Computer Vision I

Image Formation (continued)

27.10.2014

Credits:

Slides adapted from Konrad Schindler including many slides from:

Svetlana Lazebnik
Steve Seitz
Fredo Durand
Alyosha Efros

Stefan Roth

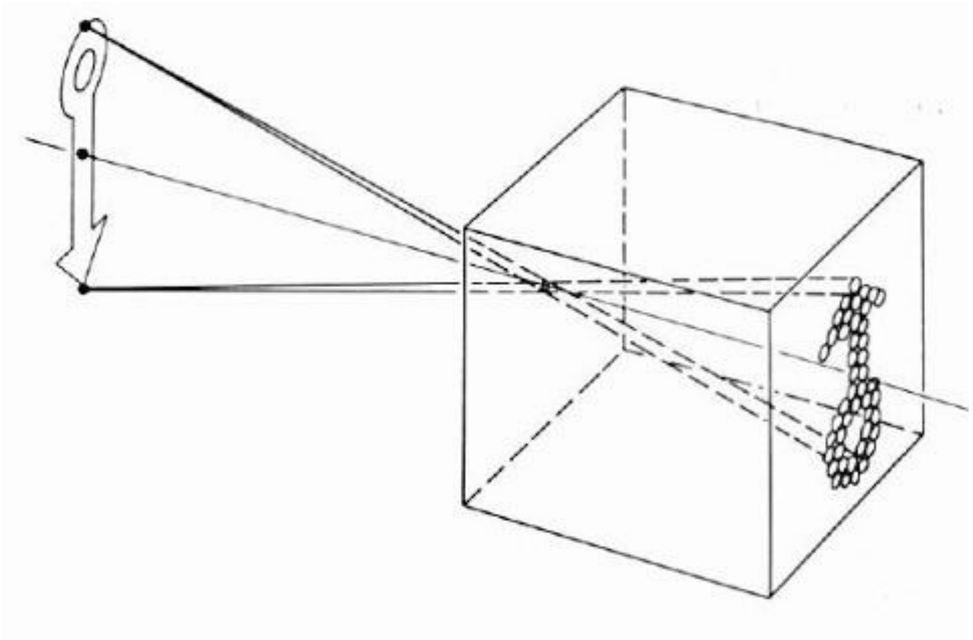
Overview

- The pinhole projection model
  - Properties of pinhole cameras
  - Algebraic model: Perspective projection matrix
- Single view geometry
  - Projection from world to image coordinates
  - Camera calibration
- Cameras with lenses
  - Depth of focus, field of view, lens aberrations
- Digital cameras
  - Types of sensors, color

Pinhole camera model

- Pinhole model
  - Captures pencil of rays – all rays through a single point
  - Projection rays are straight lines
  - The point is called *center of projection* (focal point)

  - The image is formed on the image plane
  - Two equivalent projections: positive, negative



Camera parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - Skew (non-rectangular pixels)
  - Radial distortion

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$

radial distortion

linear image

[from S. Lazebnik]

Camera parameters

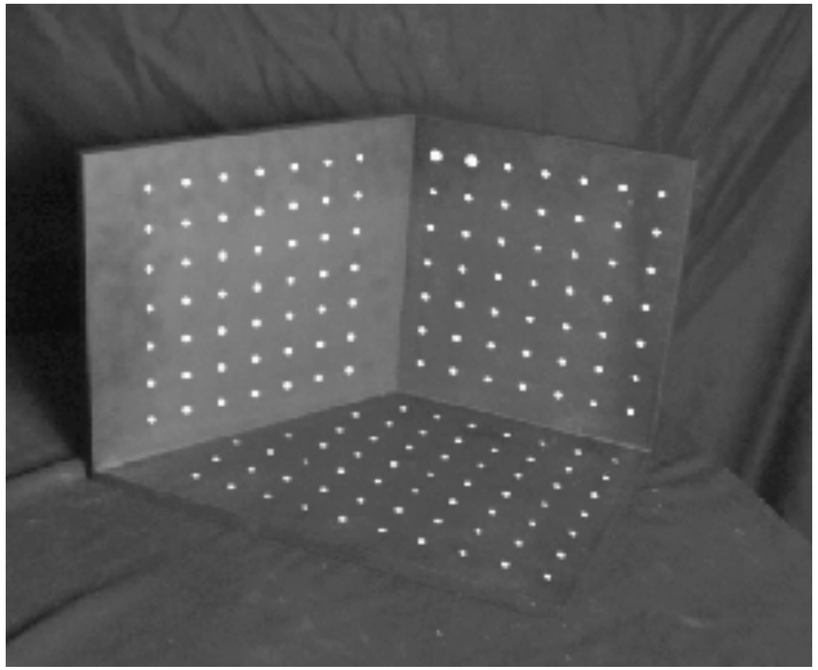
- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - Skew (non-rectangular pixels)
  - Radial distortion

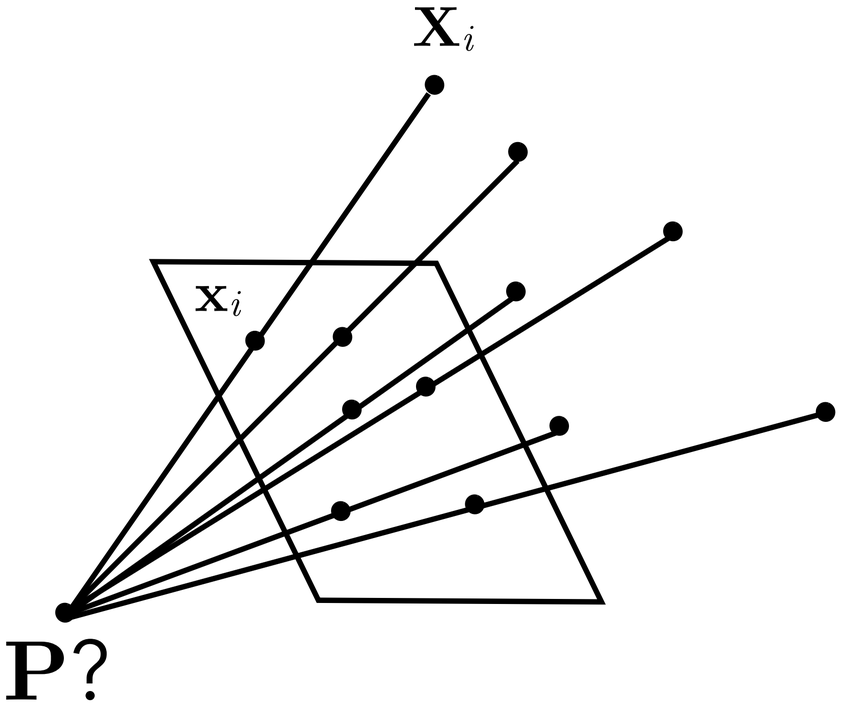
- Extrinsic parameters
  - Rotation and translation relative to world coordinate system

[from S. Lazebnik]

Camera calibration

- Given n points with known 3D coordinates $\mathbf{X}_i$ and known image projections $\mathbf{x}_i$, estimate the camera parameters





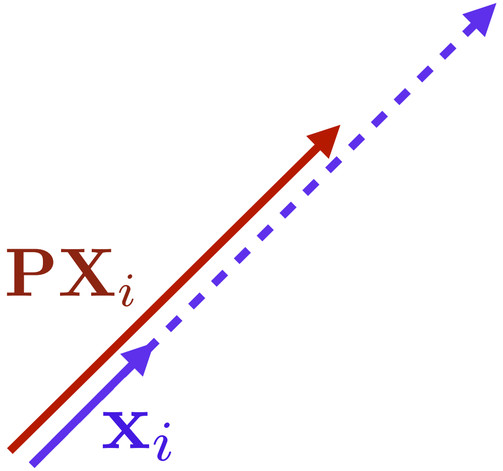
[from S. Lazebnik]

Camera calibration

$\mathbf{p}_j^{\mathrm{T}}$ are the rows of $\mathbf{P}$

- Use *collinearity*

$$\lambda \mathbf{x}_i = \mathbf{P}\mathbf{X}_i \qquad \lambda \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \mathbf{p}_3^{\mathrm{T}} \end{bmatrix} \mathbf{X}_i$$



- Write using cross-product $\mathbf{x}_i \times \mathbf{P}\mathbf{X}_i = 0$

- Expand cross-product

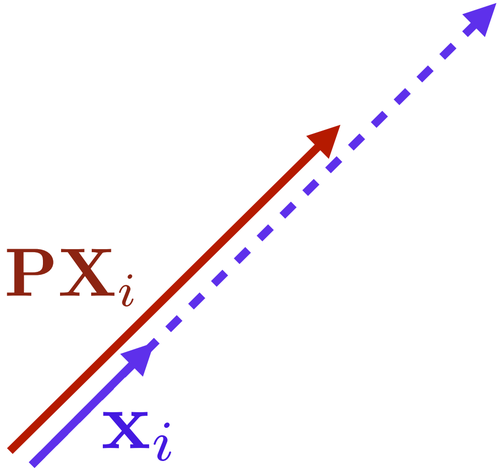
$$\begin{pmatrix} y_i \mathbf{p}_3^{\mathrm{T}} \mathbf{X}_i - \mathbf{p}_2^{\mathrm{T}} \mathbf{X}_i \\ \mathbf{p}_1^{\mathrm{T}} \mathbf{X}_i - x_i \mathbf{p}_3^{\mathrm{T}} \mathbf{X}_i \\ x_i \mathbf{p}_2^{\mathrm{T}} \mathbf{X}_i - y_i \mathbf{p}_1^{\mathrm{T}} \mathbf{X}_i \end{pmatrix} = 0$$

Camera calibration

$\mathbf{p}_j^{\mathrm{T}}$ are the rows of $\mathbf{P}$

- Use collinearity

$$\lambda \mathbf{x}_i = \mathbf{P}\mathbf{X}_i \qquad \lambda \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \mathbf{p}_3^{\mathrm{T}} \end{bmatrix} \mathbf{X}_i$$



- Write using cross-product

$$\begin{pmatrix} y_i \mathbf{p}_3^{\mathrm{T}} \mathbf{X}_i - \mathbf{p}_2^{\mathrm{T}} \mathbf{X}_i \\ \mathbf{p}_1^{\mathrm{T}} \mathbf{X}_i - x_i \mathbf{p}_3^{\mathrm{T}} \mathbf{X}_i \\ x_i \mathbf{p}_2^{\mathrm{T}} \mathbf{X}_i - y_i \mathbf{p}_1^{\mathrm{T}} \mathbf{X}_i \end{pmatrix} = 0$$

entries of $\mathbf{P}$ stacked up in a vector

- Write as equation system

$$\begin{bmatrix} 0 & -\mathbf{X}_i^{\mathrm{T}} & y_i \mathbf{X}_i^{\mathrm{T}} \\ \mathbf{X}_i^{\mathrm{T}} & 0 & -x_i \mathbf{X}_i^{\mathrm{T}} \\ -y_i \mathbf{X}_i^{\mathrm{T}} & x_i \mathbf{X}_i^{\mathrm{T}} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{0}$$

Two linearly independent equations!

Camera calibration

- P has 11 degrees of freedom (12 parameters, but scale is arbitrary)
- One 2D/3D correspondence gives us two linearly independent equations
- 6 correspondences needed for a minimal solution
- more points: least-squares estimation

$$\begin{bmatrix} 0 & -\mathbf{X}_1^{\mathrm{T}} & y_1\mathbf{X}_1^{\mathrm{T}} \\ \mathbf{X}_1^{\mathrm{T}} & 0 & -x_1\mathbf{X}_1^{\mathrm{T}} \\ \vdots & \vdots & \vdots \\ 0 & -\mathbf{X}_n^{\mathrm{T}} & y_n\mathbf{X}_n^{\mathrm{T}} \\ \mathbf{X}_n^{\mathrm{T}} & 0 & -x_n\mathbf{X}_n^{\mathrm{T}} \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{A}\mathbf{p} = \mathbf{0}$$

equation system matrix

projection matrix as vector

Camera calibration

- Problem: *trivial solution* $\mathbf{p} = 0$

- Solution: add constraint on norm of $\mathbf{p}$ (value can be arbitrary)

$$\mathbf{A}\mathbf{p} = 0 \quad \text{s.t.} \quad \|\mathbf{p}\| = 1$$

homogeneous least squares

- In practice: Find the right nullspace of $\mathbf{A}$ using *singular value decomposition* (SVD)
  - Decompose equation system matrix $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$
  - Assume that singular values are sorted, i.e. $\mathbf{S} = \mathrm{diag}(s_1, \ldots, s_{12}), \quad s_{i+1} \geq s_i$
  - Take last right singular vector $\mathbf{p} = \mathbf{v}_{12}$

- Why does this work?

Camera calibration

- Once we've recovered the values of the camera matrix, we still have to figure out the *intrinsic* and *extrinsic* parameters
  - First split the projection matrix into a 3x3 matrix and a 3x1 vector

$$\mathbf{P} = [\mathbf{M}|\mathbf{m}] = [\mathbf{KR}| - \mathbf{KR}\tilde{\mathbf{C}}]$$

  - Next, decompose $\mathbf{M}$ into upper triangular part $\mathbf{K}$ (calibration) and orthonormal part $\mathbf{R}$ (rotation) using RQ-decomposition (see e.g. Golub & Van Loan, *Matrix Computations*)
    - if required, extract 3 rotation axis and angle from $\mathbf{R}$ - its columns are the axes of the rotated coordinate system)
  - Finally, find $\mathbf{C}$ as the nullspace of $\mathbf{P}$ by means of SVD.

Camera calibration

- What have we got now?
  - We have a projection matrix $\mathbf{P}$, which maps any point in the 3D world coordinate frame onto the (infinite) image plane of the camera
  - Given a 3D point in the world, we can find out where its projection is in the image
  - One goal of vision: given a point in the images, find out where there pre-image is in the world

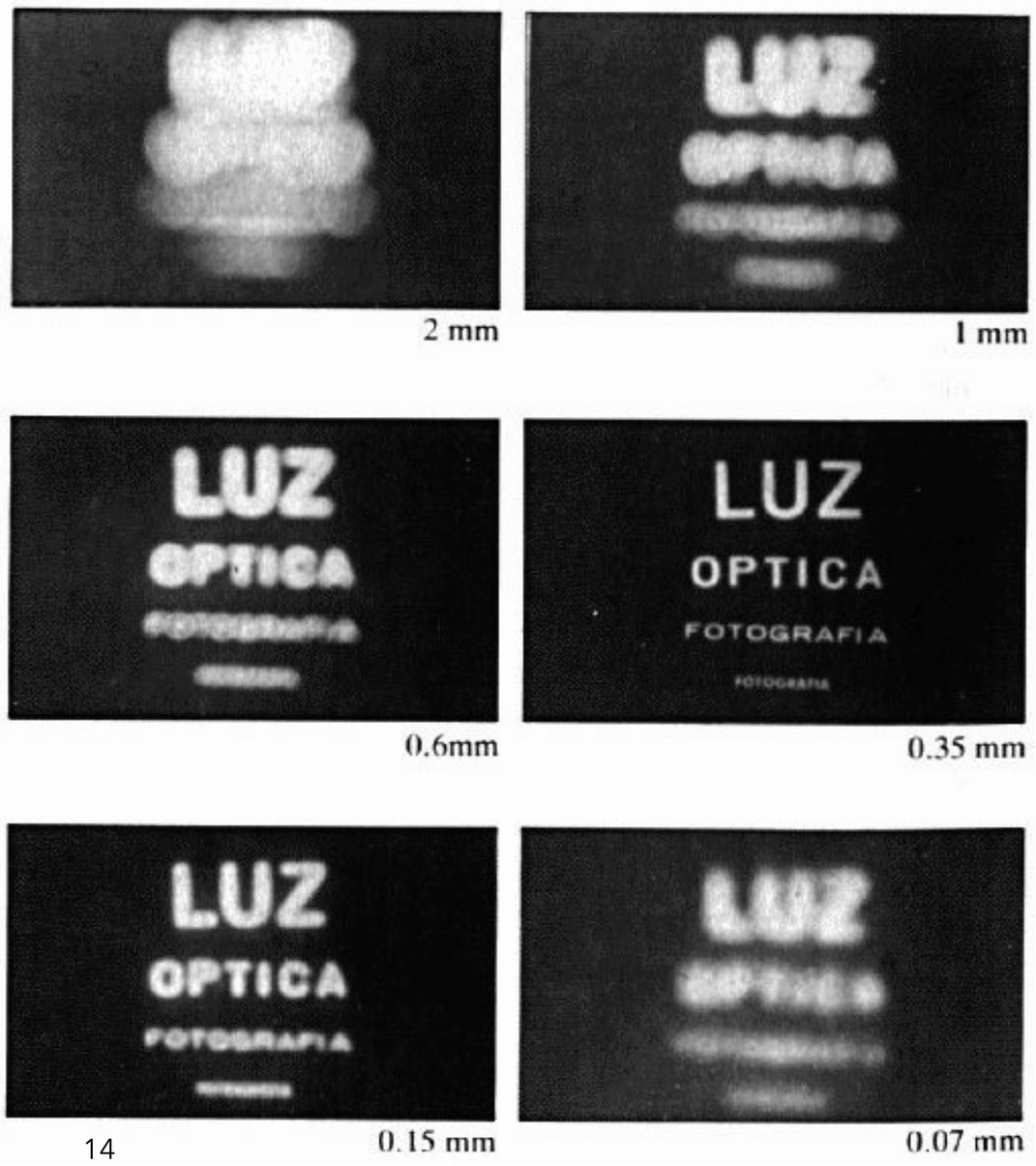
- We cannot do this from one image
  - The depth is lost, the projection matrix can only tell us the ray through an image point $\mathbf{x}$

Building a real camera

[from S. Lazebnik]

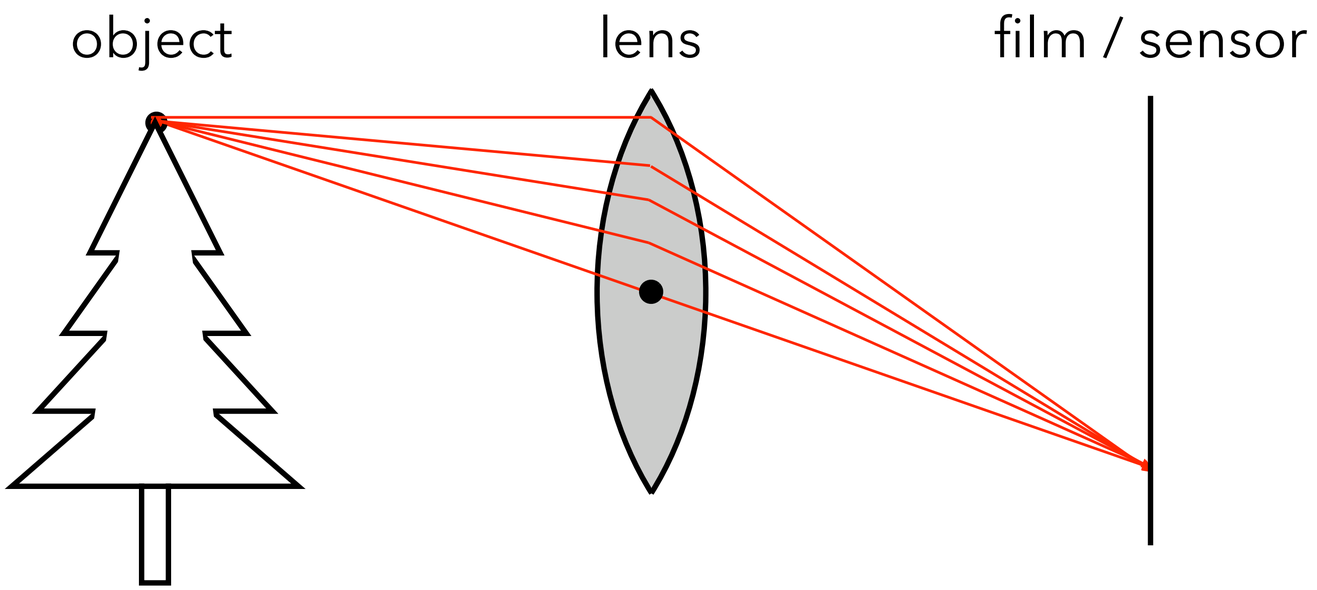
Shrinking the aperture

- Why not make the aperture as small as possible?
  - Less light gets through
  - Diffraction effects…



Adding a lens

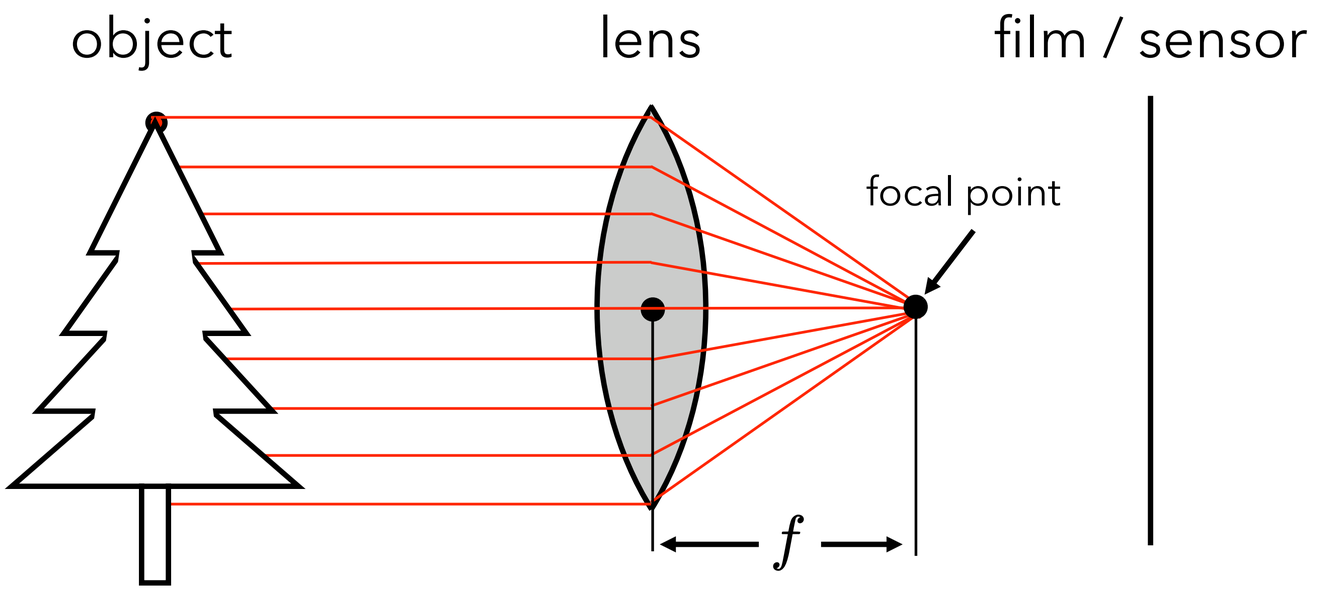
- A lens focuses light onto the film
  - Rays passing through the center are not deviated



[from Steve Seitz]

Adding a lens

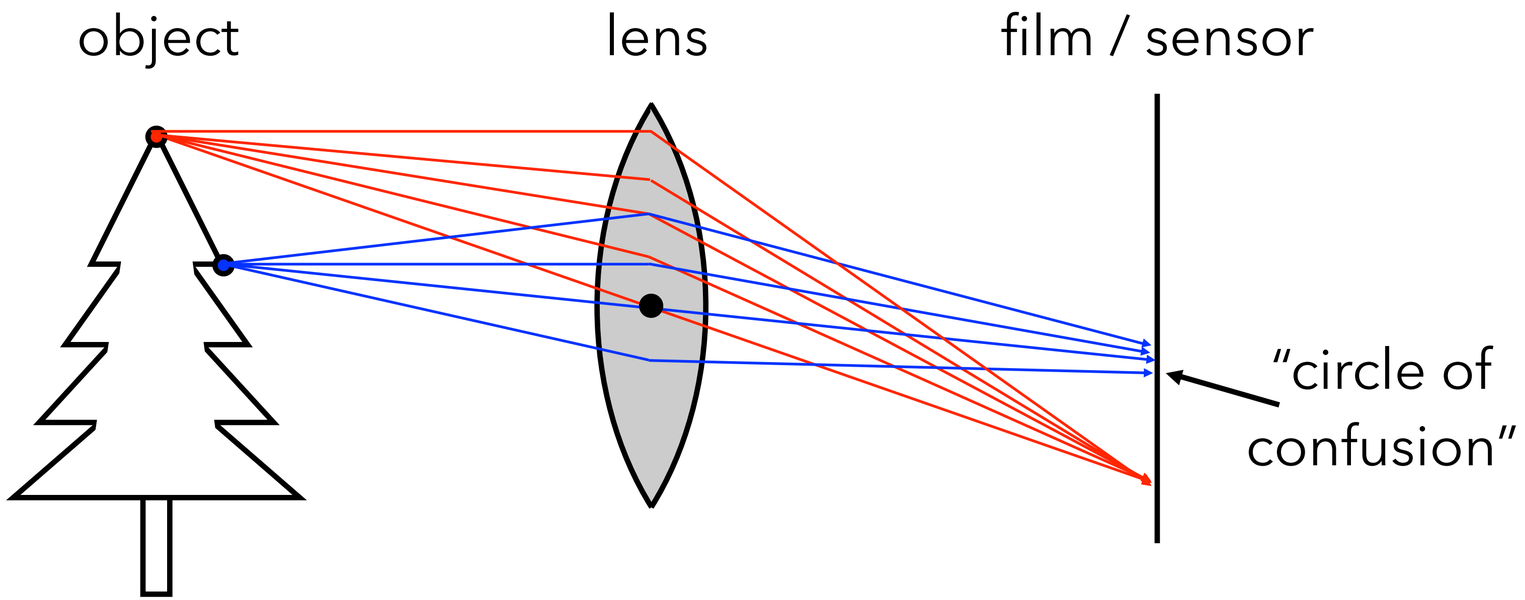
- A lens focuses light onto the film
  - Rays passing through the center are not deviated
  - All parallel rays converge to one point on a plane located at the focal length f



[from Steve Seitz]

Adding a lens

- A lens focuses light onto the film
  - There is a specific distance at which objects are “in focus”
  - other points project to a “circle of confusion” in the image

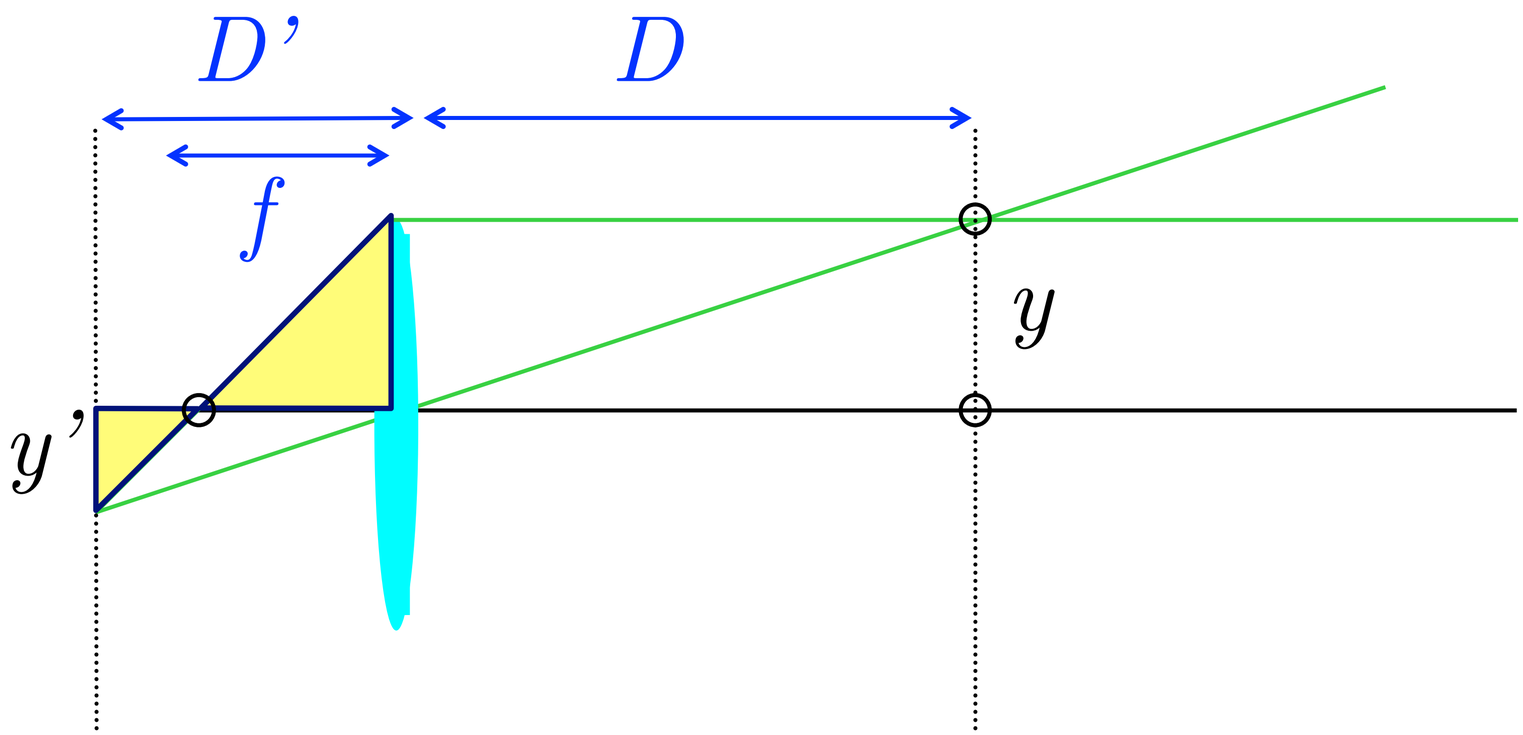


[from Steve Seitz]

Thin lens formula

Apply similar triangles twice:

$$\frac{y'}{y} = \frac{D'}{D} \qquad \frac{y'}{y} = \frac{D' - f}{f}$$

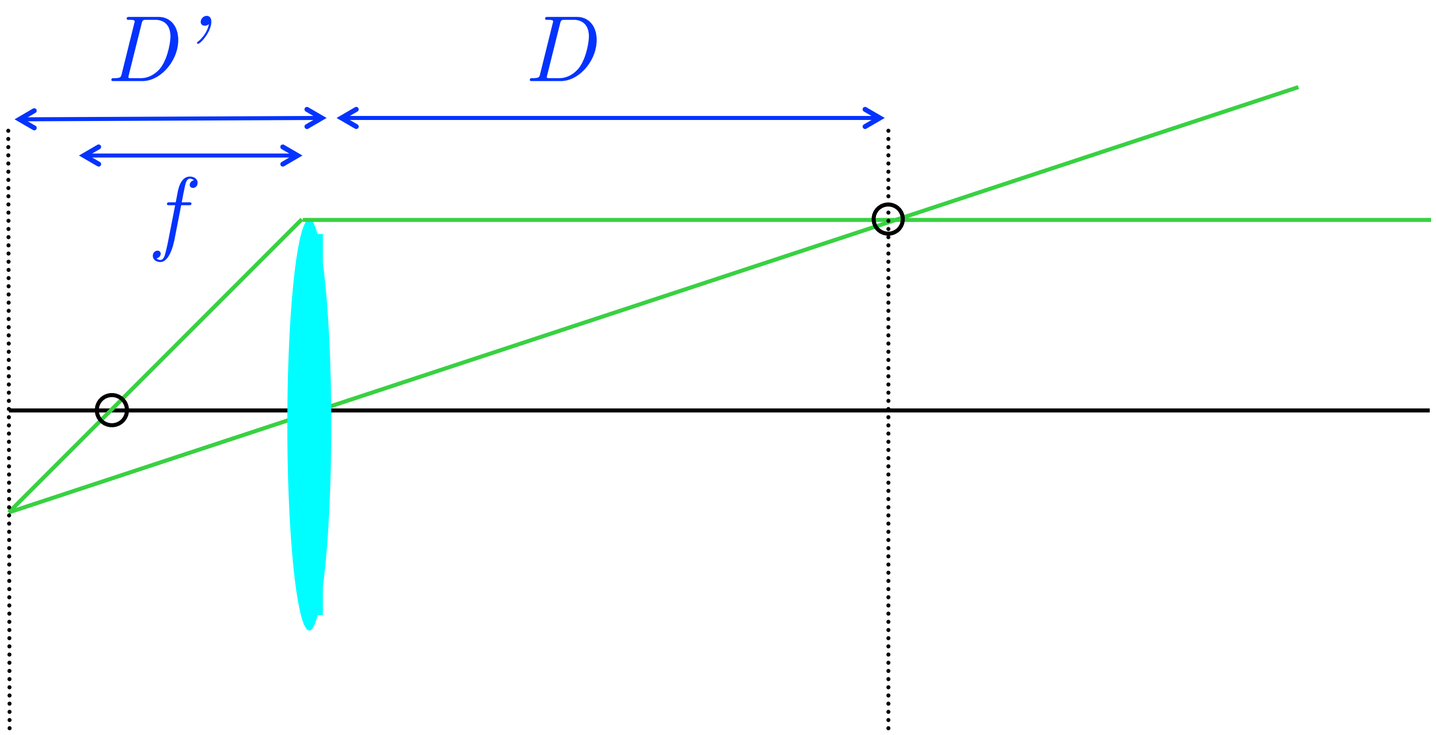


[from F. Durand]

Thin lens formula

$$\frac{1}{D'} + \frac{1}{D} = \frac{1}{f}$$

Any point satisfying the thin lens equation is in focus.



[from F. Durand]

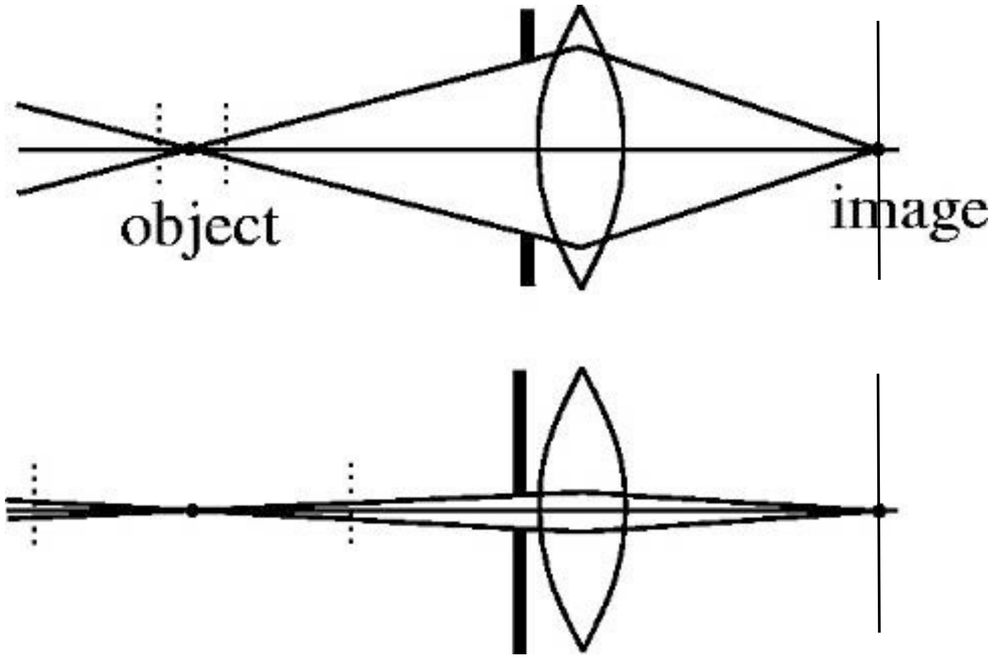
Depth of Field

http://www.cambridgeincolour.com/tutorials/depth-of-field.htm

[from A. Efros]

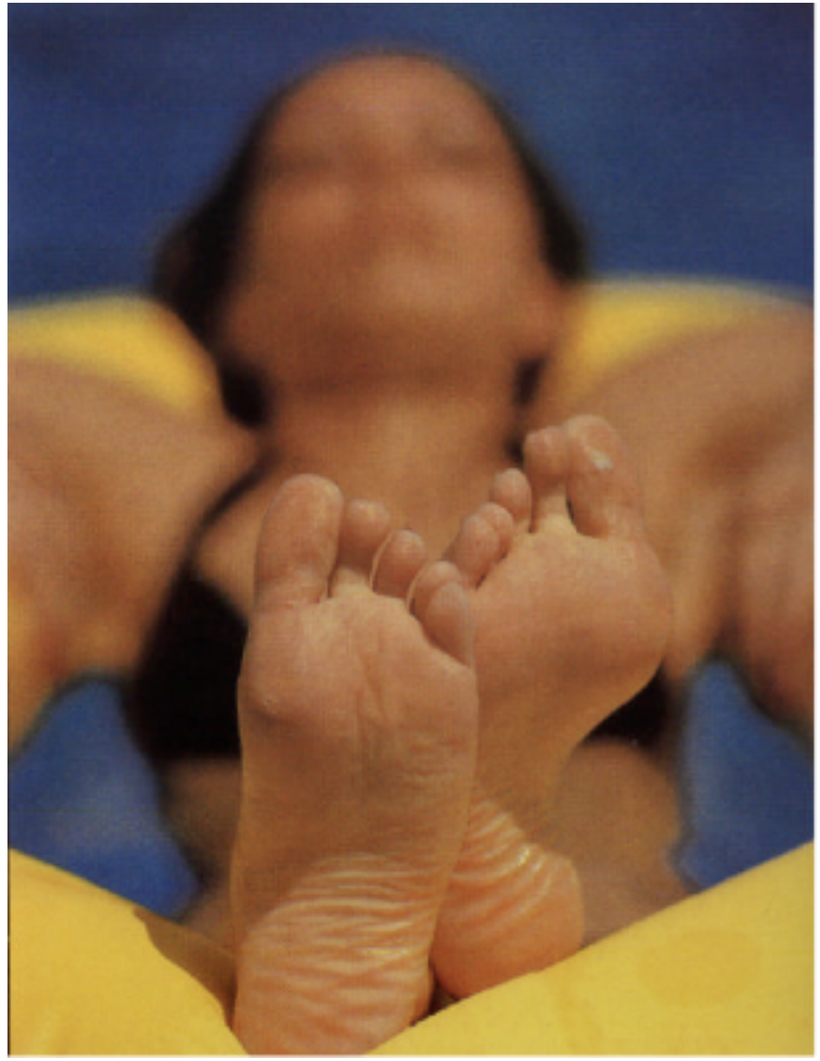
How can we control the depth of field?

- Changing the aperture size affects depth of field
  - A smaller aperture increases the range in which the object is approximately in focus
  - But small aperture reduces amount of light – need to increase exposure



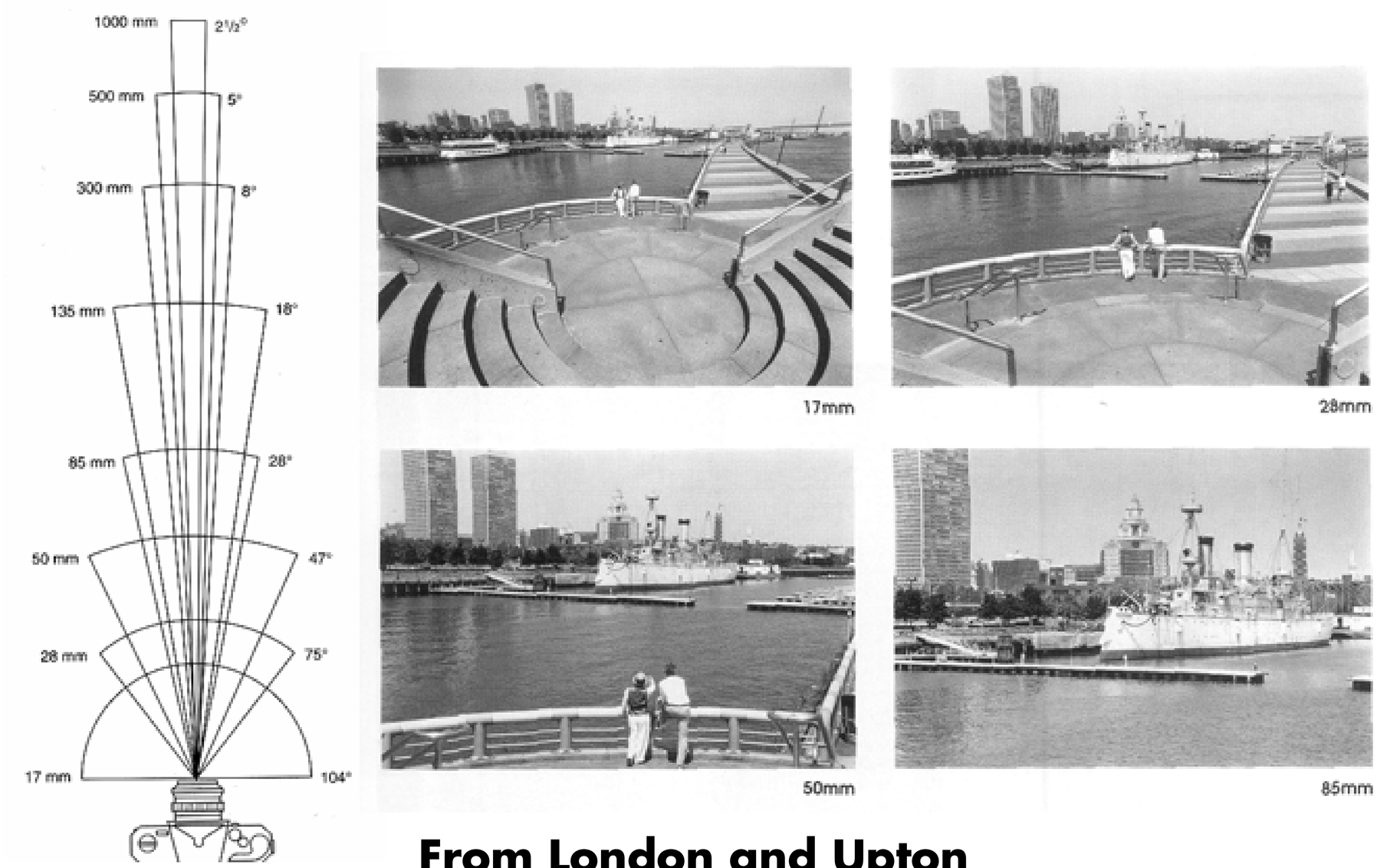
[from A. Efros]

Depth of Field Effect



[from F. Durand]

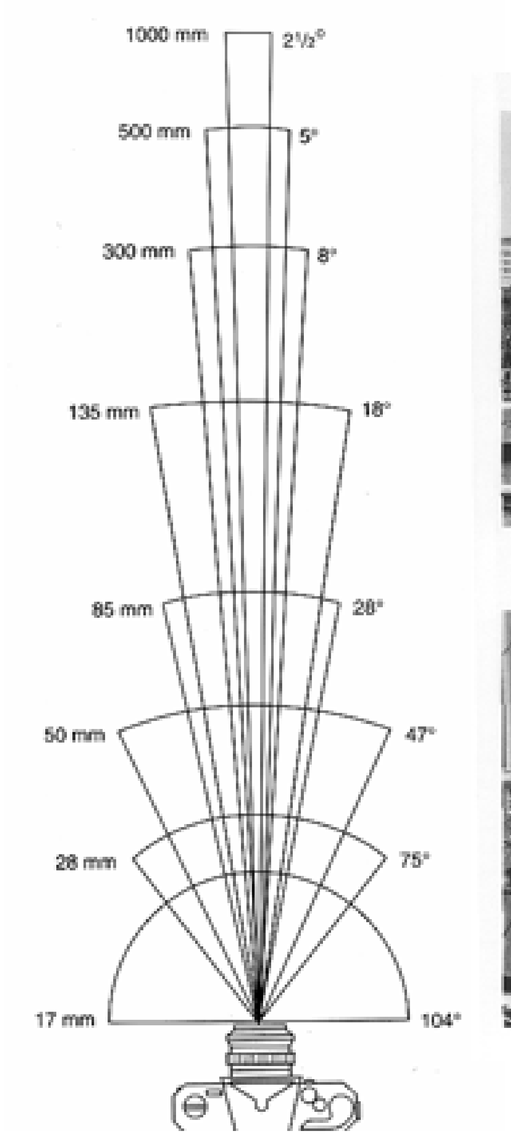
Field of View (Zoom)

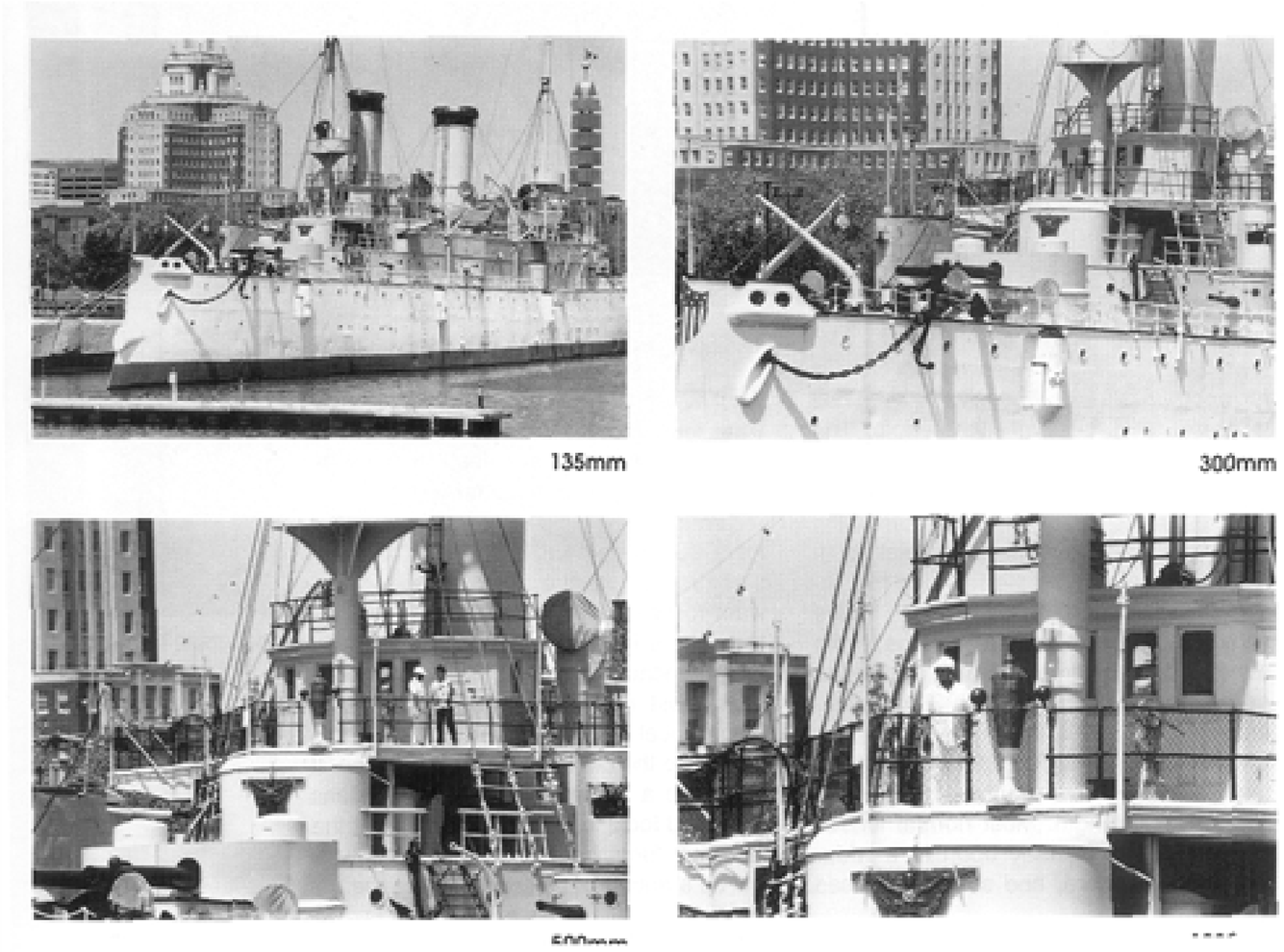


From London and Upton

[from A. Efros]

Field of View (Zoom)

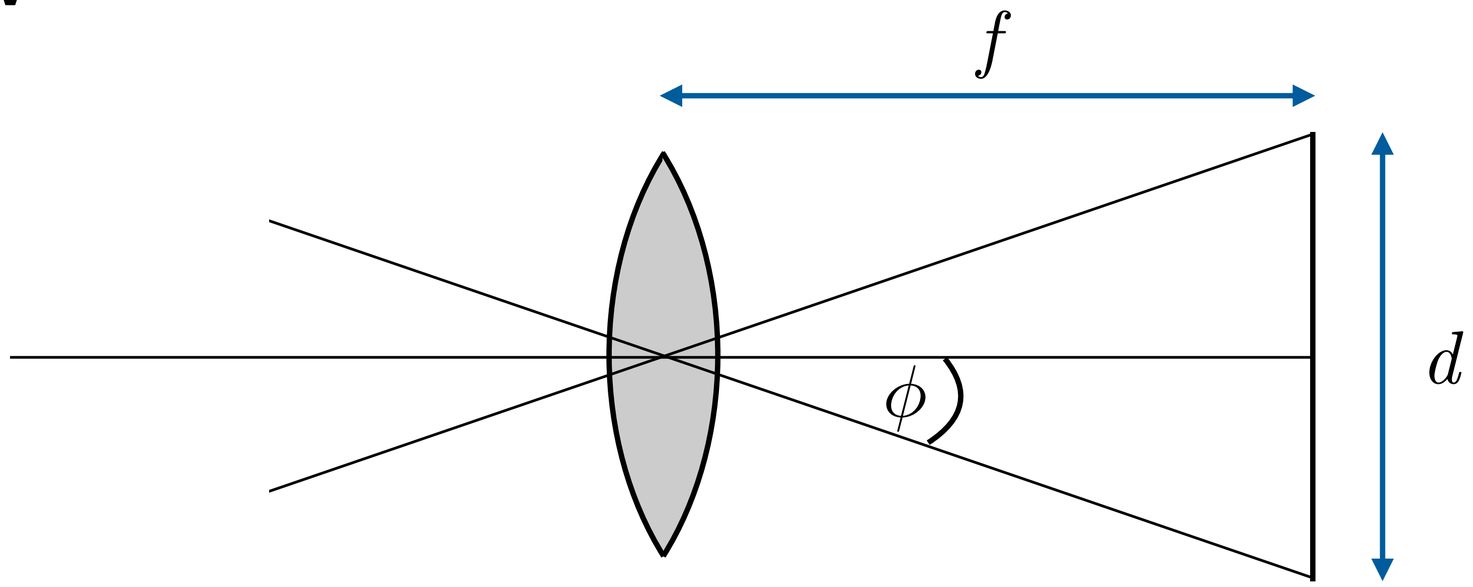




From London and Upton

[from A. Efros]

Field of View



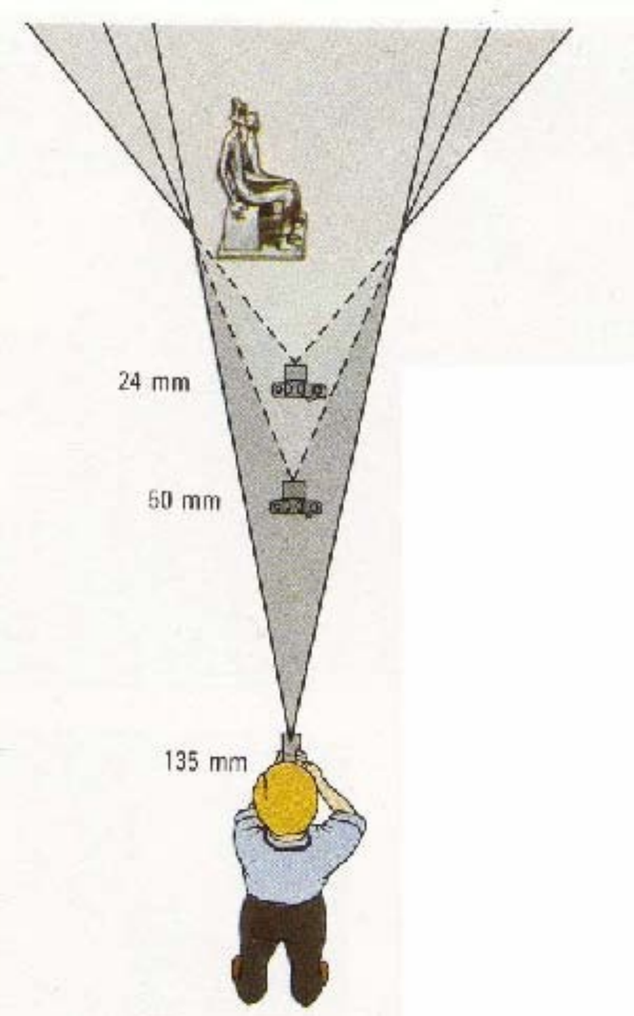
FOV depends on focal length and
the size of the camera retina (sensor)

$$\phi = \tan^{-1}\left(\frac{d}{2f}\right)$$

Smaller FOV = larger focal length

[from A. Efros]

Field of View / Focal Length



Large FOV, small f

Camera close to car

Small FOV, large f

Camera far from the car

[from A. Efros, F. Durand]

Same effect for faces

wide-angle

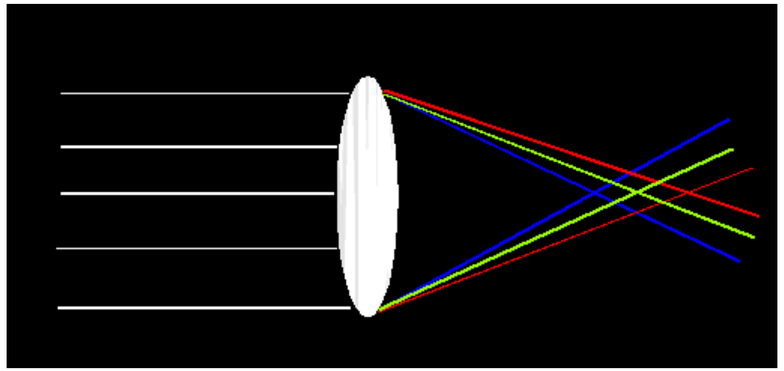
standard

telephoto

[from F. Durand]

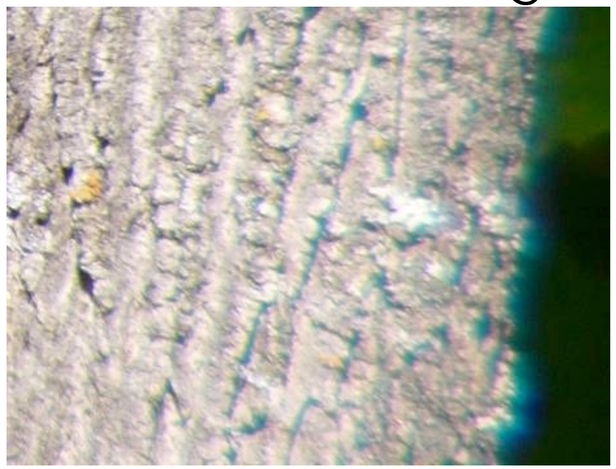
Lens flaws: Chromatic aberration

- Lens has different refractive indices for different wavelengths: causes color fringing



near lens center

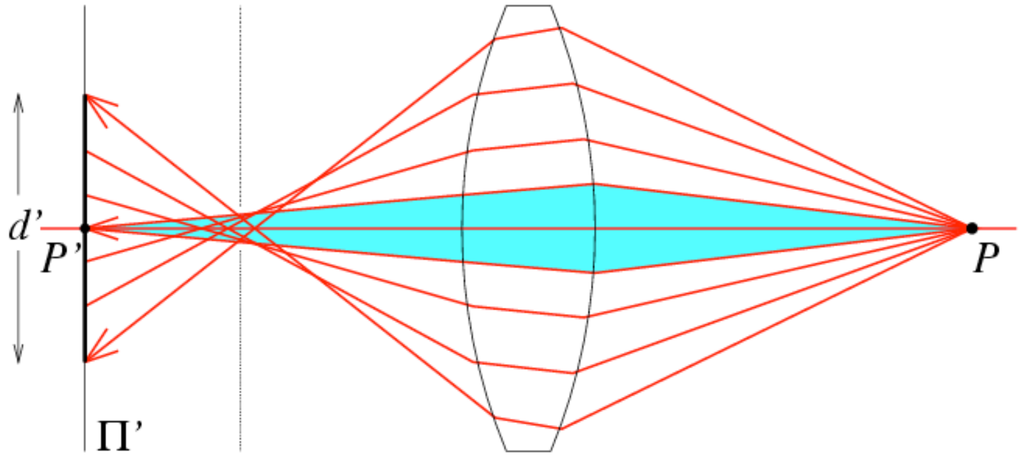
near lens outer edge



[from S. Lazebnik]

Lens flaws: Spherical aberration

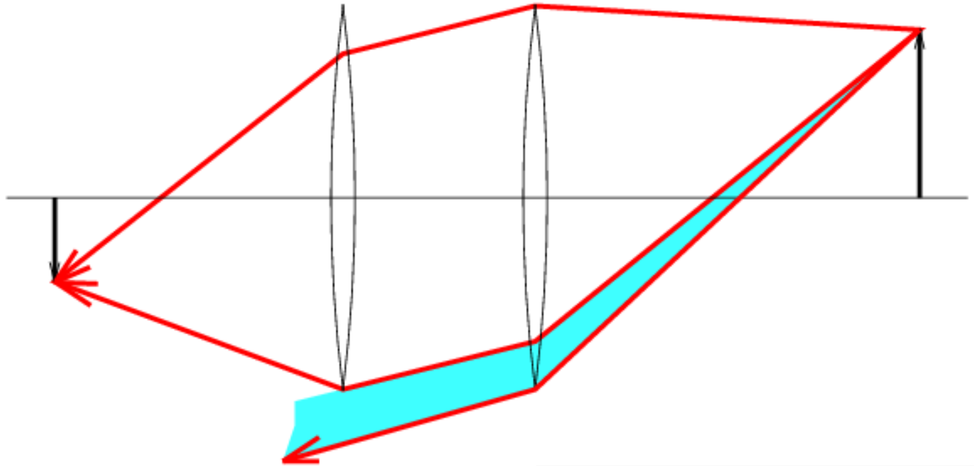
- Spherical lenses don't focus light perfectly
  - Rays farther from the optical axis focus closer



  - Causes blur away from the image center

[from S. Lazebnik]

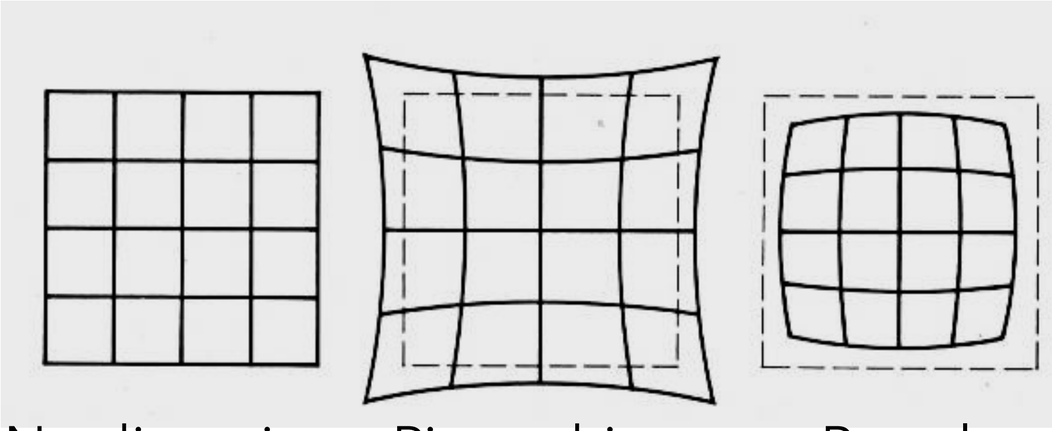
Lens flaws: Vignetting



[from S. Lazebnik]

Radial Distortion

- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens

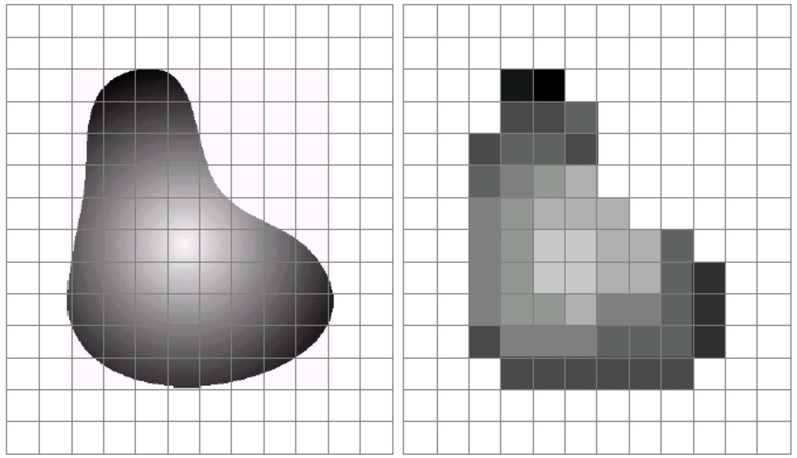


No distortion Pin cushion Barrel

[from S. Lazebnik]

Digital camera

- A digital camera replaces film with a sensor array
  - Each cell in the array is light-sensitive diode that converts photons to electrons
  - Two common types
    - Charge Coupled Device (CCD)
    - Complementary metal oxide semiconductor (CMOS)



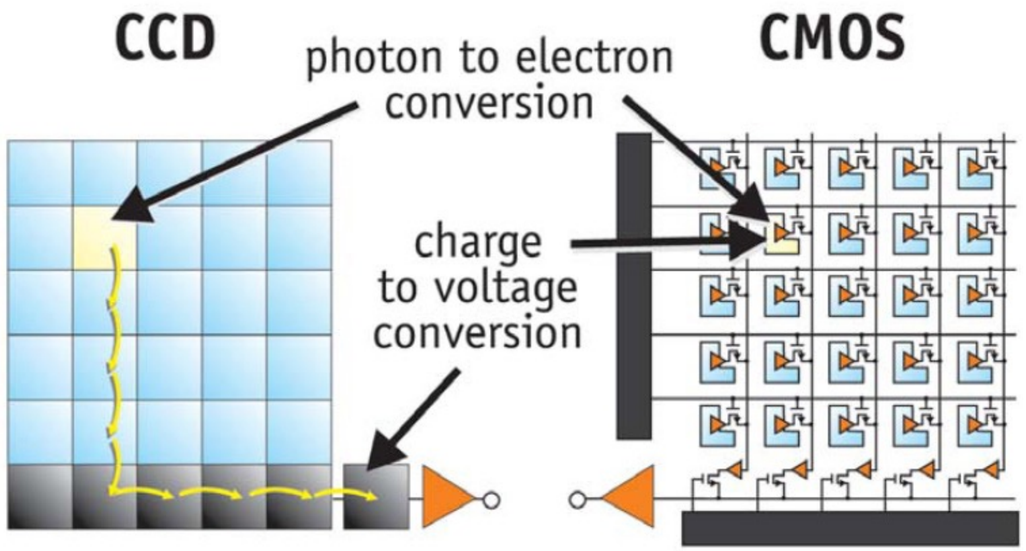
spatially continuous

spatially sampled

[from Steve Seitz]

CCD vs. CMOS

- CCD: transports the charge across the chip and reads it at one corner of the array.
- An analog-to-digital converter (ADC) then turns each pixel's value into a digital value by measuring the amount of charge at each photosite and converting that measurement to binary form.



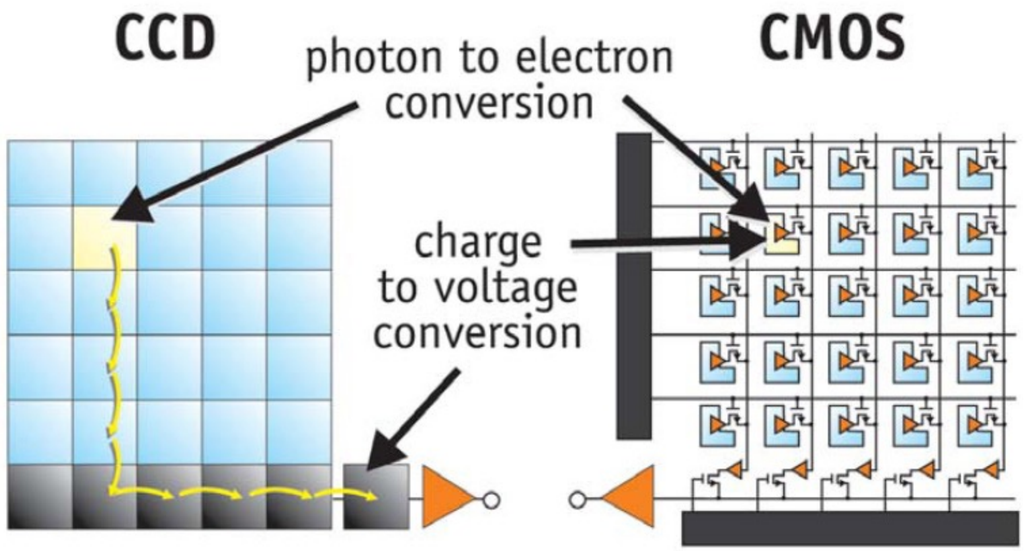
CCDs move photogenerated charge from pixel to pixel and convert it to voltage at an output node. CMOS imagers convert charge to voltage inside each pixel.

http://www.dalsa.com/shared/content/pdfs/CCD_vs_CMOS_Litwiller_2005.pdf

[from S. Lazebnik]

CCD vs. CMOS

- CMOS: uses several transistors at each pixel to amplify and move the charge using more traditional wires.
- The CMOS signal is digitized right away, so it needs no separate ADC.
  - Also often faster (video applications)



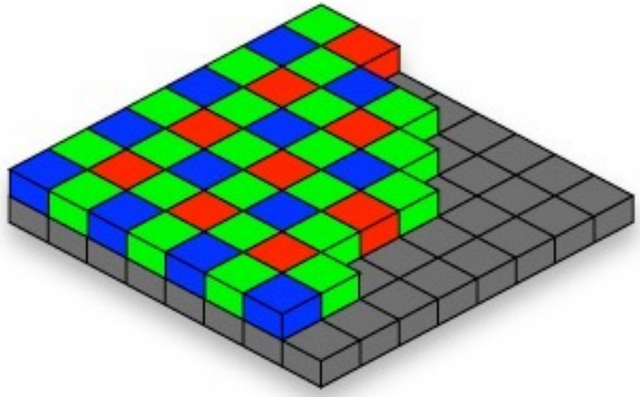
CCDs move photogenerated charge from pixel to pixel and convert it to voltage at an output node. CMOS imagers convert charge to voltage inside each pixel.

http://www.dalsa.com/shared/content/pdfs/CCD_vs_CMOS_Litwiller_2005.pdf

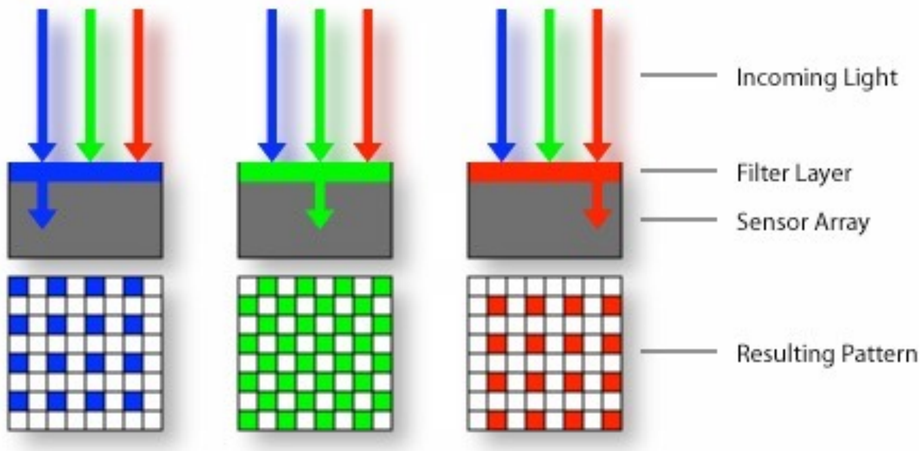
[from S. Lazebnik]

Color sensing in camera: Color filter array

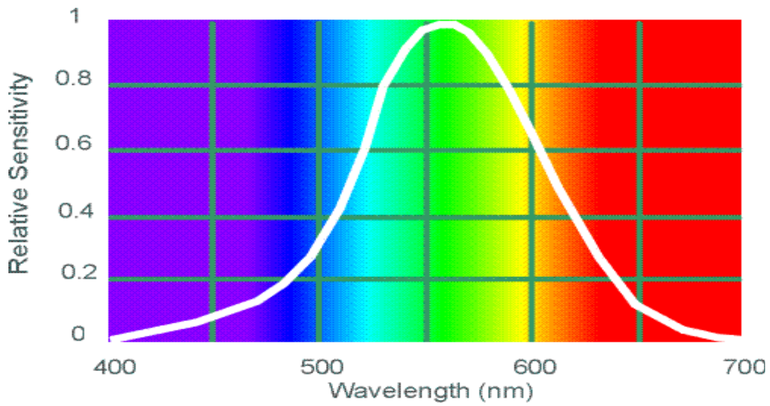
Bayer grid



Estimate missing components from neighboring values (demosaicing)



Why more green?



Human Luminance Sensitivity Function

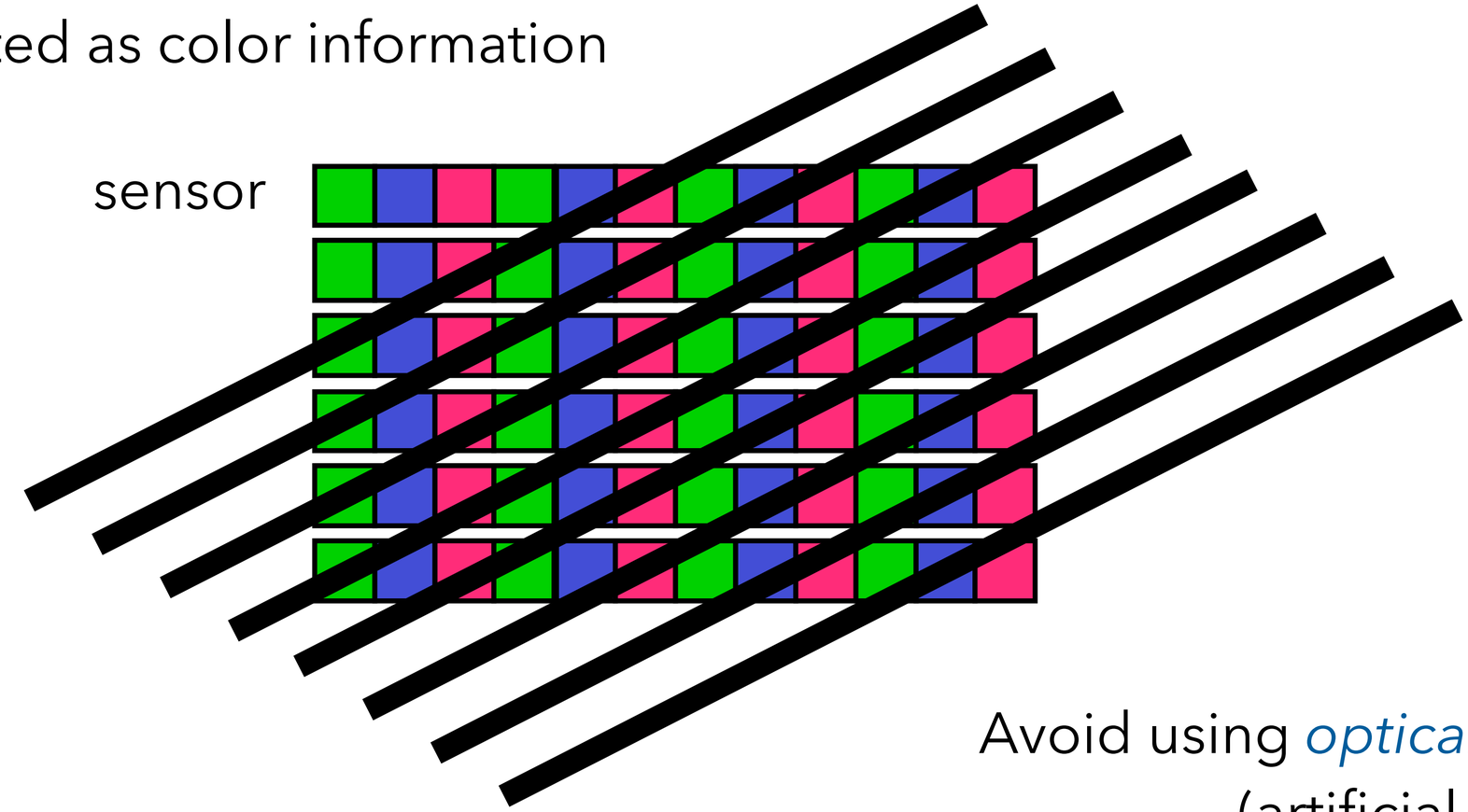
[from Steve Seitz]

Problem with demosaicing: color moiré

[from F. Durand]

The cause of color moiré

Fine black and white detail in image misinterpreted as color information

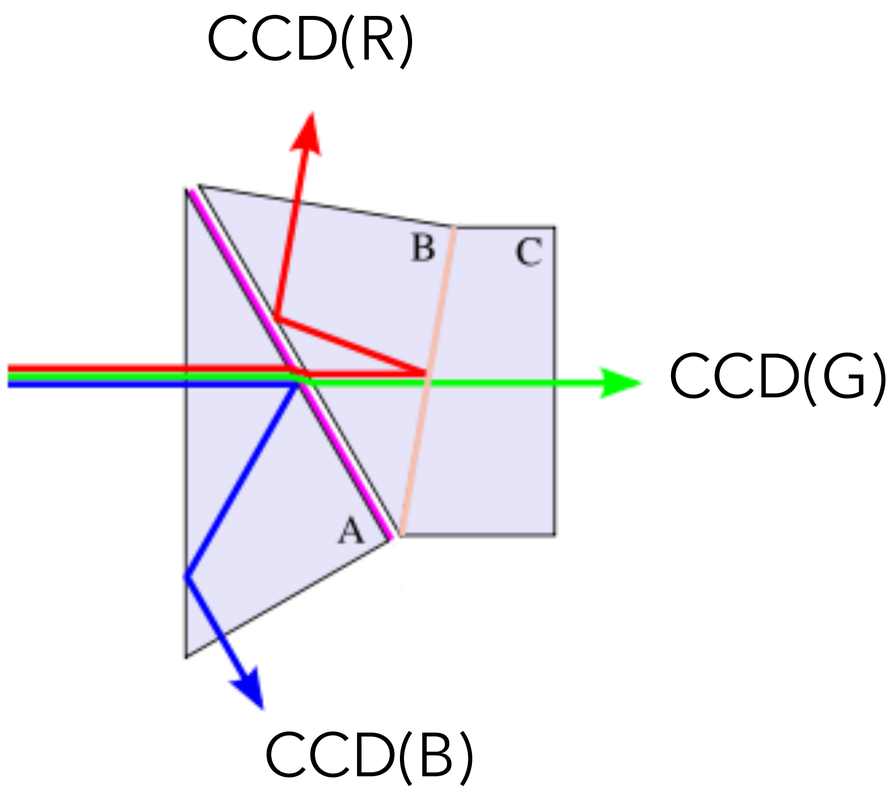


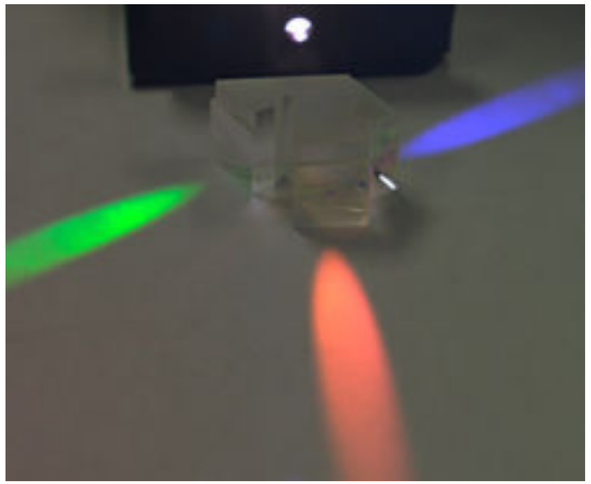
Avoid using *optical low-pass filter* (artificial blur)

[from F. Durand]

Color sensing in camera: Prism

- Requires three chips and precise alignment
- More expensive, big & heavy

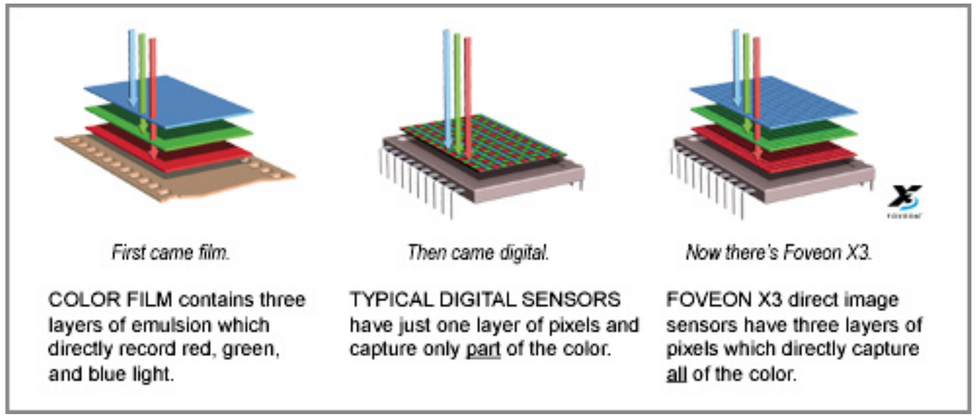




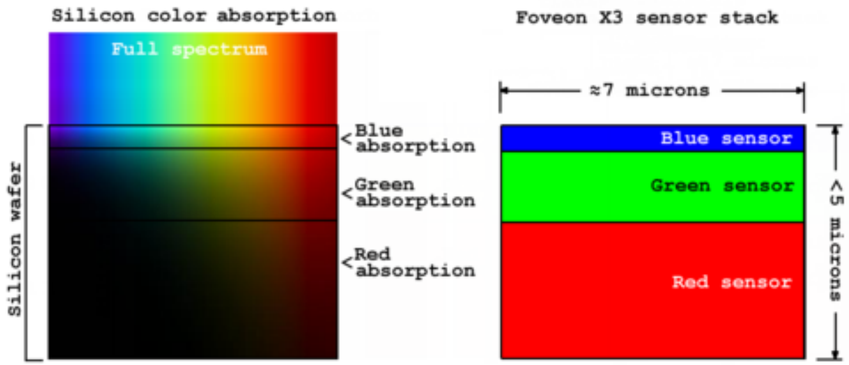
[from S. Lazebnik]

Color sensing in camera: Foveon X3

- CMOS sensor
- Takes advantage of the fact that red, blue and green light penetrate silicon to different depths

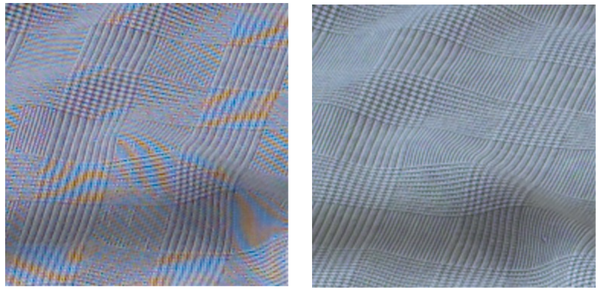


http://www.foveon.com/article.php?a=67



http://en.wikipedia.org/wiki/Foveon_X3_sensor

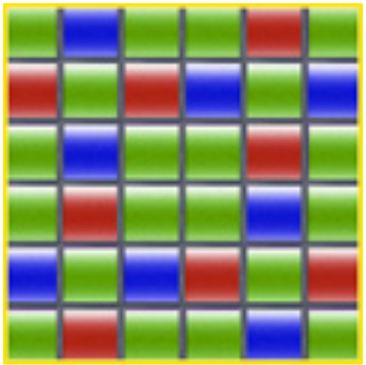
better image quality



[from M. Pollefeys]

Color sensing in camera: X-Trans

- CMOS sensor
- Use color filters as in Bayer sensor, but with pseudo-random spatial arrangement



X-trans color pattern

http://fujifilm-x.com/development_story/en/sensor/

- Avoids some demosaicing artefacts that are due to the regular Bayer grid
- Requires more complex demosaicing procedure

Issues with digital cameras

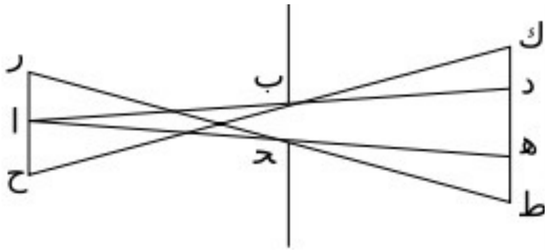
- Noise
  - low light is where you most notice noise
  - light sensitivity (ISO) / noise tradeoff
  - stuck pixels
- Resolution: Are more megapixels better?
  - requires higher quality lens
  - noise issues
- In-camera processing
  - oversharpening can produce halos
- RAW vs. compressed
  - file size vs. quality tradeoff
- Blooming
  - charge overflowing into neighboring pixels
  - white balance

More info online:

- http://electronics.howstuffworks.com/digital-camera.htm
- http://www.dpreview.com/
- http://www.dxomark.com/

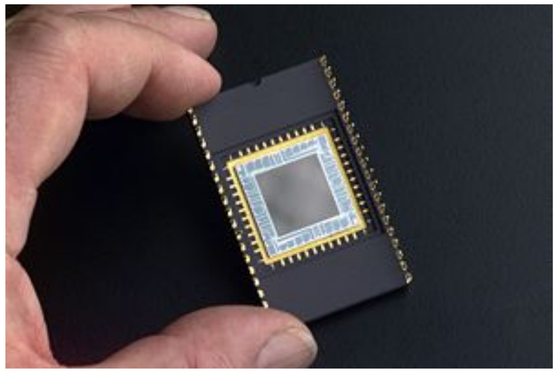
Historical context

- Pinhole model: Mozi (470-390 BC), Aristoteles (384-322 BC)
- Principles of optics (including lenses): Alhazen (965-1039 AD)
- Camera obscura: Leonardo da Vinci (1452-1519), Johann Zahn (1631-1707)
- First photo: Joseph Nicephore Niepce (1822)
- Daguerréotypes (1839)
- Photographic film (Eastman, 1889)
- Cinema (Lumière Brothers, 1895)
- Color photography (Lumière Brothers, 1908)
- Television (Baird, Farnsworth, Zworykin, 1920s)
- First consumer camera with CCD: Sony Mavica (1981)
- First fully digital camera: Kodak DCS100 (1990)



Alhazen's notes

Niepce, "La Table Servie," 1822



CCD chip [from S. Lazebnik]

Computer Vision I

Basic Image Processing

27.10.2014

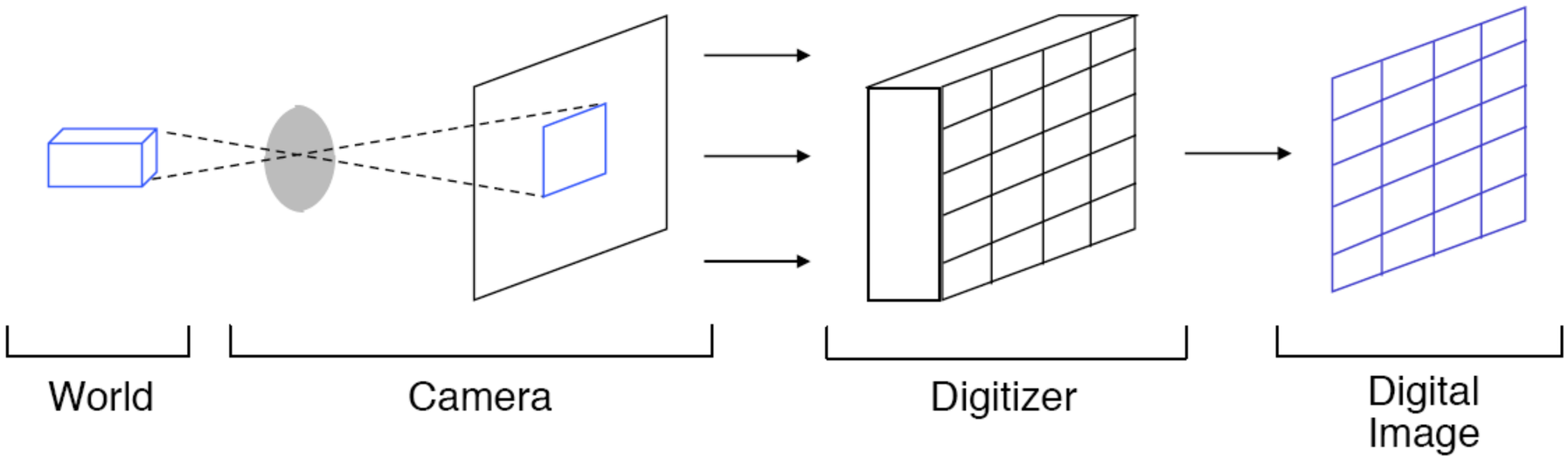
Credits:

Slides partly adapted from:
Bernt Schiele
Michael Black

Stefan Roth

Overview

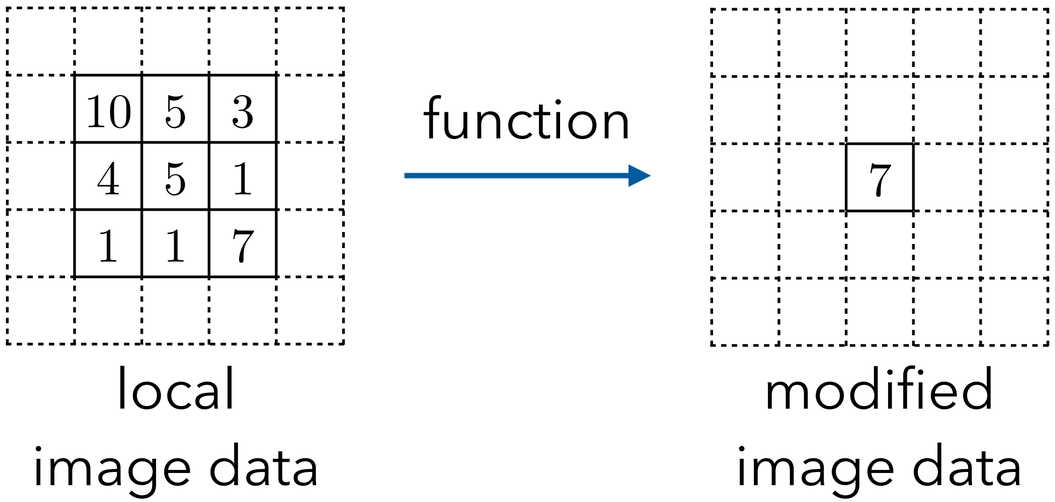
- Computer Vision: “invert” the imaging process
  - Today: 2D (2-dimensional) digital image processing
  - Later: ‘pattern recognition’, image understanding



Digital Image Processing

- Some basics
  - (digital signal processing, FFT, …)
  - Image filtering

- Image filtering
  - to reduce noise
  - to fill-in missing values/information
  - to extract image features (e.g. edges/corners)
  - …



Today: Basics of Digital Image Processing

- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - Motivation: Recognition using line drawings
  - Image derivatives (1st and 2nd order)

What is an image?

- We can think of the image as a function:

$$I(x, y), \quad I : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

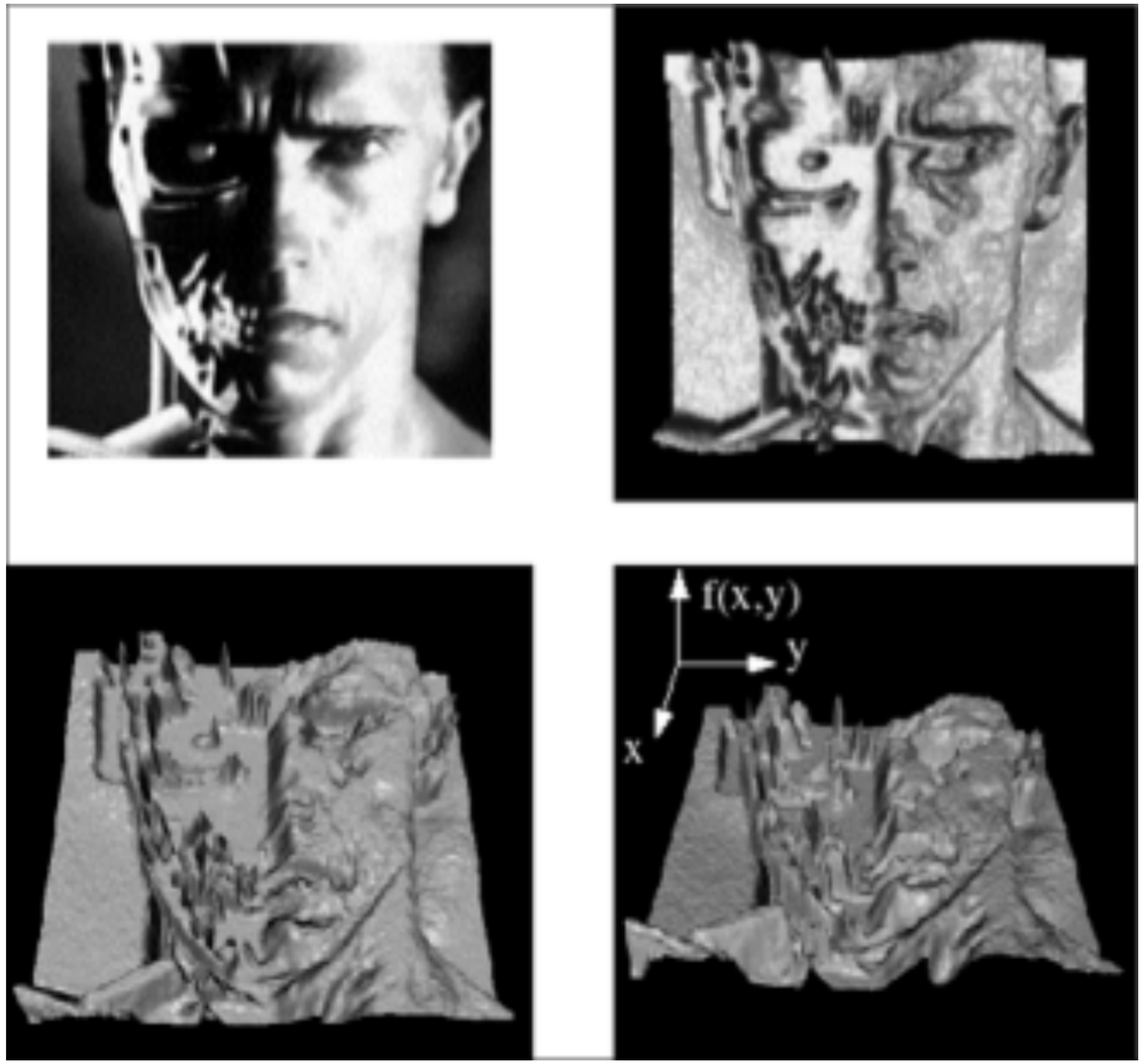
  - For every 2D point it tells us what the (light) intensity is.
  - Realistically, the size of the sensor is limited and so is the range of brightnesses it can capture:

$$I(x, y), \quad I : [a, b] \times [c, d] \rightarrow [0, m]$$

- Color images can be thought of as vector-valued functions:

$$\mathbf{I}(x, y) = \begin{pmatrix} I_R(x, y) \\ I_G(x, y) \\ I_B(x, y) \end{pmatrix}, \quad \mathbf{I} : [a, b] \times [c, d] \rightarrow [0, m]^3$$

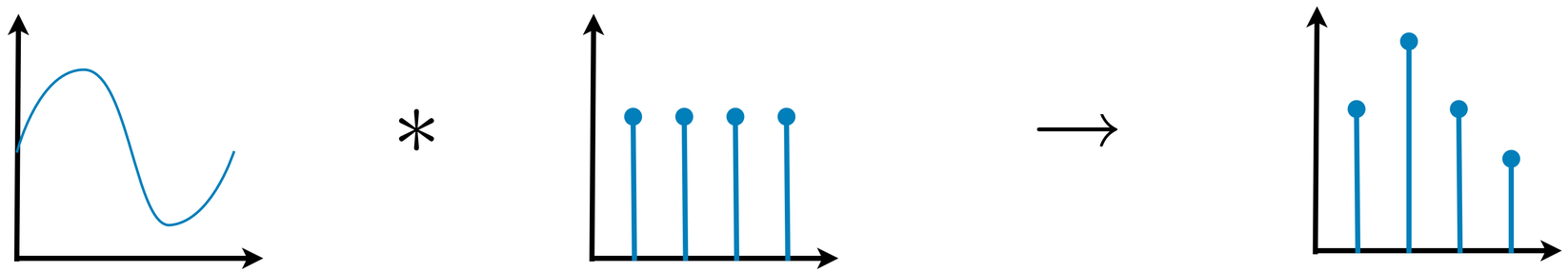
Images as functions



[from Steve Seitz]

What is a digital image?

- We usually do not work with spatially continuous functions, since our cameras do not sense in this way.
- Instead use (spatially) discrete images
  - Sample the 2D domain on a regular grid
  - 1D analog:



continuous signal sampling operator sampled image

y =	x = 58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

- Intensity/color values usually also discrete
- Quantize the values per channel (e.g. 8 bit)

Today: Basics of Digital Image Processing

- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - ‘Recognition using line drawings’
  - Image derivatives (1st and 2nd order)

Linear Operations / Systems

- Basic properties:
  - homogeneity $$T[a \cdot X] = a \cdot T[X]$$
  - additivity $$T[X + Y] = T[X] + T[Y]$$
  - superposition $$T[a \cdot X + b \cdot Y] = a \cdot T[X] + b \cdot T[Y]$$
- Examples:
  - matrix-vector operations
  - convolutions

Convolution

- Replace each pixel by a linear combination of its neighbors (and itself).
- 2D convolution (discrete):

$$g = f * h \qquad g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l),$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

image

$f(i,j)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

filter (kernel)

$h(i,j)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

filtered image

$g(i,j)$

smaller output?

Convolution / Correlation

- Convolution is
  - linear $h * (f_0 + f_1) = h * f_0 + h * f_1$
  - associative $(f * g) * h = f * (g * h)$
  - commutative $f * h = h * f$
  - shift-invariant $g(i,j) = f(i+k, j+l) \Leftrightarrow (h * g)(i,j) = (h * f)(i+k, j+l)$
  - can be represented as matrix-vector product $\mathbf{g} = \mathbf{H}\mathbf{f}$
  - Continuous versions exist, skip them here...
- Correlation:
  - Closely related, but do not mirror filter

$$g = f \otimes h \qquad g(i,j) = \sum_{k,l} f(i+k, j+l) h(k,l)$$

Filtering to Reduce Noise

- "Noise" is what we are not interested in
  - low-level noise: light fluctuations, sensor noise, quantization effects, finite precision, …
  - complex noise (not today): shadows, extraneous objects

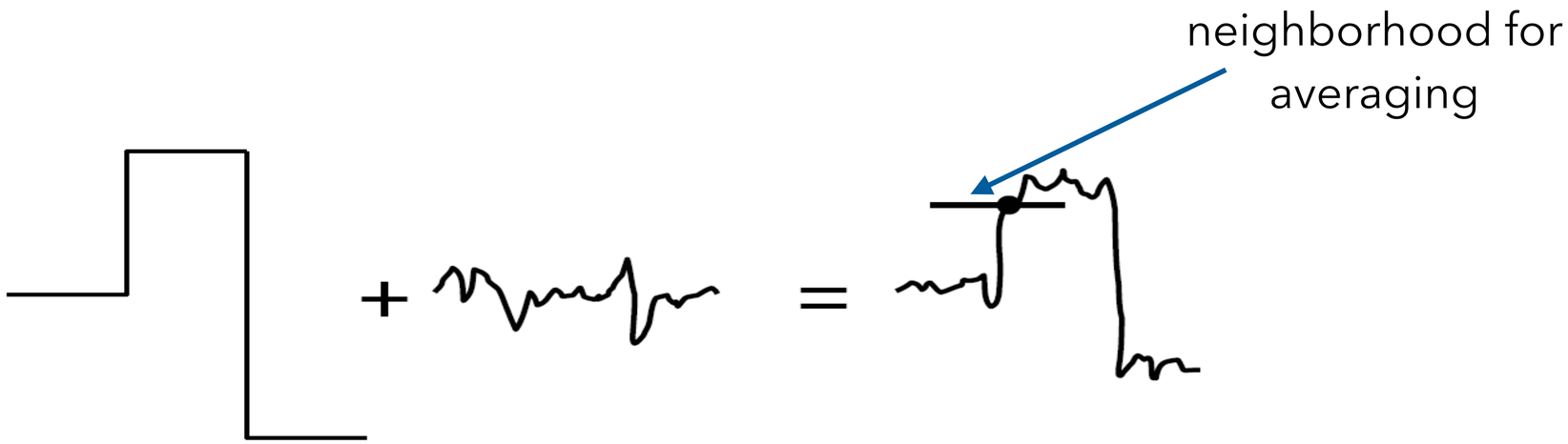
- Assumption:
  - The pixel's neighborhood contains information about its intensity

$$\begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 20 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 3 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

Model: Additive Noise

Signal + Noise = Image

$$S + N = I$$



Assumption: Nearby points tell more about the signal than distant ones

Model: Additive Noise

- Image I = Signal S + Noise N
  - I.e. noise does not depend on the signal
- We consider:
  - I_k : intensity of k-th pixel
  - $I_k = s_k + n_k$ with $E[n_k] = 0$
  - s_k deterministic
  - n_k, n_l independent for $k \neq l$
  - n_k, n_l i.i.d. (independent, identically distributed)

- Therefore:
  - Intuition: Averaging noise reduces its effect

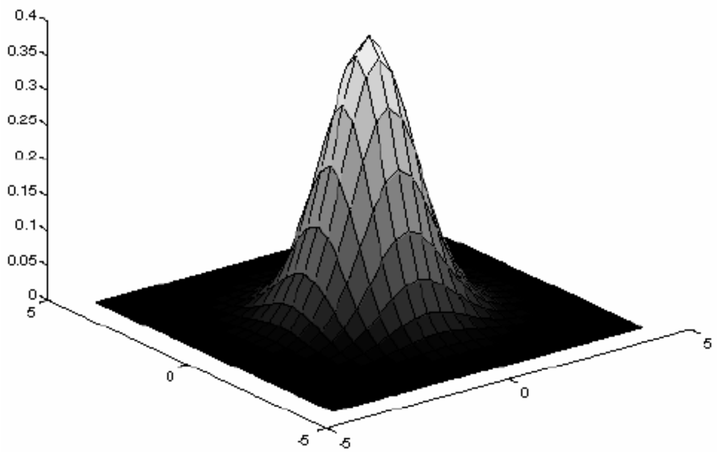
Average Filter

- Average Filter
  - replaces each pixel with an average of its neighborhood
  - Mask with positive entries that sum to 1
- If all weights are equal, it is called a box filter

$$\frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

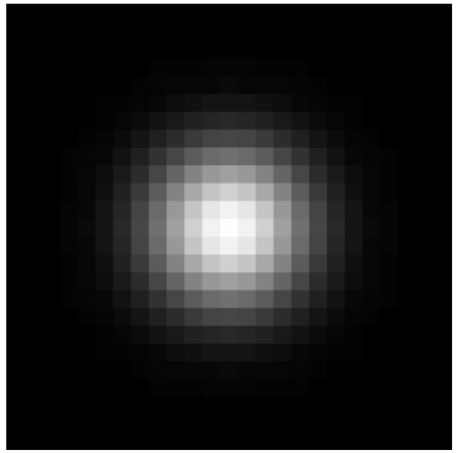
Gaussian Averaging

- Isotropic Gaussian
  (rotationally symmetric)
- Weighs nearby pixels more than distant ones



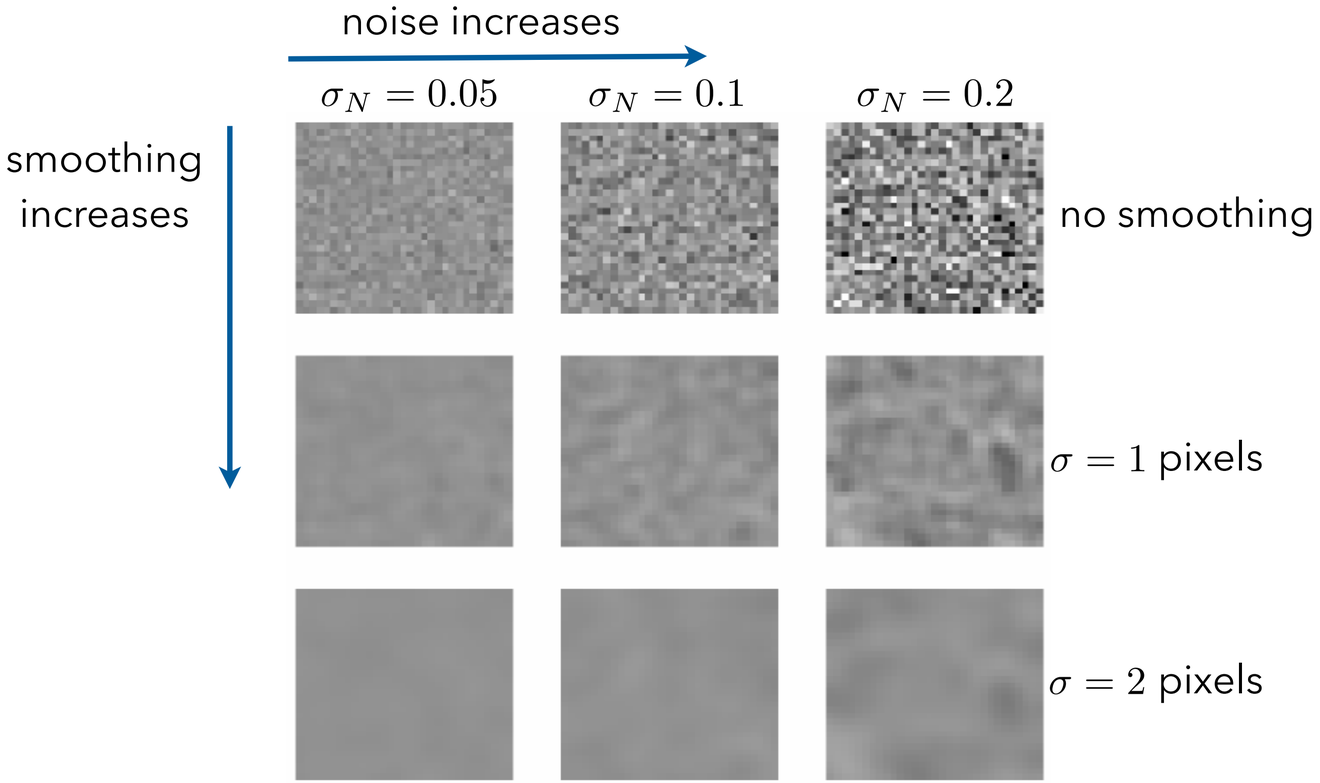
- Smoothing kernel proportional to

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



Smoothing with a Gaussian

- Effects of smoothing:
  - Each column shows realizations of an image of Gaussian noise
  - Each row shows smoothing with Gaussians of different width



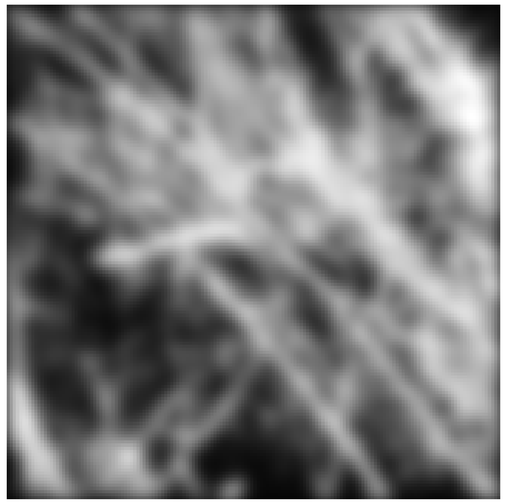
Gaussians are everywhere!

Smoothing with a Gaussian

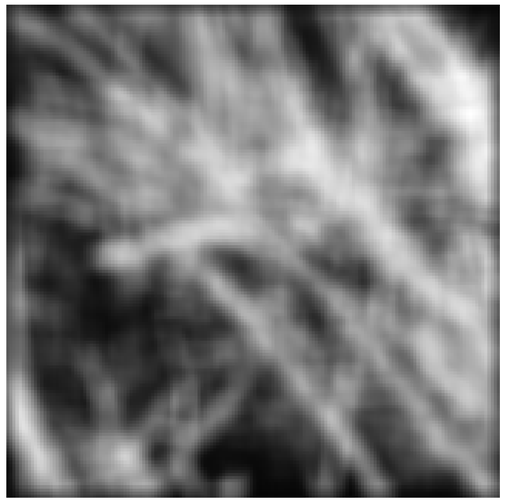
- Example:

original image

Gaussian-filtered image



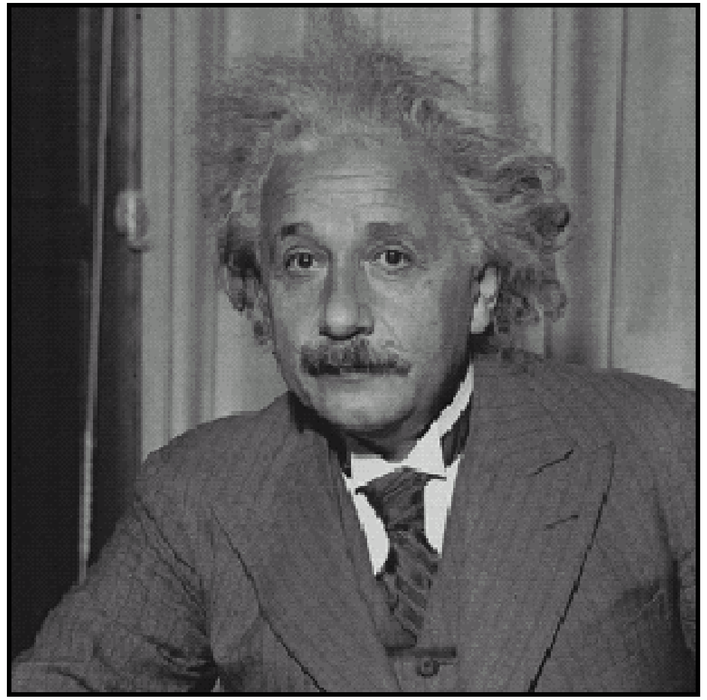
box-filtered image



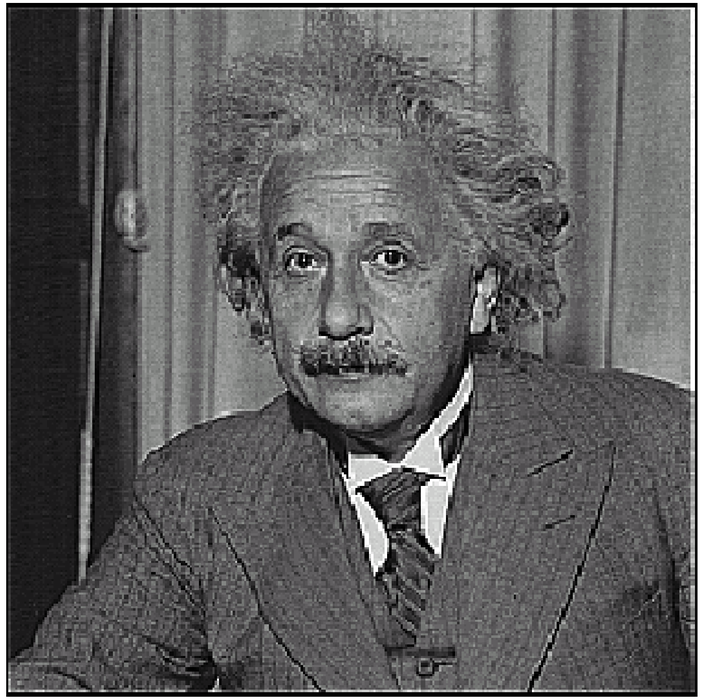
Only smoothing?

- We can also sharpen an image by amplifying what smoothing removes:

$$g_{\text{sharp}} = f + \gamma(f - h_{\text{blur}} * f)$$



original



sharpened

[Bill Freeman]

Efficient Implementation

- Both, the box filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - then convolve each column with a 1D filter

$$(f_x * f_y) * I = f_x * (f_y * I)$$

- Remember:
  - convolution is linear - associative and commutative

convolution!

- *Example:* Separable box filter

$$\underbrace{\frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{f_x * f_y} = \underbrace{\frac{1}{3} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}}_{f_x} * \underbrace{\frac{1}{3} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{f_y}$$

Example: Separable Gaussian

- Gaussian in x-direction

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- Gaussian in y-direction

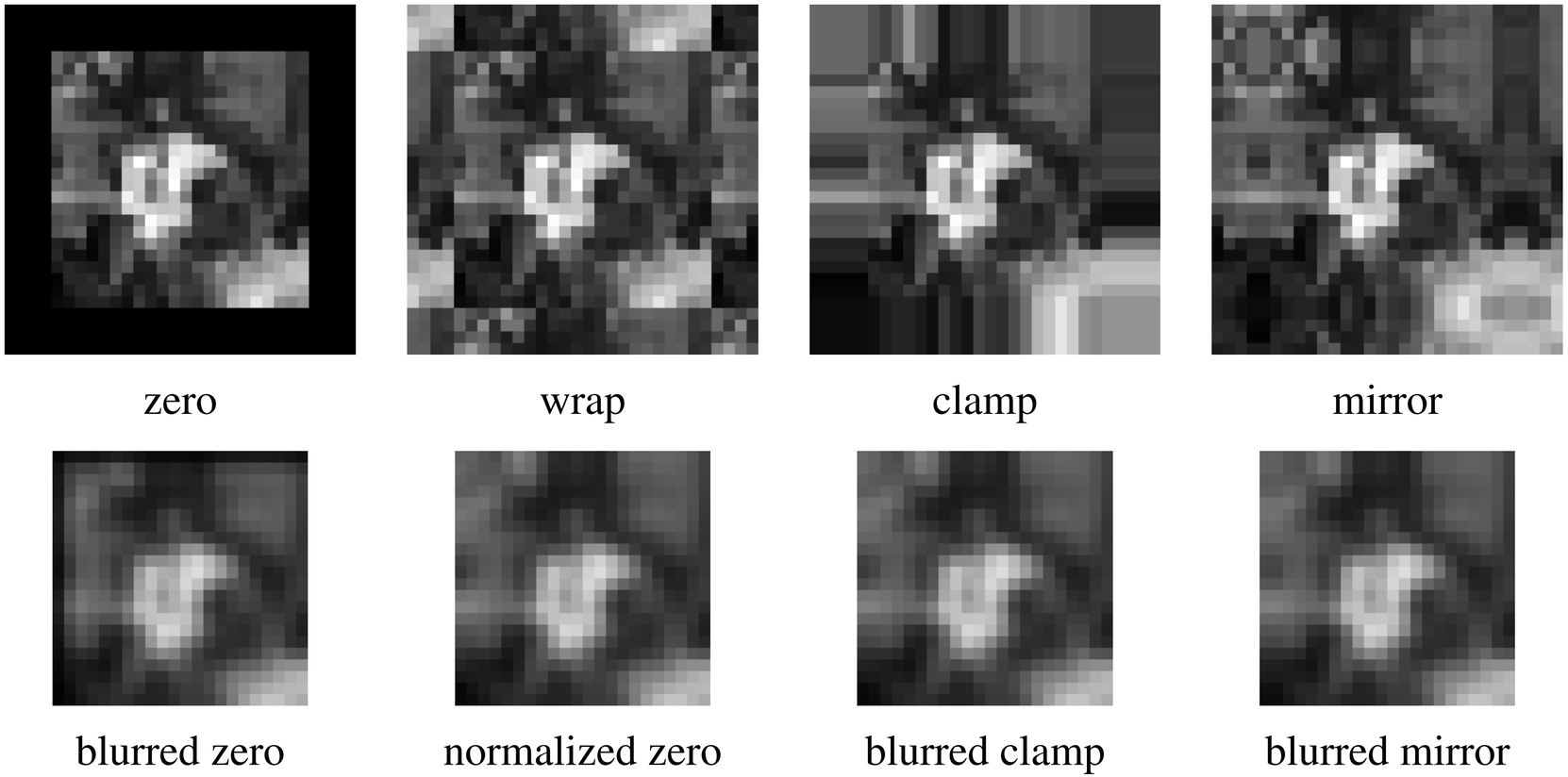
$$G_\sigma(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

- Gaussian in both directions (product)

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Boundary issues

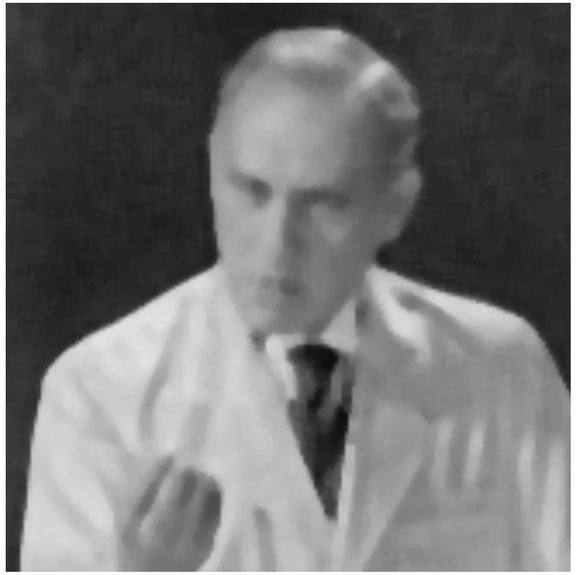
- Need to be careful at the boundaries:
  - Boundary handling strategies



Non-linear filtering

- There is a large variety of non-linear filters for noise removal.
- Simplest one: **Median filter**
  - Replace each pixel with the median in a neighborhood around it

Original



7x7 median filter

Special case: Binary Images

- Morphological operations:
  - Perform convolution with a "structuring element" s
  - This is a binary mask (often circle or square)
  - Then perform thresholding to recover a binary image:

Note: As defined the morphological operations apply only to binary images!

Generalization does exist…

$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases}$$

original | dilation $\theta(f * s, 1)$ | erosion $\theta(f * s, S)$

# of pixels in s

Morphological filters – Side note

- Generalization to grayscale images:
  - Perform min/max-convolution with a “structuring element” s
  - We get our previous behavior with flat structuring element $s(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \in B \\ -\infty & \text{otherwise} \end{cases}$

original

dilation

$$\max_{\mathbf{y}}[f(\mathbf{x}) + s(\mathbf{x} - \mathbf{y})]$$

erosion

$$\min_{\mathbf{y}}[f(\mathbf{x}) - s(\mathbf{y} - \mathbf{x})]$$