerism**: two physically different light spectra that produce the **same three cone responses**, and therefore look identical to a human observer, even though they are different distributions of energy over wavelength
- Because $C_{\text{eye}} : \mathbb{R}^N \rightarrow \mathbb{R}^3$ is a linear map from a high-dimensional space to $\mathbb{R}^3$, its **null space** is huge: any spectrum can be perturbed by a null-space vector without changing perceived color at all
- Cameras (with their own three, different, sensor sensitivity curves) may **not** perceive these two spectra as identical, this is why colors can look correct to the eye but be recorded differently by a camera (or vice versa)
- Hyperspectral imaging (measuring many more than 3 wavelength bands) can resolve metamers that fool both eyes and RGB cameras, at a cost in sensor complexity

- Human spatial acuity is **not** uniform across color: we are much more sensitive to fine spatial detail in **luminance** (brightness) than in **chrominance** (color)
- Equivalently, blurring only the color channels of an image is far less noticeable than blurring only its luminance
- This asymmetry is exploited throughout image and video compression (e.g. chroma subsampling)



(a)



(b)



(c)

Figure 39: source: visionbook

Conclusions

Summary: Three Ambiguities

perspective
size $\leftrightarrow$ depth

linear system $l_s = Al_w$
ill-posed inversion

reflectance $\times$ illuminant
color constancy

- Every stage of image formation involves a **loss of information**:
 - geometry: a 3D scene to a 2D array
 - color: a continuous spectrum to 3 numbers
 - camera: an unknown linear mixture to a finite set of measurements
- Recognizing **what** information is discarded (and how) is the first step to recovering it, whether via extra views, priors, regularization, or learned models

- **pinhole/perspective projection** is the geometric model for all of Part 2; we will discuss a full camera model, and how to estimate depth and size
- Cameras as **linear systems** $l_s = Al_w$; recovering l_w is an ill-posed inverse problem, stabilized via **regularization**
- **Color** is a 3-number compression of a full spectrum; color is influenced by material and light

- VisionBook (Ch. 5, 6, 7, 8).
 - The book has more details than the slides

- Today we treated images as the output of a physical formation process
- Next lecture we treat images as **signals**, and build the tools (Fourier transform and convolution) to analyze and manipulate them