UNIVERSITY OF CAMBRIDGE

Advanced Graphics and Image Processing

Computer Science Tripos Part 2
MPhil in Advanced Computer Science
Michaelmas Term 2023/2024

Department of
Computer Science
and Technology
The Computer Laboratory

William Gates Building
15 JJ Thomson Avenue
Cambridge
CB3 0FD

www.cst.cam.ac.uk

This handout includes copies of the slides that will be used in lectures and more detailed notes on the selected topics. These notes do not constitute a complete transcript of all the lectures and they are not a substitute for text books. They are intended to give a reasonable synopsis of the subjects discussed, but they give neither complete descriptions nor all the background material.

Material is copyright © Rafał Mantiuk, 2015-2023, except where otherwise noted.

All other copyright material is made available under the University's licence. All rights reserved.

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics & Image Processing

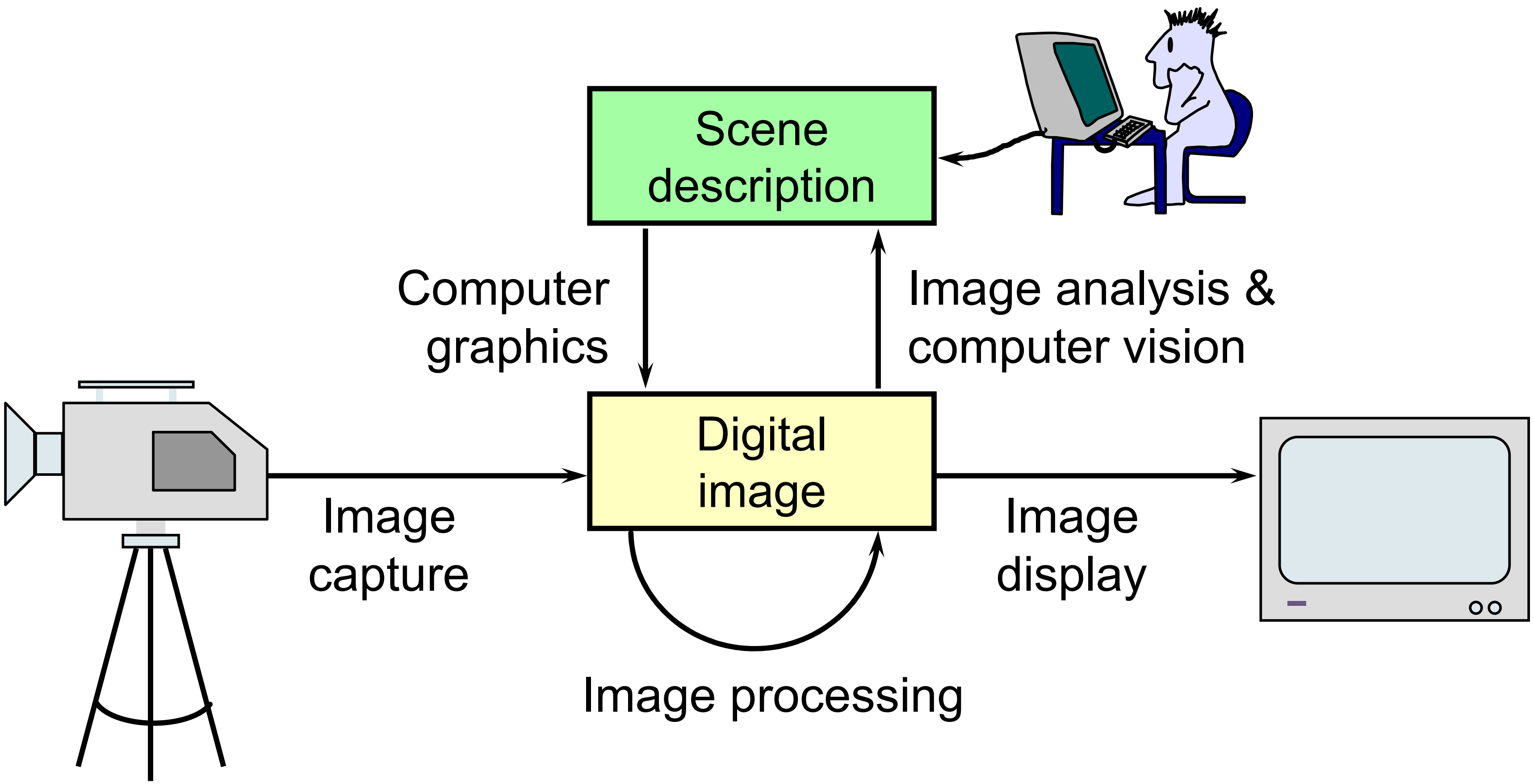
Introduction to Image Processing

Part 1/2 – Images, pixels and sampling

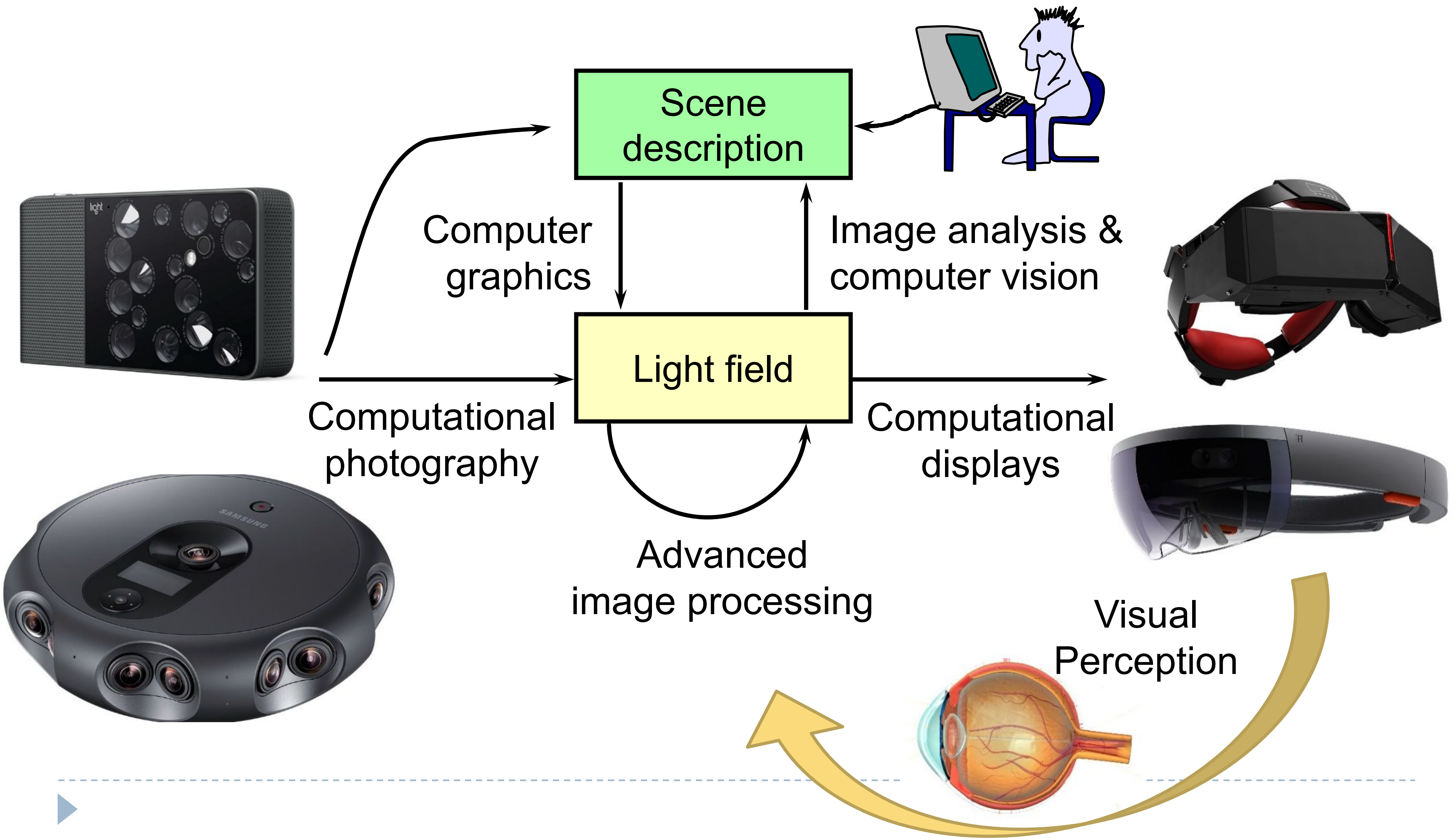
Rafał Mantiuk

Computer Laboratory, University of Cambridge

What are Computer Graphics & Image Processing?



Where are graphics and image processing heading?



What is a (computer) image?

- A digital photograph? ("JPEG")
- A snapshot of real-world lighting?

Image

From computing perspective (discrete)

2D array of pixels

- To represent images in memory
- To create image processing software

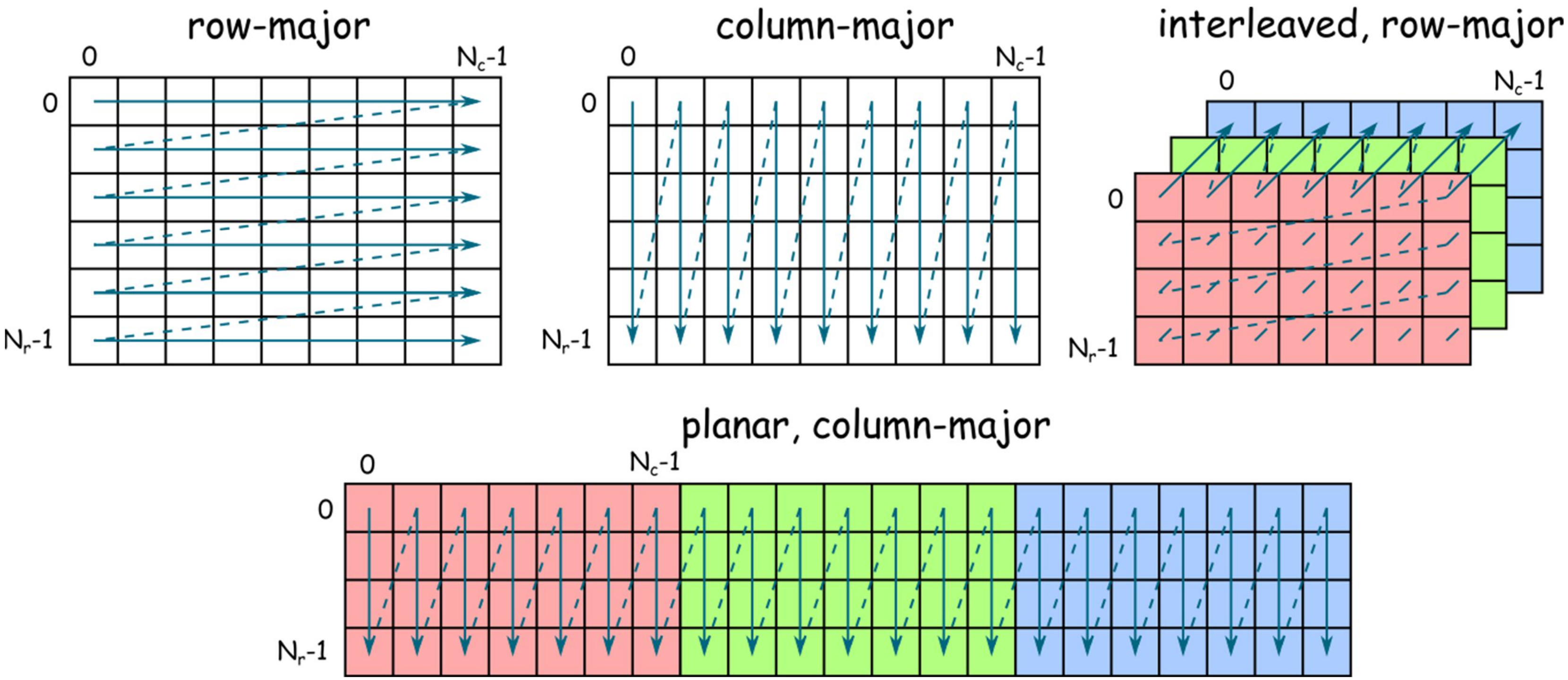
From mathematical perspective (continuous)

2D function

- To express image processing as a mathematical problem
- To develop (and understand) algorithms

Image

- 2D array of pixels
- In most cases, each pixel takes 3 bytes: one for each red, green and blue
- But how to store a 2D array in memory?



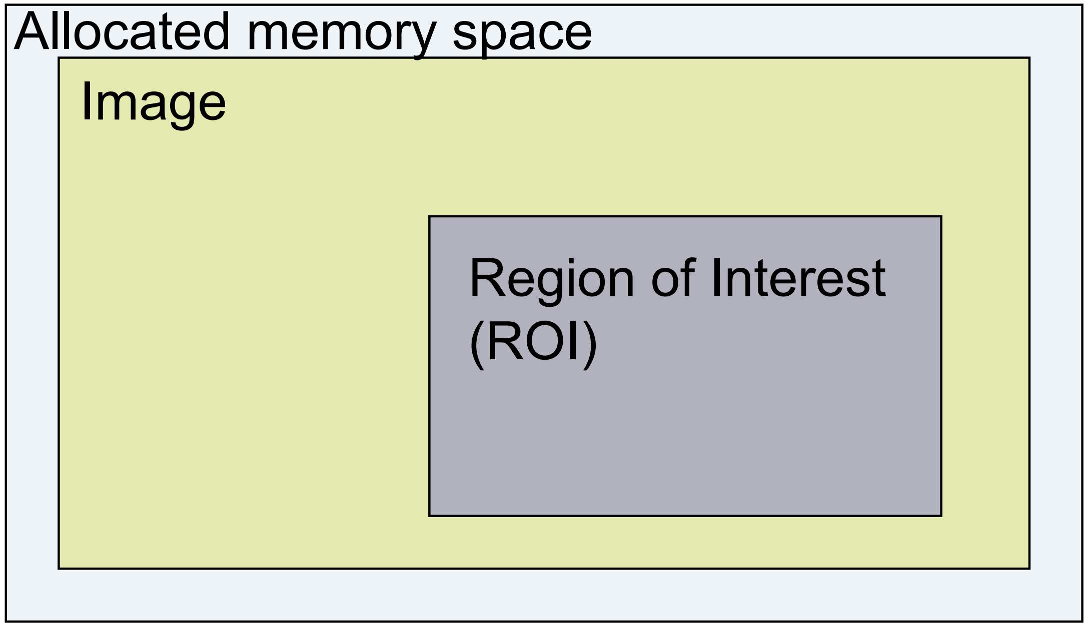
Stride

- Calculating the pixel component index in memory
  - For row-major order (grayscale)
$$i(x, y) = x + y \cdot n_c$$
  - For column-major order (grayscale)
$$i(x, y) = x \cdot n_r + y$$
  - For interleaved row-major (colour)
$$i(x, y, c) = x \cdot 3 + y \cdot 3 \cdot n_c + c$$
  - General case
$$i(x, y, c) = x \cdot s_x + y \cdot s_y + c \cdot s_c$$

where s_x, s_y and s_c are the strides for the x, y and colour dimensions

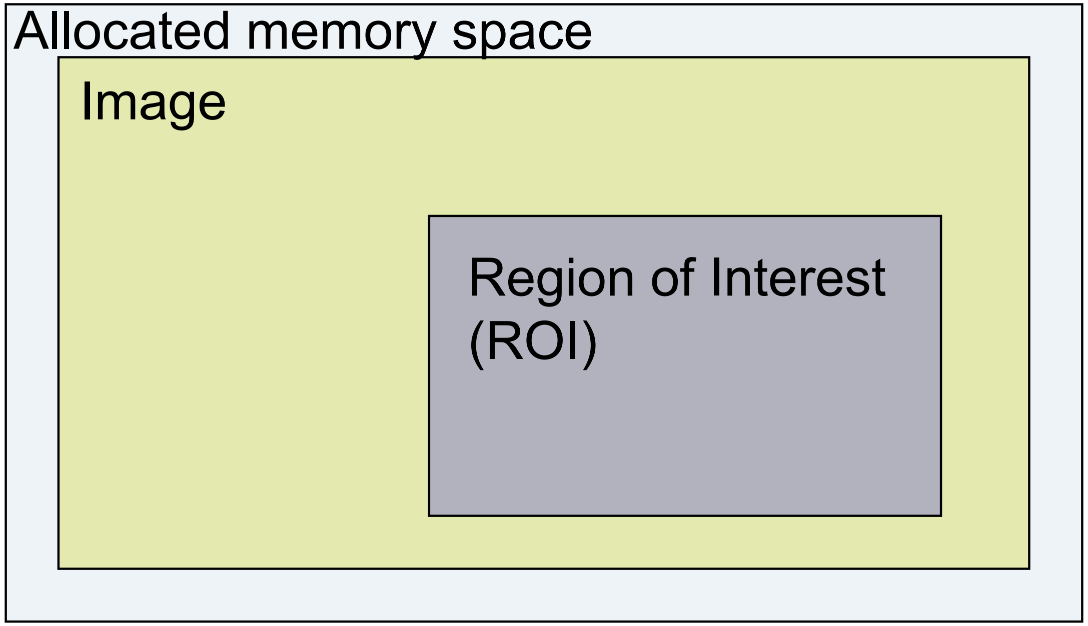
Padded images and stride

- Sometimes it is desirable to “pad” image with extra pixels
  - for example when using operators that need to access pixels outside the image border
- Or to define a region of interest (ROI)



- How to address pixels for such an image and the ROI?

Padded images and stride



$$i(x, y, c) = i_{first} + x \cdot s_x + y \cdot s_y + c \cdot s_c$$

- For row-major, interleaved
  - i_{first} =?
  - s_x =?
  - s_y =?
  - s_c =?

Pixel (PIcture ELement)

- Each pixel (usually) consist of three values describing the color

(red, green, blue)

- For example
  - (255, 255, 255) for white
  - (0, 0, 0) for black
  - (255, 0, 0) for red
- Why are the values in the 0-255 range?
- Why red, green and blue? (and not cyan, magenta, yellow)
- How many bytes are needed to store 5MPixel colour image? (uncompressed)

Pixel formats, bits per pixel, bit-depth

- Grayscale – single **color channel**, 8 bits (1 byte)
- Highcolor – 2^{16}=65,536 colors (2 bytes)

Sample Length:	5					6						5				
Channel Membership:	Red					Green						Blue				
Bit Number:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGBAX	R. G. B. A. X															
Sample Length Notation:	**5.6.5.0.0**															

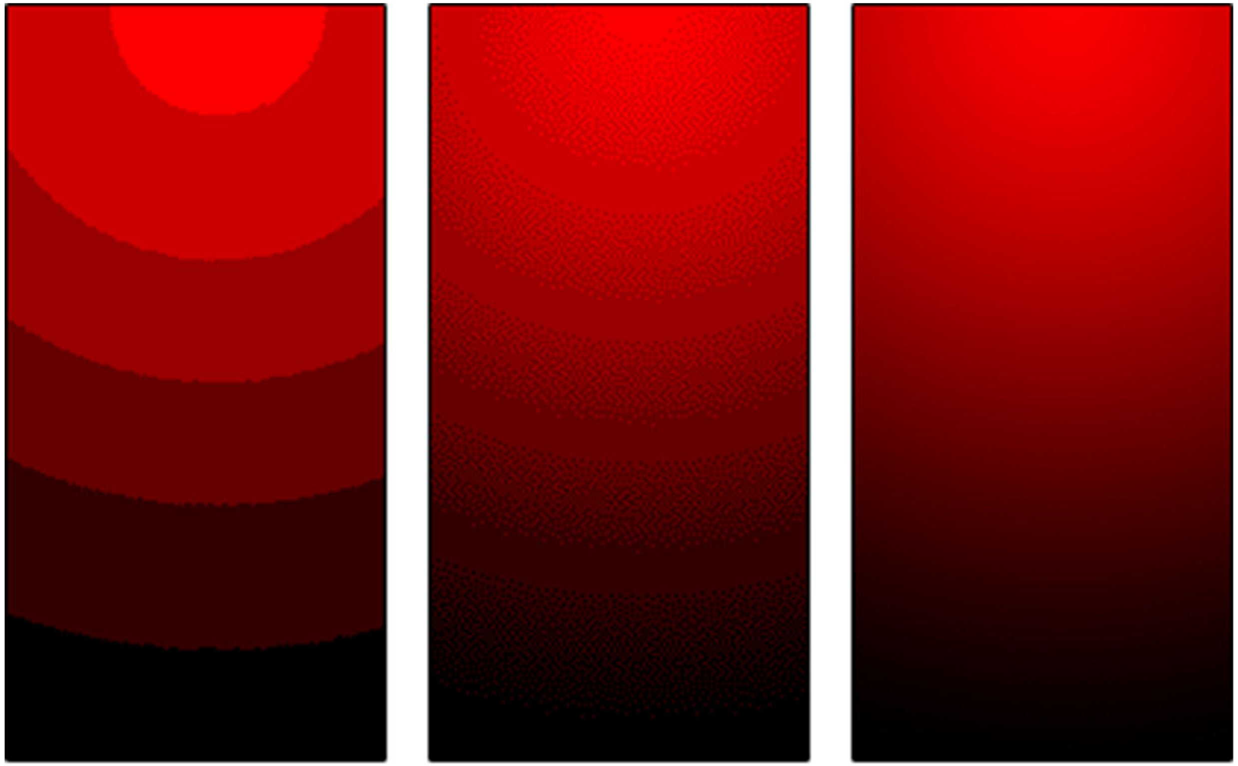
- Truecolor – 2^{24} = 16,8 million colors (3 bytes)
- Deepcolor – even more colors (>= 4 bytes)

Sample Length:	2		10										10										10									
Channel Membership:	None		Red										Green										Blue									
Bit Number:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGBAX	R . G . B . A . X																															
Sample Length Notation:	**10.10.10.0.2**																															

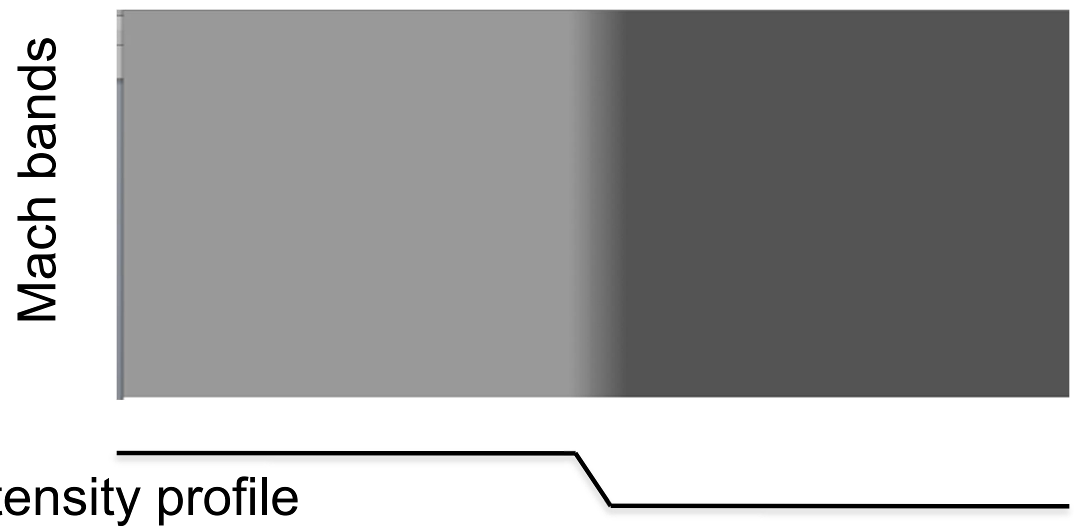
- But why?

Colour banding

- If there are not enough bits to represent colour
- Looks worse because of the **Mach band** illusion
- Dithering (added noise) can reduce banding
  - Printers
  - Many LCD displays do it too



8-bit gradient | 8-bit gradient, dithered | 24-bit gradient



What is a (computer) image?

- A digital photograph? ("JPEG")
- A snapshot of real-world lighting?

Image

From computing perspective (discrete): 2D array of pixels

From mathematical perspective (continuous): 2D function

- To represent images in memory
- To create image processing software

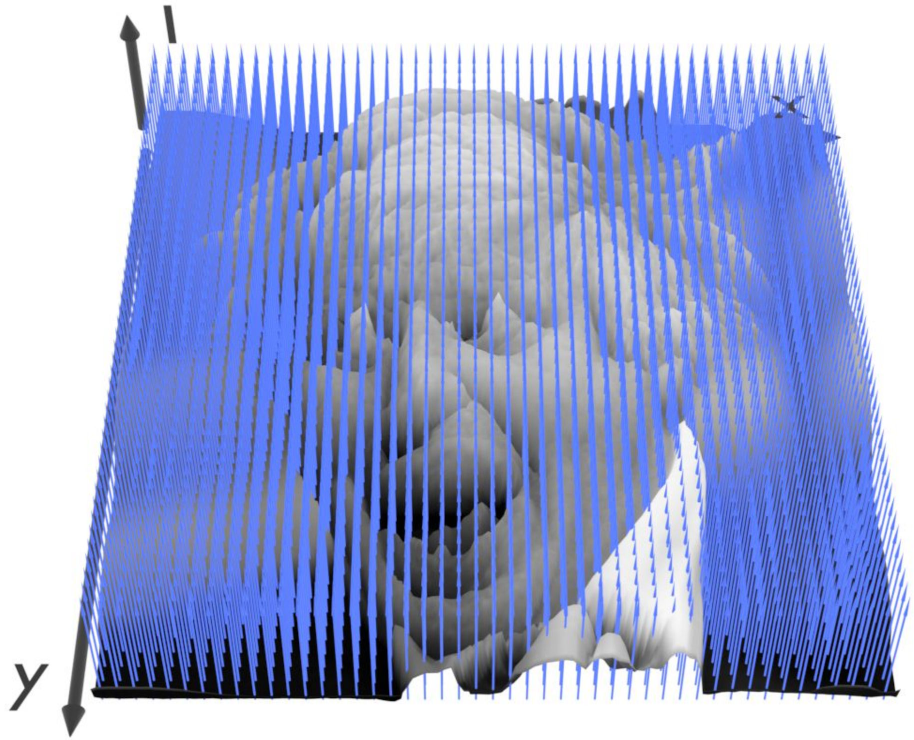
- To express image processing as a mathematical problem
- To develop (and understand) algorithms

Image – 2D function

- Image can be seen as a function I(x,y), that gives intensity value for any given coordinate (x,y)

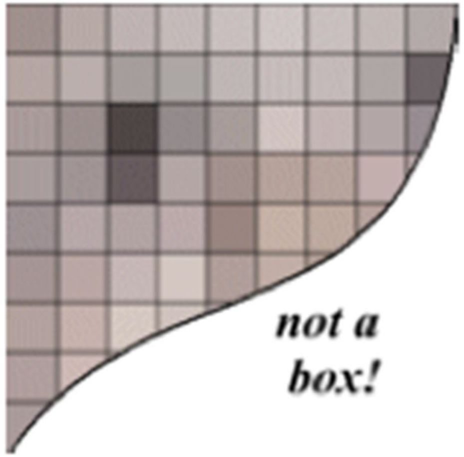
Sampling an image

- The image can be sampled on a rectangular sampling grid to yield a set of samples. These samples are pixels.

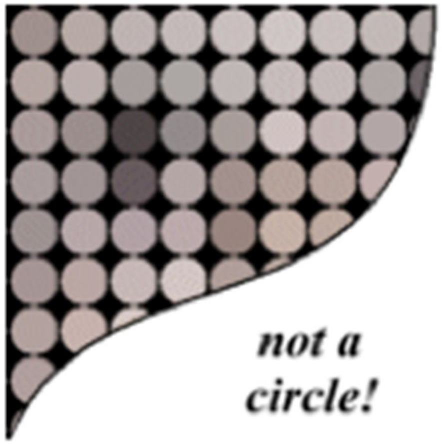


What is a pixel?

- A pixel is not
  - a box
  - a disk
  - a teeny light



- A pixel is a point
  - it has no dimension
  - it occupies no area
  - it cannot be seen
  - it has coordinates

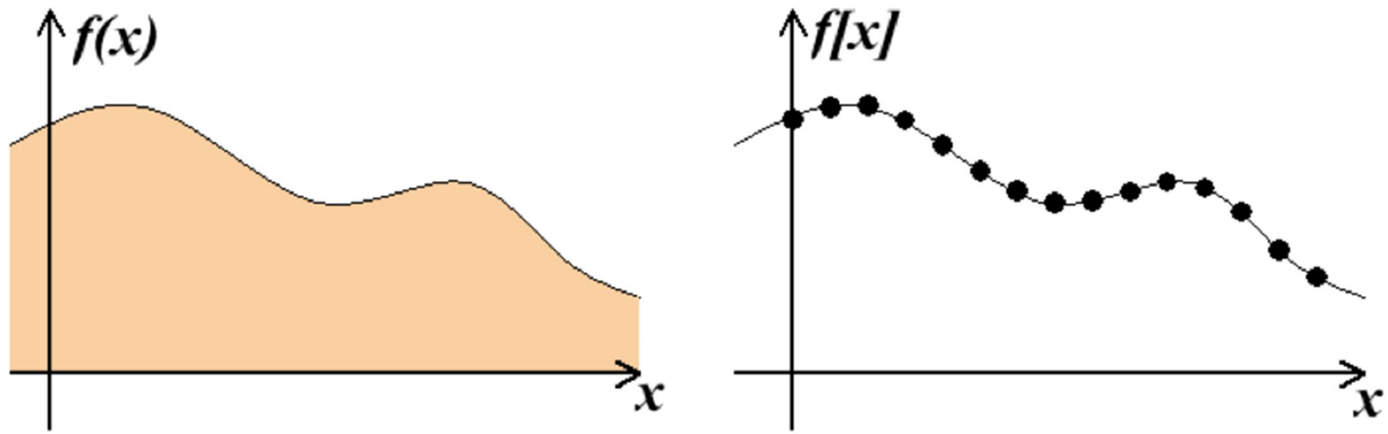


- A pixel is a **sample**

From: http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture05/lecture05.pdf

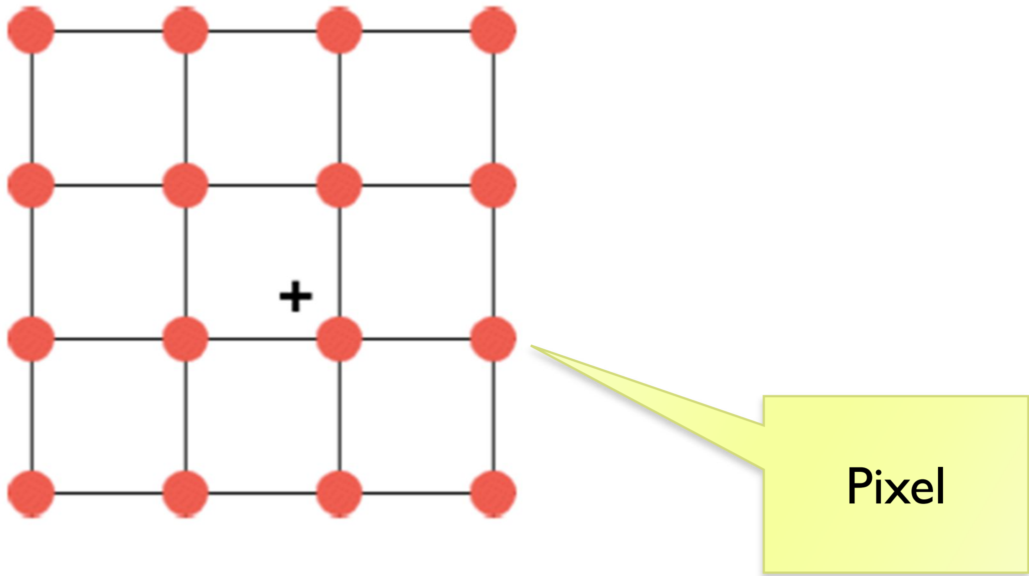
Sampling and quantization

- The physical world is described in terms of continuous quantities
- But computers work only with discrete numbers
- Sampling – process of mapping continuous function to a discrete one
- Quantization – process of mapping continuous variable to a discrete one



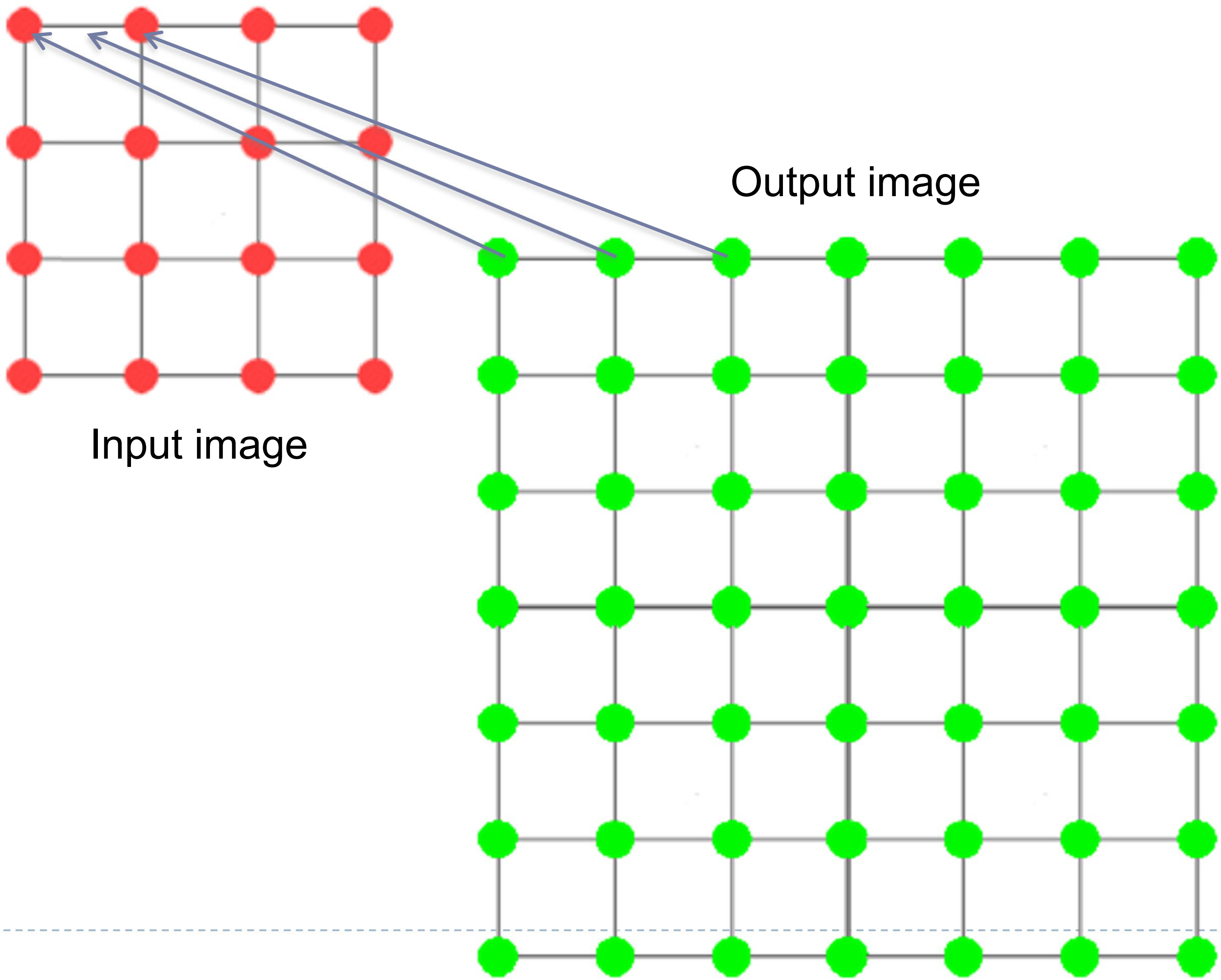
Resampling

- Some image processing operations require to know the colors that are in-between the original pixels

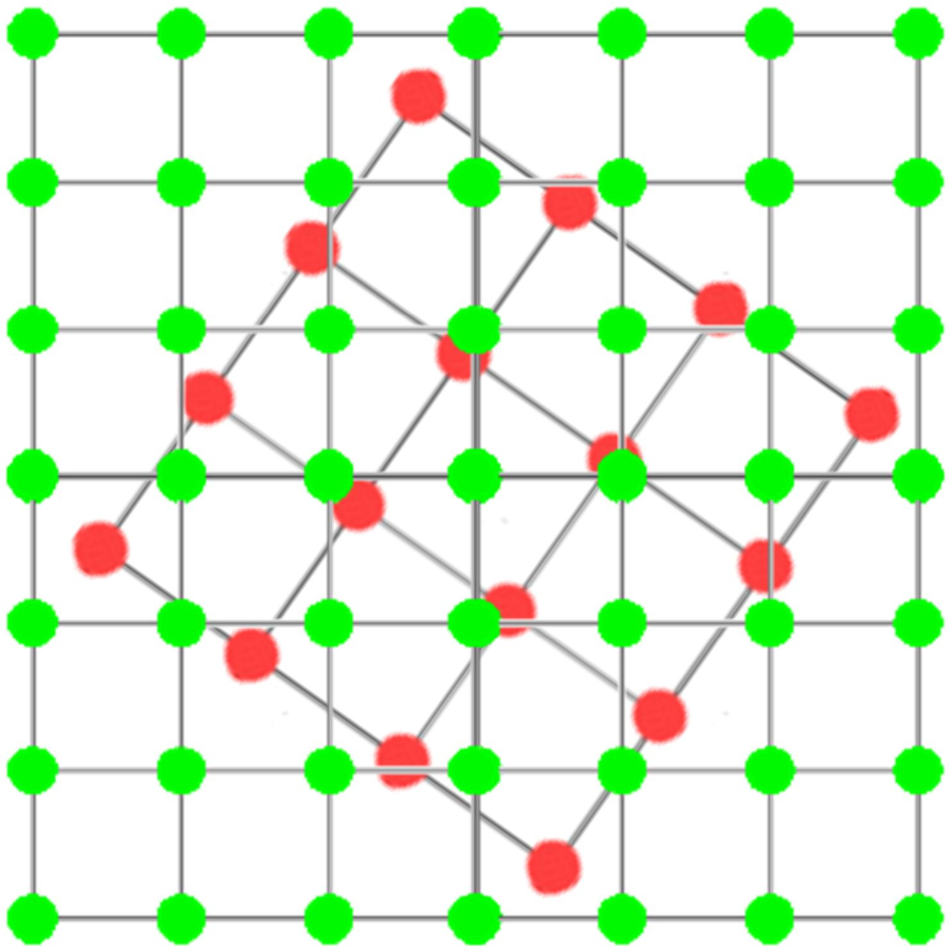


- What are those operations?
- How to find these *resampled* pixel values?

Example of resampling: magnification

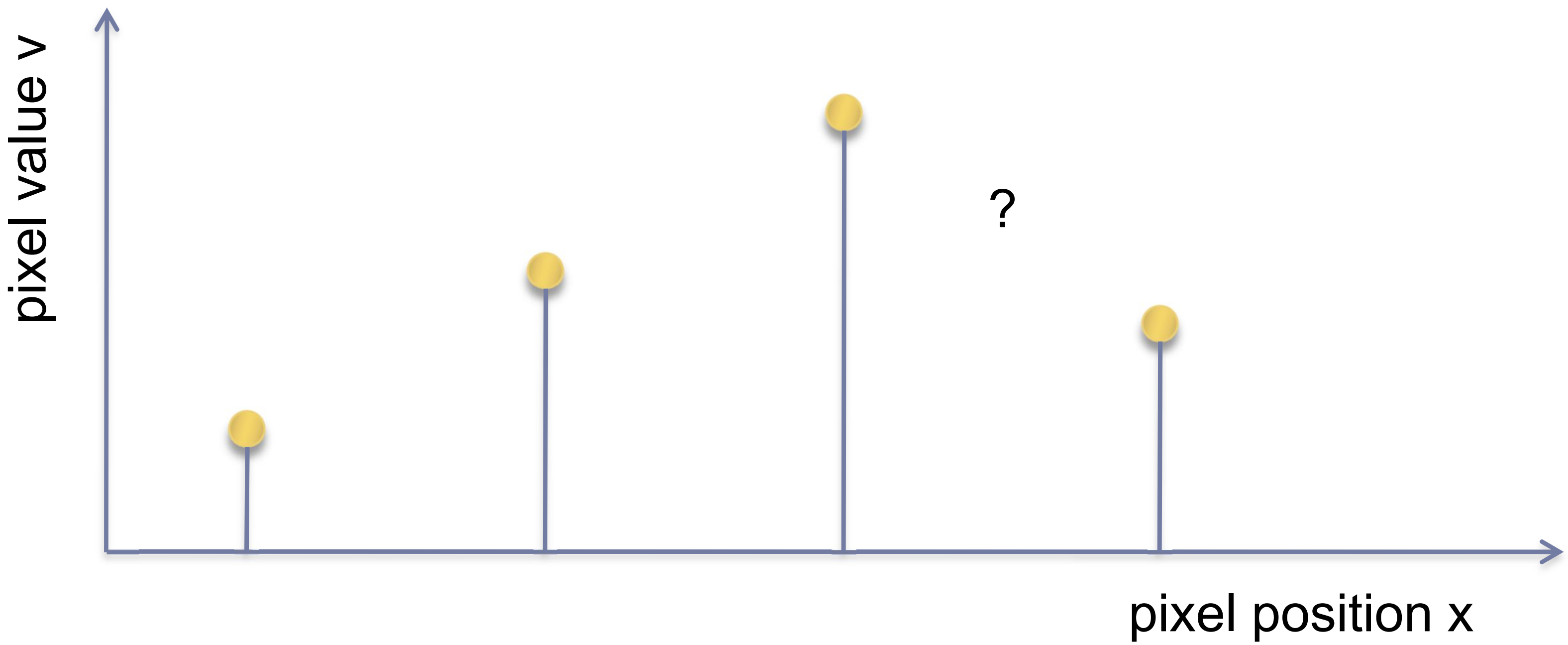


Example of resampling: scaling and rotation



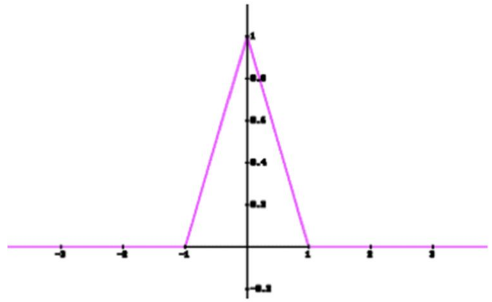
How to resample?

- In 1D: how to find the most likely resampled pixel value knowing its two neighbors?

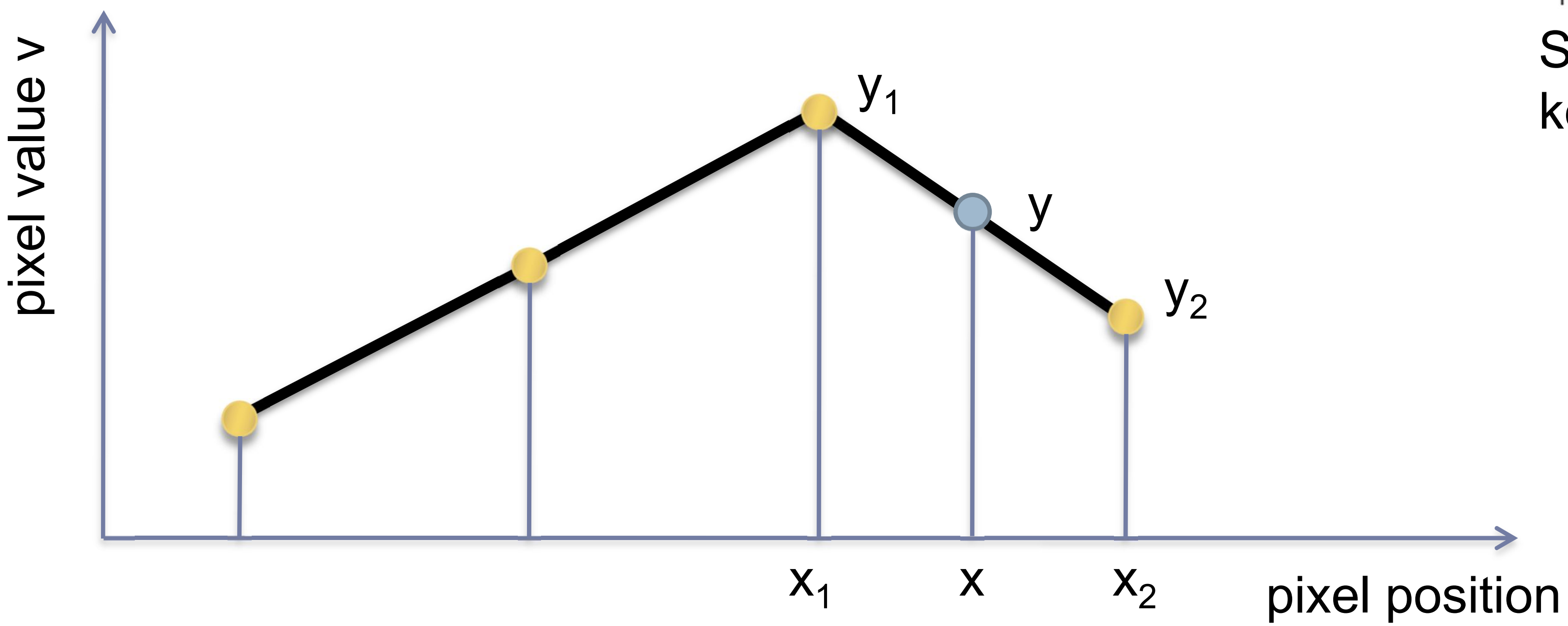


(Bi)Linear interpolation (resampling)

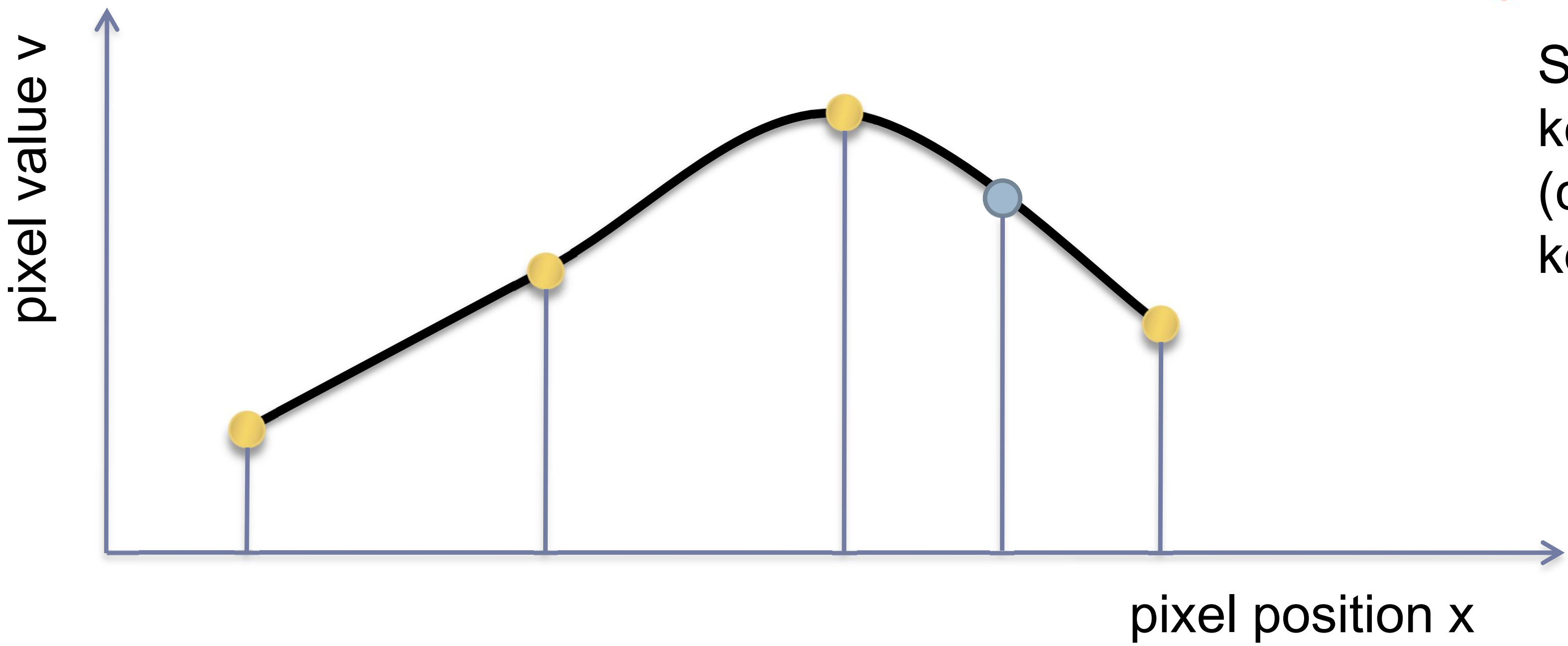
- Linear – 1D
- Bilinear – 2D

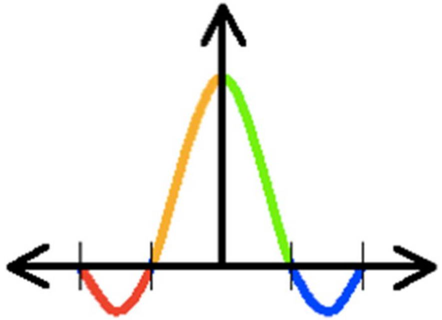


Sampling kernel



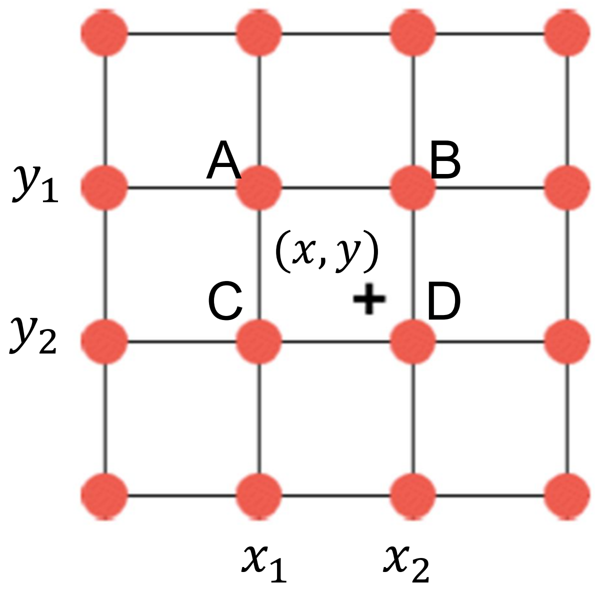
(Bi)cubic interpolation (resampling)





Sampling kernel (convolution kernel)

Bi-linear interpolation



Given the pixel values:

$$I(x_1, y_1) = A$$

$$I(x_2, y_1) = B$$

$$I(x_1, y_2) = C$$

$$I(x_2, y_2) = D$$

Calculate the value of a pixel $I(x, y) = ?$ using bi-linear interpolation.

Hint: Interpolate first between A and B, and between C and D, then interpolate between these two computed values.

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics & Image Processing

Introduction to Image Processing

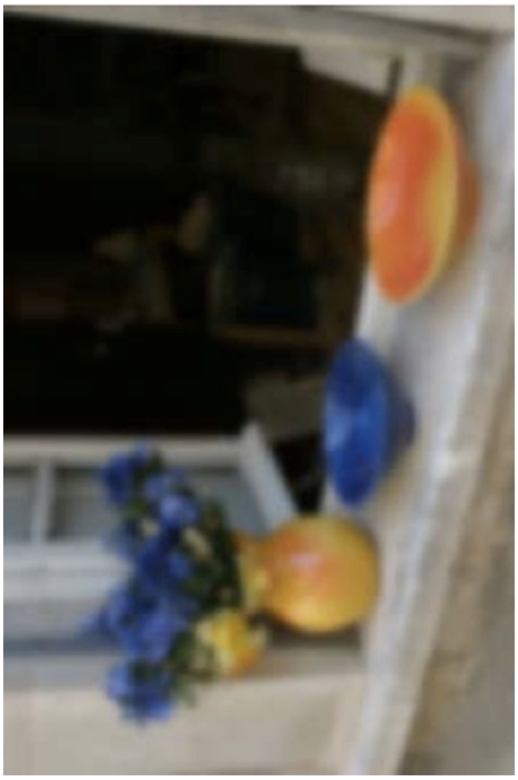
Part 2/2 – Point ops, filters and pyramids

Rafał Mantiuk

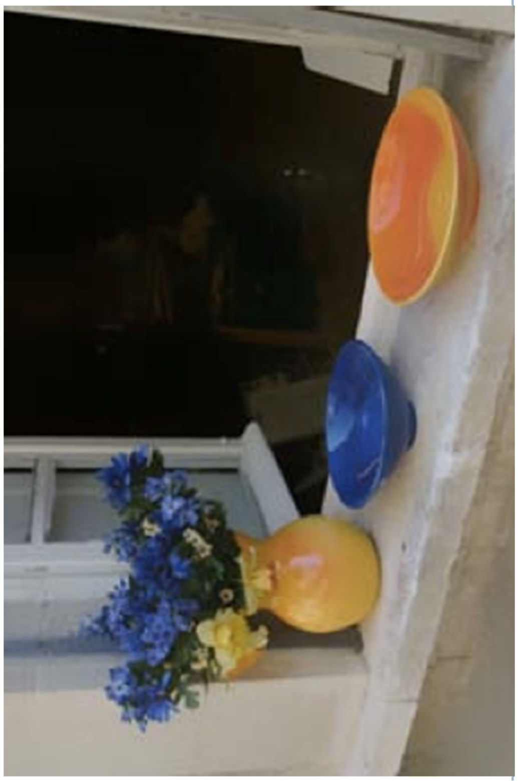
Computer Laboratory, University of Cambridge

Point operators and filters

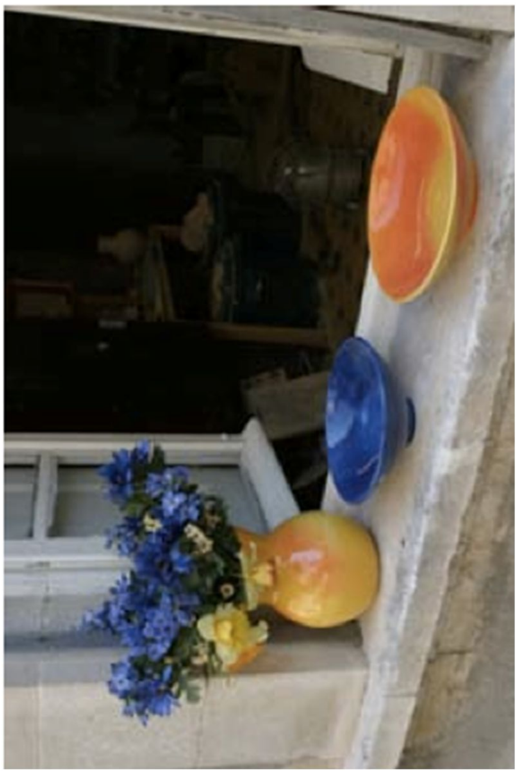
Blurred



Edge-preserving filter



Original



Sharpenned

Point operators

- Modify each pixel independent from one another
- The simplest case: multiplication and addition

Pixel position
x=(x,y)

Input pixel
value

$$g(\boldsymbol{x}) = af(\boldsymbol{x}) + b$$

Resulting pixel
value

Gain

Bias

Pixel precision for image processing

- Given an RGB image, 8-bit per color channel (uchar)
  - What happens if the value of 10 is subtracted from the pixel value of 5 ?
  - 250 + 10 = ?
  - How to multiply pixel values by 1.5 ?
    - a) Using floating point numbers
    - b) While avoiding floating point numbers

Image blending

- Cross-dissolve between two images

Pixel from image 1

Pixel from image 2

$$g(\boldsymbol{x}) = (1-\alpha)f_0(\boldsymbol{x}) + \alpha f_1(\boldsymbol{x})$$

Resulting pixel value

Blending parameter

- where α is between 0 and 1

Image matting and compositing

- Matting – the process of extracting an object from the original image
- Compositing – the process of inserting the object into a different image
- It is convenient to represent the extracted object as an RGBA image

Transparency, alpha channel

- RGBA – red, green, blue, alpha
  - alpha = 0 – transparent pixel
  - alpha = 1 – opaque pixel
- Compositing
  - Final pixel value:

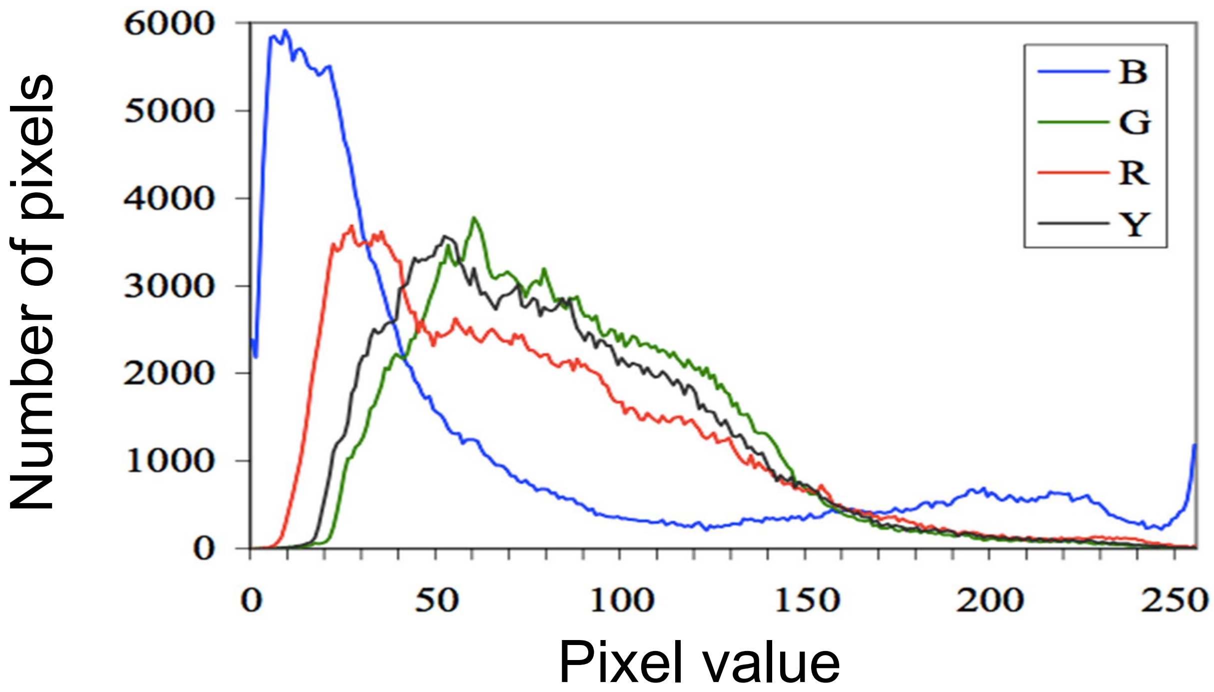
$$P = \alpha C_{pixel} + (1 - \alpha) C_{background}$$

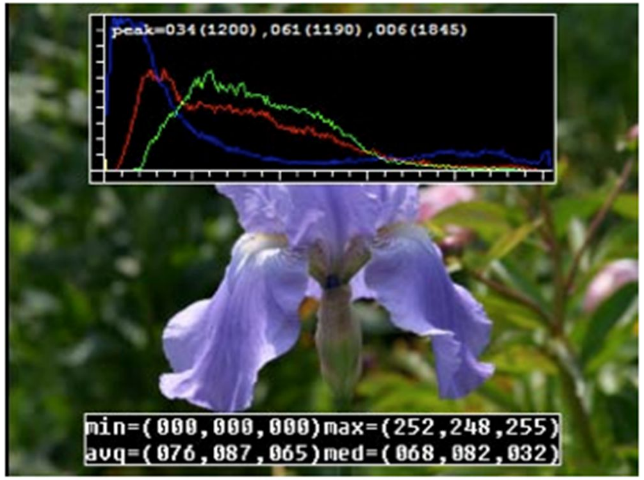
  - Multiple layers:

$$P_0 = C_{background}$$

$$P_i = \alpha_i C_i + (1 - \alpha_i) P_{i-1} \quad i = 1..N$$

Image histogram

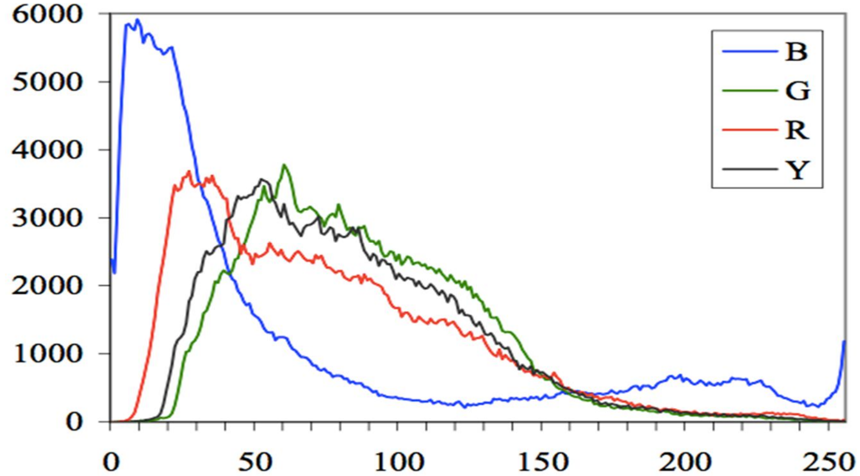




- histogram / total pixels = probability mass function
  - what probability does it represent?

Histogram equalization

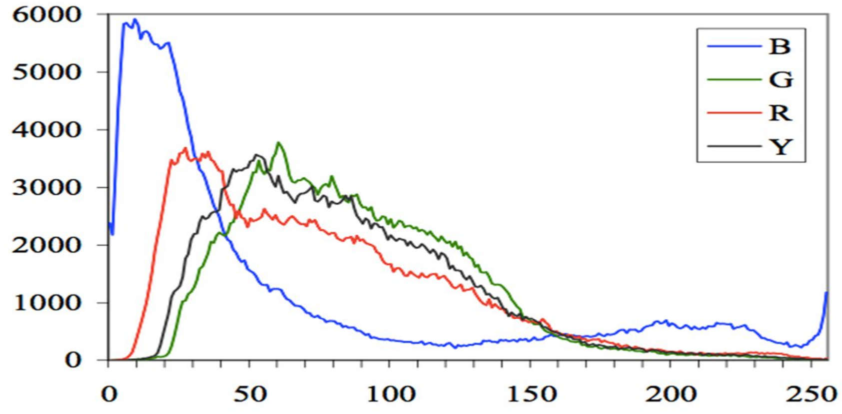
- Pixels are non-uniformly distributed across the range of values



- Would the image look better if we uniformly distribute pixel values (make the histogram more uniform)?
- How can this be done?

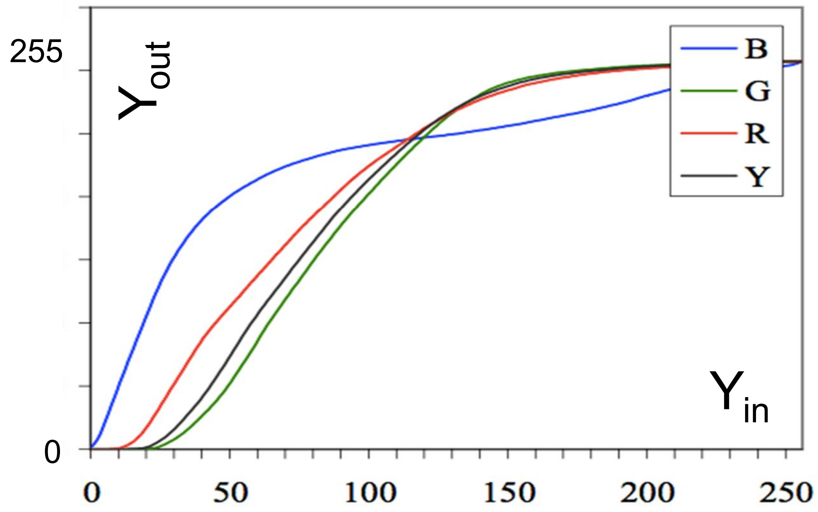
Histogram equalization

- Step 1: Compute image histogram



- Step 2: Compute a normalized cumulative histogram

$$c(I) = \frac{1}{N} \sum_{i=0}^{I} h(i)$$



- Step 3: Use the cumulative histogram to map pixels to the new values (as a look-up table)

$$Y_{out} = c(Y_{in})$$

Linear filtering

- Output pixel value is a weighted sum of neighboring pixels

Input pixel value

Kernel (filter)

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

Resulting pixel value

Sum over neighboring pixels, e.g. k=-1,0,1, j=-1,0,1 for 3x3 neighborhood

compact notation $g = f * h$

Convolution operation

Linear filter: example

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

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

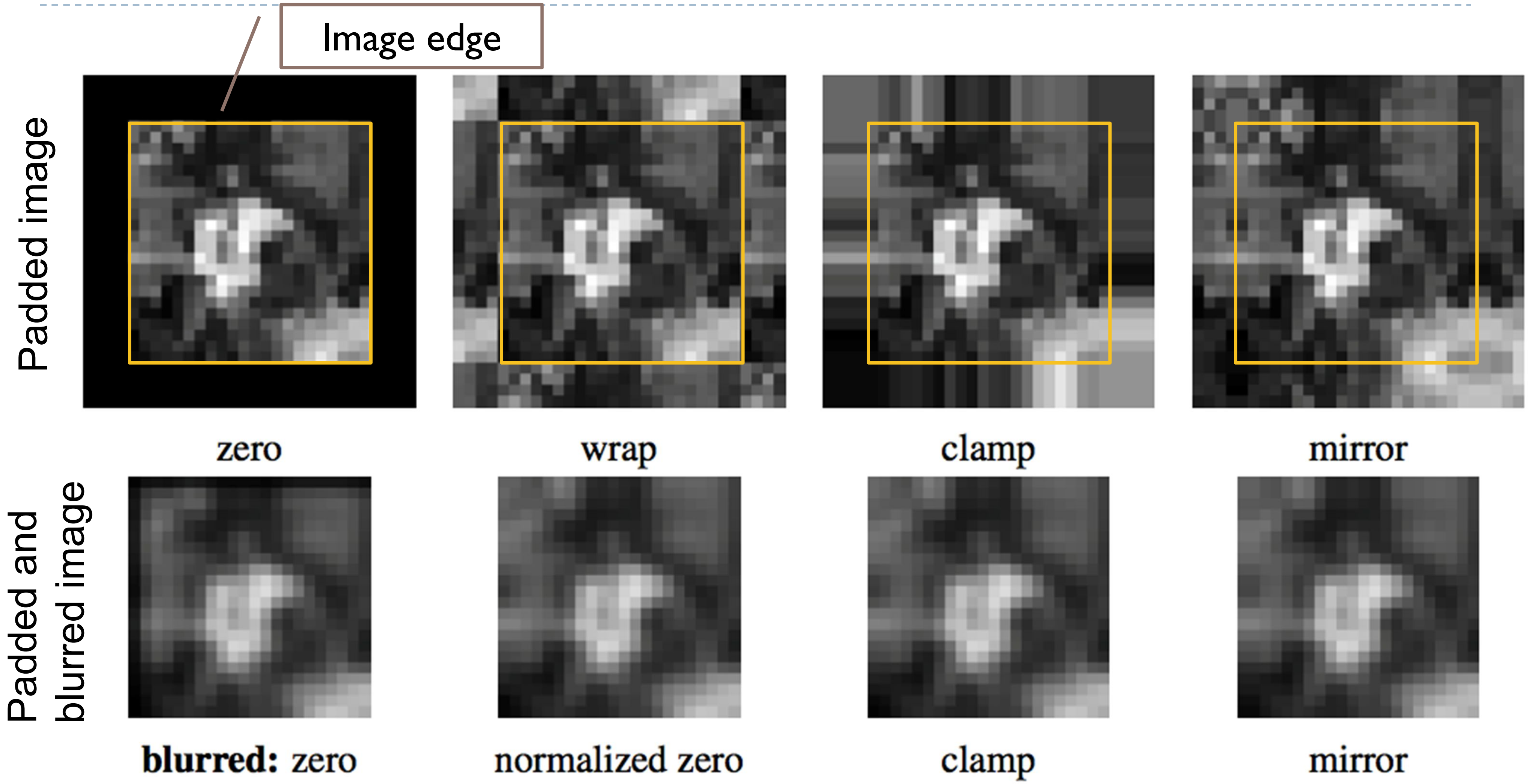
=

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

$g(x,y)$

Why is the matrix g smaller than f ?

Padding an image



What is the computational cost of the convolution?

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

- How many multiplications do we need to do to convolve 100x100 image with 9x9 kernel ?
  - The image is padded, but we do not compute the values for the padded pixels

Separable kernels

- Convolution operation can be made much faster if split into two separate steps:
  - 1) convolve all rows in the image with a 1D filter
  - 2) convolve columns in the result of 1) with another 1D filter
- But to do this, the kernel must be separable

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \cdot \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

Examples of separable filters

- Box filter:

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

- Gaussian filter:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  - What are the corresponding 1D components of this separable filter (u(x) and v(y))?

$$G(x,y) = u(x) \cdot v(y)$$

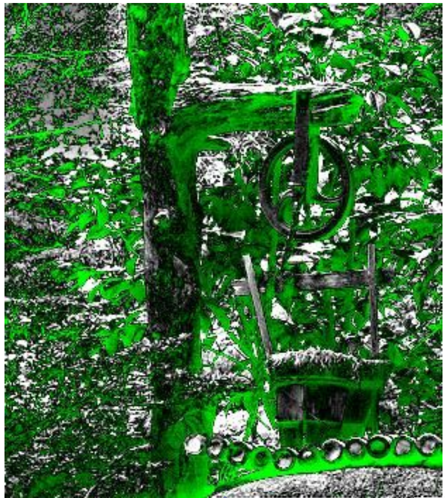
Unsharp masking

- How to use blurring to sharpen an image ?

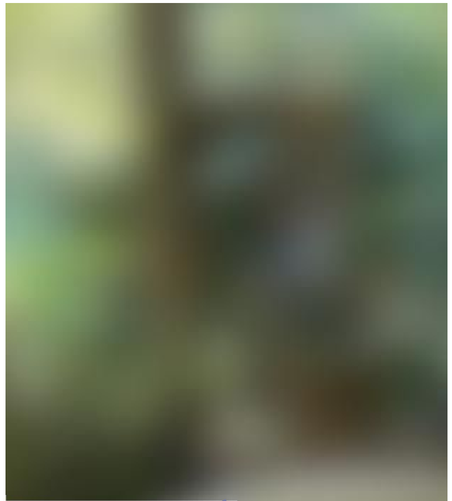
results

original image

high-pass image



blurry image



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

Why "linear" filters ?

- Linear functions have two properties:
  - Additivity: $f(x) + f(y) = f(x + y)$
  - Homogenity: $f(ax) = af(x)$ (where "f" is a linear function)
- Why is it important?
  - Linear operations can be performed in an arbitrary order
$$blur(aF + b) = a\,blur(F) + b$$
  - Linearity of the Gaussian filter could be used to improve the performance of your image processing operation
  - This is also how separable filters work:

Matrix multiplication

Convolution

The components of a separable kernel

$$(u \cdot v) * f = u * (v * f)$$

An image

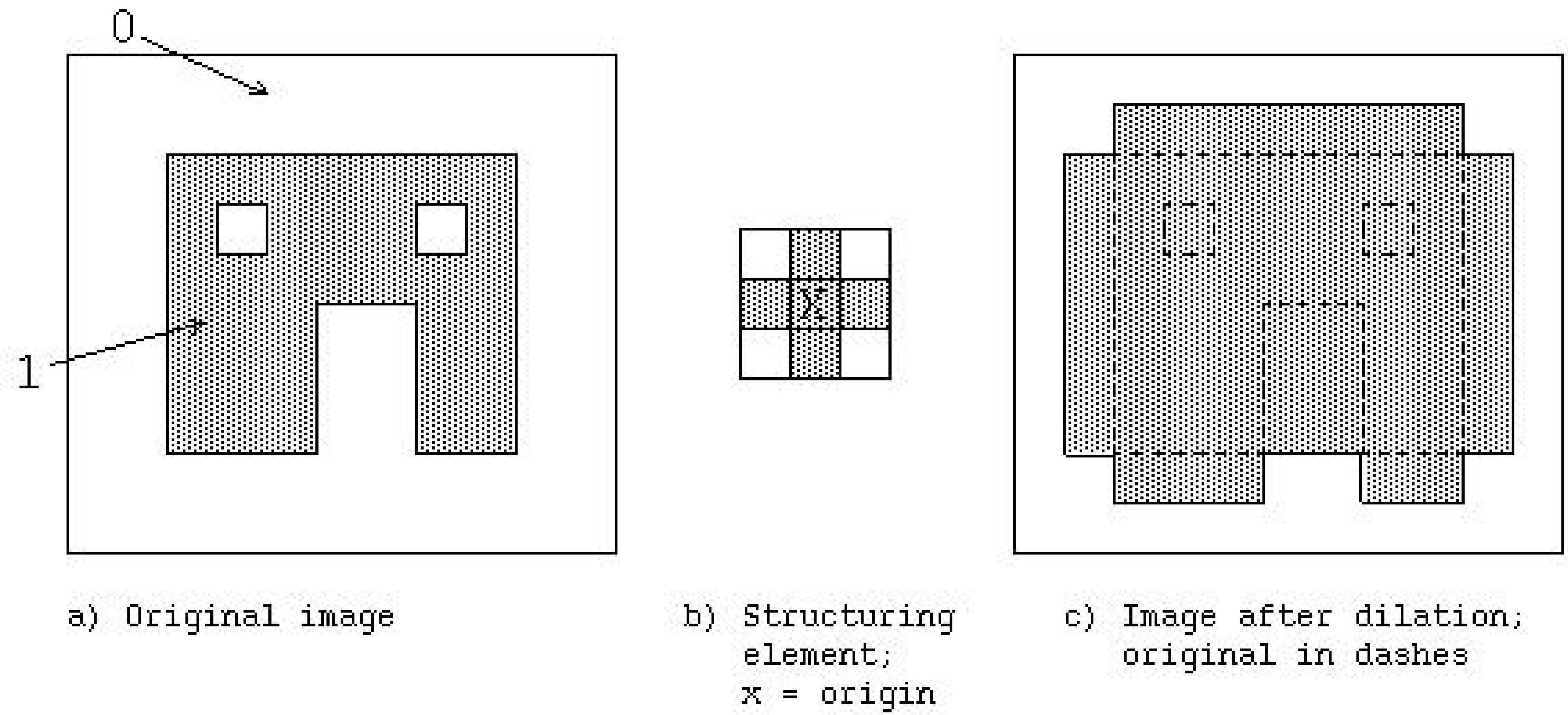
Operations on binary images

- Essential for many computer vision tasks

- Binary image can be constructed by thresholding a grayscale image

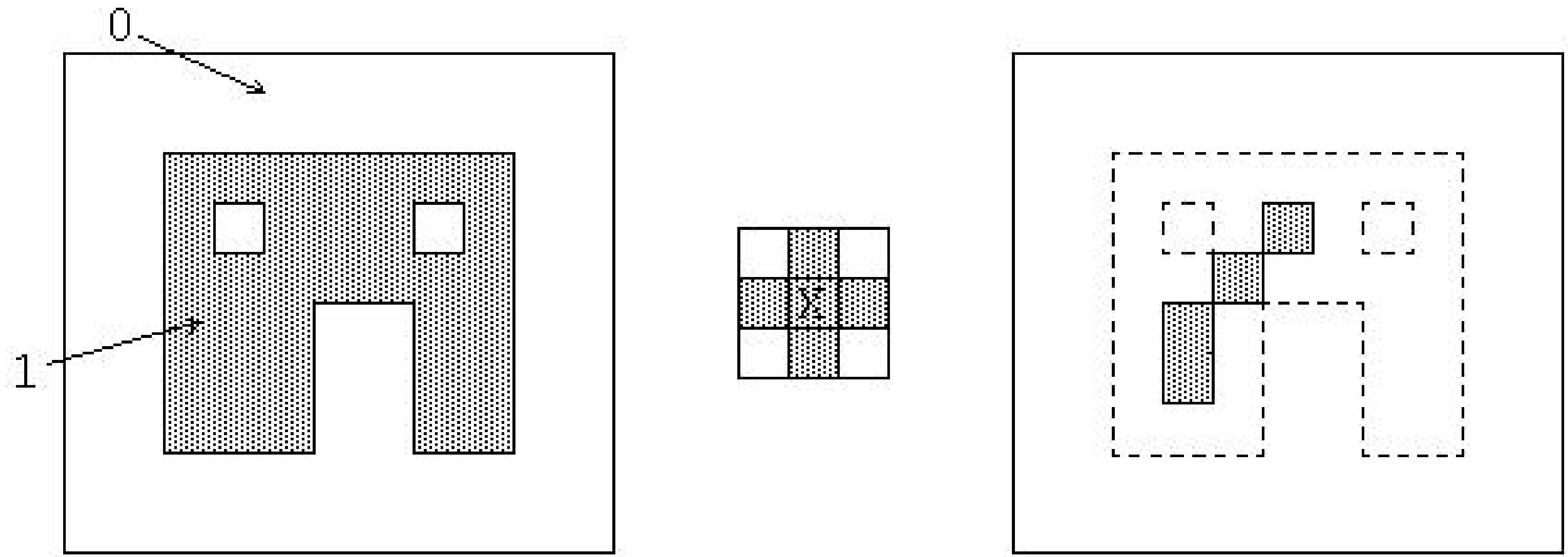
$$\theta(f,c) = \begin{cases} 1 & \text{if } f \geq c, \\ 0 & \text{else,} \end{cases}$$

Morphological filters: dilation



- Set the pixel to the maximum value of the neighboring pixels within the structuring element
- What could it be useful for ?

Morphological filters: erosion



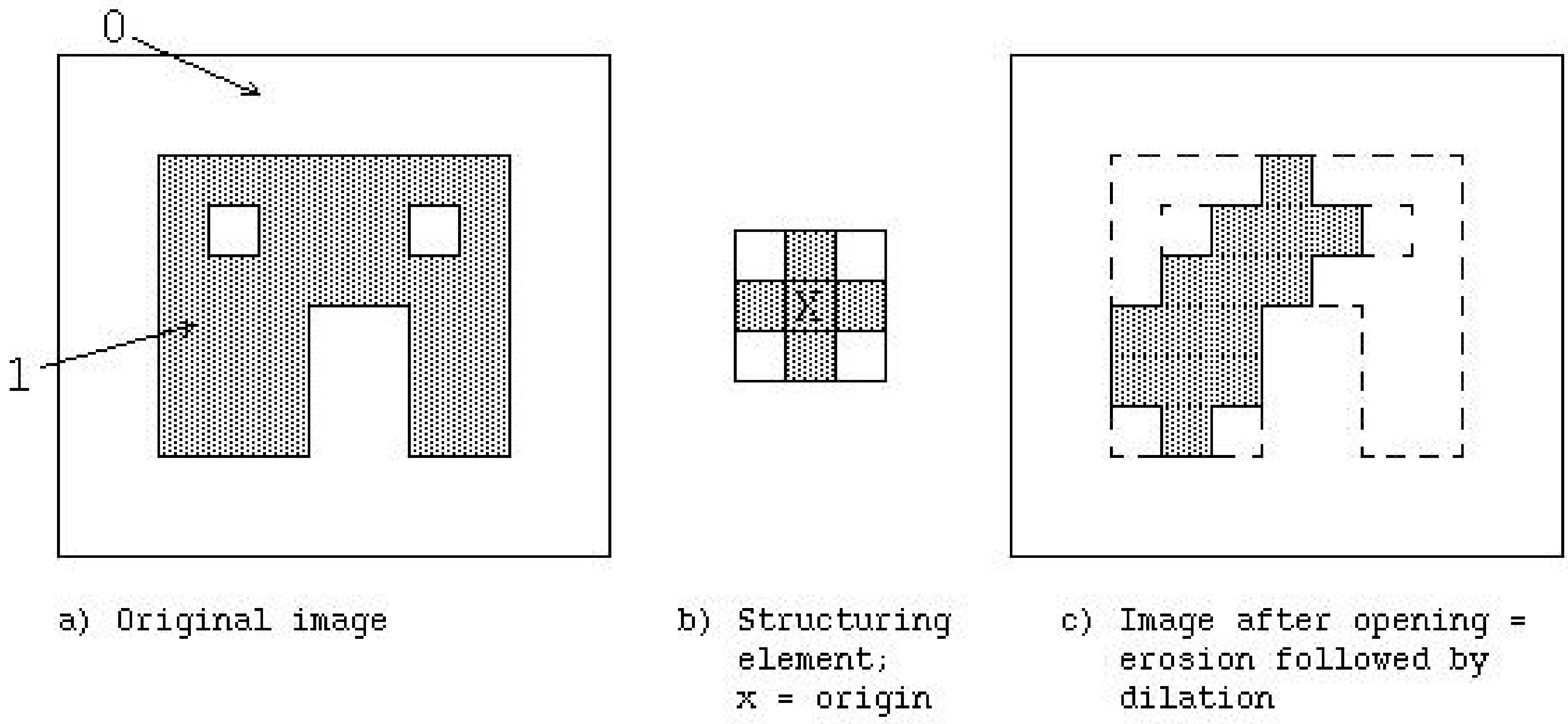
a) Original image

b) Structuring element; x = origin

c) Image after erosion; original in dashes

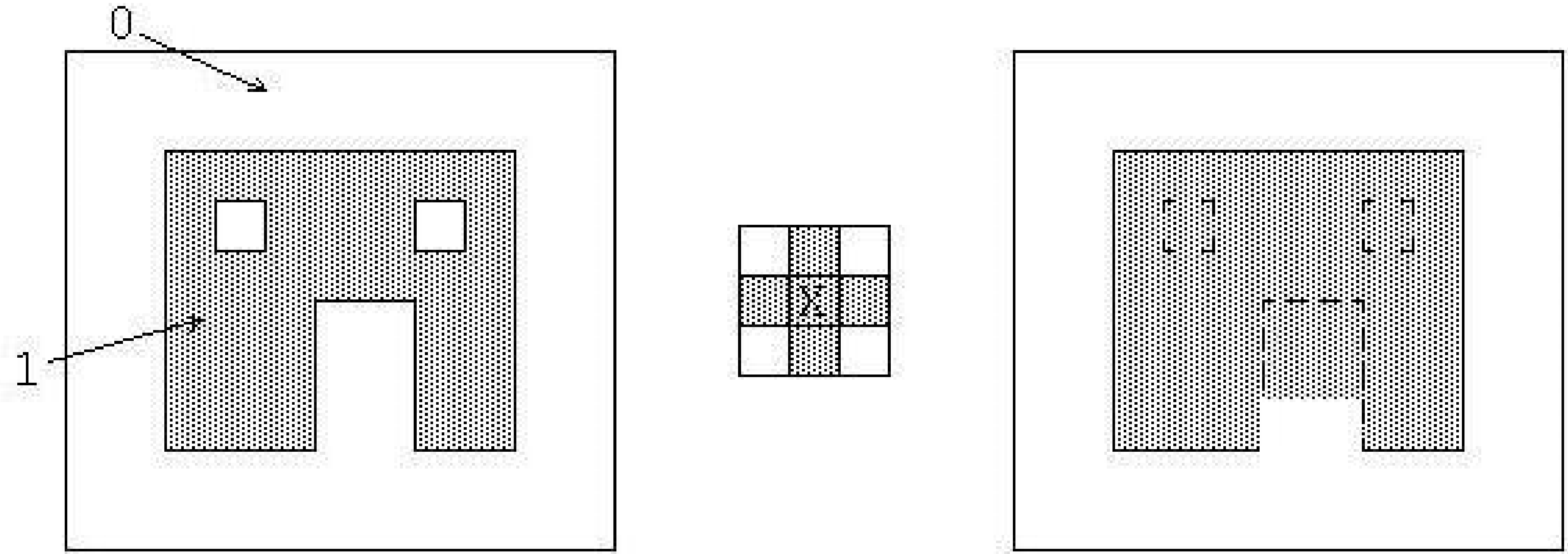
- Set the value to the minimum value of all the neighboring pixels within the structuring element
- What could it be useful for ?

Morphological filters: opening



- Erosion followed by dilation
- What could it be useful for?

Morphological filters: closing



a) Original image

b) Structuring element; x = origin

c) Image after closing = dilation followed by erosion; original in dashes.

- Dilation followed by erosion
- What could it be useful for ?

Binary morphological filters: formal definition

Binary image

Correlation (similar to convolution)

Number of 1s inside the region restricted by the structuring element

$$c = f \otimes s$$

Structuring element

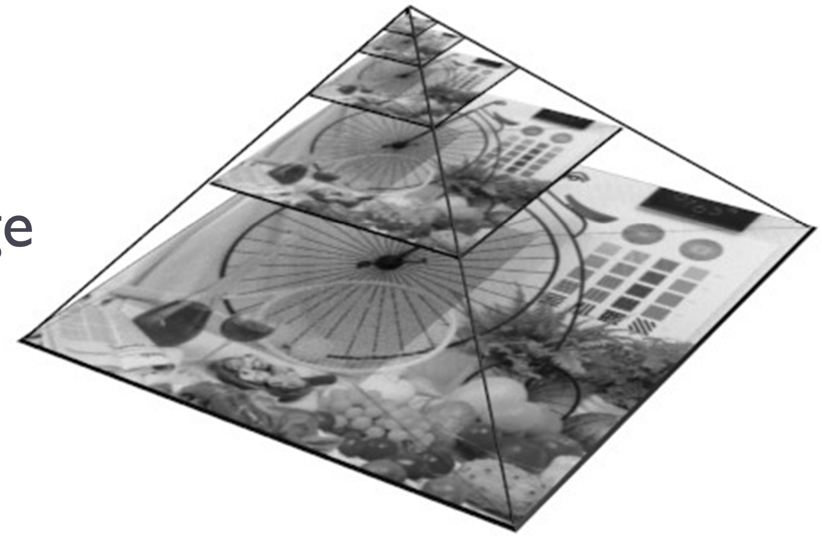
S – size of structuring element (number of 1s in the SI)

$$\theta(a, b) = \begin{cases} 1 & \text{if } a \geq b \\ 0 & \text{otherwise} \end{cases}$$

- **dilation**: $\text{dilate}(f, s) = \theta(c, 1)$;
- **erosion**: $\text{erode}(f, s) = \theta(c, S)$;
- **majority**: $\text{maj}(f, s) = \theta(c, S/2)$;
- **opening**: $\text{open}(f, s) = \text{dilate}(\text{erode}(f, s), s)$;
- **closing**: $\text{close}(f, s) = \text{erode}(\text{dilate}(f, s), s)$.

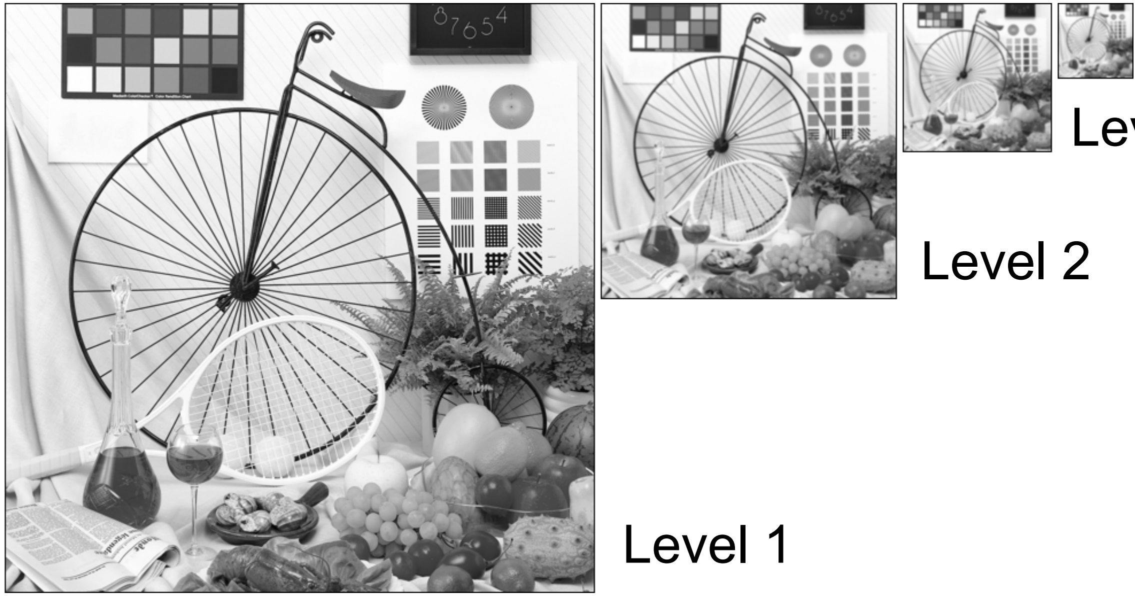
Multi-scale image processing (pyramids)

- Multi-scale processing operates on an image represented at several sizes (scales)
  - Fine level for operating on small details
  - Coarse level for operating on large features
- Example:
  - Motion estimation
    - Use fine scales for objects moving slowly
    - Use coarse scale for objects moving fast
  - Blending (to avoid sharp boundaries)



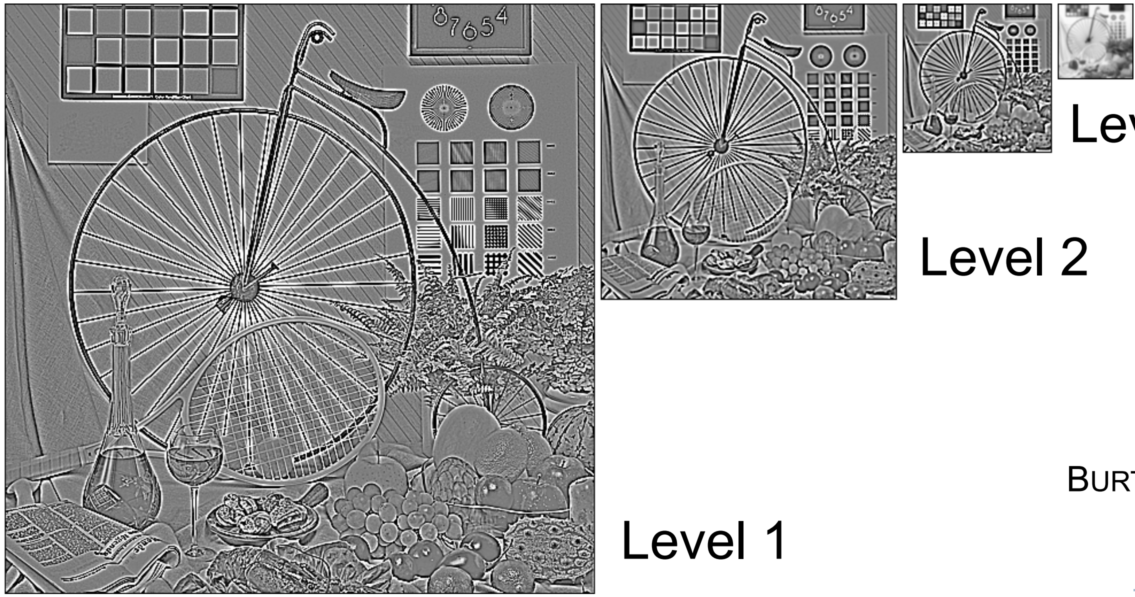
Two types of pyramids

Gaussian pyramid



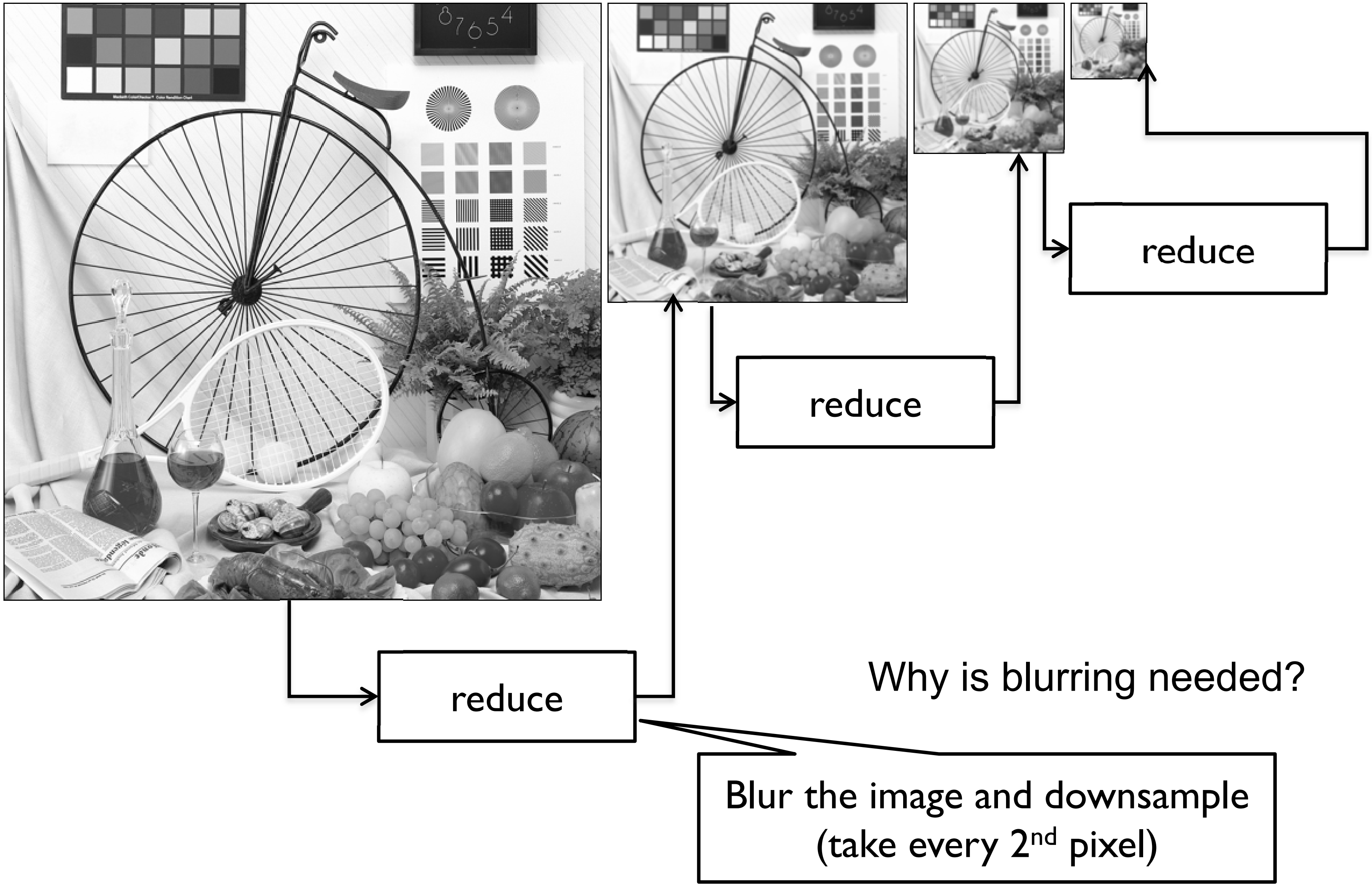
Laplacian pyramid

(a.k.a DoG Diffence of Gaussians)



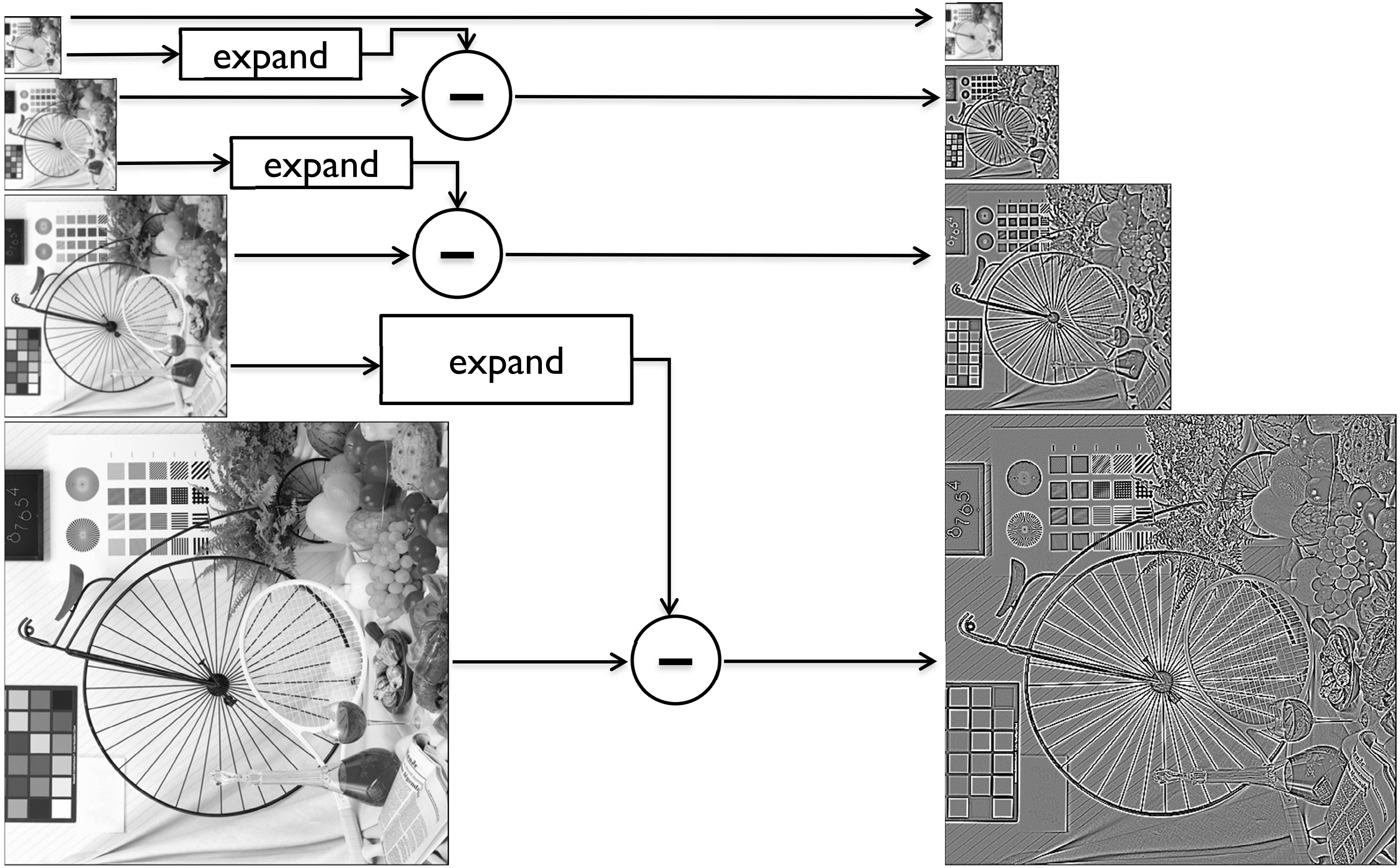
BURT, P. AND ADELSON, E. 1983. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications 31*, 4, 532–540.

Gaussian Pyramid

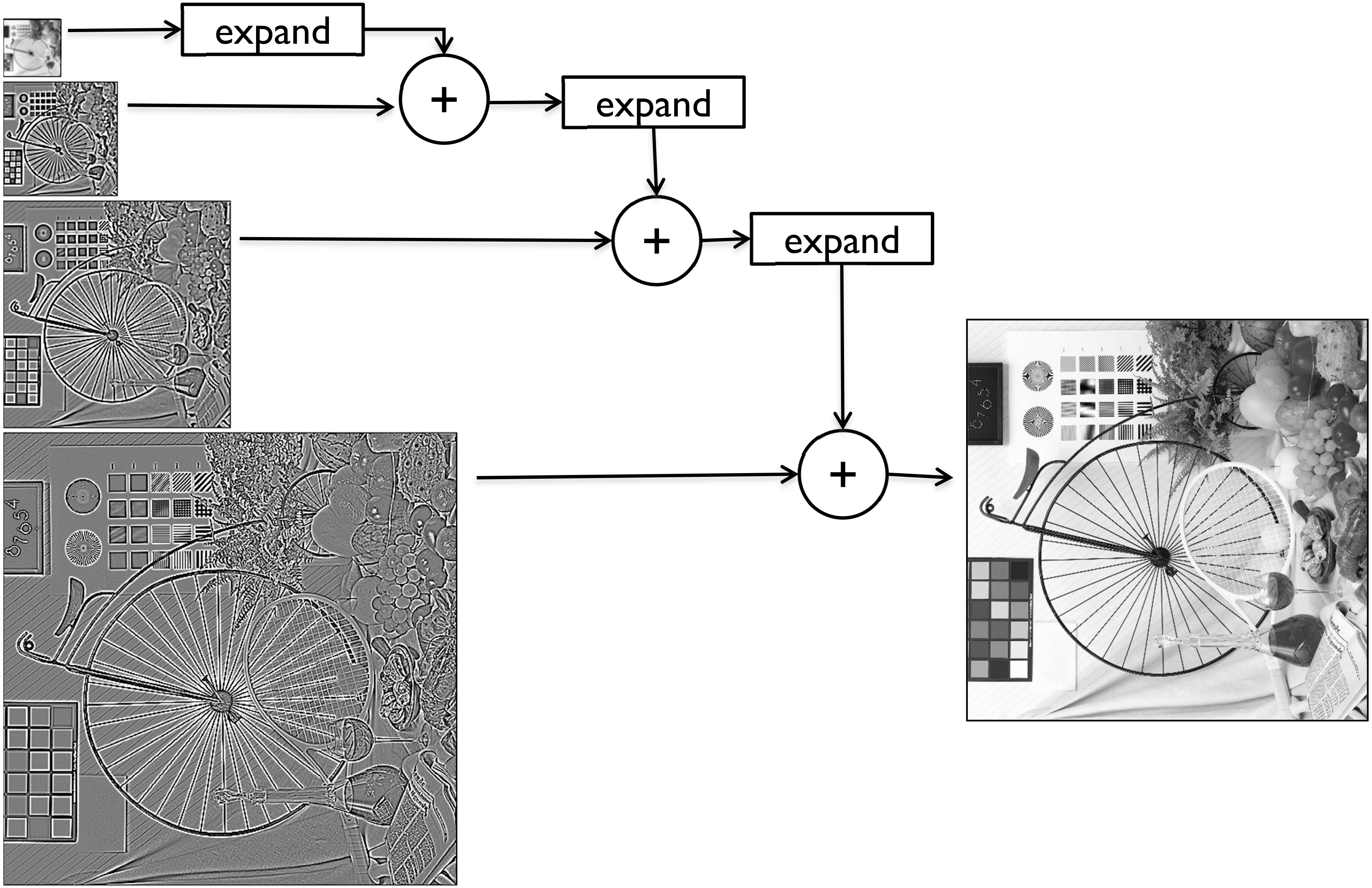


Why is blurring needed?

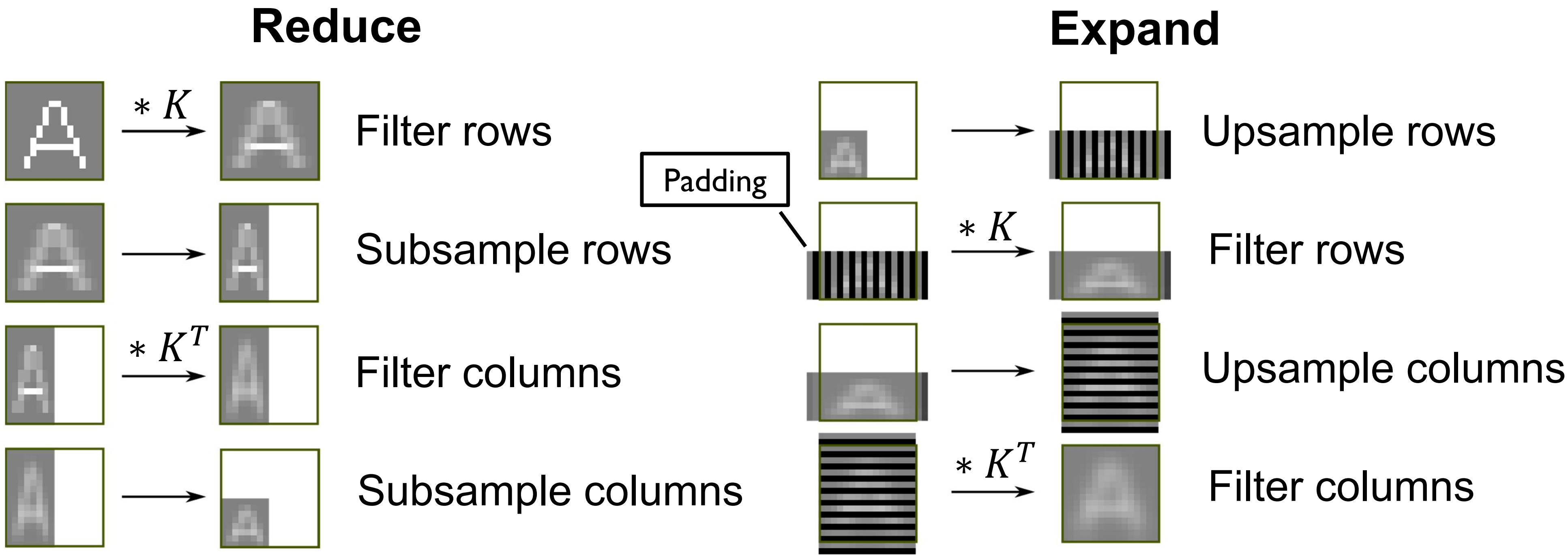
Laplacian Pyramid - decomposition



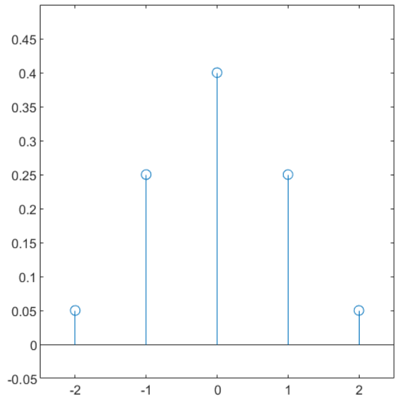
Laplacian Pyramid - synthesis

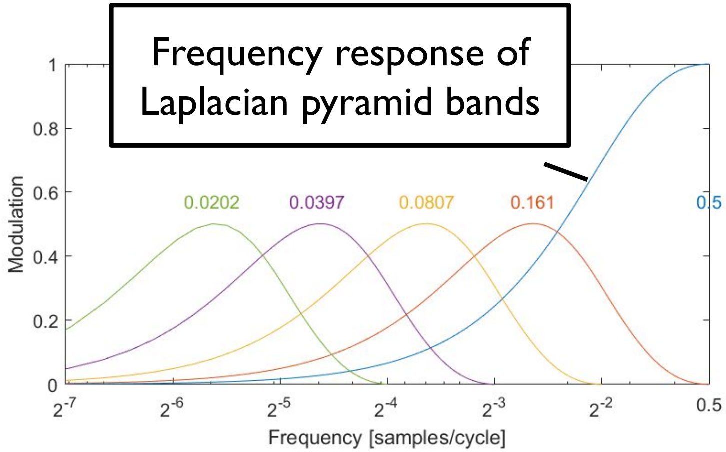


Reduce and expand



$K =$





Example: stitching and blending

Combine two images:

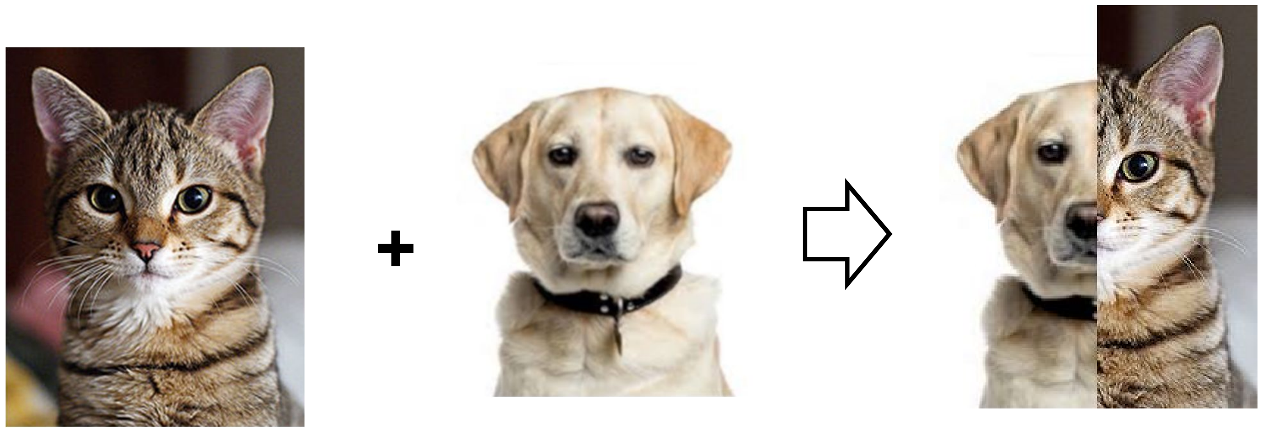


Image-space blending

Laplacian pyramid blending

References

- SZELISKI, R. 2010. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York Inc.
  - Chapter 3
  - http://szeliski.org/Book

Advanced Graphics & Image Processing

Advanced image processing

Part 1/2 – edge stopping filters

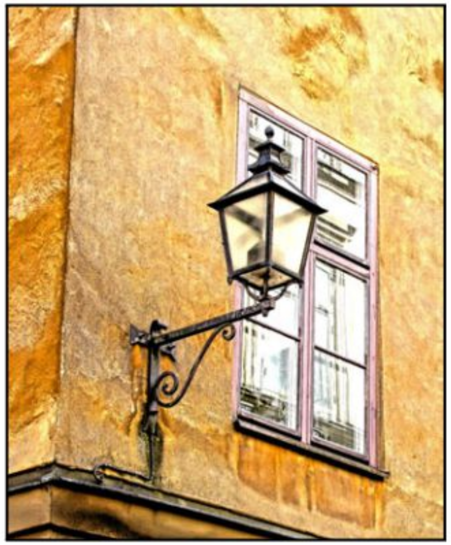
Rafał Mantiuk

Computer Laboratory, University of Cambridge

Edge stopping filters

Original

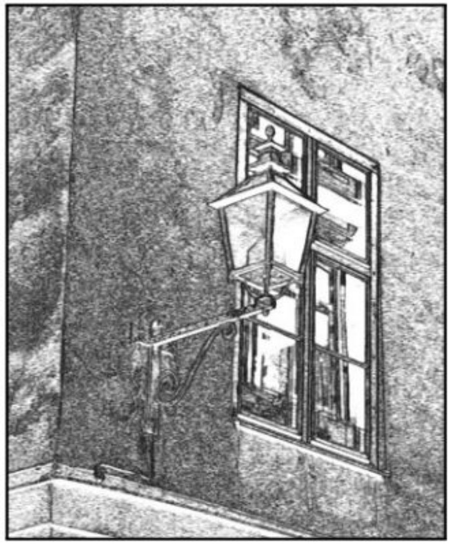
Edge-aware smoothing



Detail enhancement

Stylization

Recoloring



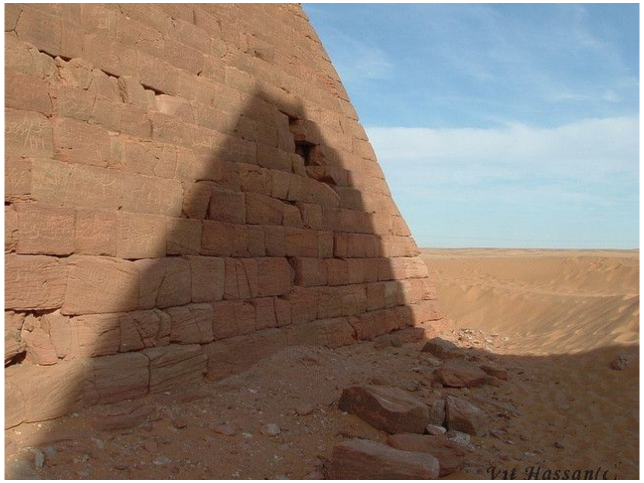
Pencil drawing

Depth-of-field

Examples from [Gastal & Oliveira 2011]

Nonlinear filters: Bilateral filter

- Goal: Smooth out an image without blurring edges

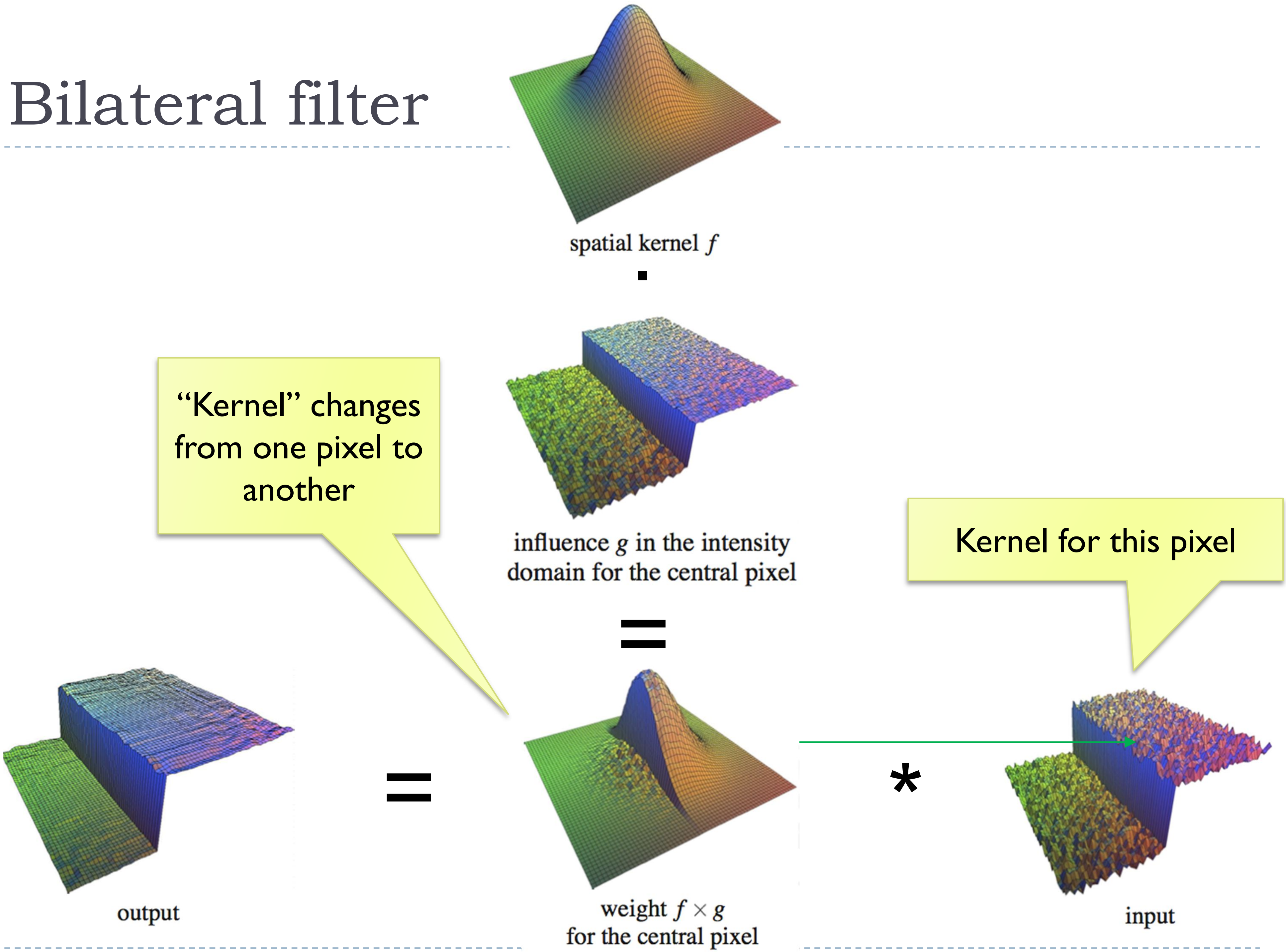


Gaussian filter

Unsharp masking

Bilateral filter

Bilateral filter



Bilateral filter

Input image

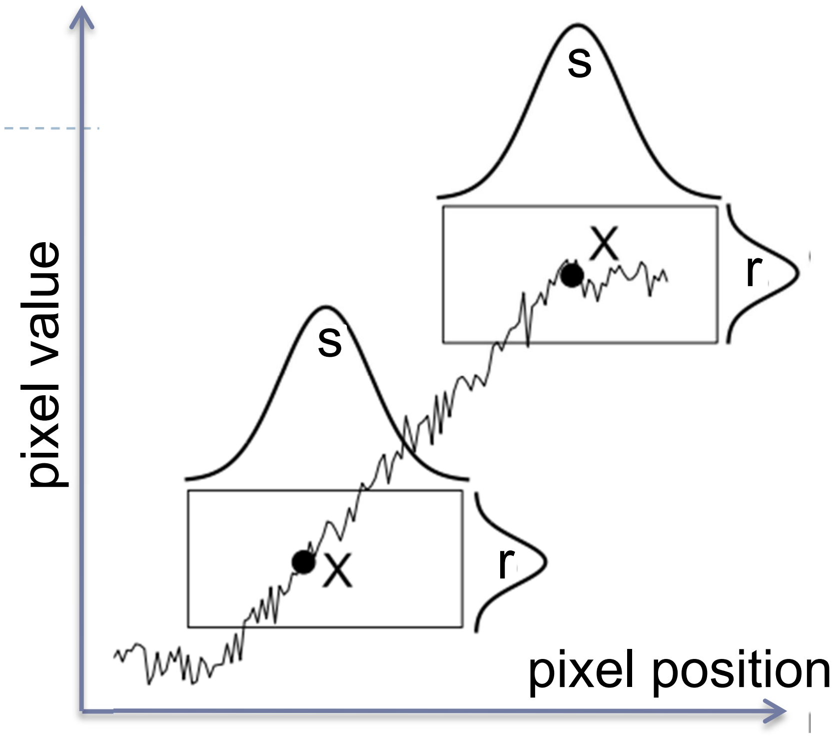
$$y(\boldsymbol{p}) = \frac{\sum_{\boldsymbol{q}\in\Omega} x(\boldsymbol{q})w(\boldsymbol{p},\boldsymbol{q})}{\sum_{\boldsymbol{q}\in\Omega} w(\boldsymbol{p},\boldsymbol{q})}$$

Pixel coordinates $\boldsymbol{p} = (i, j)$

Neighborhood of the pixel **p**

$$w(\boldsymbol{p},\boldsymbol{q}) = g_s\underbrace{(\boldsymbol{p}-\boldsymbol{q})}_{\text{distance in the spatial position (x,y)}}\,g_r\underbrace{(x(\boldsymbol{p})-x(\boldsymbol{q}))}_{\text{distance (difference) in pixel values}}$$

$$g_s(\boldsymbol{d}) = \exp\left(\frac{-\|\boldsymbol{d}\|_2}{2\sigma_s^2}\right) \qquad g_r(d) = \exp\left(\frac{-d^2}{2\sigma_r^2}\right)$$



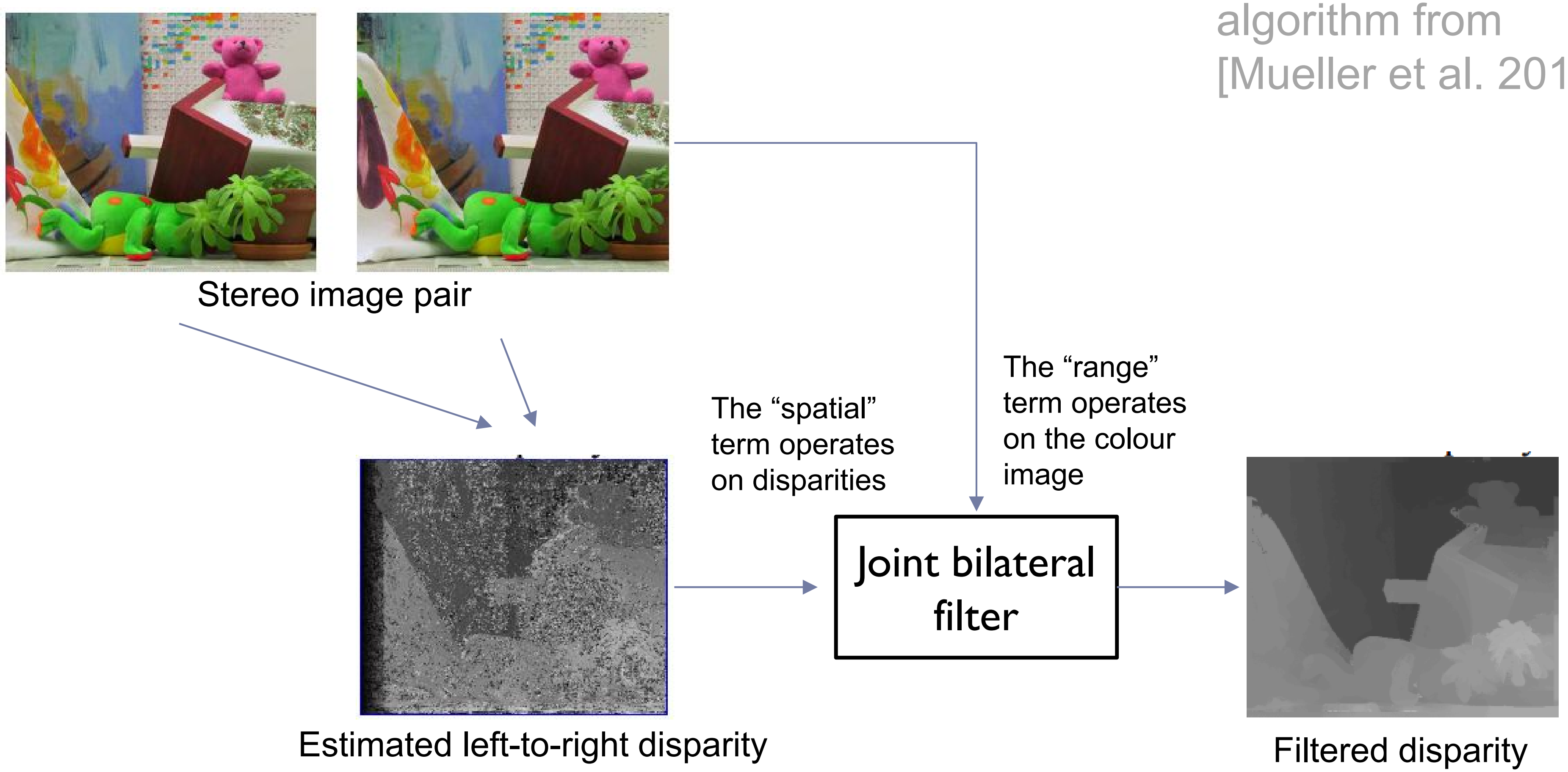
How to make the bilateral filter fast?

- A number of approximations have been proposed
  - Combination of linear filters [Durand & Dorsey 2002, Yang et al. 2009]
  - Bilateral grid [Chen et al. 2007]
  - Permutohedral lattice [Adams et al. 2010]
  - Domain transform [Gastal & Oliveira 2011]

Joint-bilateral filter (a.k.a guided/cross b.f.)

- The "range" term does not need to operate in the same domain as the filter output
  - Example:

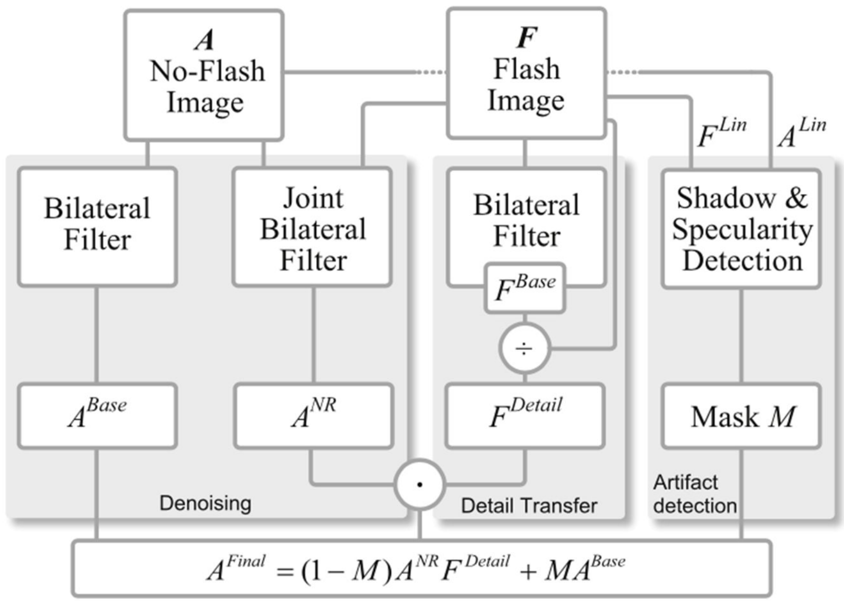
A simplified algorithm from [Mueller et al. 2010]



Joint bilateral filter: Flash / no-flash

Flash

No-flash



- Preserve colour and illumination from the no-flash image
- Use flash image to remove noise and add details
- [Petshnigg et al. 2004]

Example of edge preserving filtering

- Domain Transform for Edge-Aware Image and Video Processing
- Video:
  - https://youtu.be/UI1xh1IQrTY?t=4m10s
  - From: http://inf.ufrgs.br/~eslgastal/DomainTransform/

Advanced Graphics & Image Processing

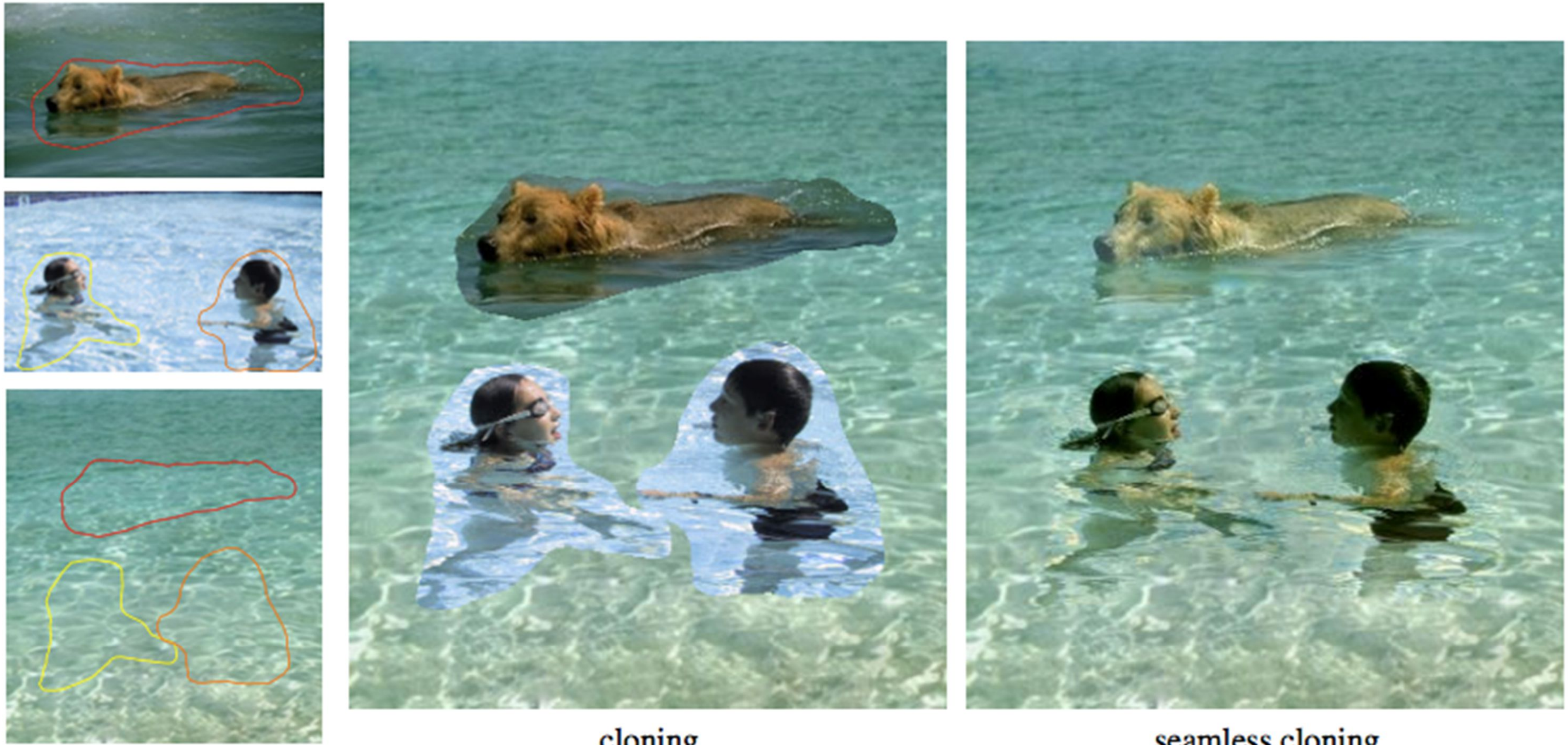
Advanced image processing

Part 1/2 – processing by optimization

Rafał Mantiuk

Computer Laboratory, University of Cambridge

Optimization-based methods



sources/destinations

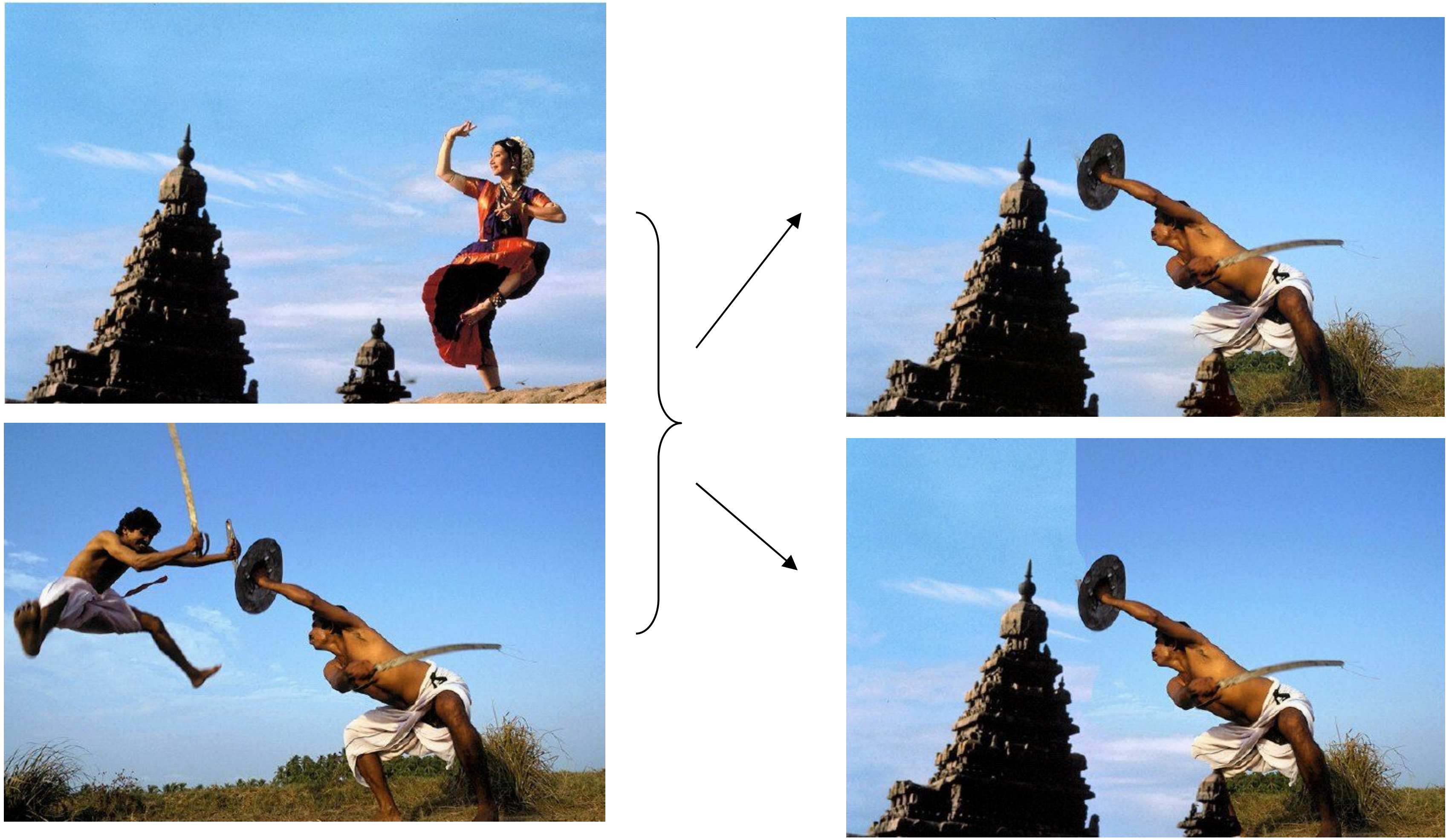
cloning

seamless cloning

Poisson image editing [Perez et al. 2003]

Gradient Domain compositing

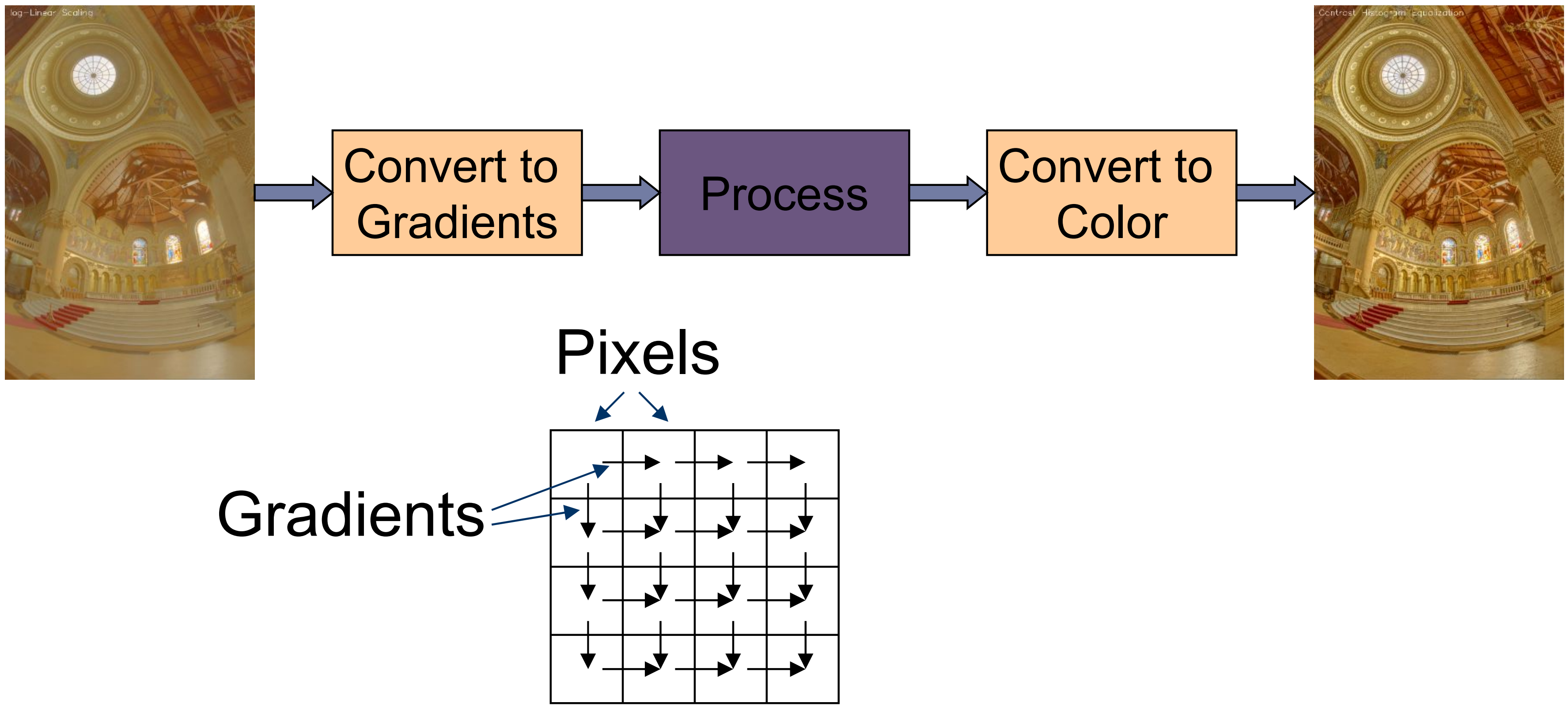
- Compositing [Wang et al. 2004]



images from [Drori at al. 2004]

Gradient domain methods

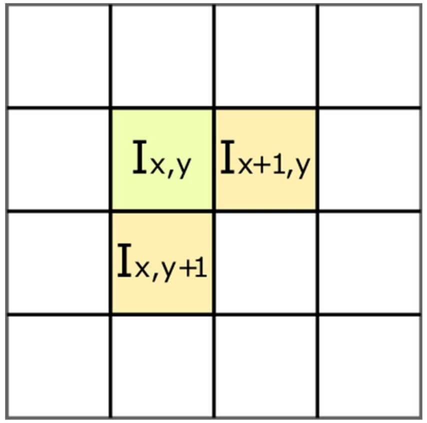
- Operate on pixel gradients instead of pixel values



Forward Transformation

- Forward Transformation
  - Compute gradients as differences between a pixel and its two neighboors

$$\nabla I_{x,y} = \begin{bmatrix} I_{x+1,y} - I_{x,y} \\ I_{x,y+1} - I_{x,y} \end{bmatrix}$$



  - Result: 2D gradient map (2 x more values than the number of pixels)

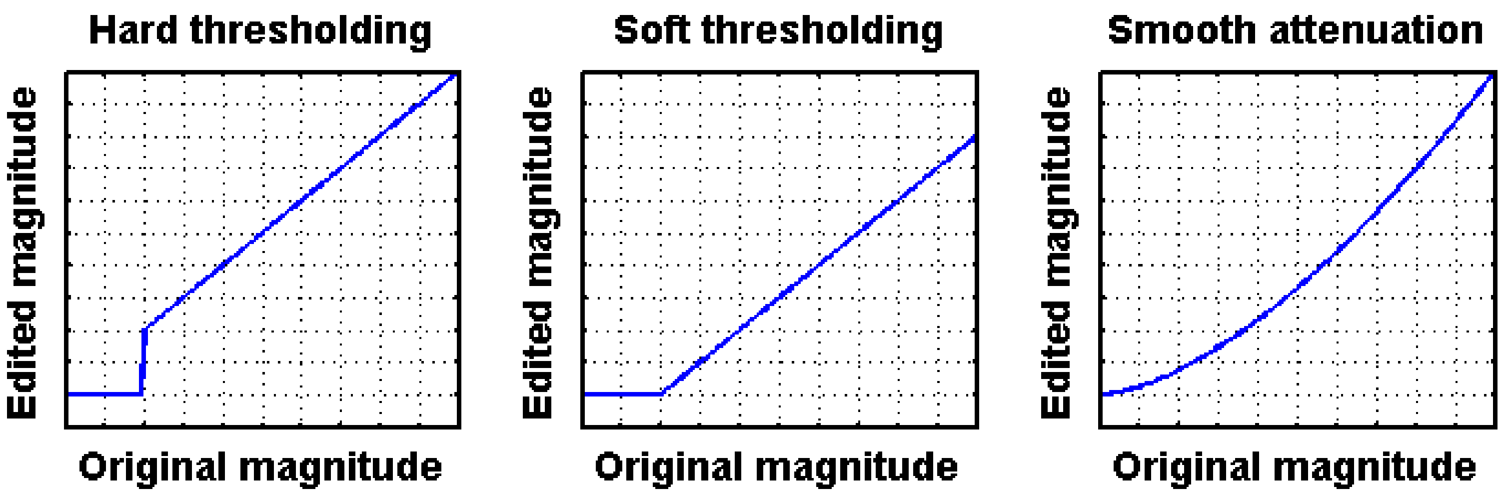
Processing gradient field

- Typically, gradient magnitudes are modified while gradient direction (angle) remains the same

$$G_{x,y} = \nabla I_{x,y} \cdot \frac{f\left(||\nabla I_{x,y}||\right)}{||\nabla I_{x,y}||}$$

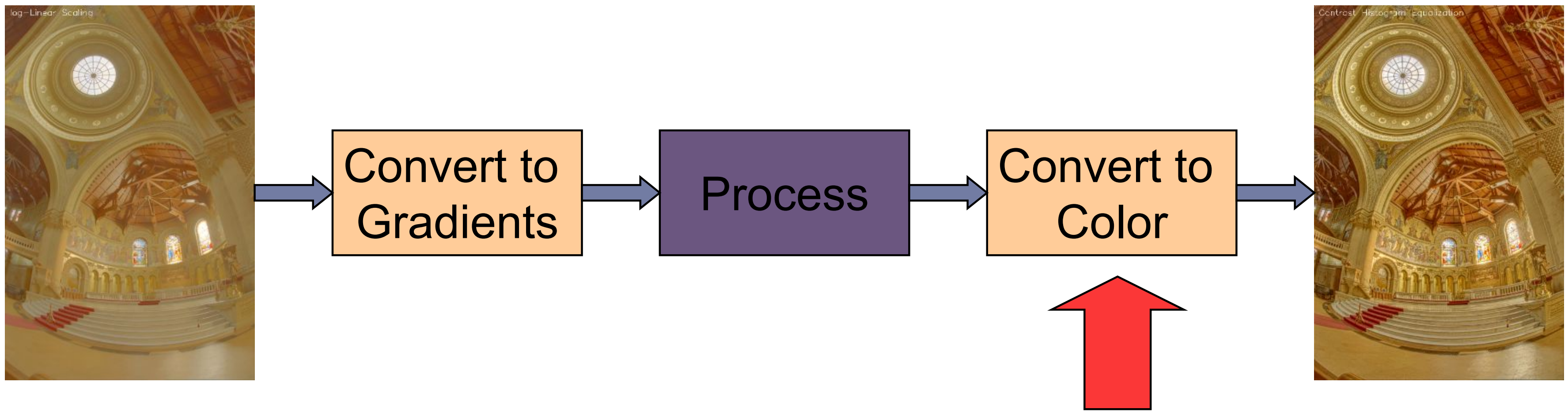
Gradient editing function

- Examples of gradient editing functions:



Inverse transform: the difficult part

- There is no strightforward transformation from gradients to luminance



- Instead, a minimization problem is solved:

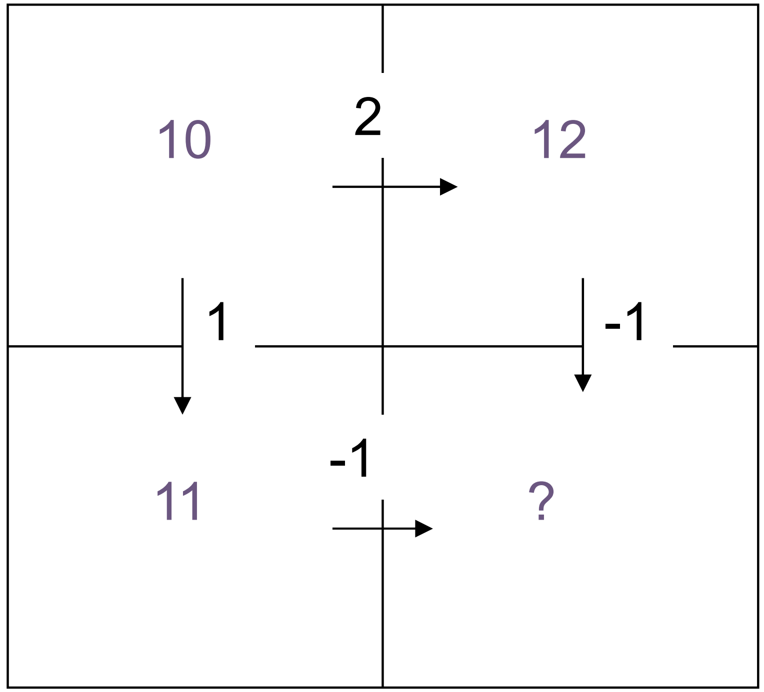
$$\arg\min_{I} \sum_{x,y} \left[\left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right]$$

Image Pixels

Desired gradients

Inverse transformation

- Convert modified gradients to pixel values
  - Not trivial!
  - Most gradient fields are inconsistent - do not produce valid images
  - If no accurate solution is available, take the best possible solution
  - Analogy: system of springs



Gradient field reconstruction: derivation

- The minimization problem is given by:

$$\arg\min_{I} \sum_{x,y} \left[\left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right]$$

- After equating derivatives over pixel values to 0 we get:
  - Derivation done in the lecture

$$I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y} = G^{(x)}_{x,y} - G^{(x)}_{x-1,y} + G^{(y)}_{x,y} - G^{(y)}_{x,y-1}$$

  - In matrix notation:

Laplace operator (NxN matrix)

$$\nabla^2 I = \mathrm{div} G$$

Divergence of a vector field (Nx1 vector)

$$\nabla^2 I_{x,y} = I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y}$$

Image as a column vector

$$\begin{bmatrix} I_{1,1} \\ I_{2,1} \\ \cdots \\ I_{N,M} \end{bmatrix}$$

$$\mathrm{div} G_{x,y} = G^{(x)}_{x,y} - G^{(x)}_{x-1,y} + G^{(y)}_{x,y} - G^{(y)}_{x,y-1}$$

Laplace operator for 3x3 image

$$\nabla^2 = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}$$

Solving sparse linear systems

- Just use "\" operator in Matlab / Octave:
  - x = A \ b;
- Great "cookbook":
  - TEUKOLSKY, S.A., FLANNERY, B.P., PRESS, W.H., AND VETTERLING, W.T. 1992. *Numerical recipes in C*. Cambridge University Press, Cambridge.
- Some general methods
  - Cosine-transform – fast but cannot work with weights (next slides) and may suffer from floating point precision errors
  - Multi-grid – fast, difficult to implement, not very flexible
  - Conjugate gradient / bi-conjugate gradient – general, memory efficient, iterative but fast converging
  - Cholesky decomposition – effective when working on sparse matrices

Pinching artefacts

- A common problem of gradient-based methods is that they may result in “pinching” artefacts (left image)
- Such artefacts can be avoided by introducing weights to the optimization problem

Weighted gradients

- The new objective function is:

$$\arg\min_I \sum_{x,y} \left[w^{(x)}_{x,y} \left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + w^{(y)}_{x,y} \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right]$$

- so that higher weights are assigned to low gradient magnitudes (in the original image).

$$w^{(x)}_{x,y} = w^{(y)}_{x,y} = \frac{1}{||\nabla I^{(o)}_{x,y}|| + \epsilon}$$

- The linear system can be derived again
  - but this is a lot of work and is error-prone

Weighted gradients - matrix notation (1)

- The objective function:

$$\arg\min_{I} \sum_{x,y} \left[w^{(x)}_{x,y} \left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + w^{(y)}_{x,y} \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right]$$

- In the matrix notation (without weights for now):

$$\arg\min_{I} \left\| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I - \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix} \right\|^2$$

> Note that I is a column vector with an image. It is not an identity matrix!

- Gradient operators (for 3x3 pixel image):

$$\nabla_x = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \nabla_y = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Weighted gradients - matrix notation (2)

- The objective function again: $\arg\min_I \left|\left| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I - \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix} \right|\right|^2$

- Such over-determined least-square problem can be solved using pseudo-inverse:

Matrix transpose

$$\begin{bmatrix} \nabla'_x & \nabla'_y \end{bmatrix} \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I = \begin{bmatrix} \nabla'_x & \nabla'_y \end{bmatrix} \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix}$$

- Or simply:

$$\left(\nabla'_x \nabla_x + \nabla'_y \nabla_y\right) I = \nabla'_x G^{(x)} + \nabla'_y G^{(y)}$$

- With weights:

$$\left(\nabla'_x W \nabla_x + \nabla'_y W \nabla_y\right) I = \nabla'_x W G^{(x)} + \nabla'_y W G^{(y)}$$

WLS filter: Edge stopping filter by optimization

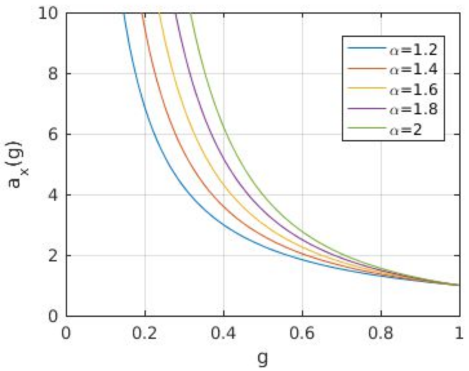
- Weighted-least-squares optimization

Make reconstructed image *u* possibly close to input *g*

Smooth out the image by making partial derivatives close to 0

$$\underset{\boldsymbol{u}}{\operatorname{argmin}} \sum_p \left((u_p - g_p)^2 + \lambda \left(a_{x,p}(g) \left(\frac{\partial u}{\partial x}\right)_p^2 + a_{y,p}(g) \left(\frac{\partial u}{\partial y}\right)_p^2 \right) \right)$$

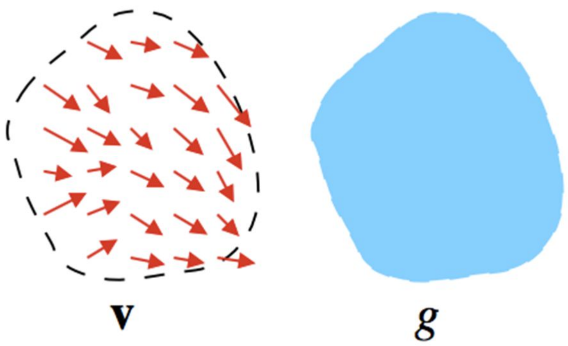
Spatially varying smoothing – less smoothing near the edges

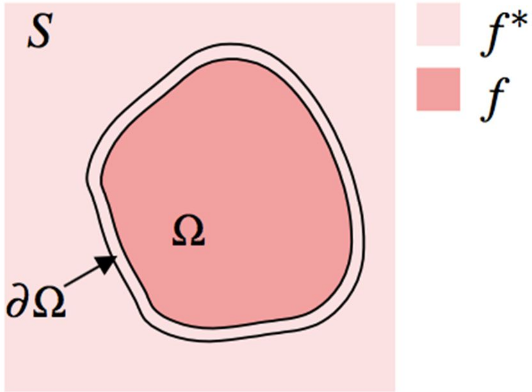


$$a_{x,p}(g) = \frac{1}{\left|\frac{\partial u}{\partial x}(g)\right|^{\alpha} + \epsilon}$$

- [Farbman, Z., Fattal, R., Lischinski, D., & Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM SIGGRAPH 2008*, 1–10.]

Poisson image editing



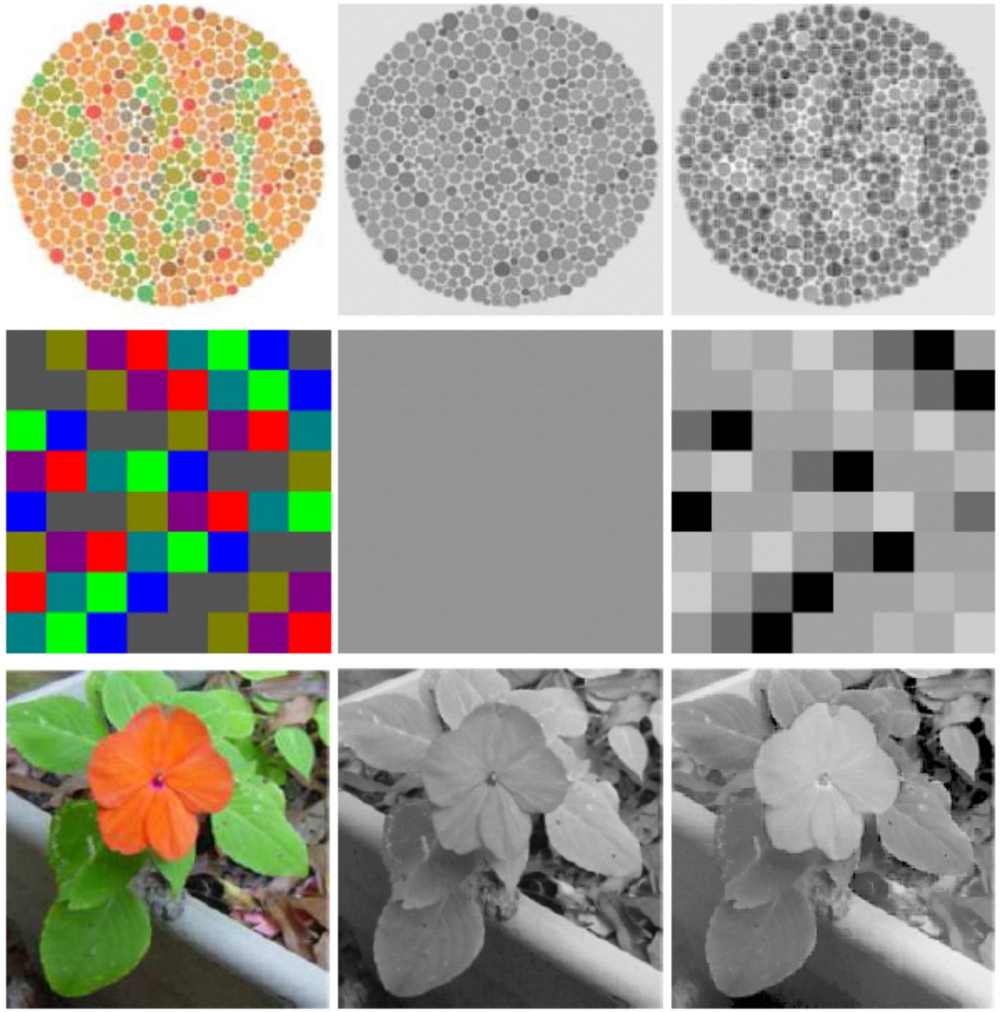


$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \qquad \text{subject to:} \qquad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

- Reconstruct unknown values *f* given a source guidance gradient field *v* and the boundary conditions $f|_{\partial\Omega} = f^*|_{\partial\Omega}$
- [Pérez, P., Michel Gangnet, & Blake, A. (2003). Poisson Image Editing. *ACM Transactions on Graphics*, *3*(22), 313–318. https://doi.org/10.1145/882262.882269]

Colour 2 Gray

- Transform colour images to gray scale
- Preserve colour saliency
  - When gradient in luminance close to 0
  - Replace it with gradient in chrominance
  - Reconstruct an image from gradients
- Gooch, A. A., Olsen, S. C., Tumblin, J., & Gooch, B. (2005). Color2Gray. *ACM Transactions on Graphics*, *24*(3), 634. https://doi.org/10.1145/1073204.1073241



Gradient Domain: applications

- More applications:
  - Lightness perception (Retinex) [Horn 1974]
  - Matting [Sun et al. 2004]
  - Color to gray mapping [Gooch et al. 2005]
  - Video Editing [Perez at al. 2003, Agarwala et al. 2004]
  - Photoshop's Healing Brush [Georgiev 2005]

References

- F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 257–266, Jul. 2002.
- E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, p. 1, Jul. 2011.
- Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Trans. Graph.* 22, 3 (July 2003), 313-318. DOI: http://dx.doi.org/10.1145/882262.882269
- Zeev Farbman, Raanan Fattal, Dani Lischinski, and Richard Szeliski. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph.* 27, 3, Article 67 (August 2008), 10 pages. DOI: https://doi.org/10.1145/1360612.1360666

Advanced Graphics & Image Processing

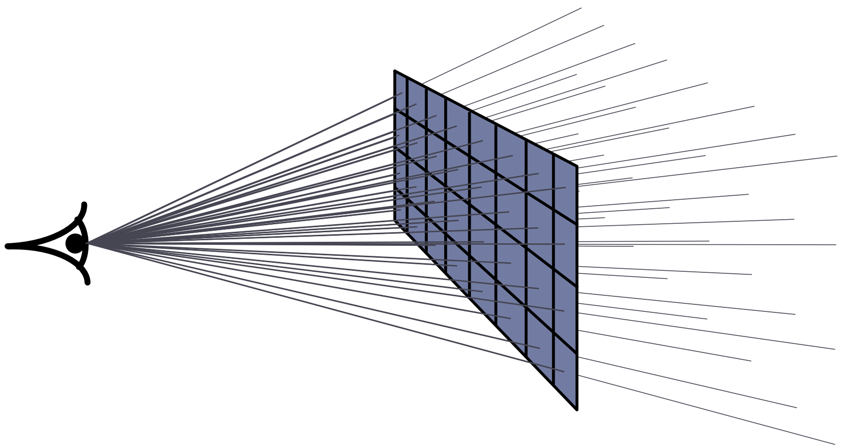
Ray tracing (refresher)

Alejandro Sztrajman, Rafał Mantiuk
Computer Laboratory, University of Cambridge

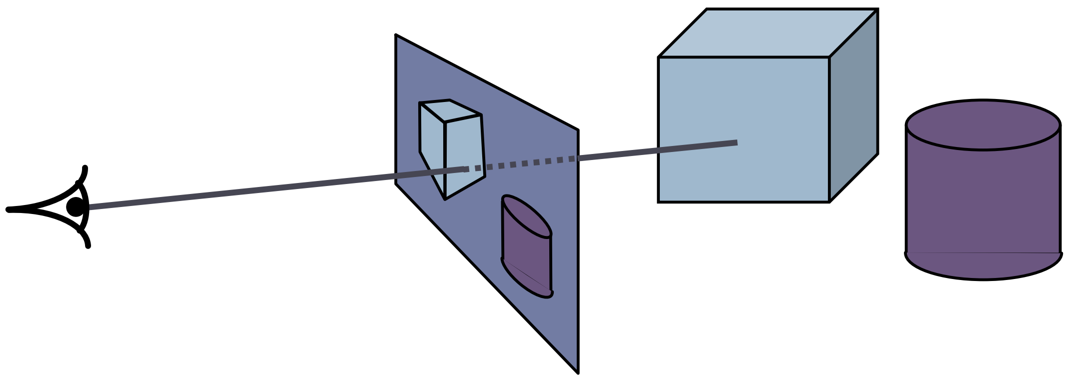
Ray tracing

Given a set of 3D objects, shoot a ray from the eye through the centre of every pixel and see what surfaces it hits

- Identify point on surface and calculate illumination



shoot a ray through each pixel



whatever the ray hits determines the colour of that pixel

Ray tracing algorithm

```
select an eye point and a screen plane

FOR every pixel in the screen plane
    determine the ray from the eye through the pixel's centre
    FOR each object in the scene
        IF the object is intersected by the ray
            IF the intersection is the closest (so far) to the eye
                record intersection point and object
            END IF ;
        END IF ;
    END FOR ;
    calculate colour for the closest intersection point (if any)
END FOR ;
```

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics & Image Processing

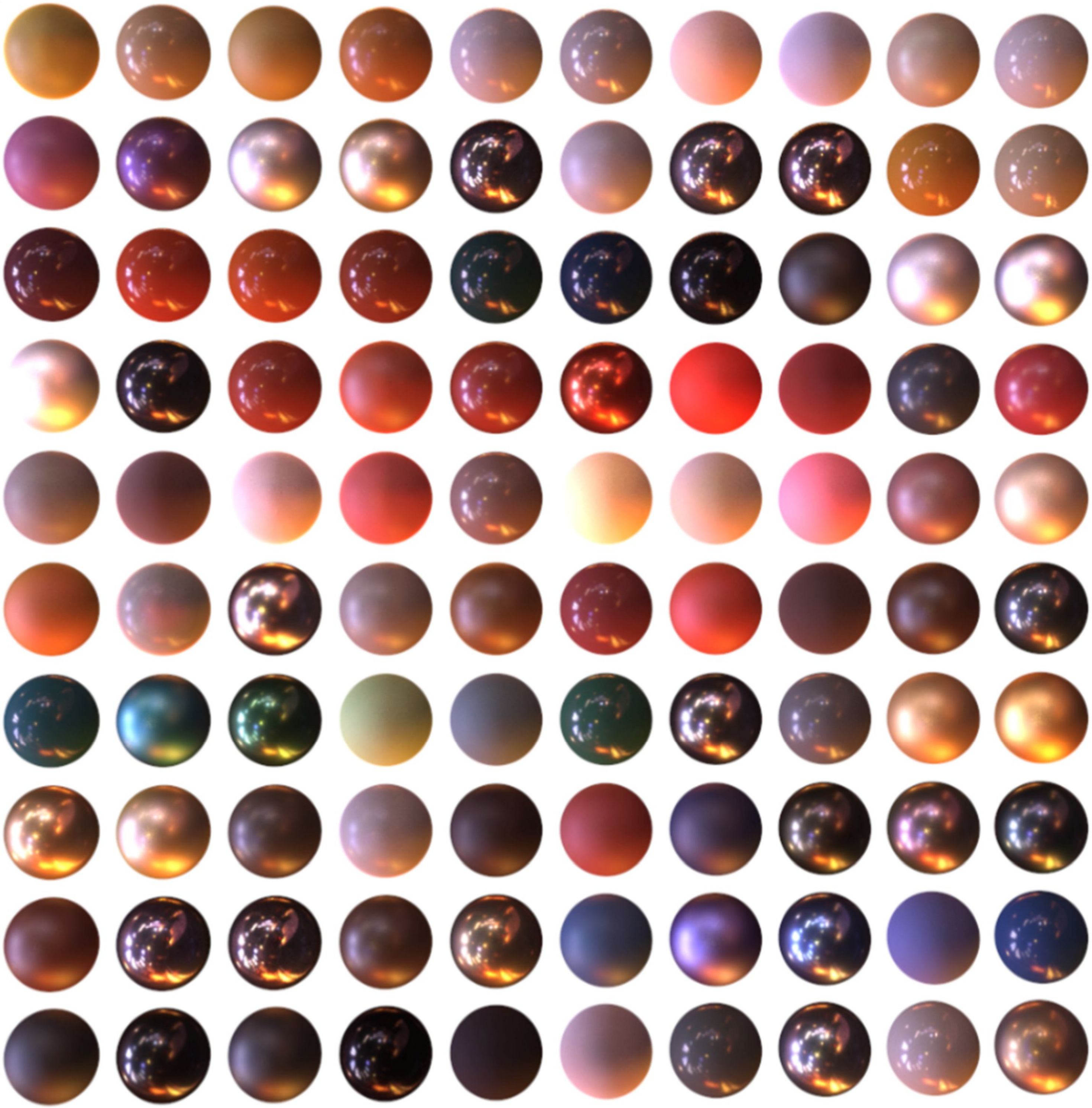
Reflection models and radiometry

Advanced Graphics

Alejandro Sztrajman, Rafał Mantiuk
Computer Laboratory, University of Cambridge

Applications

- To render realistic looking materials
- Applications also in computer vision, optical engineering, remote sensing, etc.
  - To understand how surfaces reflect light



Applications

- Many applications require faithful reproduction of material appearance

Source: http://ikea.com/

Source: http://www.mercedes-benz.co.uk/

Rendering equation

- Most rendering methods require solving an (approximation) of the rendering equation:

Incident light

Integral over the hemisphere of incident light

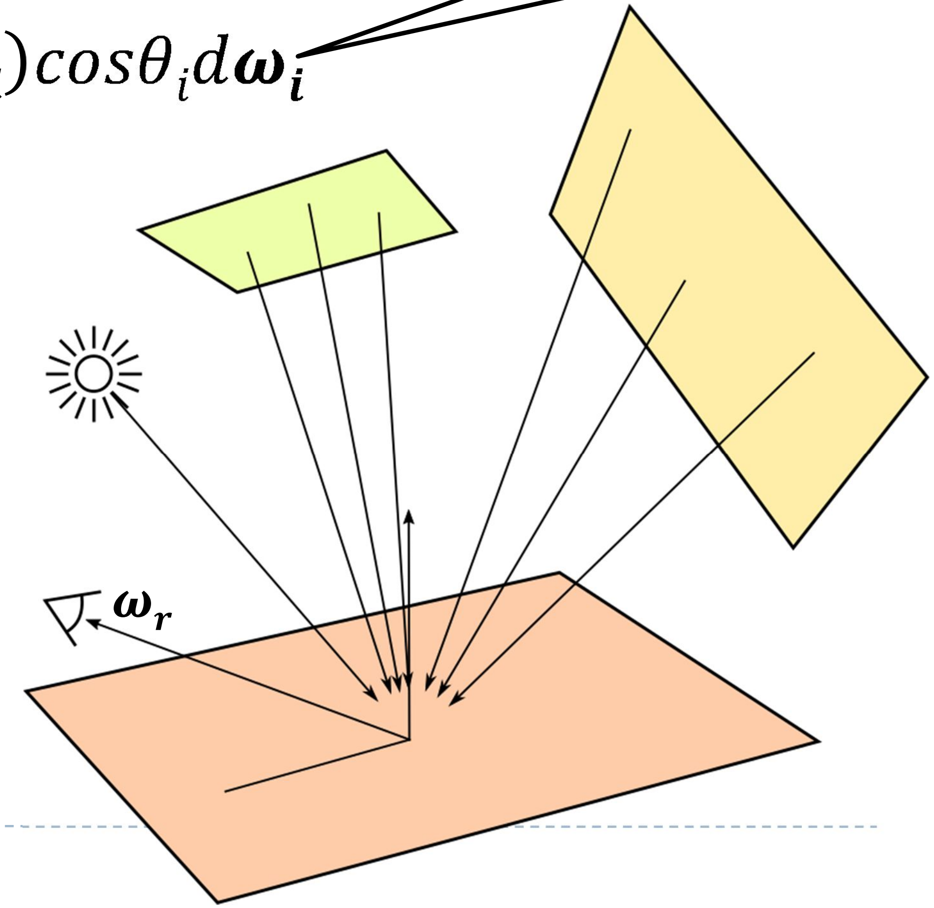
$$L_r(\boldsymbol{\omega_r}) = \int_{\Omega} \rho(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) L_i(\boldsymbol{\omega_i}) cos\theta_i d\boldsymbol{\omega_i}$$

Reflected light

BRDF

- The solution is trivial for point light sources
- Much harder to estimate the contribution of other surfaces

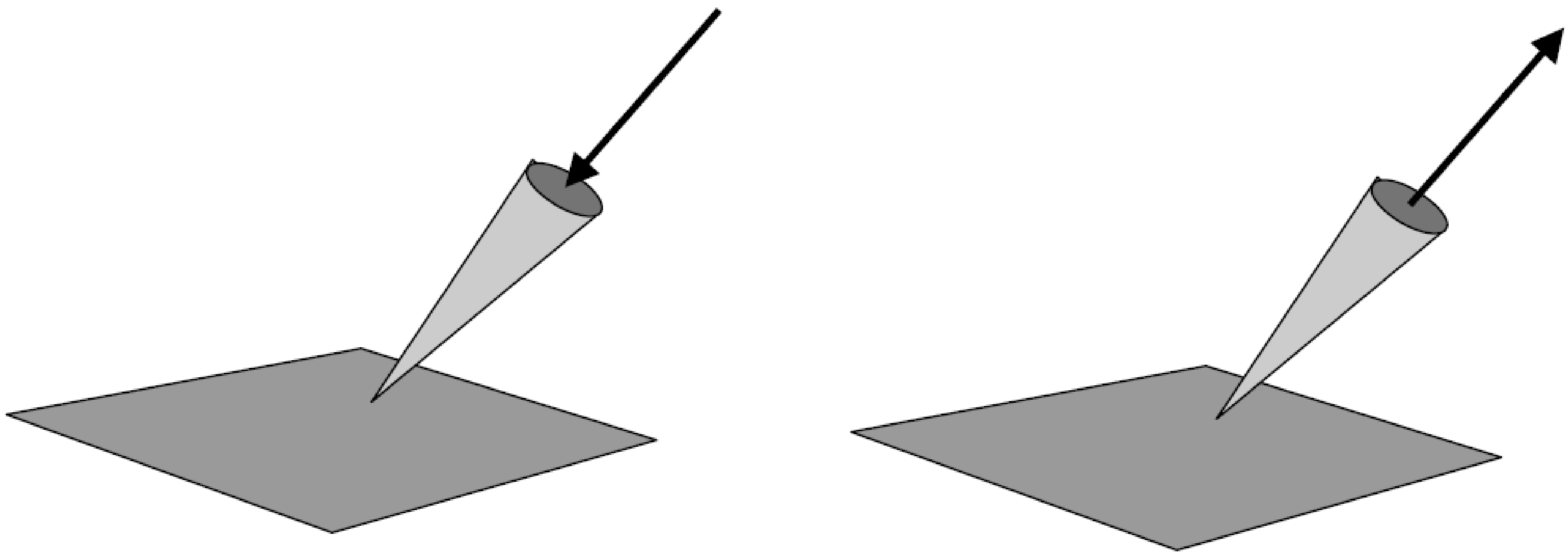
$$\boldsymbol{\omega_i} = [\phi_i, \theta_i]$$



Radiance

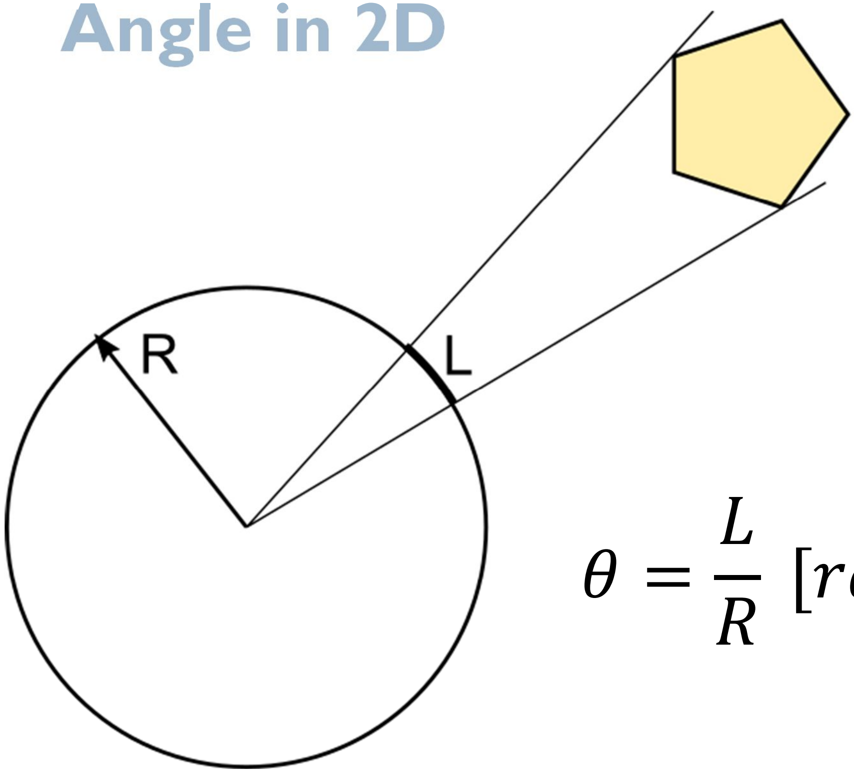
- Power of light per unit projected area per unit solid angle
- Symbol: $L(\boldsymbol{x}, \boldsymbol{\omega}_i)$
- Units: $\dfrac{W}{m^2 sr}$

Steradian



Solid angle

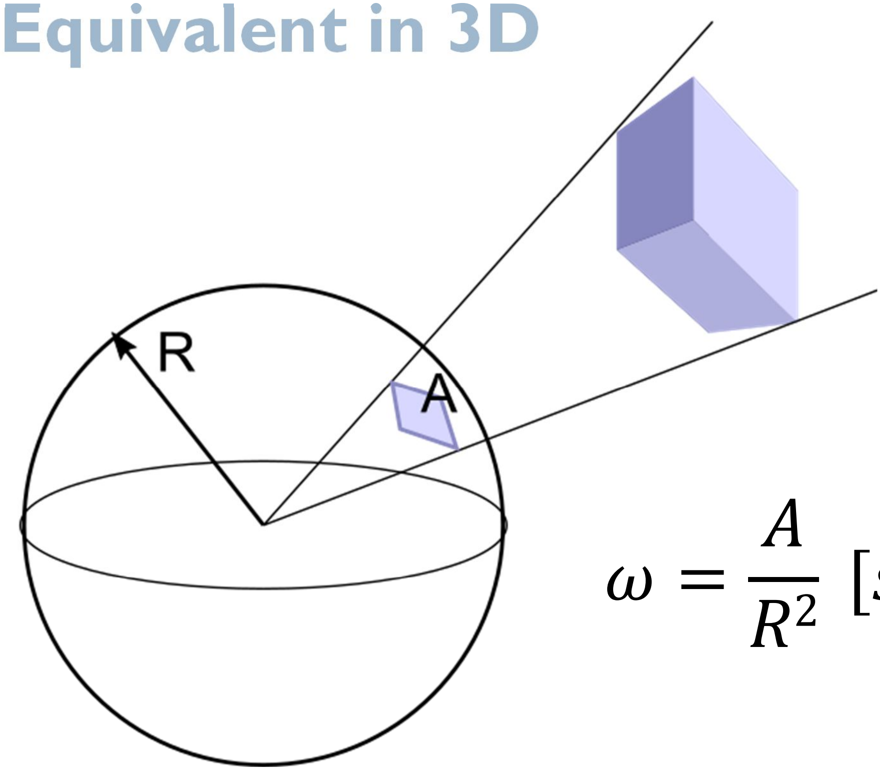
Angle in 2D



$$\theta = \frac{L}{R}\ [rad]$$

Full circle = 2π radians

Equivalent in 3D



$$\omega = \frac{A}{R^2}\ [sr]$$

- Full sphere = 4π steradians

Radiance

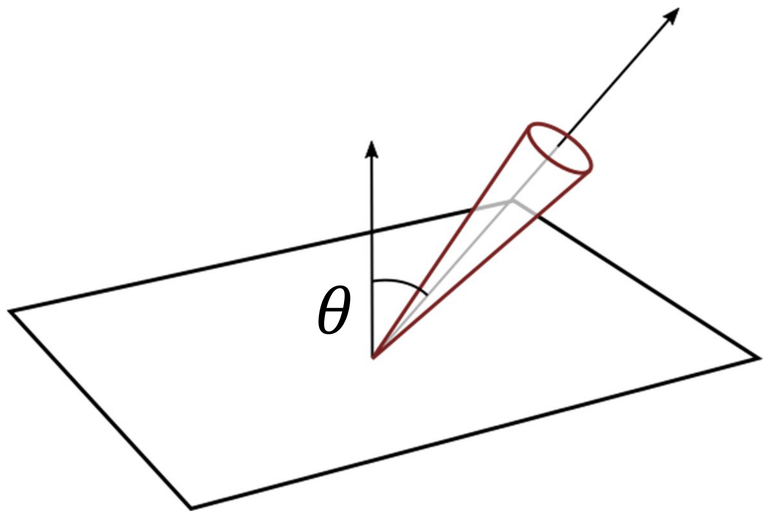
Position

Power of light

$$L(\boldsymbol{x}, \boldsymbol{\omega}_i) = \frac{d\phi}{d\omega \; dA \; cos\theta}$$

Incoming direction

Projected area

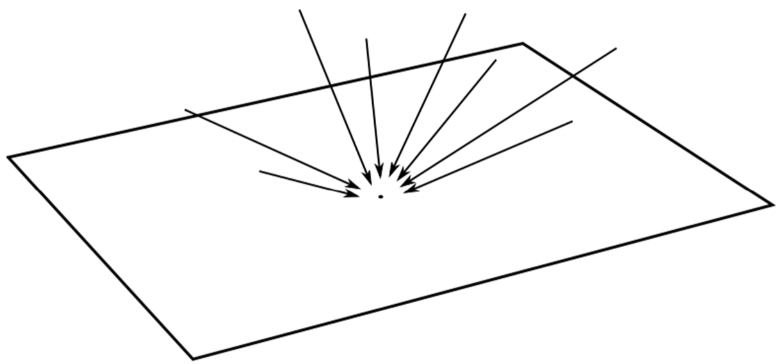


- Power per solid angle per projected surface area
- Invariant along the direction of propagation (in vacuum)
- Response of a camera sensor or a human eye is related to radiance
- Pixel values in image are related to radiance (projected along the view direction)

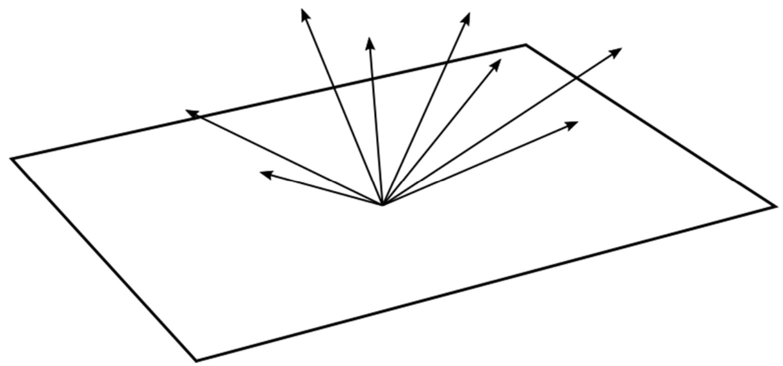
Irradiance and Exitance

- Power per unit area
- Irradiance: H(x) – incident power per unit area
- Exitance / radiosity: E(x) – exitant power per unit area
- Units: $\frac{W}{m^2}$

Irradiance



Exitance / Radiosity



Relation between Irradiance and Radiance

- Irradiance is an integral over all incoming rays
  - Integration over a hemisphere Ω:

$$H = \int_{\Omega} L(\boldsymbol{x}, \boldsymbol{\omega_i})\ cos\theta\ d\boldsymbol{\omega}$$

  - In the spherical coordinate system, the differential solid angle is:

$$d\boldsymbol{\omega} = \sin\theta d\theta\ d\phi$$

  - Therefore:

$$H = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} L(\boldsymbol{x}, \boldsymbol{\omega_i})\ cos\theta\ sin\theta\ d\theta\ d\phi$$

  - For constant radiance:

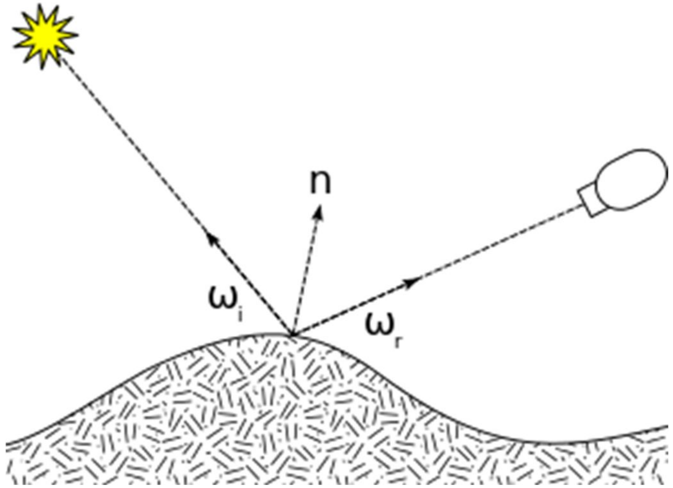
$$H = \pi L$$

BRDF: Bidirectional Reflectance Distribution Function

Differential radiance of reflected light

$$\rho(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) = \frac{dL_r(\boldsymbol{\omega_r})}{dH_i(\boldsymbol{\omega_i})} = \frac{dL_r(\boldsymbol{\omega_r})}{L_i(\boldsymbol{\omega_i})cos\theta_i d\boldsymbol{\omega_i}}$$

Differential irradiance of incoming light

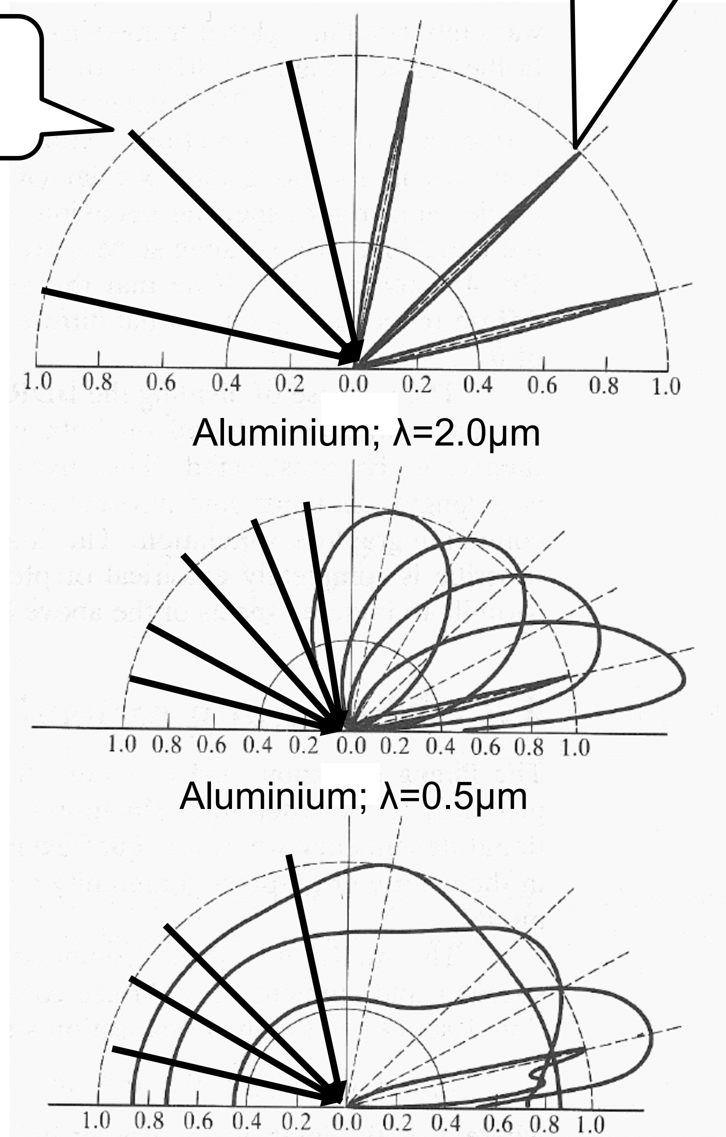


Source: Wikipedia

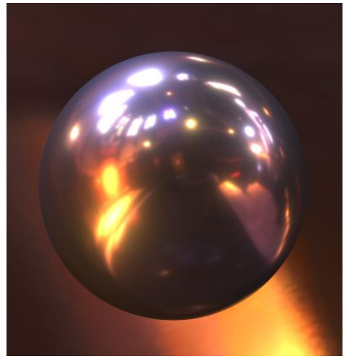
- BRDF is measured as a ratio of reflected radiance to irradiance
  - Because it is difficult to measure $L_i(\boldsymbol{\omega_i})$, BRDF is not defined as the ratio $L_r(\boldsymbol{\omega_r})/L_i(\boldsymbol{\omega_i})$

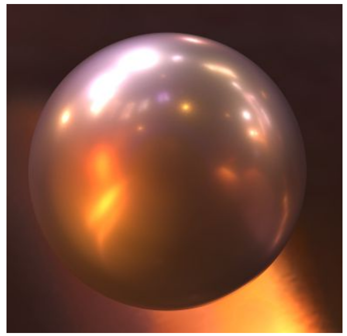
BRDF of various materials

- The diagrams show the distribution of reflected light for the given incoming direction
- The material samples are close but not accurate matches for the diagrams

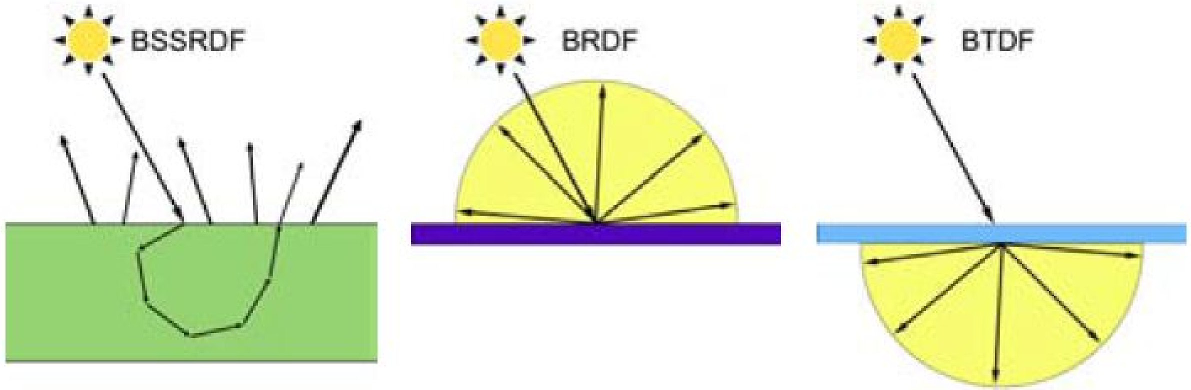


Magnesium alloy; λ=0.5μm





Other material models

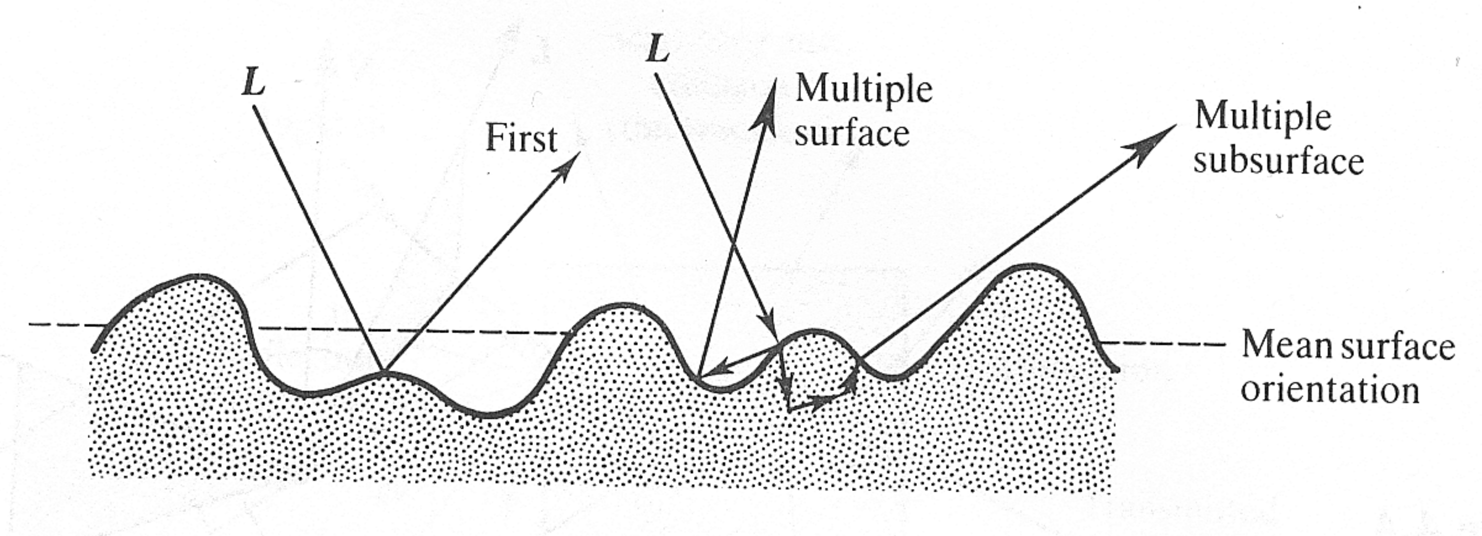


Source:
Guarnera et al. 2016

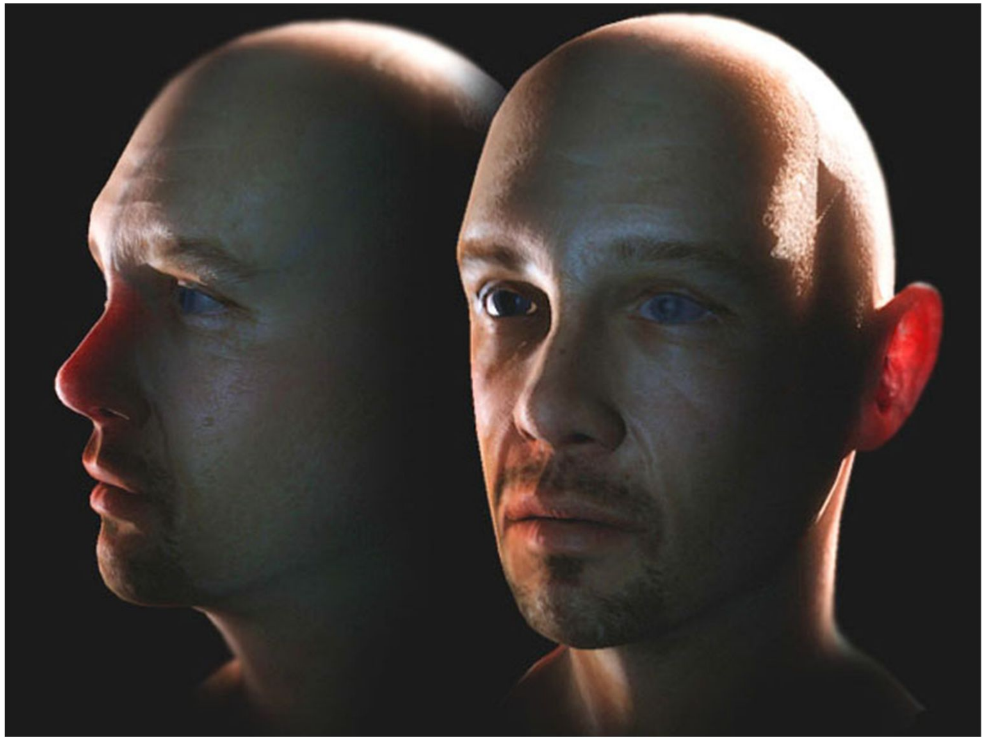
- Bidirectional Scattering Surface Reflectance Distribution F.
- **Bidirectional Reflectance Distribution Function**
- Bidirectional Transfer Distribution Function
- But also: BTF, SVBRDF, BSDF
- In this lecture we will focus mostly on BRDF

Sub-surface scattering

- Light enters material and is scattered several times before it exits
  - Examples - human skin: hold a flashlight next to your hand and see the color of the light
- The effect is expensive to compute
  - But approximate methods exist



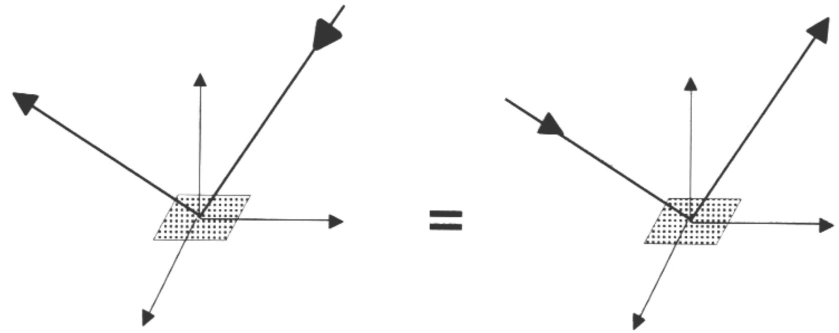
Subsurface scattering - examples



BRDF Properties

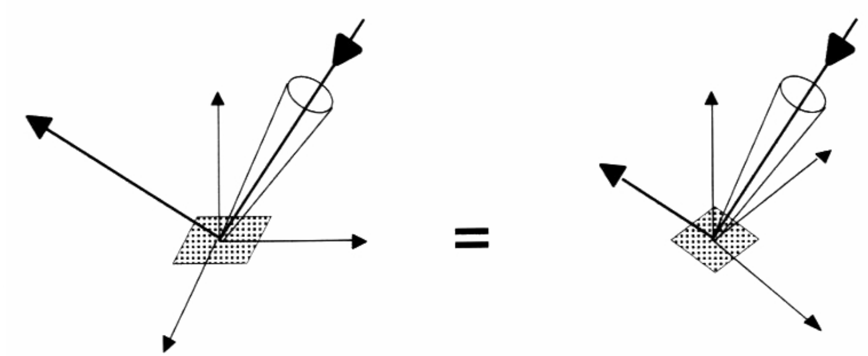
- Helmholtz reciprocity principle
  - BRDF remains unchanged if incident and reflected directions are interchanged

$$\rho(\boldsymbol{\omega_r}, \boldsymbol{\omega_i}) = \rho(\boldsymbol{\omega_i}, \boldsymbol{\omega_r})$$



- Smooth surface: isotropic BRDF
  - reflectivity independent of rotation around surface normal
  - BRDF has only 3 instead of 4 directional degrees of freedom

$$\rho(\theta_i, \theta_r, \phi_r - \phi_i)$$



BRDF Properties

- Characteristics
  - BRDF units [1/sr]
    - Not intuitive
  - Range of values:
    - From 0 (absorption) to ∞ (reflection, δ -function)
  - Energy conservation law

$$\int_{\Omega} \rho(\omega_r, \omega_i) cos\theta_i d\omega_i \leq 1$$

    - No self-emission
    - Possible absorption
  - Reflection only at the point of entry $(x_i = x_r)$
    - No subsurface scattering

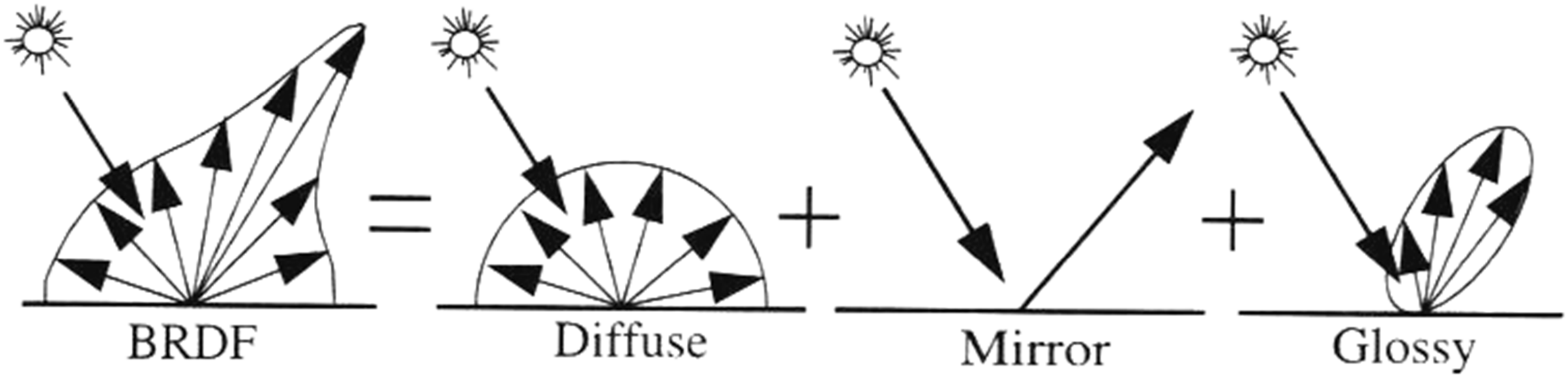
BRDF Measurement

- Gonio-Reflectometer
- BRDF measurement
  - point light source position (θ, φ)
  - light detector position (θ_o, φ_o)
- 4 directional degrees of freedom
- BRDF representation
  - m incident direction samples (θ, φ)
  - n outgoing direction samples (θ_o, φ_o)
  - mn reflectance values (large!!!)

Stanford light gantry

BRDF Modeling

- It is common to split BRDF into diffuse, mirror and glossy components
- Ideal diffuse reflection
  - Lambert's law
  - Matte surfaces
- Ideal specular reflection
  - Reflection law
  - Mirror
- Glossy reflection
  - Directional diffuse
  - Shiny surfaces

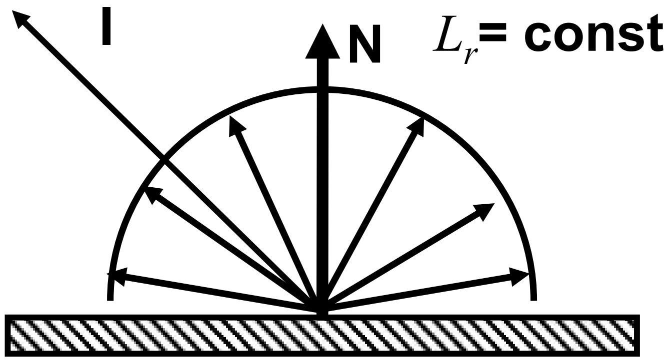


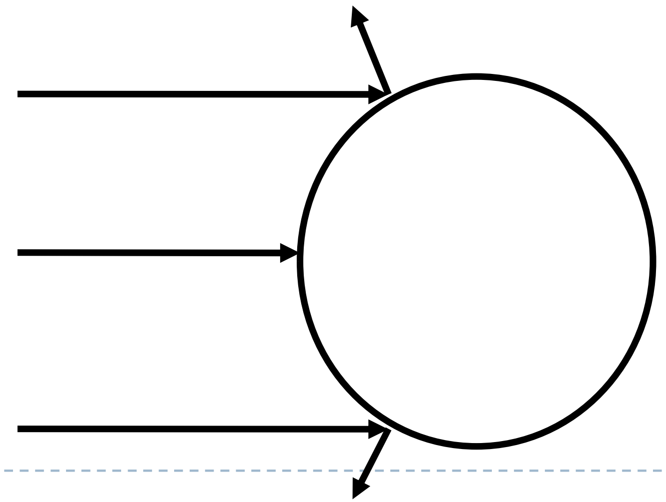
Diffuse Reflection

- Light equally likely to be reflected in any outgoing direction (independent of incident direction)
- Constant BRDF $\rho(\omega_r, \omega_i) = k_d = const$

$$L_r(\omega_r) = \int_{\Omega} k_d L_i(\omega_i) cos\theta_i d\omega_i = k_d \int_{\Omega} L_i(\omega_i) \cos\theta_i d\omega_i = k_d H$$

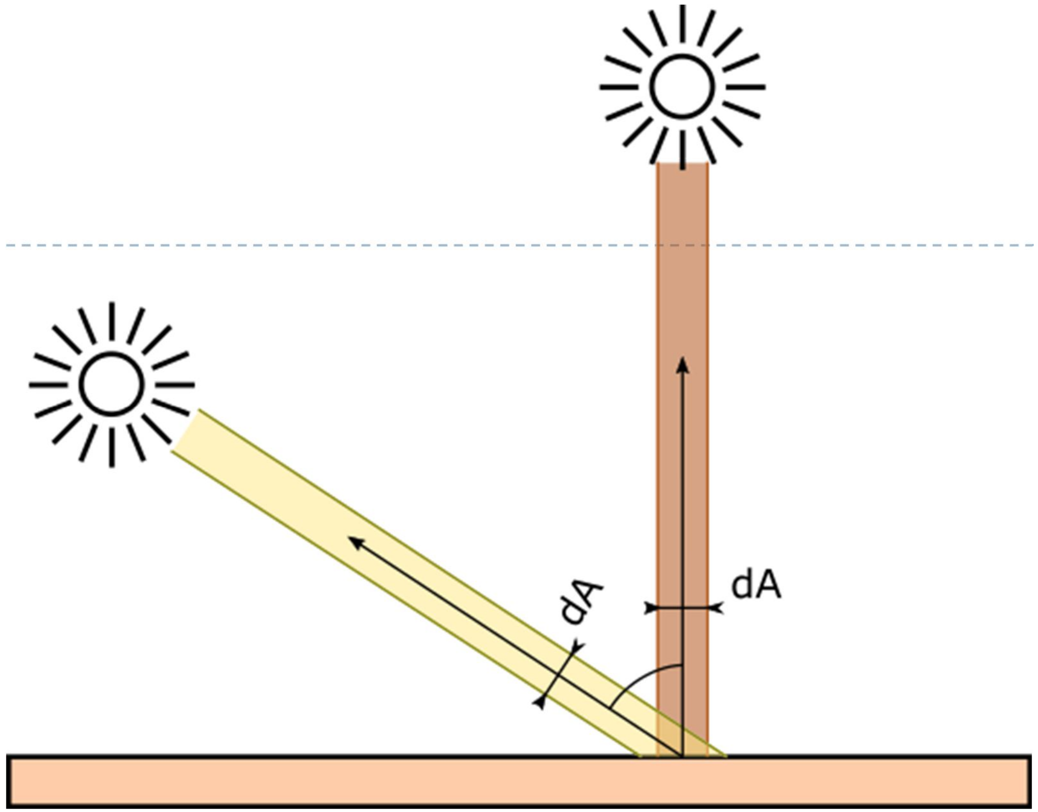
  - k_d: diffuse coefficient, material property [1/sr]



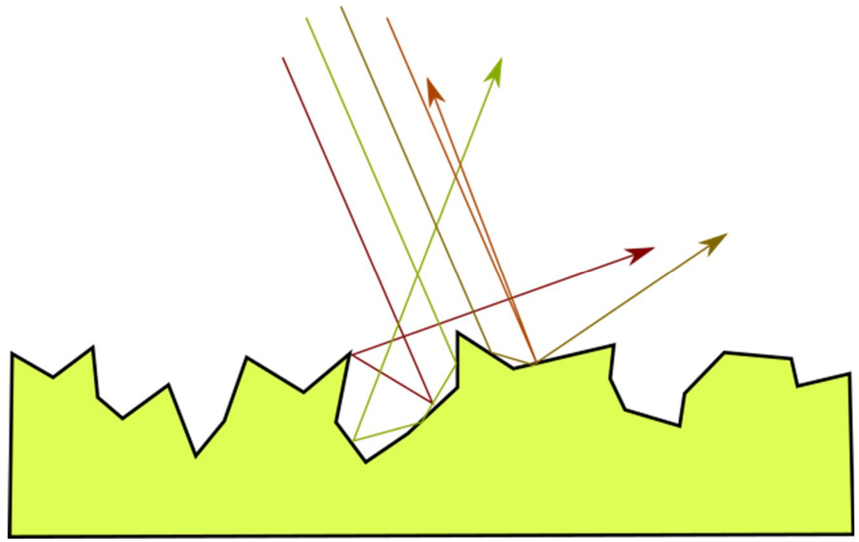


Diffuse reflection

- Cosine term
  - The surface receives less light per unit area at steep angles of incidence



- Rough, irregular surface results in diffuse reflection
  - Light is reflected in random direction
  - (this is just a model, light interaction is more complicated)

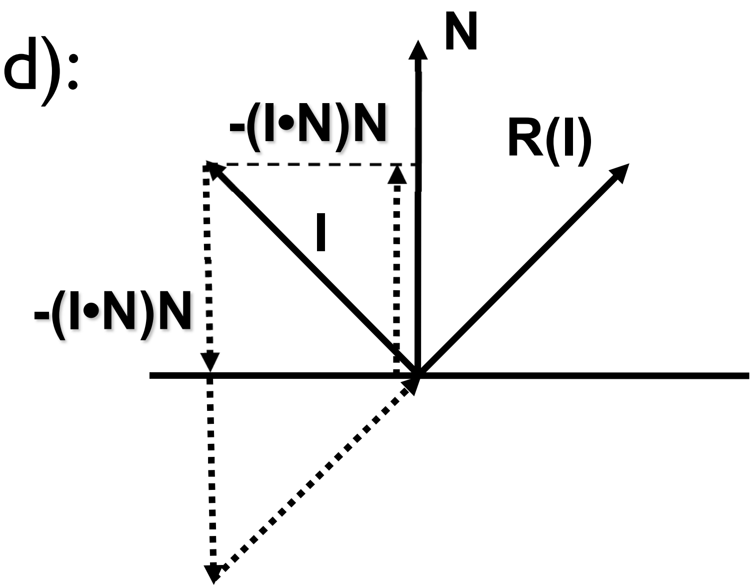


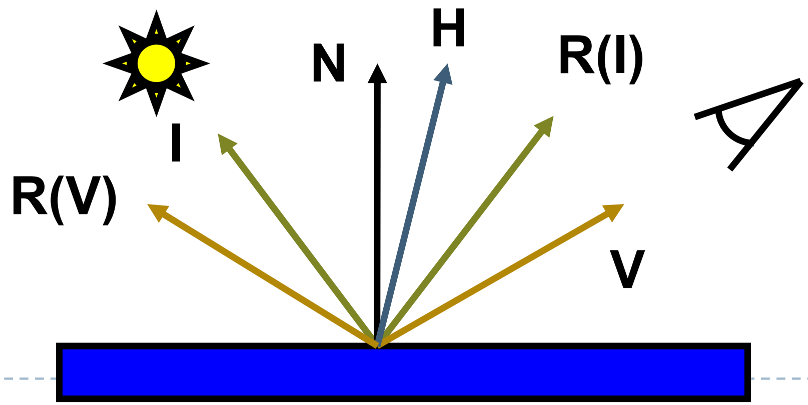
Reflection Geometry

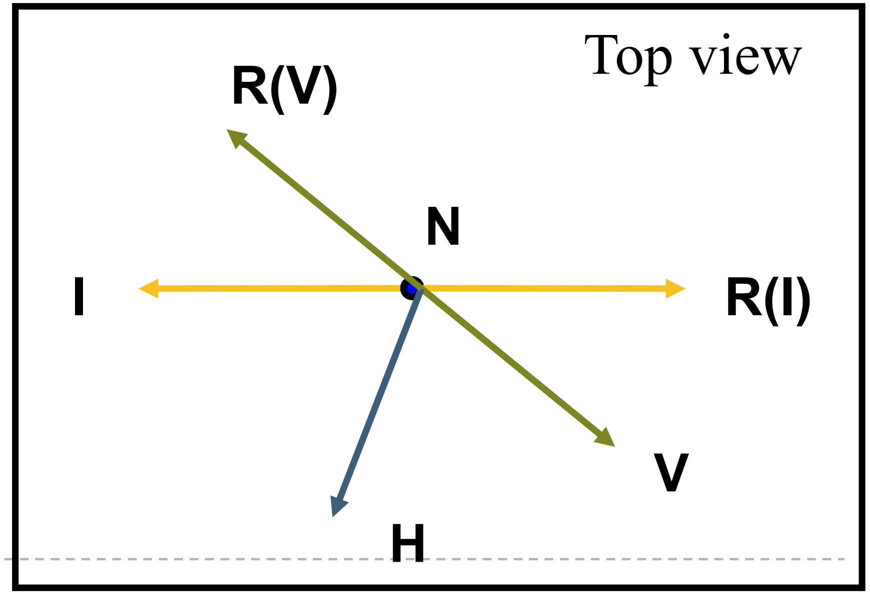
- Direction vectors (all normalized):
  - N: surface normal
  - I: vector to the light source
  - V: viewpoint direction vector
  - H: halfway vector

    H= (I + V) / |I + V|
  - R(I): reflection vector

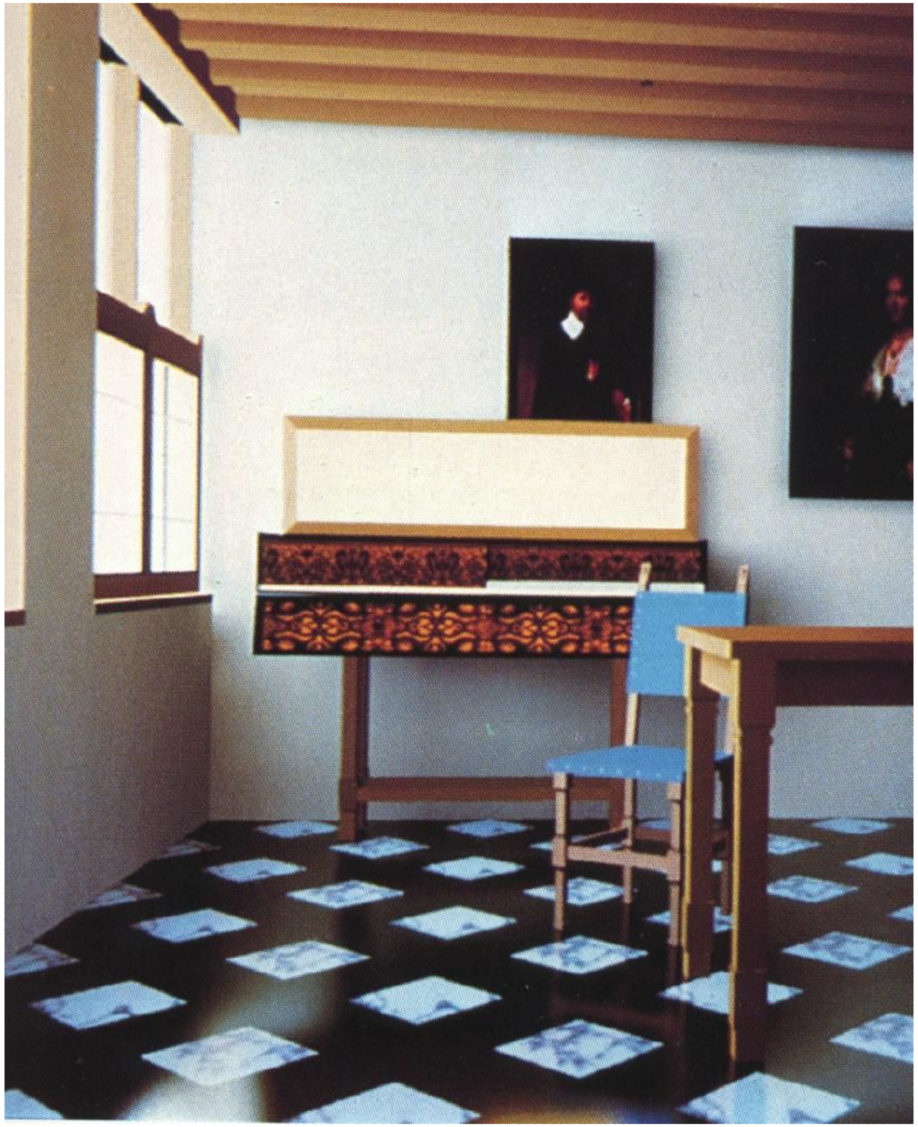
    R(I)= I - 2(I•N)N
  - Tangential surface: local plane





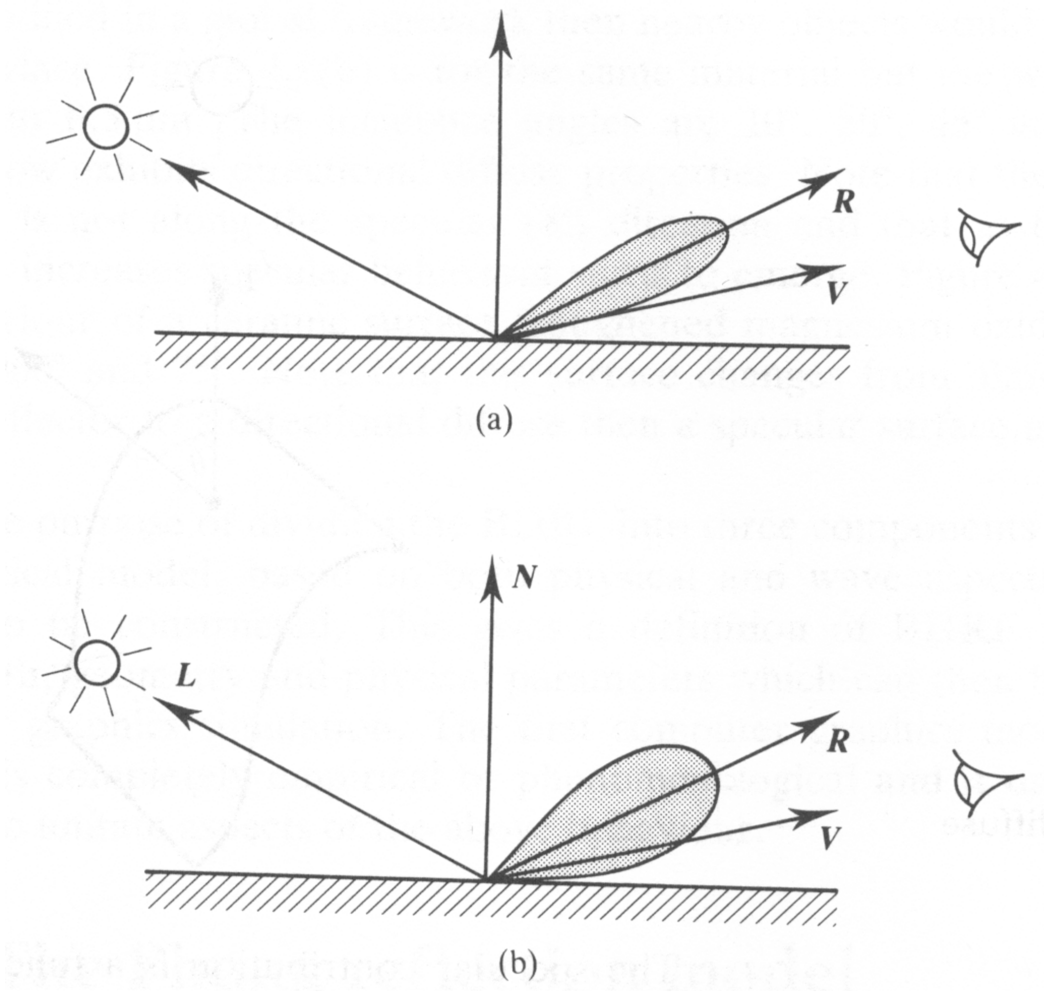


Glossy Reflection



Glossy Reflection

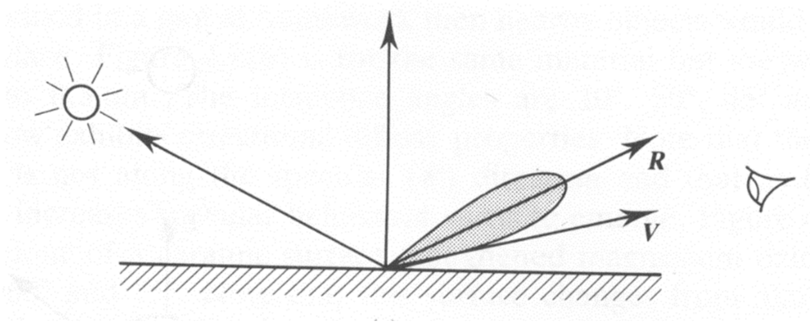
- Due to surface roughness
- Empirical models
  - Phong
  - Blinn-Phong
- Physical models
  - Blinn
  - Cook & Torrance



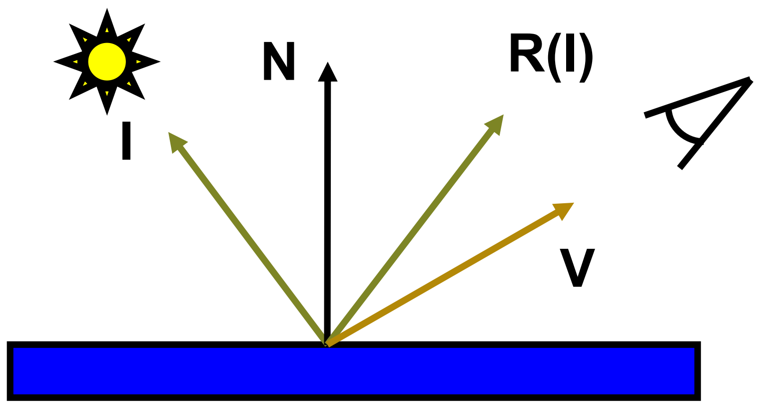
Phong Reflection Model

- Cosine power lobe

$$\rho(\omega_r, \omega_i) = k_s (R \cdot V)^{k_e}$$



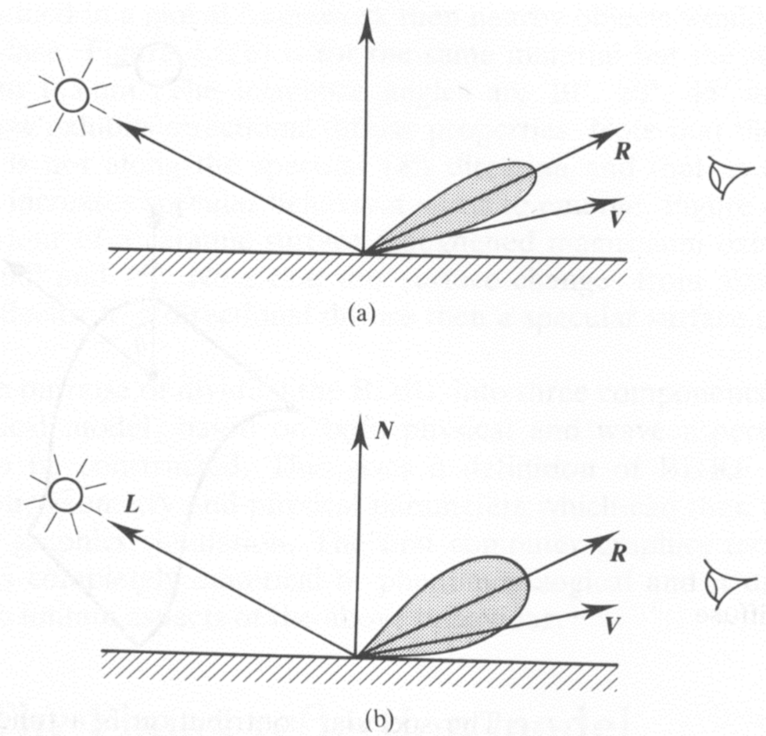
- Dot product & power
- Not energy conserving/reciprocal
- Plastic-like appearance

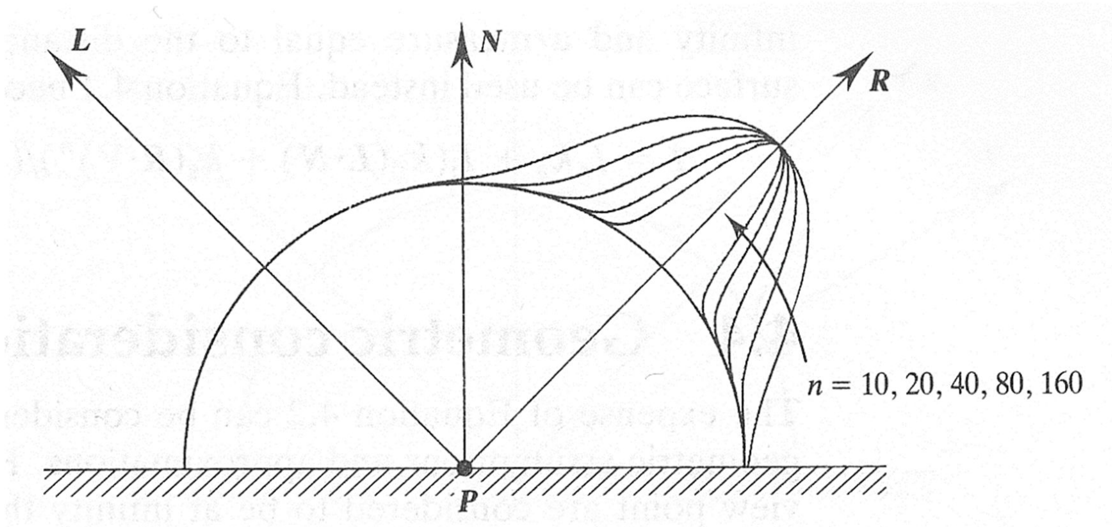


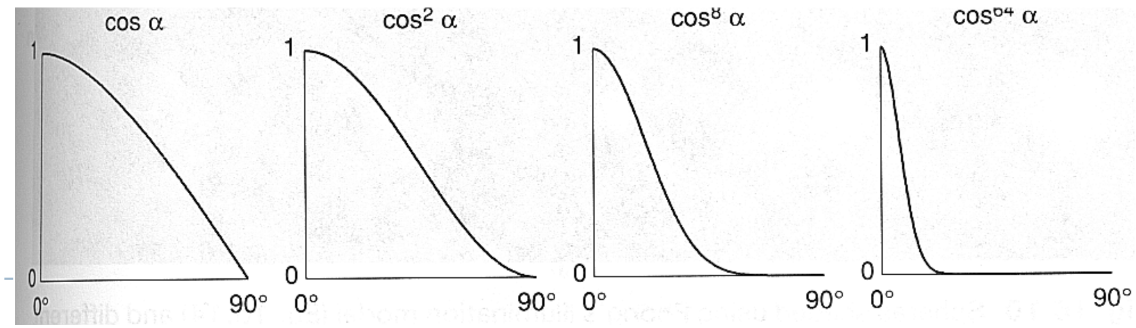
Phong Exponent k_e

$$\rho(\omega_r, \omega_i) = k_s (R \cdot V)^{k_e}$$

- Determines the size of highlight







Phong Illumination Model

- Extended light sources: l point light sources

$$L_r = k_a L_{i,a} + k_d \sum_l L_l (I_l \cdot N) + k_s \sum_l L_l (R(I_l) \cdot V)^{k_e} \quad \text{(Phong)}$$

$$L_r = k_a L_{i,a} + k_d \sum_l L_l (I_l \cdot N) + k_s \sum_l L_l (H_l \cdot N)^{k_e} \quad \text{(Blinn)}$$

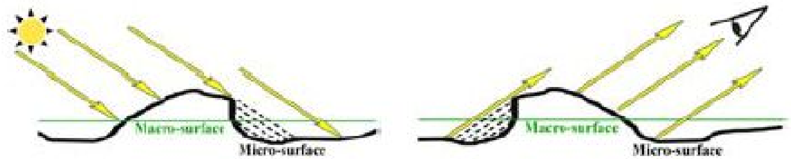
- Colour of specular reflection equal to light source
- Heuristic model
  - Contradicts physics
  - Purely local illumination
    - Only direct light from the light sources
    - No further reflection on other surfaces
    - Constant ambient term
- Often: light sources & viewer assumed to be far away

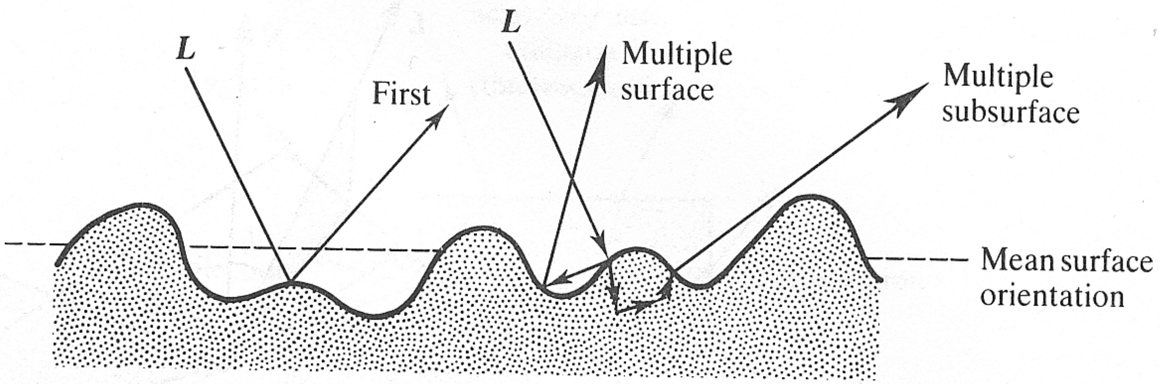
Micro-facet model

- We can assume that at small scale materials are made up of small facets
  - Facets can be described by the distribution of their sizes and directions D
  - Some facets are occluded by other, hence there is also a geometrical attenuation term G
  - And we need to account for Fresnel reflection (see next slides)

$$\rho(\omega_i, \omega_r) = \frac{D \cdot G \cdot F}{4 \cos \theta_i \theta_r}$$

Rendering of glittery materials [Jakob et al. 2014]





Cook-Torrance model

- Can model metals and dielectrics
- Sum of diffuse and specular components

$$\rho(\omega_i, \omega_r) = \rho_d(\omega_i) + \rho_s(\omega_i, \omega_r)$$

- Specular component:

$$\rho_s(\omega_i, \omega_r) = \frac{D(h)\, G(I,V)\, F(\omega_i)}{\pi \cos\theta_i \cos\theta_r}$$

- Distribution of microfacet orientations: $D(h) = \cos\theta_r\, e^{-\left(\frac{\alpha}{m}\right)^2}$

Roughness parameter

- Geometrical attenuation factor
  - To account to self-masking and shadowing

$$G(I,V) = min\left\{1, \frac{2(N\cdot H)(N\cdot V)}{V\cdot H}, \frac{2(N\cdot H)(N\cdot I)}{V\cdot H}\right\}$$

GGX model

- Multiple PBR models have been defined by modifying the definitions of the D and G functions.

$$\rho_s(\omega_i, \omega_r) = \frac{D(h)\ G(I,V)\ F(\omega_i)}{\pi \cos\theta_i \cos\theta_r}$$

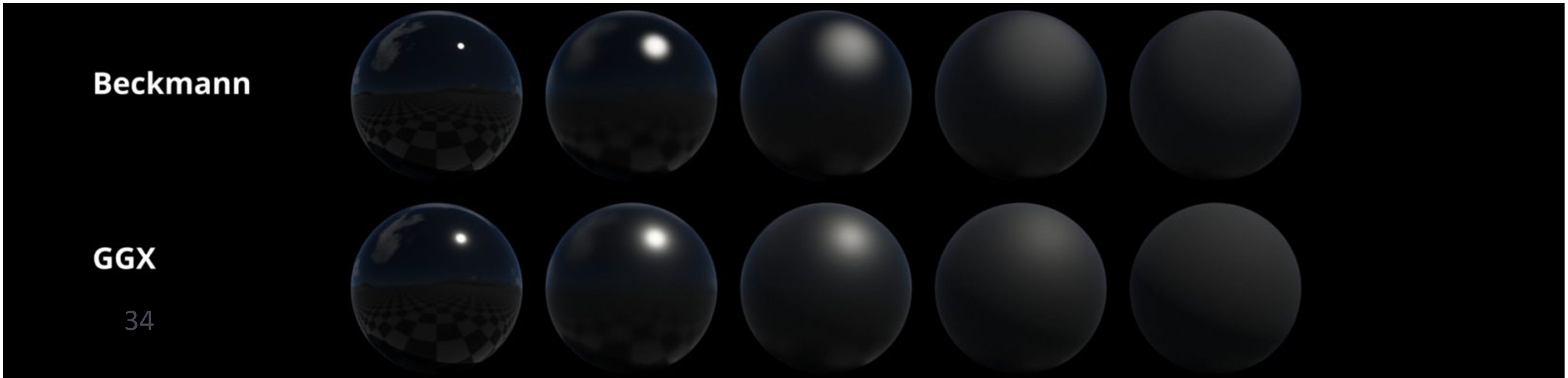
- Distribution of microfacet orientations:

$$D_{\mathrm{GGX}}(\vec{h}) = \frac{\alpha^2}{\pi\left[(\vec{h}\cdot\hat{n})^2(\alpha^2-1)+1\right]^2}$$

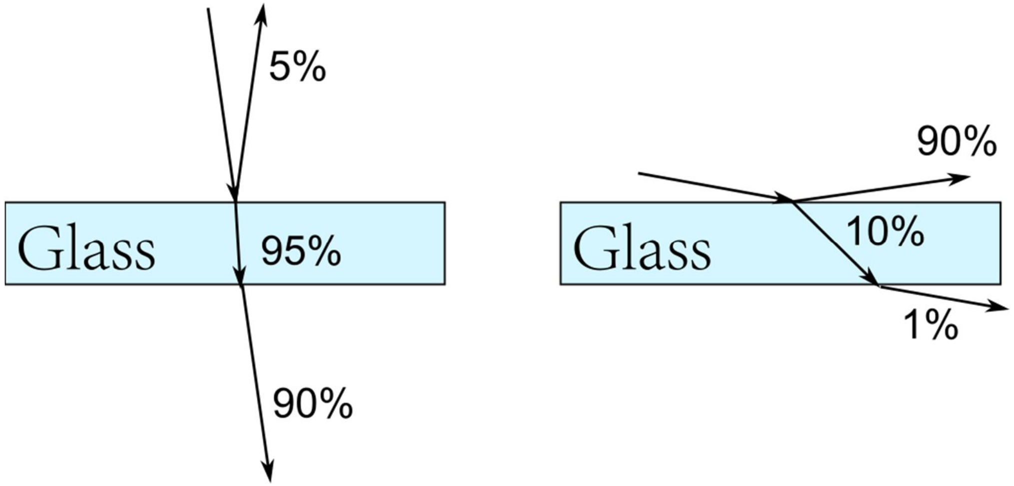
Roughness parameter

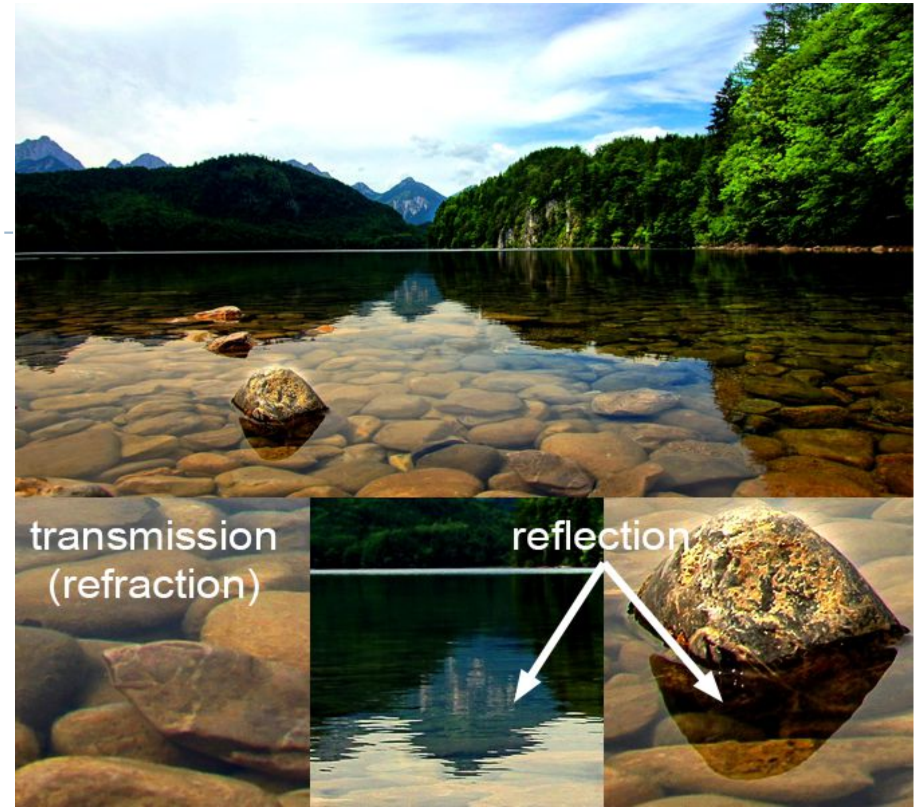
- More computationally efficient than Beckmann
- More realistic, especially high roughness materials
- Longer tails (higher intensity reflections at grazing angles)
- Currently used by most real-time renderers

Source: https://planetside.co.uk/news/terragen-4-5-releas



Fresnel term





Example from: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel

- The light is more likely to be reflected rather than transmitted near grazing angles
- The effect is modelled by Fresnel equation: it gives the probability that a photon is reflected rather than transmitted (or absorbed)

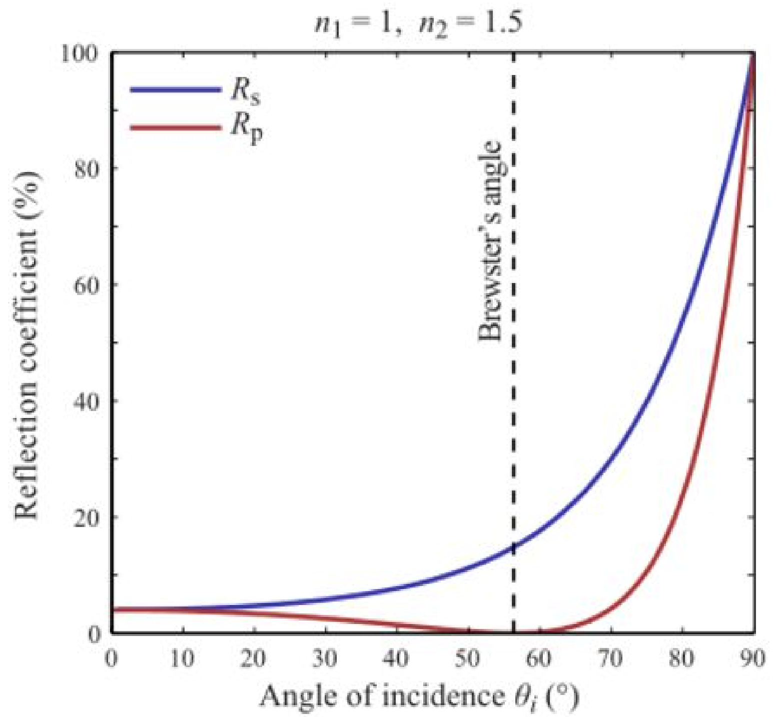
Fresnel equations

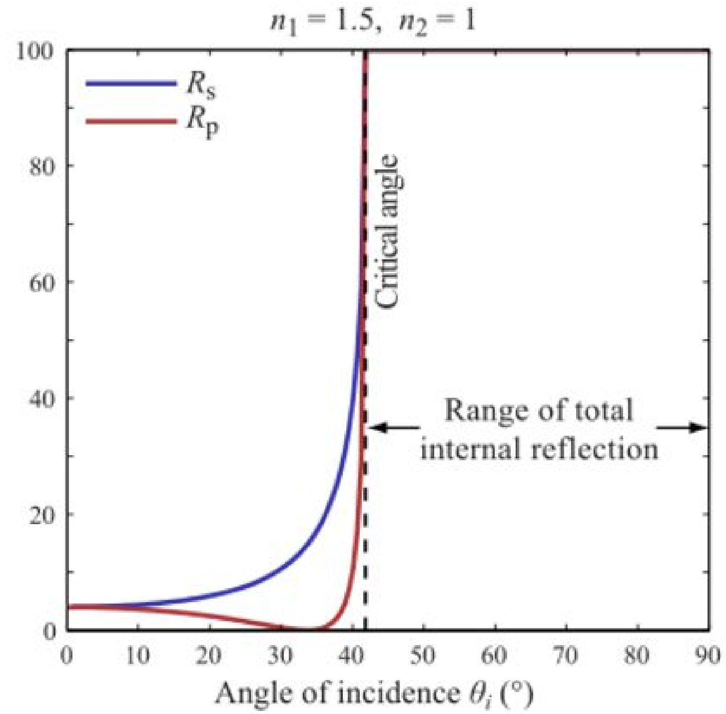
- Reflectance for s-polarized light:

$$R_s = \left|\frac{n_1\cos\theta_i - n_2\cos\theta_t}{n_1\cos\theta_i + n_2\cos\theta_t}\right|^2 = \left|\frac{n_1\cos\theta_i - n_2\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}{n_1\cos\theta_i + n_2\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}\right|^2,$$

- Reflectance for p-polarized light:

$$R_p = \left|\frac{n_1\cos\theta_t - n_2\cos\theta_i}{n_1\cos\theta_t + n_2\cos\theta_i}\right|^2 = \left|\frac{n_1\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2} - n_2\cos\theta_i}{n_1\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2} + n_2\cos\theta_i}\right|^2.$$





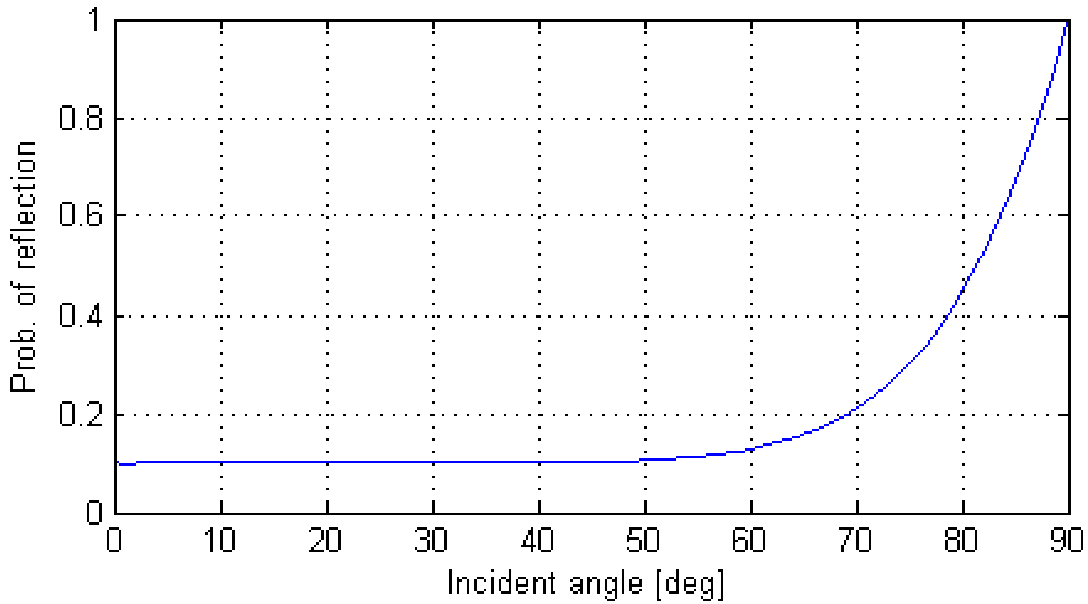
Fresnel term

- In Computer Graphics the Fresnel equation is approximated by Schlick's formula [Schlick, 94]:

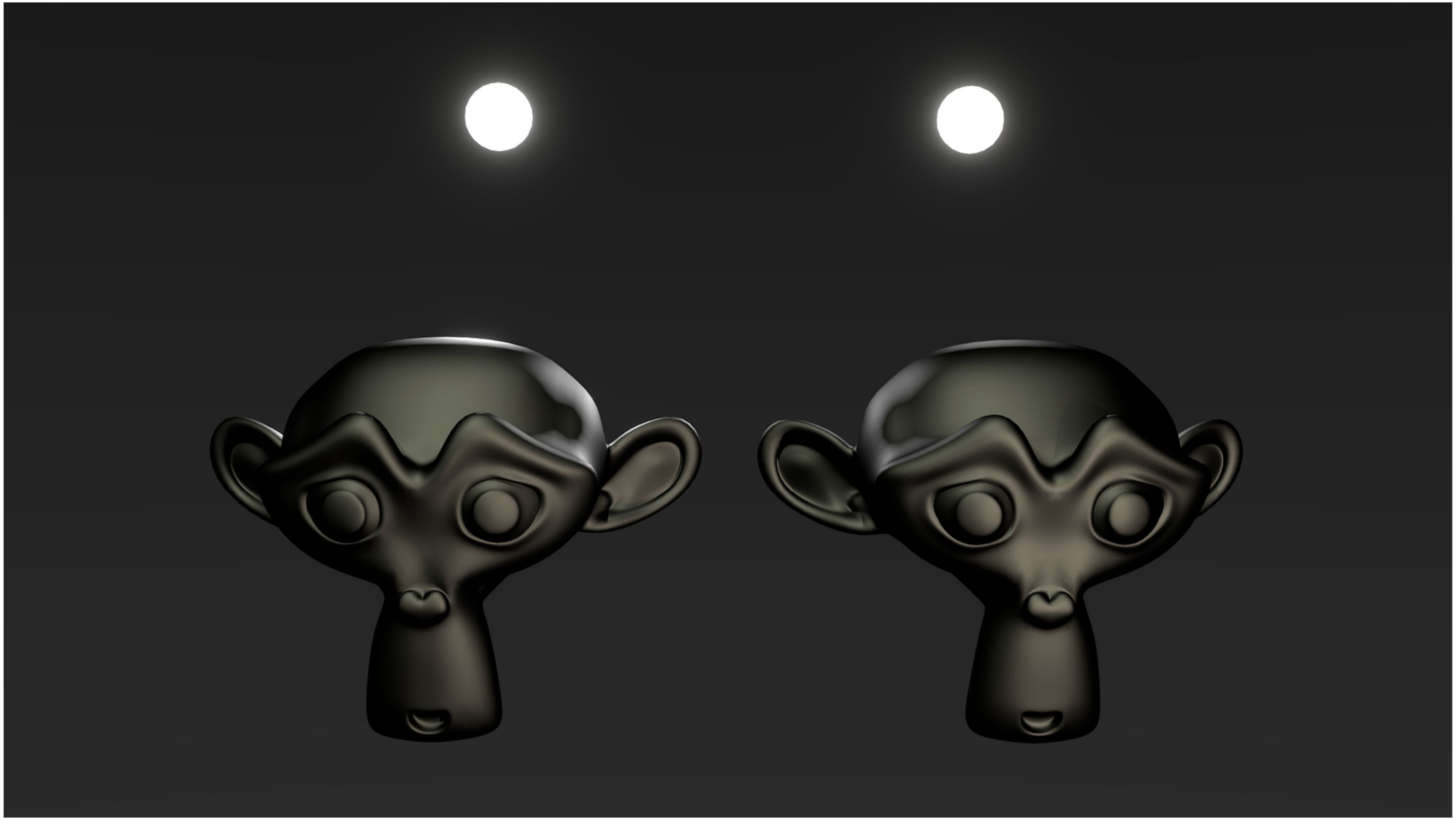
$$R(\theta,\lambda) = R_0(\lambda) + \big(1 - R_0(\lambda)\big)(1 - cos\theta)^5$$

- where $R_0(\lambda)$ is reflectance at normal incidence and λ is the wavelength of light
- For dielectrics (such as glass):

$$R_0(\lambda) = \left(\frac{n(\lambda) - 1}{n(\lambda) + 1}\right)^2$$

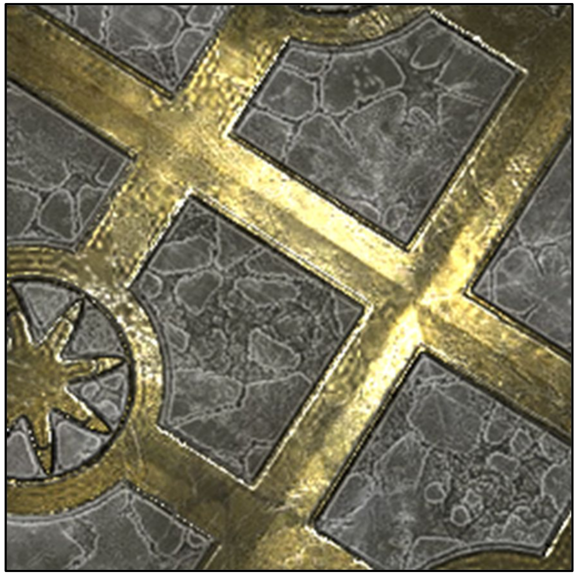


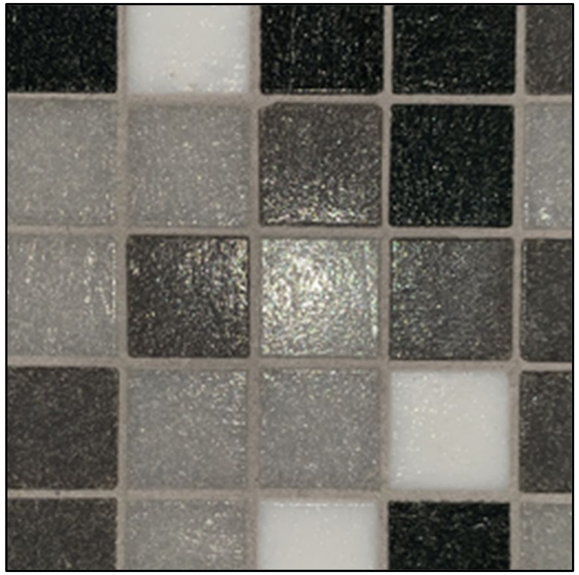
Which one is Phong / Cook-Torrance ?



Spatially-Varying Materials

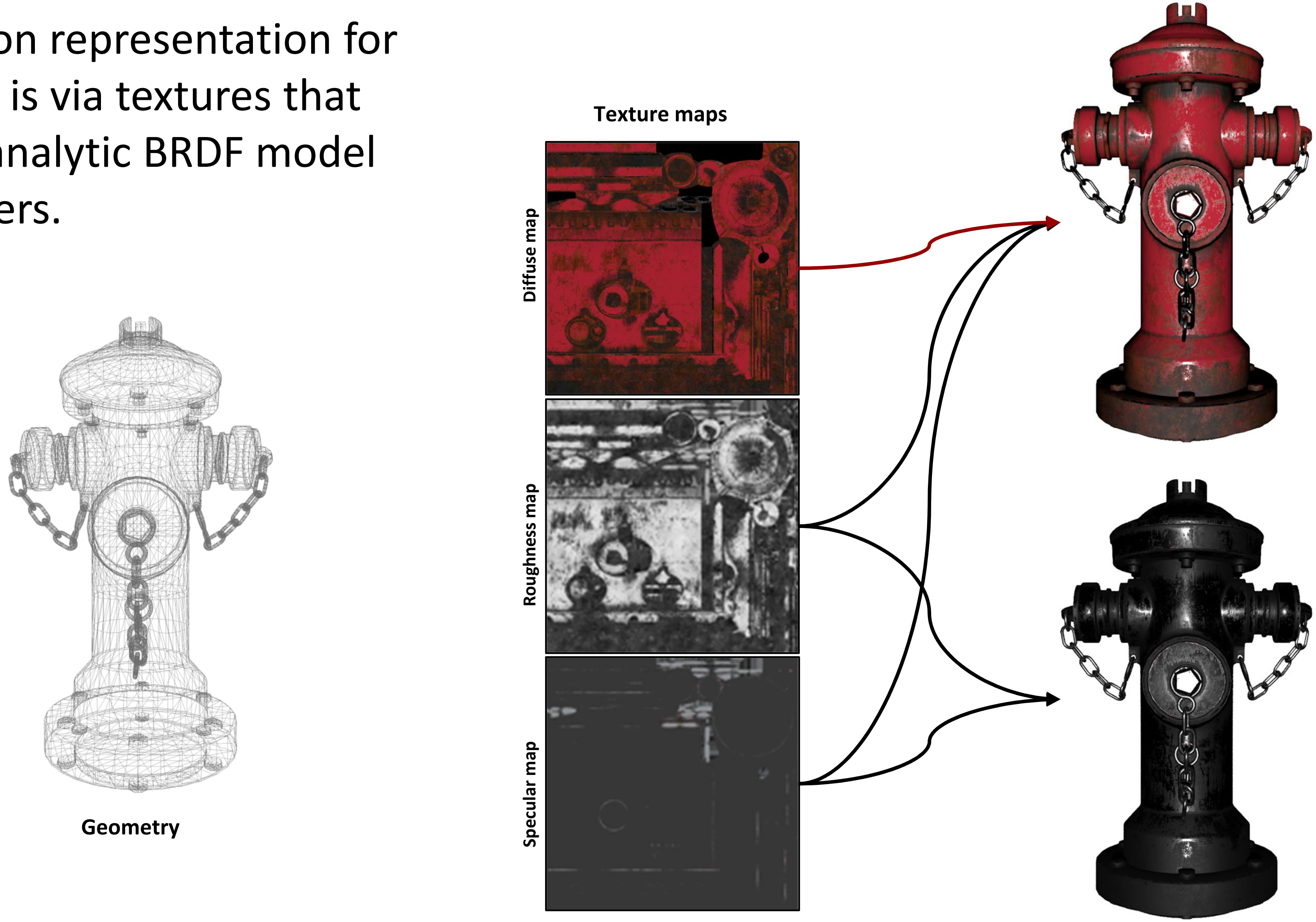
- In spatially-varying materials the reflectance has an additional dependence on the position over the material surface: $f_r(\vec{\omega}_i, \vec{\omega}_o, \vec{x})$





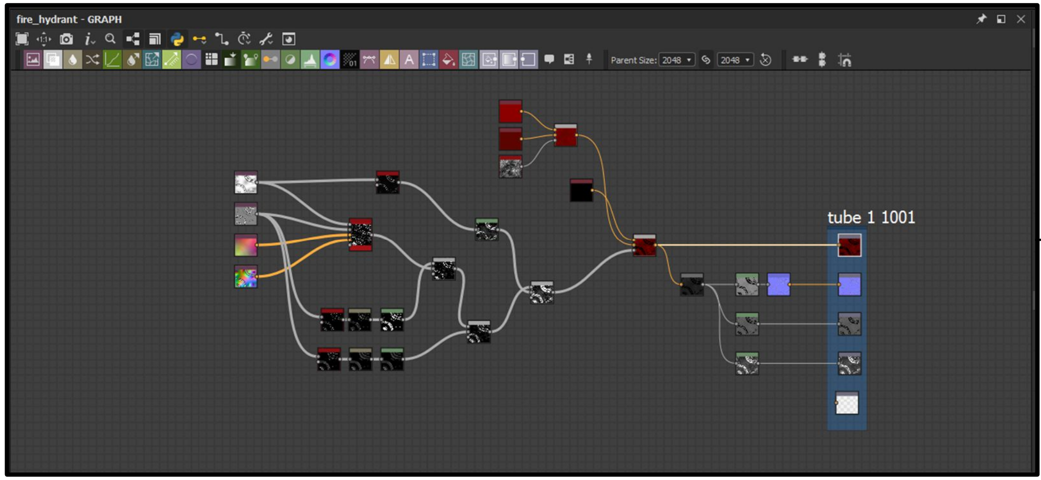
Spatially-Varying Materials

A common representation for SVBRDFs is via textures that encode analytic BRDF model parameters.



Spatially-Varying Materials

- In most commercial renderers, SVBRDFs are designed through procedural graphs. This gives the user great editing flexibility.
- Texture map representations can also be used as input or output of the graph (baking).



Source: Blender.

Image based lighting (IBL)

1. Capture an HDR image of a light probe

2. Create an illumination (cube) map

3. Use the illumination map as a source of light in the scene

The scene is surrounded by a cube map

Source: Image-based lighting, Paul Debevec, HDR Symposium 2009


Nick Bertke


Nick Bertke
pogo767@gmail.com


Nick Bertke
pogo767@gmail.com

Blender monkeys + IBL (path tracing)

Further reading

- A. Watt, 3D Computer Graphics
  - Chapter 7: Simulating light-object interaction: local reflection models
- Matt Pharr, Wenzel Jakob, Greg Humphreys, "Physically Based Rendering From Theory to Implementation" (2017)
- Eurographics 2016 tutorial
  - D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, and M. Glencross
  - BRDF Representation and Acquisition
  - **DOI:** 10.1111/cgf.12867
- Some slides have been borrowed from Computer Graphics lecture by Hendrik Lensch
  - http://resources.mpi-inf.mpg.de/departments/d4/teaching/ws200708/cg/slides/CG07-Brdf+Texture.pdf

Advanced Graphics & Image Processing

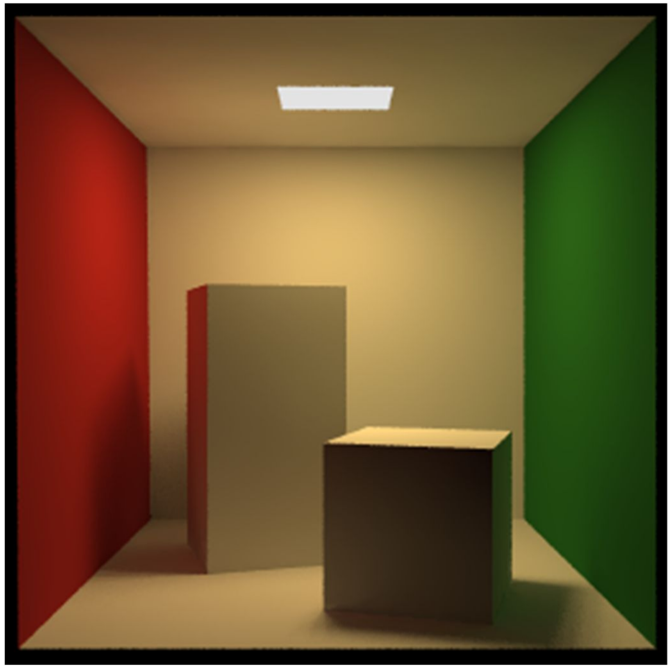
Global Illumination

Rafał Mantiuk

Computer Laboratory, University of Cambridge

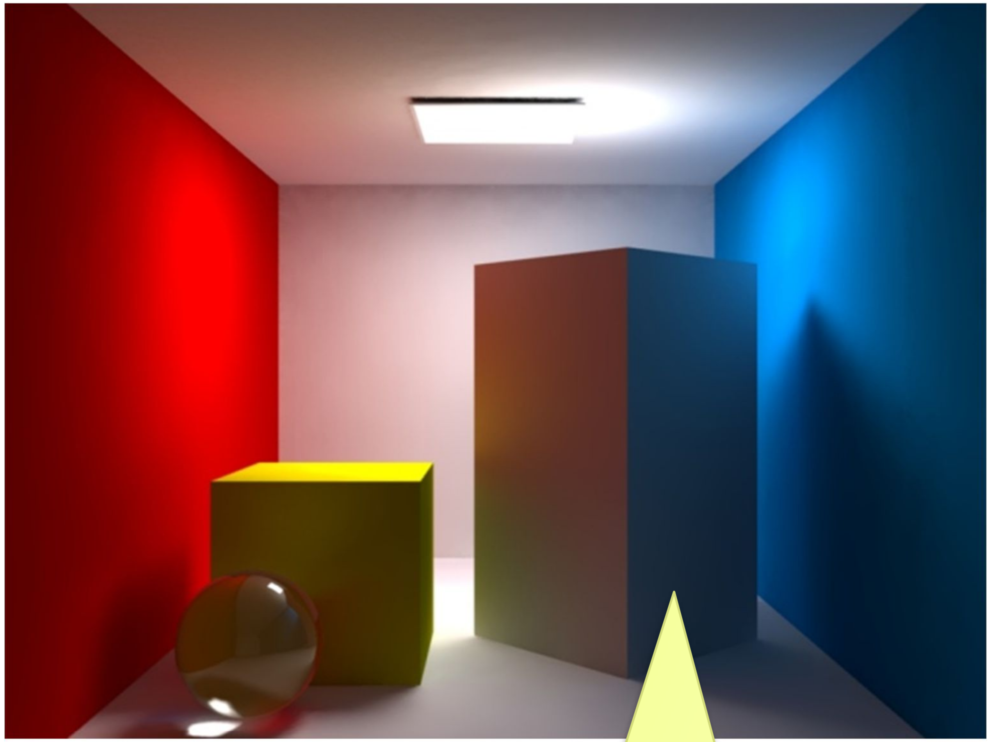
What's wrong with recursive raytracing?

- Soft shadows are expensive
- Shadows of transparent objects require further coding or hacks
- Lighting off reflective objects follows different shadow rules from normal lighting
- Hard to implement diffuse reflection (color bleeding, such as in the Cornell Box—notice how the sides of the inner cubes are shaded red and green)
- Fundamentally, the ambient term is a hack and the diffuse term is only one step in what should be a recursive, self-reinforcing series.



The *Cornell Box* is a test for rendering Software, developed at Cornell University in 1984 by Don Greenberg. An actual box is built and photographed; an identical scene is then rendered in software and the two images are compared.

Global illumination examples



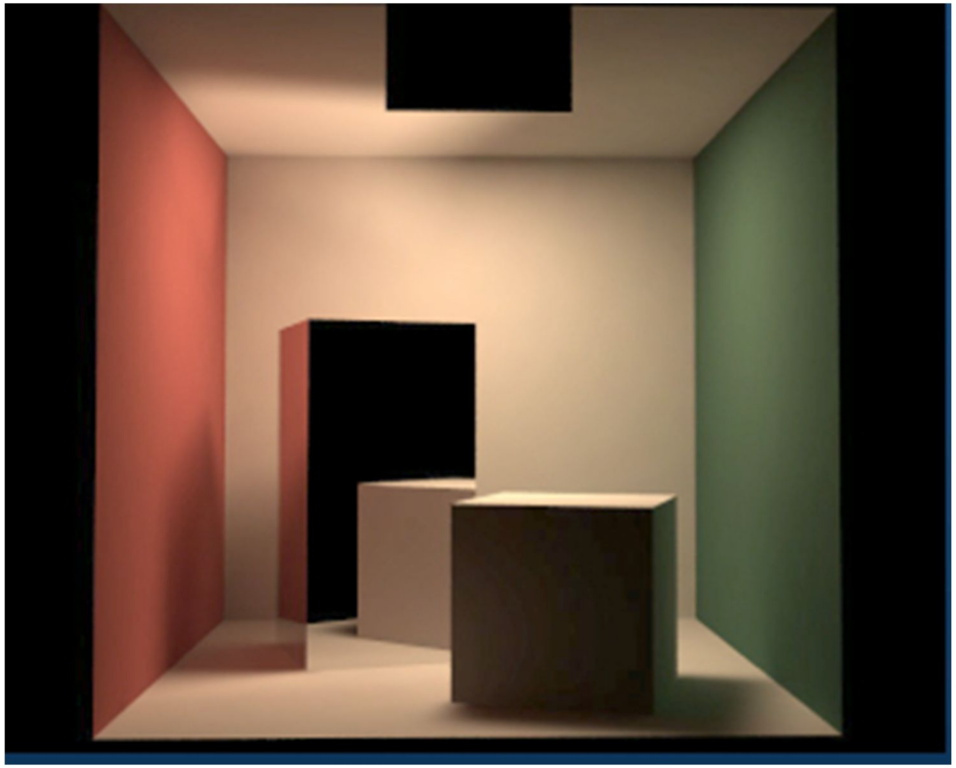
This box is white!

Global Illumination in real-time graphics

Pre-GI

Post-GI

Cornell Box: a rendering or photograph?



Rendering

Photograph

Rendering equation (revisited)

- Most rendering methods require solving an (approximation) of the rendering equation:

Incident light

Integral over the hemisphere of incident light

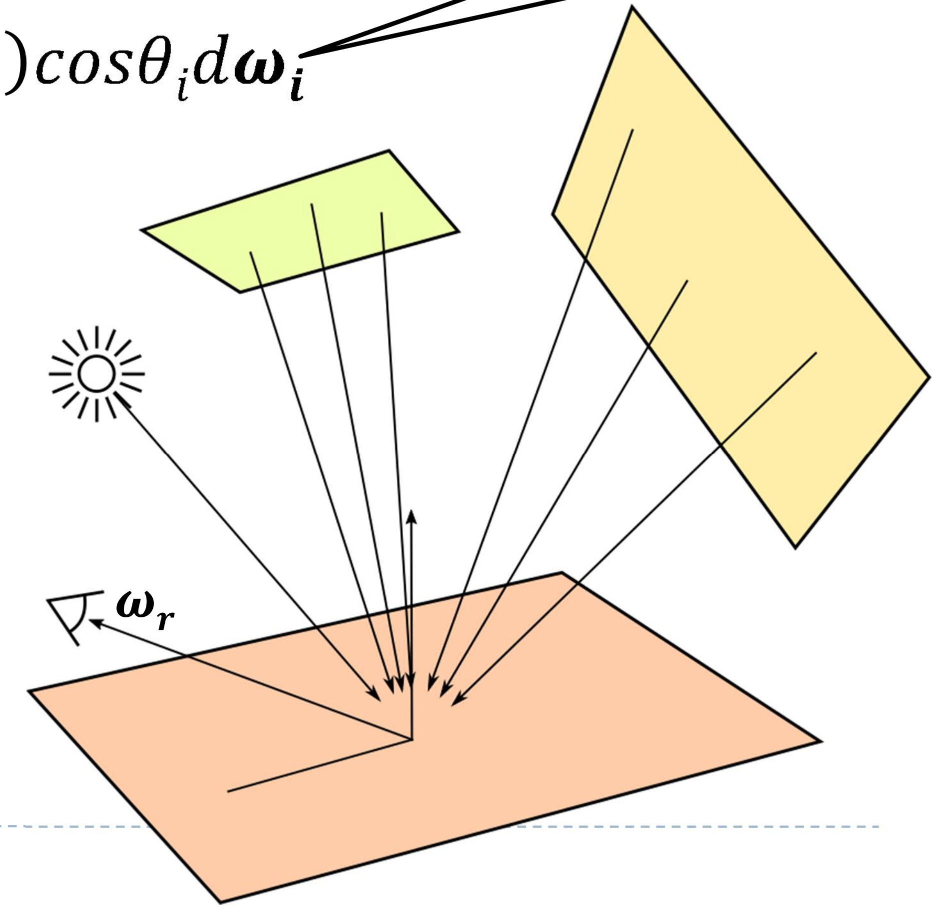
$$L_r(\boldsymbol{\omega_r}) = \int_{\Omega} \rho(\boldsymbol{\omega_i}, \boldsymbol{\omega_r}) L_i(\omega_i) cos\theta_i d\boldsymbol{\omega_i}$$

Reflected light

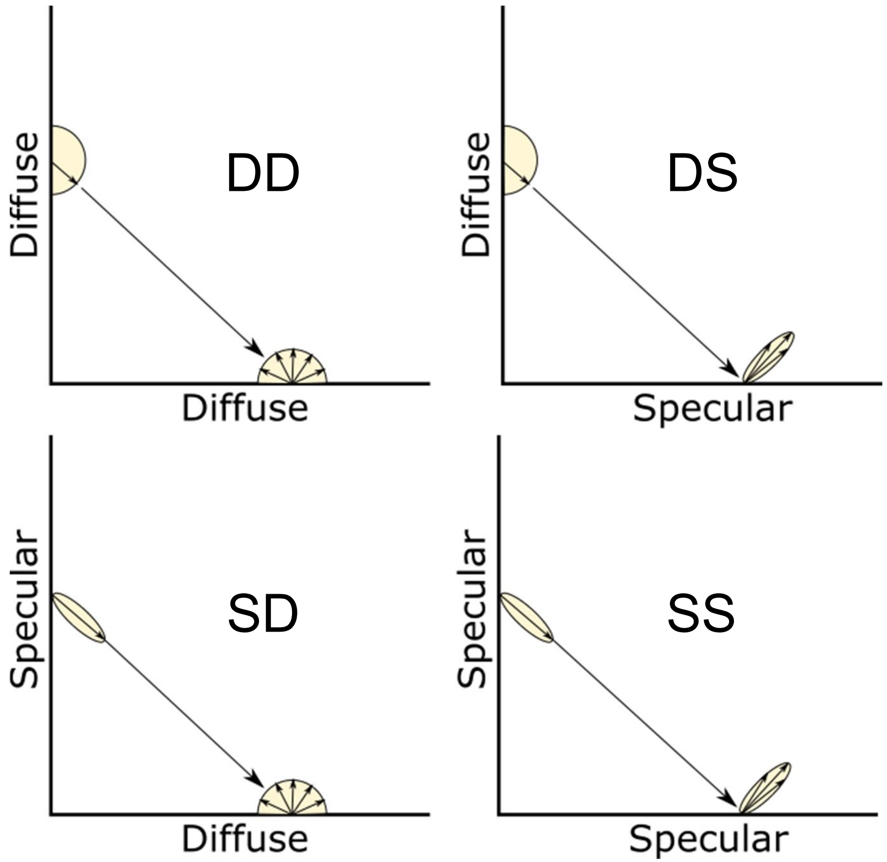
BRDF

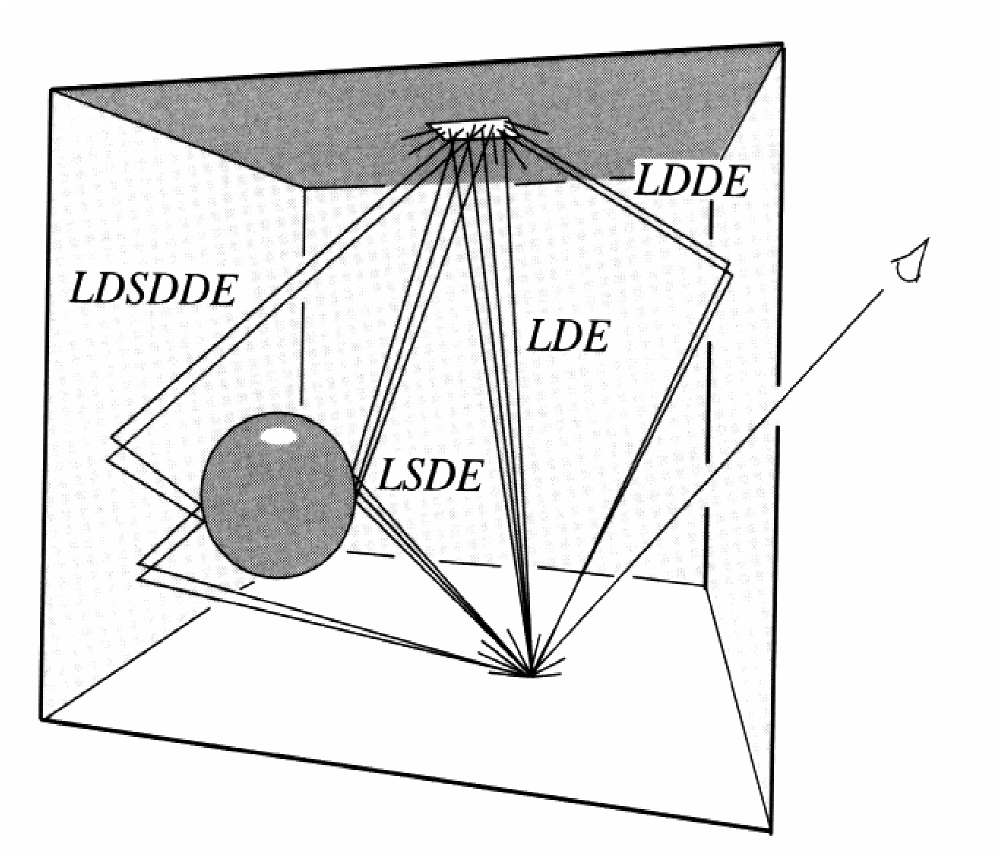
$$\boldsymbol{\omega_i} = [\phi_i, \theta_i]$$

- The solution is trivial for point light sources
- Much harder to estimate the contribution of other surfaces



Light transport





Shadows, refraction and caustics

- Problem: shadow ray strikes transparent, refractive object.
  - Refracted shadow ray will now miss the light.
  - This destroys the validity of the boolean shadow test.
- Problem: light passing through a refractive object will sometimes form *caustics* (right), artifacts where the envelope of a collection of rays falling on the surface is bright enough to be visible.

This is a photo of a real pepper-shaker.
Note the caustics to the left of the shaker, in and outside of its shadow.
Photo credit: Jan Zankowski

Shadows, refraction and caustics

- Solutions for shadows of transparent objects:
  - Backwards ray tracing (Arvo)
    - *Very* computationally heavy
    - Improved by stencil mapping (Shenya et al)
  - Shadow attenuation (Pierce)
    - Low refraction, no caustics

- More general solution:
  - *Path tracing*
  - *Photon mapping* (Jensen)→

Image from http://graphics.ucsd.edu/~henrik/
Generated with photon mapping

Path tracing

- Trace the rays from the camera (as in recursive ray tracing)
- [Russian roulette] When a surface is hit, either (randomly):
  - shoot another ray in the random direction sampled using the BRDF [importance sampling];
  - or terminate
- For each hit sample sample light sources (direct illumination) and other directions (indirect illumination)
- 40-1000s rays must be traced for each pixel
- The method converges to the exact solution of the rendering equation
  - But very slowly
  - Monte Carlo approach to solving the rendering equation

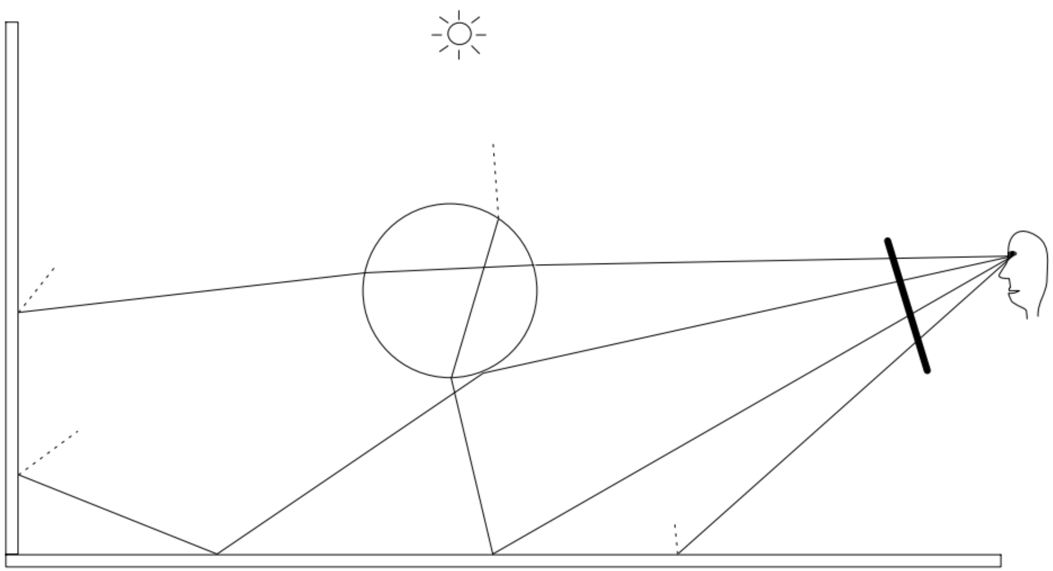


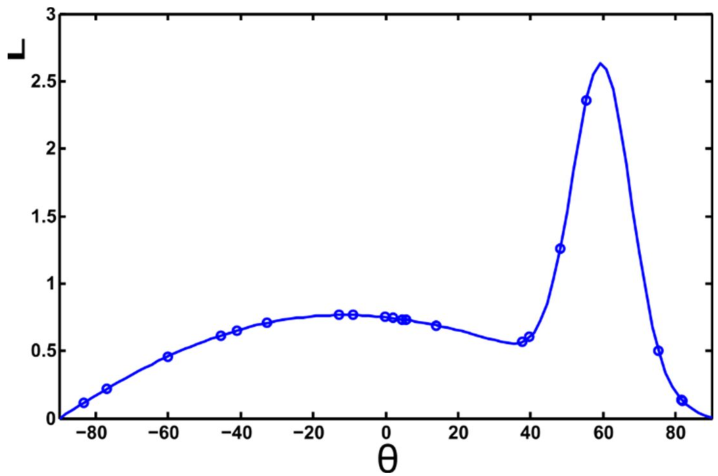
Image from A Practical Guide to Global Illumination using Photon Maps by Henrik Jensen (2000)

Monte-Carlo methods

- Path tracing estimates rendering equation by shooting rays in random directions (sampling) and averaging the contributions
- This is equivalent to estimating integral using Monte-Carlo sampling

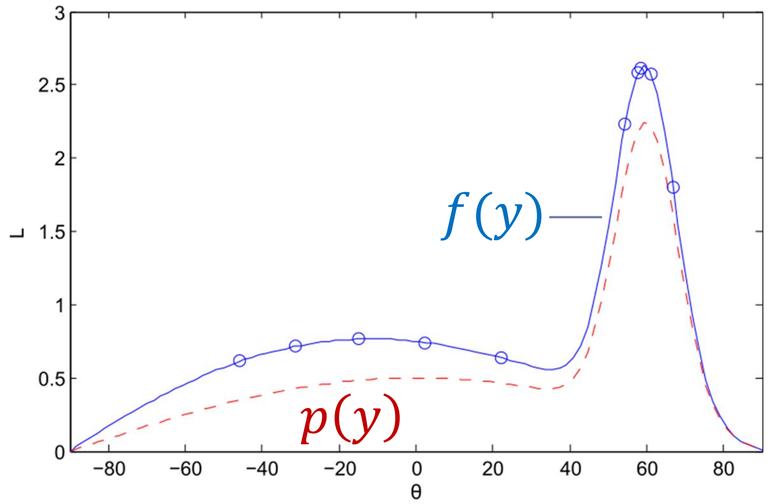
$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)(b-a)$$

where x_i are randomly drawn from Uniform(a,b)



Importance sampling

- Monte-Carlo sampling converges faster if ray directions with dominant contribution are sampled more often
- Dominant directions are unknown
  - But BRDF could be used as an estimate of importance
- When the sampling distribution is non-uniform, we need to use different estimator:

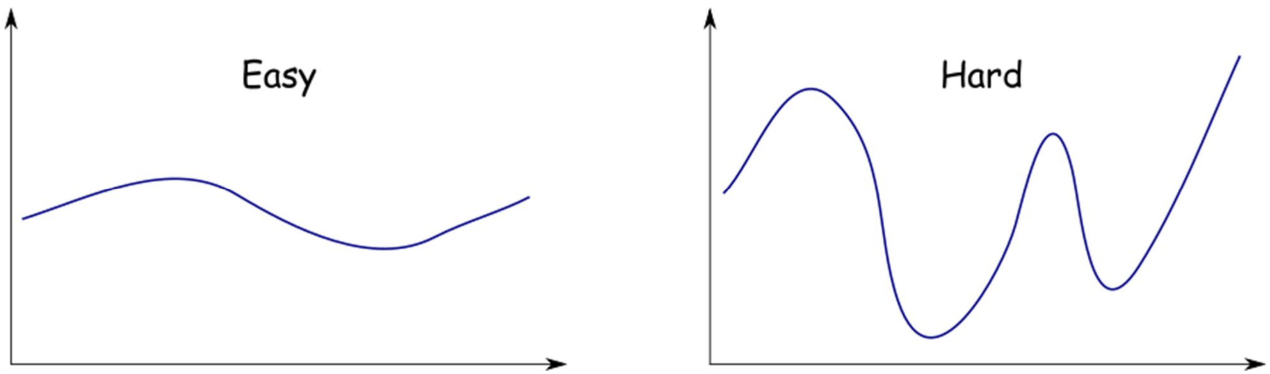


$$\int f(x)dx = \int \frac{f(x)}{p(x)} p(x)dx = E\left[\frac{f(y)}{p(y)}\right] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(y_i)}{p(y_i)}$$

Where y is sampled from the distribution $p(y)$ - shown as red-dashed line in the plot

Importance sampling (intuition)

- Monte-carlo integration requires less samples when the integrated function varies less



- One way to make the integrated function vary less: divide by an approximation of the integrated function

$$\hat{f}(x) = \frac{f(x)}{p(x)}$$

Russian roulette

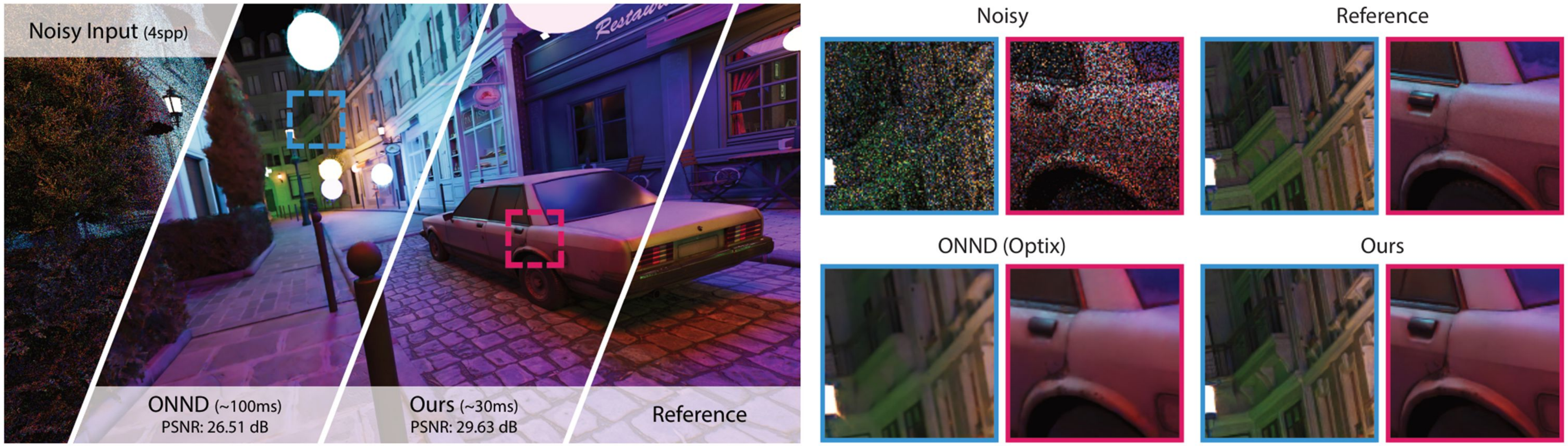
- Intuition: consecutive light bounces contribute less and less to the final color
  - But we cannot stop after N bounces as it will introduce bias (under-estimation)
- Instead: (after the first one or two bounces) terminate the current path with the probability q
  - Then, the estimator becomes

$$F' = \begin{cases} \frac{F}{1-q} & if\ \tau > q \\ 0 & otherwise \end{cases}, \quad \tau \sim Uniform(0,1), \quad F \text{ – next bounce radiance}$$

  - Longer paths (with more vertices) become unlikely
  - Works the best if we know the contribution of F is likely to be small
  - If q is too large, we may end up with high variance and *fireflies*

Denoising for Monte-Carlo rendering

- Instead of tracing 1000s of rays, we can trace 4-8 rays per pixel and employ a denoiser
  - Modern denoisers are (convolutional) neural networks that take as input sample radiance, geometric and material features (G-buffer) and warped samples from the previous frame(s)



From: Balint et al. 2023 http://dx.doi.org/10.1145/3588432.3591562

Photon mapping

Photon mapping is the process of emitting photons into a scene and tracing their paths probabilistically to build a *photon map*, a data structure which describes the illumination of the scene independently of its geometry.

This data is then combined with ray tracing to compute the global illumination of the scene.

Image by Henrik Jensen (2000)

Photon mapping—algorithm (1/2)

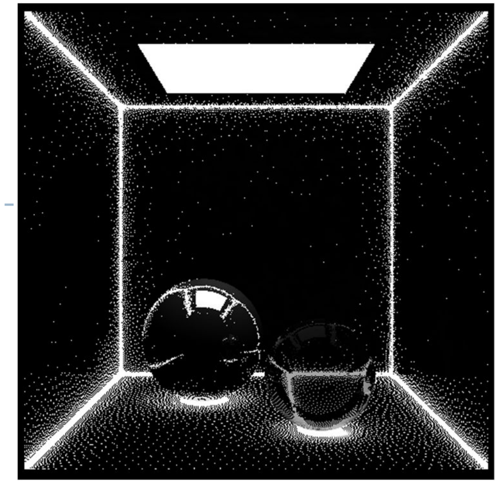


Image by Zack Waters

Photon mapping is a two-pass algorithm:

1. Photon scattering
   A. Photons are fired from each light source, scattered in randomly-chosen directions.
      The number of photons per light is a function of its surface area and brightness.
   B. Photons fire through the scene (re-use that raytracer). Where they strike a surface they are either absorbed, reflected or refracted.
   C. Wherever energy is absorbed, cache the location, direction and energy of the photon in the *photon map*. The photon map data structure must support fast insertion and fast nearest-neighbor lookup; a *kd-tree*[1] is often used.

Photon mapping—algorithm (2/2)

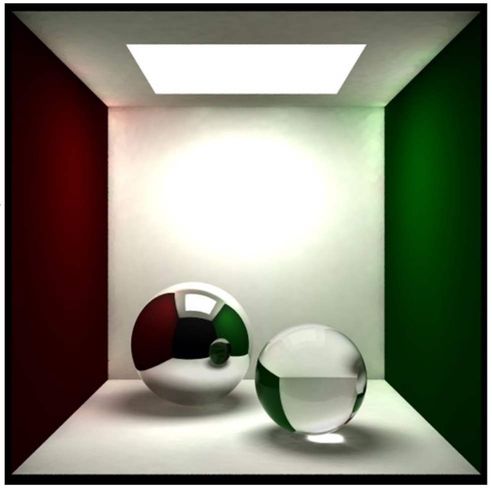


Image by Zack Waters

Photon mapping is a two-pass algorithm:

2. Rendering
   A. Ray trace the scene from the point of view of the camera.
   B. For each first contact point *P* use the ray tracer for specular but compute diffuse from the photon map.
   C. Compute radiant illumination by summing the contribution along the eye ray of all photons within a sphere of radius *r* of *P*.
   D. Caustics can be calculated directly here from the photon map. For accuracy, the caustic map is usually distinct from the radiance map.

Photon mapping is probabilistic

This method is a great example of *Monte Carlo integration*, in which a difficult integral (the lighting equation) is simulated by randomly sampling values from within the integral's domain until enough samples average out to about the right answer.

- This means that you're going to be firing *millions* of photons. Your data structure is going to have to be very space-efficient.

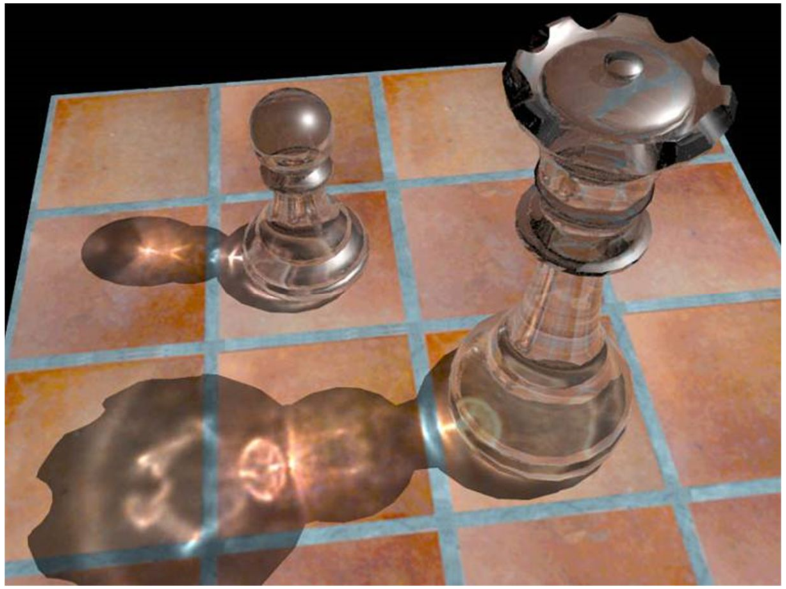
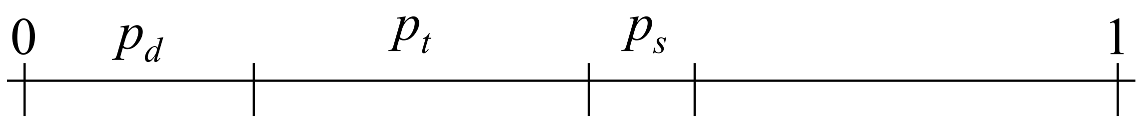


Image credit: http://www.okino.com/conv/imp_jt.htm

Photon mapping is probabilistic

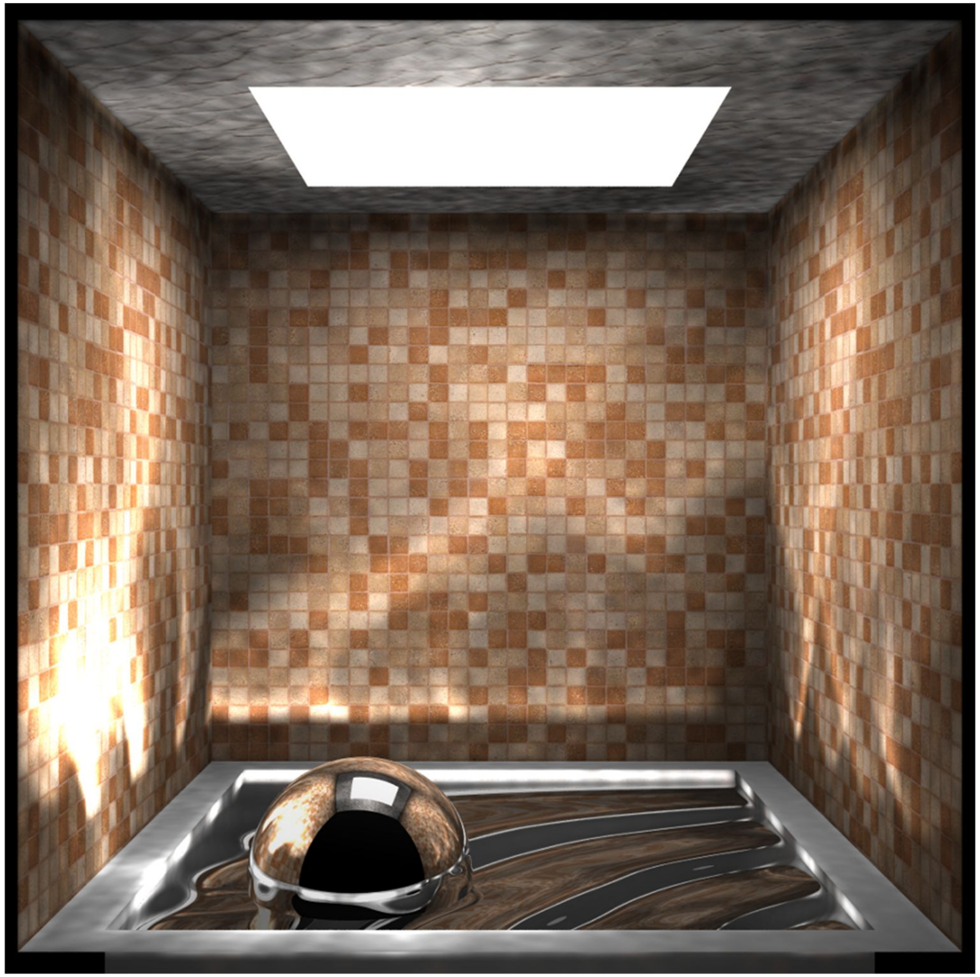
- Initial photon direction is random. Constrained by light shape, but random.
- What exactly happens each time a photon hits a solid also has a random component:
  - Based on the diffuse reflectance, specular reflectance and transparency of the surface, compute probabilities p_d, p_s and p_t where $(p_d+p_s+p_t) \leq 1$. This gives a probability map:



This surface would have minimal specular highlight.

  - Choose a random value $p \in [0,1]$. Where p falls in the probability map of the surface determines whether the photon is reflected, refracted or absorbed.

Photon mapping gallery

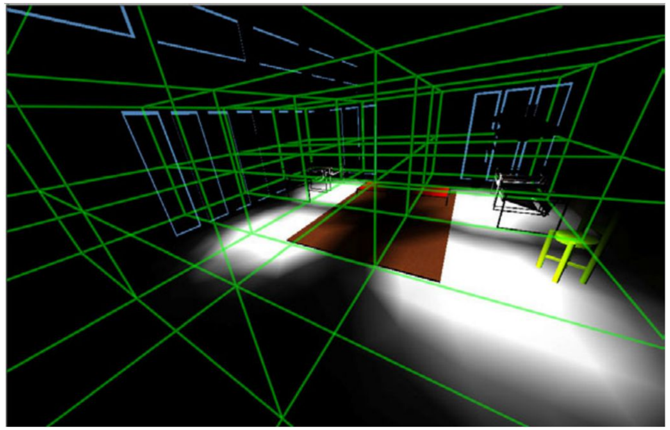


http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html

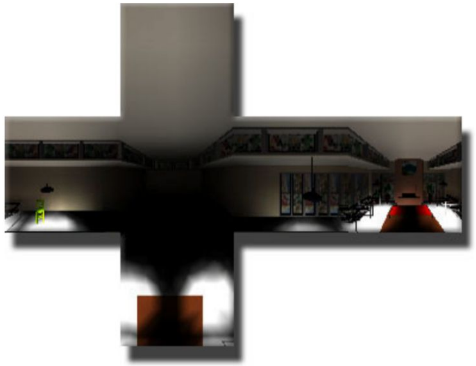
http://graphics.ucsd.edu/~henrik/images/global.html

http://www.pbrt.org/gallery.php

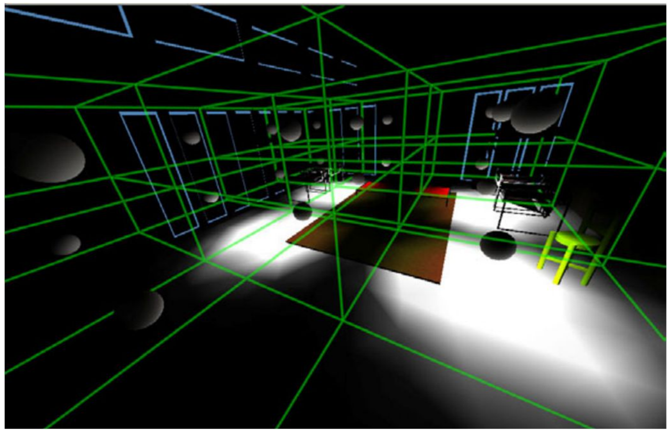
Real-time global illumination: irradiance probes (diffuse GI only)



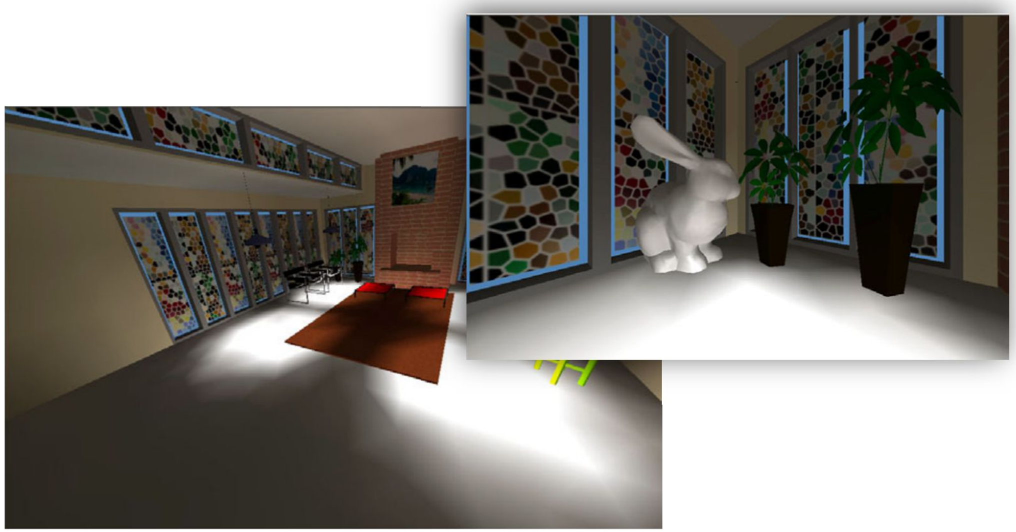
Step 1: Create a voxel grid



Step 2: For each voxel centre, render a cube map (or sample with a ray-tracer). For the first bounce, render direct illumination only.



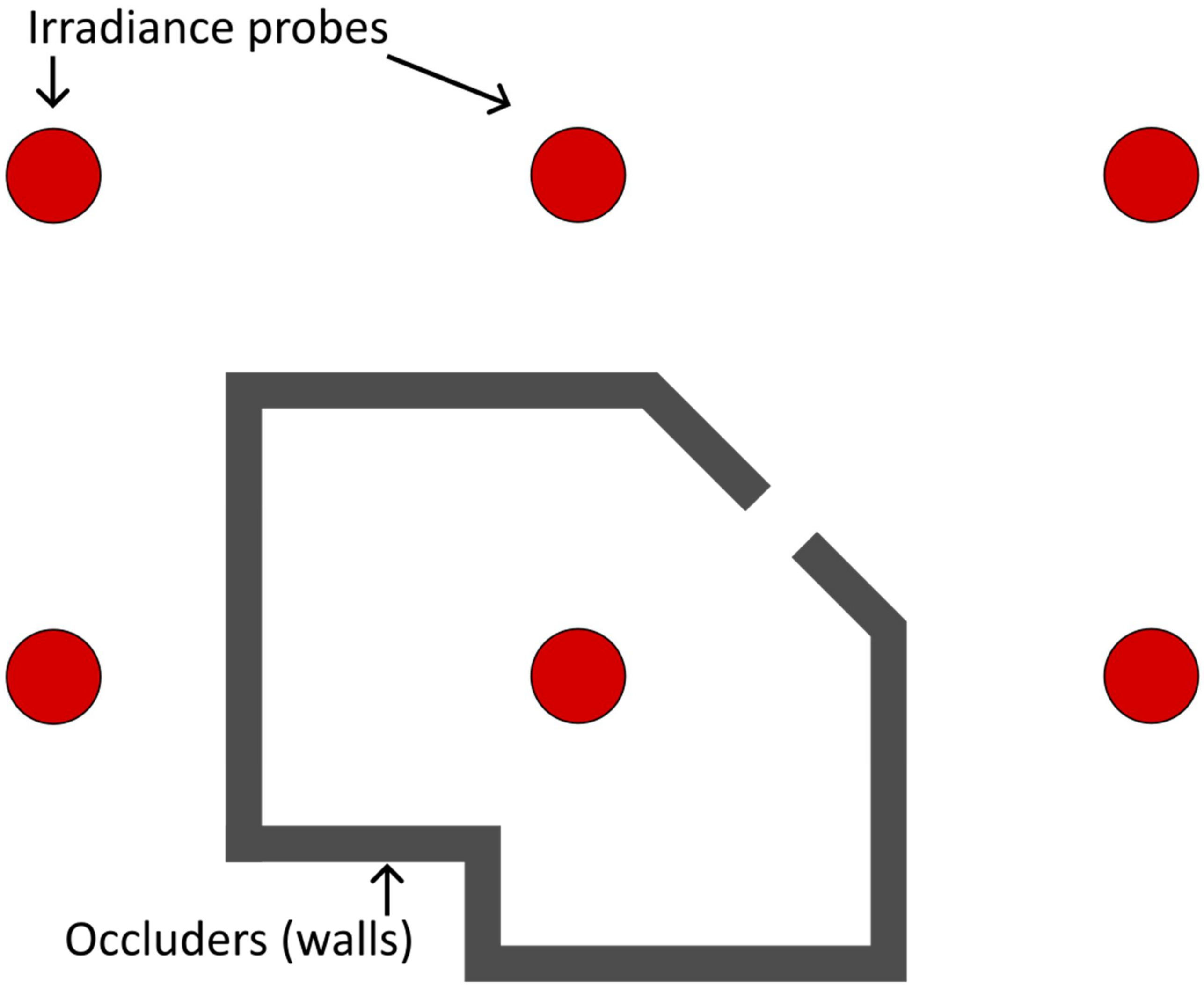
Step 3: Integrate incoming light over a hemisphere (compute irradiance probes)



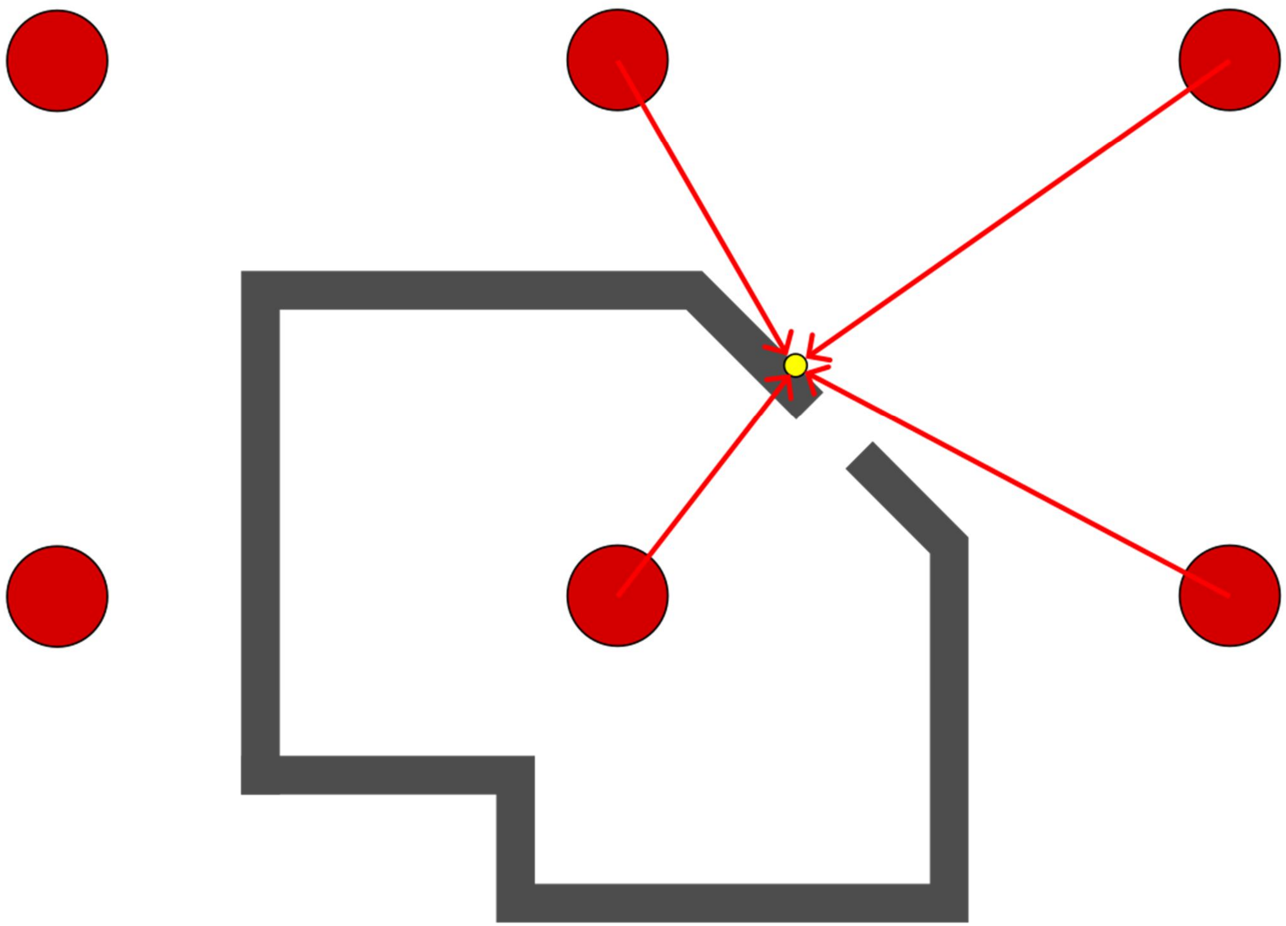
Step 4: Render the scene using interpolated values from the irradiance probes to look up indirect illumination

Repeat Steps 2 and 3 (potentially over consecutive frames) to simulate more bounces of light

Dynamic Diffuse Global Illumination

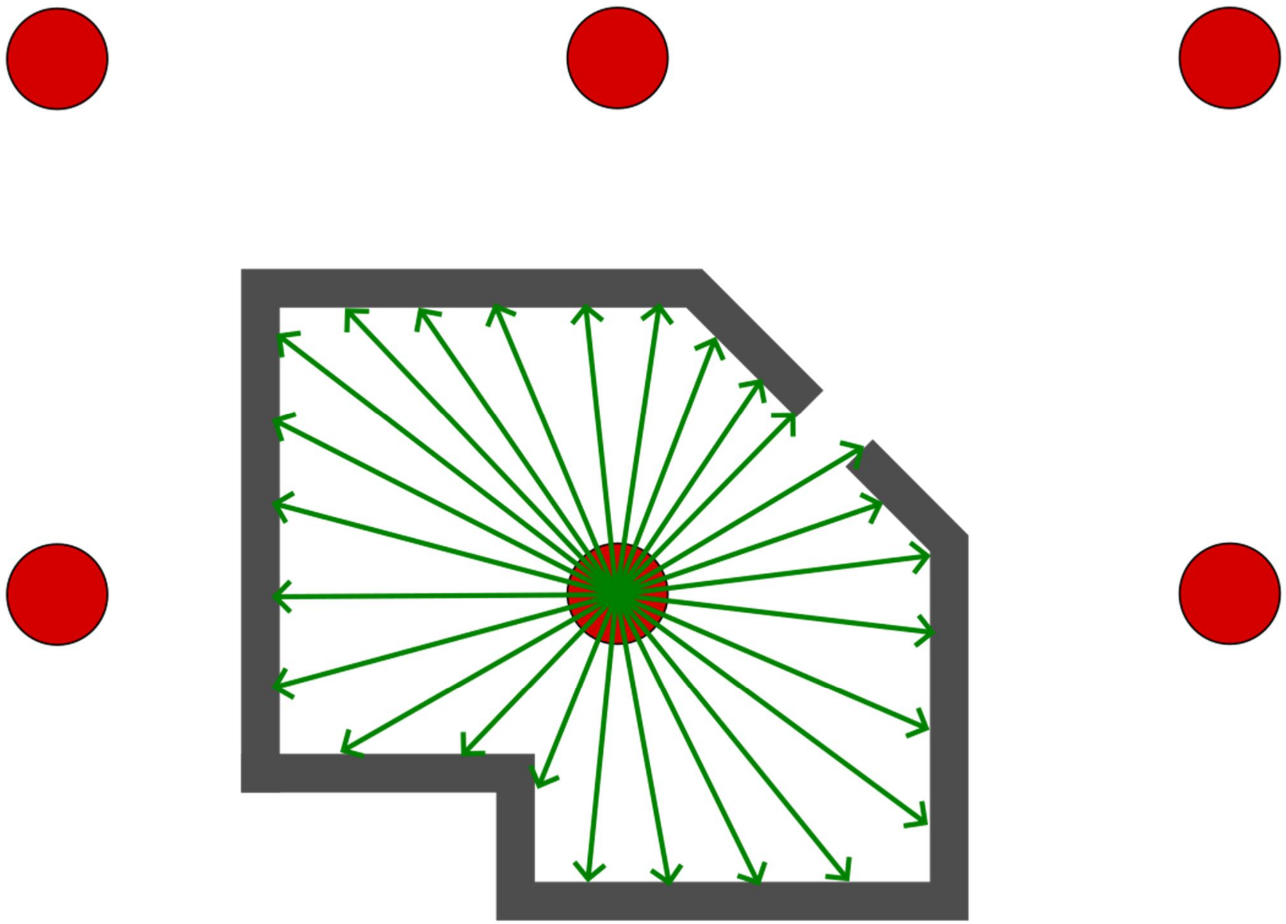


Dynamic Diffuse Global Illumination



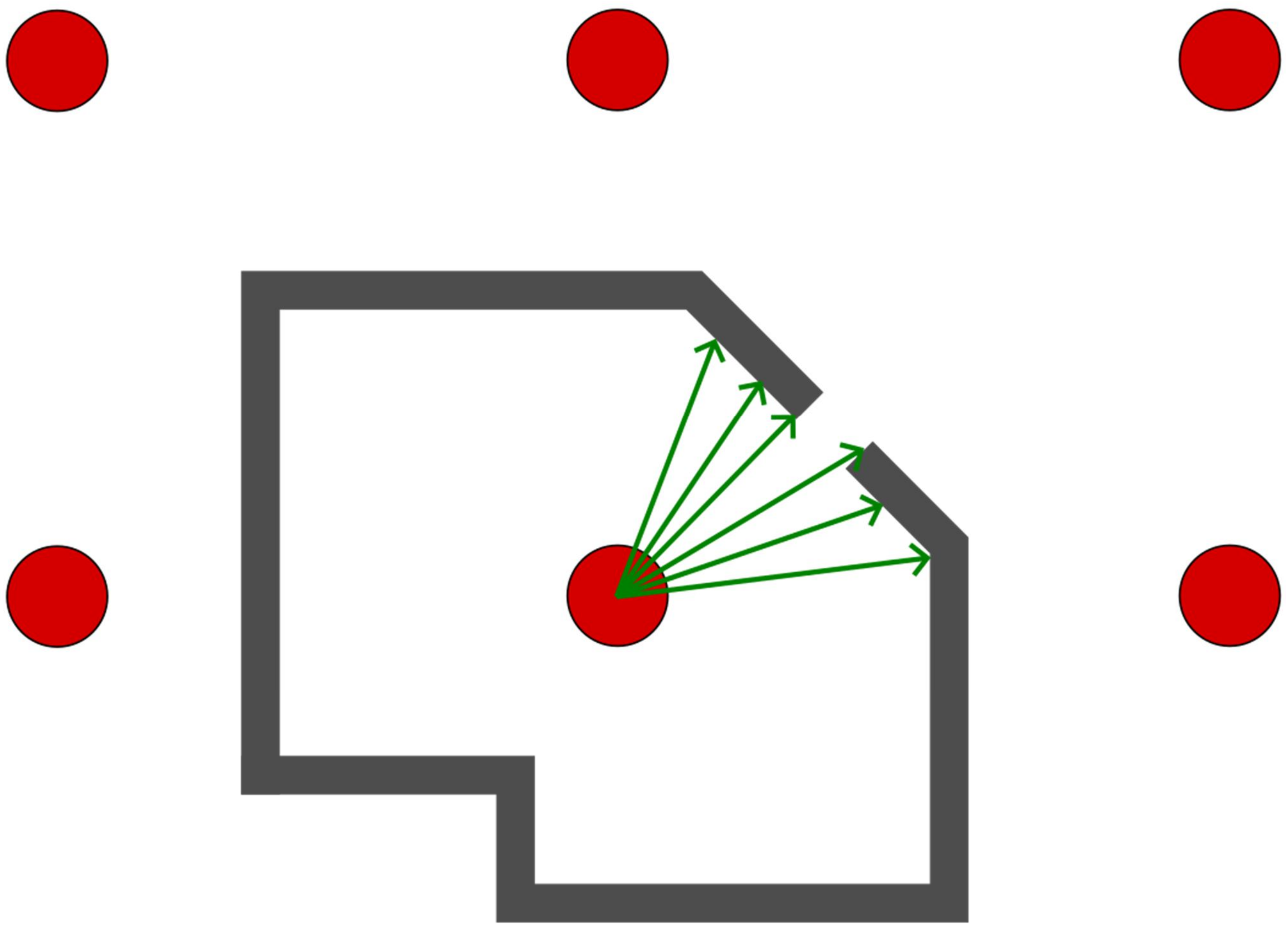
Main issue: how to discount the effect of the occluded probes

Dynamic Diffuse Global Illumination



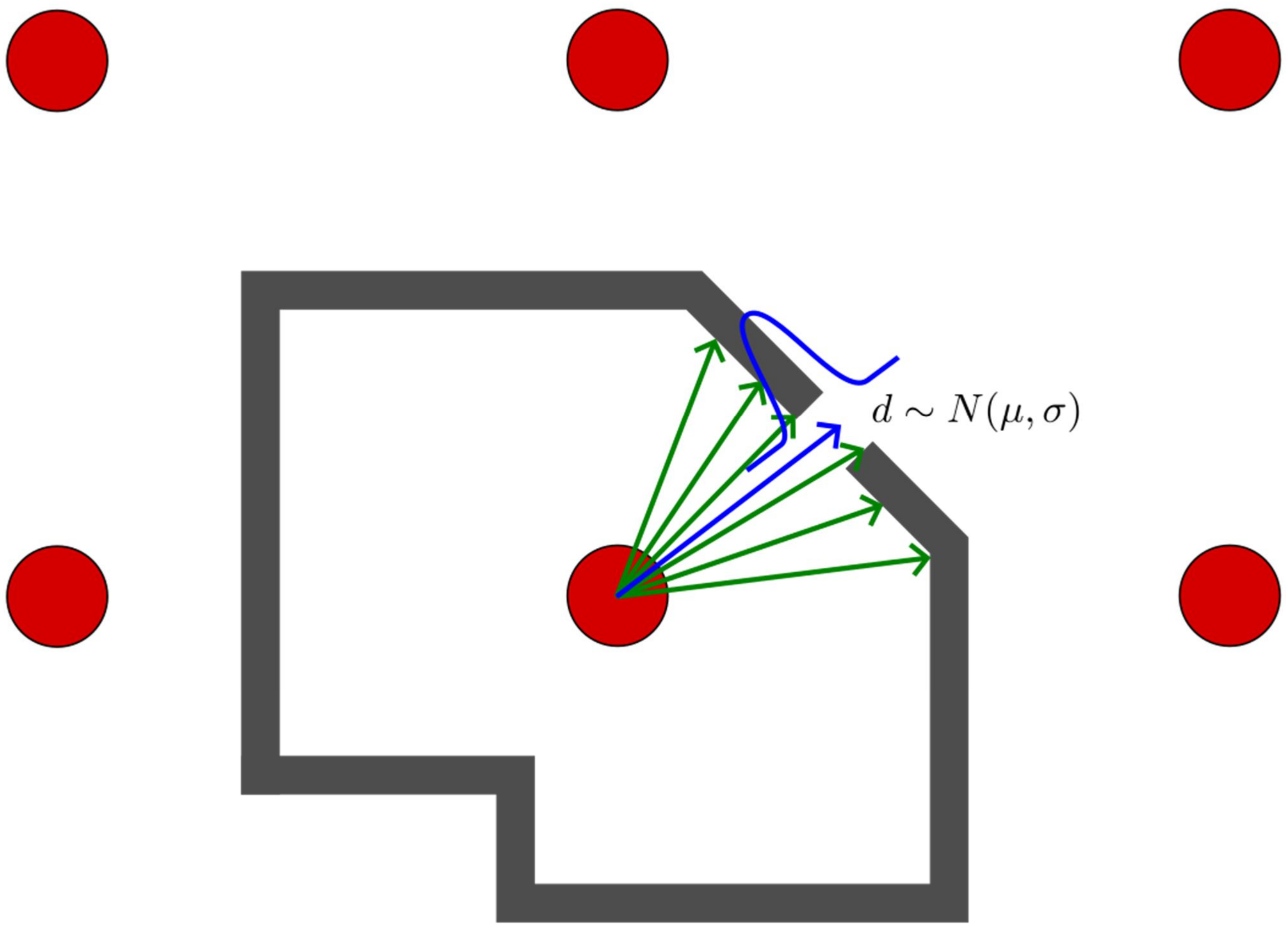
Storing depth per sample is too expensive

Dynamic Diffuse Global Illumination



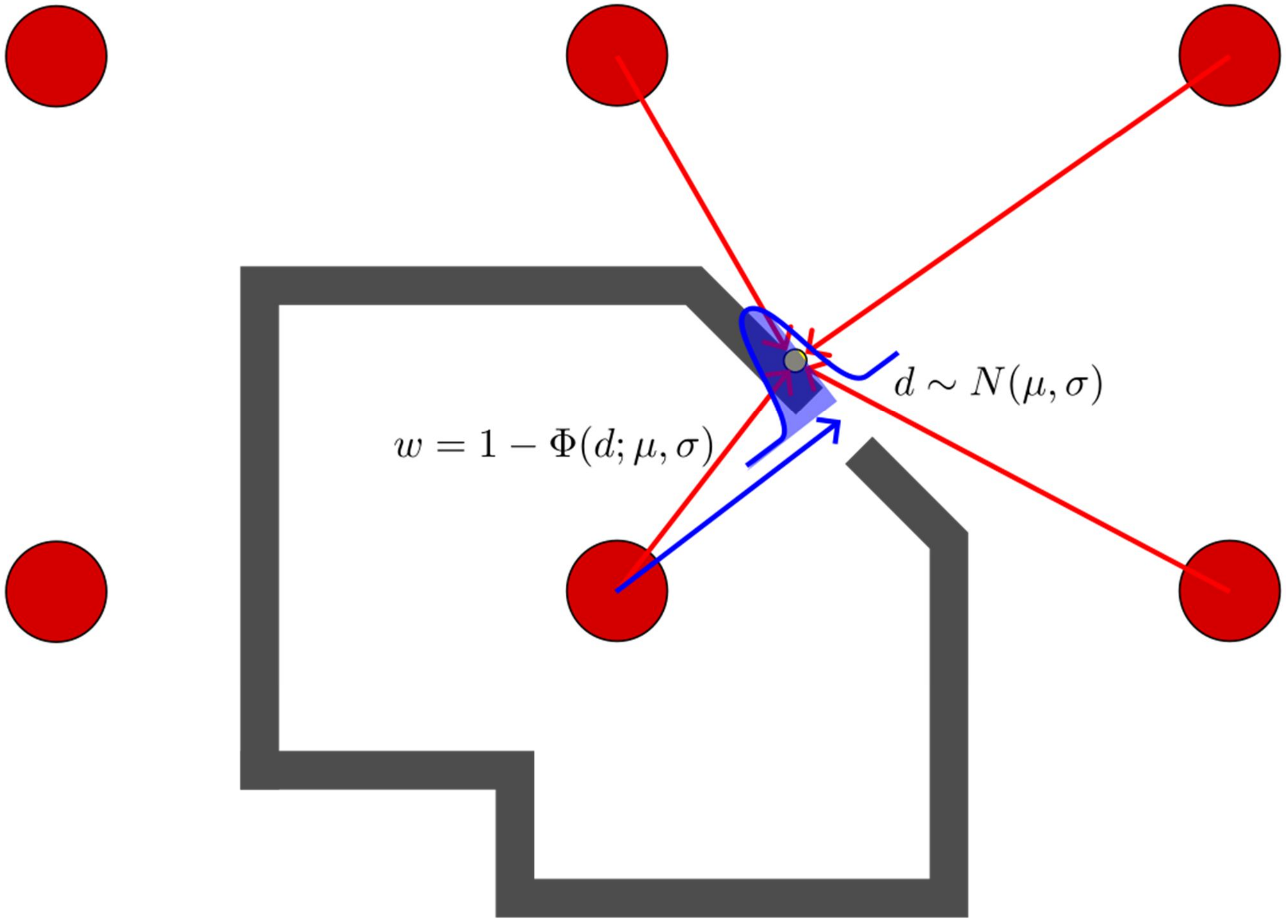
We want to store depth per a range of directions

Dynamic Diffuse Global Illumination



To encode the variation in depth, we store the distribution of depths (mean and variance)

Dynamic Diffuse Global Illumination



When interpolating irradiance from the probes, use the cumulative distribution to determine weight due to shadowing

Ambient occlusion

- Approximates global illumination
- Estimate how much occluded is each surface
  - And reduce the ambient light it receives accordingly
- Much faster than a full global illumination solution, yet appears very plausible
  - Commonly used in animation, where plausible solution is more important than physical accuracy

Image generated with ambient component only (no light) and modulated by ambient occlusion factor.

Ambient occlusion in action

Car photos from John Hable's presentation at GDC 2010, "Uncharted 2: HDR Lighting" (filmicgames.com/archives/6)

Ambient occlusion in action

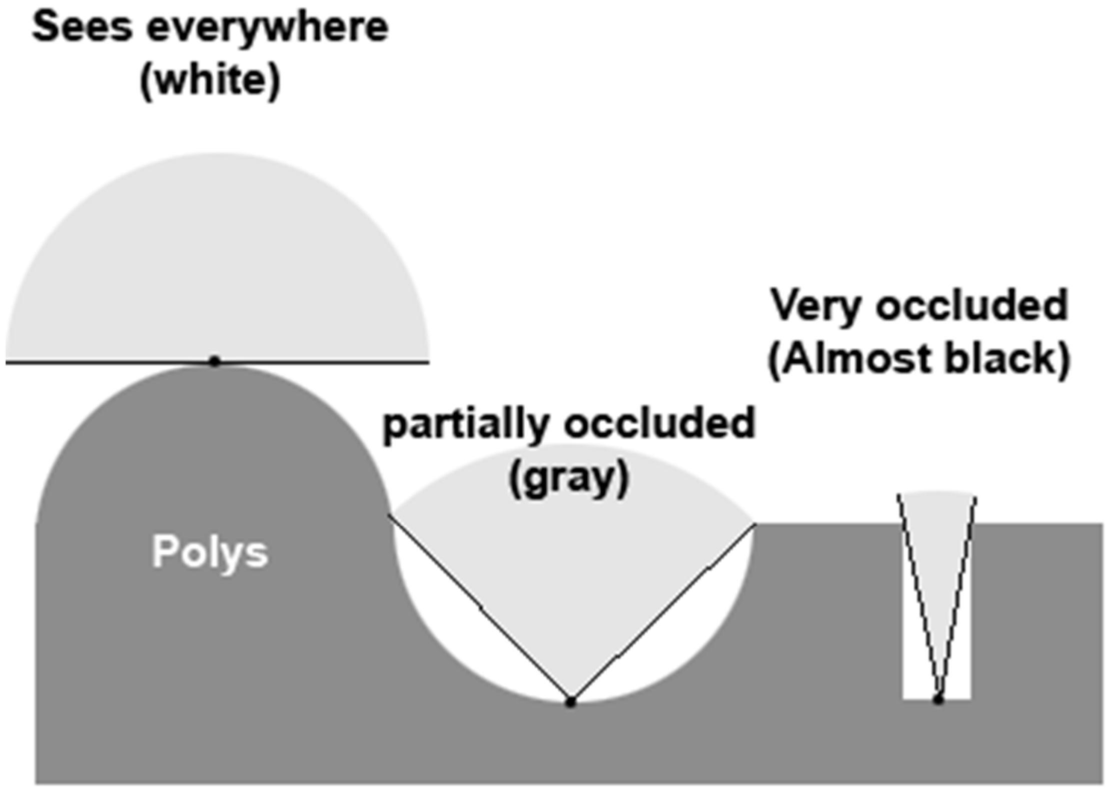
Car photos from John Hable's presentation at GDC 2010, "Uncharted 2: HDR Lighting" (filmicgames.com/archives/6)

Ambient occlusion

- For a point on a surface, shoot rays in random directions
- Count how many of these rays hit objects
- The more rays hit other objects, the more occluded is that point
  - The darker is the ambient component

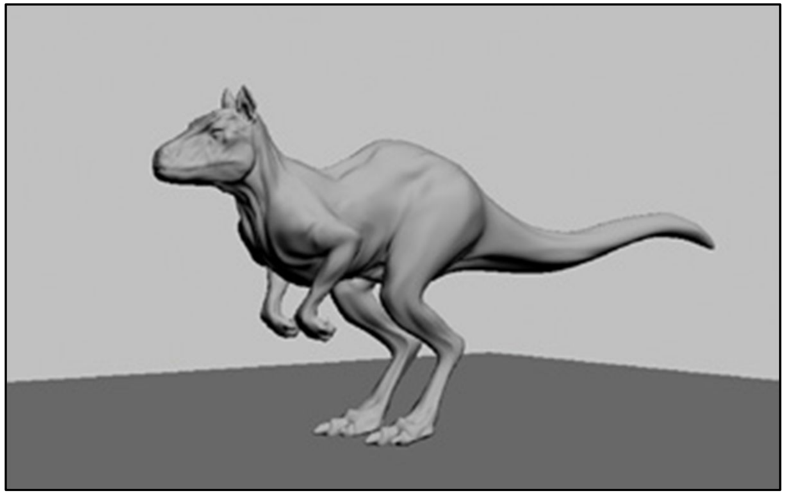
$$A_{\bar{p}} = \frac{1}{\pi} \int_{\Omega} V_{\bar{p},\hat{\omega}} (\hat{n} \cdot \hat{\omega}) d\omega$$

A_p	occlusion at point p
n	normal at point p
$V_{p,\omega}$	visibility from p in direction ω
Ω	integrate over a hemisphere



Ambient occlusion - Theory

- This approach is very flexible
- Also very expensive!
- To speed up computation, randomly sample rays cast out from each polygon or vertex (this is a *Monte-Carlo* method)
- Alternatively, render the scene from the point of view of each vertex and count the background pixels in the render
- Best used to pre-compute per-object "*occlusion maps*", texture maps of shadow to overlay onto each object
- But pre-computed maps fare poorly on animated models...



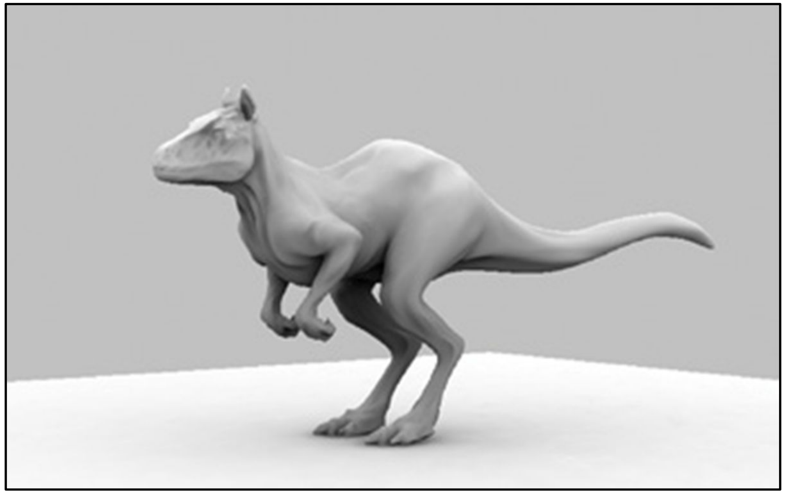


Image credit: "GPU Gems 1", nVidia, 2004.
Top: without AO.   Bottom: with AO.

Screen Space Ambient Occlusion - SSAO

"True ambient occlusion is hard, let's go hacking."

- Approximate ambient occlusion by comparing z-buffer values in screen space!
- Open plane = unoccluded
- Closed 'valley' in depth buffer = shadowed by nearby geometry
- Multi-pass algorithm
- Runs entirely on the GPU

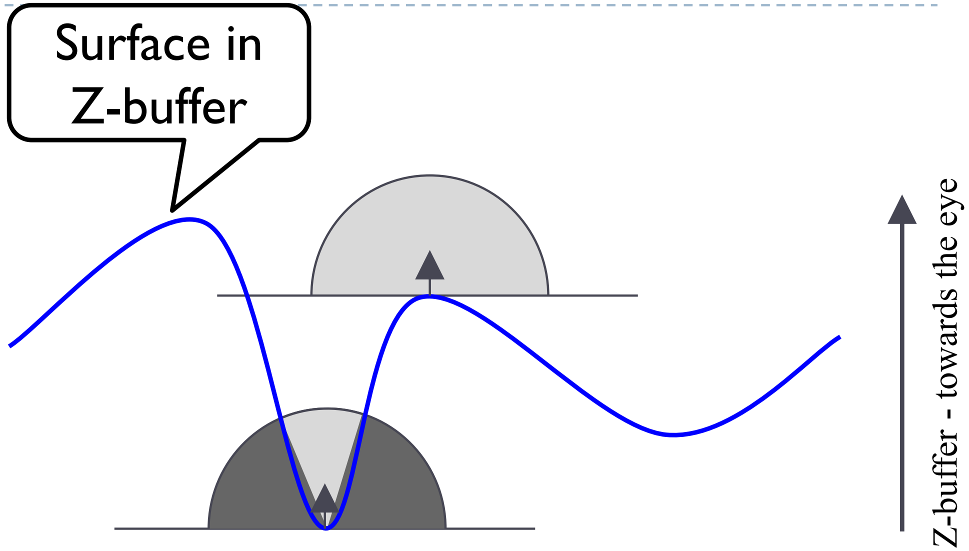


Image: CryEngine 2. M. Mittring, "Finding Next Gen – CryEngine 2.0, Chapter 8", SIGGRAPH 2007 Course 28

References

Shirley and Marschner, "Fundamentals of Computer Graphics", Chapter 24 (2009)

Matt Pharr, Wenzel Jakob, Greg Humphreys, "Physically Based Rendering From Theory to Implementation" (2017)

Dynamic Diffuse Global Illumination
- Majercic et al. "Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields"

Ambient occlusion and SSAO
- "GPU Gems 2", nVidia, 2005. Vertices mapped to illumination. http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter14.html
- MITTRING, M. 2007. Finding Next Gen – CryEngine 2.0, Chapter 8, SIGGRAPH 2007 Course 28 – Advanced Real-Time Rendering in 3D Graphics and Games, Siggraph 2007, San Diego, CA, August 2007. http://developer.amd.com/wordpress/media/2012/10/Chapter8-Mittring-Finding_NextGen_CryEngine2.pdf
- John Hable's presentation at GDC 2010, "Uncharted 2: HDR Lighting" (filmicgames.com/archives/6)

Photon mapping
- Henrik Jensen, "Global Illumination using Photon Maps": http://graphics.ucsd.edu/~henrik/
- Henrik Jensen, "Realistic Image Synthesis Using Photon Mapping"
- Zack Waters, "Photon Mapping": http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html

Some slides are the curtesy of Alex Benton

Advanced Graphics & Image Processing

Image-based rendering

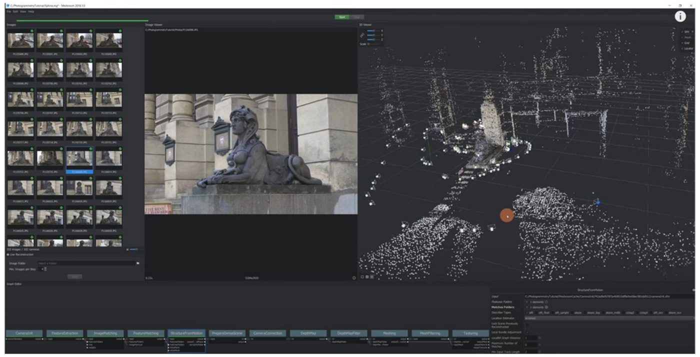
Rafał Mantiuk

Computer Laboratory, University of Cambridge

What is image-based rendering (IBR)?

- IBR ≈ use images for 3D rendering

3D mesh + textures + shading



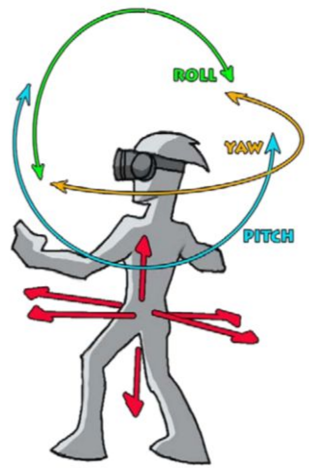
Photogrammetry

Neural Radiance Fields

- Our focus: methods that let us capture content with cameras

Motivation: why do we need image-based rendering?

- For inexpensive creation of high-quality 3D content
  - Minimize manual steps
  - Use cameras, which are good and abundant
- Why do we need 3D content?
  - AR/VR (+ novel display tech)
  - User-created content
  - 3D-printing
  - E-commerce



3D computer graphics

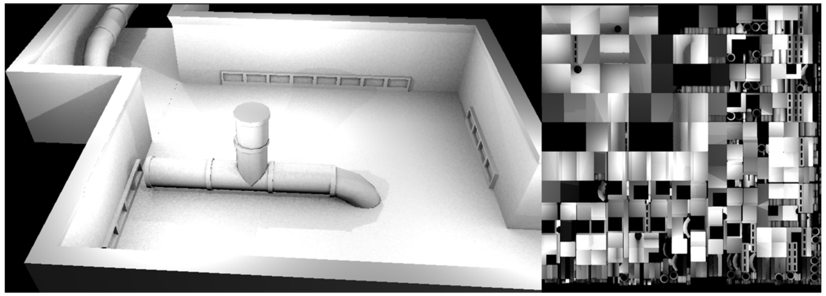
- We need:
  - Geometry + materials + textures
  - Lights
- Full control of illumination, realistic material appearance
- Graphics assets are expensive to create
- Rendering can be expensive
  - Shading tends to takes most of the computation

Cyberpunk 2077 (C) 2020 by CD Projekt RED

Baked / precomputed illumination

- We need:
  - Geometry + textures + (light maps)
- No need to scan and model materials
- Much faster rendering – simplified shading

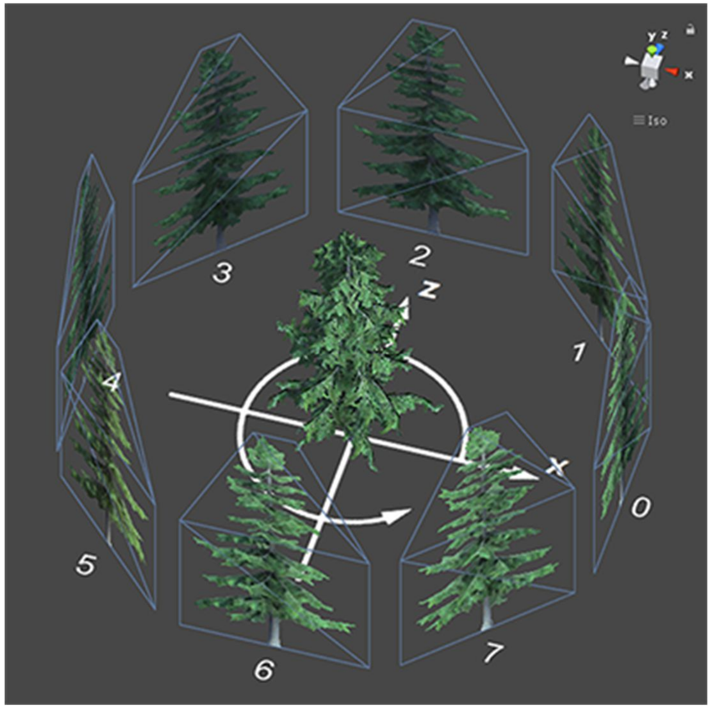
Google Earth



Precomputed light maps (from Wikipedia)

Billboards / Sprites

- We need:
  - Simplified geometry + textures (with alpha)
  - Lights
- Much faster to render than objects with 1000s of triangles
- Used for distant objects
  - or a small rendering budget
- Can be pre-computed from complex geometry



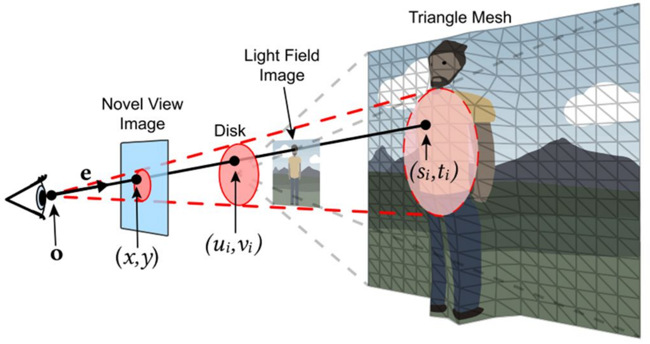
A tree rendered from a set of billboards
From: https://docs.unity3d.com/ScriptReference/BillboardAsset.html

Light fields

- We need:
  - Images of the scene
    - Or a microlens image
- Does not need any geometry
  - But requires a large number of images for good quality
- Photographs are rep-projected on a (focal) plane
- No relighting

Light fields + depth

- We need:
  - Depth map
  - Images of the object/scene
- We can use camera-captured images
- View-dependent shading
- Depth-map can be computed using multi-view stereo techniques
- No relighting

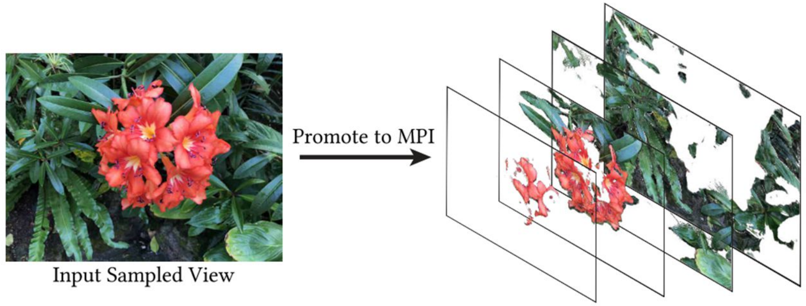


A depth map is approximated by triangle mesh and rasterized. From: Overbeck et al. TOG 2018, https://doi.org/10.1145/3272127.3275031.

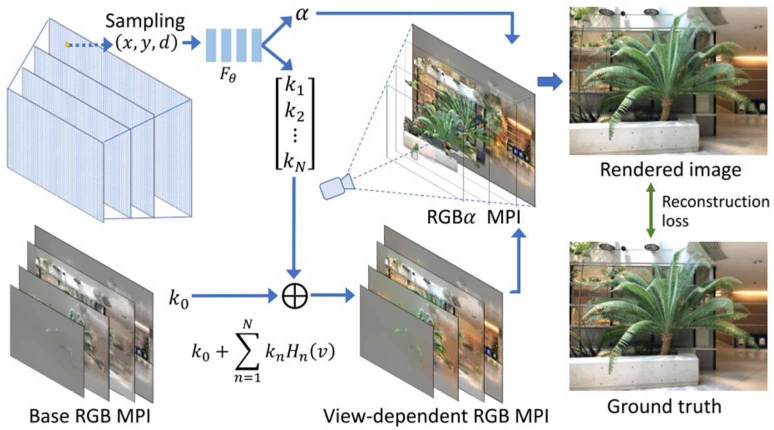
Demo: https://augmentedperception.github.io/welcome-to-lightfields/

Multi-plane images (MPI)

- We need:
  - Images of the scene + camera poses
- Each plane: RGB + alpha
  - Decomposition formulated as an optimization problem
  - Differential rendering
- Only front view



From [1]

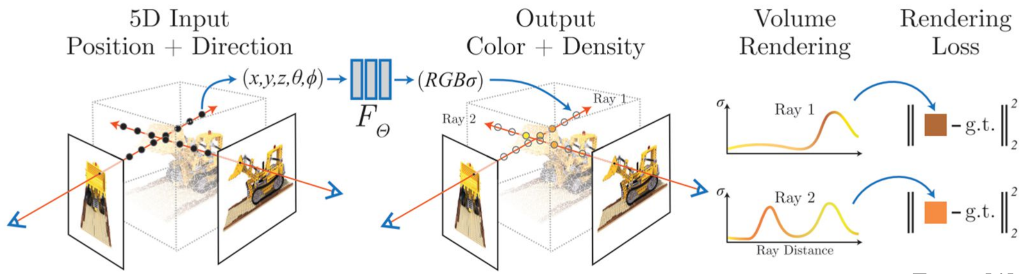


From [2]

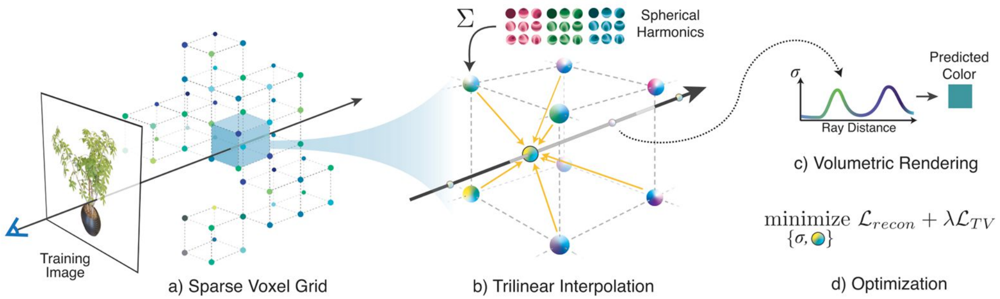
[1] Mildenhall, et al. "Local Light Field Fusion." *ACM Transactions on Graphics* 38, no. 4 (July 12, 2019): 1–14.
https://doi.org/10.1145/3306346.3322980
[2] Wizadwongsa et al. "NeX: Real-Time View Synthesis with Neural Basis Expansion." In *CVPR*, 8530–39. IEEE, 2021.
https://doi.org/10.1109/CVPR46437.2021.00843
https://nex-mpi.github.io/

Neural Radiance Fields (NeRF)

- We need
  - Images of the scene
    + camera poses
- Similar to MPI but stored in a volumetric data structure
  - Implicit: multi-layer perceptron
  - Explicit: Voxel grid
- Volumetric differential rendering



From [1]



From [2]

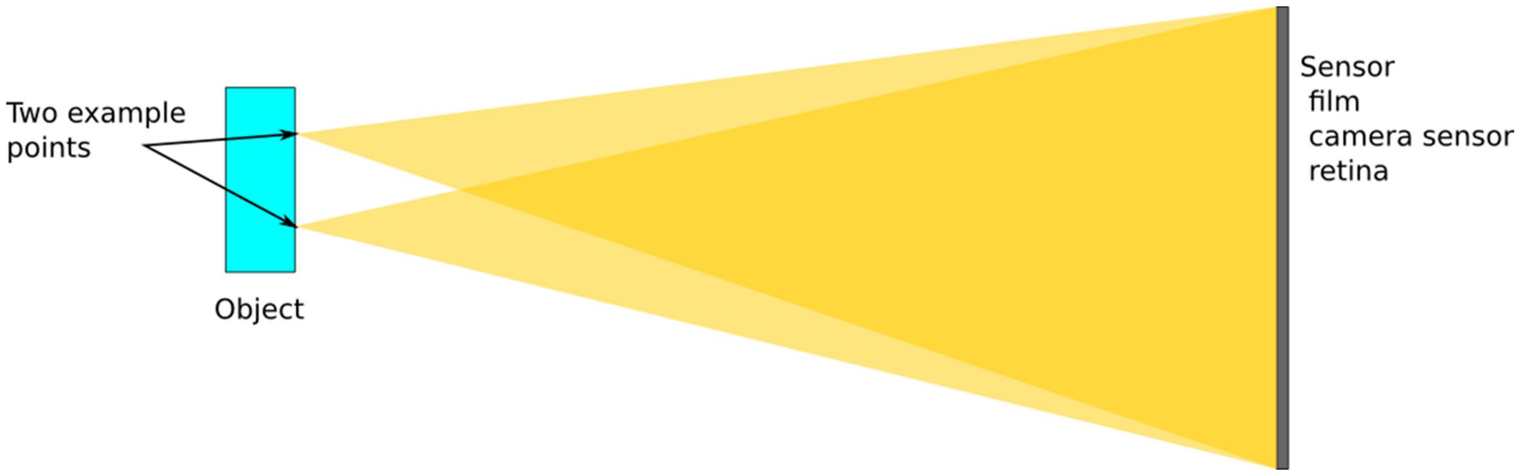
[1] Mildenhall, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," 405–21, 2020. https://doi.org/10.1007/978-3-030-58452-8_24.
[2] Yu et al. "Plenoxels: Radiance Fields without Neural Networks." In *CVPR*, 5501–10, 2022. http://arxiv.org/abs/2112.05131.

Finite aperture imaging

imaging and lens

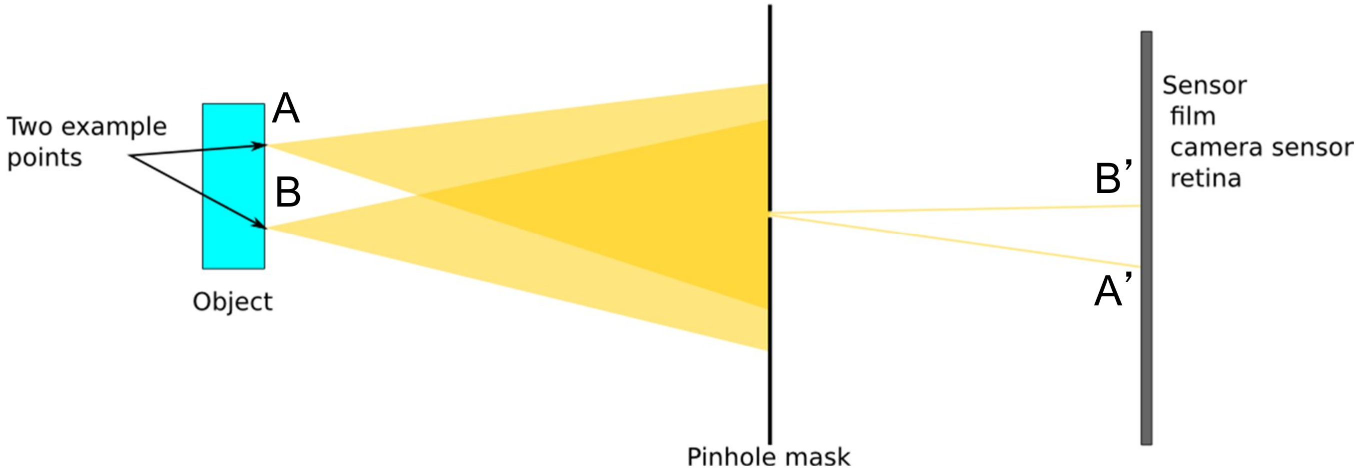
Rafał Mantiuk
Computer Laboratory, University of Cambridge

Imaging – without lens



Every point in the scene illuminates every point (pixel) on a sensor. Everything overlaps - no useful image.

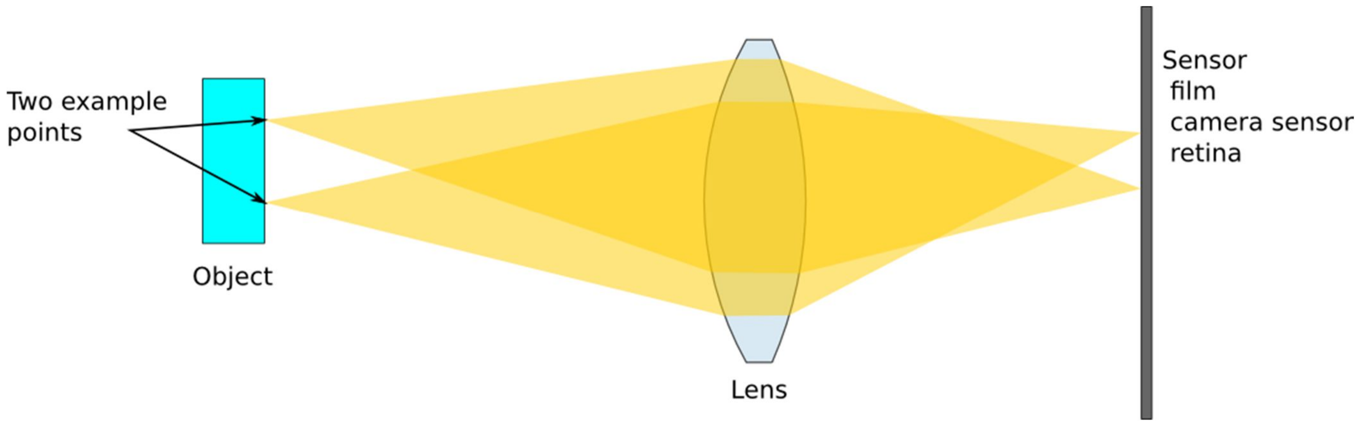
Imaging – pinhole camera



Pinhole masks all but only tiny beams of light. The light from different points is separated and the image is formed.

But very little light reaches the sensor.

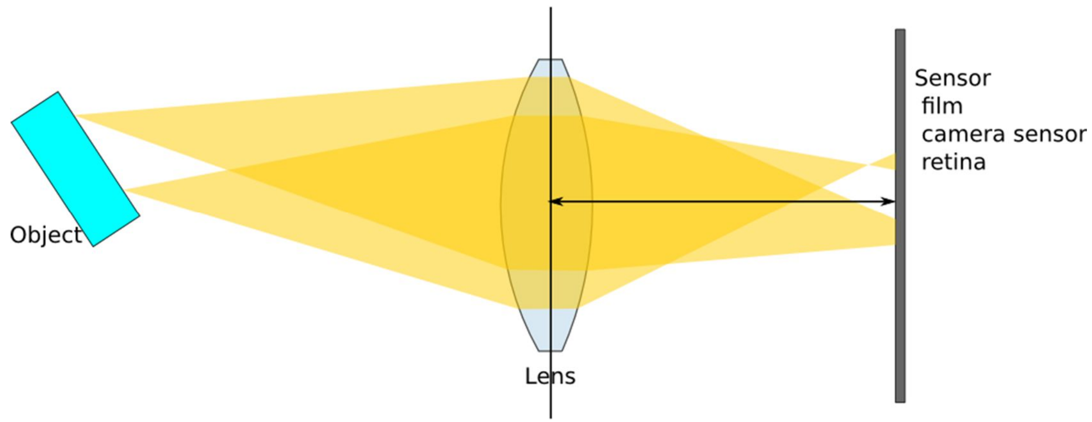
Imaging – lens



Lens can focus a beam of light on a sensor (focal plane).

Much more light-efficient than the pinhole.

Imaging – lens

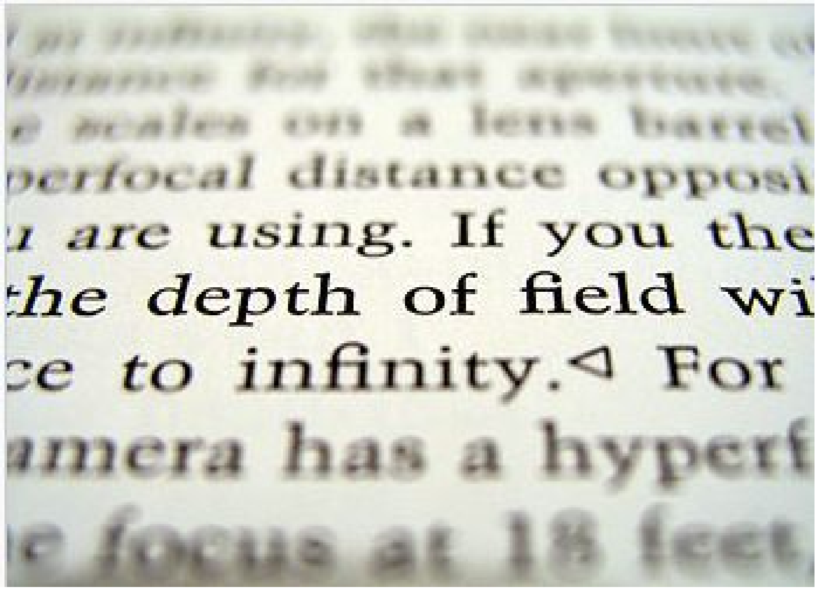


But it the light beams coming from different distances are not focused on the same plane.
These points will appear blurry in the resulting image.

Camera needs to move lens to focus an image on the sensor.

Depth of field

- Depth of field – range of depths that provides sufficient focus

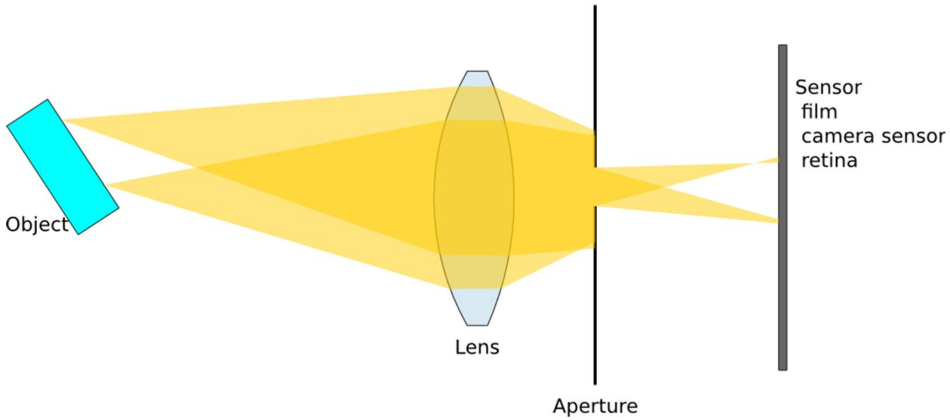


Defocus blur is often desirable

To separate the object of interest from background

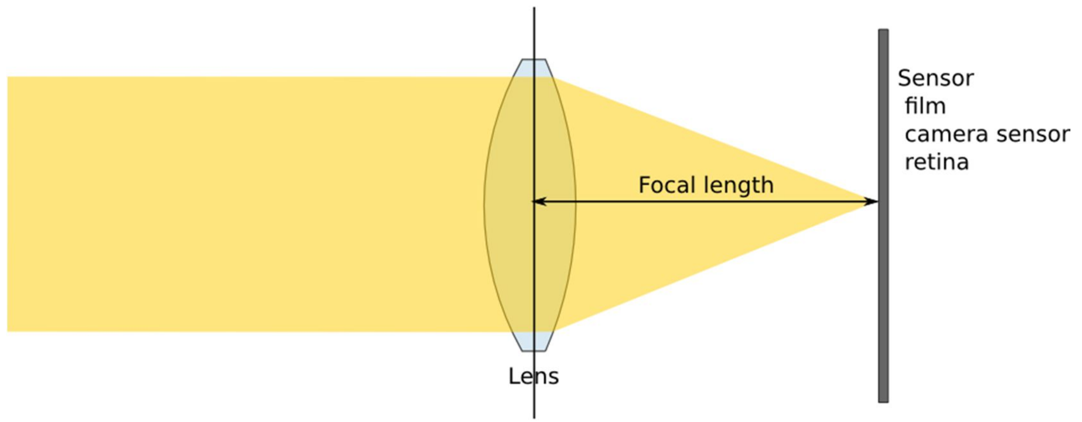
Defocus blur is a strong depth cue

Imaging – aperture



Aperture (introduced behind the lens) reduces the amount of light reaching sensor, but it also reduces blurriness from defocus (increases depth-of-field).

Imaging – lens



Focal length – length between the sensor and the lens that is needed to focus light coming from an infinite distance.

Larger focal length of a lens – more or less magnification?

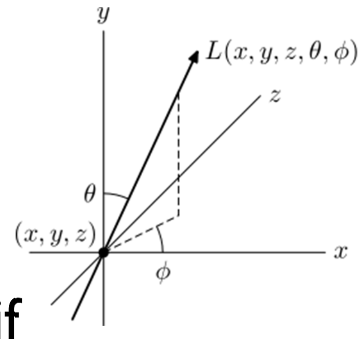
Light fields

Rafał Mantiuk

Computer Laboratory, University of Cambridge

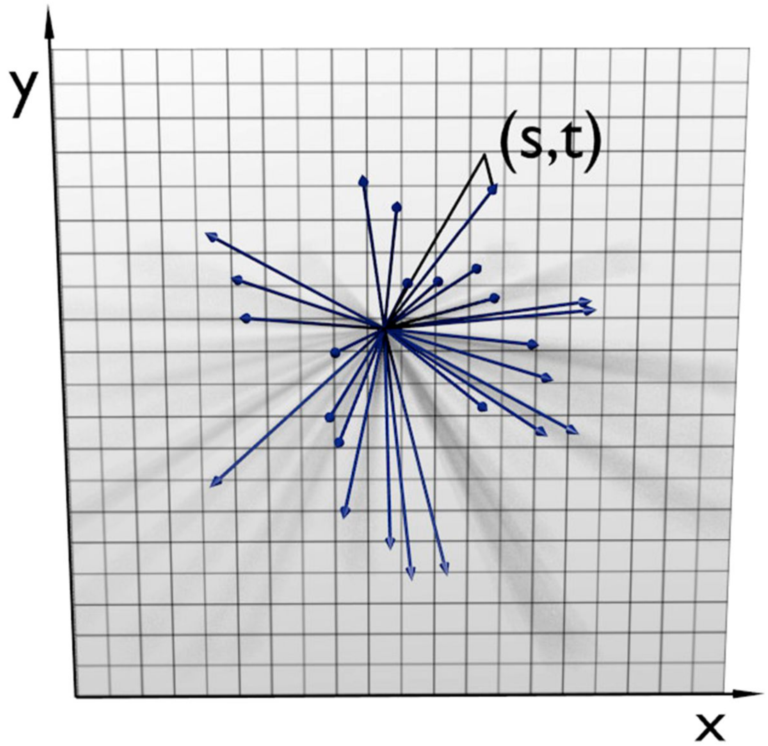
From a plenoptic function to a light field

- Plenoptic function – describes all possible rays in a 3D space
  - Function of position (x, y, z) and ray direction (θ, ϕ)
  - But also wavelength λ and time t
  - Between 5 and 7 dimensions



- But the number of dimensions can be reduced if
  - The camera stays outside the convex hull of the object
  - The light travels in uniform medium
  - Then, radiance L remains the same along the ray (until the ray hits an object)
  - This way we obtain a **4D light field**

Planar 4D light field

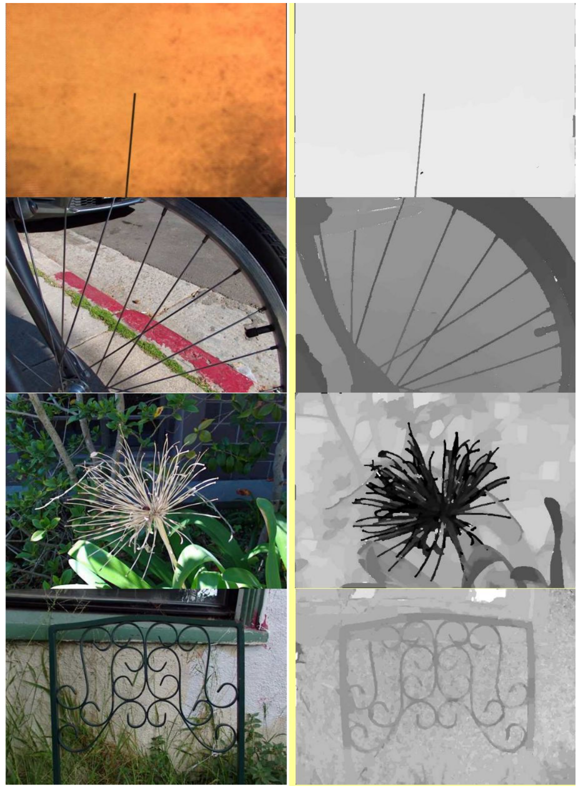


Refocusing and view point adjustment

Screen capture from http://www.lytro.com/

Depth estimation from light field

- Passive sensing of depth
- Light field captures multiple depth cues
  - Correspondance (disparity) between the views
  - Defocus
  - Occlusions



From: ***Ting-Chun Wang, Alexei A. Efros, Ravi Ramamoorthi***; The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3487-3495

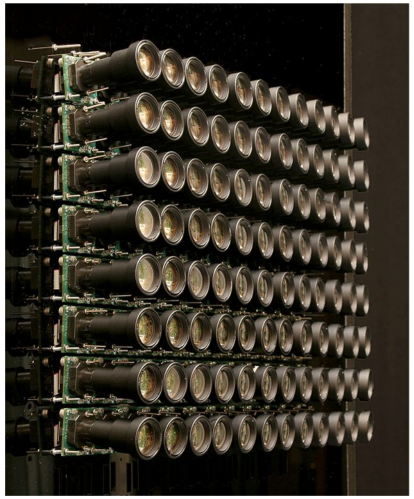
Two methods to capture light fields

Micro-lens array

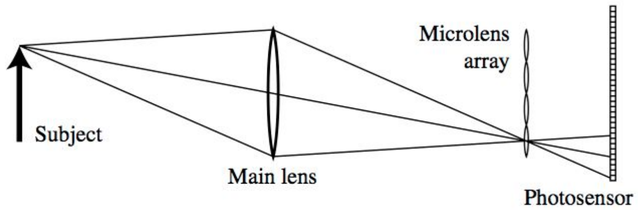
- Small baseline
- Good for digital refocusing
- Limited resolution

Camera array

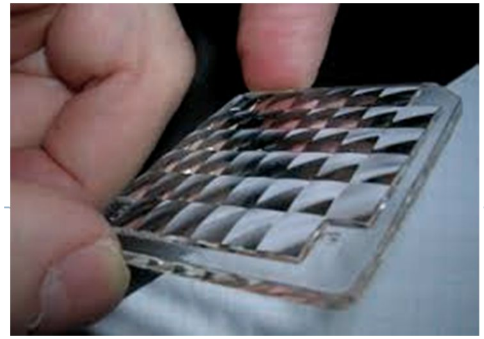
- Large baseline
- High resolution
- Rendering often requires approximate depth



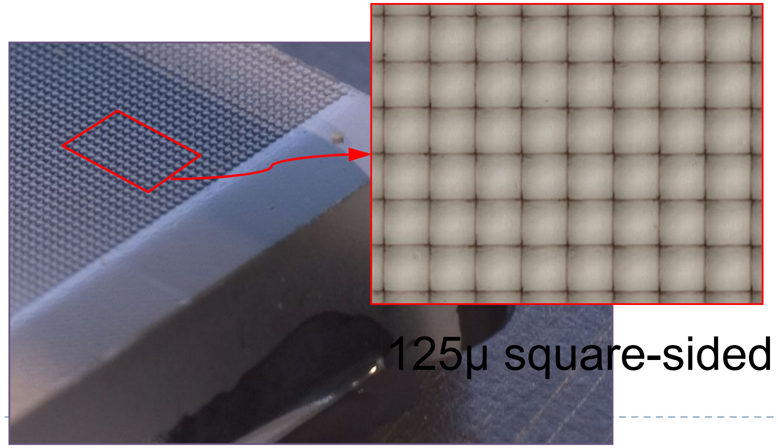
Light field image – with microlens array



Digital Refocusing using Light Field Camera



Lenslet array



125μ square-sided microlenses

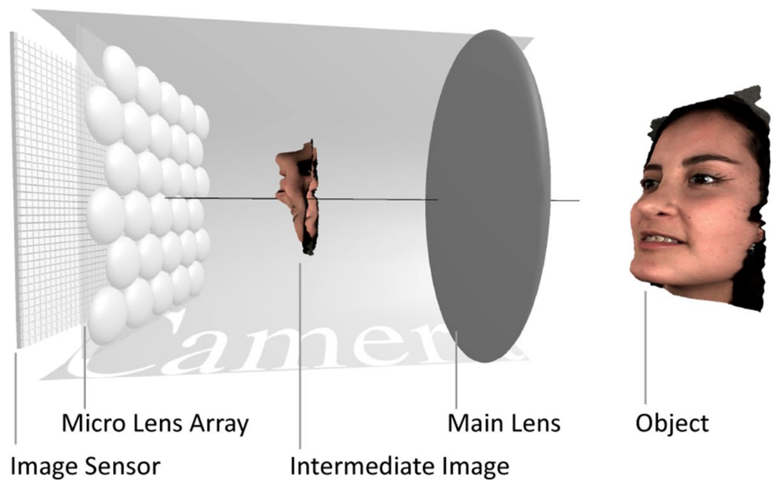
[Ng et al 2005]

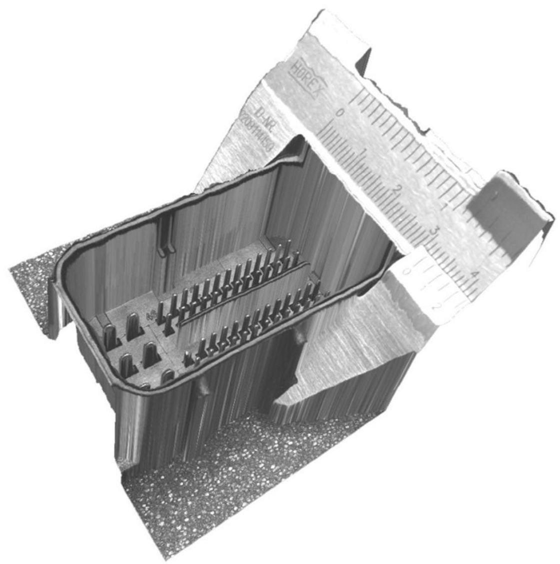
Lytro-cameras

- First commercial light-field cameras
- Lytro illum camera
  - 40 Mega-rays
  - 2D resolution: 2450 x 1634 (4 MPixels)

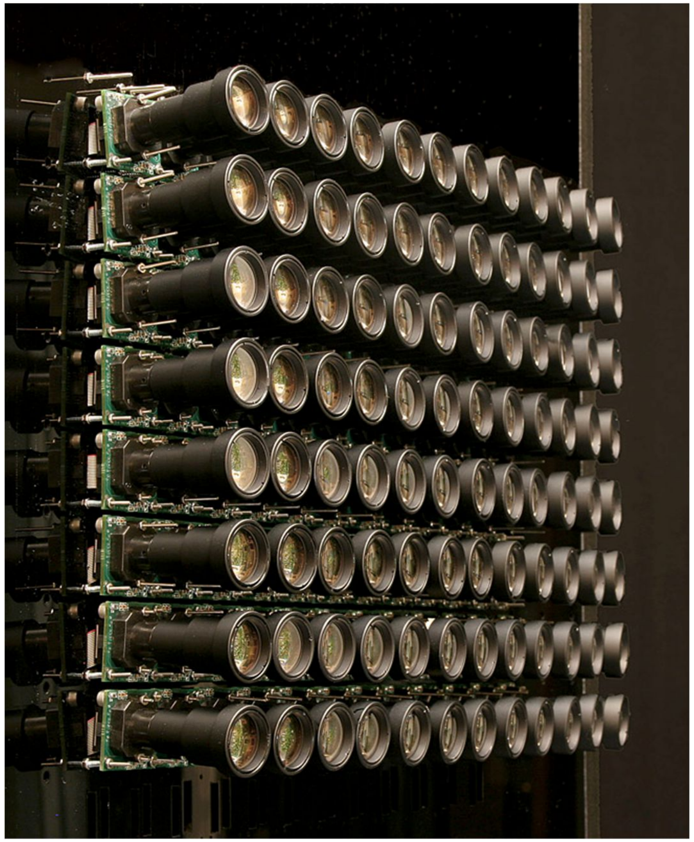
Raytrix camera

- Similar technology to Lytro
- But profiled for computer vision applications



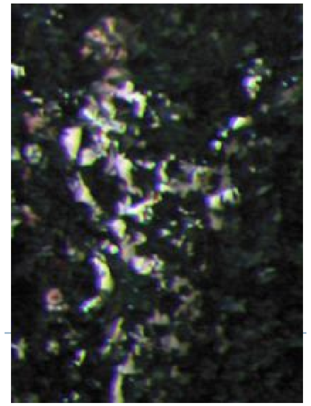


Stanford camera array



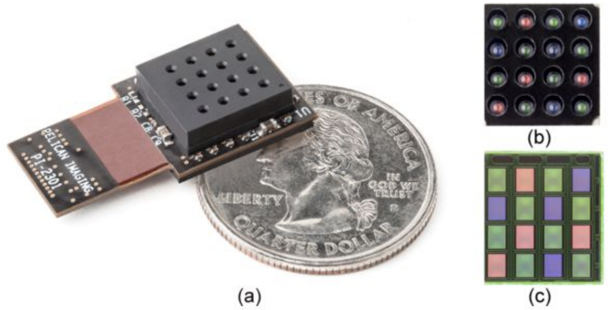
96 cameras

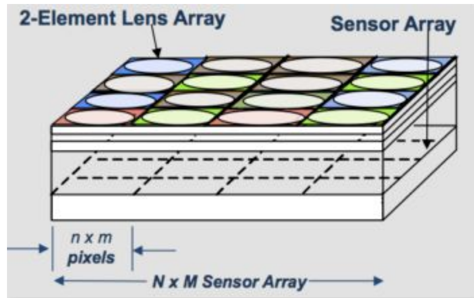
Application: Reconstruction of occluded surfaces

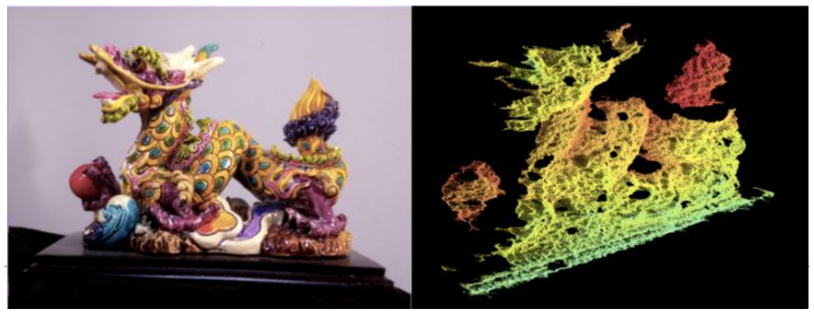


PiCam camera array module

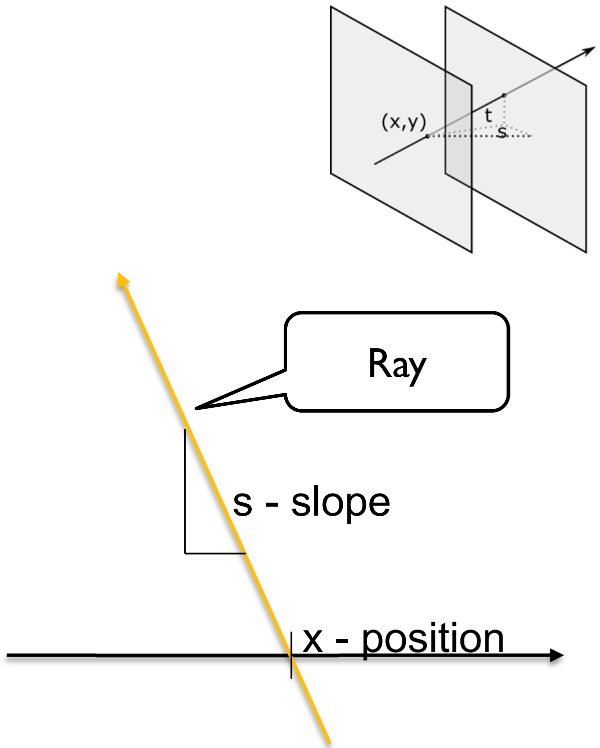
- Array of 4 x 4 cameras on a single chip
- Each camera has its own lens and senses only one spectral colour band
  - Optics can be optimized for that band
- The algorithm needs to reconstruct depth



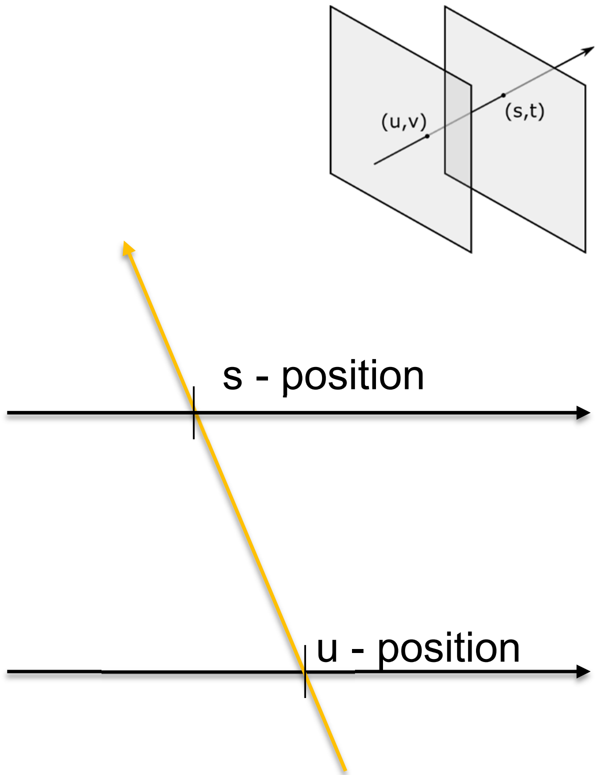




Light fields: two parametrisations (shown in 2D)

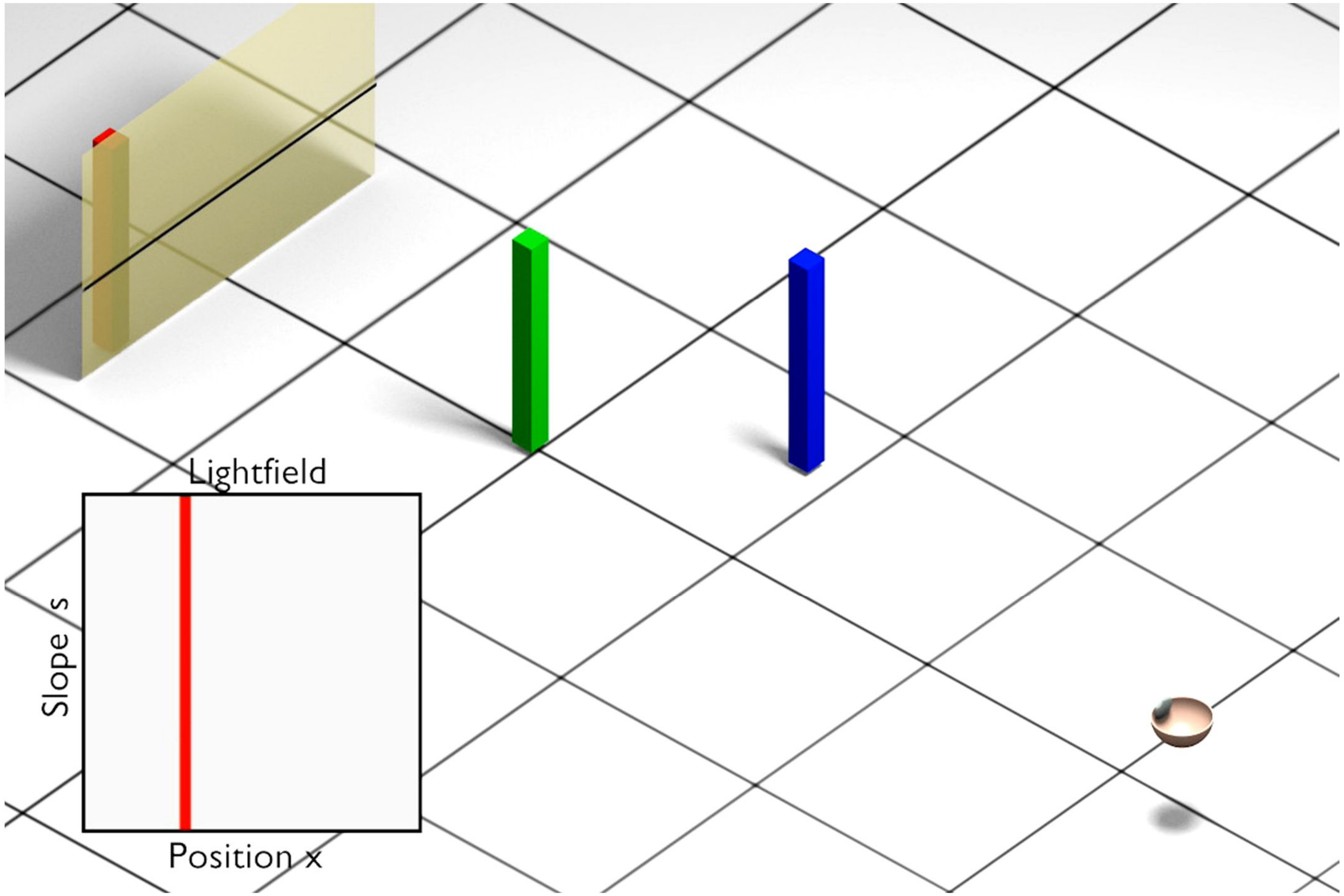


Position and slope
(slope - tangent of the angle)

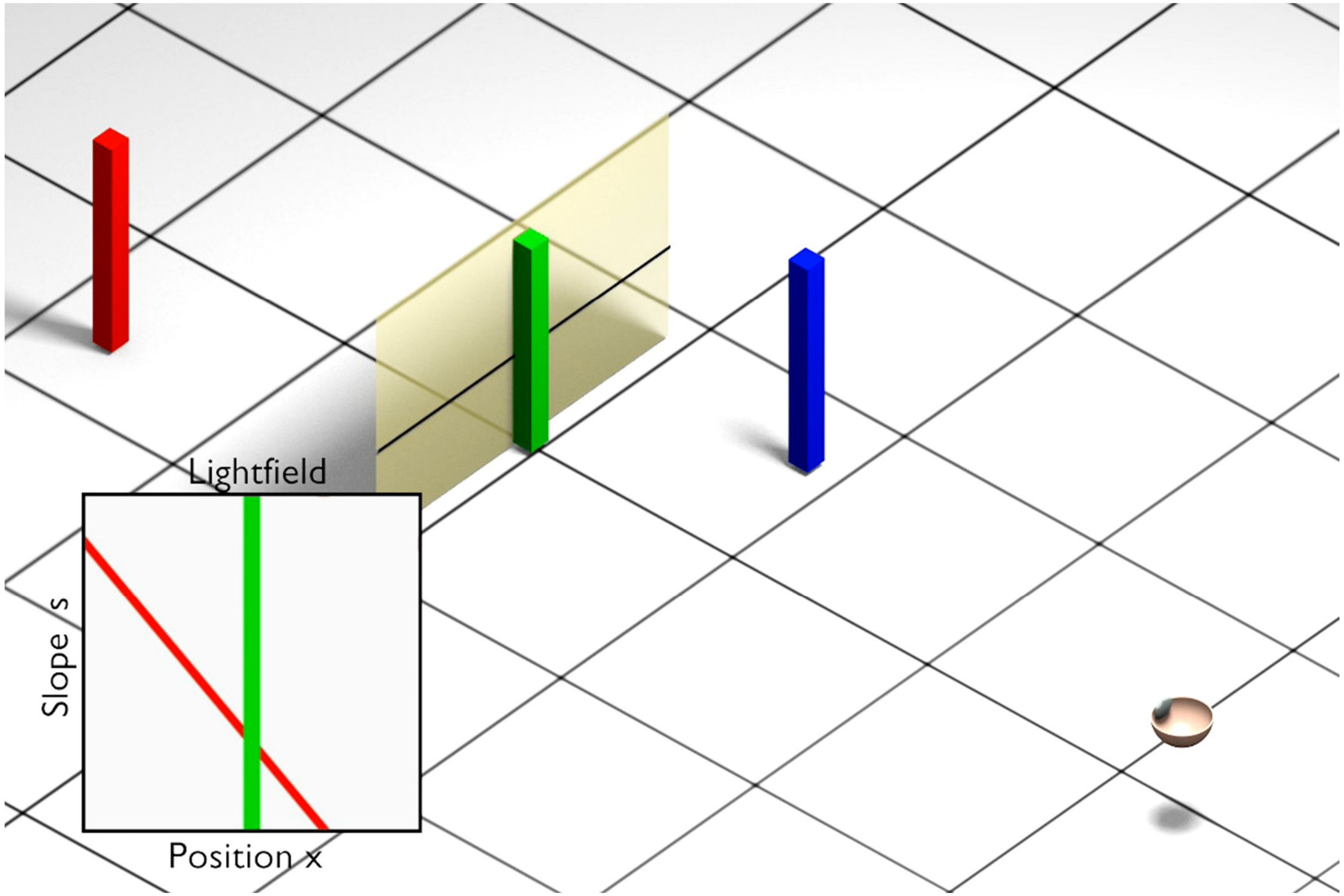


Two planes

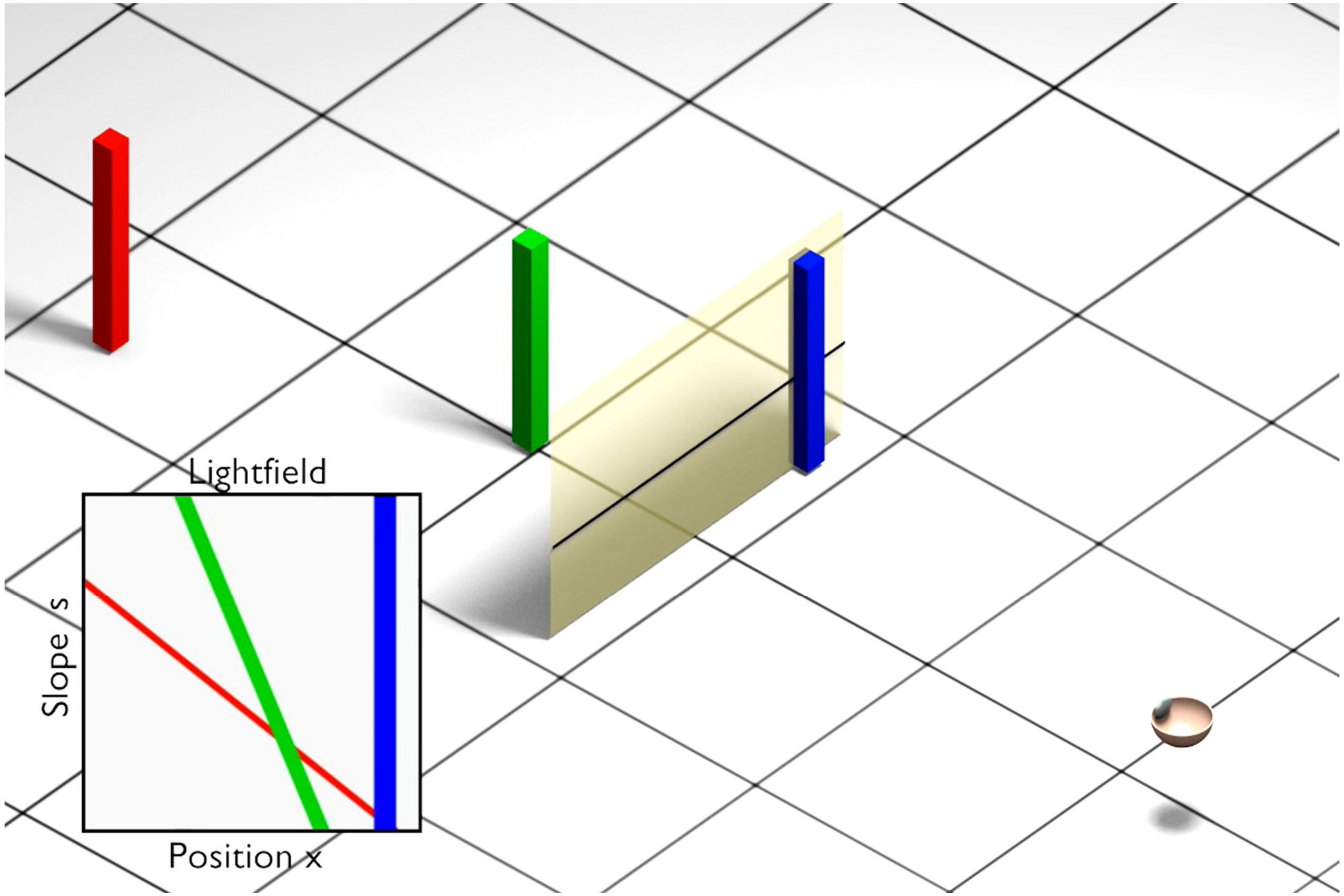
Lightfield - example



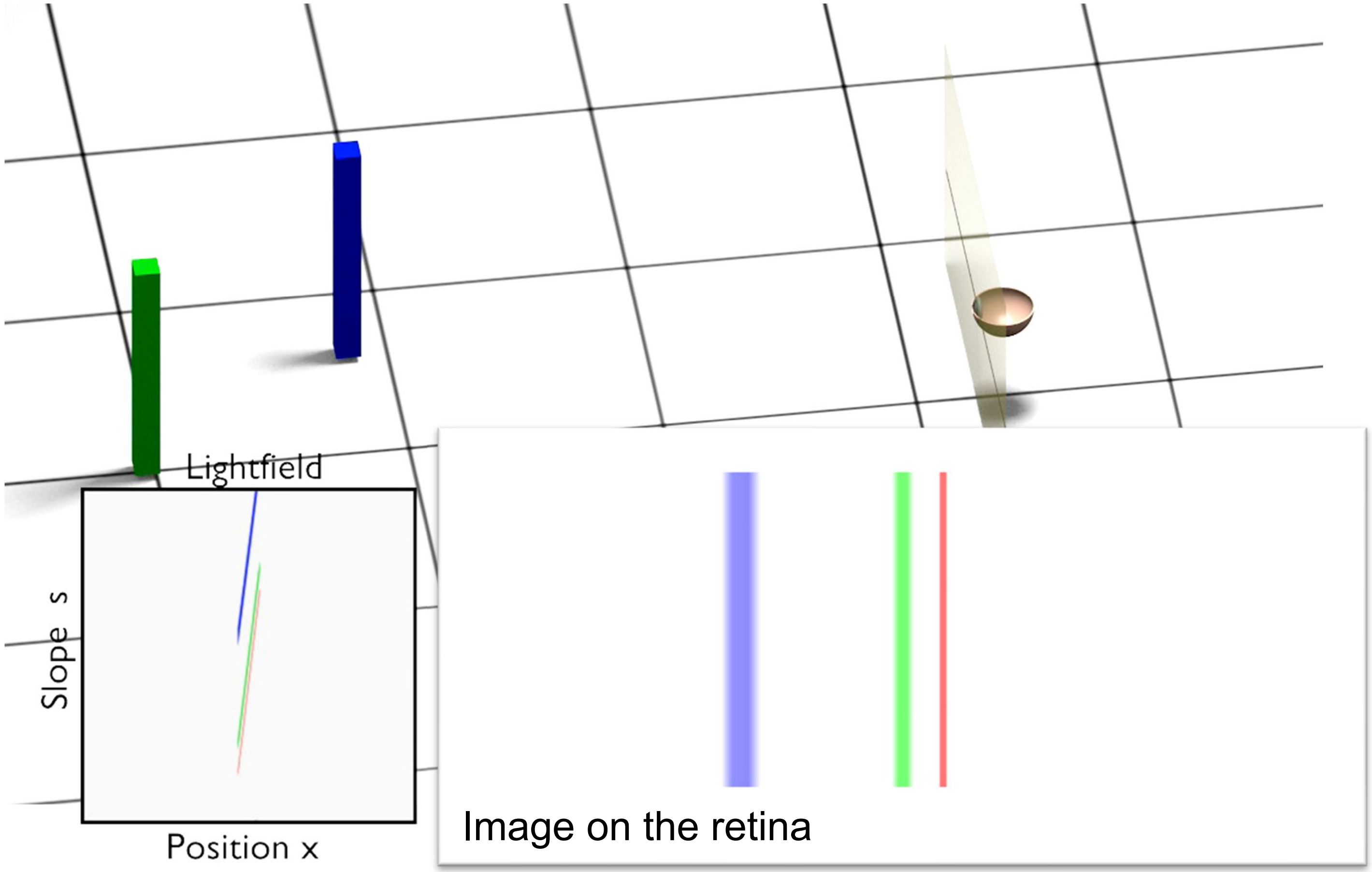
Lightfield - example



Lightfield - example



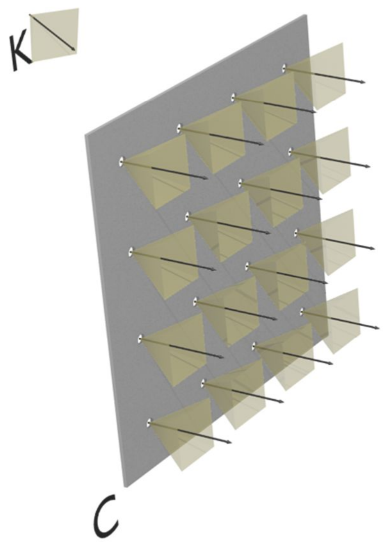
Lightfield - example



Light field rendering

Rafał Mantiuk
Computer Laboratory, University of Cambridge

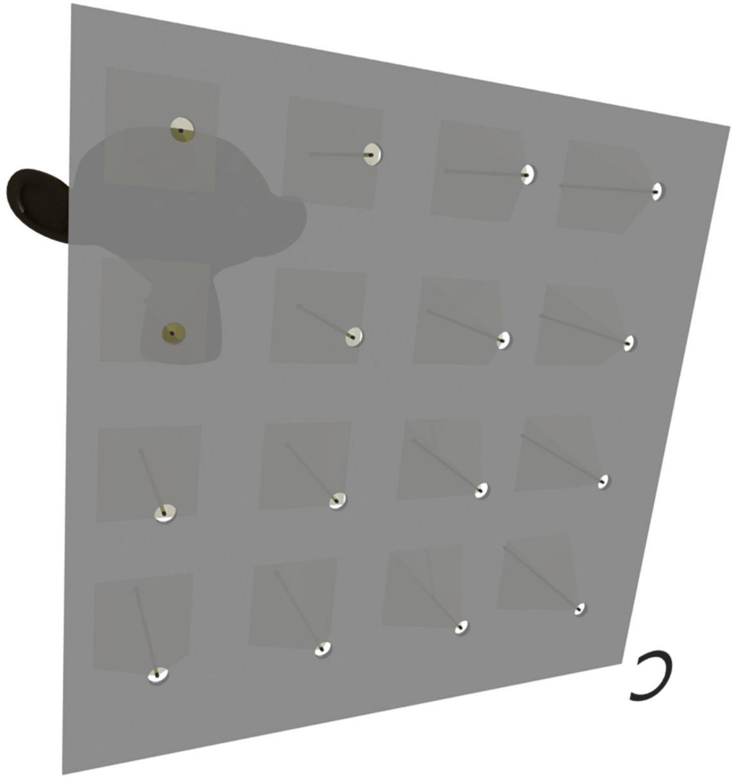
Light field rendering (1/3)



We want to render a scene (Blender monkey) as seen by camera K. We have a light field captured by a camera array. Each camera in the array has its aperture on plane C.

Light field rendering (2/3)

From the viewpoint of camera K

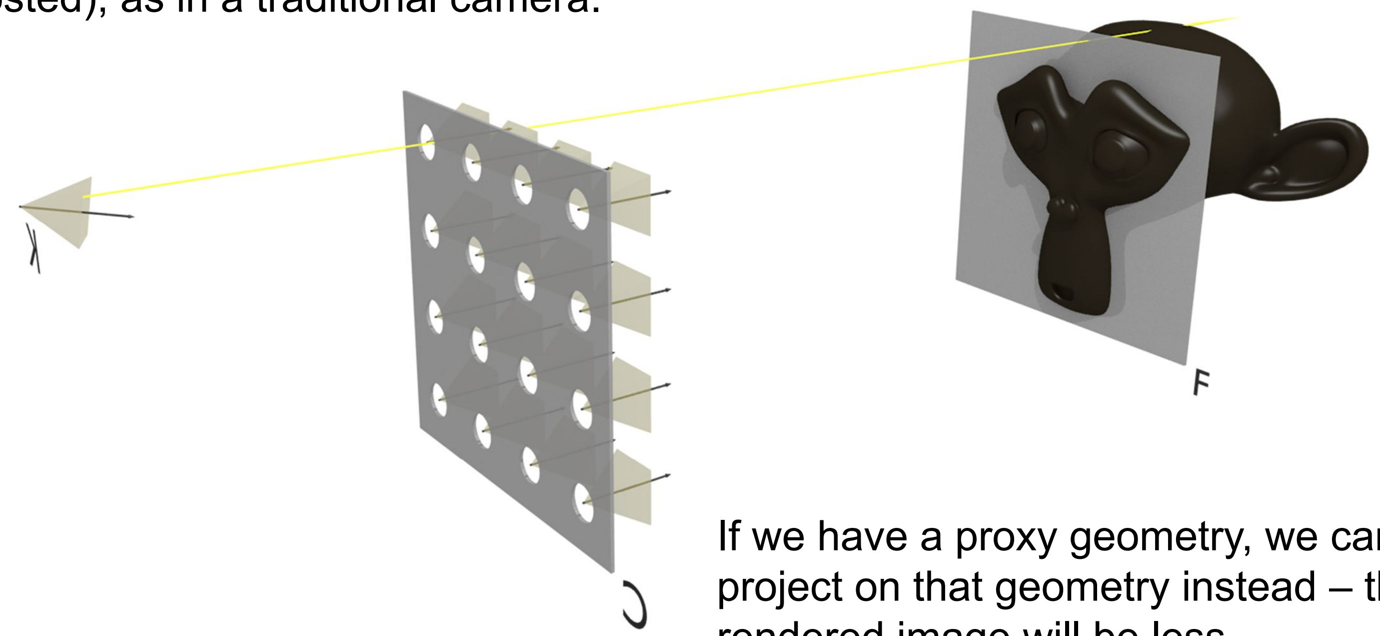


Each camera in the array provides accurate light measurements only for the rays originating from its pinhole aperture.

The missing rays can be either interpolated (reconstructed) or ignored.

Light field rendering (3/3)

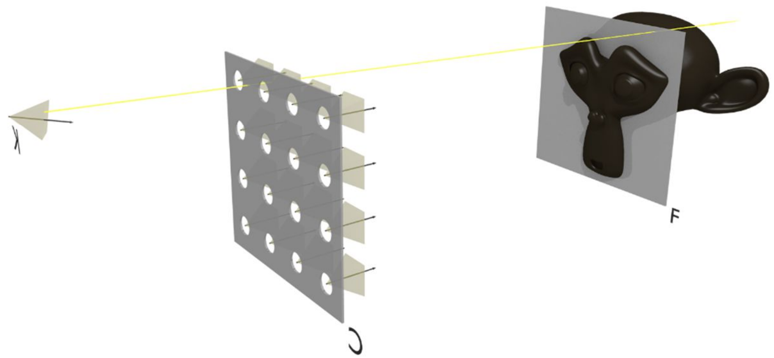
The rays from the camera need to be projected on the focal plane F. The objects on the focal plane will be sharp, and the objects in front or behind that plane will be blurry (ghosted), as in a traditional camera.



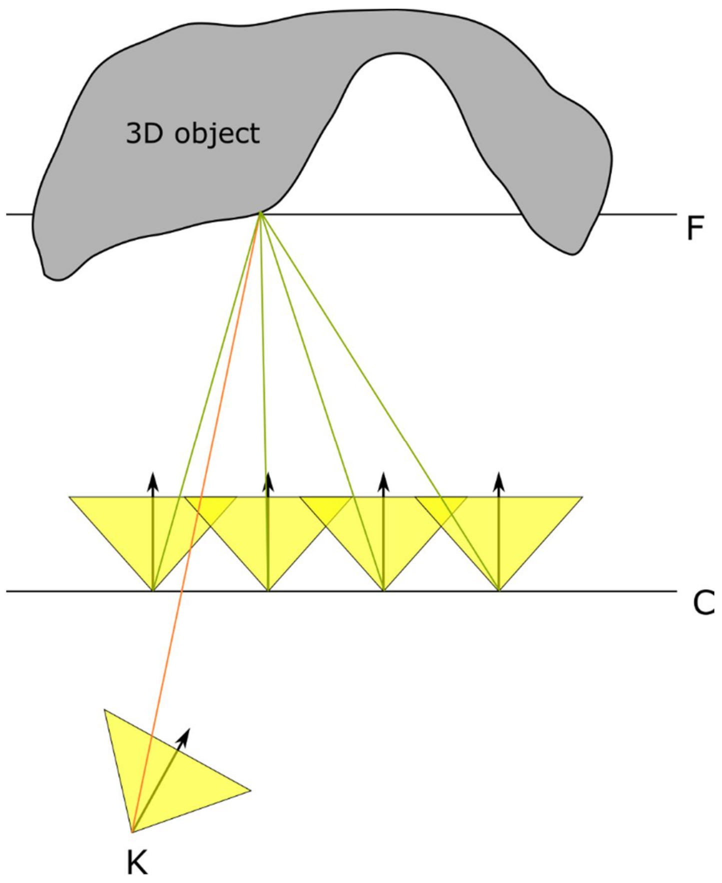
If we have a proxy geometry, we can project on that geometry instead – the rendered image will be less ghosted/blurry

Intuition behind light field rendering

- For large virtual aperture (use all cameras in the array)
  - Each camera in the array captures the scene
  - Then, each camera projects its image on the focal plane F
  - The virual camera K captures the projection
- For small virtual aperture (pinhole)
  - For each ray from the virtual camera
    - interpolate rays from 4 nearest camera images
  - Or use the nearest-neighbour ray

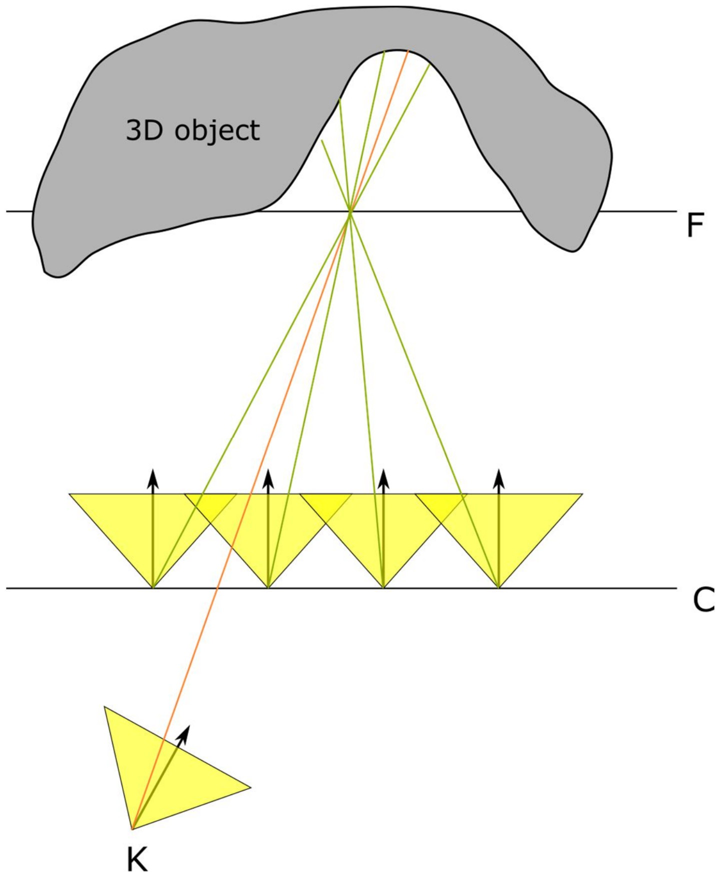


LF rendering – focal plane



- For a point on the focal plane, all cameras capture the same point on the 3D object
- They also capture approximately the same colour (for diffuse objects)
- Averaged colour will be the colour of the point on the surface

LF rendering – focal plane

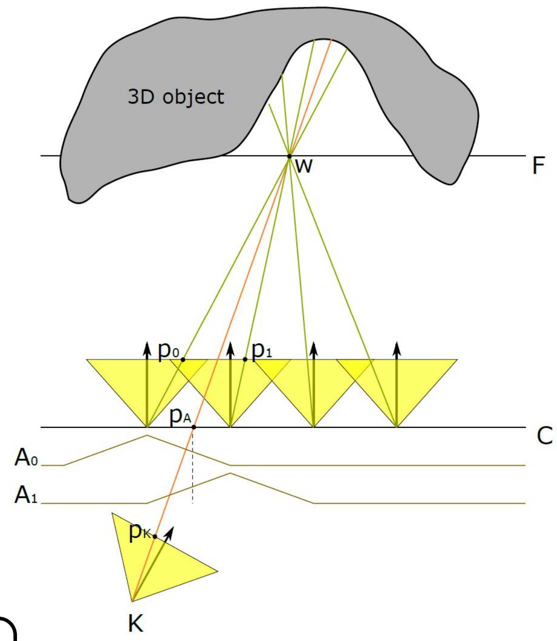


- If the 3D object does not lie on the focal plane, all camaras capture different points on the object
- Averaging colour values will produce a „ghosted" image
- If we had unlimited number of cameras, this would produce a depth-of-field effect

Finding homographic transformation 1/3

- For the pixel coordinates $\boldsymbol{p}_k$ of the virtual camera K, we want to find the corresponding coordinates $\boldsymbol{p}_i$ in the camera array image
- Given the world 3D coordinates of a point $\boldsymbol{w}$:

$$\boldsymbol{p}_i = \underbrace{\boldsymbol{K}}_{\text{Intrinsic camera matrix}} \underbrace{\boldsymbol{P}}_{\text{Projection matrix}} \underbrace{\boldsymbol{V}_i}_{\text{View matrix}} \boldsymbol{w}$$



$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Finding homographic transformation 2/3

- A homography between two views is usually found as:

$$\boldsymbol{p}_K = \boldsymbol{K}_K \boldsymbol{P} \boldsymbol{V}_K \boldsymbol{w}$$
$$\boldsymbol{p}_i = \boldsymbol{K}_i \boldsymbol{P} \boldsymbol{V}_i \boldsymbol{w}$$

hence

$$\boldsymbol{p}_i = \boldsymbol{K}_i \boldsymbol{P} \boldsymbol{V}_i \boldsymbol{V}_K^{-1} \boldsymbol{P}^{-1} \boldsymbol{K}_K^{-1} \boldsymbol{p}_K$$

- But, $\boldsymbol{K}_K \boldsymbol{P} \boldsymbol{V}_K$ is not a square matrix and cannot be inverted
  - To find the correspondence, we need to constrain 3D coordinates $\boldsymbol{w}$ to lie on the plane:

$$\boldsymbol{N} \cdot (\boldsymbol{w} - \boldsymbol{w}_F) = 0 \quad \text{or} \quad d = \begin{bmatrix} n_x & n_y & n_z & -\boldsymbol{N} \cdot \boldsymbol{w}_F \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Finding homographic transfo

The plane in
the camera coordinates
(not world coordinates)

- Then, we add the plane equation to the projection matrix

$$\underbrace{\begin{bmatrix} x_i \\ y_i \\ d_i \\ w_i \end{bmatrix}}_{\hat{\boldsymbol{p}}_i} = \underbrace{\begin{bmatrix} f_x & 0 & 0 & c_x \\ 0 & f_y & 0 & c_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\hat{\boldsymbol{K}}_i} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ n_x^{(c)} & n_y^{(c)} & n_z^{(c)} & -\boldsymbol{N}^{(c)} \cdot \boldsymbol{w}_F^{(c)} \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\hat{\boldsymbol{P}}} \underbrace{\begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{V}_i} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\boldsymbol{w}}$$

- Where d_i is the distance to the plane
- Hence

$$\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{K}}_i \hat{\boldsymbol{P}} \boldsymbol{V}_i \boldsymbol{V}_K^{-1} \hat{\boldsymbol{P}}^{-1} \hat{\boldsymbol{K}}_K^{-1} \hat{\boldsymbol{p}_K}$$

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Neural radiance fields

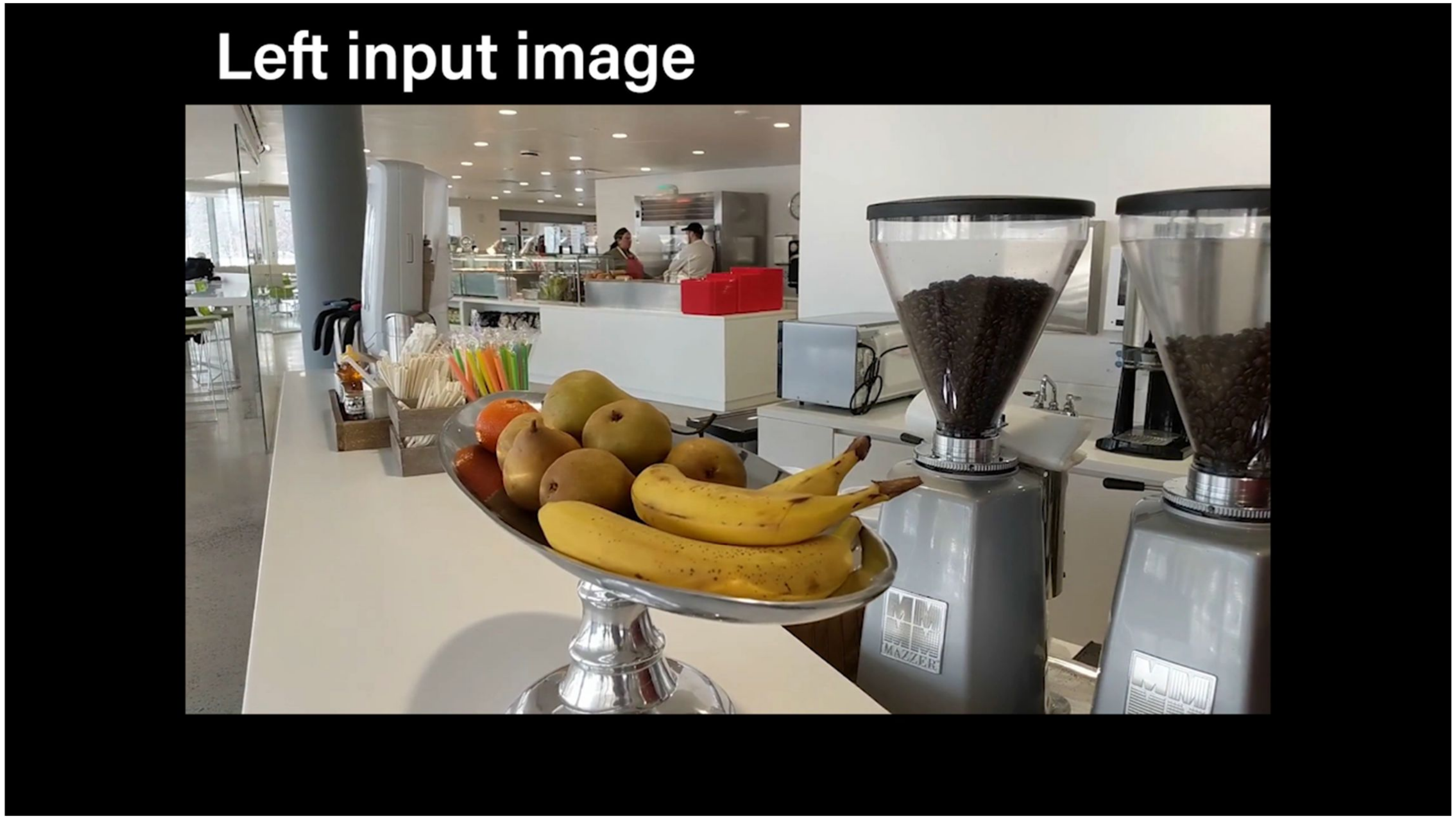
differentiable volumetric rendering

Rafał Mantiuk

Computer Laboratory, University of Cambridge

Stereo magnification:
learning view synthesis using multiplane images

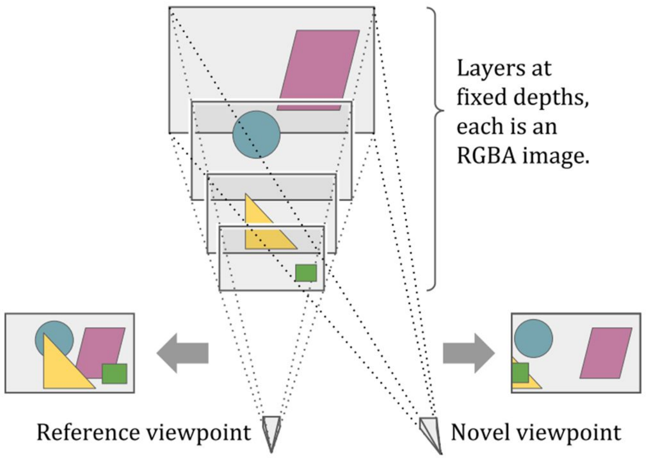
- Synthetize motion parallax from two (stereo) views

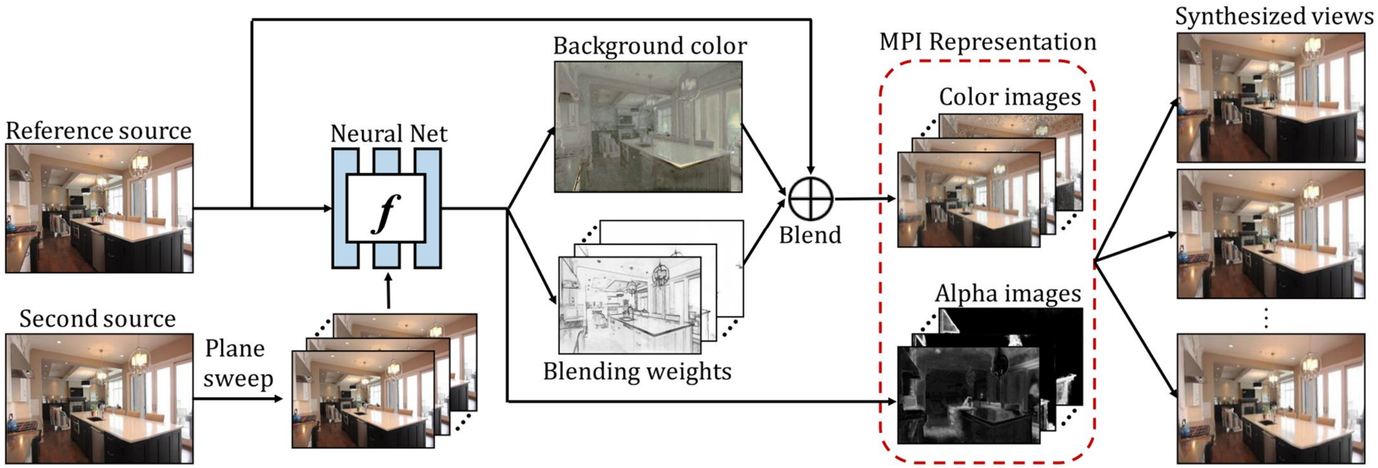


Zhou, Tinghui, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. “Stereo Magnification: Learning View Synthesis Using Multiplane Images.” *ACM Transactions on Graphics* 37, no. 4 (August 31, 2018): 1–12. https://doi.org/10.1145/3197517.3201323.

Stereo magnification: learning view synthesis using multiplane images

- Goal: decompose images into multiple planes with an alpha channel (MPI)
- Intermediate representation: background and foreground images
  - To better handle occlusions
  - The network is overfitted to each scene

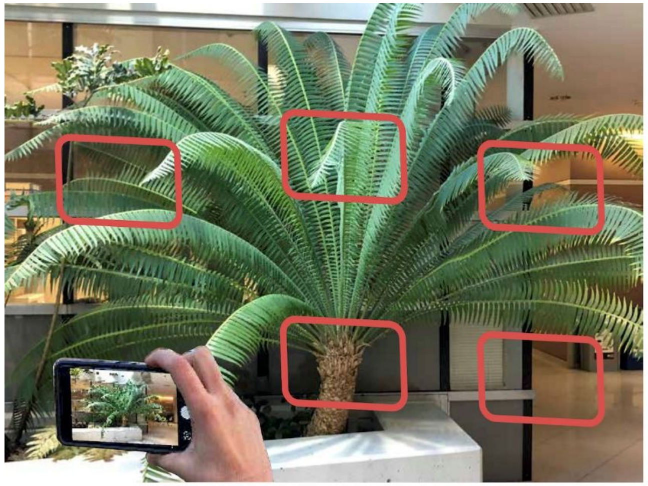




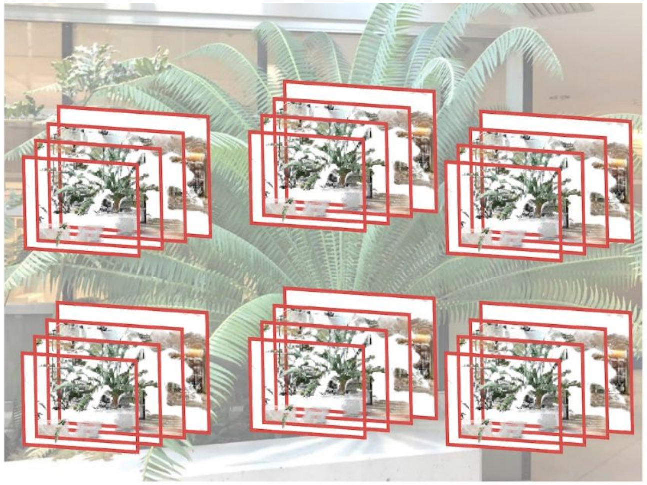
Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines

- Reconstruct multiple MPIs, then blend them
- This is to better capture view-dependent effects
  - E.g. specular reflections

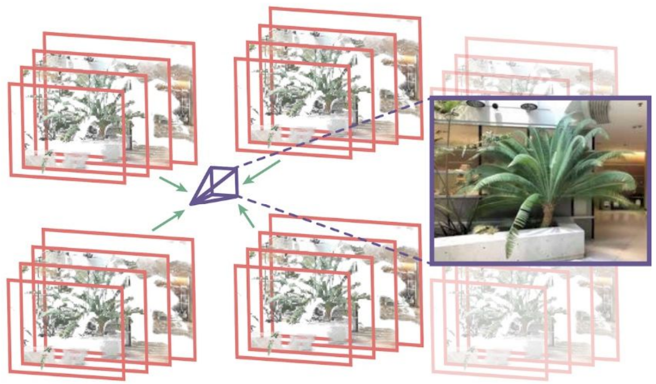
Mildenhall, Ben et al. "Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines." *ACM Trans. on Graphics* 38, no. 4 (July 12, 2019): 1–14. https://doi.org/10.1145/3306346.3322980.



Fast and easy handheld capture with guideline: closest object moves at most D pixels between views



Promote sampled views to local light field via layered scene representation



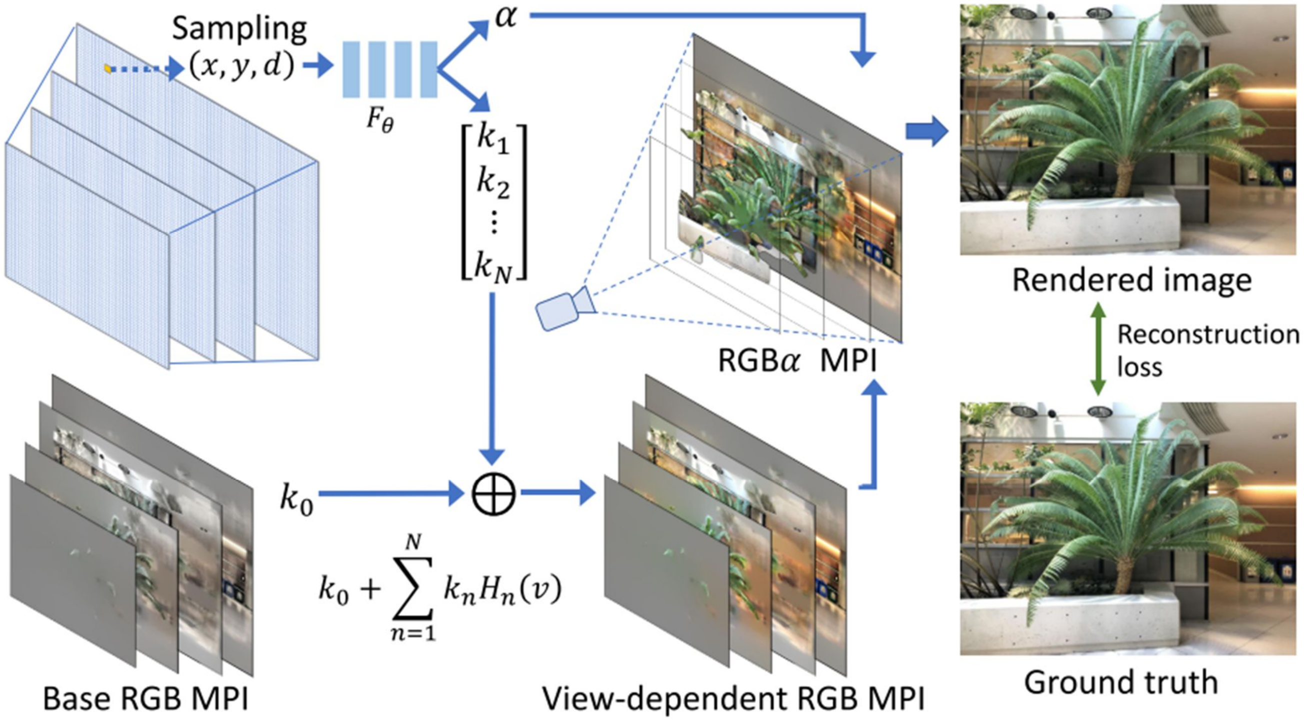
Blend neighboring local light fields to render novel views

NeX: Real-time View Synthesis with Neural Basis Expansion

- MPI + view-dependent color encoding
- High quality reproduction of the view-dependent effects
  - Specular reflections
  - Diffraction
  - …

NeX: Real-time View Synthesis with Neural Basis Expansion

- The color is encoded as a linear combination of the basis functions
- The basis functions are trainable

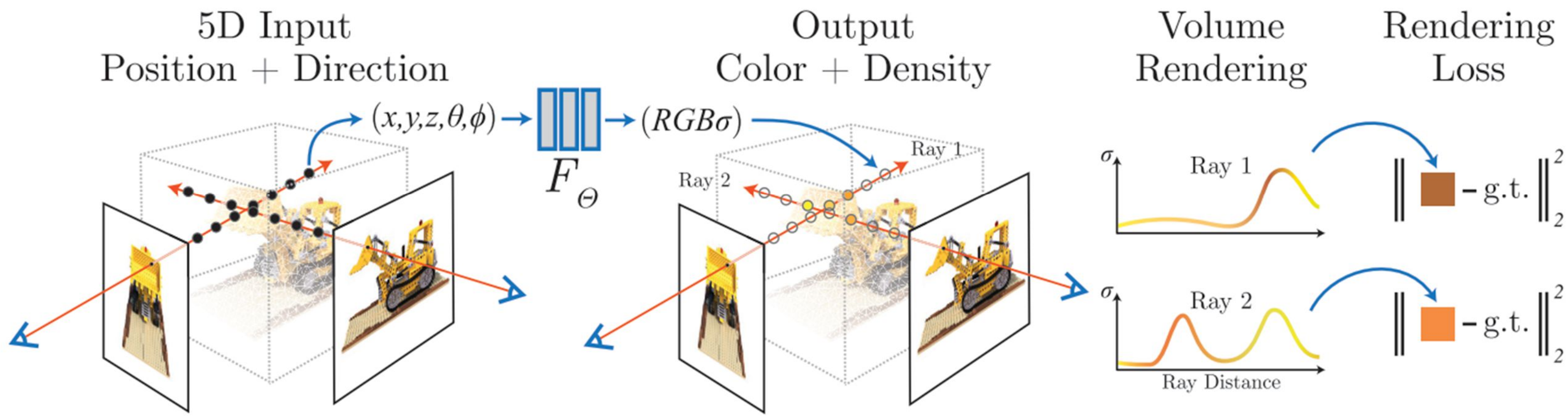


NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Models a volume rather than a set of discrete planes
- 360 or front facing
- Uses MLP to represent the colour and opacity

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," 405–21, 2020. https://doi.org/10.1007/978-3-030-58452-8_24.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



Pixel colour

(x,y,z) coordinates along the ray

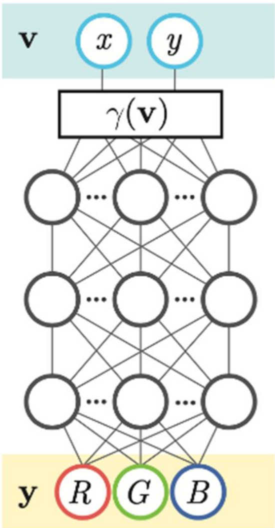
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Computed as a (differentiable) stratified sampling

Colour at (x,y,z) in the direction d (stored in an MLP)

"opacity" or probability of ray terminating (stored in an MPL)

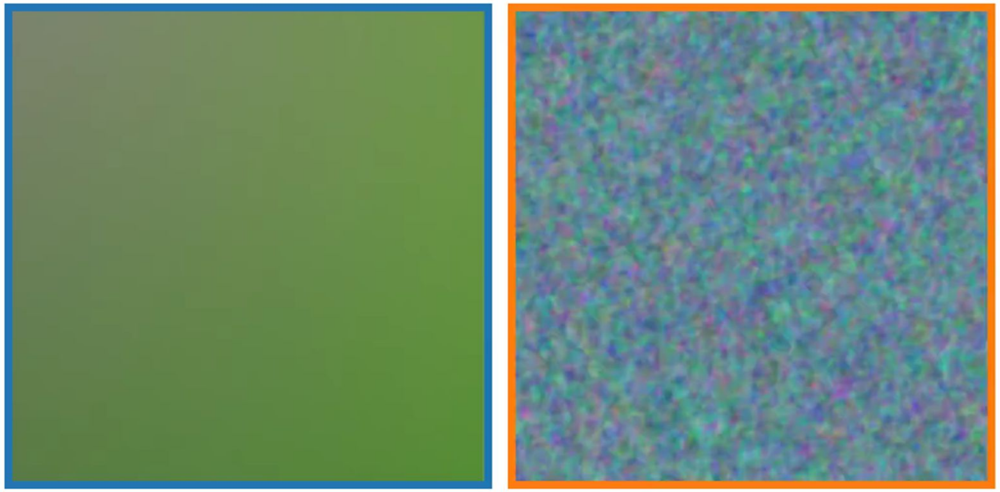
Positional encoding

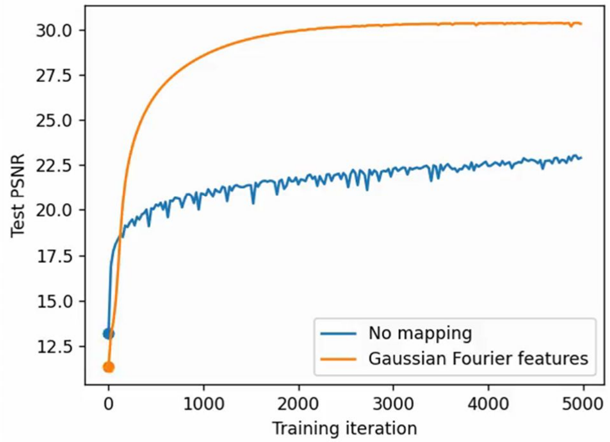


$$y = f(\gamma(p); w)$$

$$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right) \right)$$

- Encoding coordinates as the Fourier "features" allows MPL to learn high frequencies
- Works with other basis functions

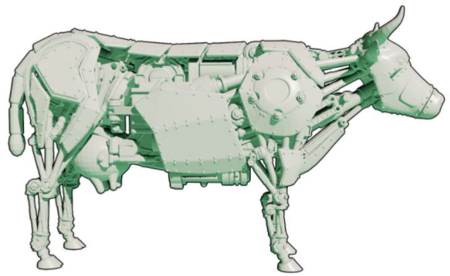


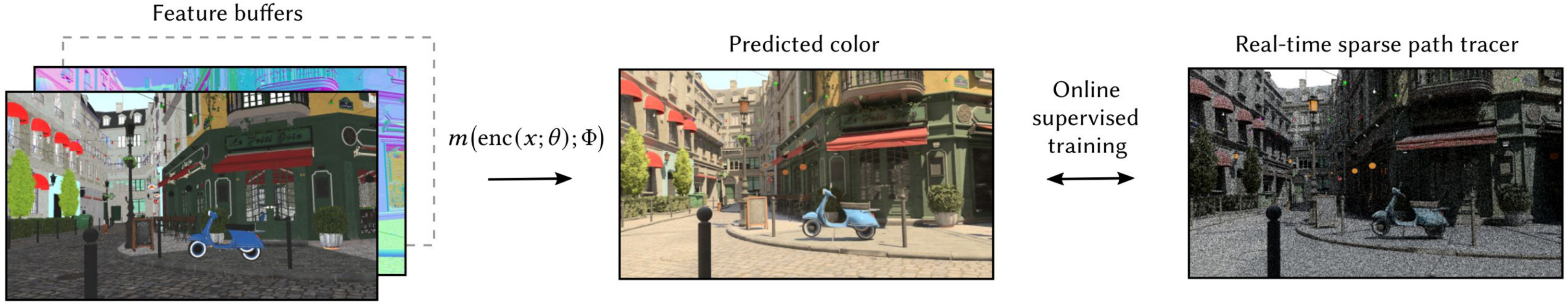


No positional encoding With positional encoding

Implicit (neural) (volumetric/n-dim) representations

- Neural signed distance function
  - A function that stores a distance to a surface
  - $d = f(x, y, z; \boldsymbol{\phi})$
- Neural radiance caching
  - Predict colour from feature buffers independently for each pixel

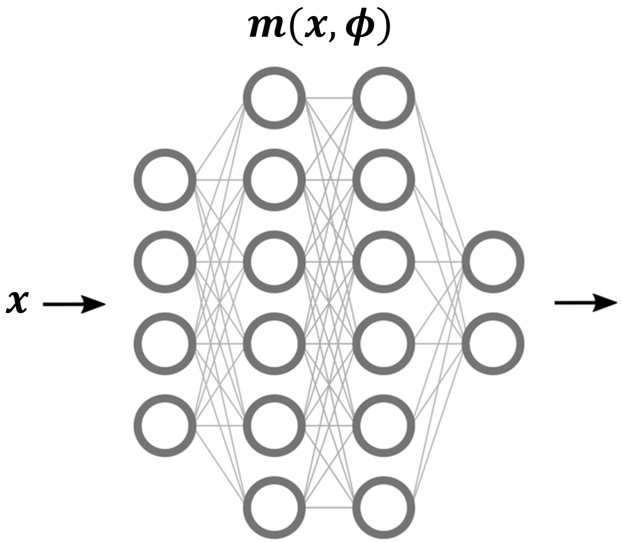




- Learning a giga-pixel image
  - $RGB = f(x, y; \boldsymbol{\phi})$

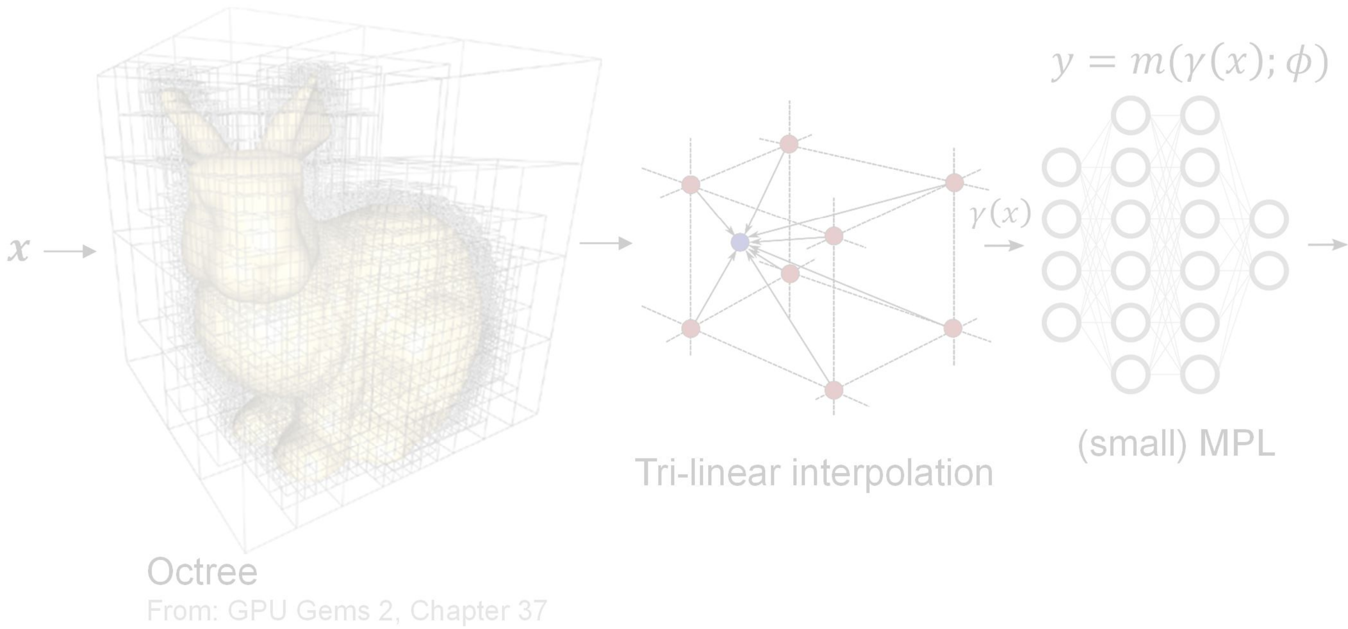
Illustrations from: https://dl.acm.org/doi/10.1145/3528223.3530127

Reducing the cost of the MLP



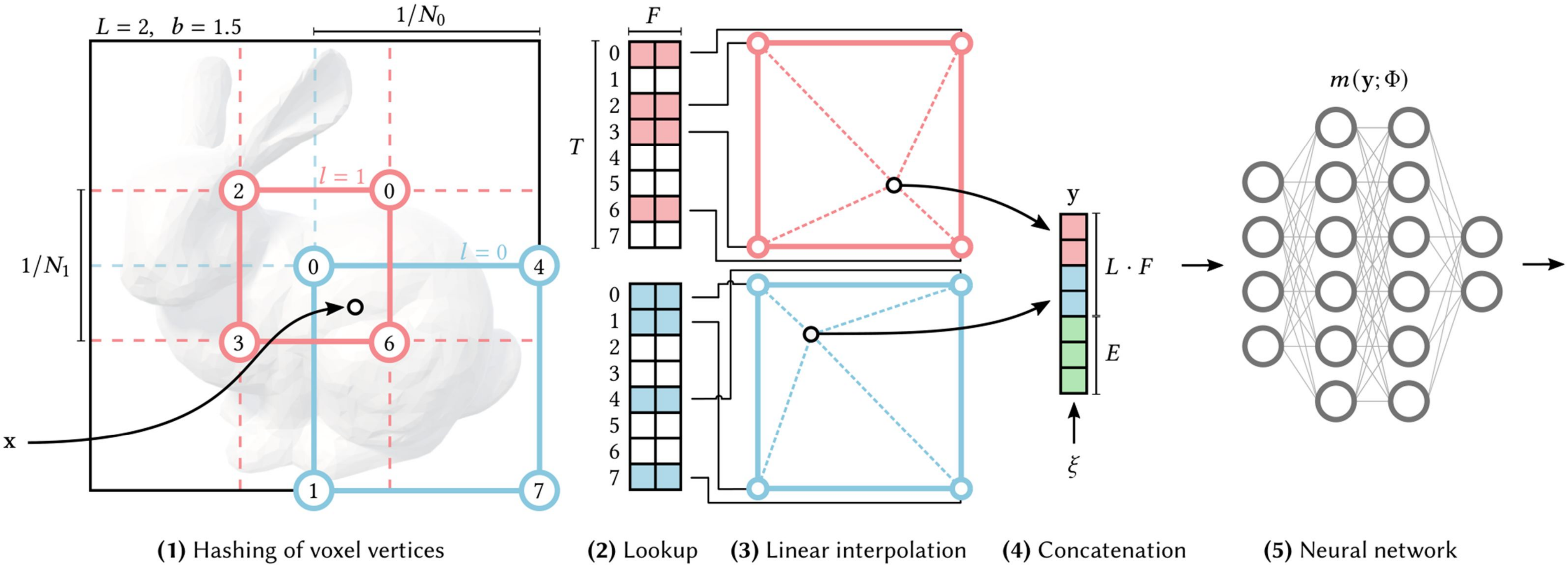
Given input coordinates $\boldsymbol{x}$, only a small portion of the network activations will contribute to the output. This is inefficient.

Solution: encode a part of the information in a spatial data structure that can be directly querried, such as a (sparse) voxel grid or octree.



Instant neural graphics primitives with a multiresolution hash encoding

$$y = m(\gamma(x); \phi)$$



Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." *ACM Transactions on Graphics* 41, no. 4 (July 22, 2022): 1–15. https://doi.org/10.1145/3528223.3530127.

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

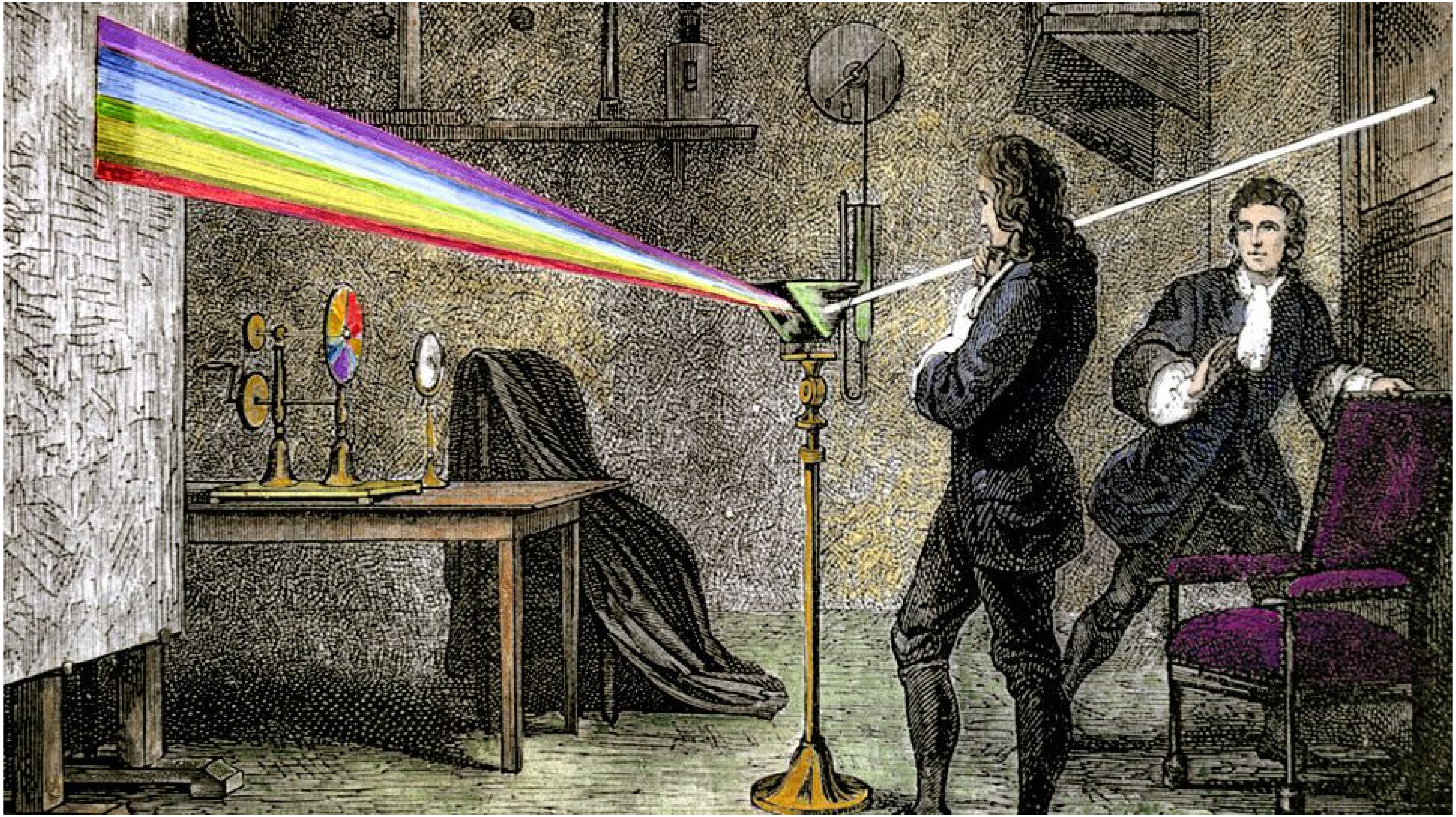
RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

Colour perception and colour spaces

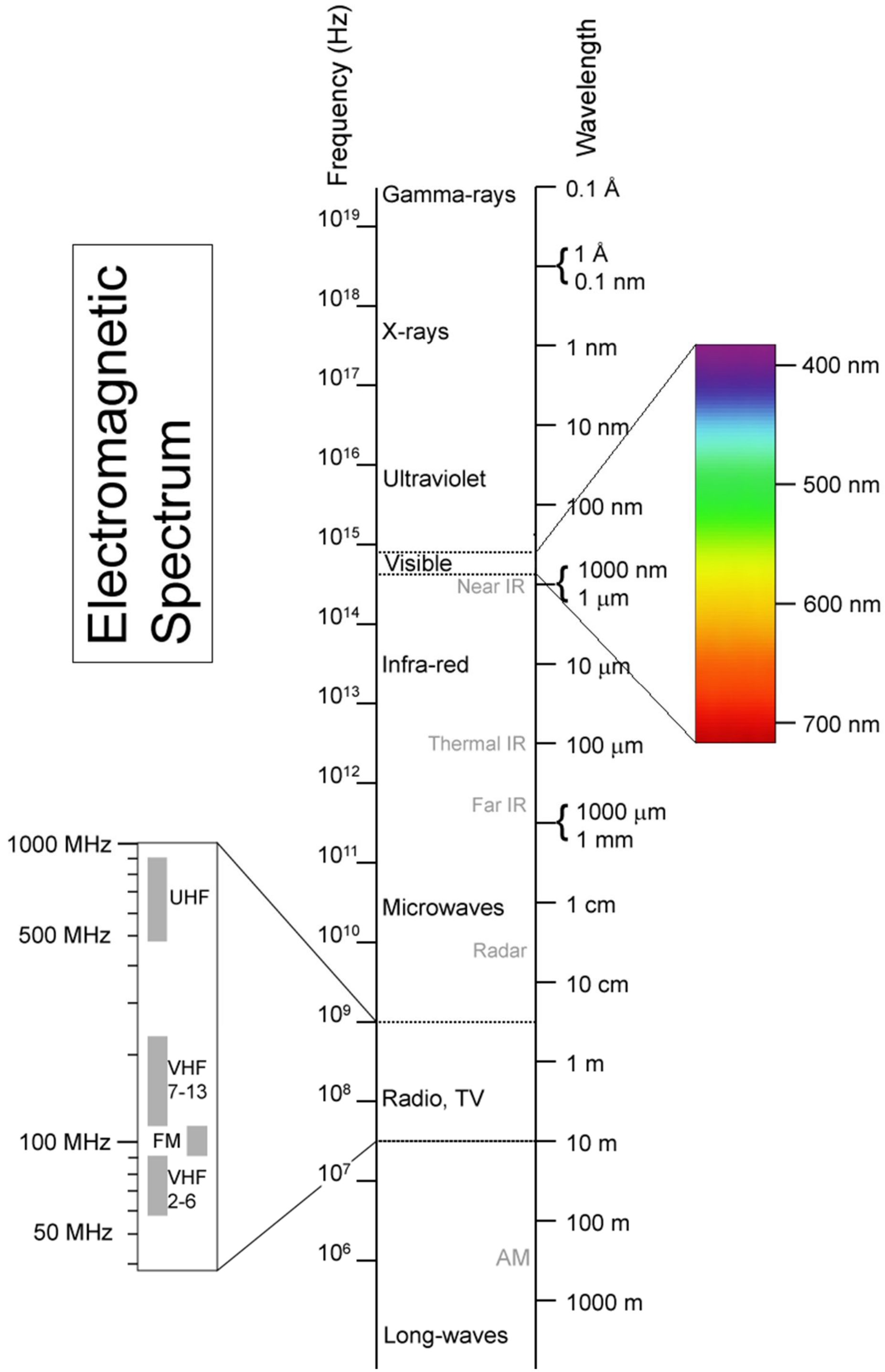
Part 1/5 – physics of light

Rafał Mantiuk
Computer Laboratory, University of Cambridge



Electromagnetic spectrum

- Visible light
  - Electromagnetic waves of wavelength in the range 380nm to 730nm
  - Earth's atmosphere lets through a lot of light in this wavelength band
  - Higher in energy than thermal infrared, so heat does not interfere with vision

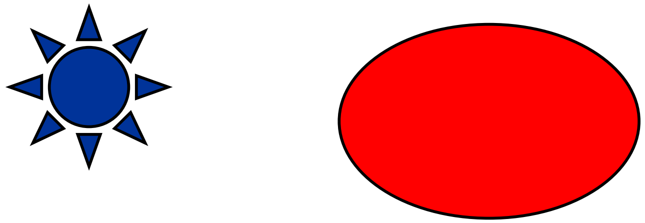


Colour

- There is no physical definition of colour – colour is the result of our perception

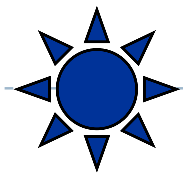
- For reflective displays / objects

$$\text{colour} = \text{perception}(\ \text{illumination} \times \text{reflectance}\ )$$



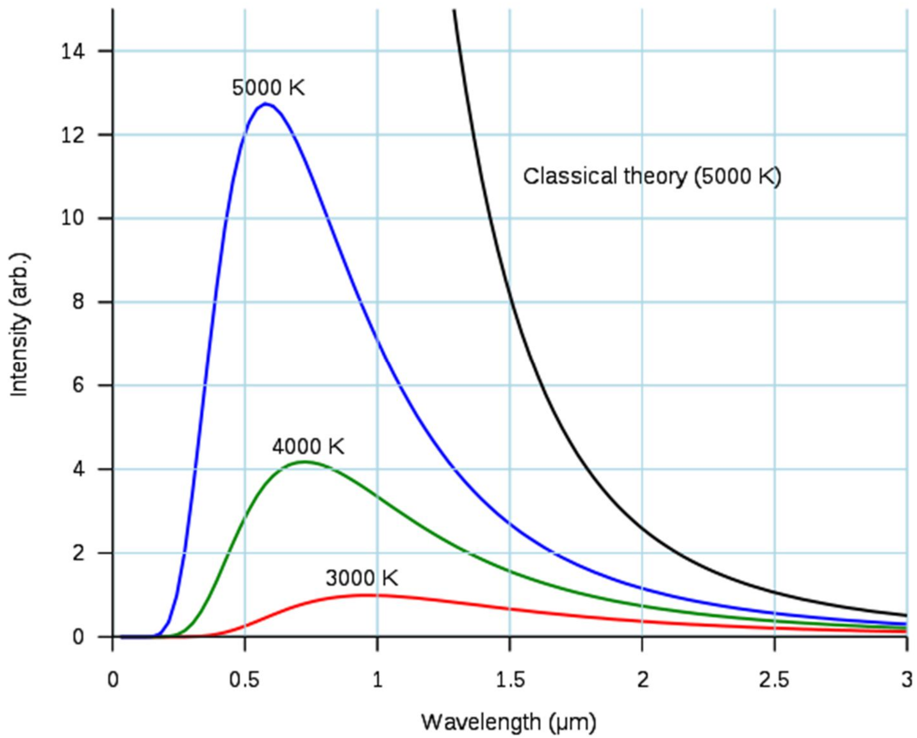
- For emissive objects or displays

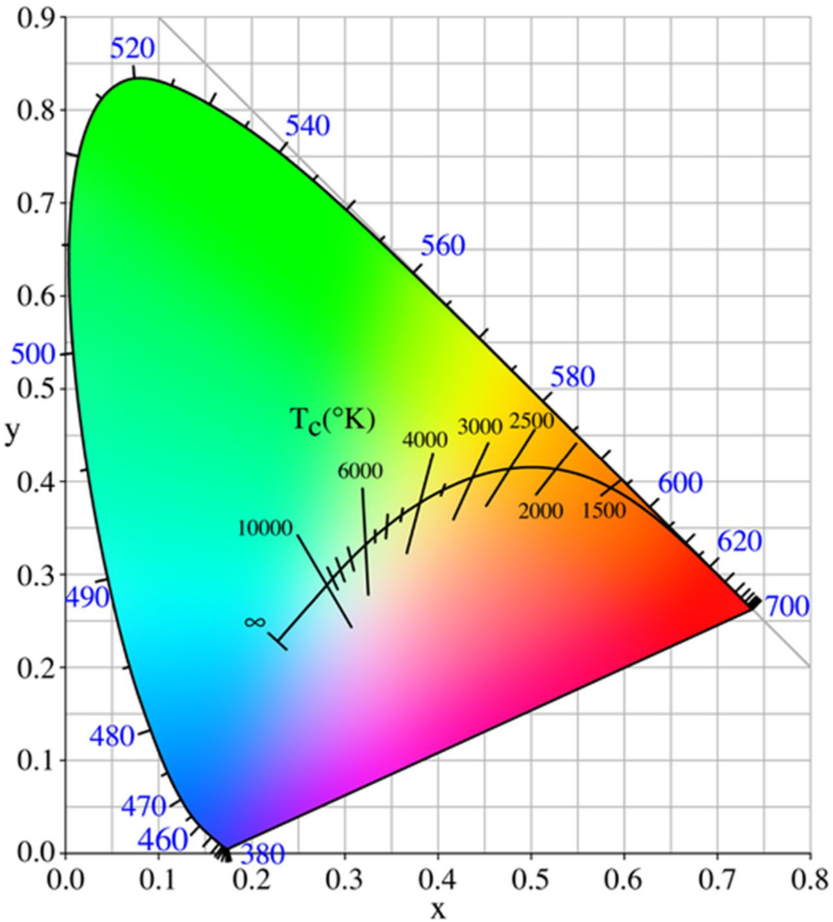
$$\text{colour} = \text{perception}(\ \text{emission}\ )$$



Black body radiation

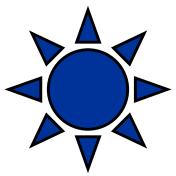
- Electromagnetic radiation emitted by a perfect absorber at a given temperature
  - Graphite is a good approximation of a black body





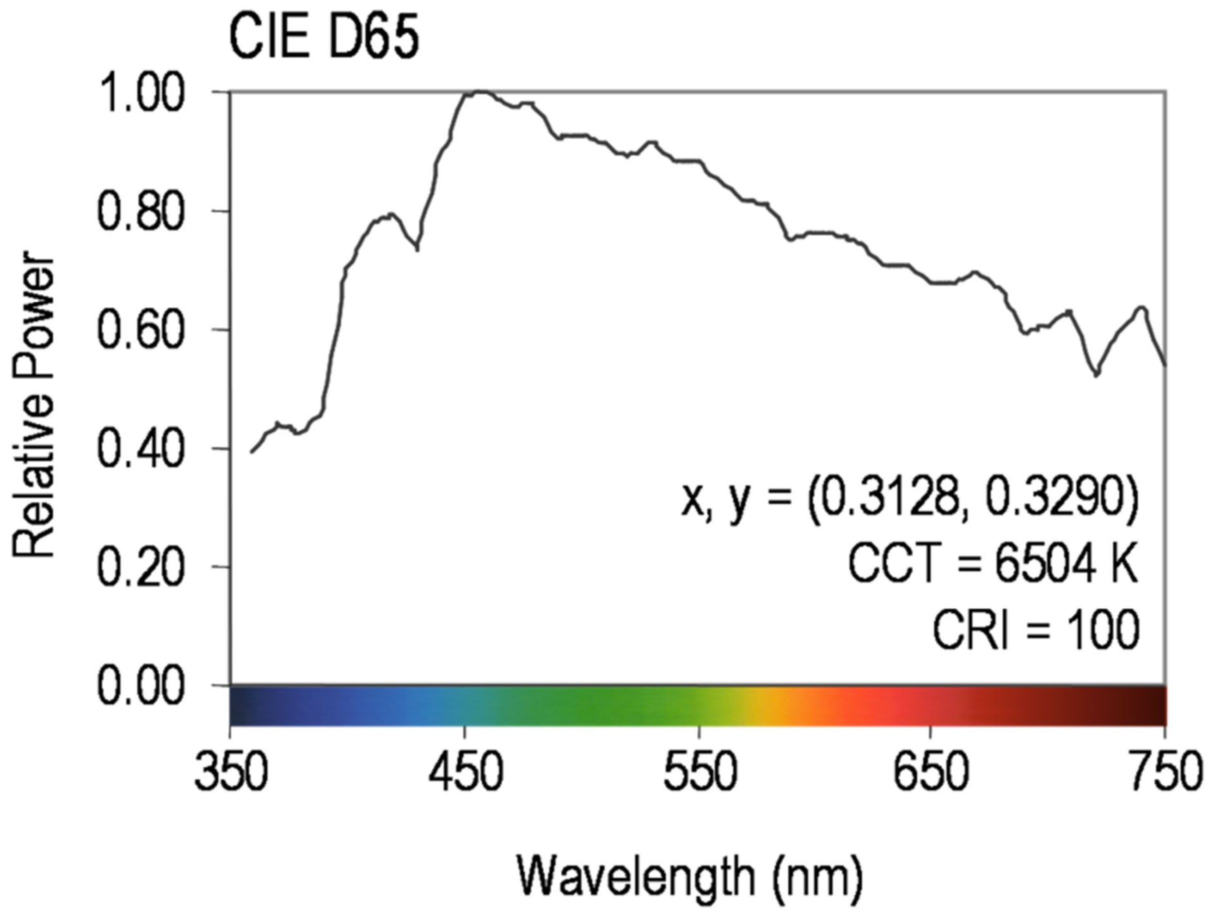
Correlated colour temperature

- The temperature of a black body radiator that produces light most closely matching the particular source
- Examples:
  - Typical north-sky light: 7500 K
  - Typical average daylight: 6500 K
  - Domestic tungsten lamp (100 to 200 W): 2800 K
  - Domestic tungsten lamp (40 to 60 W): 2700 K
  - Sunlight at sunset: 2000 K
- Useful to describe colour of the **illumination** (source of light)



Standard illuminant D65

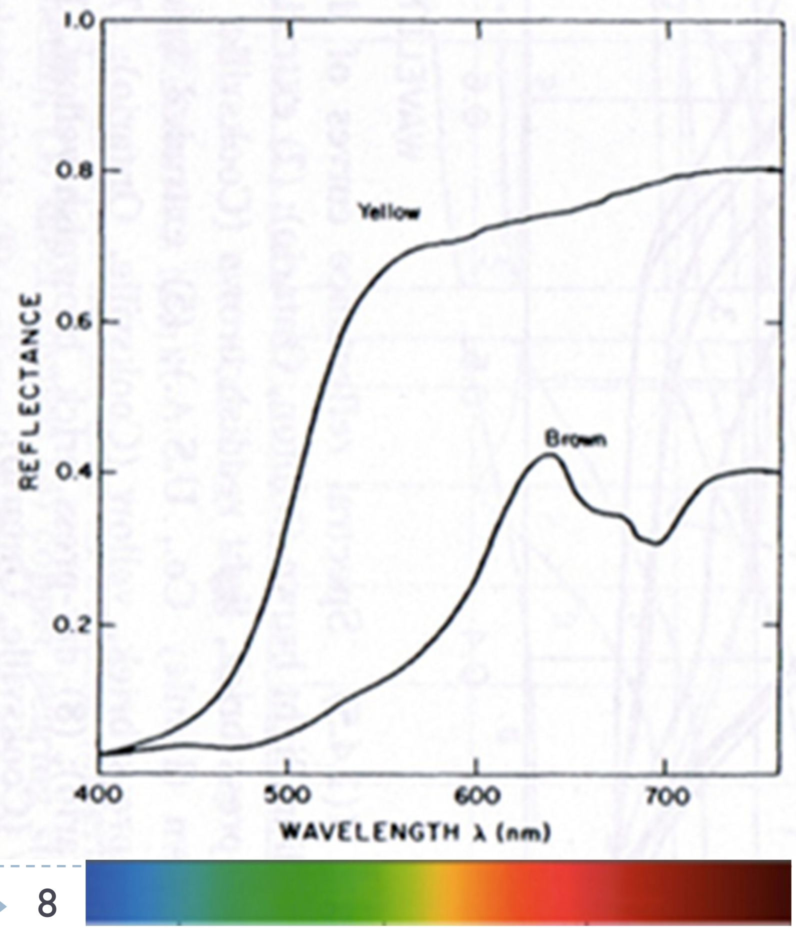
- Mid-day sun in Western Europe / Northern Europe
- Colour temperature approx. 6500 K

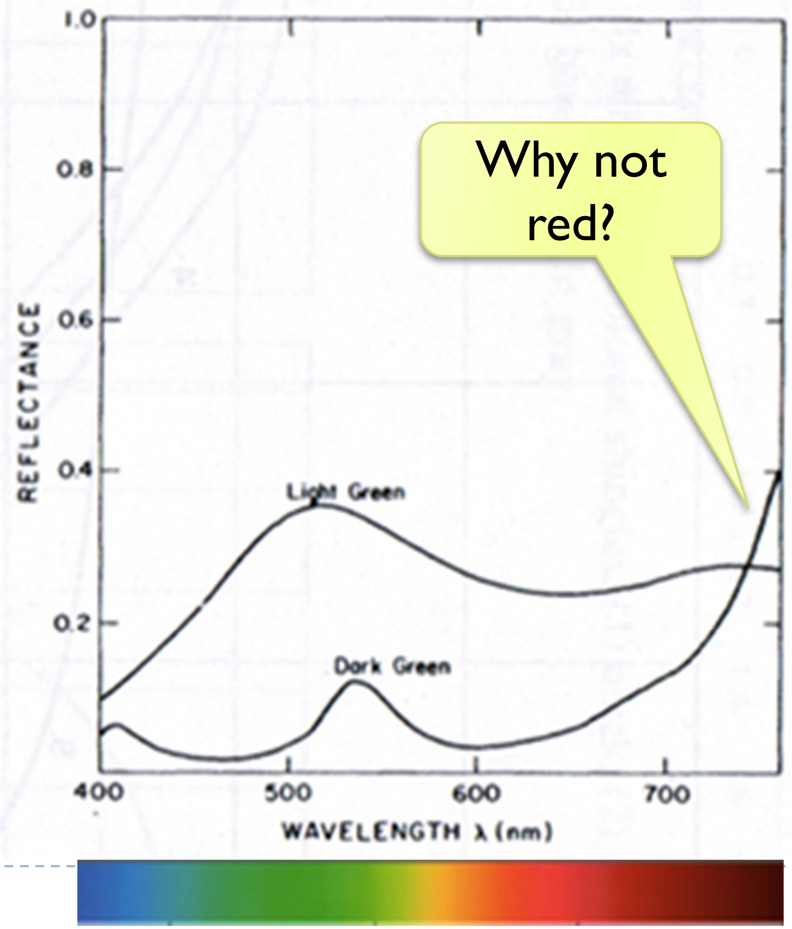


Reflectance

- Most of the light we see is reflected from objects
- These objects absorb a certain part of the light spectrum

Spectral reflectance of ceramic tiles

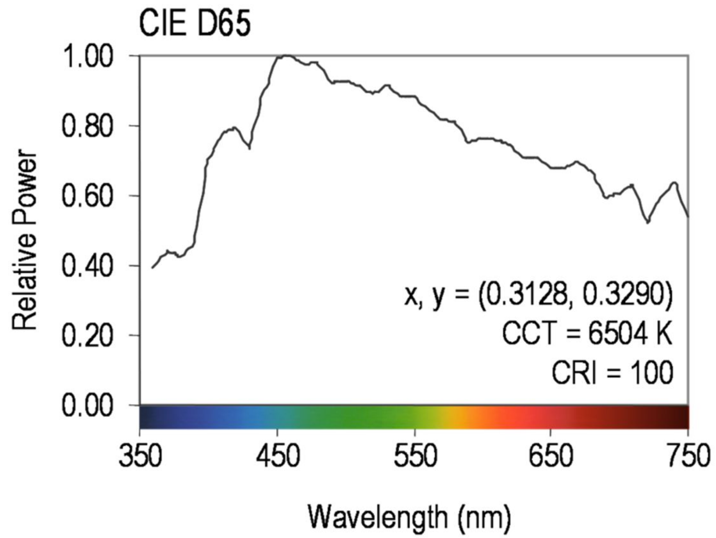


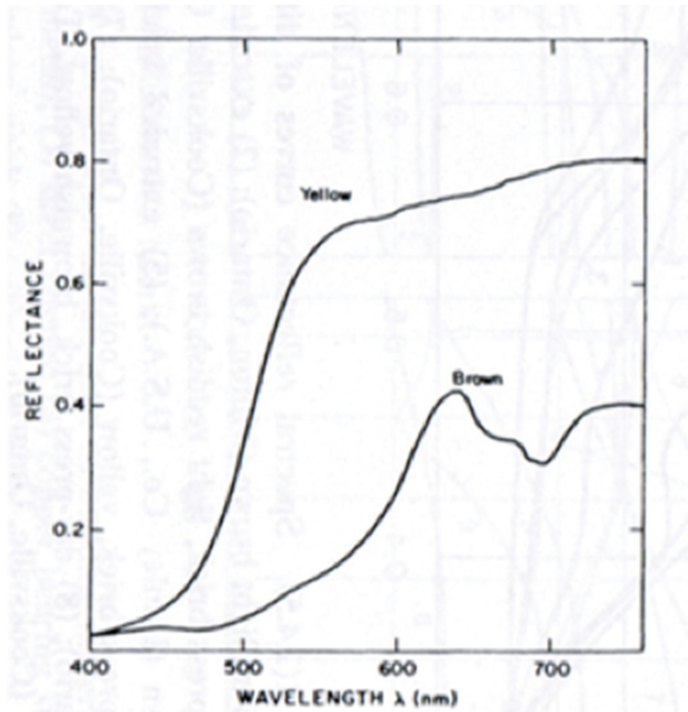


Reflected light

$$L(\lambda) = I(\lambda)R(\lambda)$$

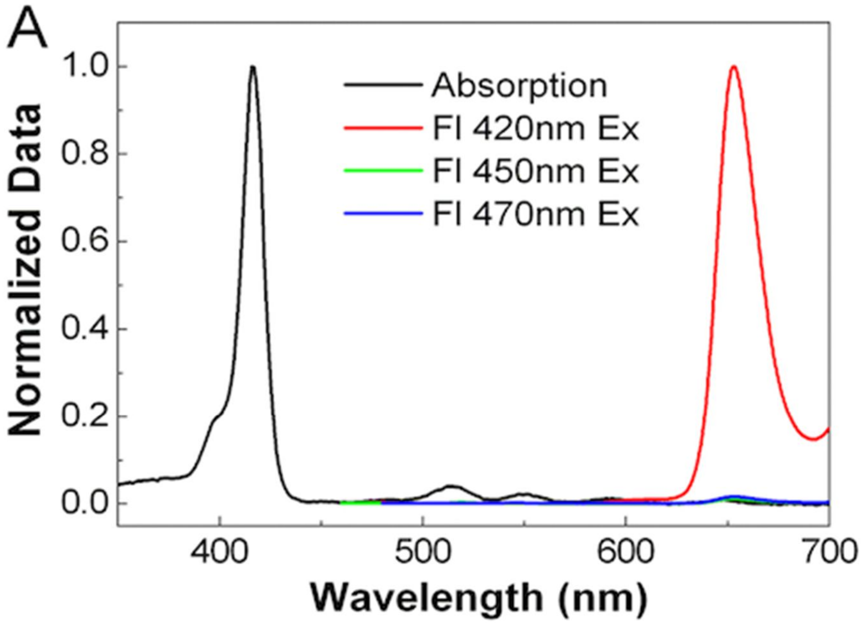
- Reflected light = illumination × reflectance

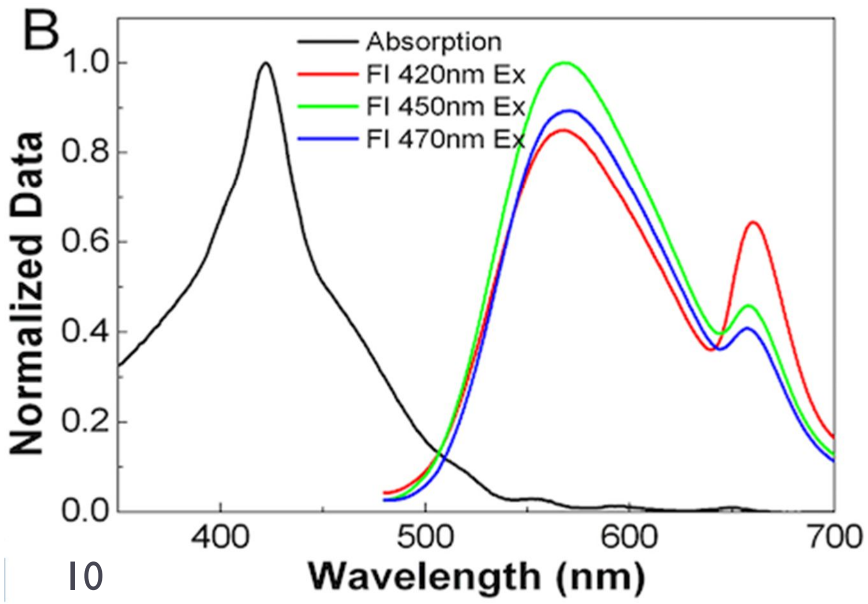


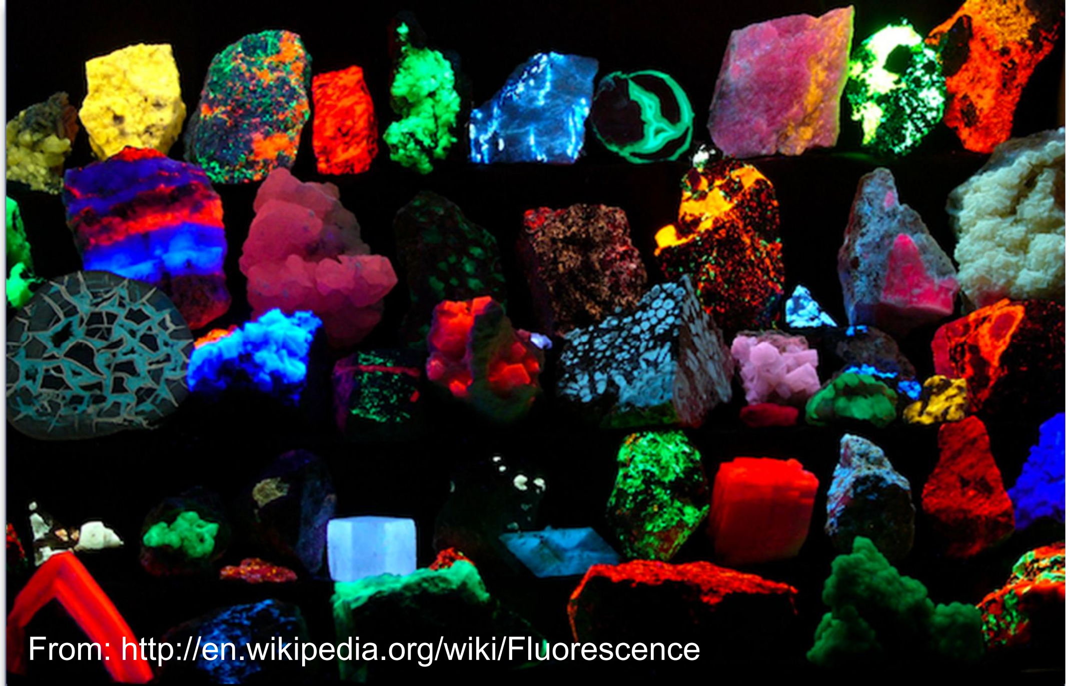


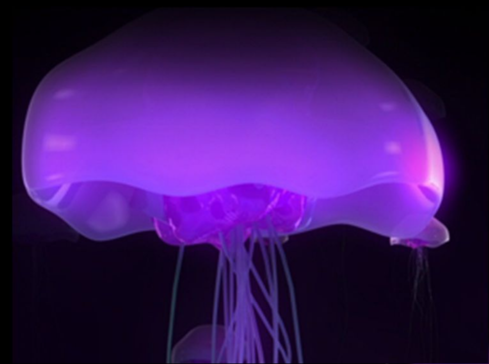
The same object may appear to have different color under different illumination.

Fluorescence









UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

Colour perception and colour spaces

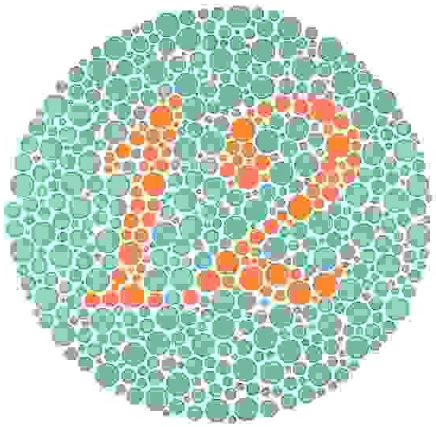
Part 2/5 – perception, cone fundamentals

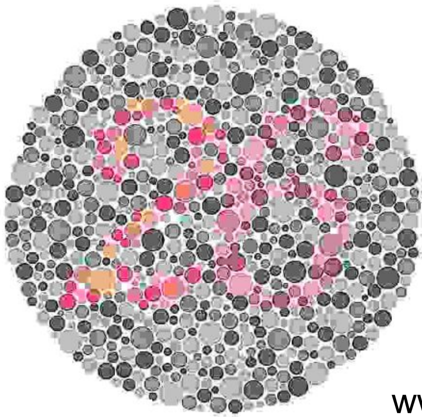
Rafał Mantiuk
Computer Laboratory, University of Cambridge

Colour perception

- Di-chromaticity (dogs, cats)
  - Yellow & blue-violet
  - Green, orange, red indistinguishable
- Tri-chromaticity (humans, monkeys)
  - Red-ish, green-isn, blue-ish
  - Colour-deficiency
    - Most often men, green-red colour-deficiency

www.lam.mus.ca.us/cats/color/

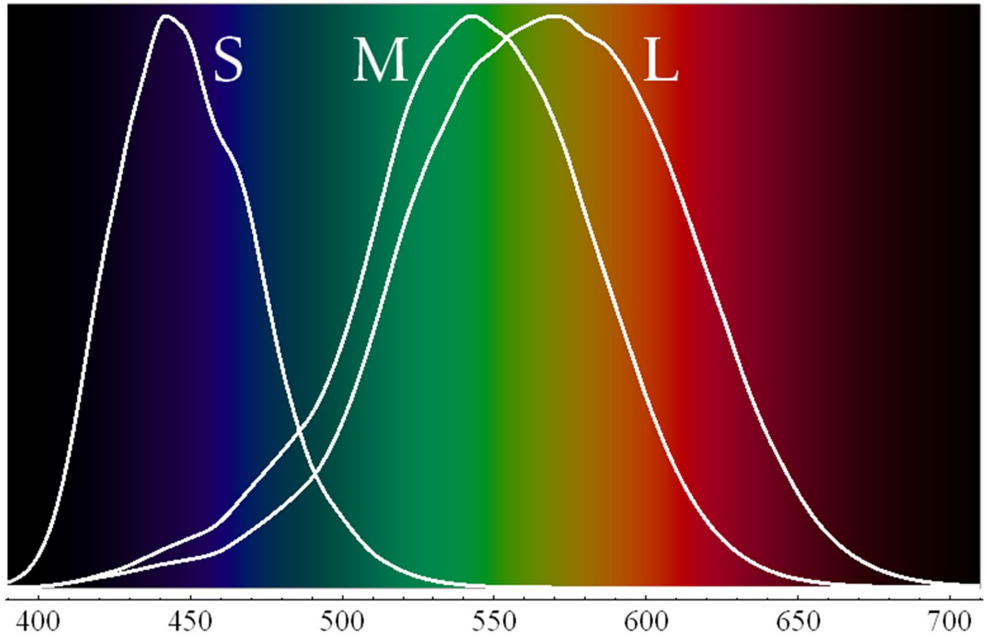




www.colorcube.com/illusions/clrblnd.html

Colour vision

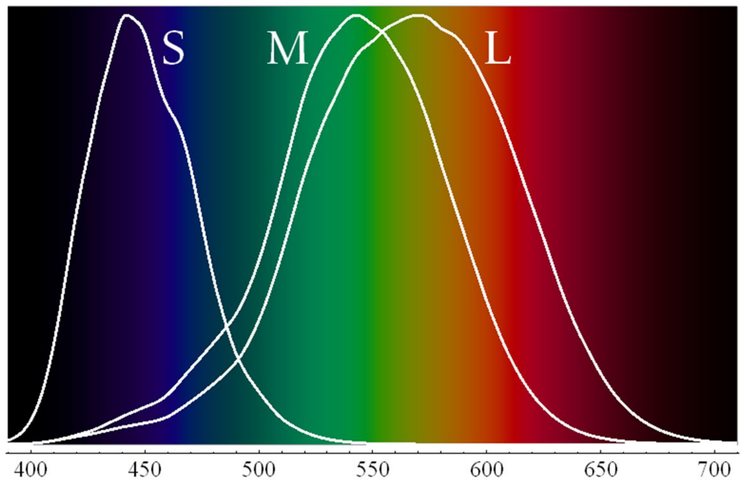
- Cones are the photreceptors responsible for colour vision
  - Only daylight, we see no colours when there is not enough light
- Three types of cones
  - S – sensitive to short wavelengths
  - M – sensitive to medium wavelengths
  - L – sensitive to long wavelengths

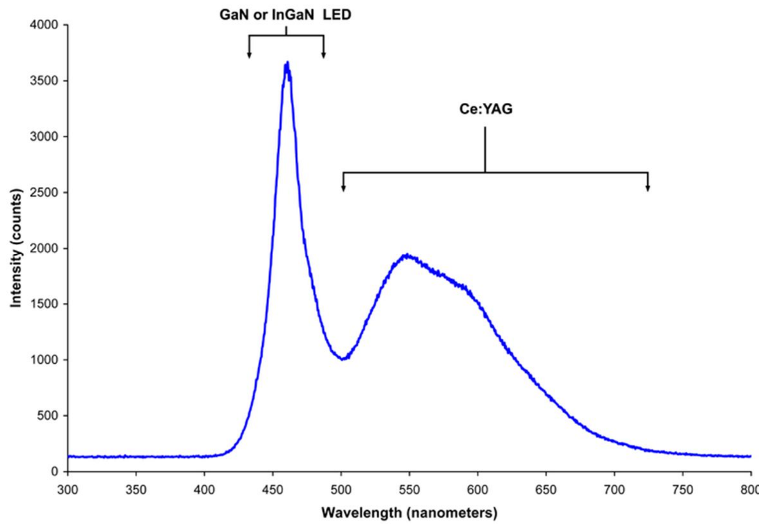


Sensitivity curves – probability that a photon of that wavelengths will be absorbed by a photoreceptor. S,M and L curves are normalized in this plot.

Perceived light

- cone response = sum(sensitivity × reflected light)





Although there is an infinite number of wavelengths, we have only three photoreceptor types to sense differences between light spectra

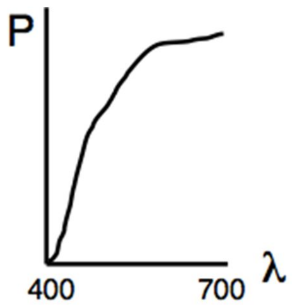
Formally

$$R_S = \int_{380}^{730} S_S(\lambda) \cdot L(\lambda) d\lambda$$

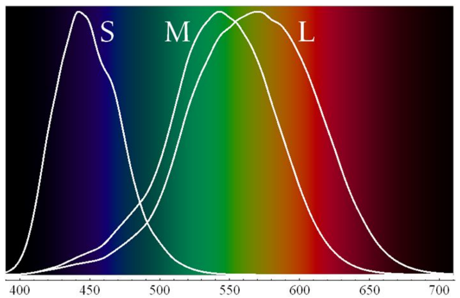
Index S for S-cones

Metamers

- Even if two light spectra are different, they may appear to have the same colour
- The light spectra that appear to have the same colour are called **metamers**
- Example:

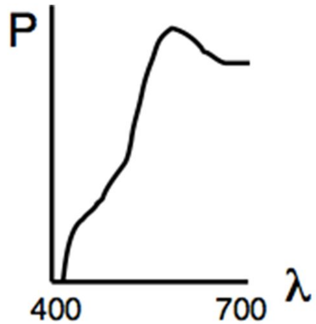


*

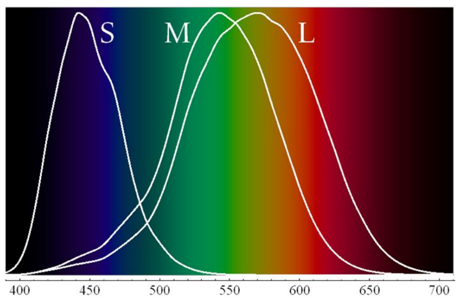


$= [L_1, M_1, S_1]$

$=$



*

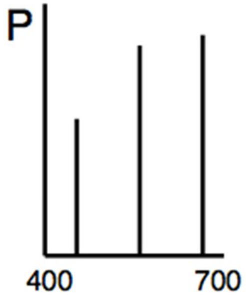


$= [L_2, M_2, S_2]$

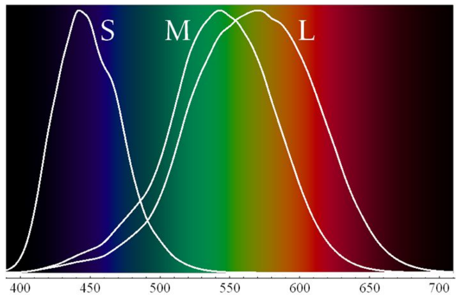
Practical application of metamerism

- Displays do not emit the same light spectra as real-world objects
- Yet, the colours on a display look almost identical

On the display

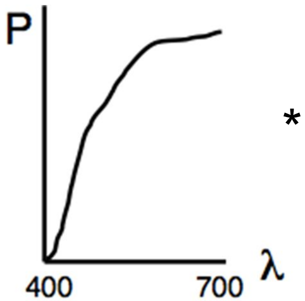


*

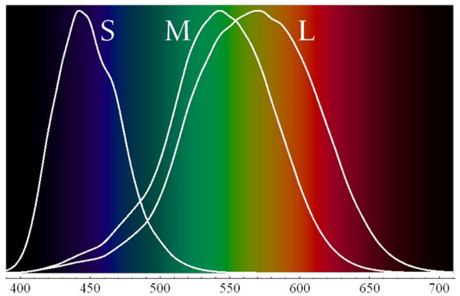


$= [L_1, M_1, S_1]$

$=$



*

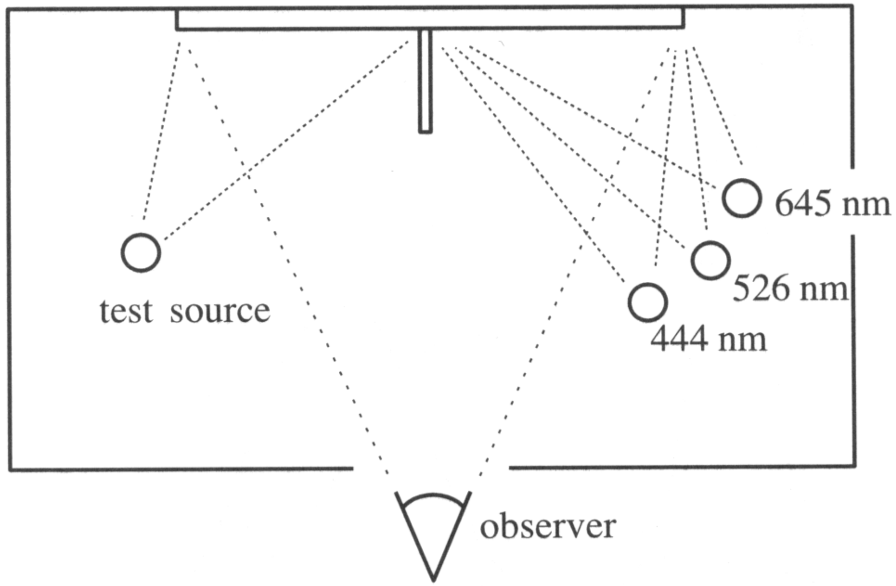


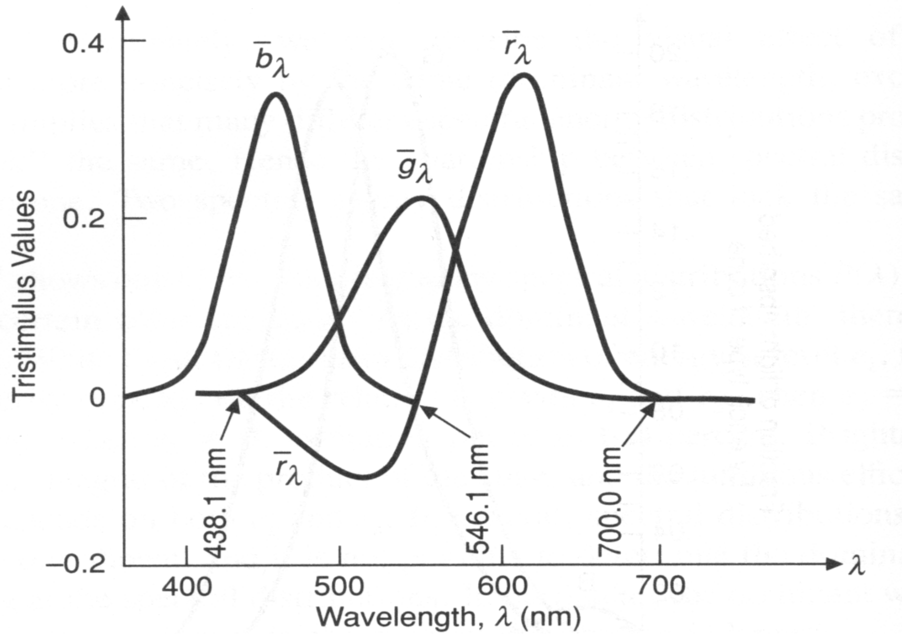
$= [L_2, M_2, S_2]$

In real world

Tristimulus Colour Representation

- Observation
  - Any colour can be matched using three linear independent reference colours
  - May require "negative" contribution to test colour
  - Matching curves describe the value for matching mono-chromatic spectral colours of equal intensity
    - With respect to a certain set of primary colours



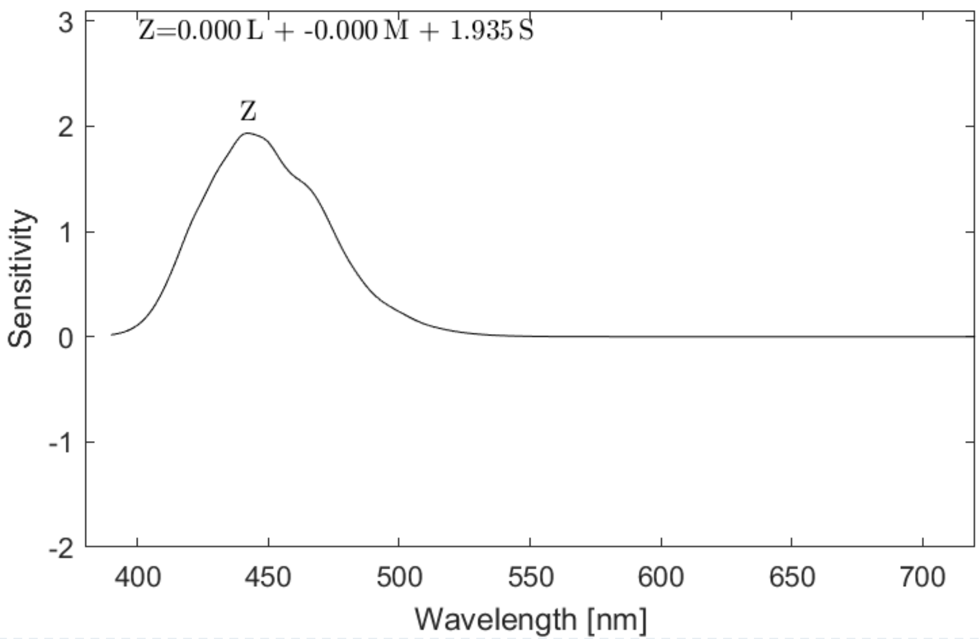


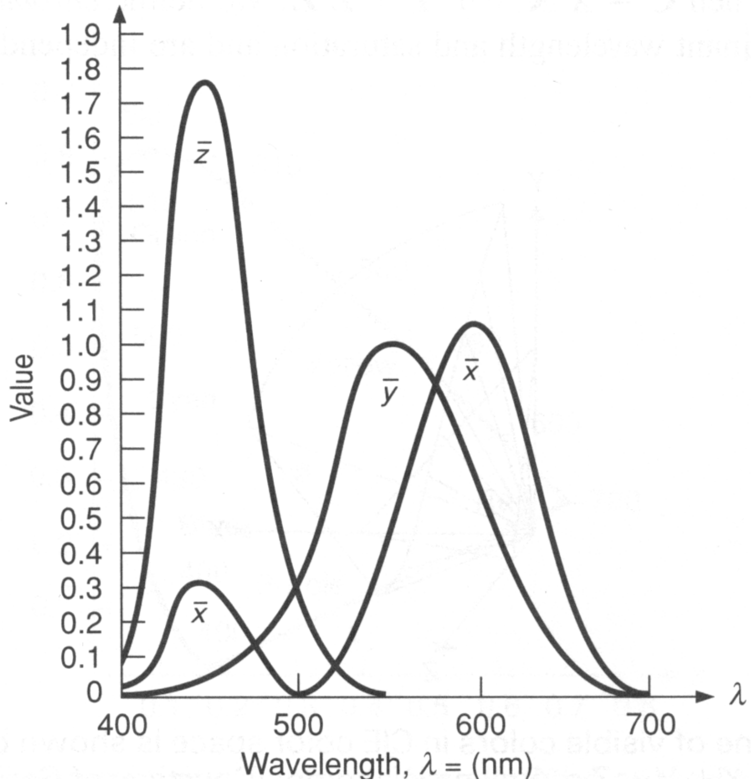
Standard Colour Space CIE-XYZ

- CIE Experiments [Guild and Wright, 1931]
  - Colour matching experiments
  - Group ~12 people with „normal" colour vision
  - 2 degree visual field (fovea only)
- CIE 2006 XYZ
  - Derived from LMS colour matching functions by Stockman & Sharpe
  - S-cone response differs the most from CIE 1931
- CIE-XYZ Colour Space
  - Goals
    - Abstract from concrete primaries used in an experiment
    - All matching functions are positive
    - Primary „Y" is roughly proportionally to achromatic response (luminance)

Standard Colour Space CIE-XYZ

- Standardized imaginary primaries CIE XYZ (1931)
  - Could match all physically realizable colour stimuli
  - Cone sensitivity curves can be obtained by a linear transformation of CIE XYZ



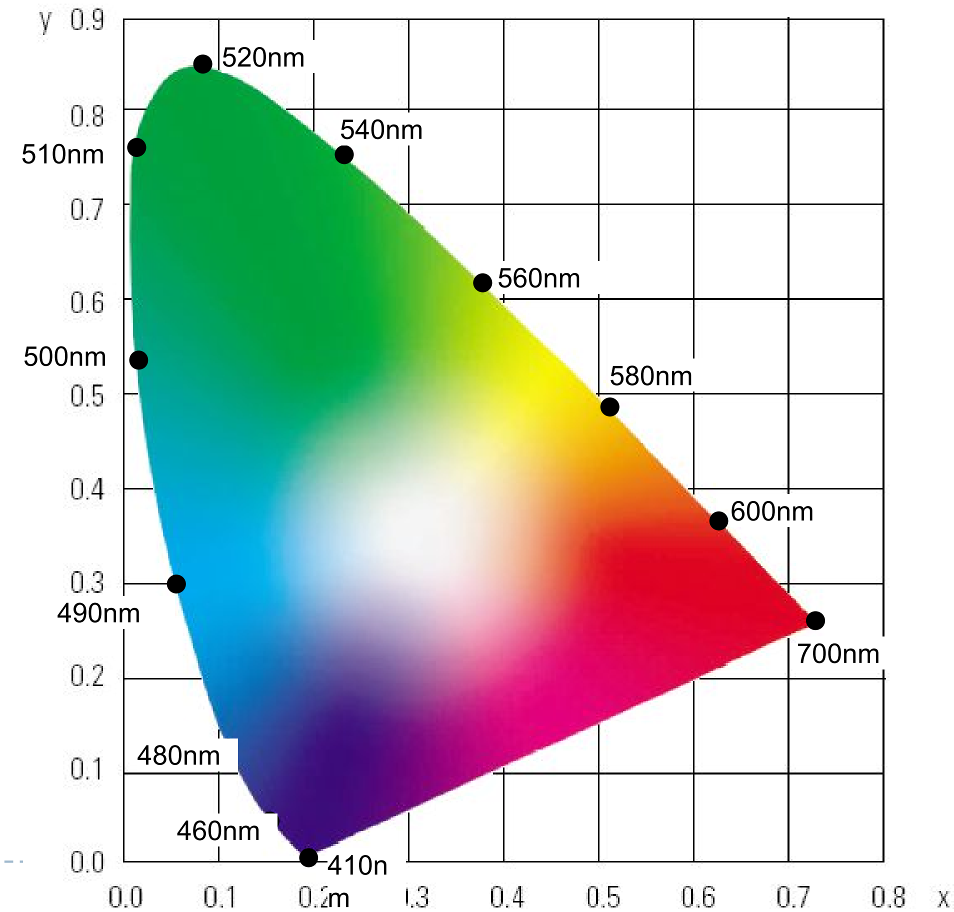


CIE chromaticity diagram

- *chromaticity* values are defined in terms of x, y, z

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z} \qquad x + y + z = 1$$

  - ignores luminance
  - can be plotted as a 2D function
- pure colours (single wavelength) lie along the outer curve
- all other colours are a mix of pure colours and hence lie inside the curve
- points outside the curve do not exist as colours



UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

Colour perception and colour spaces

Part 3/5 – colour opponent processing

Rafał Mantiuk

Computer Laboratory, University of Cambridge

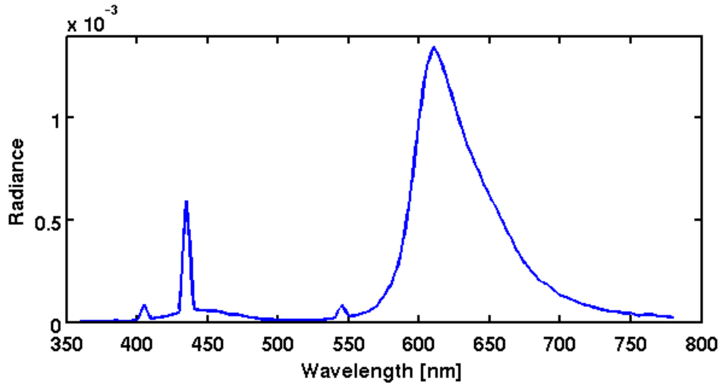
Luminance

- Luminance – measure of light weighted by the response of the achromatic mechanism. Units: cd/m^2 (ISO) or nit

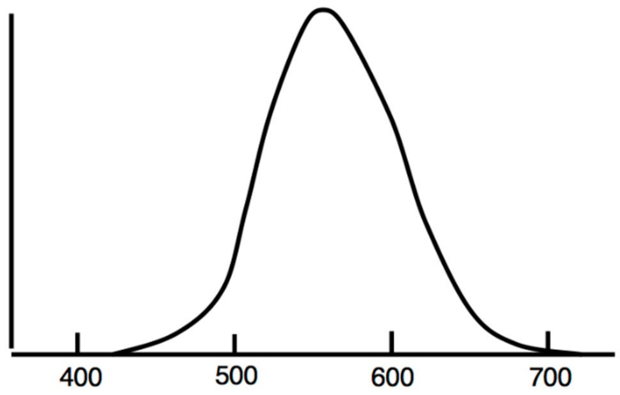
Luminance

$$L_V = \int_{350}^{700} kL(\lambda)V(\lambda)d\lambda \qquad k = 683.002$$

Light spectrum (radiance)

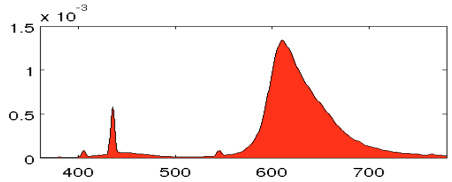


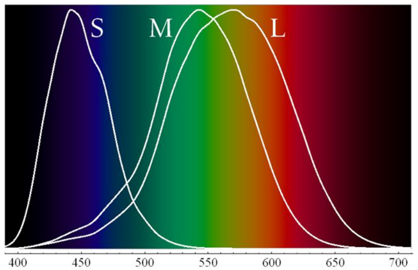
Luminous efficiency function (weighting)

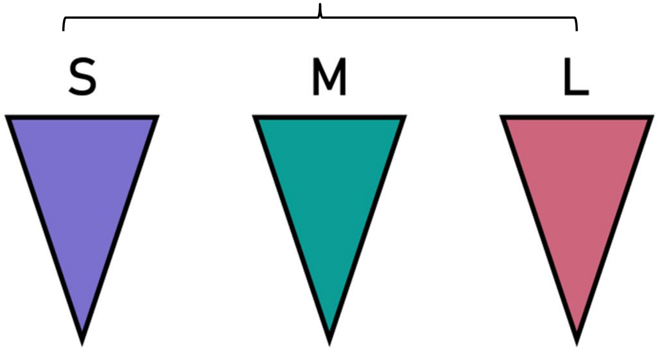


Achromatic/chromatic vision mechanisms

Light spectra

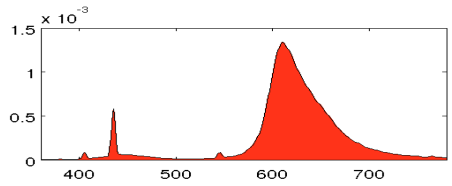


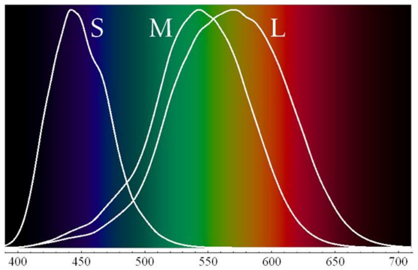


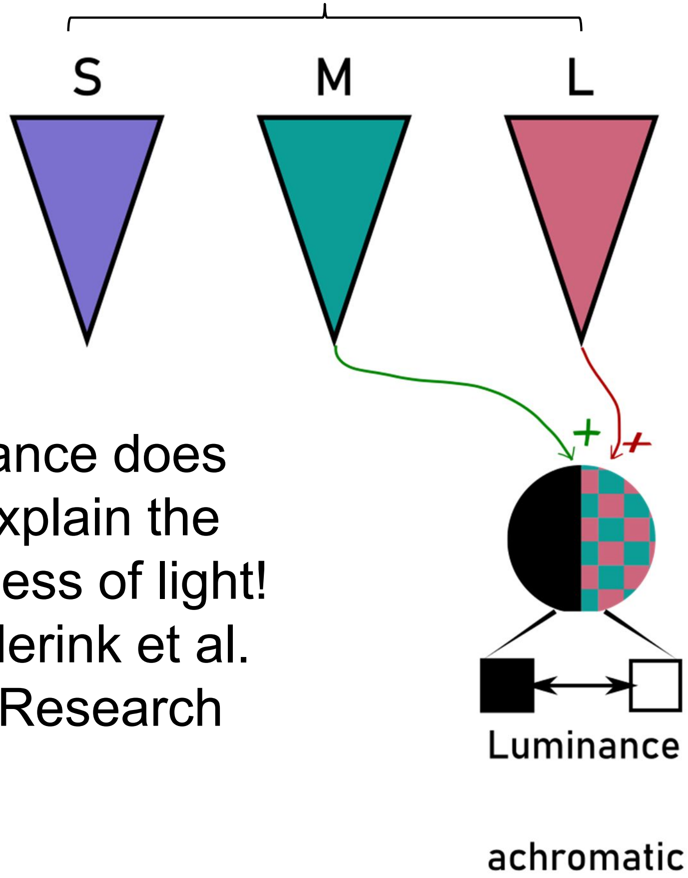


Achromatic/chromatic vision mechanisms

Light spectra

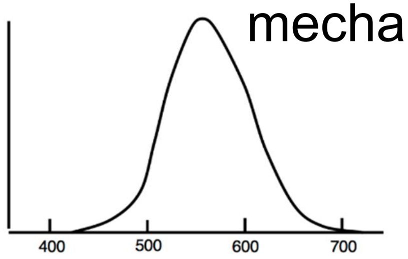




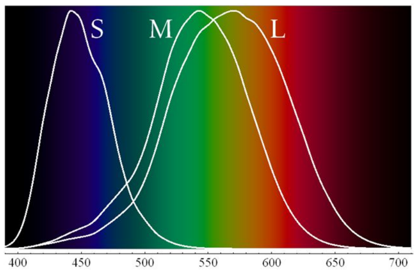


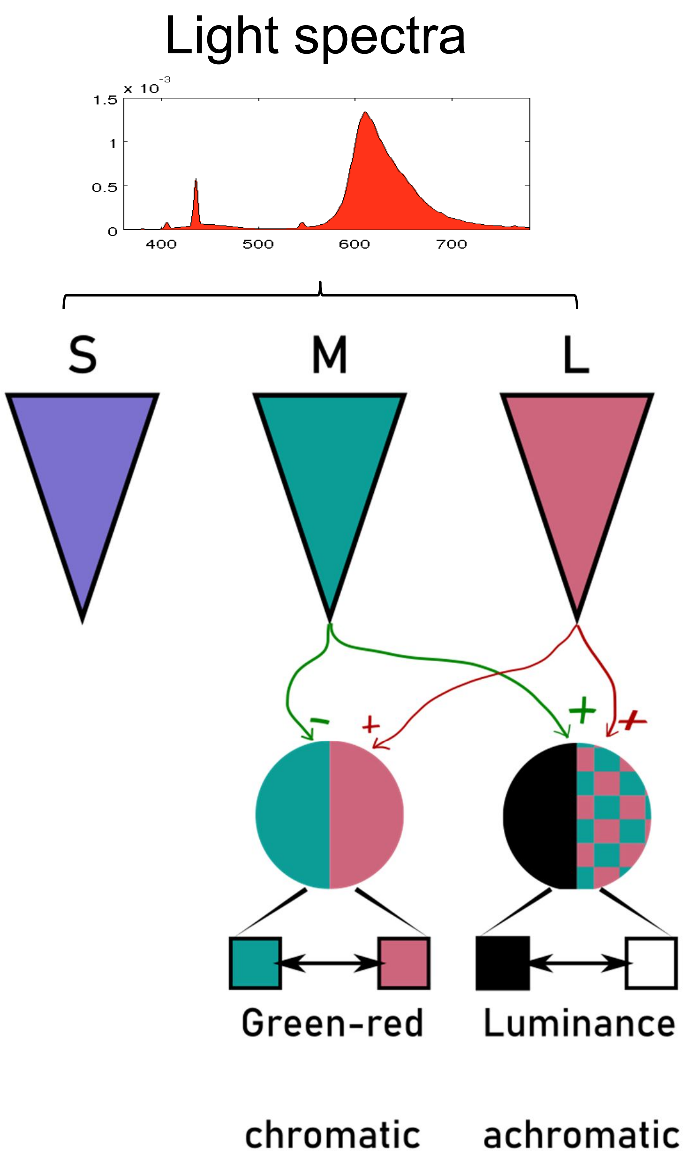
Luminance does NOT explain the brightness of light! [Koenderink et al. Vision Research 2016]

Sensitivity of the achromatic mechanism

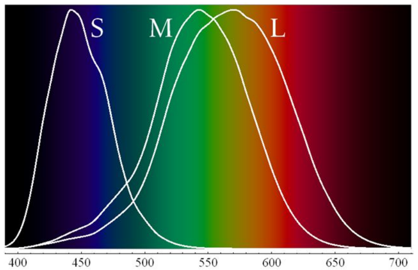


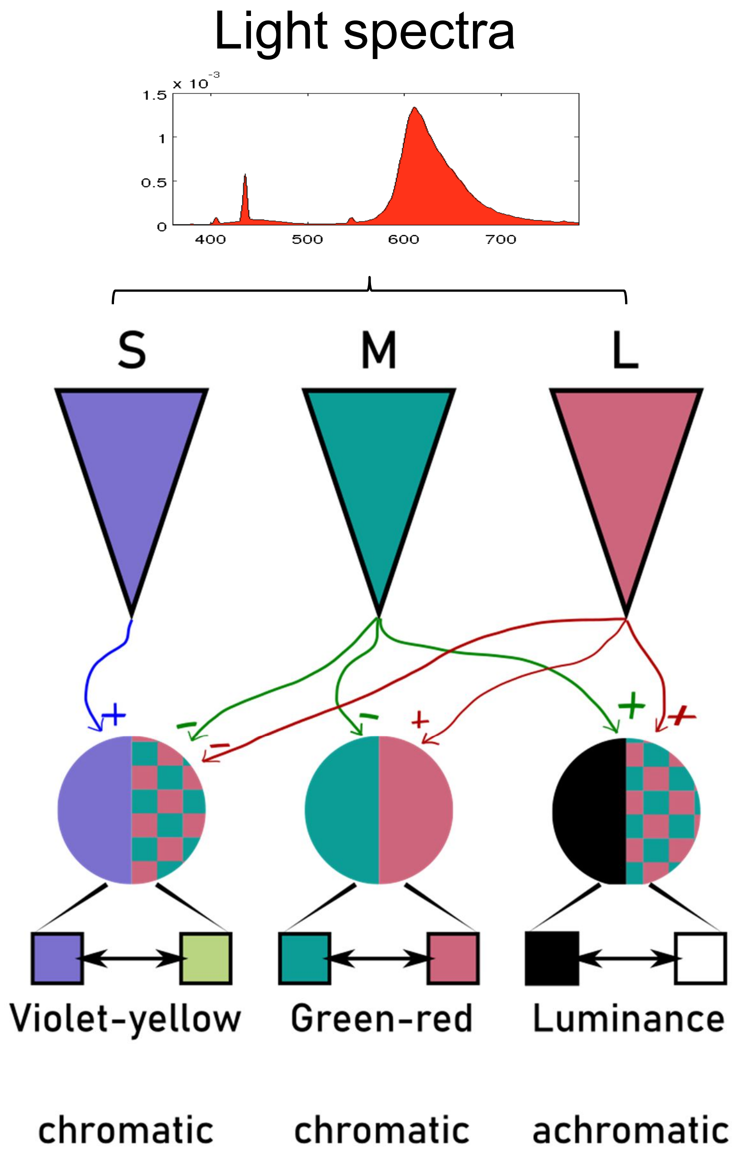
Achromatic/chromatic vision mechanisms



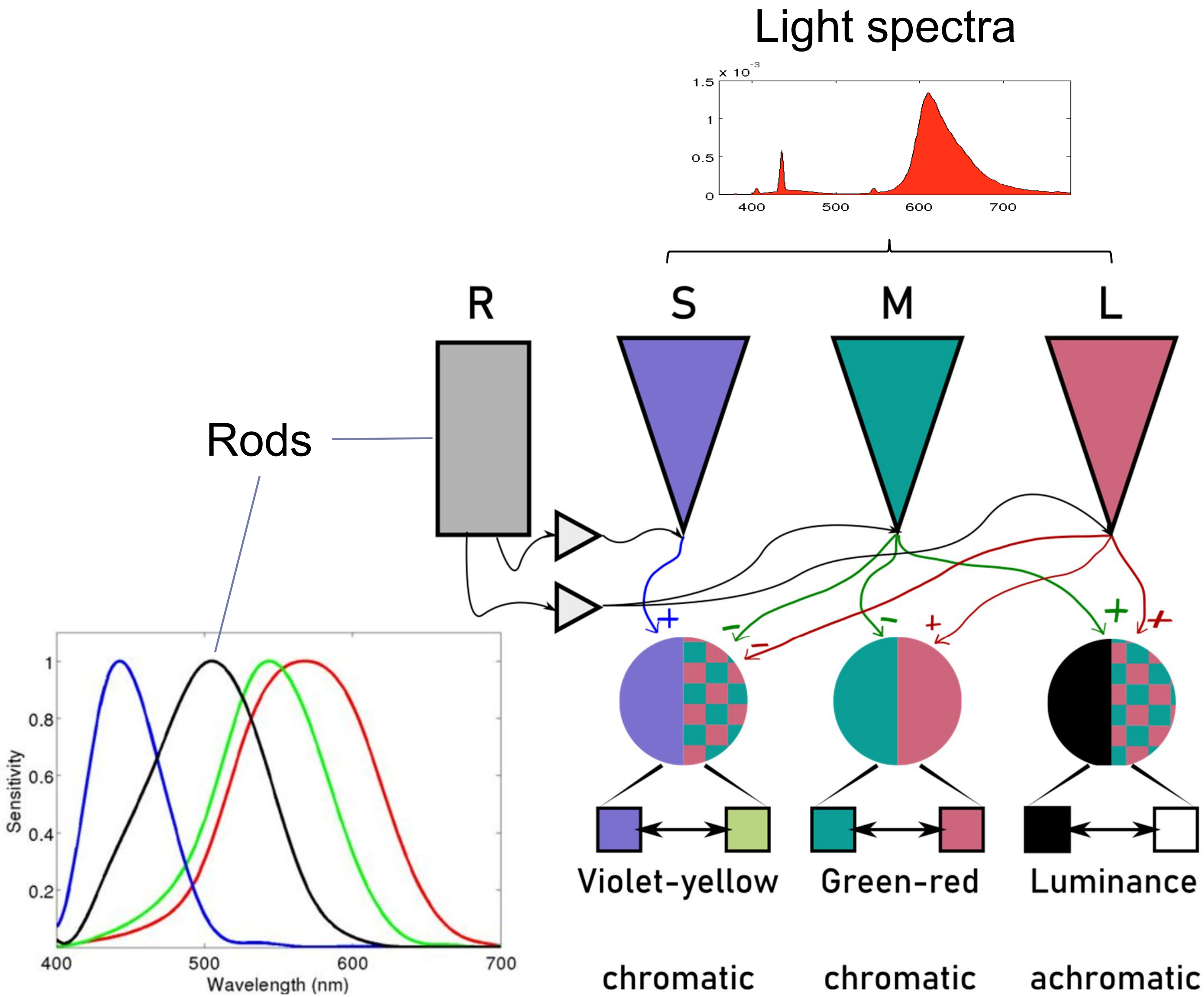


Achromatic/chromatic vision mechanisms





Achromatic/chromatic vision mechanisms



UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

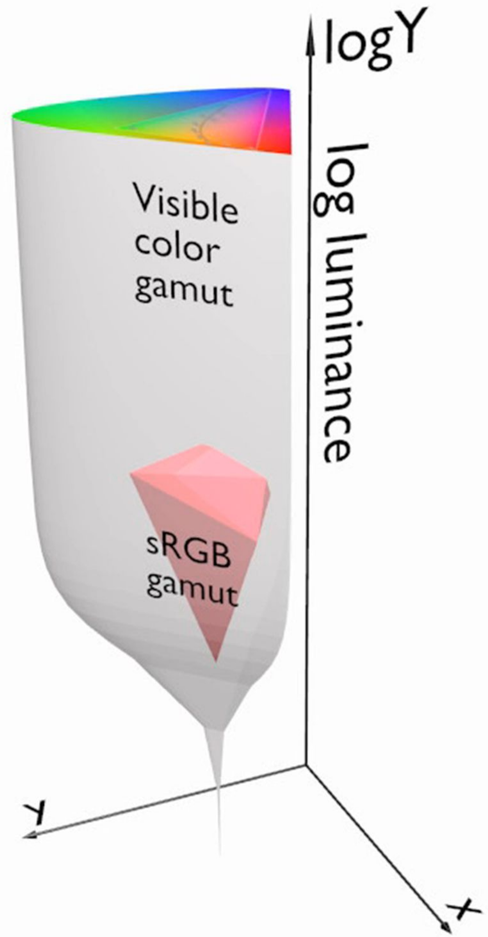
Colour perception and colour spaces

Part 4/5 – gamuts, linear and gamma-encoded colour

Rafał Mantiuk
Computer Laboratory, University of Cambridge

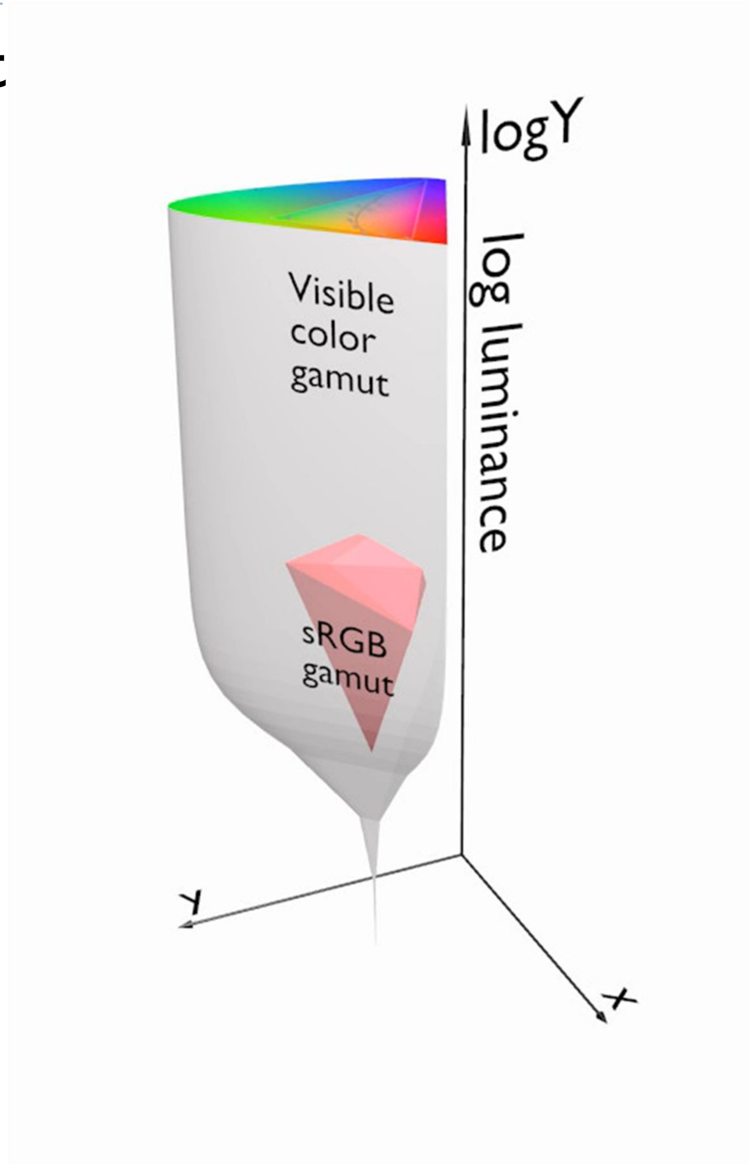
Visible vs. displayable colours

- All physically possible and visible colours form a solid in the XYZ space
- Each display device can reproduce a subspace of that space
- A chromacity diagram is a projection of a slice taken from a 3D solid in XYZ space
- Colour Gamut – the solid in a colour space
  - Usually defined in XYZ to be device-independent

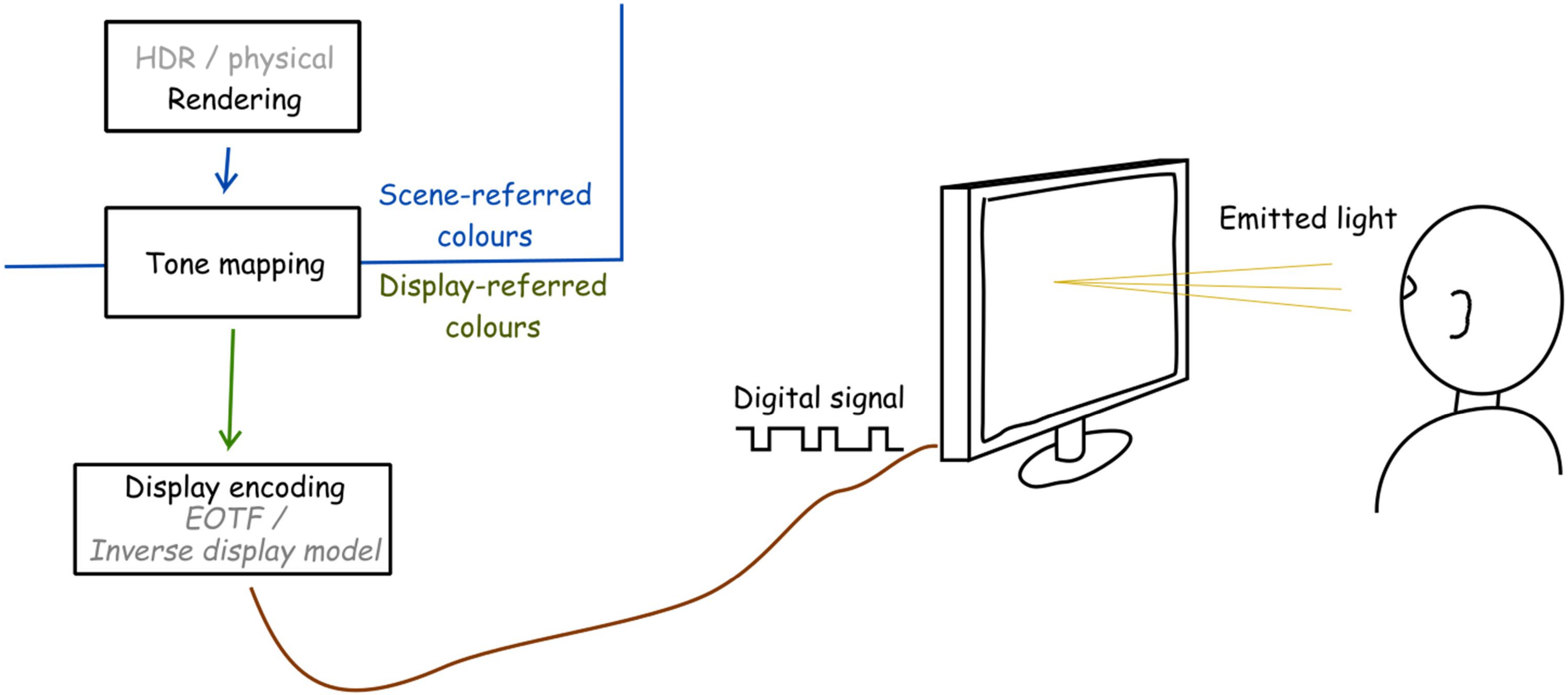


Standard vs. High Dynamic Range

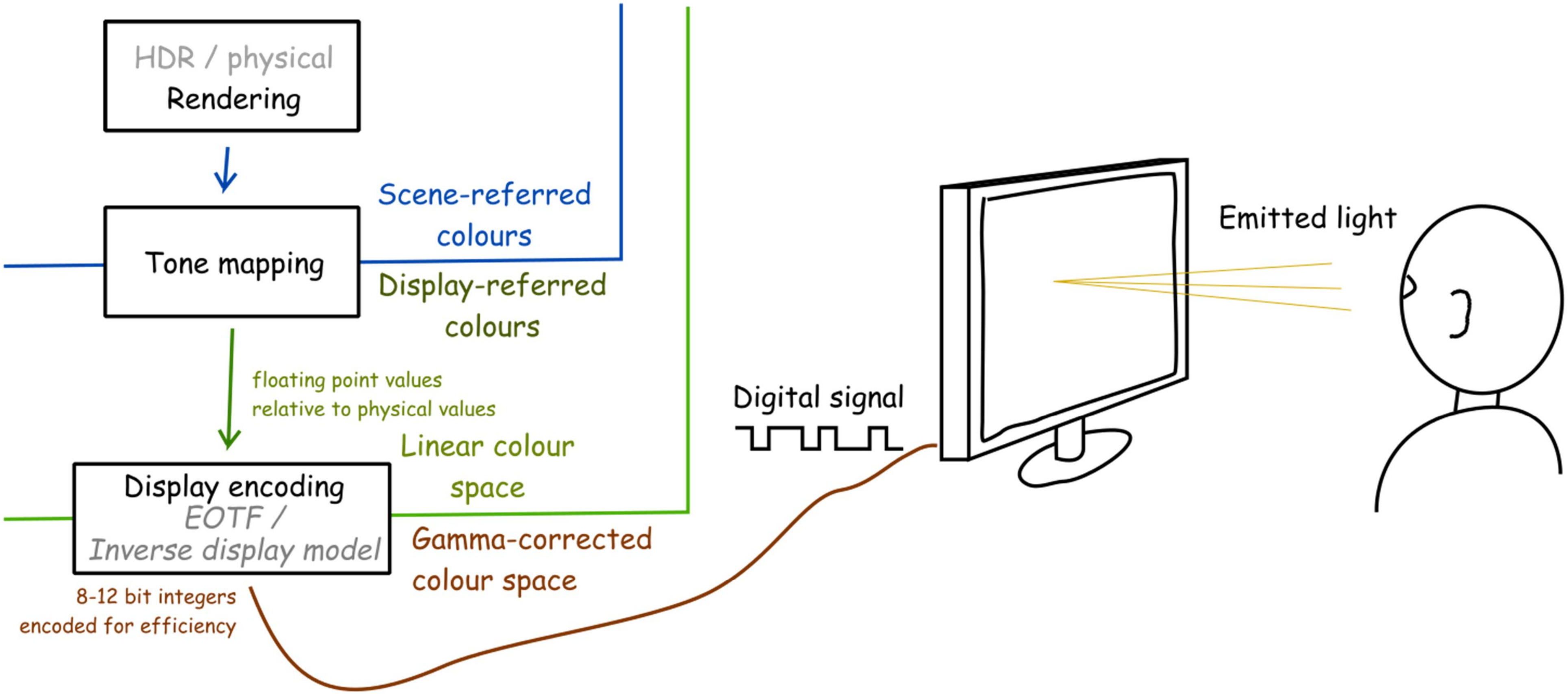
- **HDR** cameras/formats/displays attempt capture/represent/reproduce (almost) all visible colours
  - They represent scene colours and therefore we often call this representation *scene-referred*
- **SDR** cameras/formats/devices attempt to capture/represent/reproduce only colours of a standard sRGB colour gamut, mimicking the capabilities of CRTs monitors
  - They represent display colours and therefore we often call this representation *display-referred*



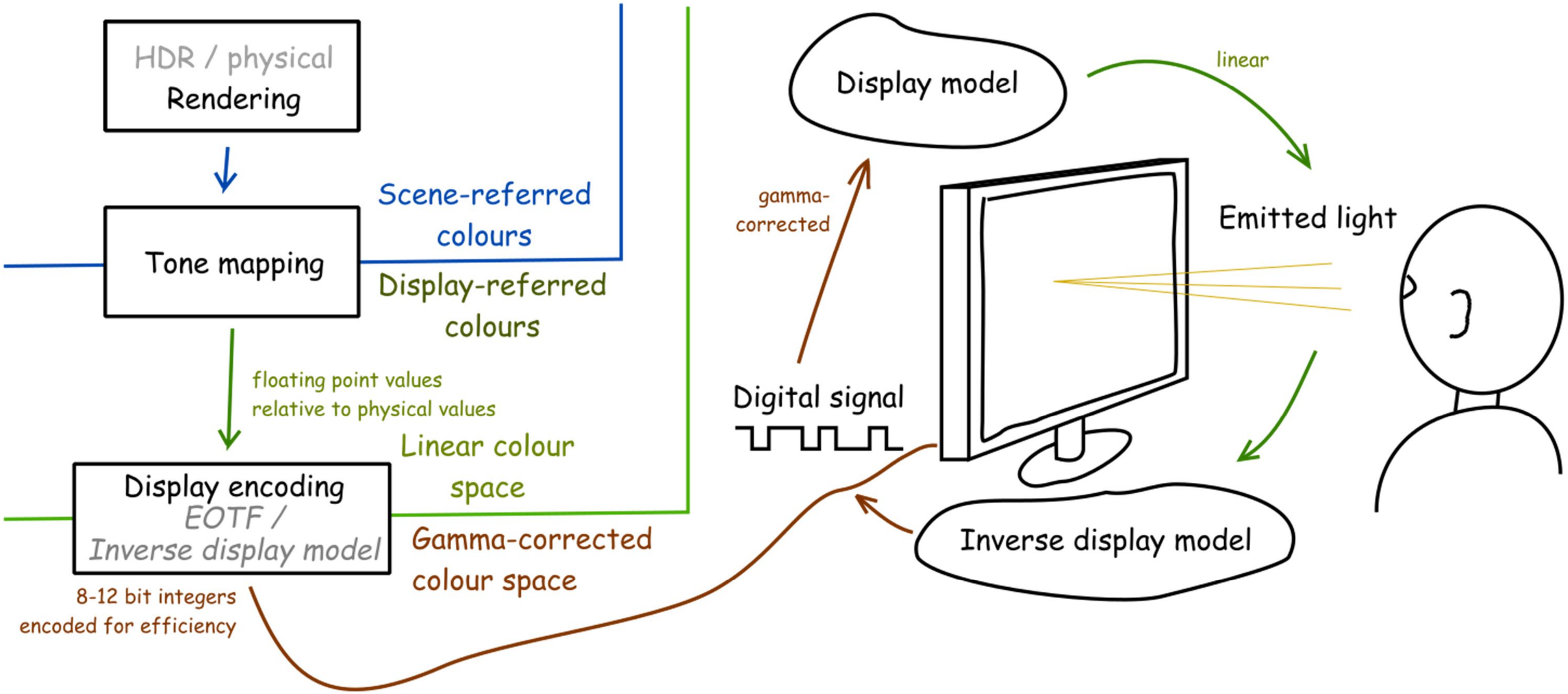
From rendering to display



From rendering to display



From rendering to display



Display encoding for SDR: gamma

- Gamma correction is often used to encode luminance or tri-stimulus color values (RGB) in imaging systems (displays, printers, cameras, etc.)

Gain

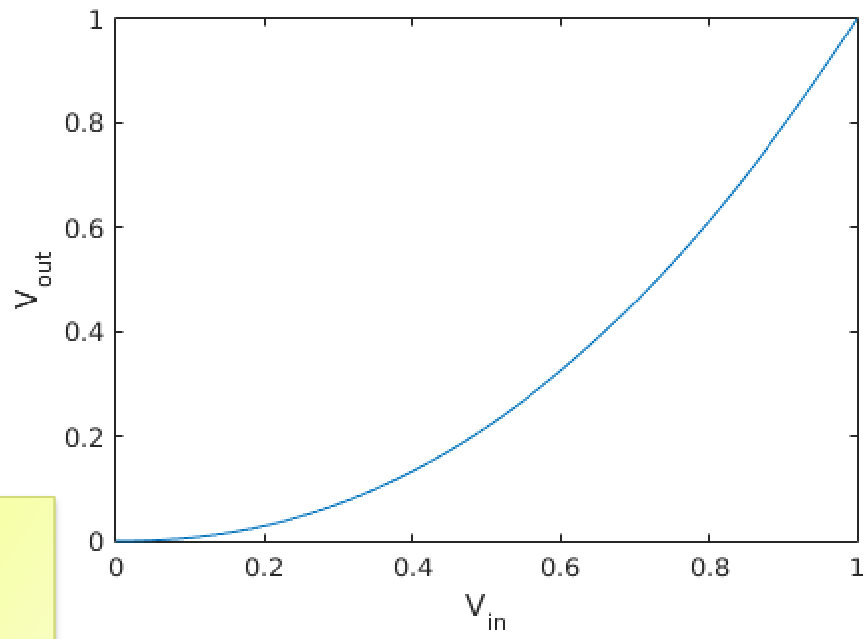
Gamma (usually =2.2)

$$\mathrm{V}_{\mathrm{out}} = \mathrm{a} \cdot V_{in}^{\gamma}$$

(relative) Luminance
Physical signal

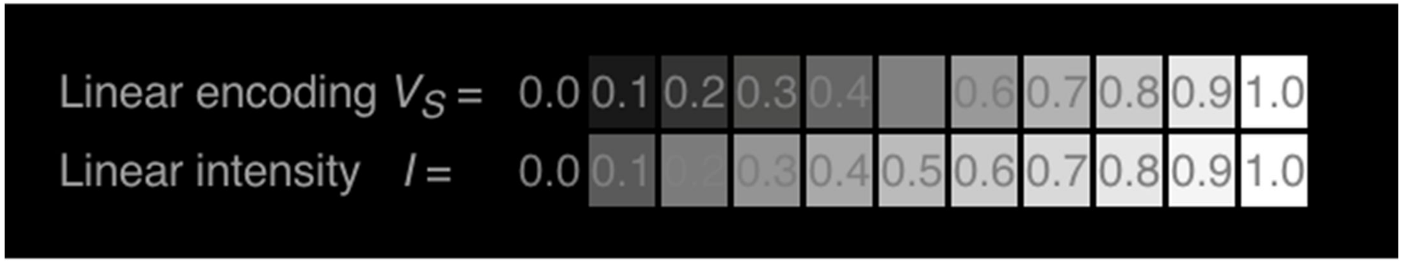
Luma
Digital signal (0-1)

Inverse: $\mathrm{V}_{\mathrm{in}} = \left(\frac{1}{a} \cdot V_{out}\right)^{\frac{1}{\gamma}}$



Colour: the same equation applied to red, green and blue colour channels.

Why is gamma needed?



<- Pixel value (luma)

<- Luminance

- *Gamma-corrected* pixel values give a scale of brightness levels that is more perceptually uniform
- At least 12 bits (instead of 8) would be needed to encode each color channel without gamma correction
- And accidentally it was also the response of the CRT gun

Luma – gray-scale pixel value

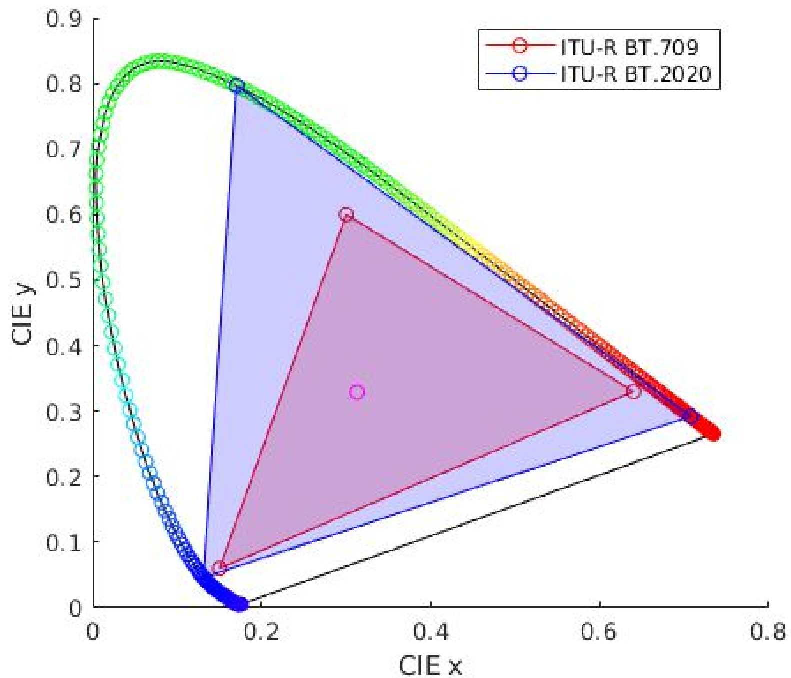
- **Luma** - pixel "brightness" in *gamma corrected* units
$$L' = 0.2126R' + 0.7152G' + 0.0722B'$$
  - R', G' and B' are *gamma-corrected* colour values
  - Prime symbol denotes *gamma corrected*
  - Used in image/video coding

- Note that relative **luminance** if often approximated with
$$L = 0.2126R + 0.7152G + 0.0722B$$
$$= 0.2126(R')^{\gamma} + 0.7152(G')^{\gamma} + 0.0722(B')^{\gamma}$$
- $R, G,$ and B are *linear* colour values
- Luma and luminace are different quantities despite similar formulas

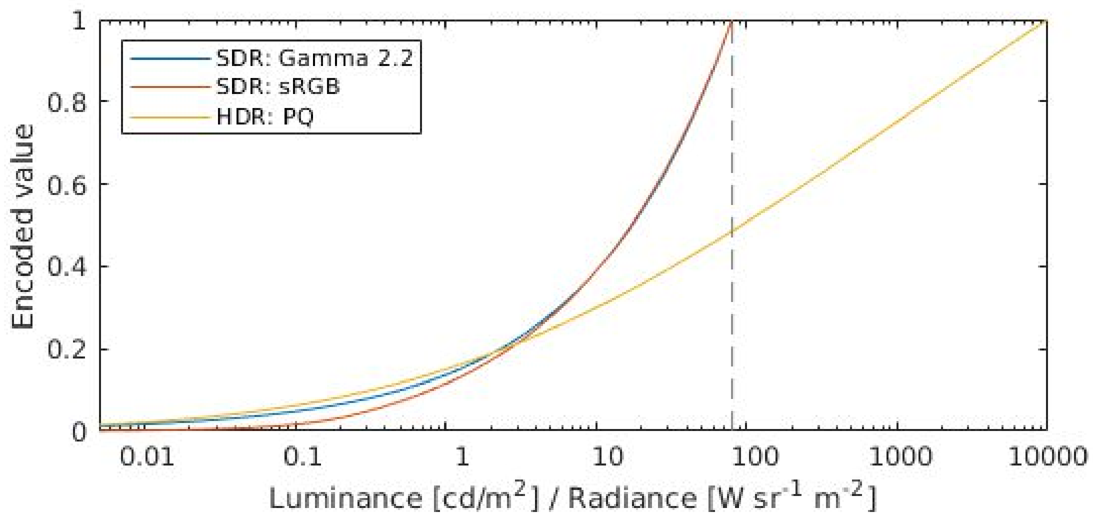
Standards for display encoding

Display type	Colour space	EOTF	Bit depth
Standard Dynamic Range	ITU-R 709	2.2 gamma / sRGB	8 to 10
High Dynamic Range	ITU-R 2020	ITU-R 2100 (PQ/HLG)	10 to 12

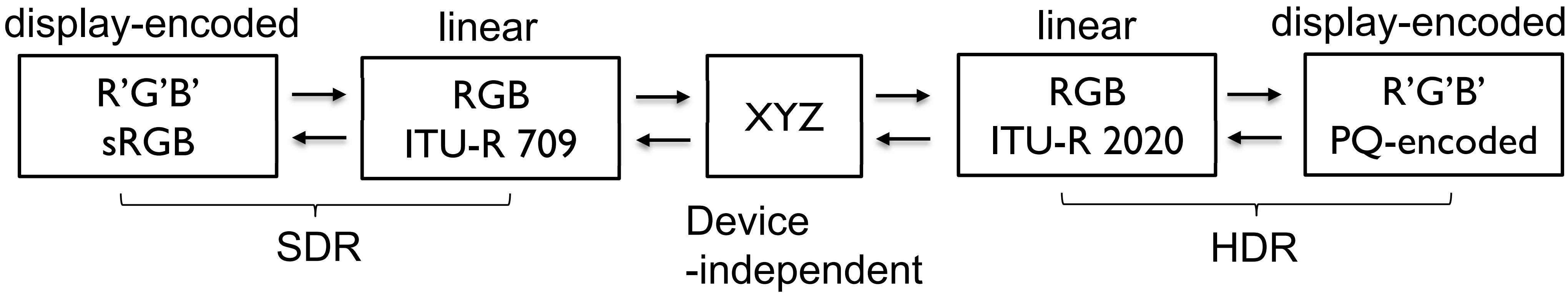
Colour space
What is the XYZ of "pure" red, green and blue



Electro-Optical Transfer Function
How to efficiently encode each primary colour



How to transform between RGB colour spaces (SDR and HDR)?



- From ITU-R 709 RGB to XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}_{R709toXYZ} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}_{R709}$$

Relative XYZ of the red primary

Relative XYZ of the green primary

Relative XYZ of the blue primary

Relative RGB (0-1) in the R709 space

How to transform between RGB colour spaces?

- From ITU-R **709** RGB to ITU-R **2020** RGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix}_{R2020} = M_{XYZtoR2020} \cdot M_{R709toXYZ} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}_{R709}$$

- From ITU-R **2020** RGB to ITU-R **709** RGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix}_{R709} = M_{XYZtoR709} \cdot M_{R2020toXYZ} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}_{R2020}$$

- Where:

$$M_{R709toXYZ} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \text{ and } M_{XYZtoR709} = M_{R709toXYZ}^{-1}$$

$$M_{R2020toXYZ} = \begin{bmatrix} 0.6370 & 0.1446 & 0.1689 \\ 0.2627 & 0.6780 & 0.0593 \\ 0.0000 & 0.0281 & 1.0610 \end{bmatrix} \text{ and } M_{XYZtoR2020} = M_{R2020toXYZ}^{-1}$$

Exercise 1: Map colour to a display

- We have:
  - Spectrum of the colour we want to reproduce: L (Nx1 vector)
  - XYZ sensitivities: S_{XYZ} (Nx3 matrix)
  - Spectra of the RGB primaries: P_{RGB} (Nx3 matrix)
  - Display gamma: $\gamma = 2.2$
- We need to find display-encoded R'G'B' colour values
  - Step 1: Find XYZ of the colour
  $$[X \quad Y \quad Z]^T = S_{XYZ}^T\, L$$
  - Step 2: Find a linear combination of RGB primaries
  $$S_{XYZ}^T\ P_{RGB} = M_{RGB\rightarrow XYZ}$$
  - Step 3: Convert and display-encode linear colour values
  $$[R \quad G \quad B]^T = M_{RGB\rightarrow XYZ}^{-1}\,[X \quad Y \quad Z]^T$$
  $$[R' \quad G' \quad B'] = [R^{1/\gamma} \quad G^{1/\gamma} \quad B^{1/\gamma}]$$

Exercise 2: Find a camera colour correction matrix

- We have:
  - XYZ sensitivities: S_{XYZ} (Nx3 matrix)
  - Spectral sensitivities of camera's RGB pixels: C_{RGB} (Nx3 matrix)
  - Spectrum in the real world: L (Nx1 vector)
- Find a 3x3 matrix mapping from camera's native RGB to XYZ

$$M_{C\to XYZ} C_{RGB}^T \, L \approx S_{XYZ}^T \, L$$

$$\mathrm{argmin}_{M_{C\to XYZ}} \left\| M_{C\to XYZ} C_{RGB}^T \, L - S_{XYZ}^T \, L \right\|_2$$

$$M_{C\to XYZ}^T = \left(C_{RGB}^T C_{RGB} \right)^{-1} C_{RGB}^T S_{XYZ}$$

- Show that a camera is colour-accurate if $C_{RGB}^T = N S_{XYZ}^T$

$$M N \, S_{XYZ}^T = S_{XYZ}^T \text{, where } M = N^{-1}$$

Any full rank 3x3 matrix

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

Colour perception and colour spaces

Part 5/5 – colour spaces

Rafał Mantiuk

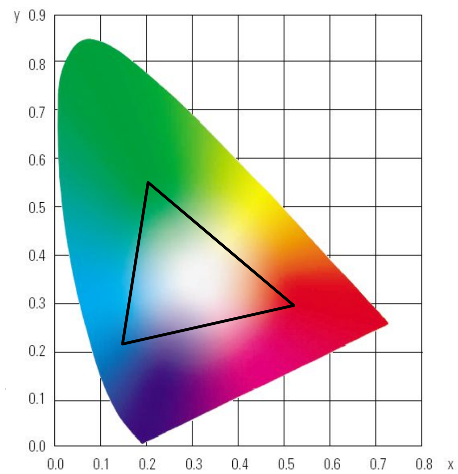
Computer Laboratory, University of Cambridge

Representing colour

- We need a way to represent colour in the computer by some set of numbers
  - A) preferably a small set of numbers which can be quantised to a fairly **small number of bits** each
    - Gamma corrected RGB, sRGB and CMYK for printers
  - B) a set of numbers that are **easy to interpret**
    - Munsell's *artists'* scheme
    - HSV, HLS
  - C) a set of numbers in a 3D space so that the (Euclidean) distance in that space corresponds to approximately **perceptually uniform** colour differences
    - CIE Lab, CIE Luv

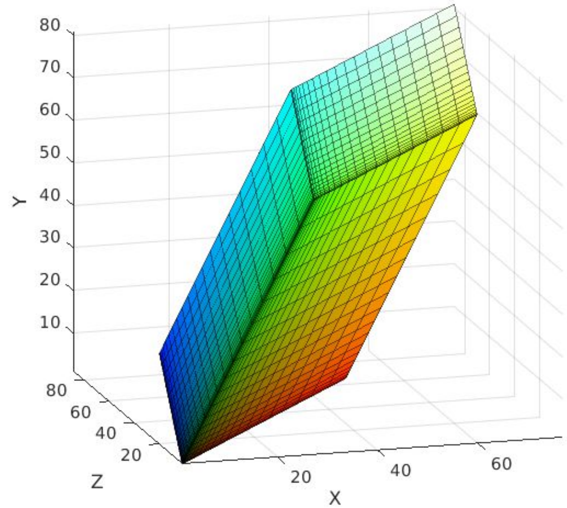
RGB spaces

- Most display devices that output light mix red, green and blue lights to make colour
  - televisions, CRT monitors, LCD screens
- RGB colour space
  - Can be **linear** (RGB) or **display-encoded** (R'G'B')
  - Can be **scene-referred** (HDR) or **display-referred** (SDR)
- There are multiple RGB colour spaces
  - ITU-R 709 (sRGB), ITU-R 2020, Adobe RGB, DCI-P3
    - Each using different primary colours
  - And different OETFs (gamma, PQ, etc.)
- Nominally, *RGB* space is a cube

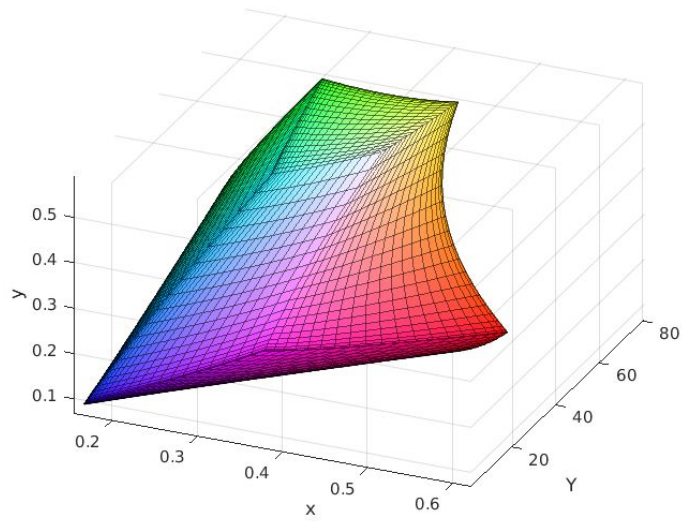


RGB in CIE *XYZ* space

- Linear RGB colour values can be transformed into CIE XYZ
  - by matrix multiplication
  - because it is a rigid transformation the colour gamut in CIE XYZ is a rotate and skewed cube
- Transformation into Yxy
  - is non-linear (non-rigid)
  - colour gamut is more complicated



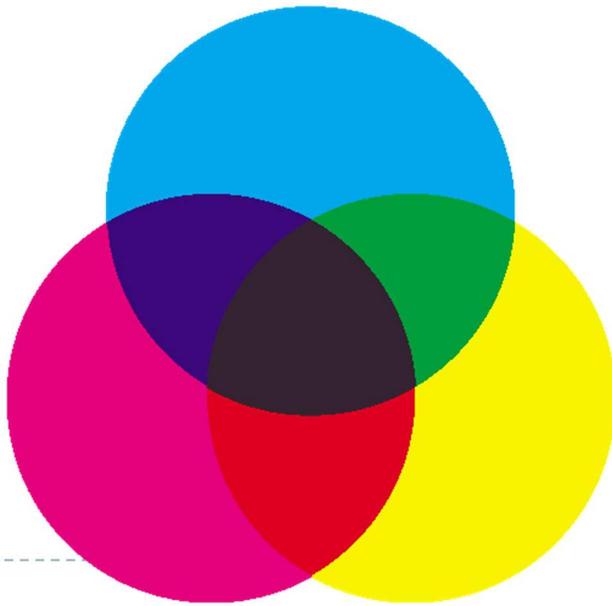
RGB gamut in XYZ colour space



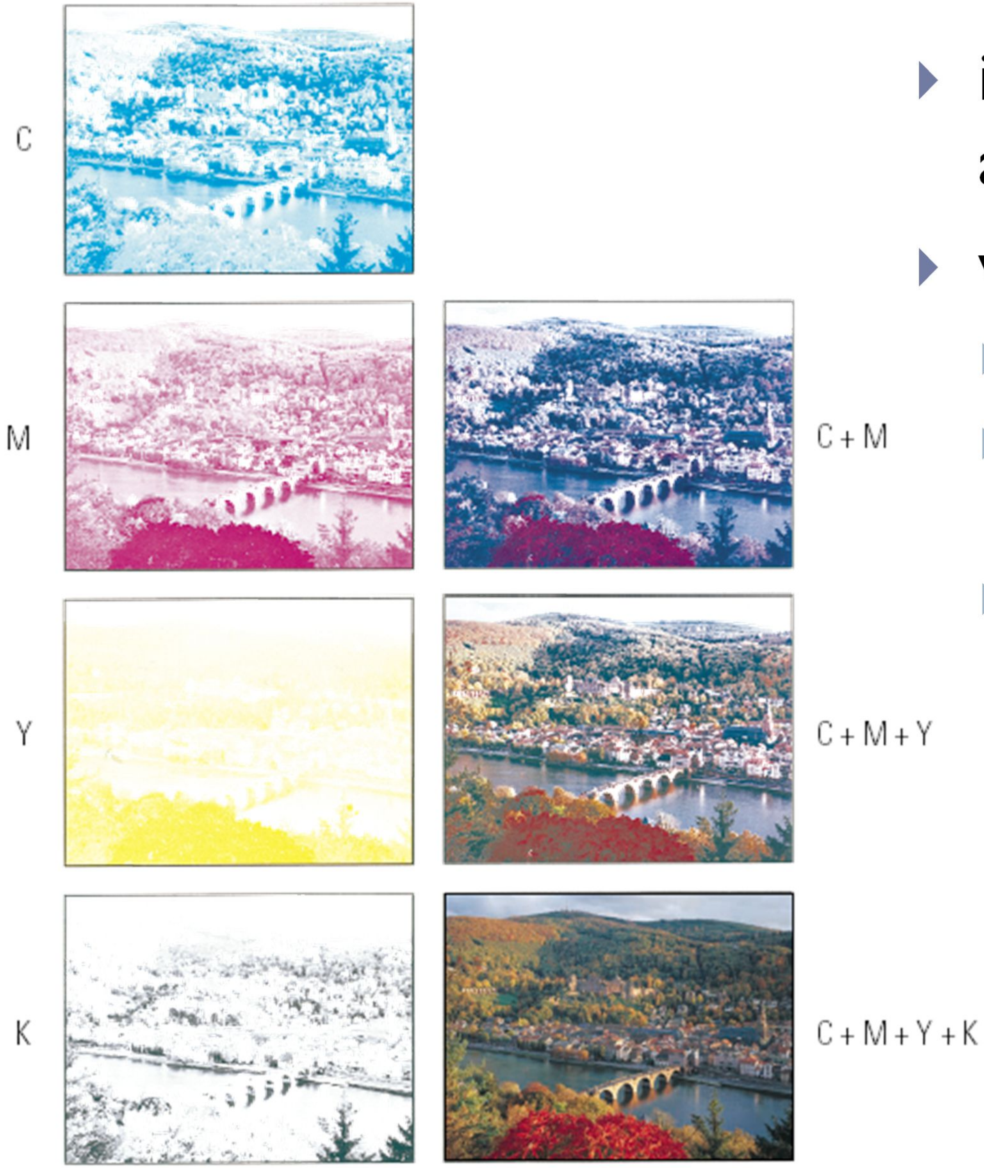
RGB gamut in Yxy colour space

CMY space

- printers make colour by mixing coloured inks
- the important difference between inks (*CMY*) and lights (*RGB*) is that, while lights *emit* light, inks *absorb* light
  - cyan absorbs red, reflects blue and green
  - magenta absorbs green, reflects red and blue
  - yellow absorbs blue, reflects green and red
- *CMY* is, at its simplest, the inverse of *RGB*
- *CMY* space is nominally a cube



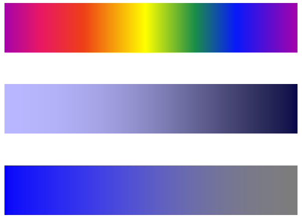
CMYK space

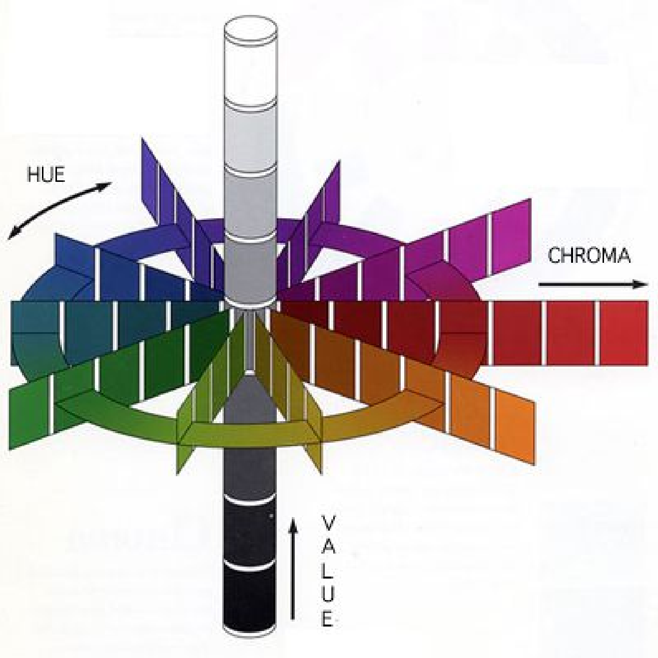


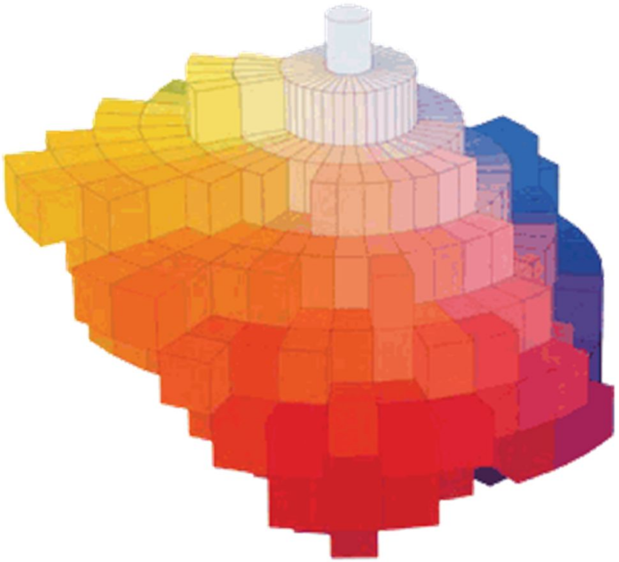
- in real printing we use black (key) as well as *CMY*
- why use black?
  - inks are not perfect absorbers
  - mixing $C + M + Y$ gives a muddy grey, not black
  - lots of text is printed in black: trying to align C, M and Y perfectly for black text would be a nightmare

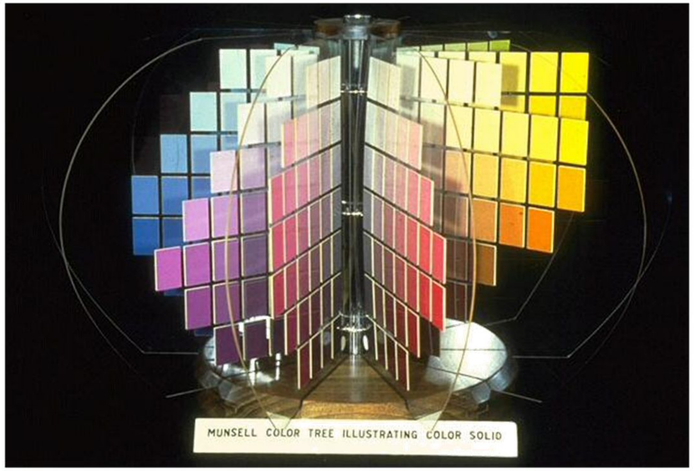
Munsell's colour classification system

- three axes
  - hue ➢ the dominant colour
  - value ➢ bright colours/dark colours
  - chroma ➢ vivid colours/dull colours
- can represent this as a 3D graph



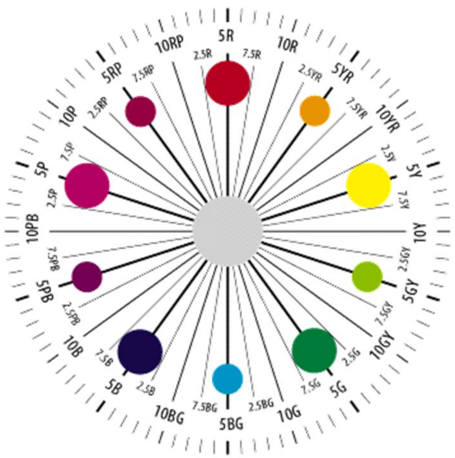


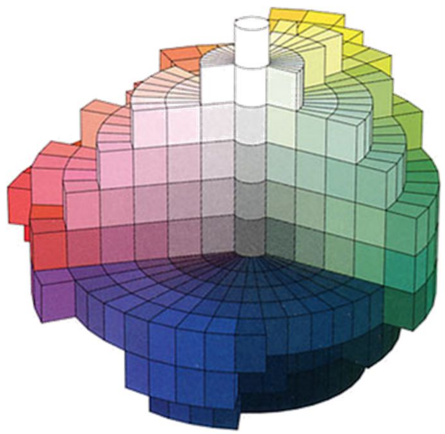


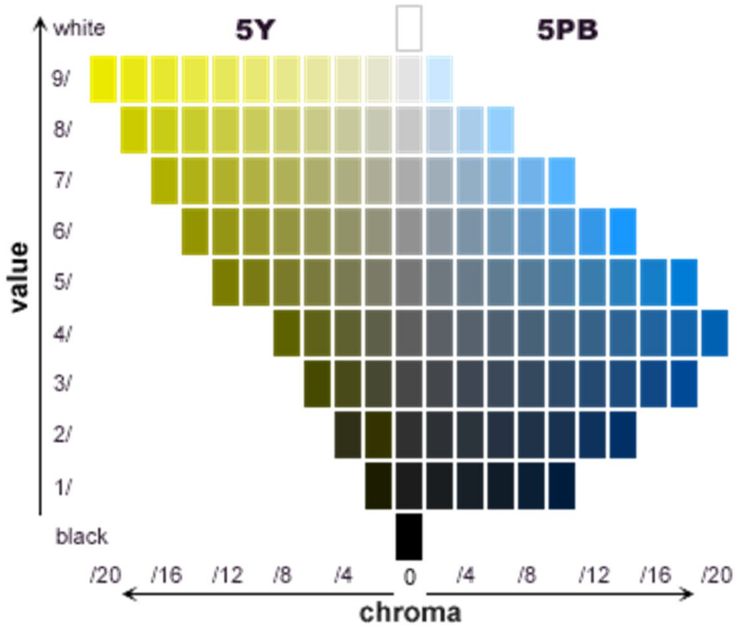


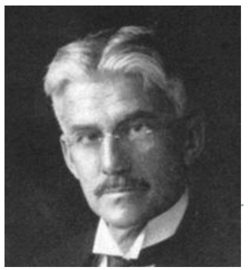
Munsell's colour classification system

- any two adjacent colours are a standard "perceptual" distance apart
  - worked out by testing it on people
  - a highly irregular space
    - e.g. vivid yellow is much brighter than vivid blue









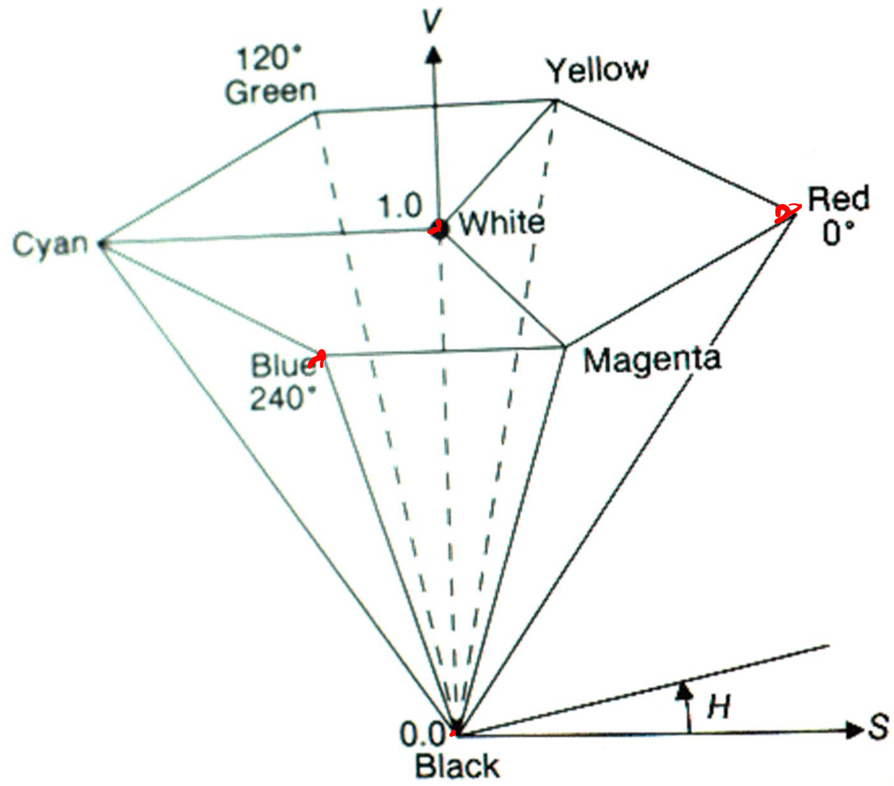
invented by Albert H. Munsell, an American artist, in 1905 in an attempt to systematically classify colours

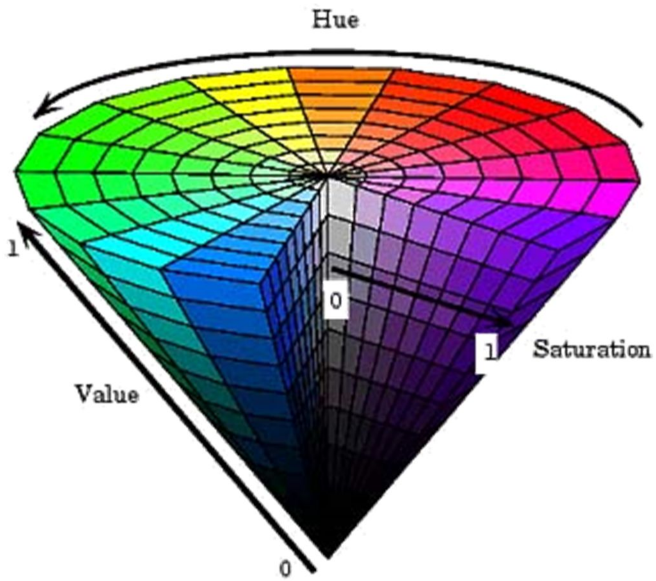
Colour spaces for user-interfaces

- *RGB* and *CMY* are based on the physical devices which produce the coloured output
- *RGB* and *CMY* are difficult for humans to use for selecting colours
- Munsell's colour system is much more intuitive:
  - hue — what is the principal colour?
  - value — how light or dark is it?
  - chroma — how vivid or dull is it?
- computer interface designers have developed basic transformations of *RGB* which resemble Munsell's human-friendly system

HSV: hue saturation value

- three axes, as with Munsell
  - hue and value have same meaning
  - the term "saturation" replaces the term "chroma"
  - simple conversion from gamma-corrected RGB to HSV

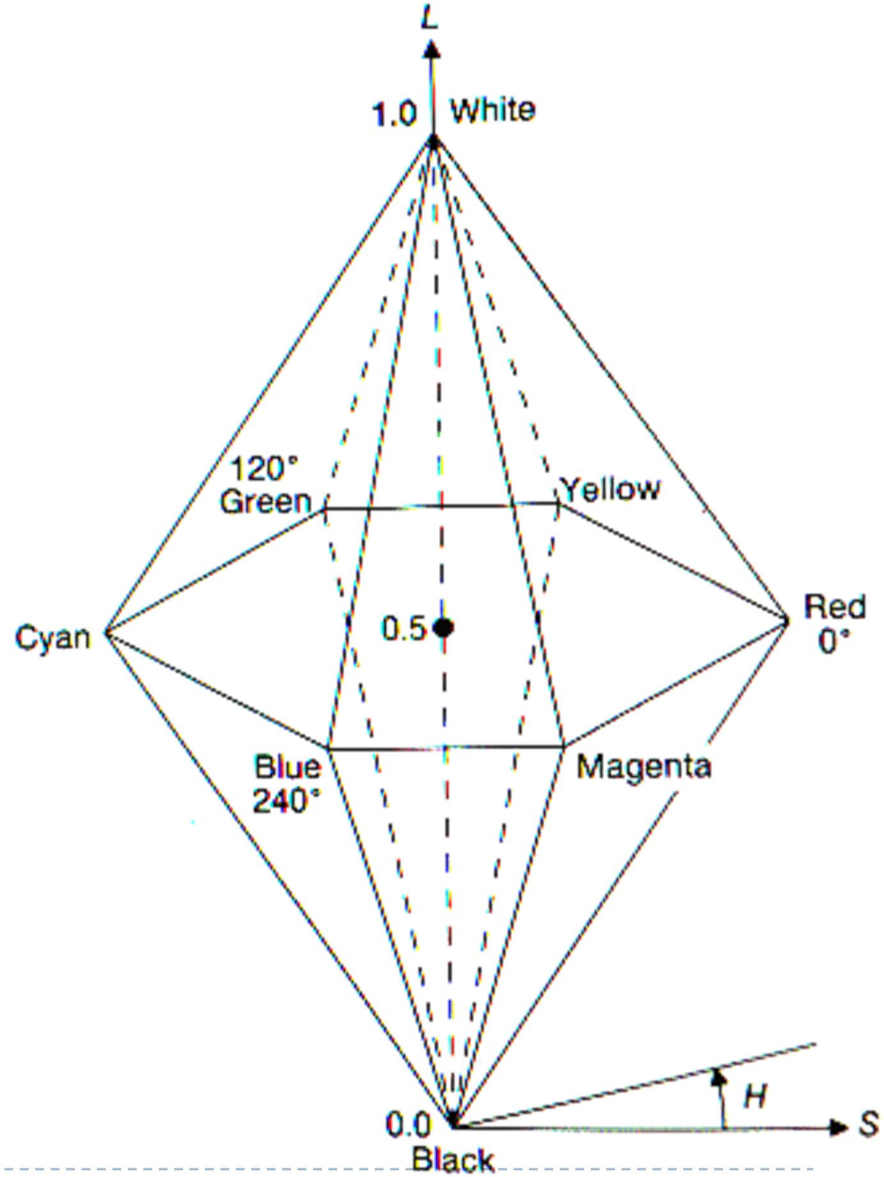




- designed by Alvy Ray Smith in 1978
- algorithm to convert *HSV* to *RGB* and back can be found in Foley et al., Figs 13.33 and 13.34

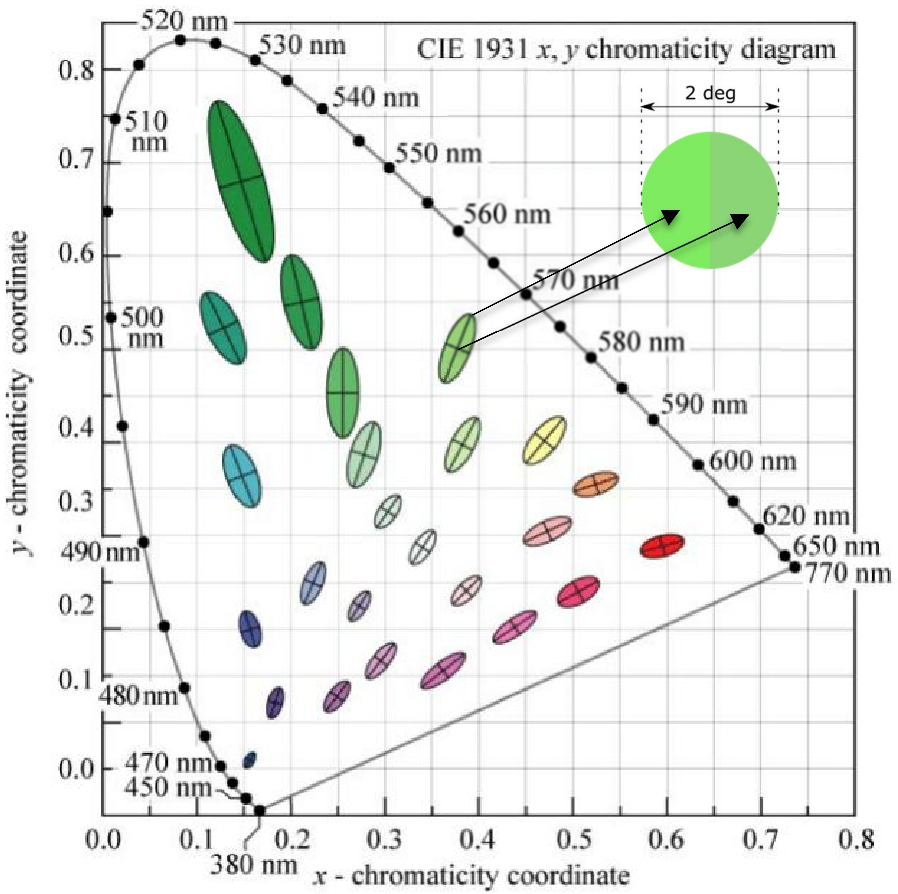
HLS: hue lightness saturation

- a simple variation of *HSV*
  - hue and saturation have same meaning
  - the term "lightness" replaces the term "value"
- designed to address the complaint that *HSV* has all pure colours having the same lightness/value as white
  - designed by Metrick in 1979
  - algorithm to convert *HLS* to *RGB* and back can be found in Foley et al., Figs 13.36 and 13.37

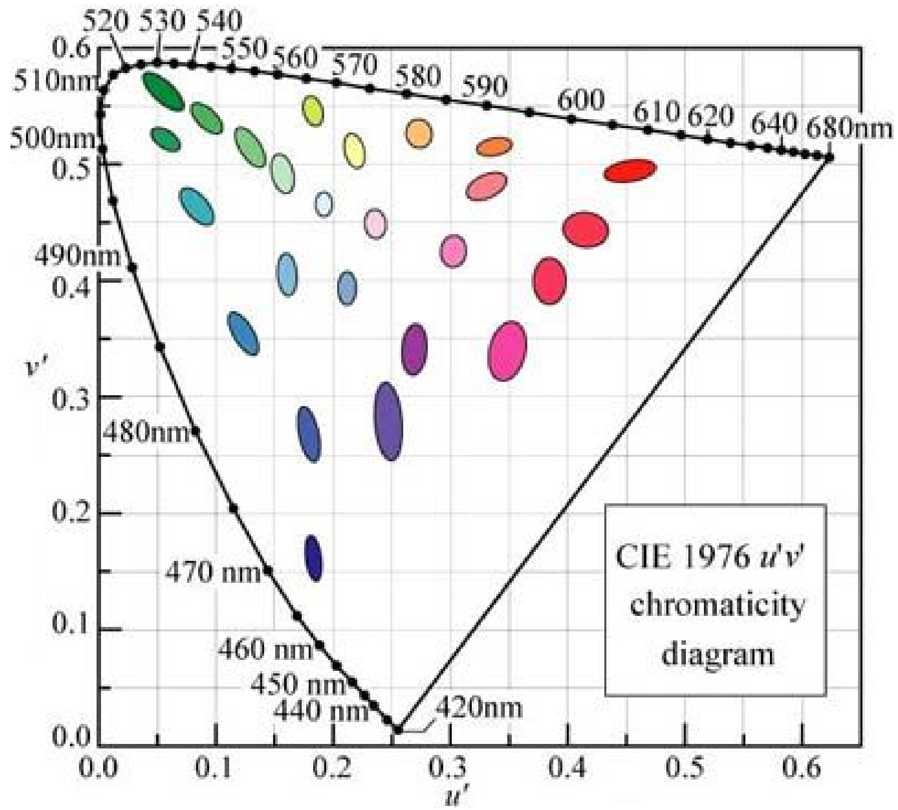


Perceptually uniformity

- MacAdam ellipses & visually indistinguishable colours

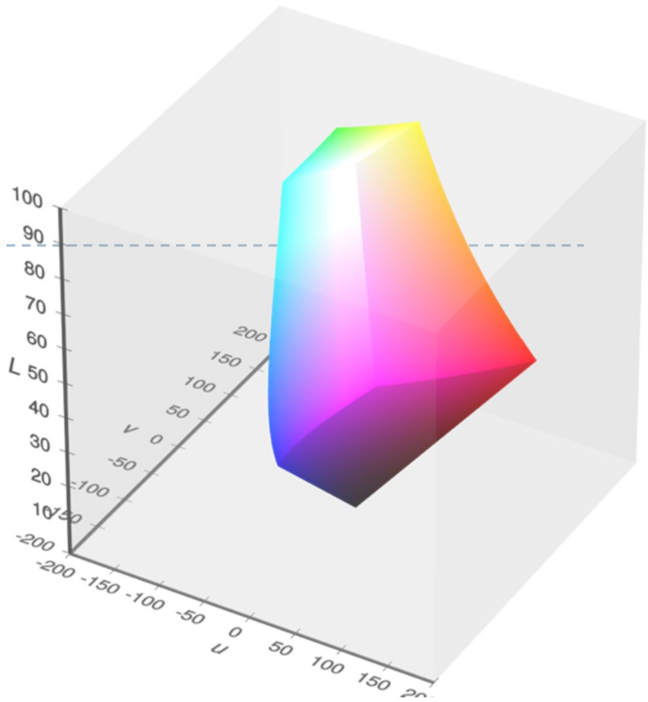


In CIE xy chromatic coordinates



In CIE u'v' chromatic coordinates

CIE L*u*v* and u'v'



sRGB in CIE L*u*v*

- Approximately perceptually uniform
- u'v' chromacity

$$u' = \frac{4X}{X + 15Y + 3Z} = \frac{4x}{-2x + 12y + 3}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} = \frac{9y}{-2x + 12y + 3}$$

- CIE LUV

Lightness

$$L^* = \begin{cases} \left(\frac{29}{3}\right)^3 Y/Y_n, & Y/Y_n \leq \left(\frac{6}{29}\right)^3 \\ 116(Y/Y_n)^{1/3} - 16, & Y/Y_n > \left(\frac{6}{29}\right)^3 \end{cases}$$

Chromacity coordinates

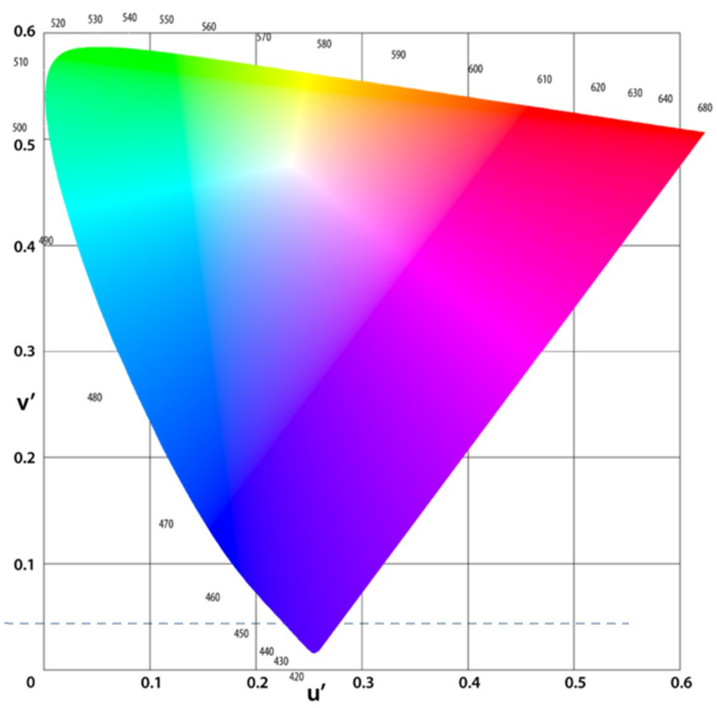
$$u^* = 13L^* \cdot (u' - u'_n)$$
$$v^* = 13L^* \cdot (v' - v'_n)$$

Colours less distinguishable when dark

- Hue and chroma

$$C^*_{uv} = \sqrt{(u^*)^2 + (v^*)^2}$$

$$h_{uv} = \operatorname{atan2}(v^*, u^*),$$



CIE L*a*b* colour space

- Another approximately perceptually uniform colour space

$$L^\star = 116 f\left(\frac{Y}{Y_n}\right) - 16$$

$$a^\star = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right)$$

$$b^\star = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)$$

Trichromatic values of the white point, e.g.

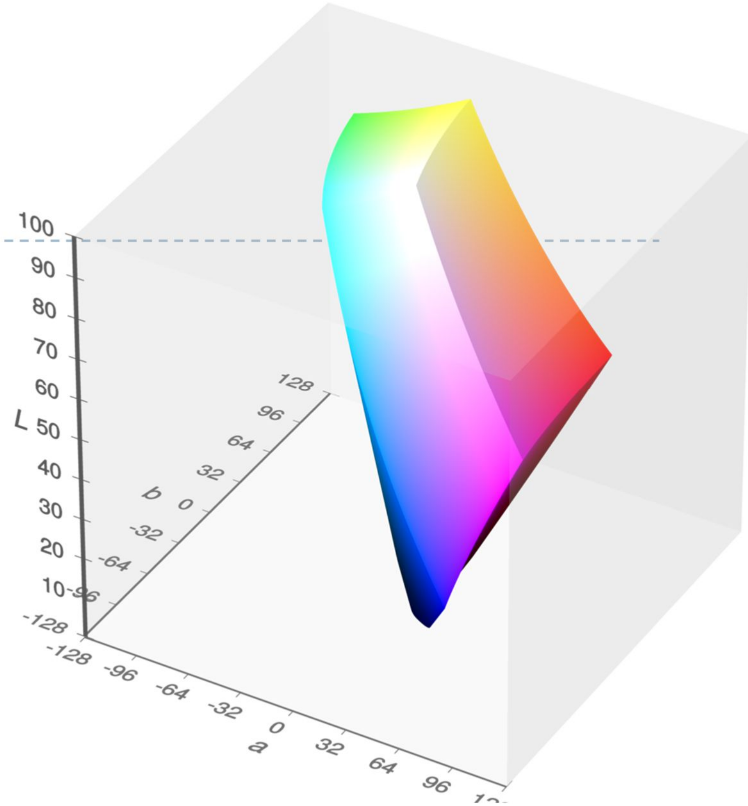
$$X_n = 95.047,$$
$$Y_n = 100.000,$$
$$Z_n = 108.883$$

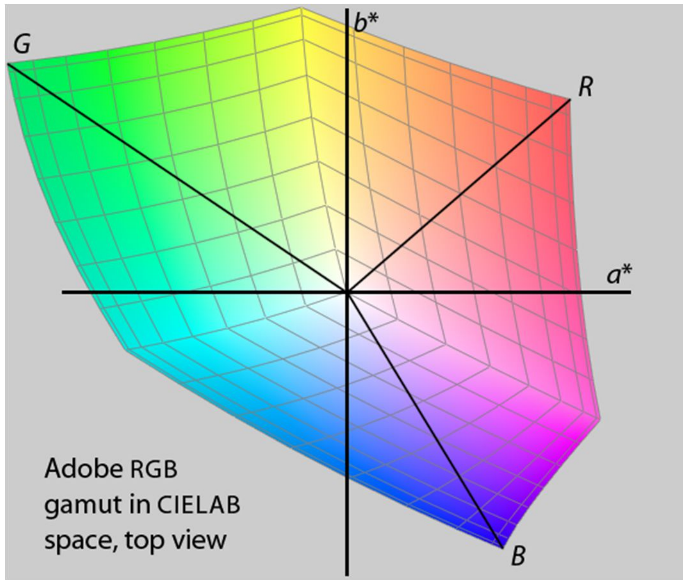
$$f(t) = \begin{cases} \sqrt[3]{t} & \text{if } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29} & \text{otherwise} \end{cases}$$

$$\delta = \frac{6}{29}$$

- Chroma and hue

$$C^\star = \sqrt{a^{\star 2} + b^{\star 2}}, \qquad h^\circ = \arctan\left(\frac{b^\star}{a^\star}\right)$$





Adobe RGB gamut in CIELAB space, top view

Lab space

- this visualization shows those colours in *Lab* space which a human can perceive
- again we see that human perception of colour is not uniform
  - perception of colour diminishes at the white and black ends of the L axis
  - the maximum perceivable chroma differs for different hues

Colour - references

- Chapters „Light" and „Colour" in
  - Shirley, P. & Marschner, S., *Fundamentals of Computer Graphics*
- Textbook on colour appearance
  - Fairchild, M. D. (2005). *Color Appearance Models* (second.). John Wiley & Sons.
- Comprehensive review of colour research
  - Wyszecki, G., & Stiles, W. S. (2000). *Color science: concepts and methods, quantitative data, and formulae* (Second ed.). John Wiley & Sons.

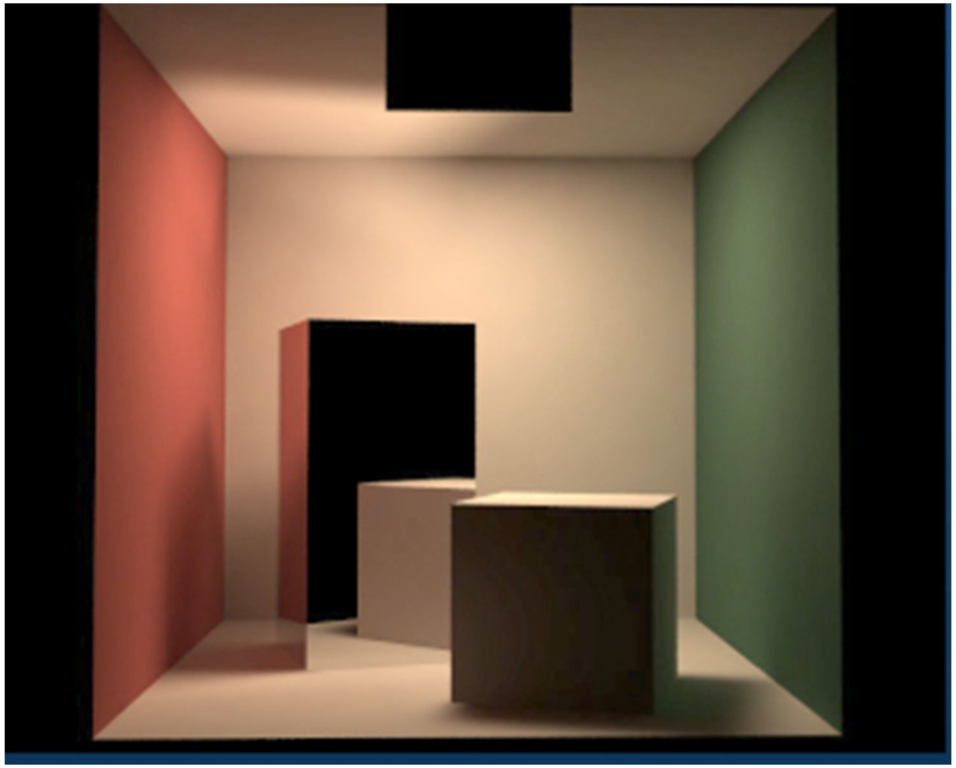
Advanced Graphics and Image Processing

High dynamic range and tone mapping

Part 1/2 – context, the need for tone-mapping

Rafał Mantiuk
Computer Laboratory, University of Cambridge

Cornell Box: need for tone-mapping in graphics



Rendering

Photograph

Real-world scenes are more challenging

- The match could not be achieved if the light source in the top of the box was visible
- The display could not reproduce the right level of brightness

Dynamic range

Luminance

$$\frac{\max\ L}{\min\ L}$$

(for SNR>3)

Dynamic range (contrast)

- As ratio:

$$C = \frac{L_{\max}}{L_{\min}}$$

  - Usually written as C:1, for example 1000:1.

- As "orders of magnitude" or log10 units:

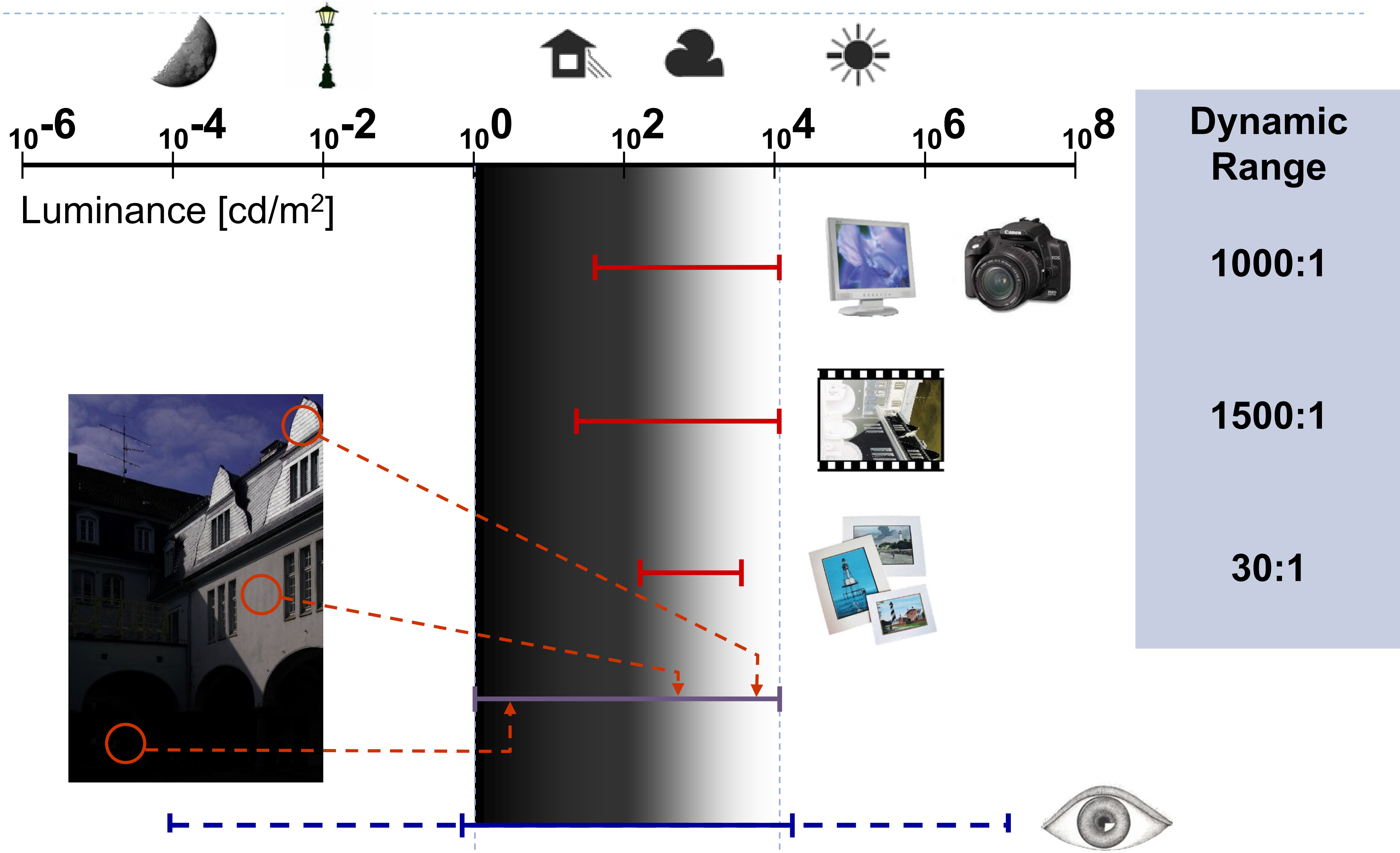
$$C_{10} = \log_{10} \frac{L_{\max}}{L_{\min}}$$

- As stops:

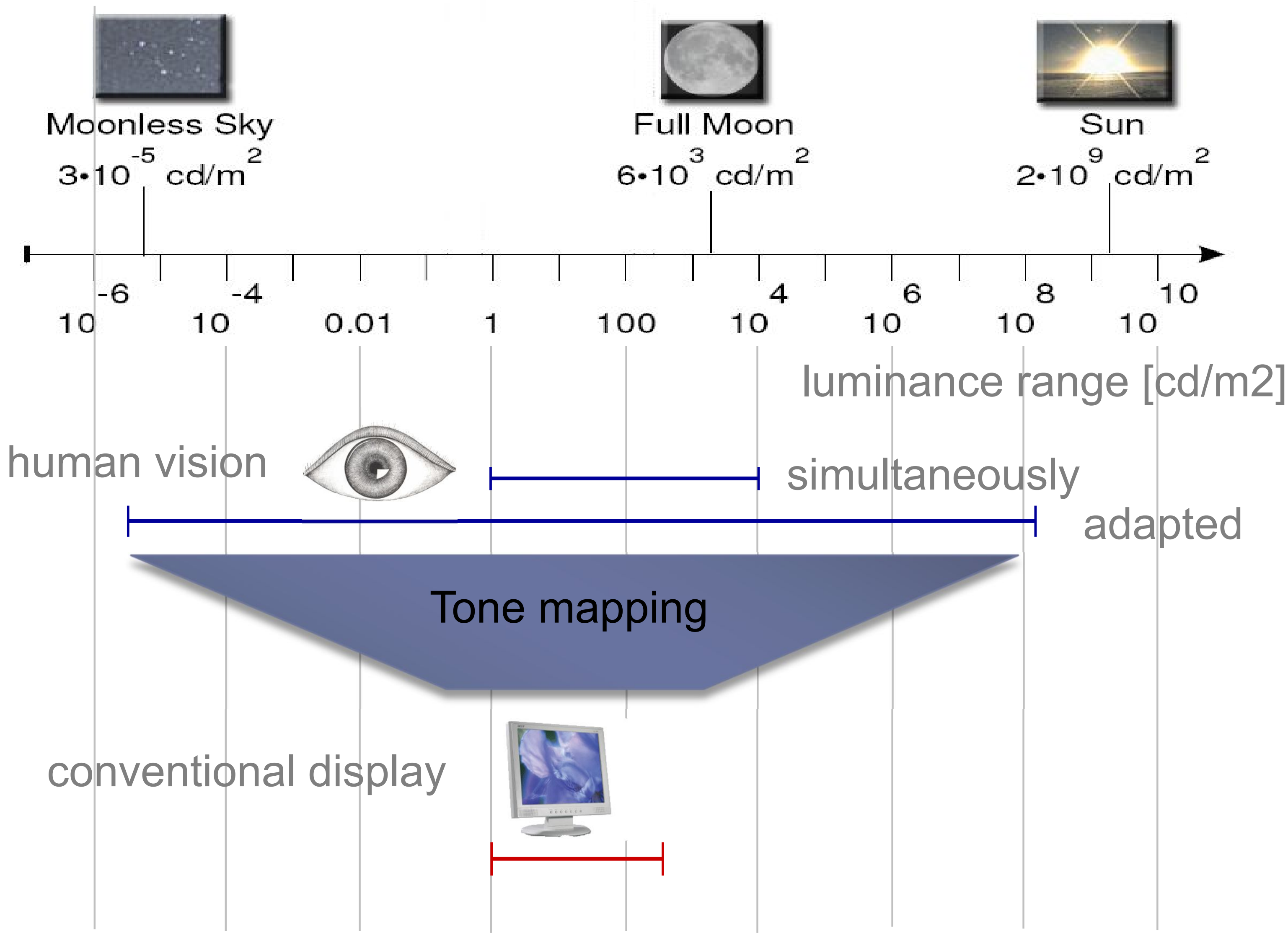
$$C_2 = \log_2 \frac{L_{\max}}{L_{\min}}$$

One stop is doubling of halving the amount of light

High dynamic range (HDR)



Tone-mapping problem



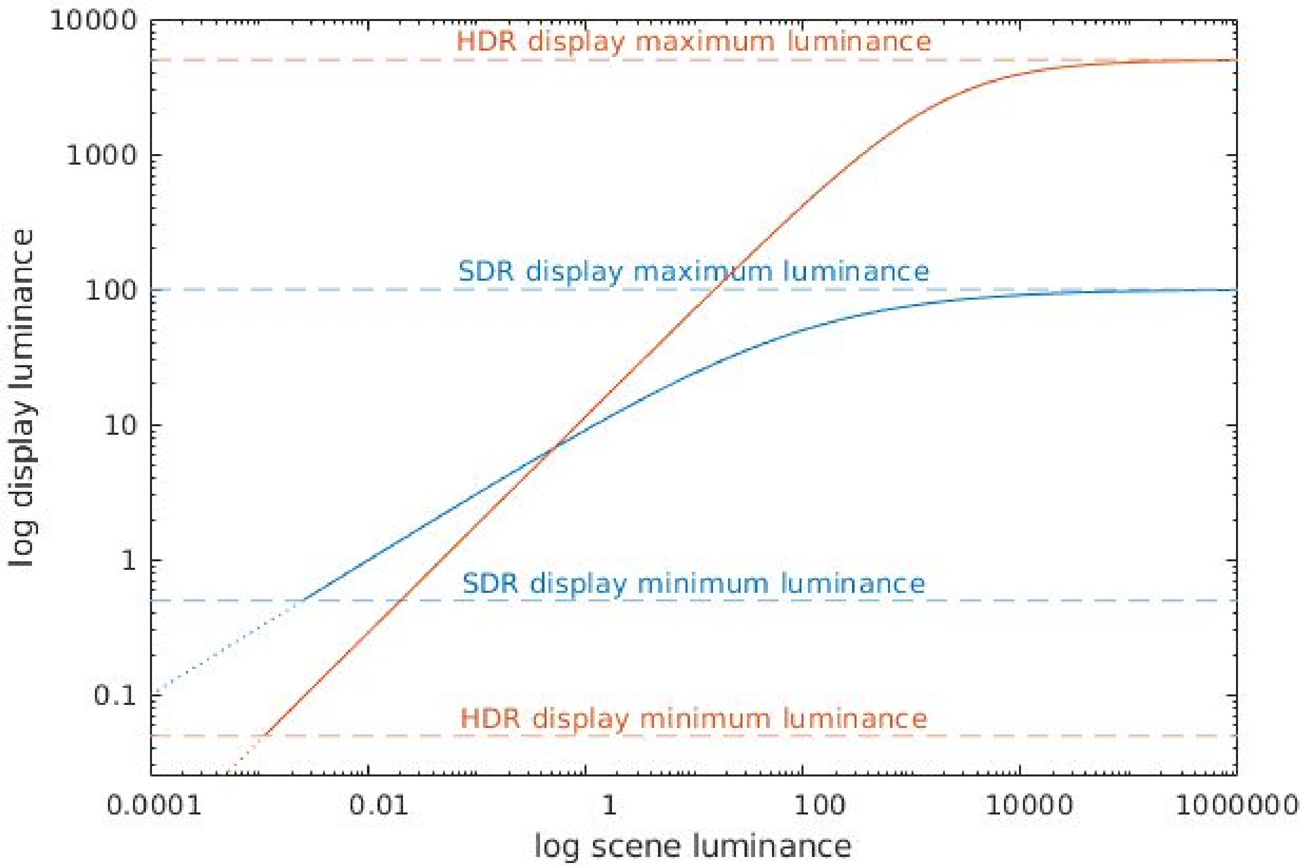
Why do we need tone mapping?

- To **reduce dynamic range**
- To **customize the look**
  - colour grading
- To **simulate human vision**
  - for example night vision
- To adapt displayed images to a **display and viewing conditions**
- To make rendered images look **more realistic**
- To map from **scene- to display-referred** colours

- Different tone mapping operators achieve different goals

From scene- to display-referred colours

- The primary purpose of tone mapping is to transform an image from *scene-referred* to *display-referred* colours



Tone-mapping in rendering

- Any physically-based rendering requires tone-mapping
- “HDR rendering” in games is pseudo-physically-based rendering
- Goal: to simulate a camera or the eye
- Greatly enhances realism

LDR illumination
No tone-mapping

HDR illumination
Tone-mapping

Half-Life 2: Lost coast

Rendering engine → Linear RGB → Tone mapping → Linear RGB → Display encoding →

SDR: Gamma-encoded
HDR: PQ-encoded

Simulate  or 

Basic tone-mapping and display coding

- The simplest form of tone-mapping is the exposure/brightness adjustment:

Display-referred red value

$$R_d = \frac{R_s}{L_{white}}$$

Scene-referred

Scene-referred luminance of white

  - R for red, the same for green and blue
  - No contrast compression, only for a moderate dynamic range
- The simplest form of display coding is the "gamma"

Prime (') denotes a gamma-corrected value

$$R' = (R_d)^{\frac{1}{\gamma}}$$

Typically γ=2.2

  - For SDR displays only

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

High dynamic range and tone mapping

Part 2/2 – tone mapping techniques

Rafał Mantiuk

Computer Laboratory, University of Cambridge

Techniques

- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Arithmetic of HDR images

- How do the basic arithmetic operations
  - Addition
  - Multiplication
  - Power function

affect the appearance of an HDR image?

- We work in the luminance space (NOT luma)
- The same operations can be applied to linear RGB
  - Or only to luminance and the colour can be transferred

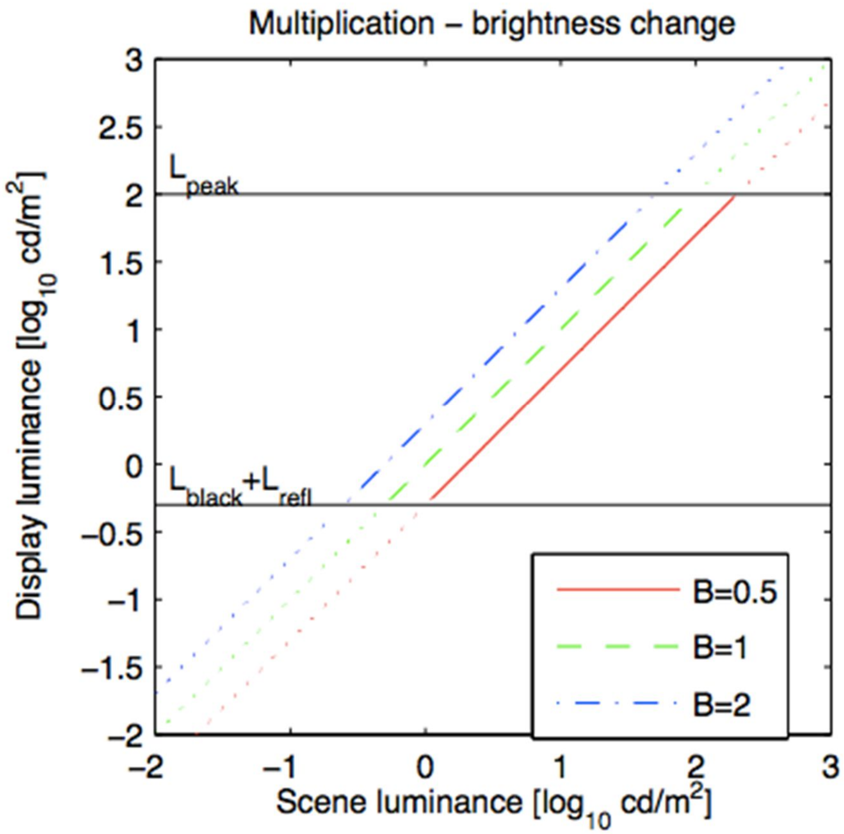
Multiplication – brightness change

Resulting luminance

Input luminance

$$T(L_p) = B \cdot L_p$$

Brightness change parameter



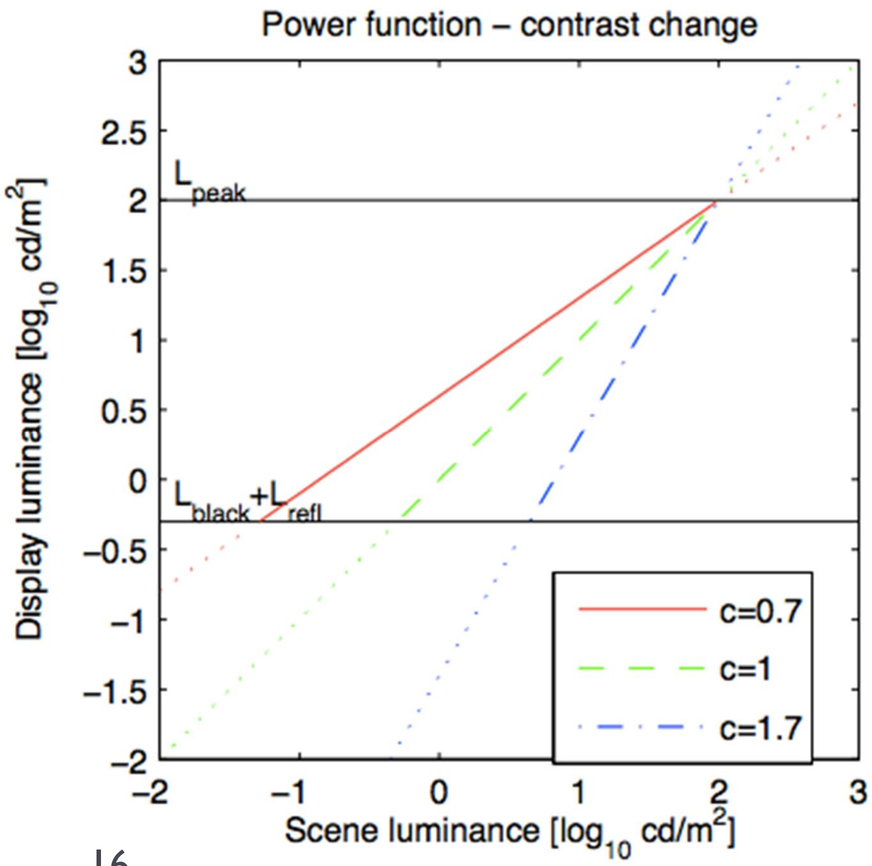
- Multiplication makes the image brighter or darker
- It does not change the dynamic range!

Power function – contrast change

$$T(L_p) = L_{peak}\left(\frac{L_p}{L_{white}}\right)^c$$

Contrast change (gamma)

Luminance to be mapped to white

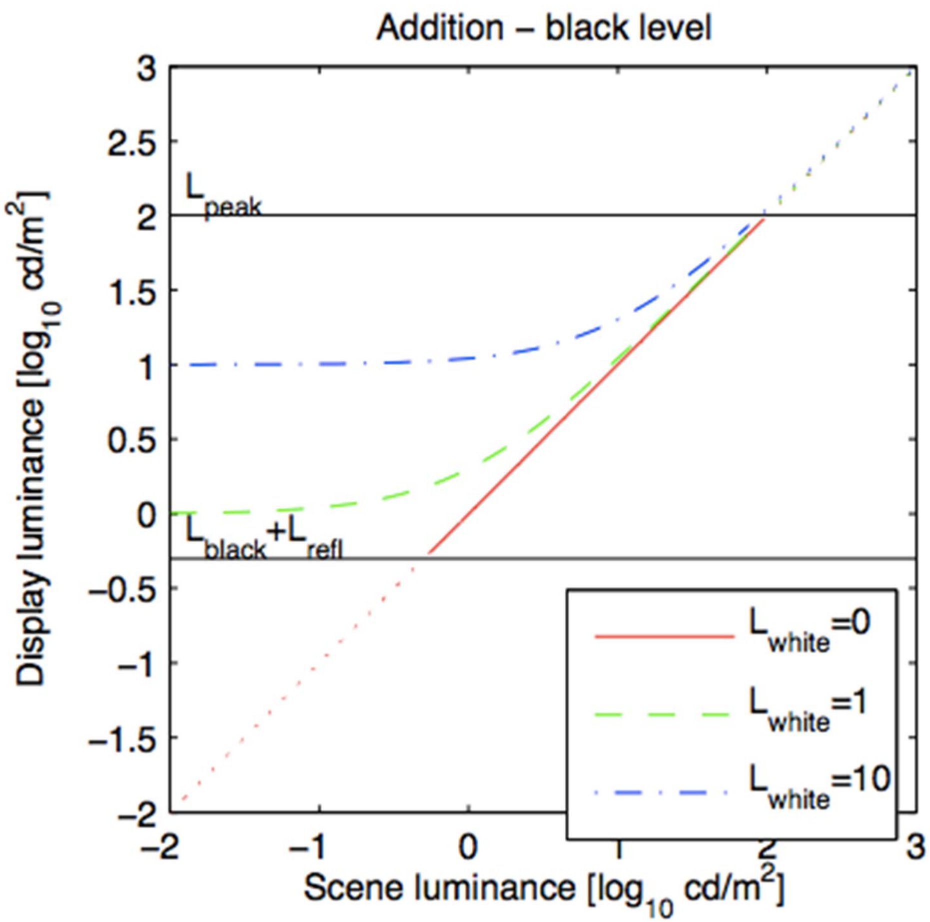


- Power function stretches or shrinks the dynamic range of an image
- It is usually performed relative to a reference white colour (and luminance)
- Side effect: brightness of the dark image part will change
- Slope on a log-log plot explains contrast change

Addition – black level

$$T(L_p) = L_p + F$$

Black level (flare, fog)



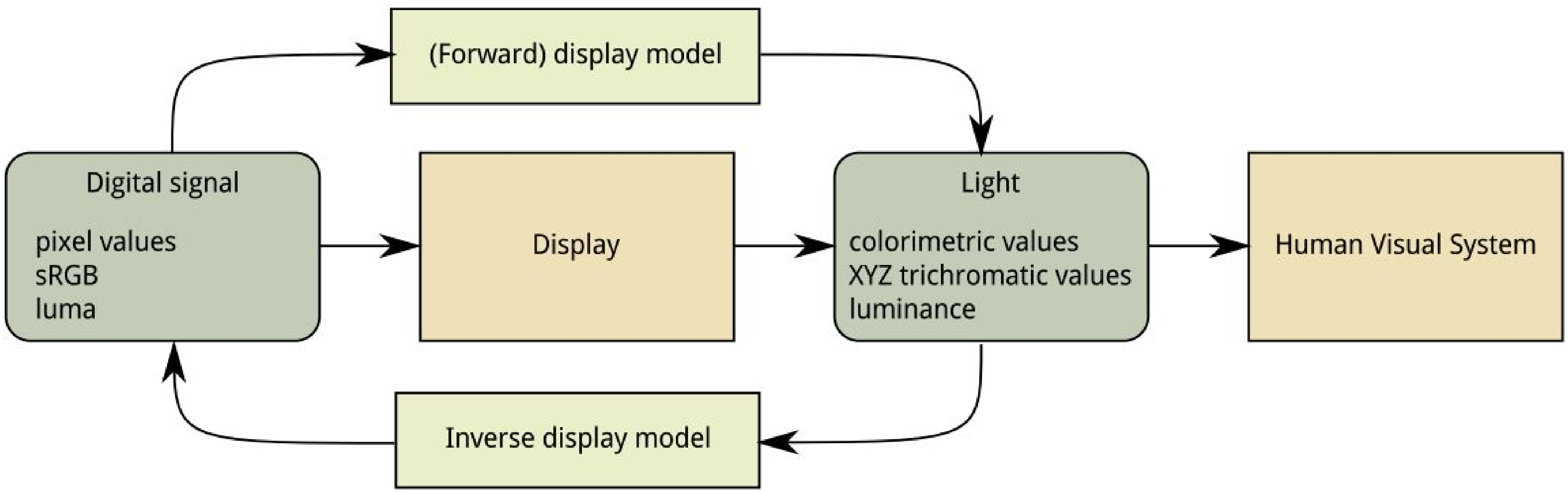
- Addition elevates black level, adds „fog" to an image
- It affects mostly darker tones
- It reduces image dynamic range
- Subtraction can compensate for ambient light (shown next)

Techniques

- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Display-adaptive tone mapping

- Tone-mapping can account for the physical model of a display
  - How a display transforms pixel values into emitted light
  - Useful for ambient light compensation



Has a similar role as display encoding, but can account for viewing conditions

(Forward) Display model

- GOG: Gain-Offset-Gamma

Luminance

Peak luminance

Gamma

Display black level

Screen reflections

$$L = \underbrace{(L_{peak} - L_{black})}_{\text{Gain}} V^{\gamma} + \underbrace{L_{black} + L_{refl}}_{\text{Offset}}$$

Pixel value 0-1

Reflectance factor (0.01)

$$L_{refl} = \frac{k}{\pi} E_{amb}$$

Ambient illumination (in lux)

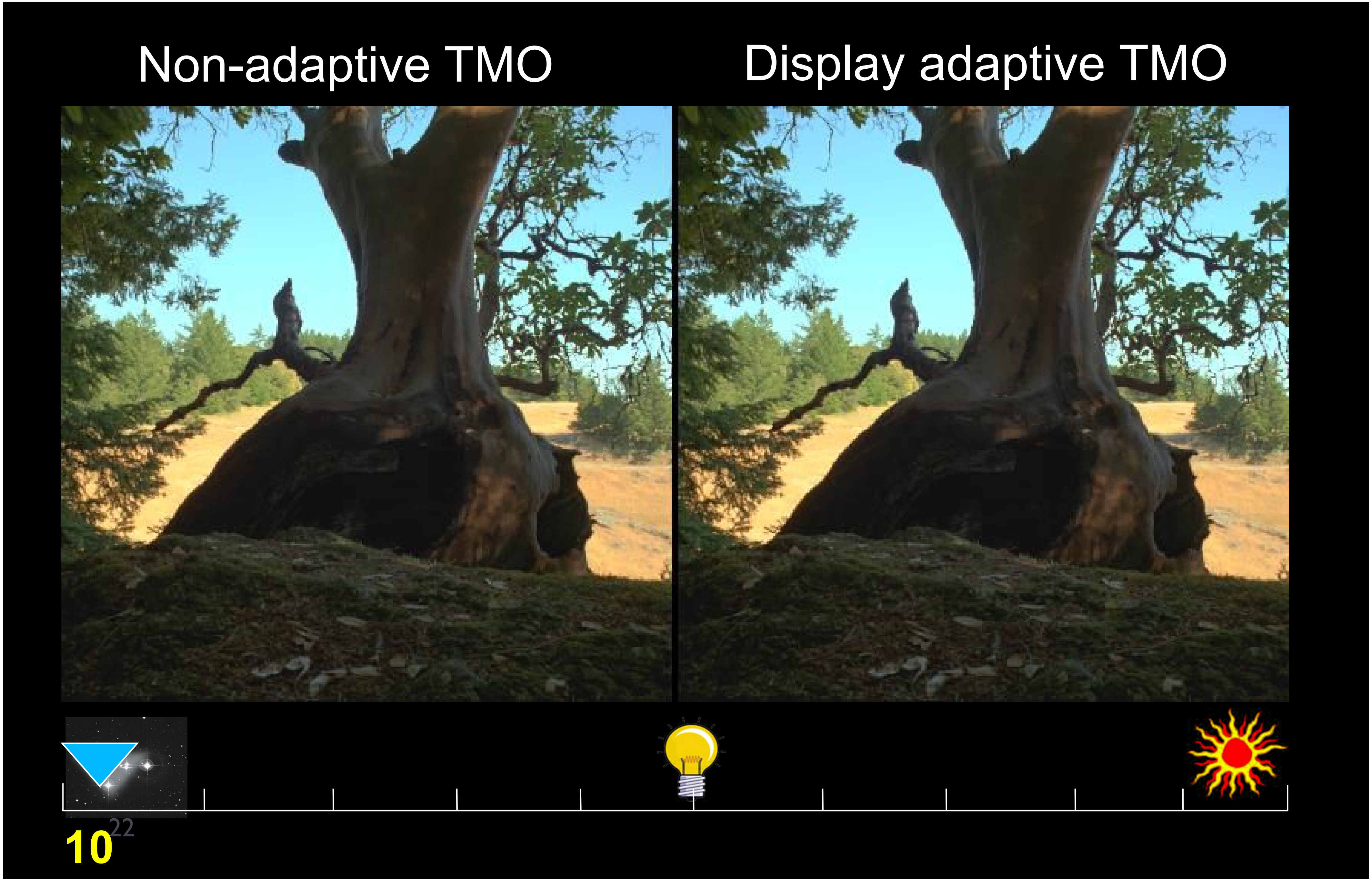
Inverse display model

Symbols are the same as for the forward display model

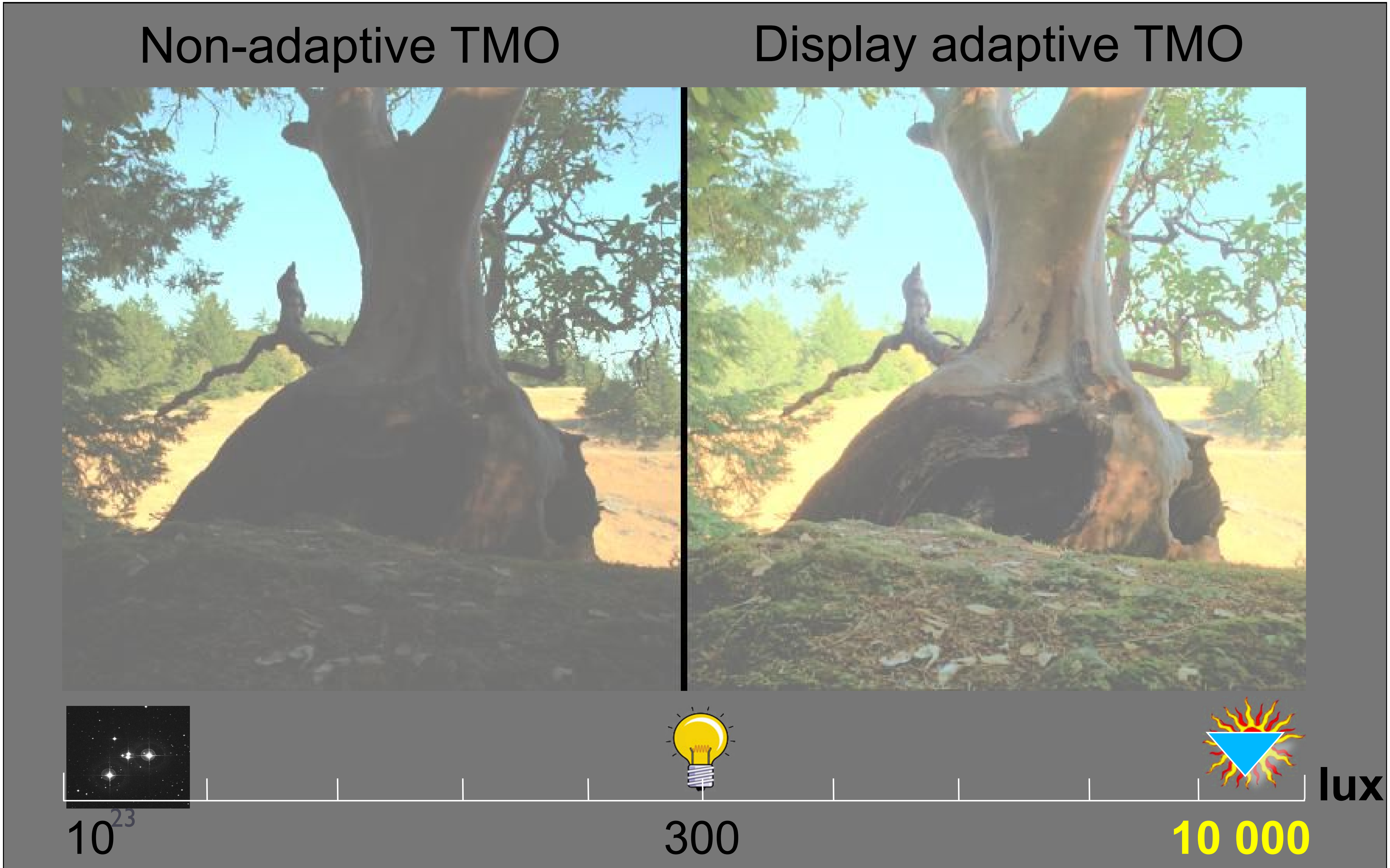
$$V = \left(\frac{L - L_{black} - L_{refl}}{L_{peak} - L_{black}} \right)^{(1/\gamma)}$$

Note: This display model does not address any colour issues. The same equation is applied to red, green and blue color channels. The assumption is that the display primaries are the same as for the sRGB color space.

Ambient illumination compensation



Ambient illumination compensation



Example: Ambient light compensation

- We are looking at the screen in bright light

$L_{peak} = 100\ [cd \cdot m^{-2}]$ $\qquad k = 0.005$

Modern screens have reflectivity of around 0.5%

$L_{black} = 0.1\ [cd \cdot m^{-2}]$

$E_{amb} = 2000\ [lux]$ $\qquad L_{refl} = \frac{0.005}{\pi} 2000 = 3.183\ [cd \cdot m^{-2}]$

- We assume that the dynamic of the input is 2.6 (≈400:1)

$$r_{in} = 2.6 \qquad r_{out} = \log_{10} \frac{L_{peak}}{L_{black} + L_{refl}} = 1.77$$

- First, we need to compress contrast to fit the available dynamic range, then compensate for ambient light

$$L_{out} = \left(\frac{L_{in}}{L_{wp}}\right)^{\frac{r_{out}}{r_{in}}} - L_{refl}$$

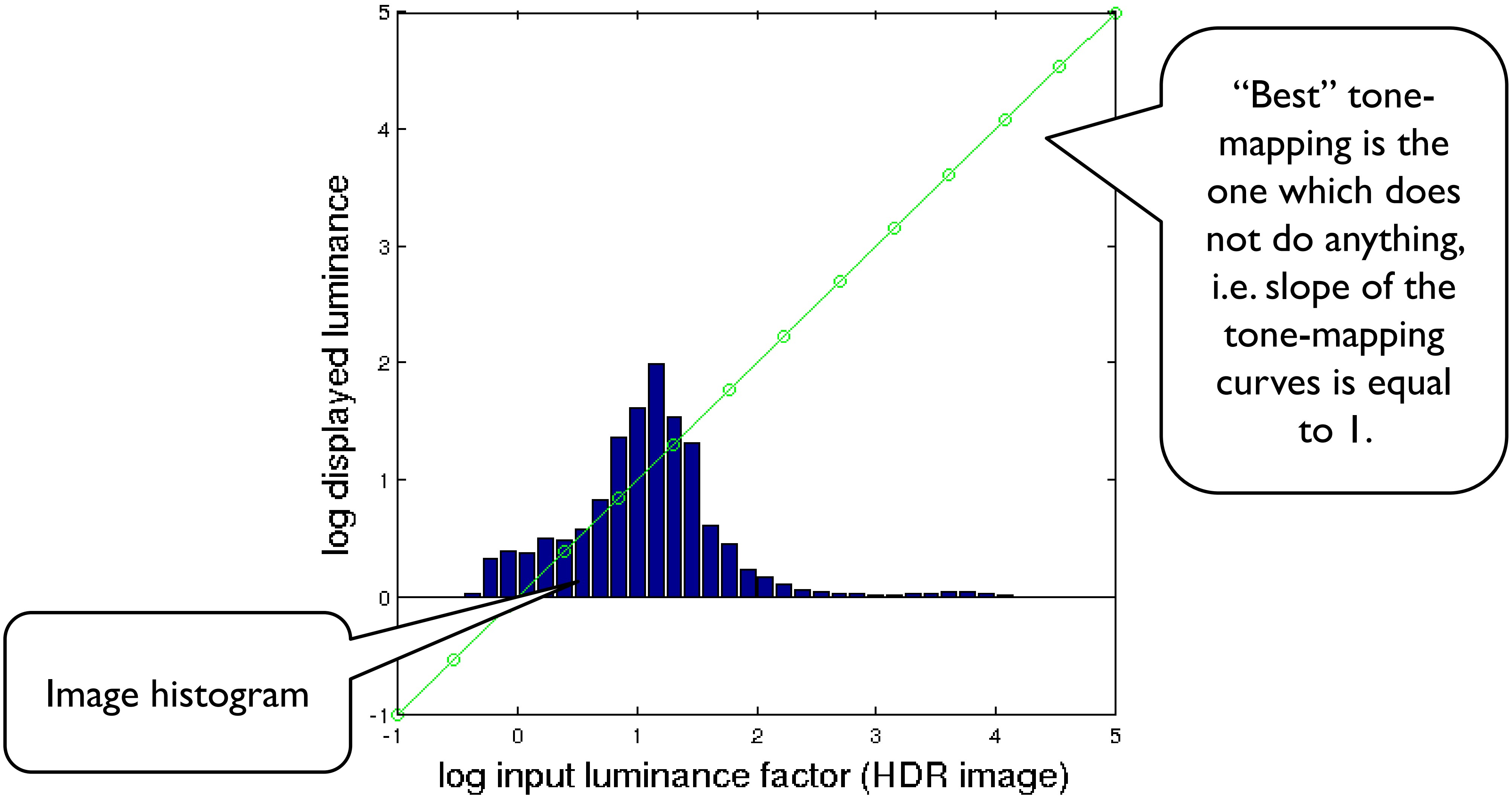
The resulting value is in luminance, must be mapped to display luma / gamma corrected values (display encoded)

Simplest, but not the best tone mapping

Techniques

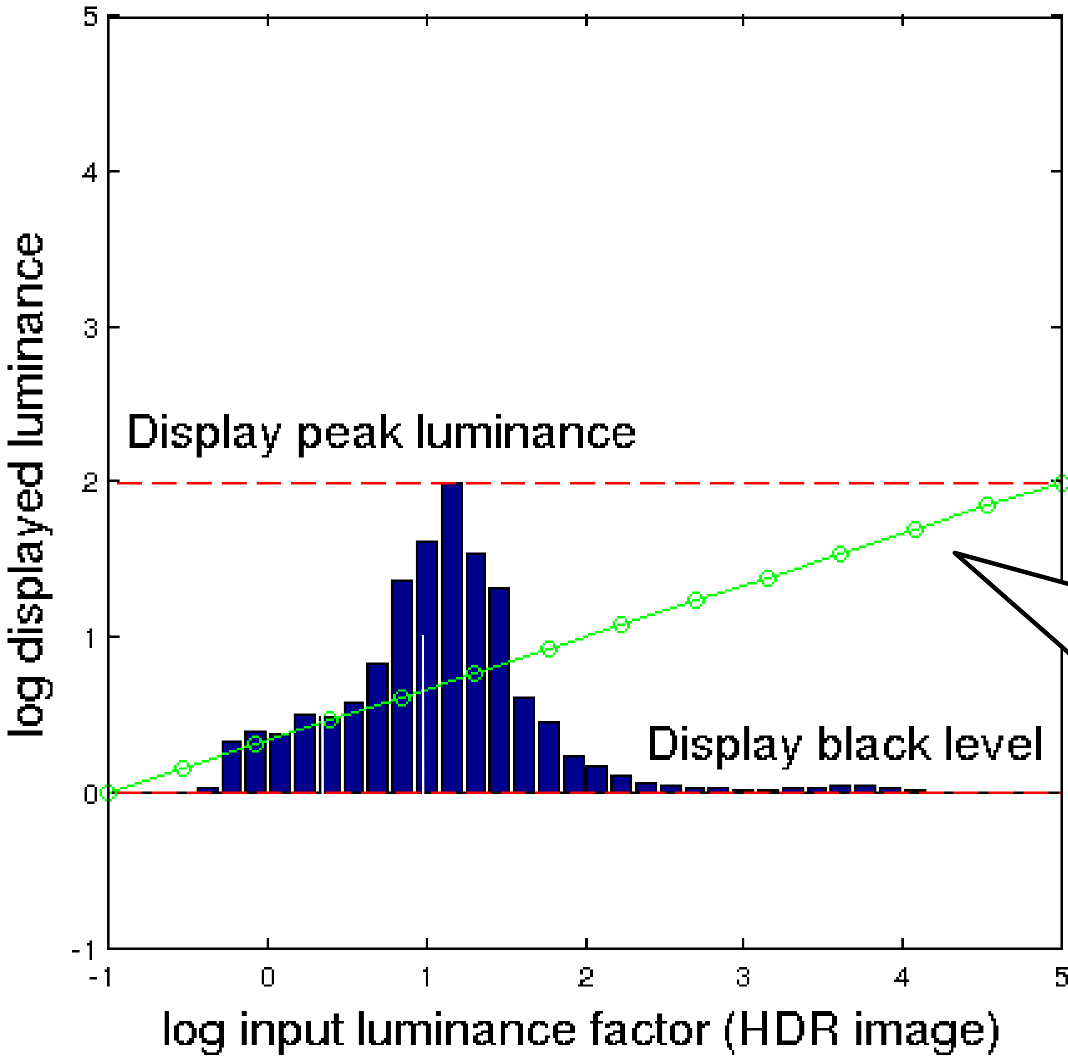
- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Tone-curve



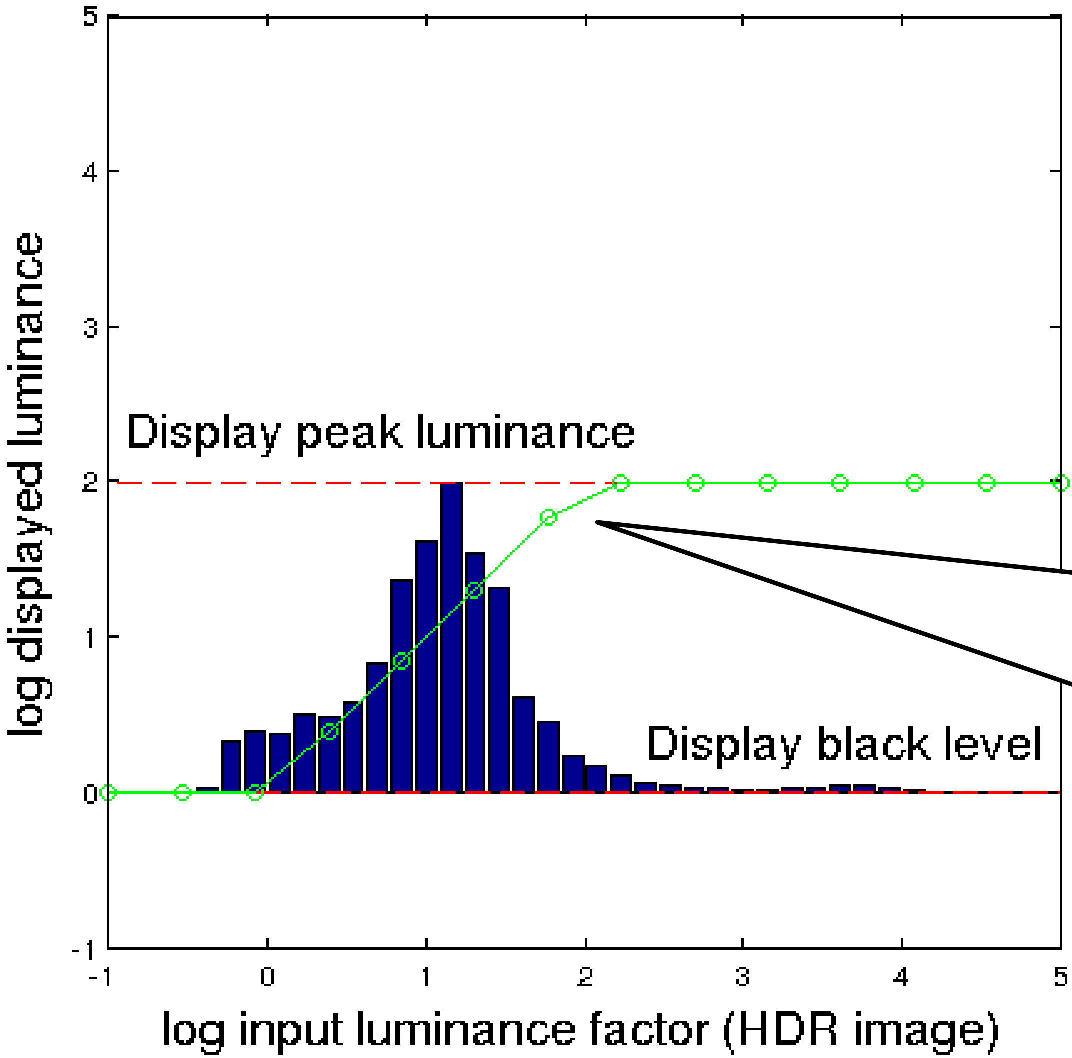
5
4
3
2
1
0
-1
log displayed luminance
-1
0
1
2
3
4
5
log input luminance factor (HDR image)
"Best" tone-mapping is the one which does not do anything, i.e. slope of the tone-mapping curves is equal to 1.
Image histogram

Tone-curve



But in practice contrast (slope) must be limited due to display limitations.

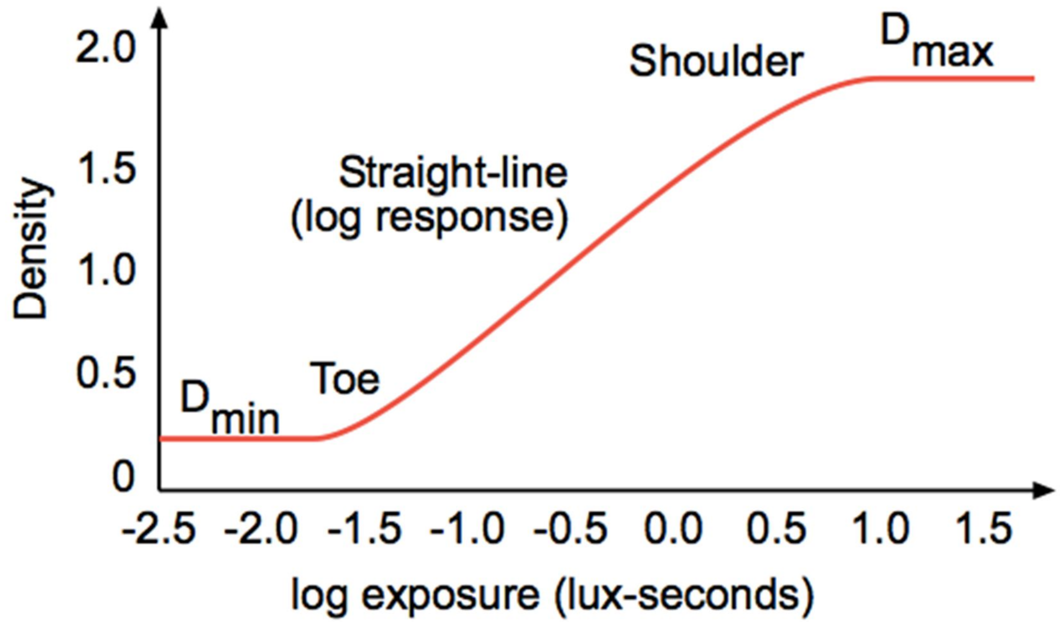
Tone-curve



Global tone-mapping is a compromise between clipping and contrast compression.

Sigmoidal tone-curves

- Very common in digital cameras
  - Mimic the response of analog film
  - Analog film has been engineered over many years to produce good tone-reproduction
- Fast to compute



Sigmoidal tone mapping

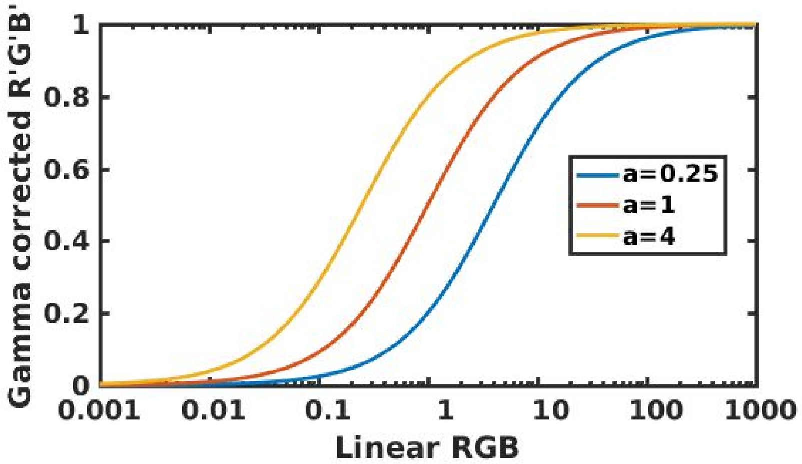
- Simple formula for a sigmoidal tone-curve:

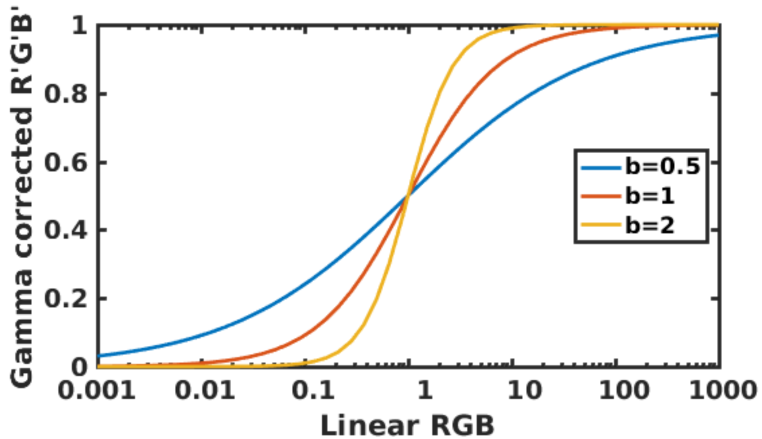
$$R'(x,y) = \frac{R(x,y)^b}{\left(\frac{L_m}{a}\right)^b + R(x,y)^b}$$

where L_m is the geometric mean (or mean of logarithms):

$$L_m = exp\left(\frac{1}{N}\sum_{(x,y)} \ln(L(x,y))\right)$$

and $L(x,y)$ is the luminance of the pixel (x,y).





Sigmoidal tone mapping example

a=0.25

a=1

a=4

b=0.5 b=1 b=2

Histogram equalization

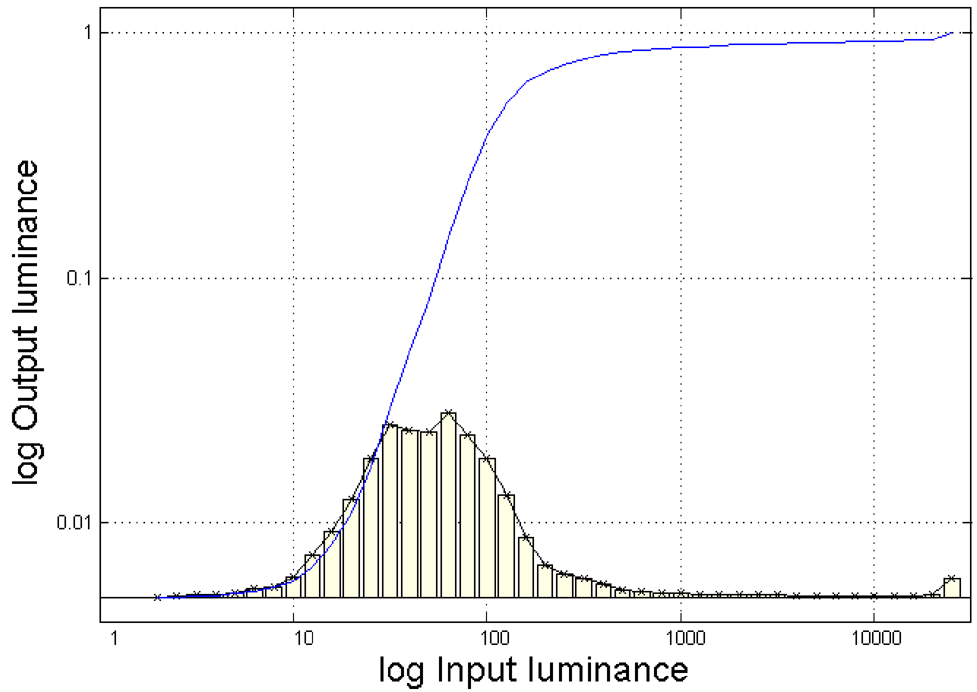
- 1. Compute normalized cummulative image histogram

$$c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$$

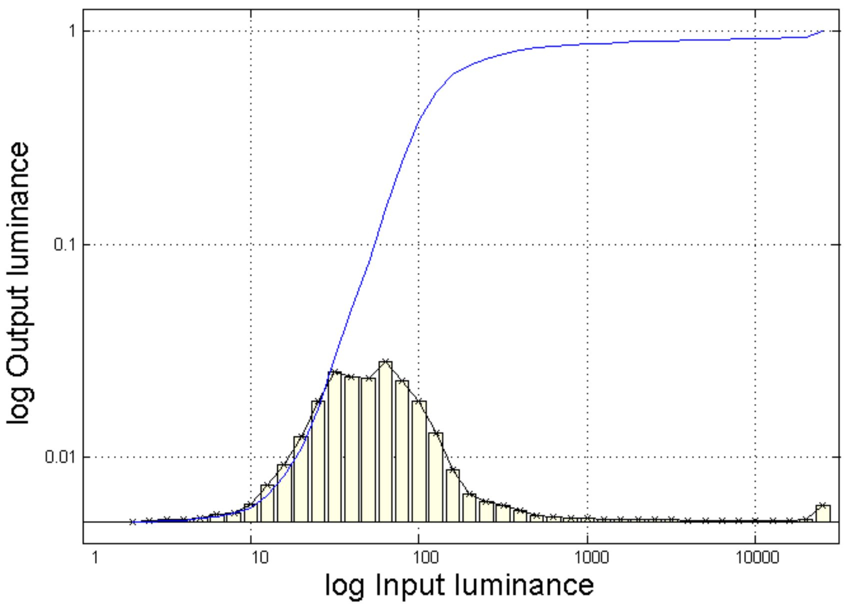
  - For HDR, operate in the log domain
- 2. Use the cummulative histogram as a tone-mapping function

$$Y_{out} = c(Y_{in})$$

  - For HDR, map the log-10 values to the [-dr_{out} ; 0] range
    - where dr_{out} is the target dynamic range (of a display)



Histogram equalization



- Steepest slope for strongly represented bins
  - If many pixels have the same value - enhance contrast
  - Reduce contrast, if few pixels
- Histogram Equalization distributes contrast distortions relative to the "importance" of a brightness level

CLAHE: Contrast-Limited Adaptive Histogram Equalization

- [Pizer et al. Adaptive histogram equalization and its variations. Comput Vision, Graph Image Process 1987], [Larson et al. 1997, IEEE TVCG]

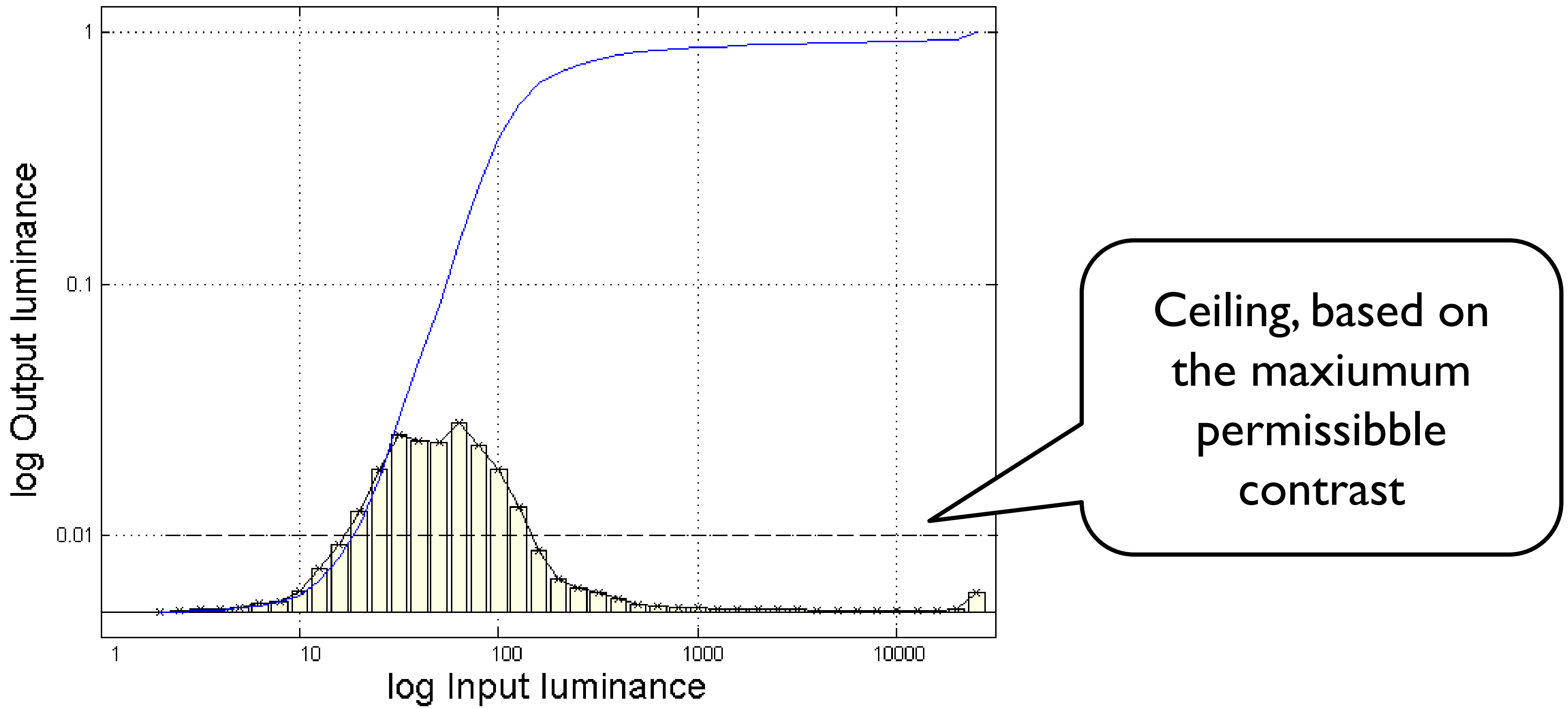
Linear mapping

Histogram equalization

CLAHE

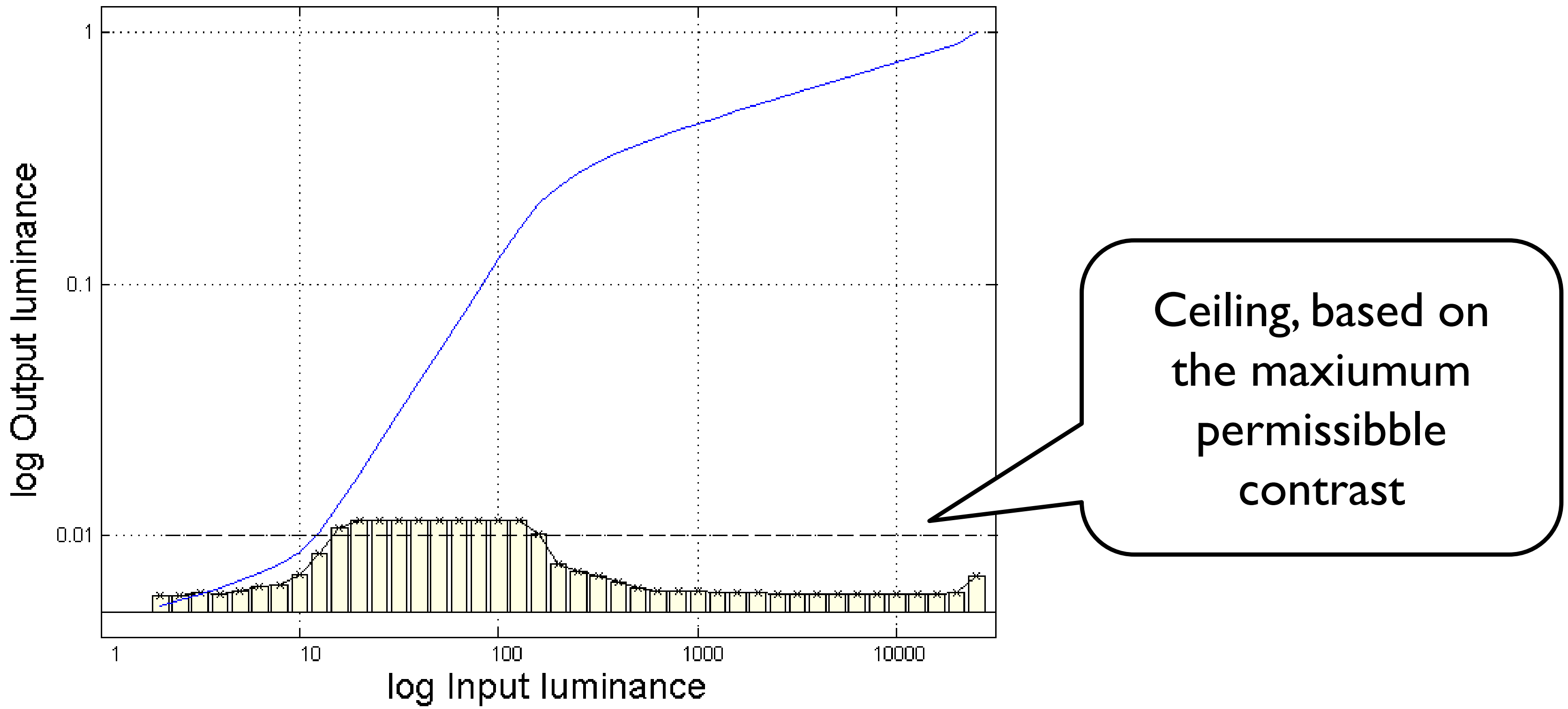
CLAHE: Contrast-Limited Adaptive Histogram Equalization

- Truncate the bins that exceed the ceiling;
- Distribute the removed counts to all bins;
- Repeat until converges



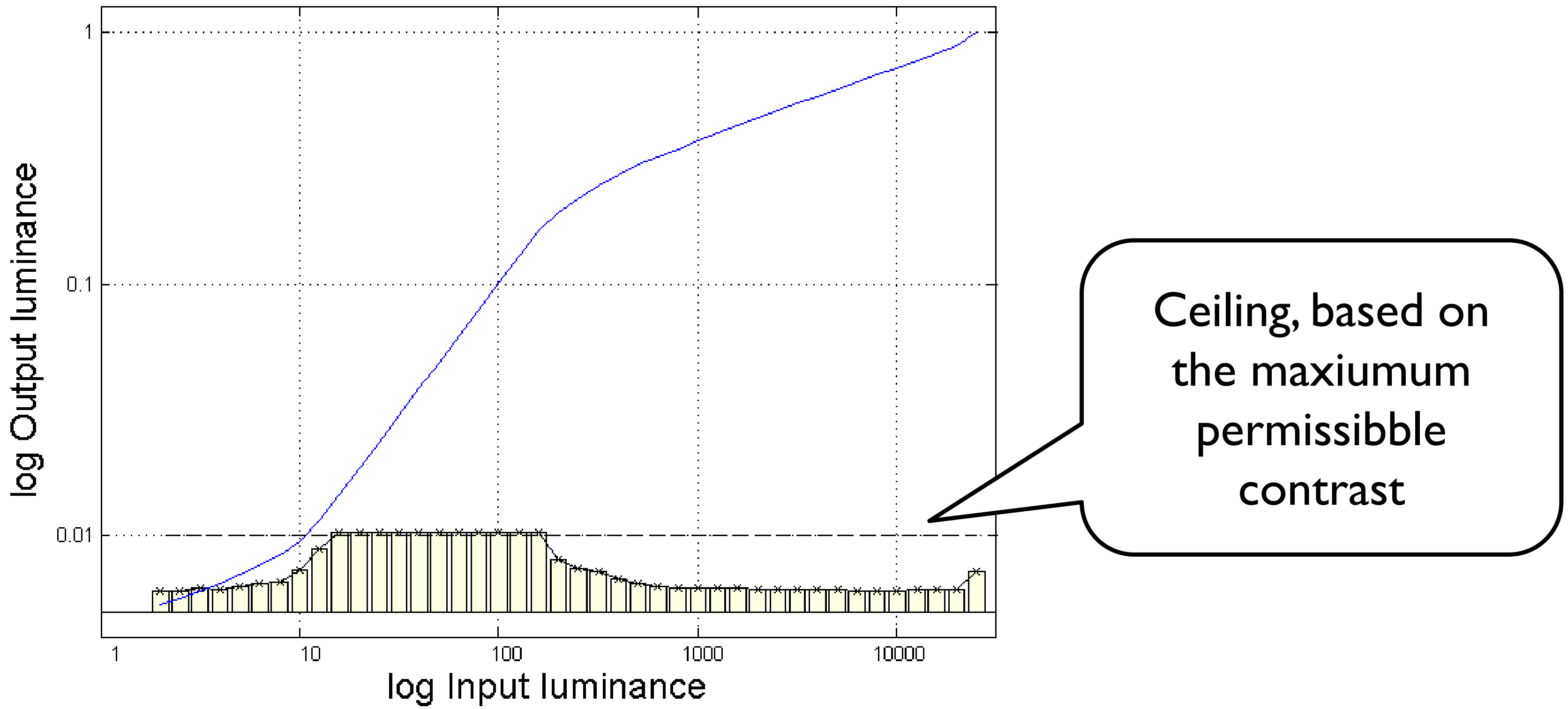
CLAHE: Contrast-Limited Adaptive Histogram Equalization

- Truncate the bins that exceed the ceiling;
- Distribute the removed counts to all bins;
- Repeat until converges



CLAHE: Contrast-Limited Adaptive Histogram Equalization

- Truncate the bins that exceed the ceiling;
- Distribute the removed counts to all bins;
- Repeat until converges



Techniques

- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Colour transfer in tone-mapping

- Many tone-mapping operators work on luminance, mean or maximum colour channel value
  - For speed
  - To avoid colour artefacts
- Colours must be transferred later form the original image
- Colour transfer in the linear RGB colour space:

Output color channel (red)

$$R_{out} = \left(\frac{R_{in}}{L_{in}} \right)^{s} \cdot L_{out}$$

Saturation parameter

Resulting luminance

- The same formula applies to green (G) and blue (B) linear colour values

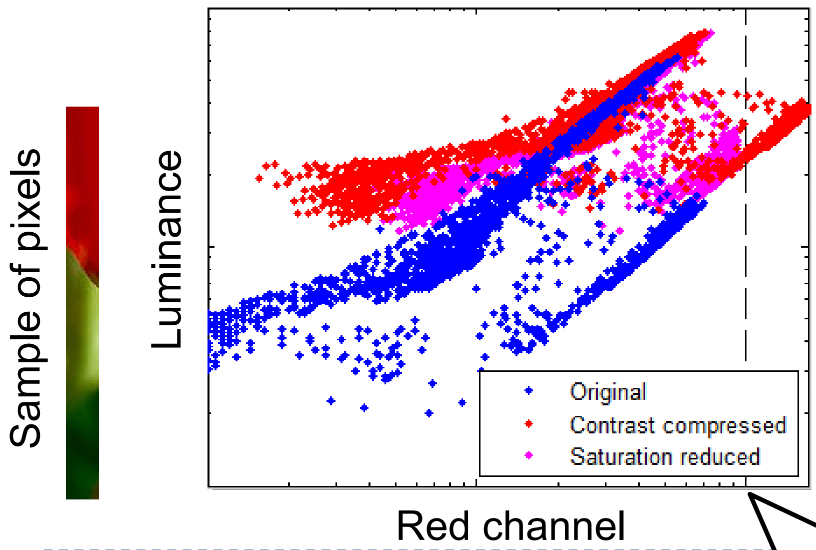
Colour transfer: out-of-gamut problem

- Colours often fall outside the colour gamut when contrast is compressed

Original image

Contrast reduced (s=1)

Colours before/after processing



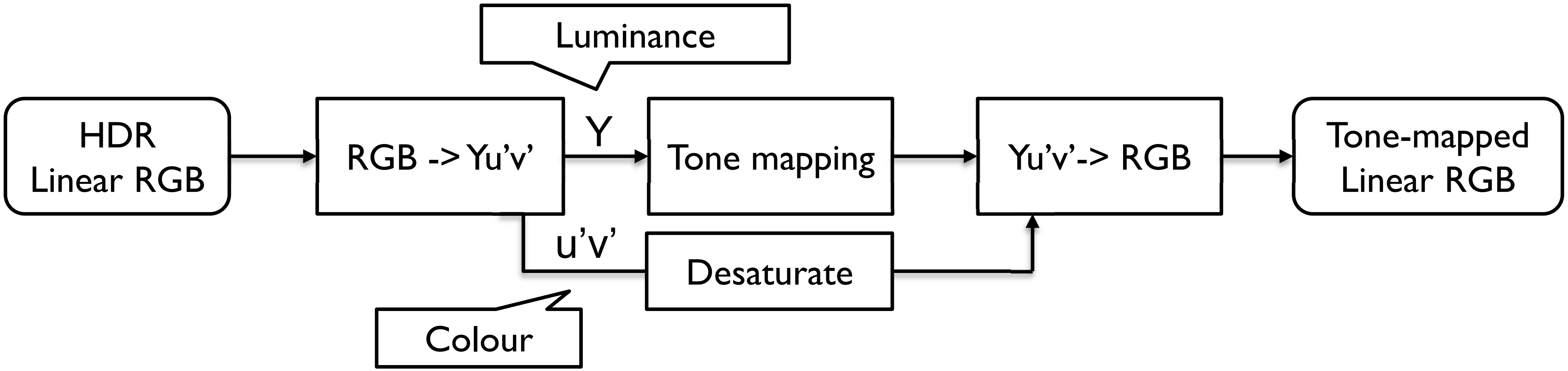
Gamut boundary

- Reduction in saturation is needed to bring the colors into gamut

Saturation reduced (s=0.6)

Colour transfer: alternative method

- Colour transfer in linear RGB will alter resulting luminance
- Colours can be also transferred, and saturation adjusted using CIE u'v' chromatic coordinates



Chroma of the white

- To correct saturation:

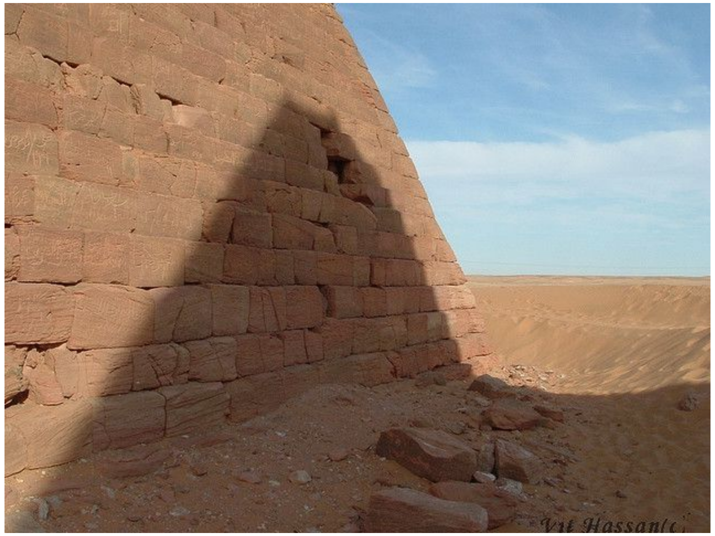
$$u'_{out} = (u'_{in} - u'_{w}) \cdot s + u'_{w} \qquad u'_{w} = 0.1978$$

$$v'_{out} = (v'_{in} - v'_{w}) \cdot s + v'_{w} \qquad v'_{w} = 0.4683$$

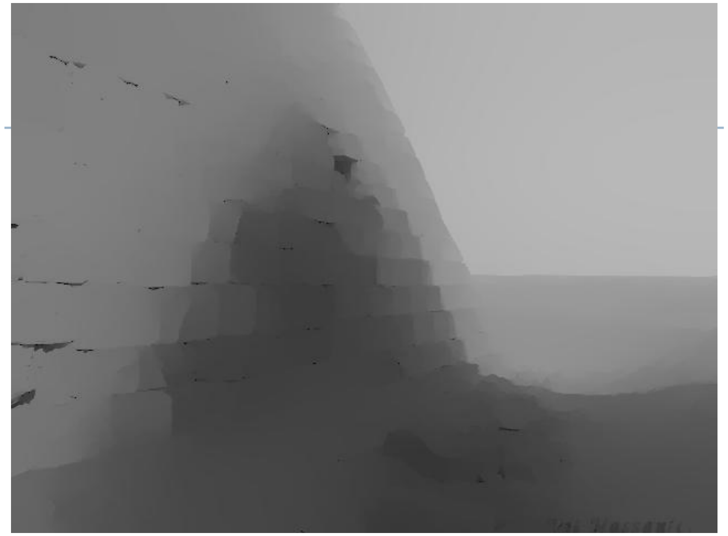
Techniques

- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Illumination & reflectance separation



Input



Illumination

Reflectance

$$Y = I \cdot R$$

Image

Illumination

Reflectance

Illumination and reflectance

Reflectance

- White ≈ 90%
- Black ≈ 3%
- Dynamic range < 100:1
- Reflectance critical for object & shape detection

Illumination

- Sun ≈ 10^{9} cd/m^{2}
- Lowest perceivable luminance ≈ 10^{-6} cd/m^{2}
- Dynamic range 10,000:1 or more
- Visual system partially discounts illumination

Reflectance & Illumination TMO

- Hypothesis: *Distortions in reflectance are more apparent than the distortions in illumination*
- Tone mapping could preserve reflectance but compress illumination

Illumination

Tone-mapped image

$$L_d = R \cdot T(I)$$

Reflectance

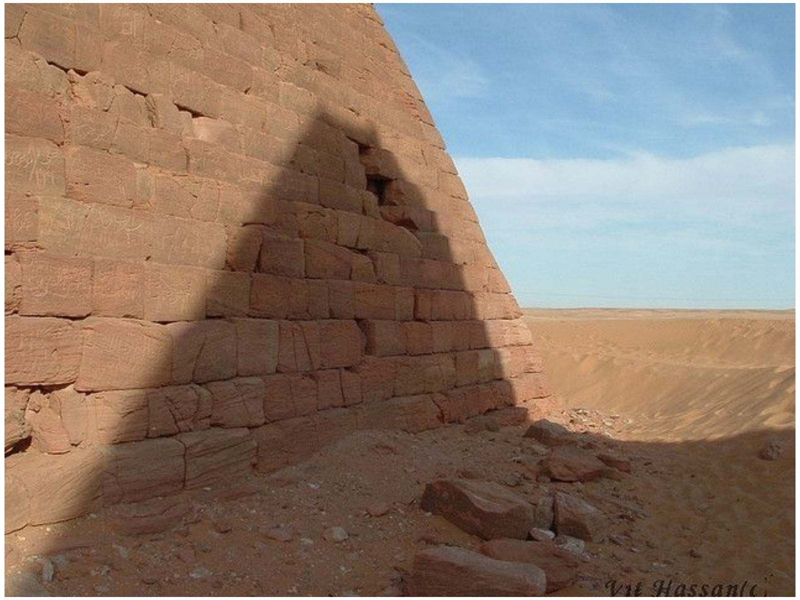
Tone-mapping

- for example:

$$L_d = R \cdot (I / L_{white})^c \cdot L_{white}$$

How to separate the two?

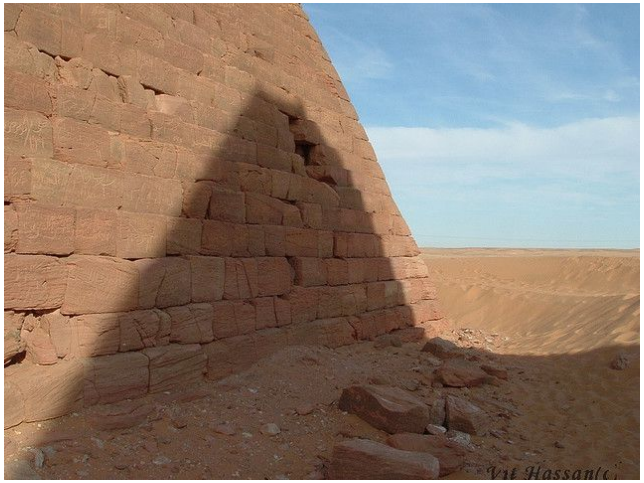
- (Incoming) illumination – slowly changing
  - except very abrupt transitions on shadow boundaries
- Reflectance – low contrast and high frequency variations



Gaussian filter

$$f(x) = \frac{1}{2\pi\sigma_s} e^{\frac{-x^2}{2\sigma_s^2}}$$

- First order approximation



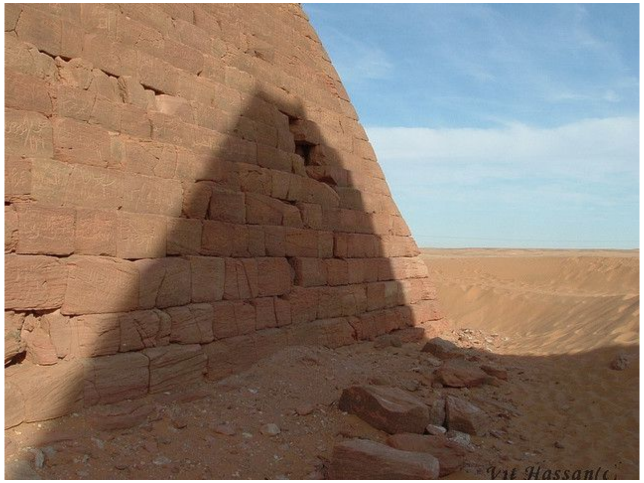
- Blurs sharp boundaries
- Causes halos

Tone mapping result

Bilateral filter

$$I_p \approx \frac{1}{k_s}\sum_{t\in\Omega} f(p-t)\,g(L_p-L_t)\,L_p$$

- Better preserves sharp edges



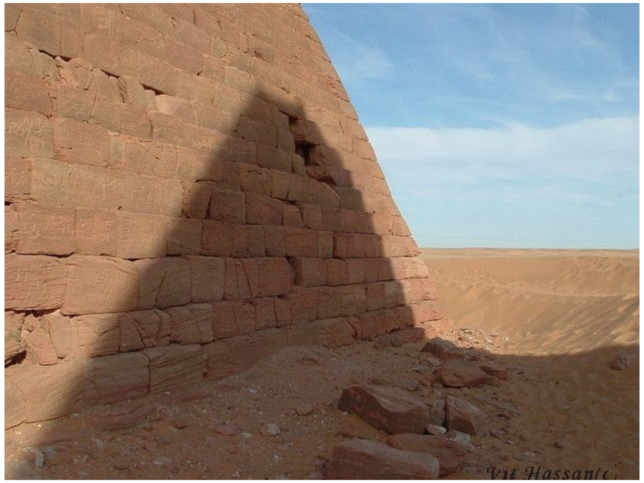
Tone mapping result

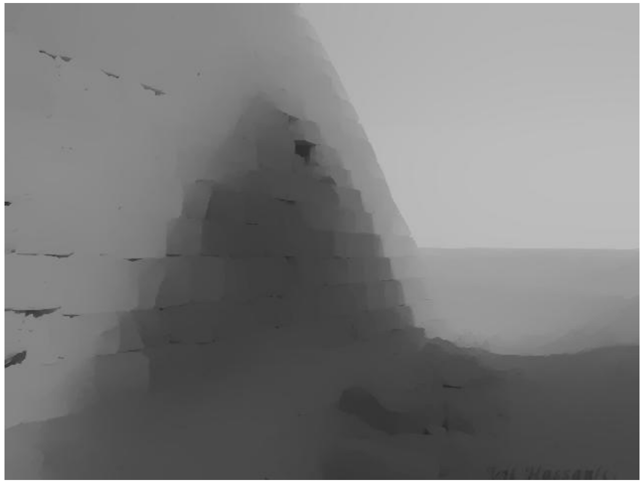
- Still some blurring on the edges
- Reflectance is not perfectly separated from illumination near edges

[Durand & Dorsey, SIGGRAPH 2002]

Weighted-least-squares (WLS) filter

- Stronger smoothing and still distinct edges





- Can produce stronger effects with fewer artifacts
- See „Advanced image processing” lecture

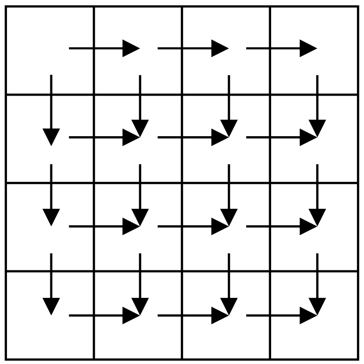
Tone mapping result

[Farbman et al., SIGGRAPH 2008]

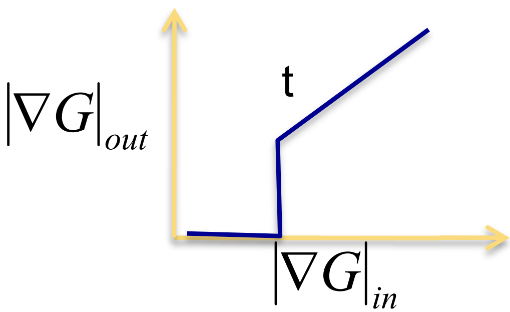
Retinex

- Retinex algorithm was initially intended to separate reflectance from illumination [Land 1964]
  - There are many variations of Retinex, but the general principle is to eliminate small gradients from an image. Small gradients are attributed to the illumination

1 step: compute gradients in log domain



2nd step: set to 0 gradients less than the threshold



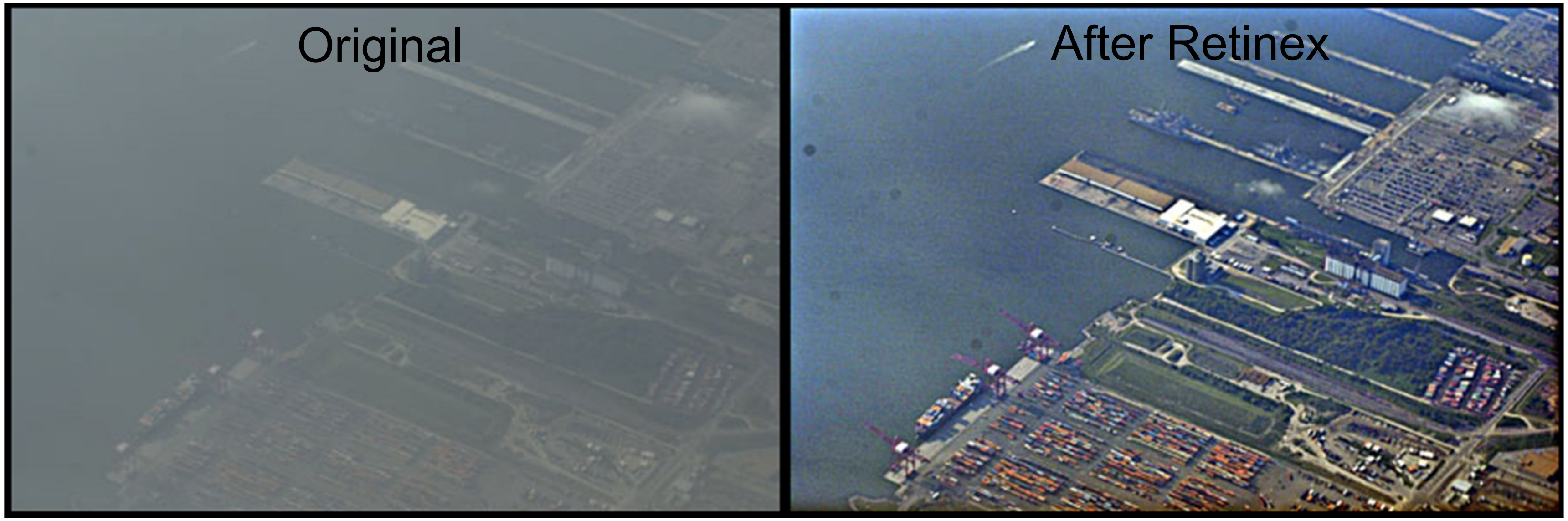
3rd step: reconstruct an image from the vector field

$$\nabla^2 I = \operatorname{div} G$$

For example by solving the Poisson equation

Retinex examples

From: http://dragon.larc.nasa.gov/retinex/757/

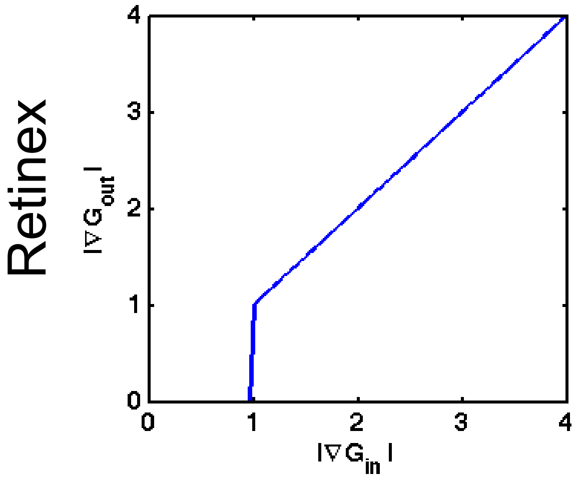


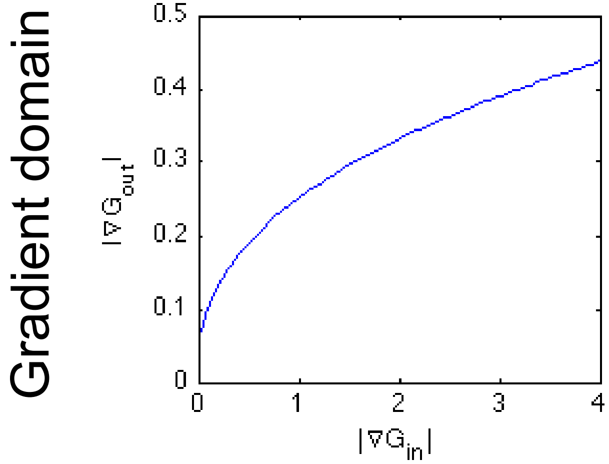
From:http://www.ipol.im/pub/algo/lmps_retinex_poisson_equation/#ref_1

Gradient domain HDR compression

[Fattal et al., SIGGRAPH 2002]

- Similarly to Retinex, it operates on log-gradients
- But the function amplifies small contrast instead of removing it





- Contrast compression achieved by global contrast reduction
  - Enhance reflectance, then compress everything

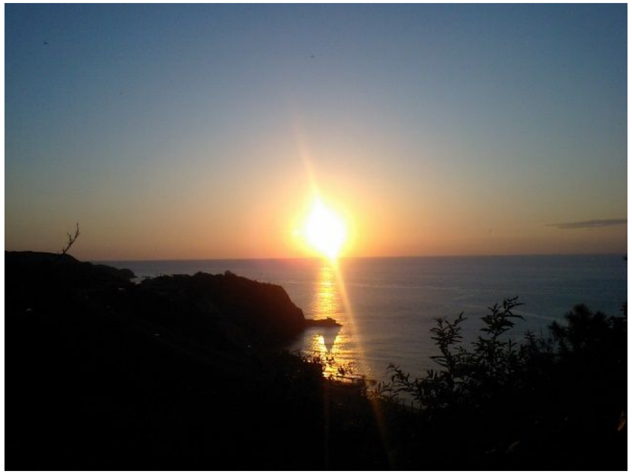
Techniques

- Arithmetic of HDR images
- Display model
- Tone-curve
- Colour transfer
- Base-detail separation
- Glare

Glare

"Alan Wake" © Remedy Entertainment

Glare Illusion

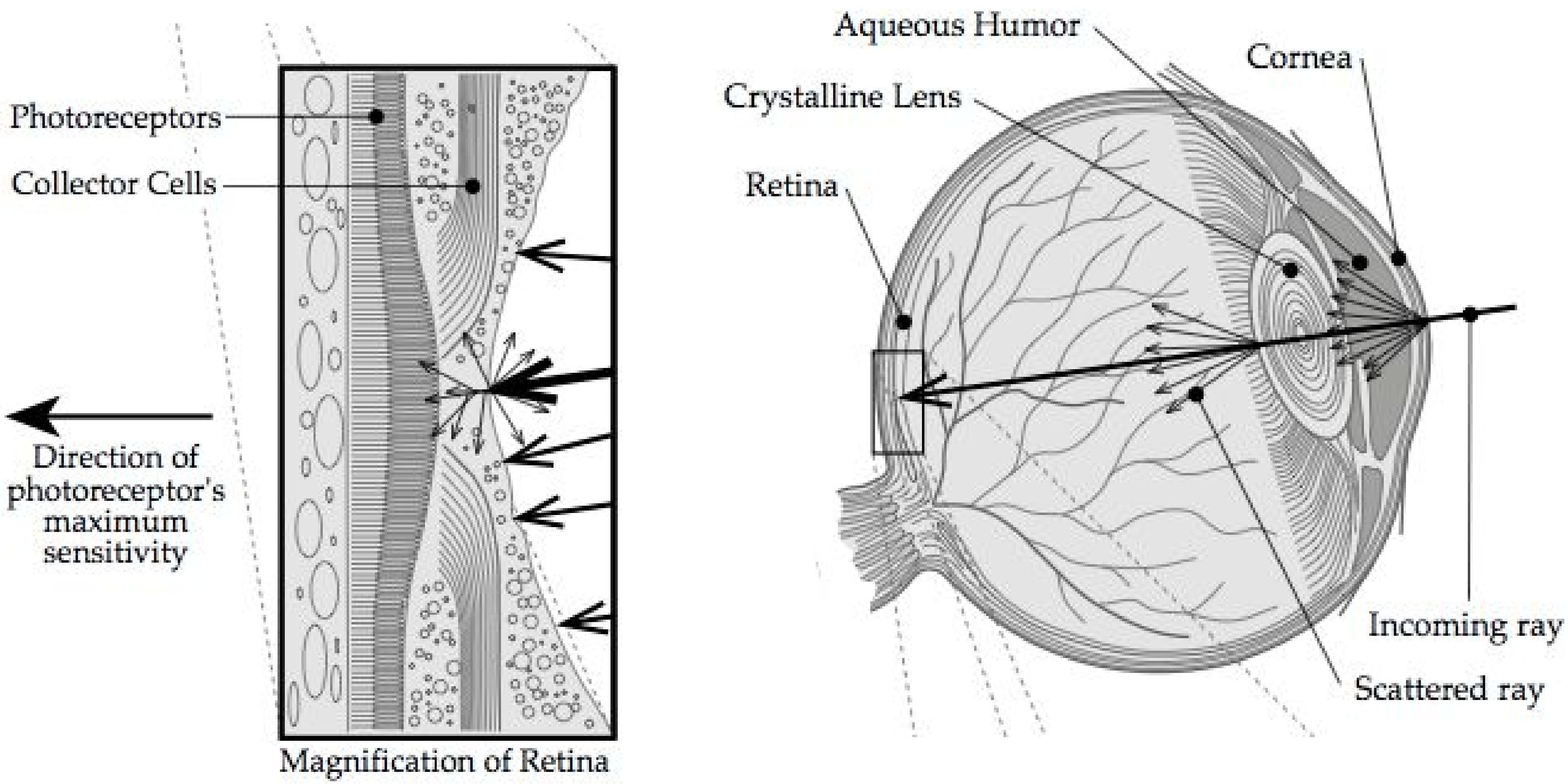


Photography

Painting

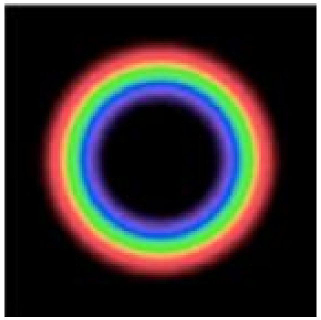
Computer Graphics
HDR rendering in games

Scattering of the light in the eye

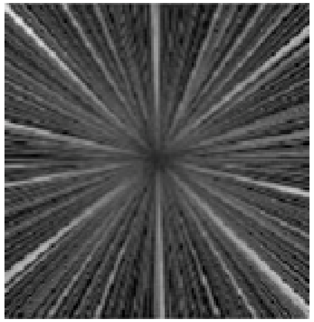


From: Sekuler, R., and Blake, R. Perception, second ed. McGraw- Hill, New York, 1990

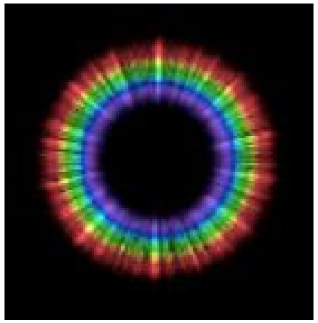
Ciliary corona and lenticular halo



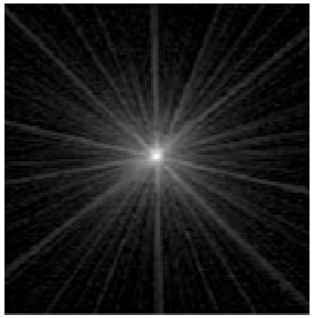
*



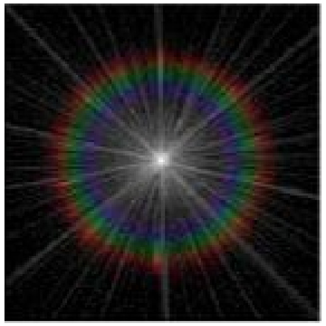
=

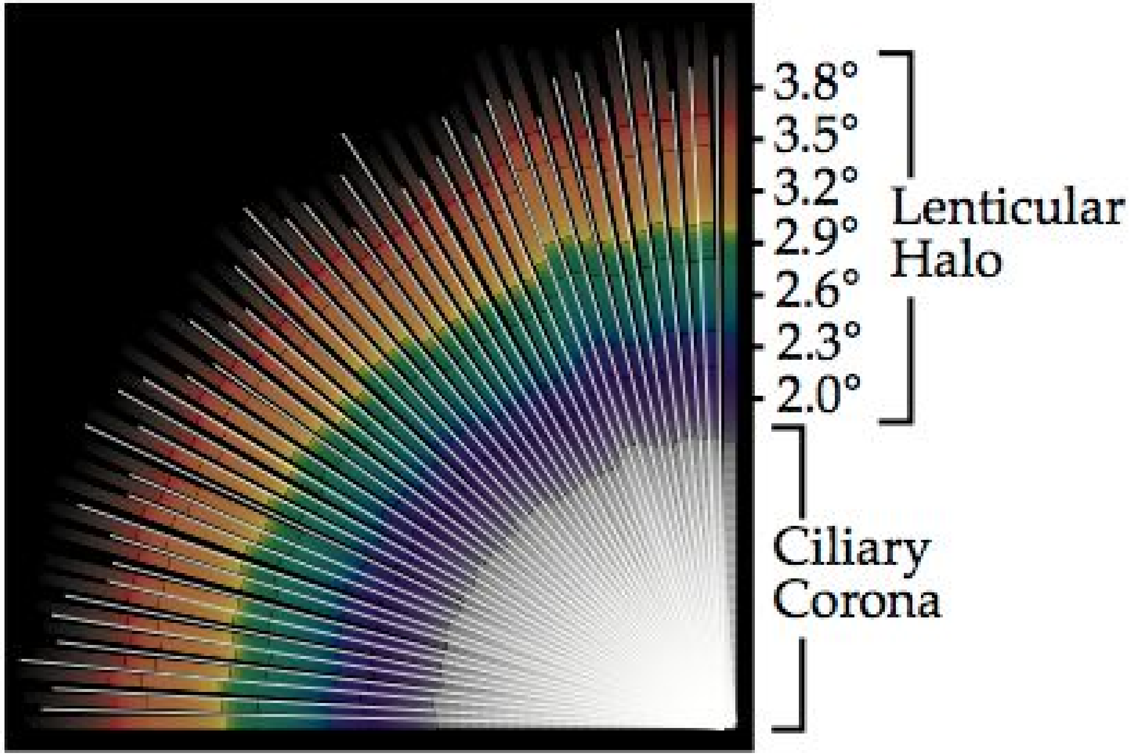


+



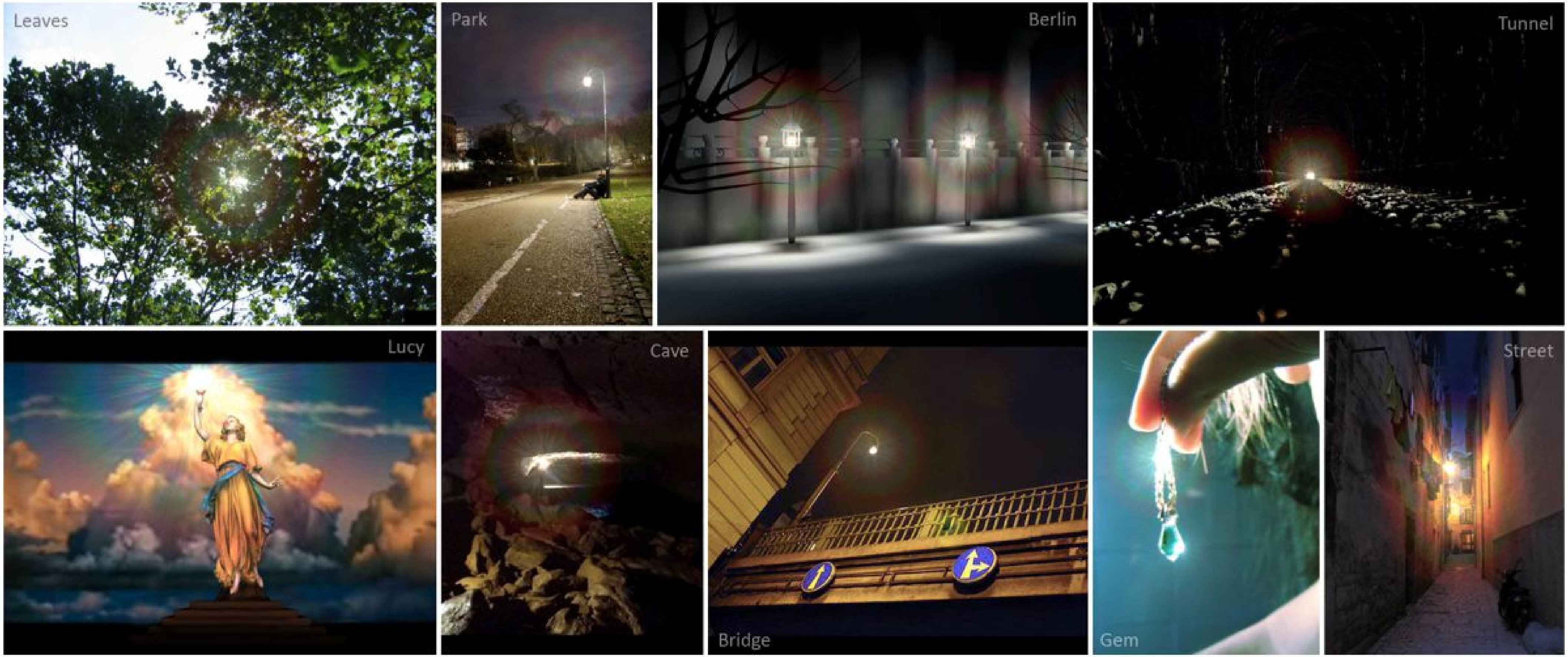
=





From: Spencer, G. et al. 1995. Proc. of SIGGRAPH. (1995)

Examples of simulated glare

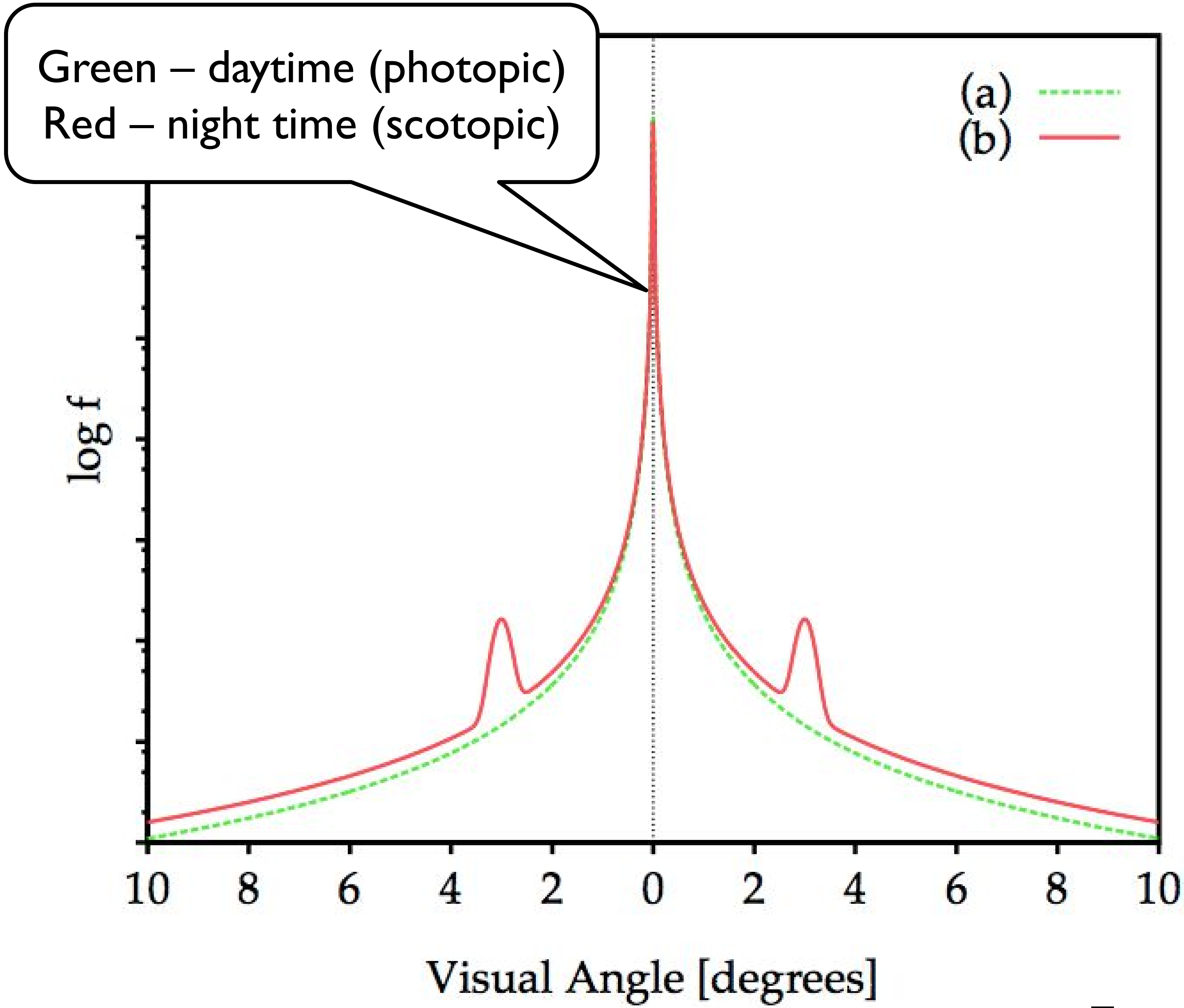


[From Ritschel et al, Eurographics 2009]

Temporal glare

[From Ritschel et al, Eurographics 2009]

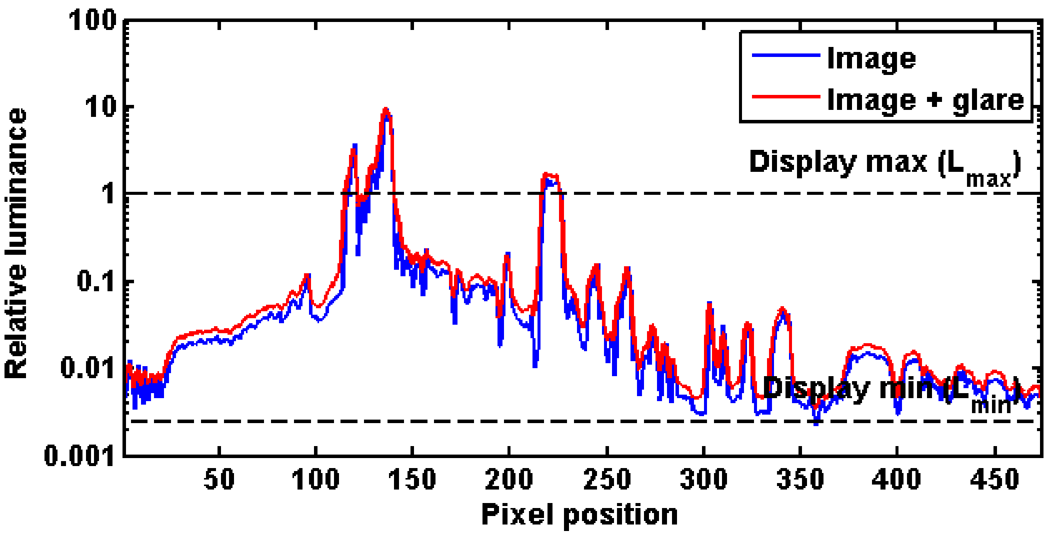
Point Spread Function of the eye



- What portion of the light is scattered towards a certain visual angle
- To simulate:
  - construct a digital filter
  - convolve the image with that filter

From: Spencer, G. et al. 1995. Proc. of SIGGRAPH. (1995)

Selective application of glare

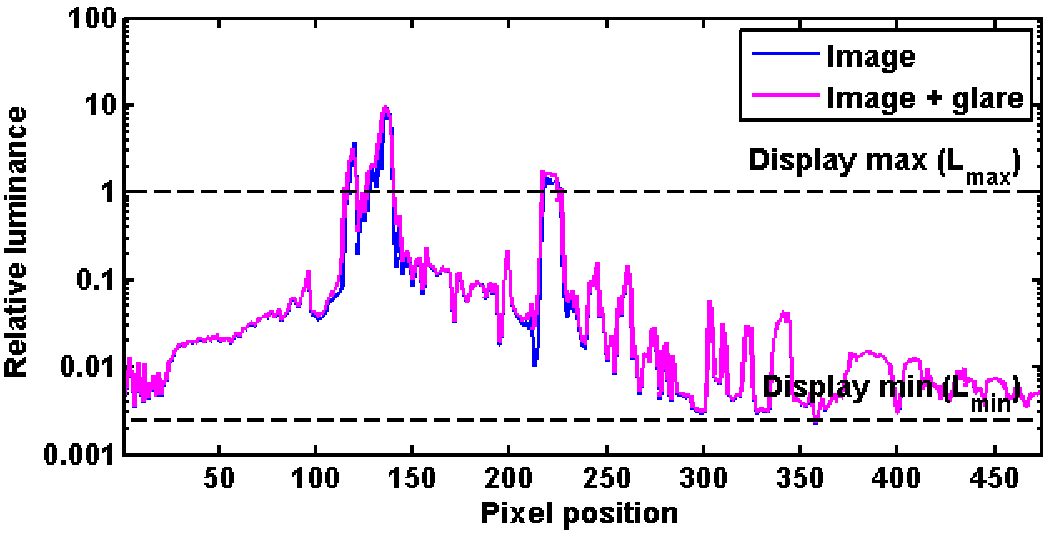


- A) Glare applied to the entire image

$$I_g = I * G$$

Glare kernel (PSF)

- Reduces image contrast and sharpness



B) Glare applied only to the clipped pixels

$$I_g = I + I_{cliped} * G - I_{cliped}$$

where $I_{cliped} = \begin{cases} I & for\ I > 1 \\ 0 & otherwise \end{cases}$

Better image quality

Selective application of glare

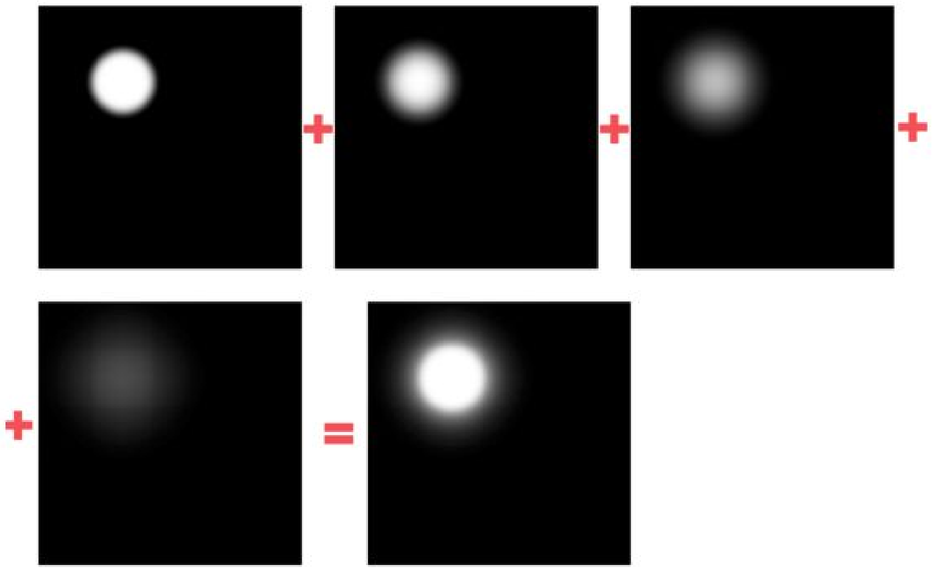
A) Glare applied to
the entire image

Original image

B) Glare applied to
clipped pixels only

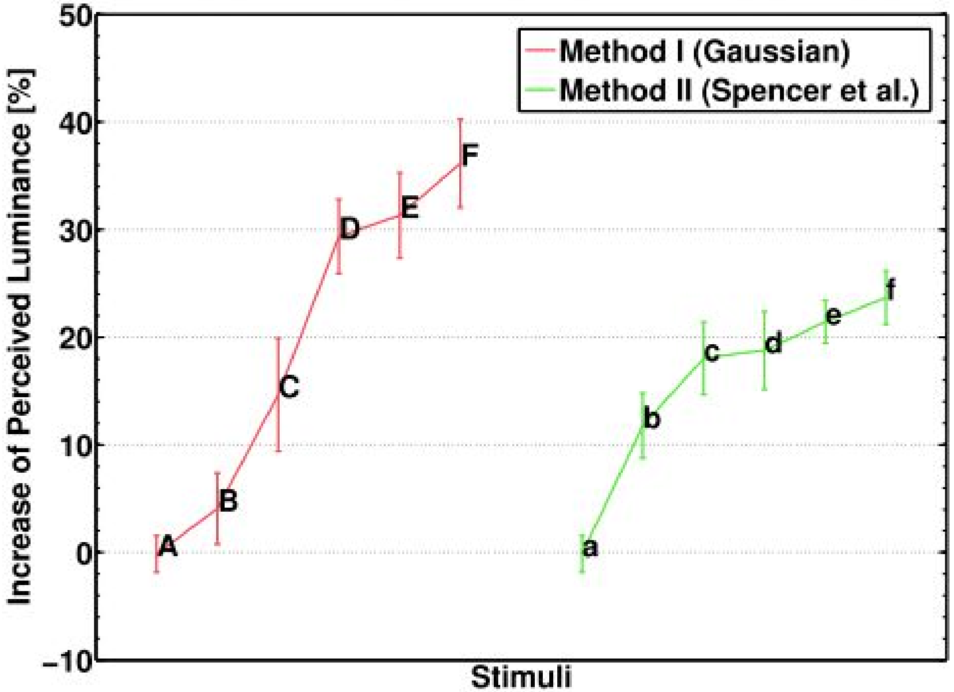
Glare (or bloom) in games

- Convolution with large, non-separable filters is too slow
- The effect is approximated by a combination of Gaussian filters
  - Each filter with different "sigma"
- The effect is meant to look good, not be be accurate model of light scattering
- Some games simulate camera rather than the eye



Does the exact shape of the PSF matter?

- The illusion of increased brightness works even if the PSF is very different from the PSF of the eye



red - Gaussian

green - accurate

[Yoshida et al., APGV 2008]

HDR rendering – motion blur

From LDR pixels

From HDR pixels

References

- Comprehensive book on HDR Imaging
  - E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski, High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting, 2nd editio. Morgan Kaufmann, 2010.
- Overview of HDR imaging & tone-mapping
  - http://www.cl.cam.ac.uk/~rkm38/hdri_book.html
- Review of recent video tone-mapping
  - A comparative review of tone-mapping algorithms for high dynamic range video *Gabriel Eilertsen, Rafal K. Mantiuk, Jonas Unger*, Eurographics State-of-The-Art Report 2017.
- Selected papers on tone-mapping:
  - G. W. Larson, H. Rushmeier, and C. Piatko, "A visibility matching tone reproduction operator for high dynamic range scenes," *IEEE Trans. Vis. Comput. Graph.*, vol. 3, no. 4, pp. 291–306, 1997.
  - R. Wanat and R. K. Mantiuk, "Simulating and compensating changes in appearance between day and night vision," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 33, no. 4, p. 147, 2014.
  - Spencer, G. et al. 1995. Physically-Based Glare Effects for Digital Images. Proceedings of SIGGRAPH. (1995), 325–334
  - Ritschel, T. et al. 2009. Temporal Glare: Real-Time Dynamic Simulation of the Scattering in the Human Eye. Computer Graphics Forum. 28, 2 (Apr. 2009), 183–192
  - …

Advanced Graphics and Image Processing

Models of early visual perception

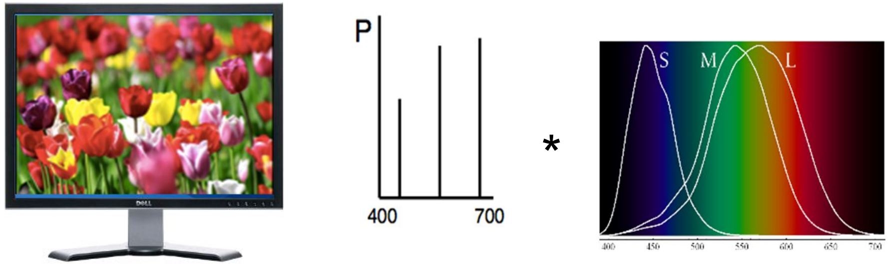
Part 1/6 – perceived brightness of light

Rafal Mantiuk

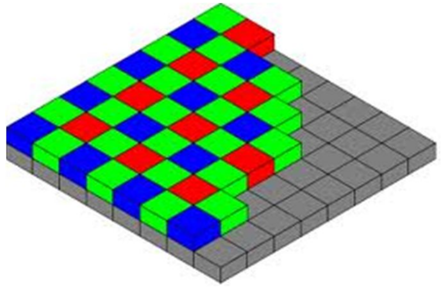
Computer Laboratory, University of Cambridge

Many graphics/display solutions are motivated by visual perception

Image & video compression

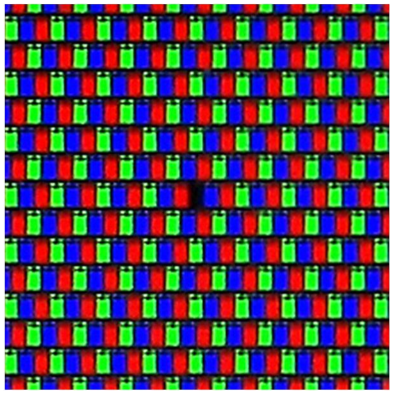


Display spectral emission - metamerism

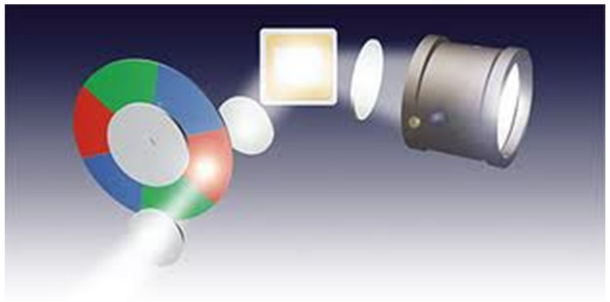


Camera's Bayer pattern

Halftonning



Display's subpixels



Color wheel in DLPs

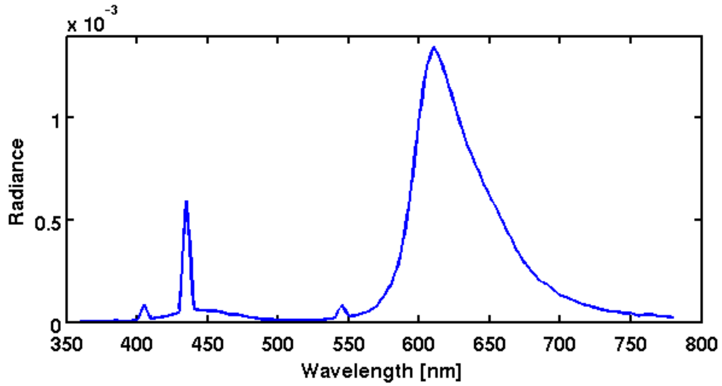
Luminance (again)

- Luminance – measure of light weighted by the response of the achromatic mechanism. Units: cd/m^2

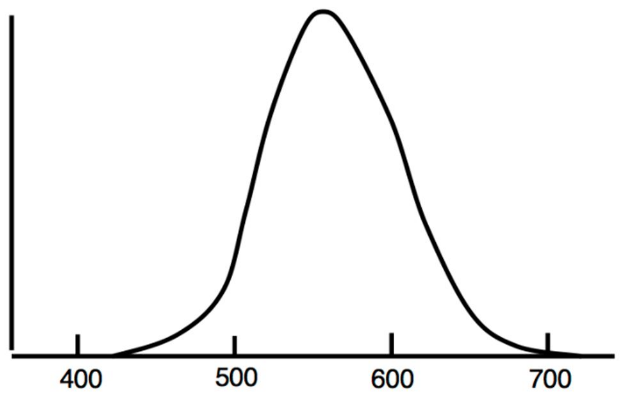
Luminance

$$L_V = \int_{350}^{700} kL(\lambda)V(\lambda)d\lambda \qquad k = 683.002$$

Light spectrum (radiance)



Luminous efficiency function (weighting)



Steven's power law for brightness

- Stevens (1906-1973) measured the perceived magnitude of physical stimuli
  - Loudness of sound, tastes, smell, warmth, electric shock and brightness
  - Using the magnitude estimation methods
    - Ask to rate loudness on a scale with a known reference
- All measured stimuli followed the power law:

Perceived magnitude

$$\varphi(I) = kI^{a}$$

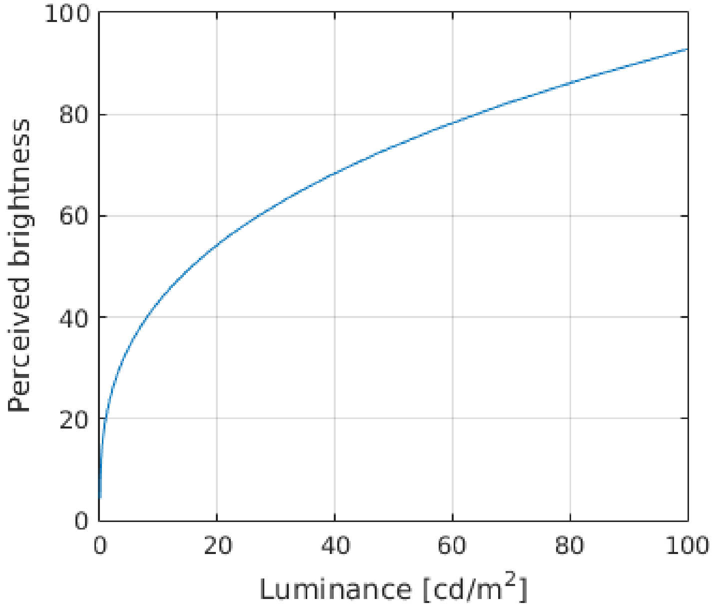
Exponent

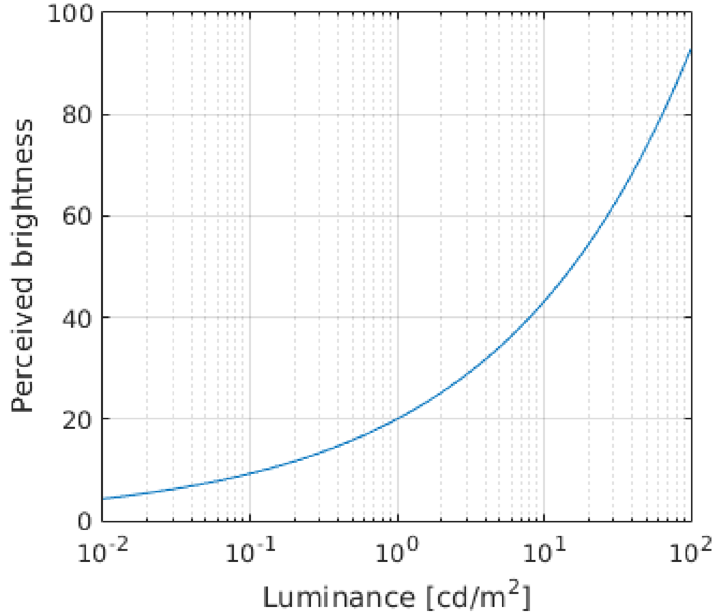
Constant

Physical stimulus

- For brightness (5 deg target in dark), a = 0.3

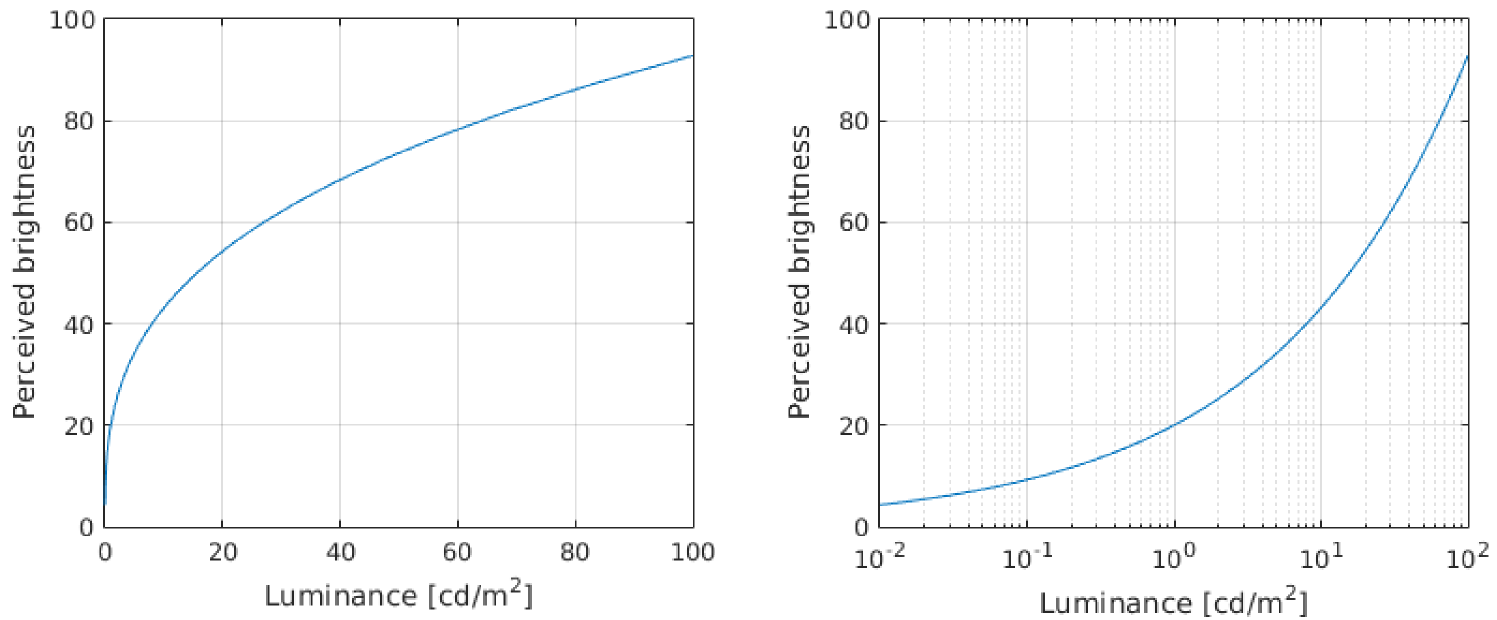
Steven's law for brightness



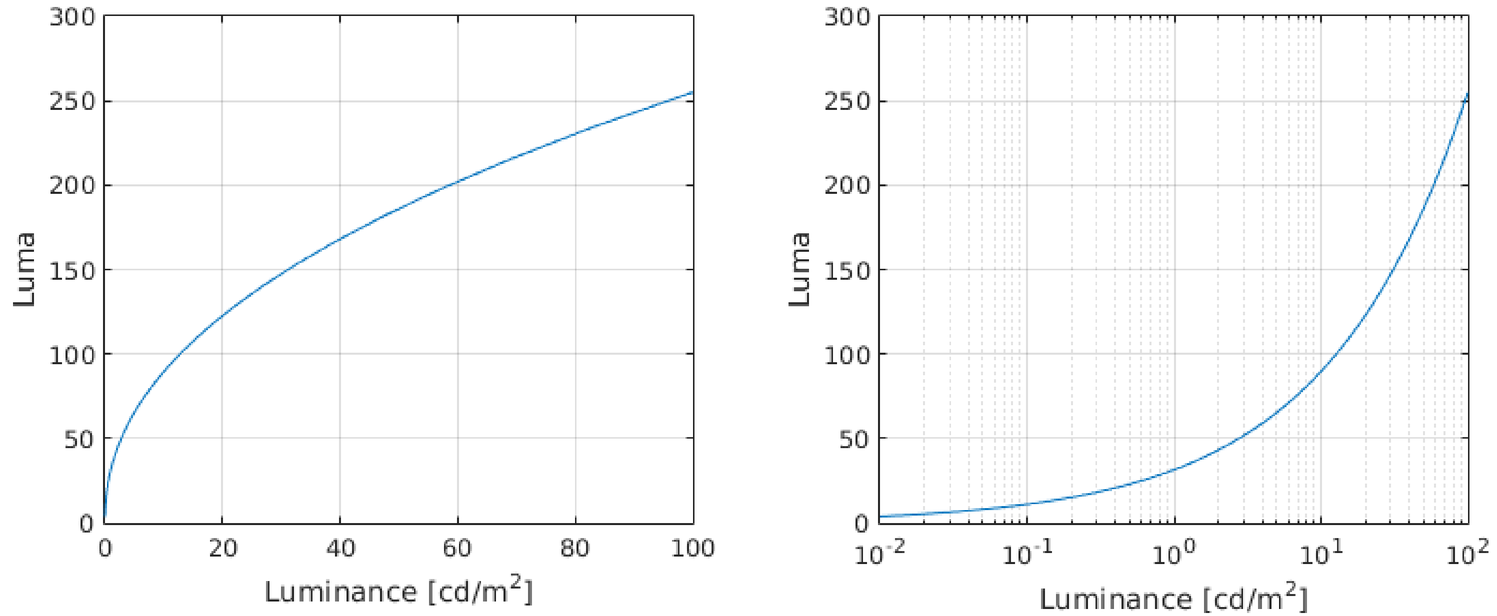


Steven's law vs. Gamma correction

Stevens' law
a=0.3



Gamma function
Gamma = 2.2



UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

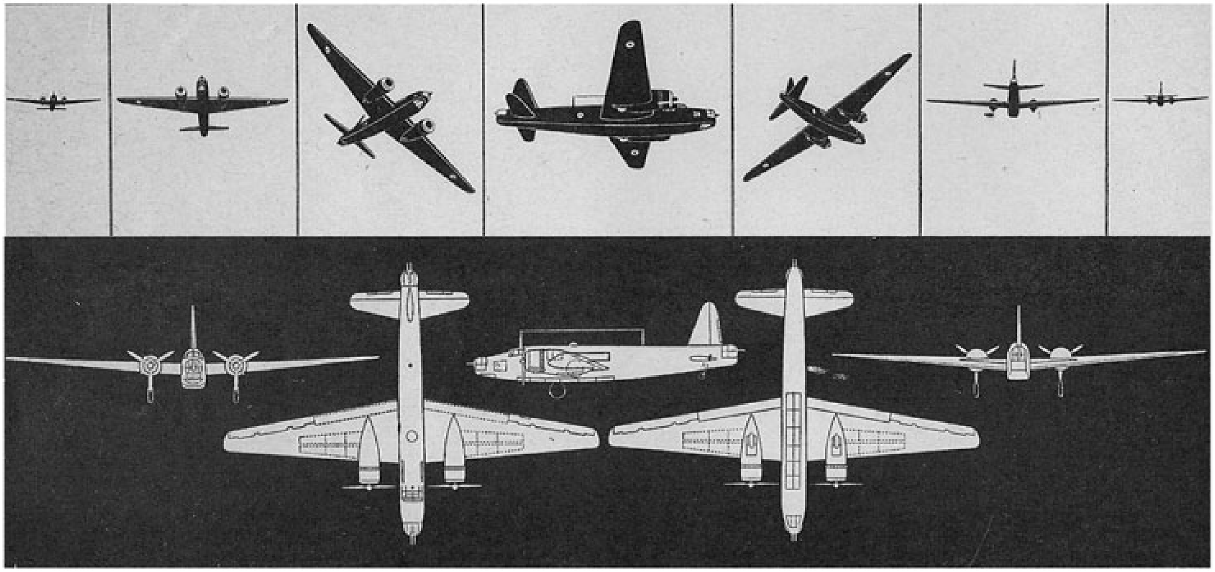
Models of early visual perception

Part 2/6 – contrast detection

Rafal Mantiuk

Computer Laboratory, University of Cambridge

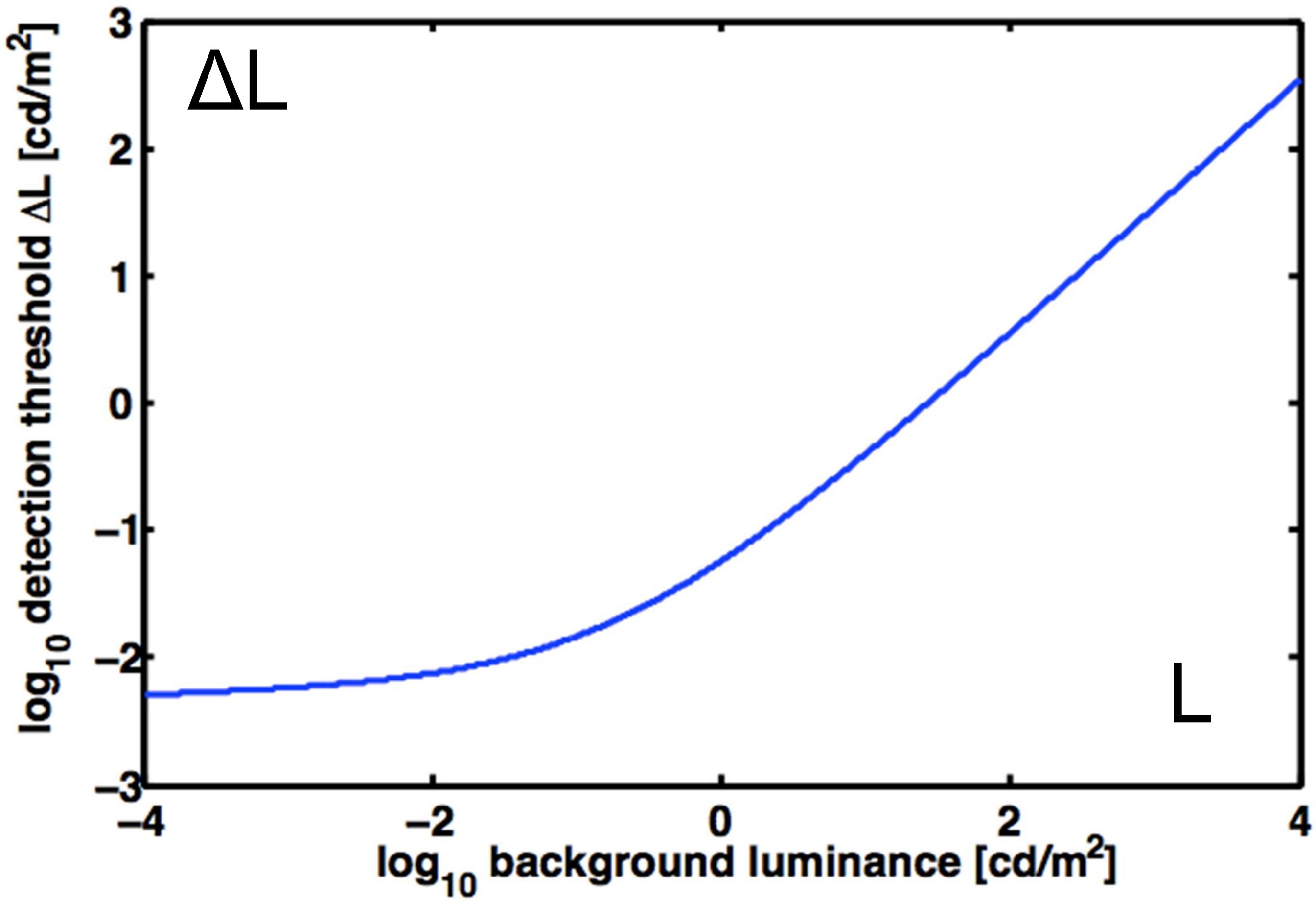
Detection thresholds

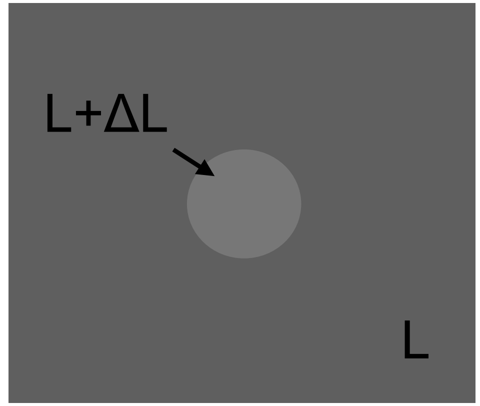


- The smallest detectable difference between
  - the luminance of the object and
  - the luminance of the background

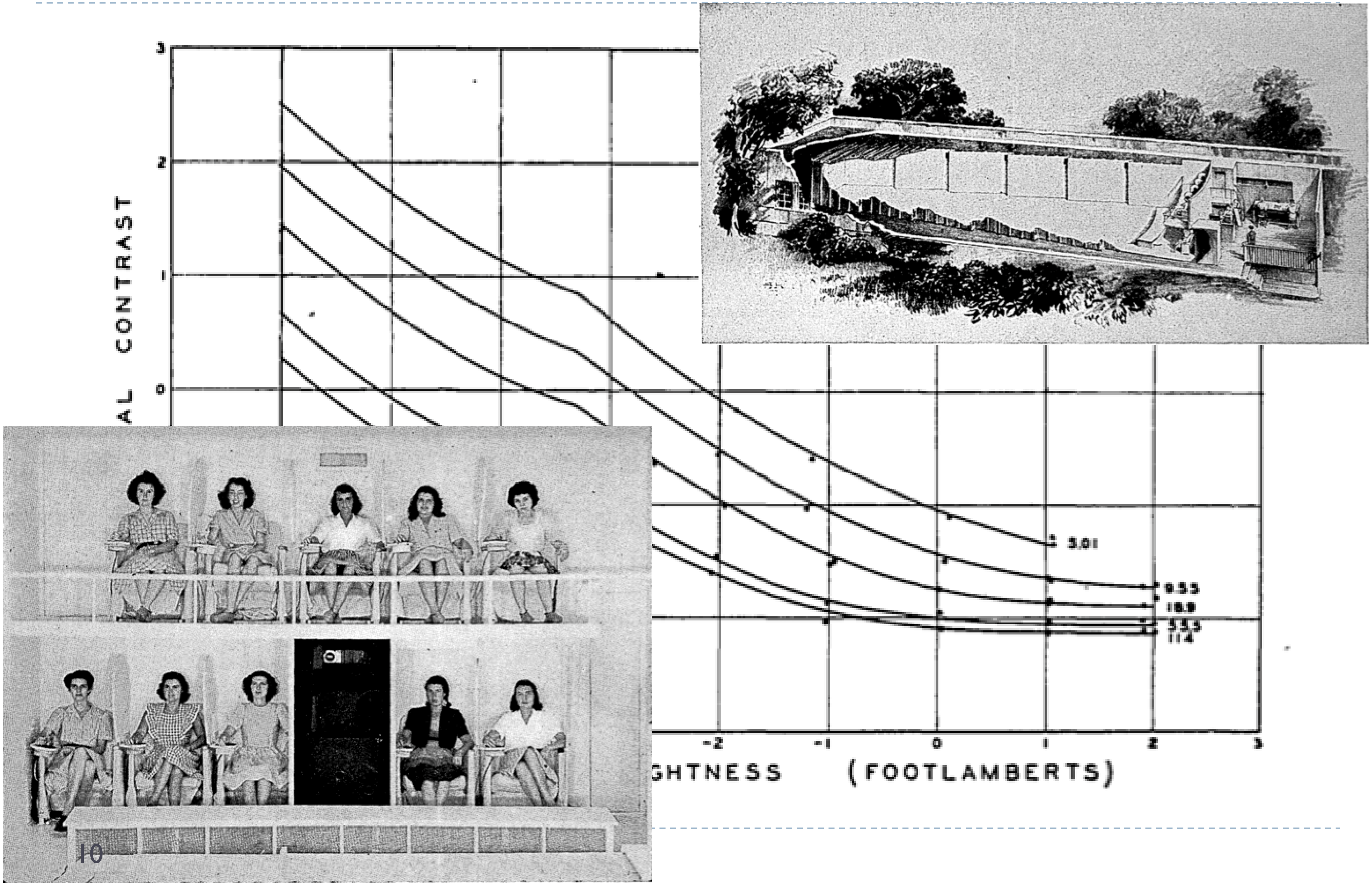
Threshold versus intensity (t.v.i.) function

- The smallest detectable difference in luminance for a given background luminance

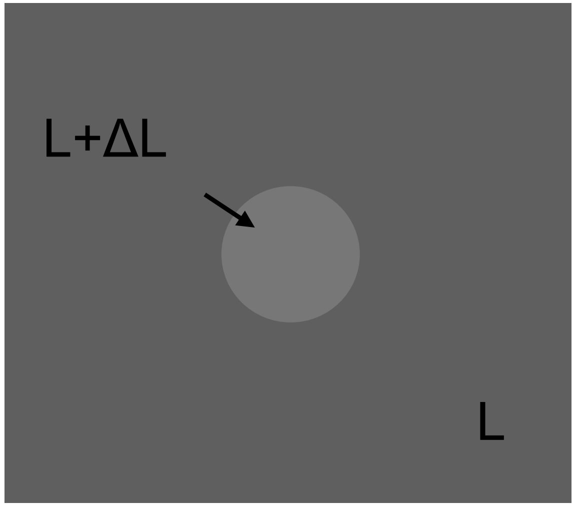


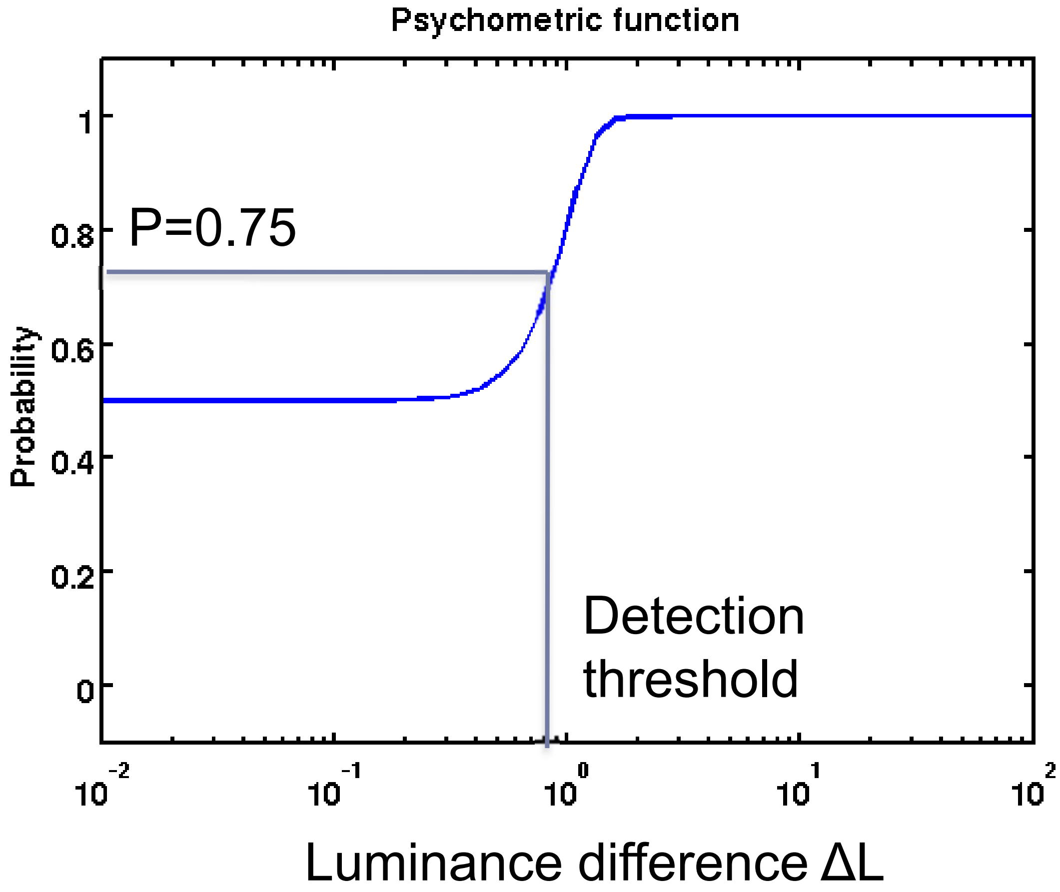


t.v.i. measurements – Blackwell 1946



Psychophysics
Threshold experiments

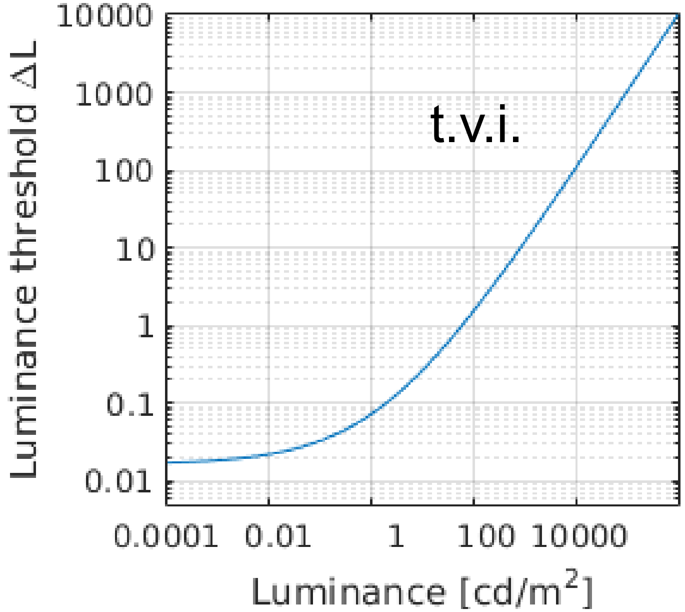




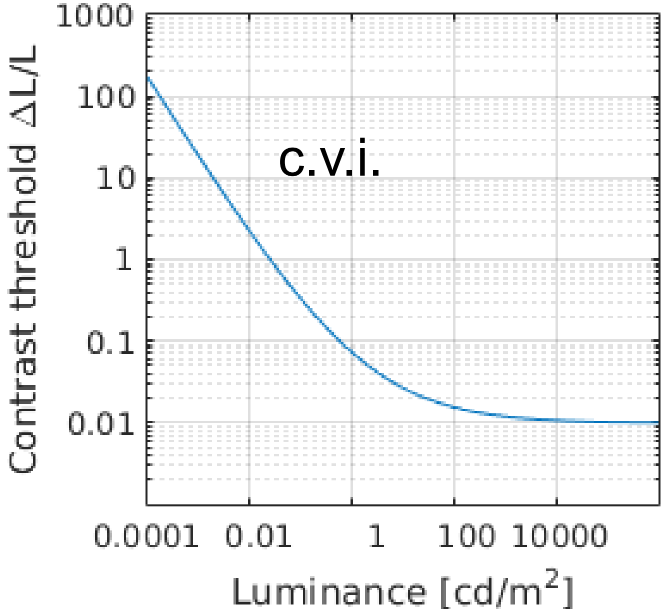
t.v.i function / c.v.i. function / Sensitivity

- The same data, different representation

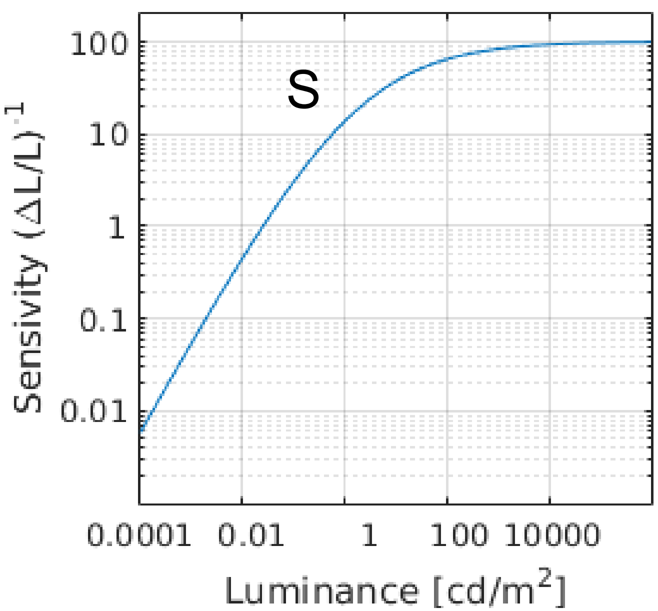
Threshold vs. intensity



Contrast vs. intensity



Sensitivity



$$\Delta L = L_{disk} - L_{background}$$

$$C = \frac{\Delta L}{L_{background}}$$

$$S = \frac{1}{C} = \frac{L_{background}}{\Delta L}$$

Sensitivity to luminance

- Weber-law – the just-noticeable difference is proportional to the magnitude of a stimulus

Ernst Heinrich Weber
[From wikipedia]

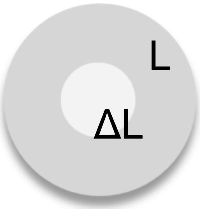
$$\frac{\Delta L}{L} = k$$

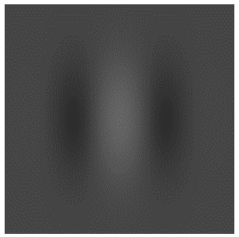
The smallest detectable luminance difference → ΔL

Background (adapting) luminance → L

Constant → k

Typical stimuli:





Consequence of the Weber-law

- Smallest detectable difference in luminance

$$\frac{\Delta L}{L} = k$$

For k=1%

L	ΔL
100 cd/m^2	1 cd/m^2
1 cd/m^2	0.01 cd/m^2

- Adding or subtracting luminance will have different visual impact depending on the background luminance
- Unlike LDR luma values, luminance values are **not** perceptually uniform!

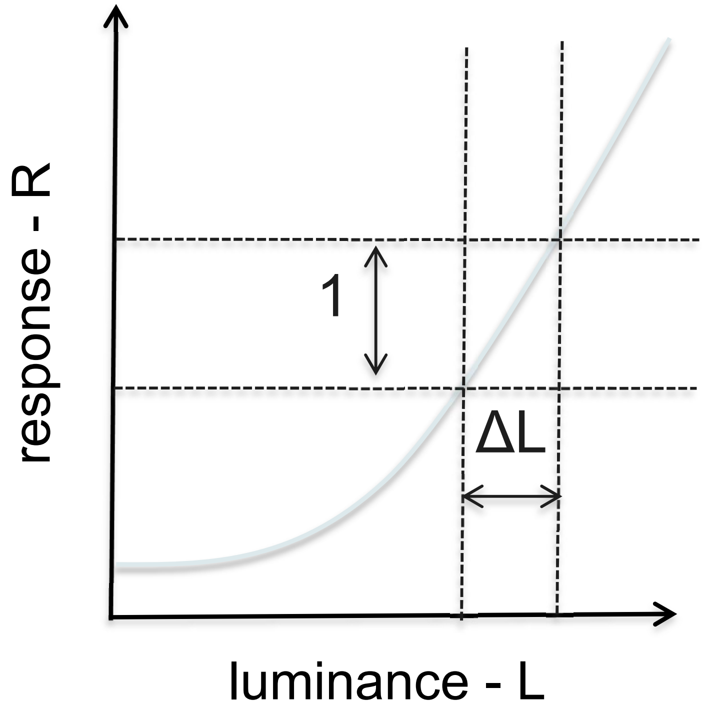
How to make luminance (more) perceptually uniform?

- Using "Fechnerian" integration

$$\frac{dR}{dl}(L) = \frac{1}{\Delta L(L)}$$

Derivative of response

Detection threshold



Luminance transducer: $R(L) = \int_{L_{min}}^{L} \frac{1}{\Delta L(l)} dl$

Assuming the Weber law

$$\frac{\Delta L}{L} = k$$

- and given the luminance transducer

$$R(L) = \int \frac{1}{\Delta L(l)} dl$$

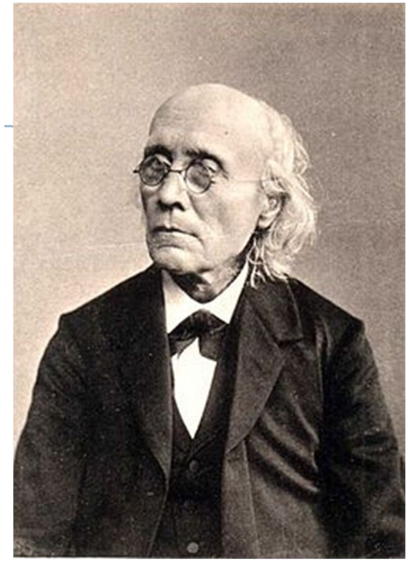
- the response of the visual system to light is:

$$R(L) = \int \frac{1}{kL} dL = \frac{1}{k} \ln(L) + k_1$$

Fechner law

$$R(L) = a \ln(L)$$

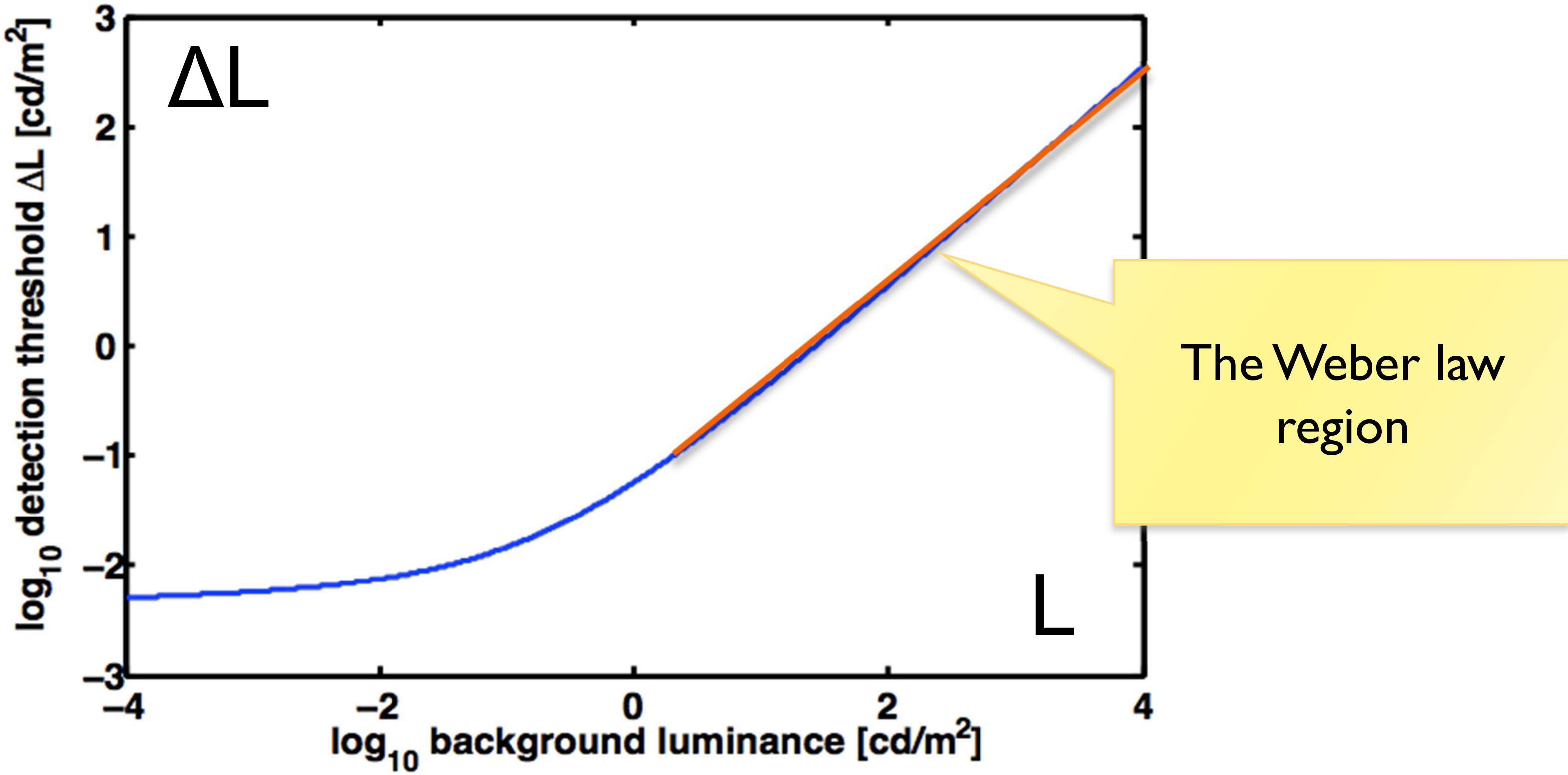
- Response of the visual system to luminance is **approximately** logarithmic



Gustav Fechner
[From Wikipedia]

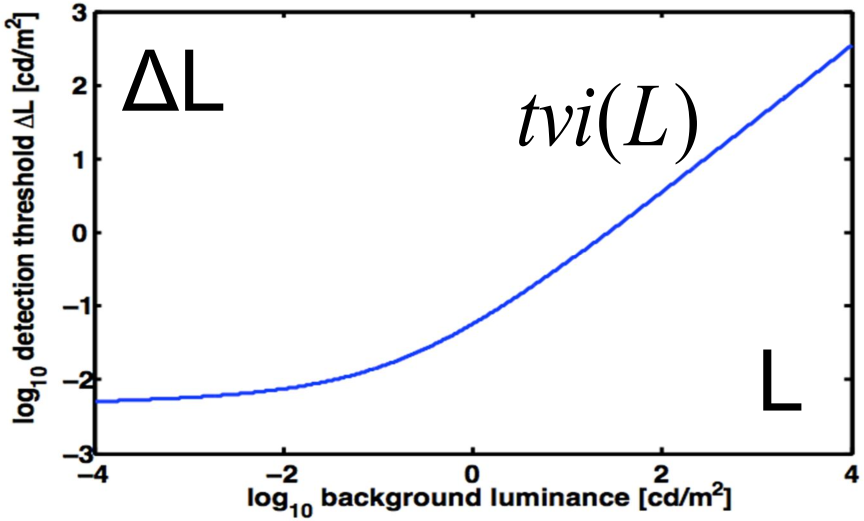
But…the Fechner law does not hold for the full luminance range

- Because the Weber law does not hold either
- Threshold vs. intensity function:



Weber-law revisited

- If we allow detection threshold to vary with luminance according to the t.v.i. function:

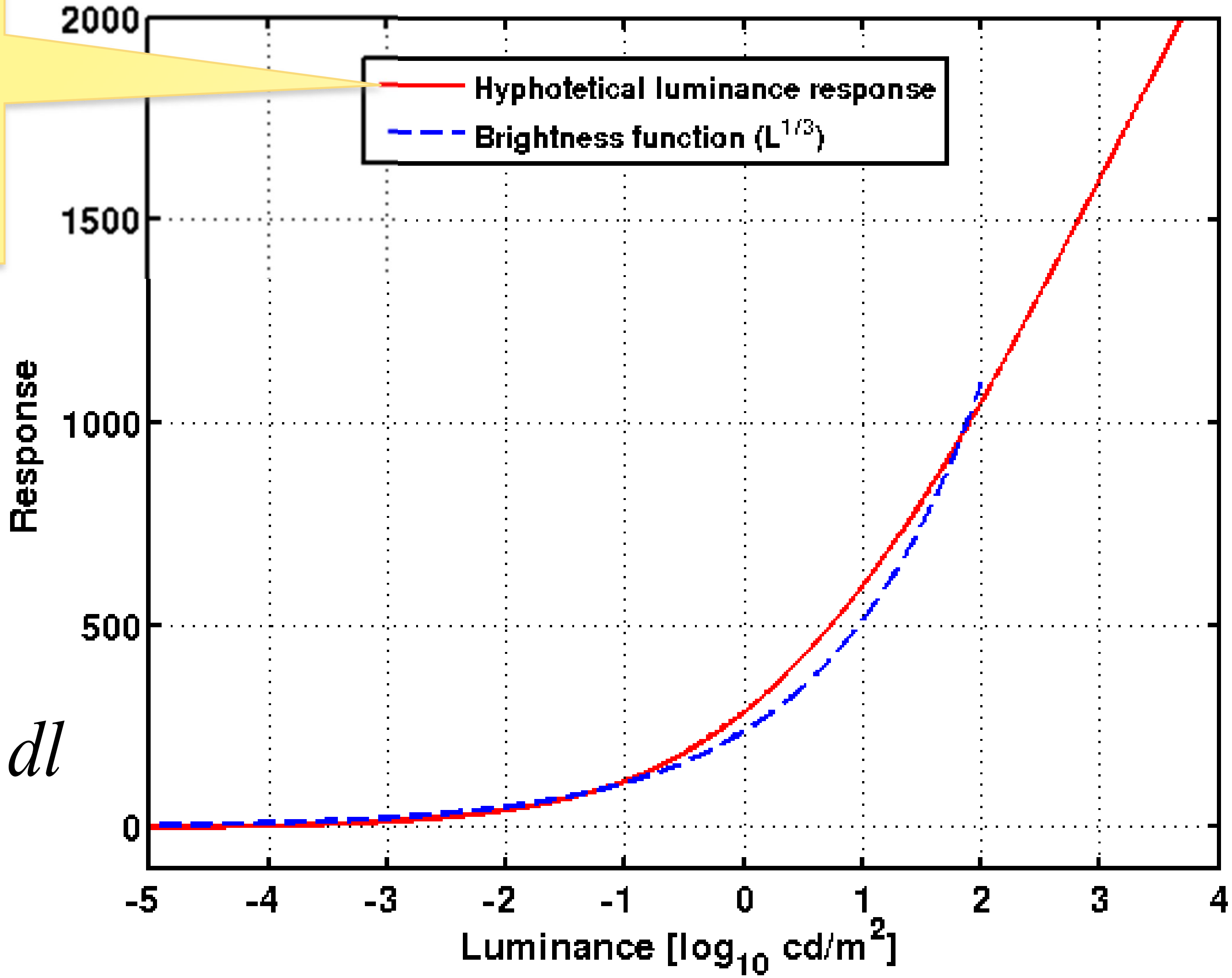


- we can get a more accurate estimate of the "response":

$$R(L) = \int_0^L \frac{1}{tvi(l)} dl$$

Fechnerian integration and Stevens' law

R(L) - function derived from the t.v.i. function

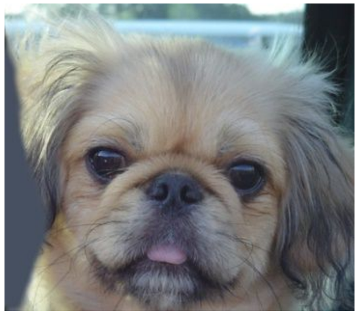


$$R(L) = \int_0^L \frac{1}{tvi(l)} dl$$

Applications of JND encoding – R(L)

- DICOM grayscale function
  - Function used to encode signal for medial monitors
  - 10-bit JND-scaled (just noticeable difference)
  - Equal visibility of gray levels
- HDMI 2.0a (HDR10)
  - PQ (Perceptual Quantizer) encoding
  - Dolby Vision
  - To encode pixels for high dynamic range images and video

DOLBY
VISION
The Future of Vision




© 2006 Aude Oliva & Antonio Torralba

Advanced Graphics and Image Processing

Models of early visual perception

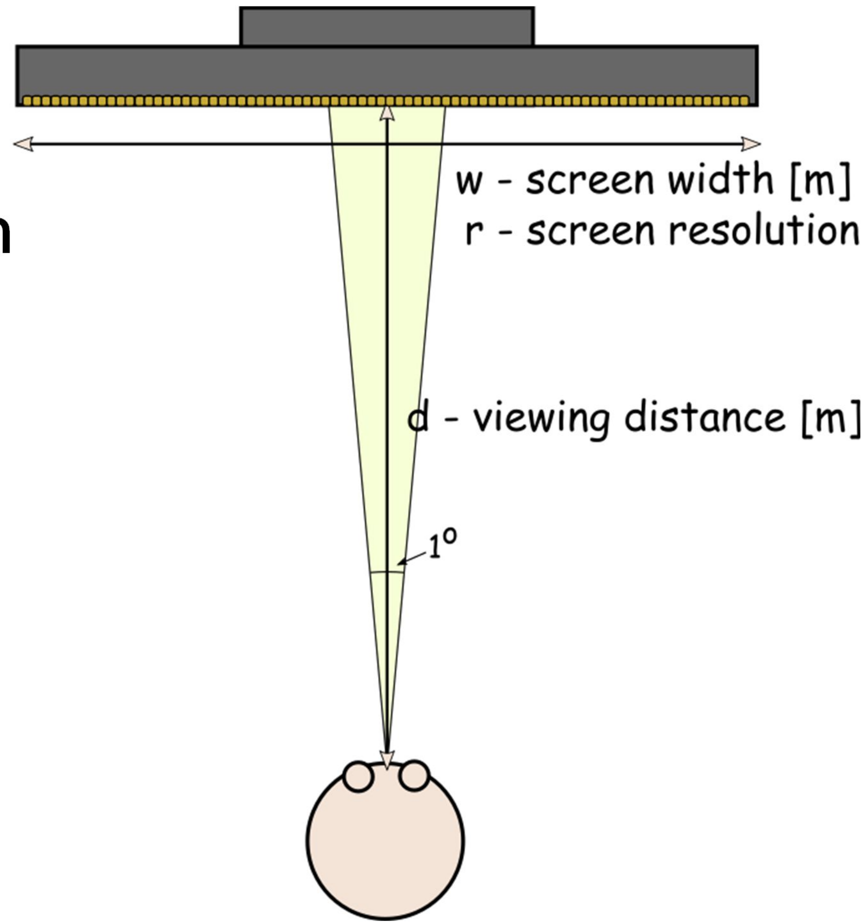
Part 3/6 – spatial contrast sensitivity and contrast constancy

Rafal Mantiuk

Computer Laboratory, University of Cambridge

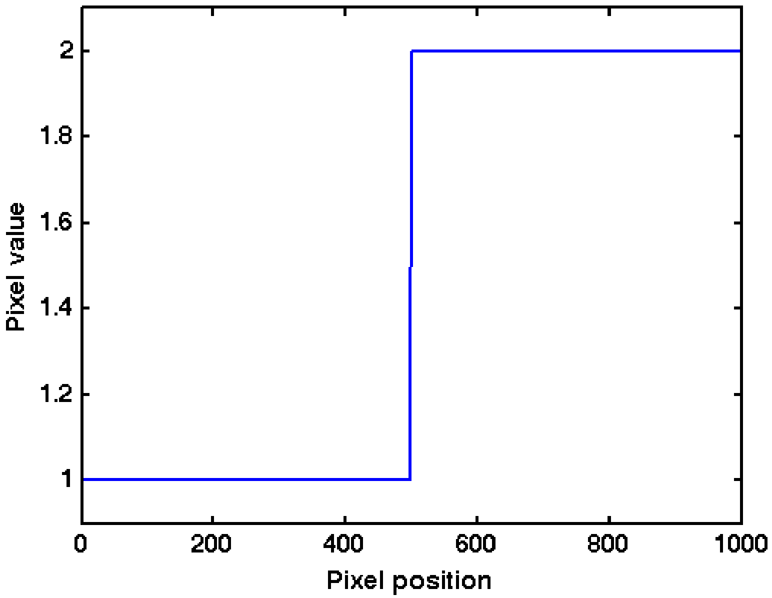
Resolution and sampling rate

- Pixels per inch [ppi]
  - Does not account for vision
- The visual resolution depends on
  - screen size
  - screen resolution
  - viewing distance
- The right measure
  - Pixels per visual degree [ppd]
  - In frequency space
    - Cycles per visual degree [cpd]

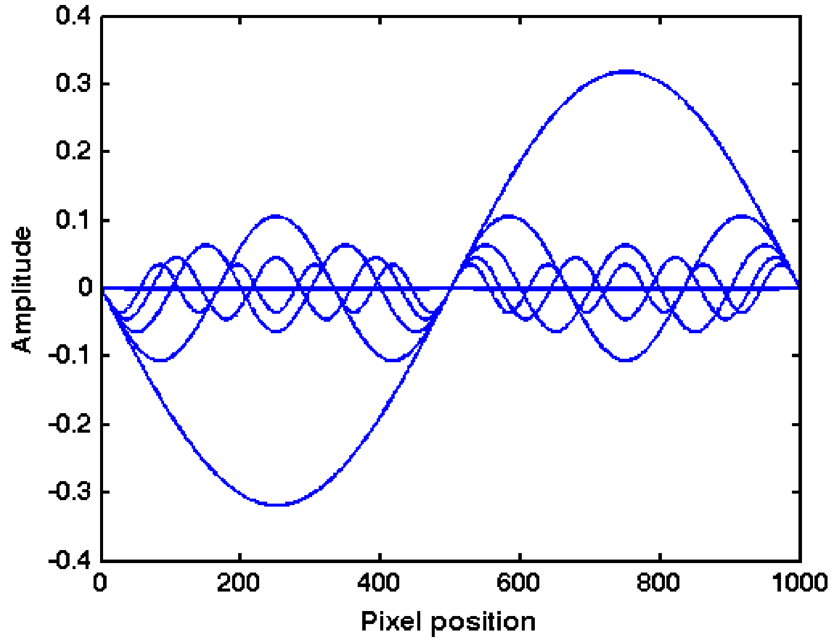


Fourier analysis

- Every N-dimensional function (including images) can be represented as a sum of sinusoidal waves of different frequency and phase



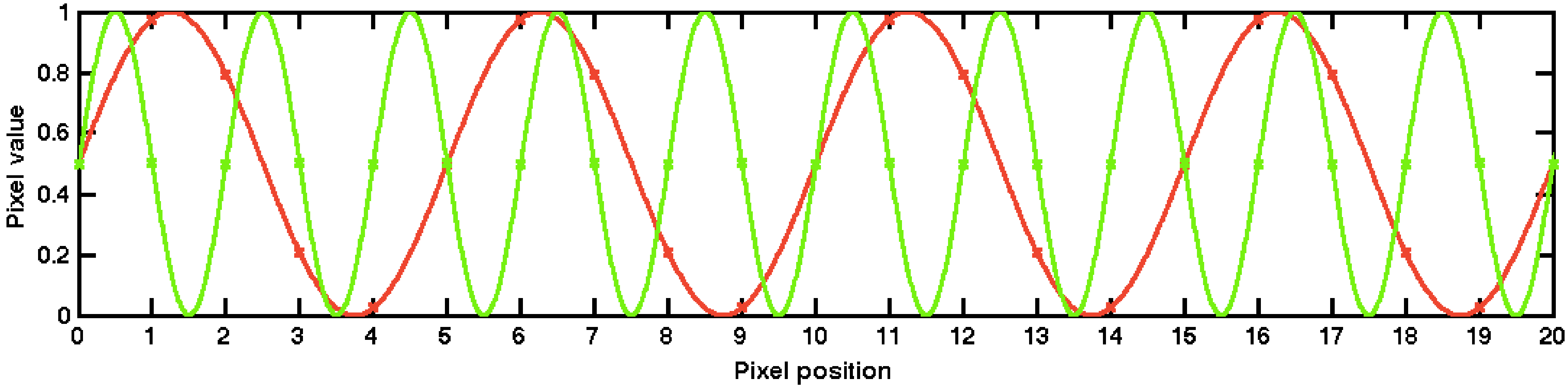
$$=\sum$$



- Think of “equalizer” in audio software, which manipulates each frequency

Spatial frequency in images

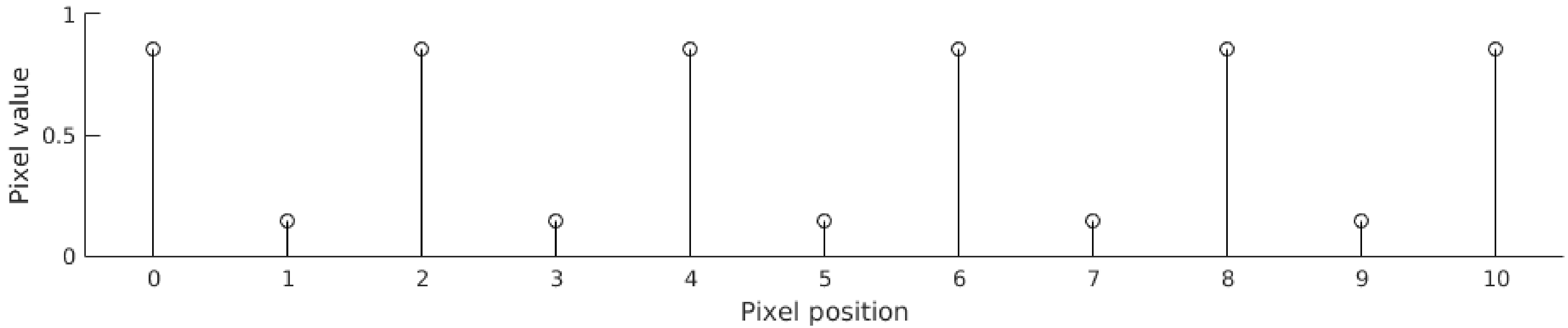
- Image space units: cycles per sample (or cycles per pixel)



  - What are the screen-space frequencies of the red and green sinusoid?
- The visual system units: cycles per degree
  - If the angular resolution of the viewed image is 55 pixels per degree, what is the frequency of the sinusoids in cycles per degree?

Nyquist frequency

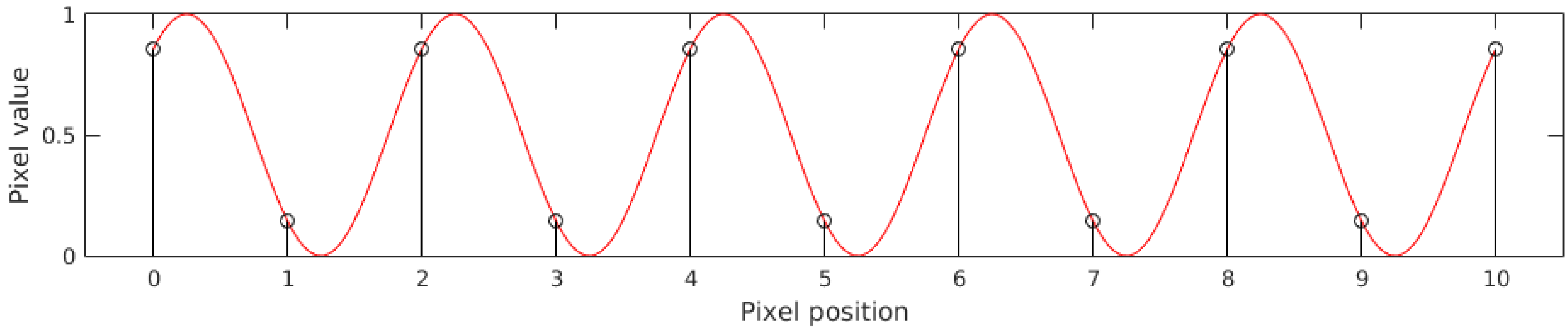
- Sampling density restricts the highest spatial frequency signal that can be (uniquely) reconstructed
  - Sampling density – how many pixels per image/visual angle/…



  - Any number of sinusoids can be fitted to this set of samples
  - It is possible to fit an infinite number of sinusoids if we allow infinitely high frequency

Nyquist frequency

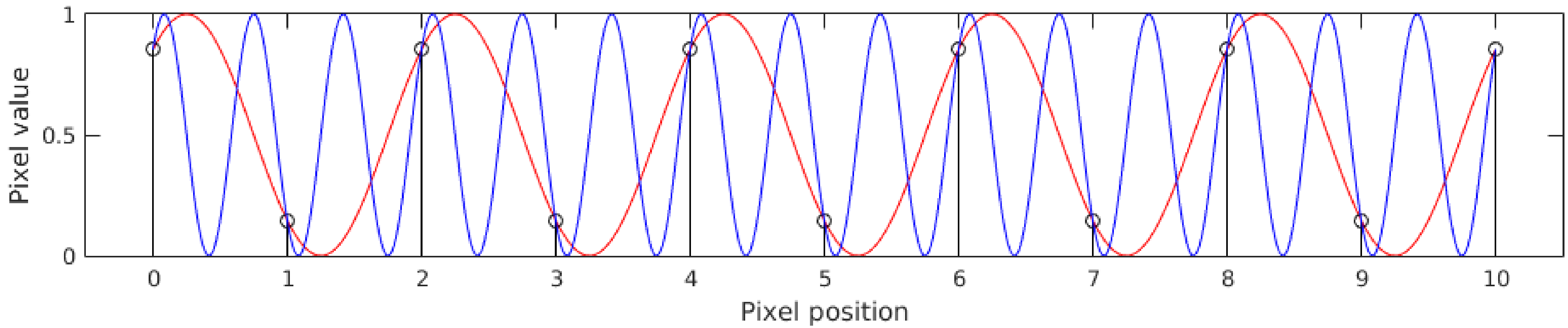
- Sampling density restricts the highest spatial frequency signal that can be (uniquely) reconstructed
  - Sampling density – how many pixels per image/visual angle/…



  - Any number of sinusoids can be fitted to this set of samples
  - It is possible to fit an infinite number of sinusoids if we allow infinitely high frequency

Nyquist frequency

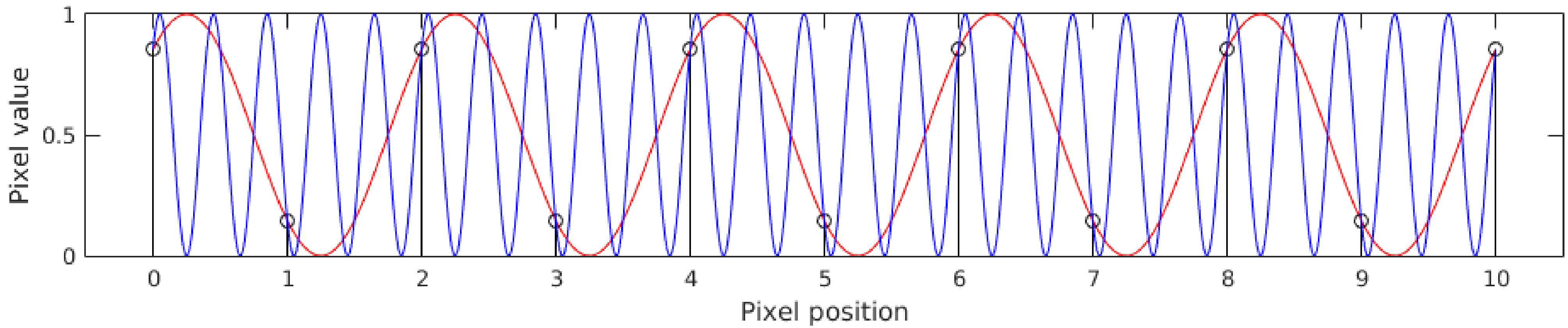
- Sampling density restricts the highest spatial frequency signal that can be (uniquely) reconstructed
  - Sampling density – how many pixels per image/visual angle/…



  - Any number of sinusoids can be fitted to this set of samples
  - It is possible to fit an infinite number of sinusoids if we allow infinitely high frequency

Nyquist frequency

- Sampling density restricts the highest spatial frequency signal that can be (uniquely) reconstructed
  - Sampling density – how many pixels per image/visual angle/…

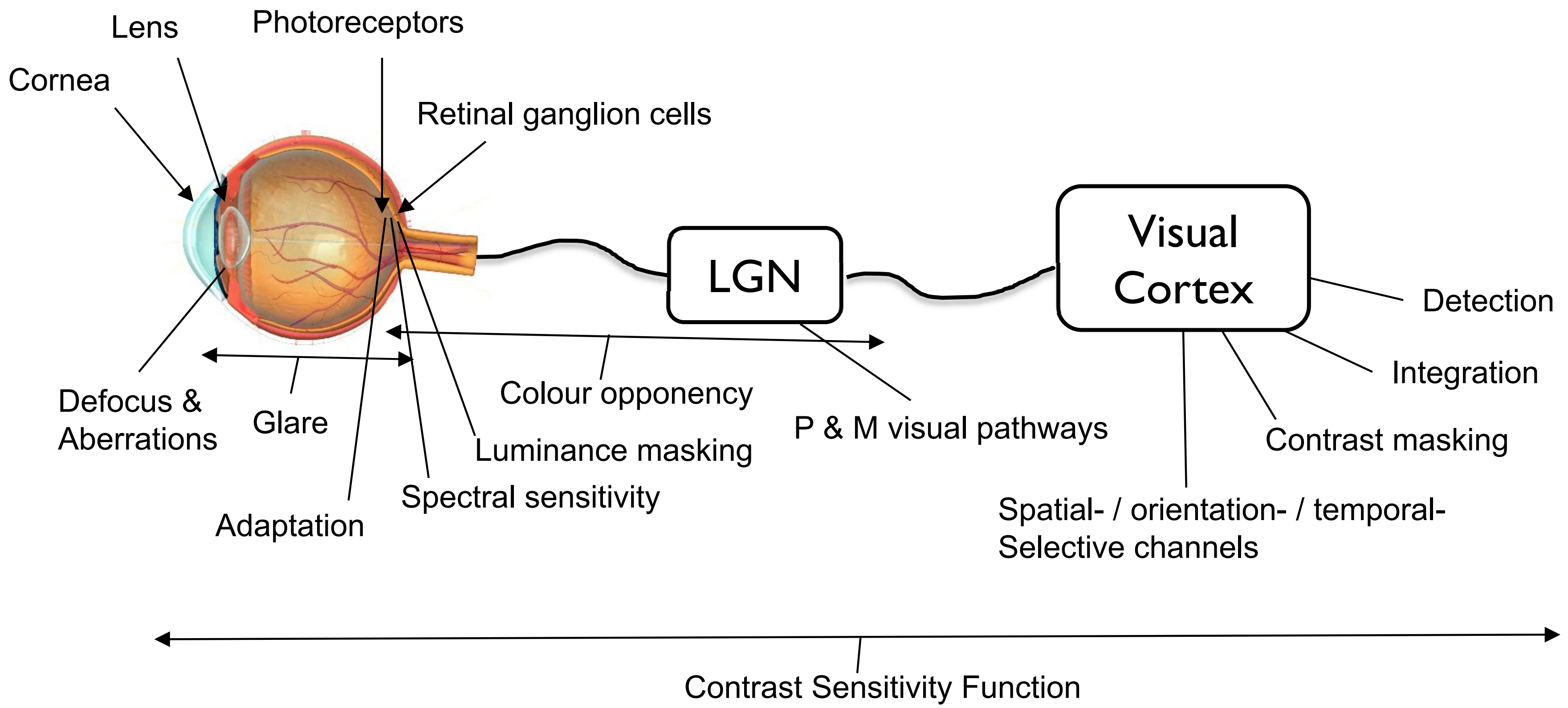


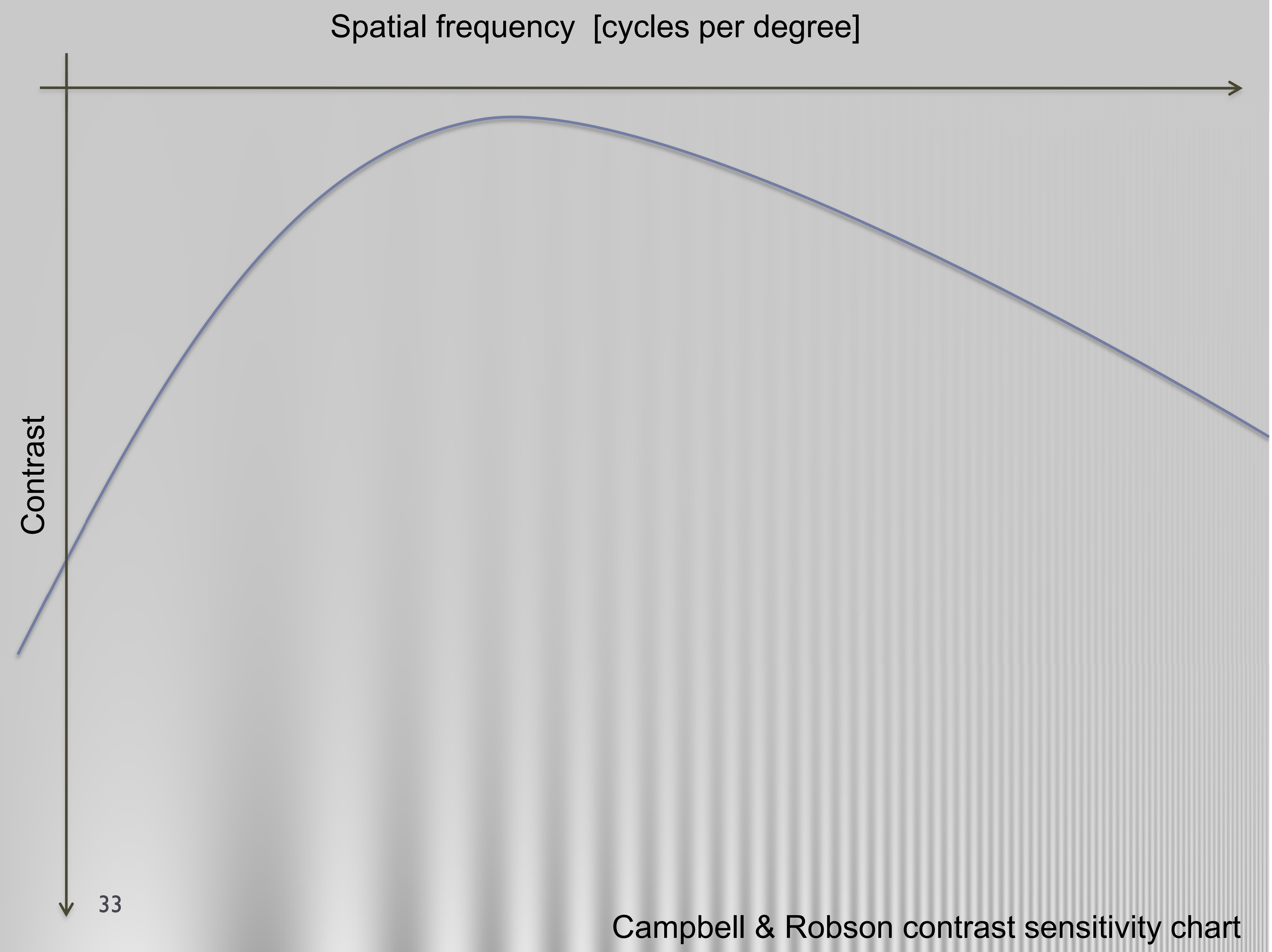
  - Any number of sinusoids can be fitted to this set of samples
  - It is possible to fit an infinite number of sinusoids if we allow infinitely high frequency

Nyquist frequency / aliasing

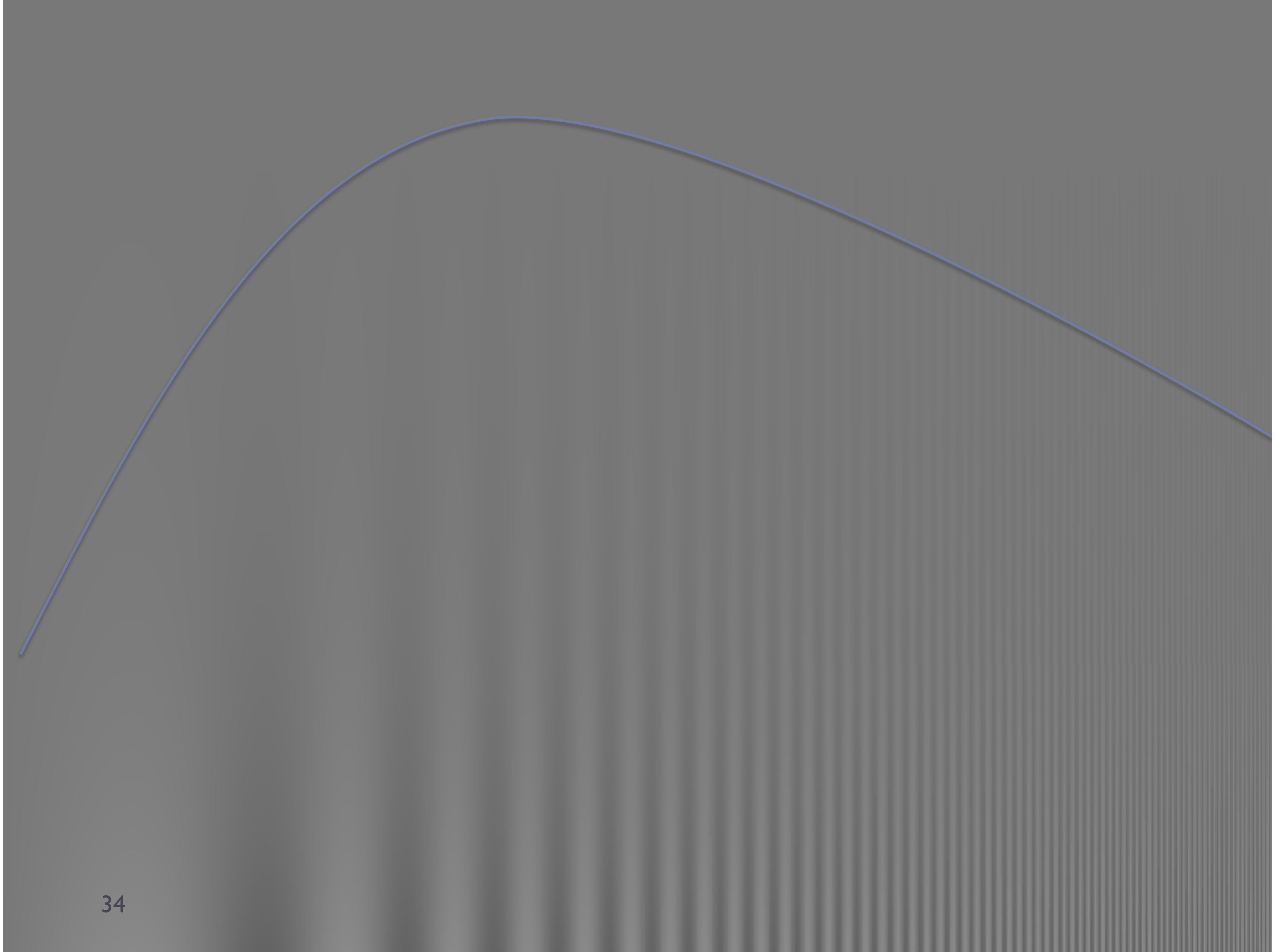
- Nuquist frequency is the highest frequency that can be represented by a discrete set of uniform samples (pixels)
- Nuquist frequency = 0.5 sampling rate
  - For audio
    - If the sampling rate is 44100 samples per second (audio CD), then the Nyquist frequency is 22050 Hz
  - For images (visual degrees)
    - If the sampling rate is 60 pixels per degree, then the Nyquist frequency is 30 cycles per degree
- When resampling an image to lower resolution, the frequency content above the Nyquist frequency needs to be removed (reduced in practice)
  - Otherwise **aliasing** is visible

Modeling contrast detection





Campbell & Robson contrast sensitivity chart



Contrast sensitivity function

Temporal frequency

Stimulus size

Spatial frequency

Eccentricity

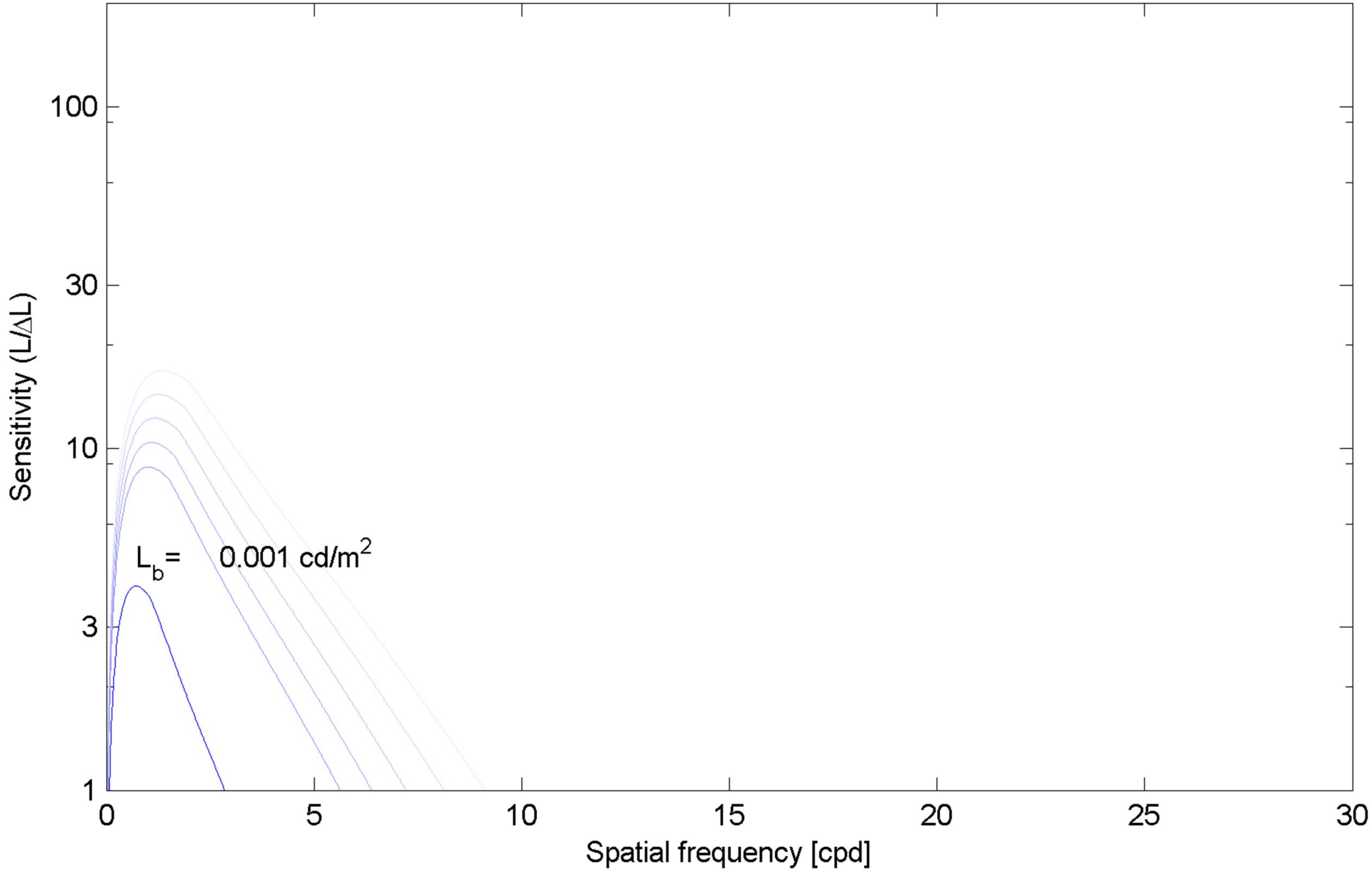
$$CSF = S(\rho, \theta, \omega, l, i^2, d, e)$$

Orientation

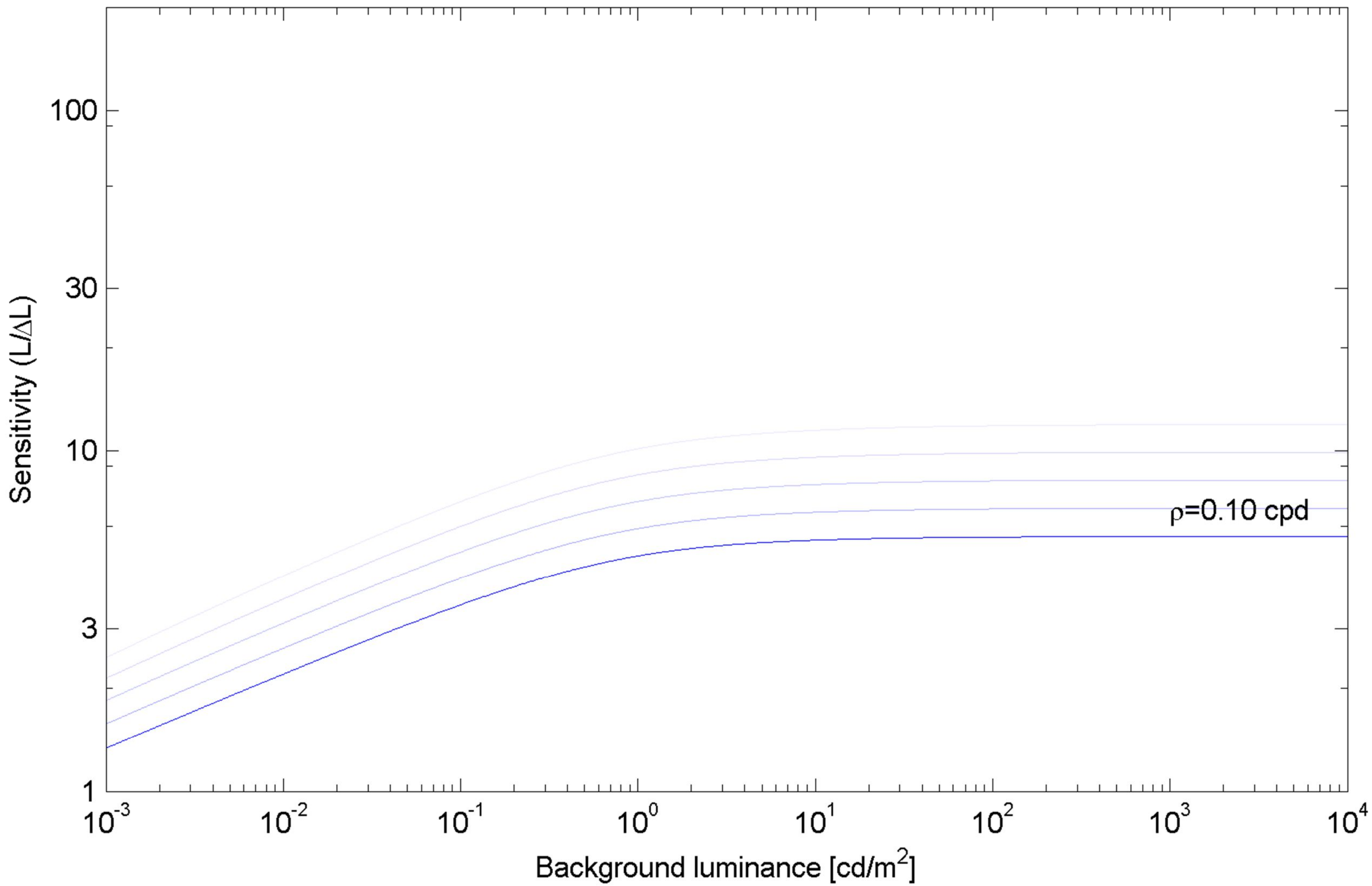
Viewing distance

Adapting luminance

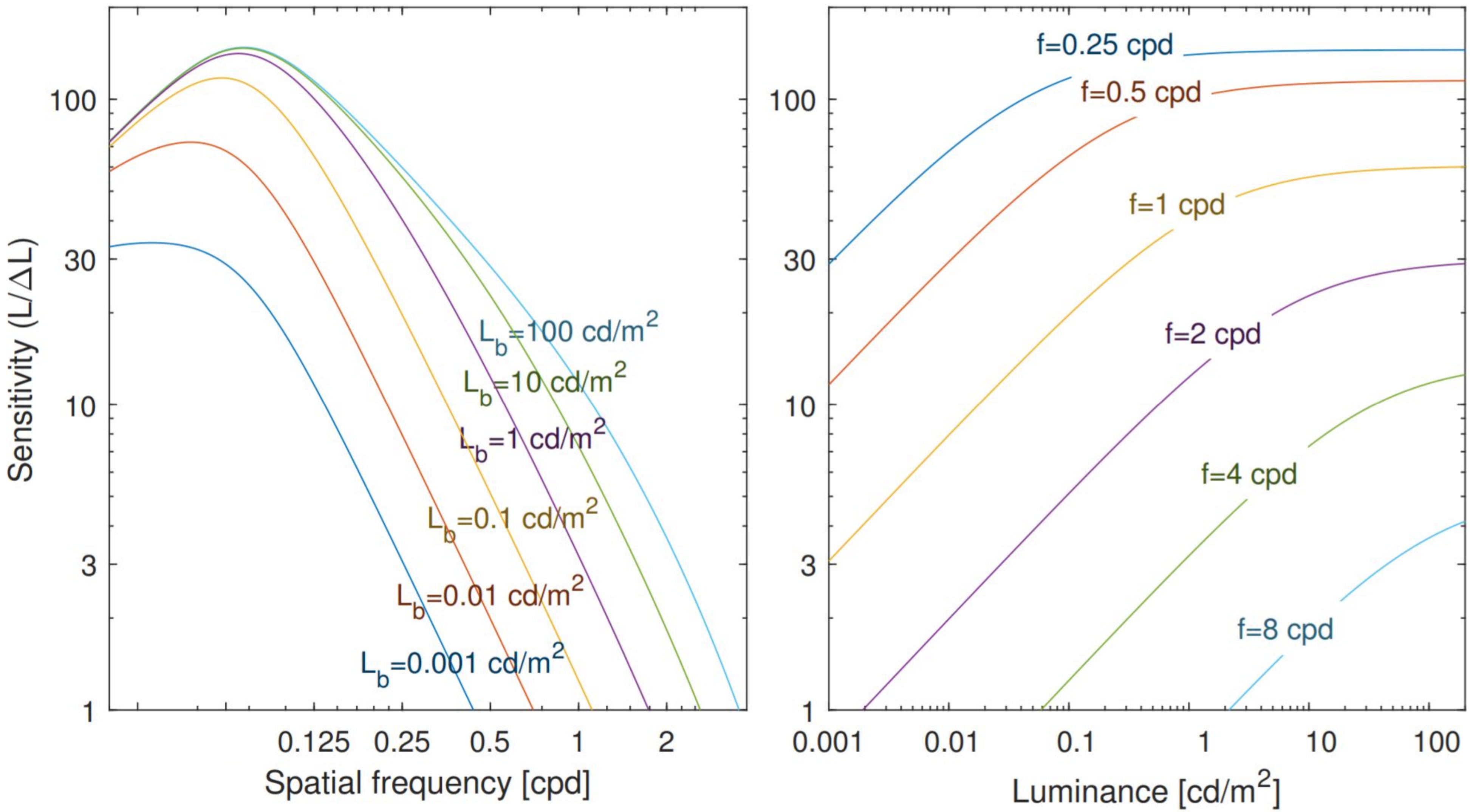
CSF as a function of spatial frequency



CSF as a function of background luminance



CSF as a function of spatial frequency and background luminance



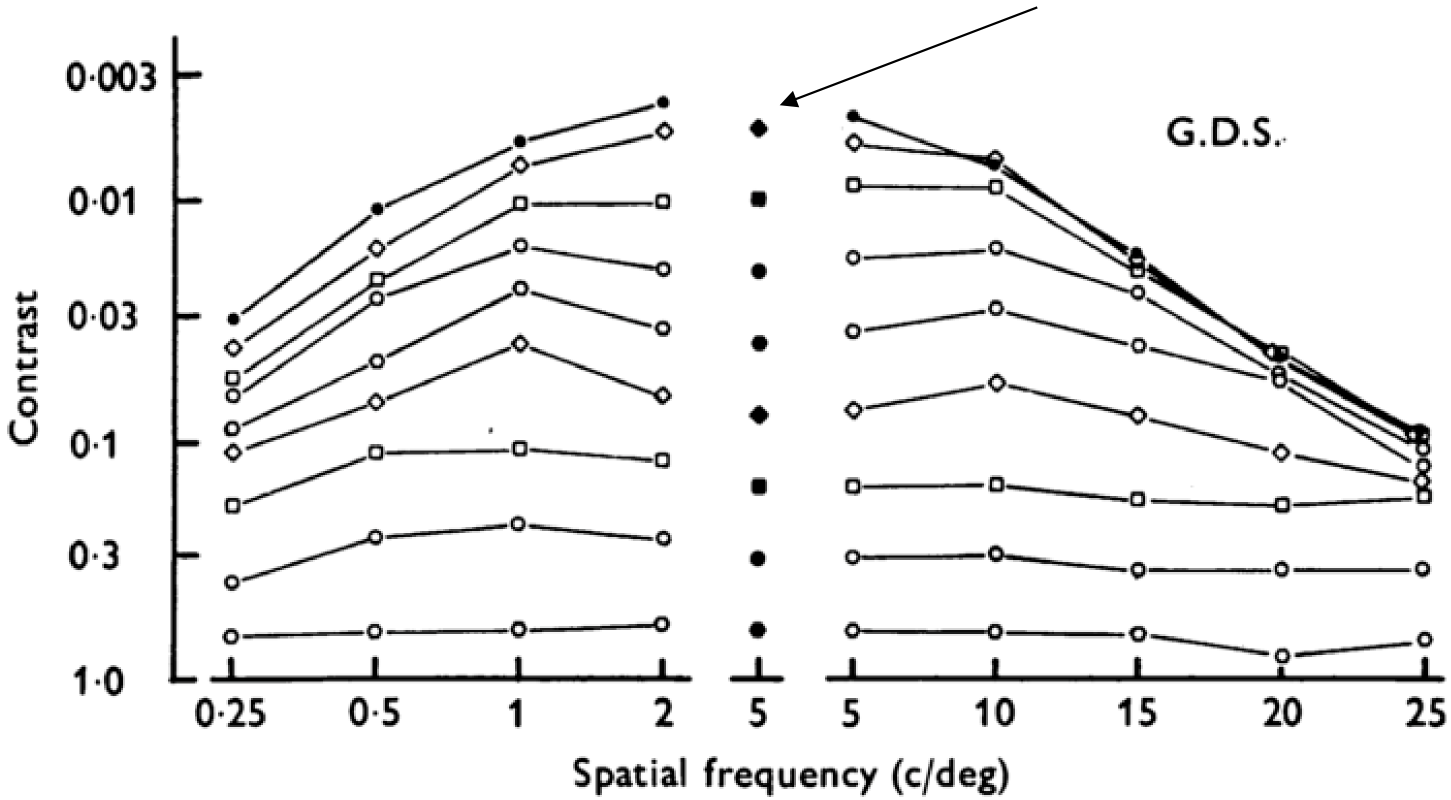
Contrast constancy

Experiment: Adjust the amplitude of one sinusoidal grating until it matches the perceived magnitude of another sinusoidal grating.

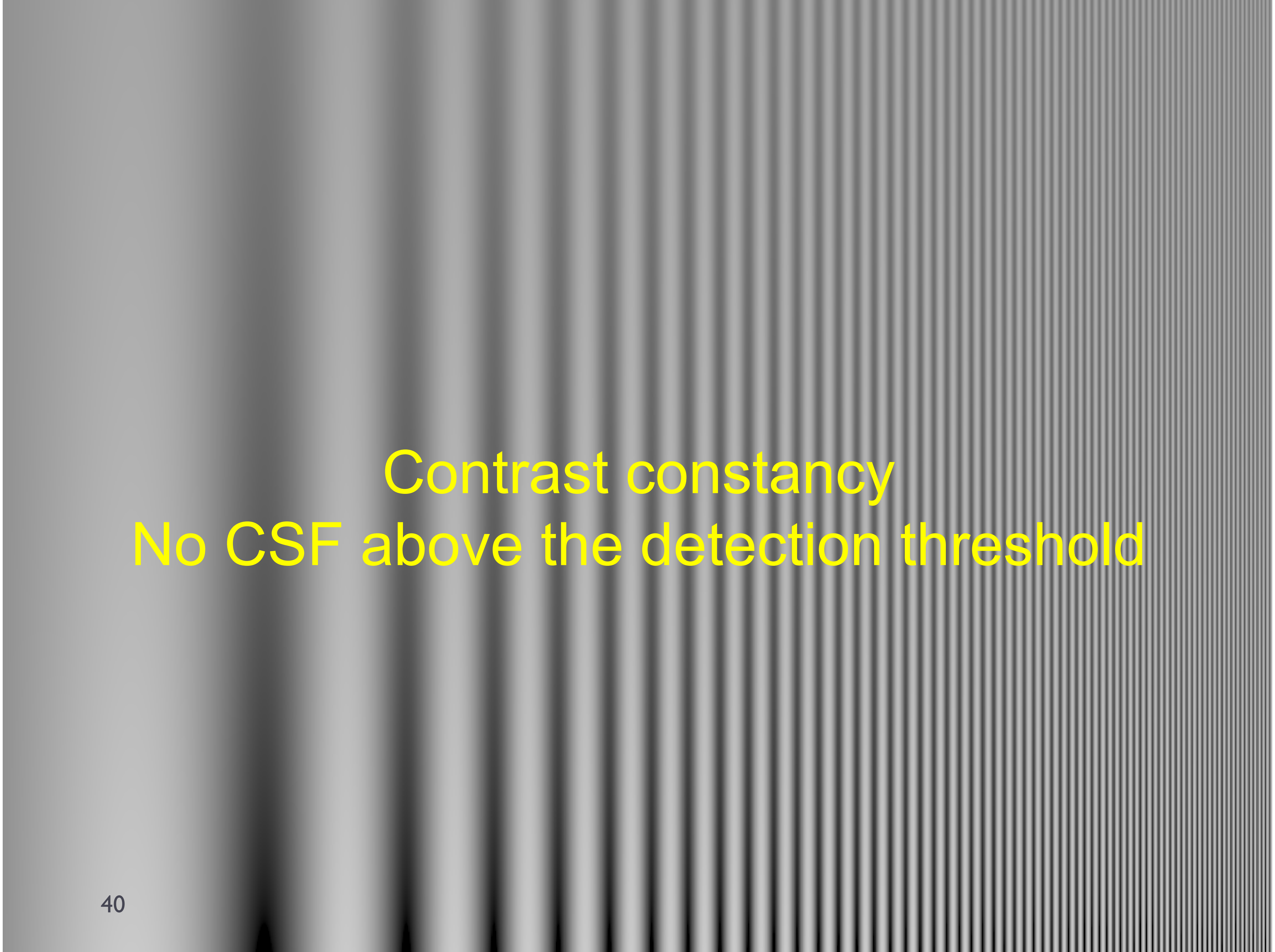
Test

Match?

Reference



From: Georgeson and Sullivan. 1975. J. Phsysio.

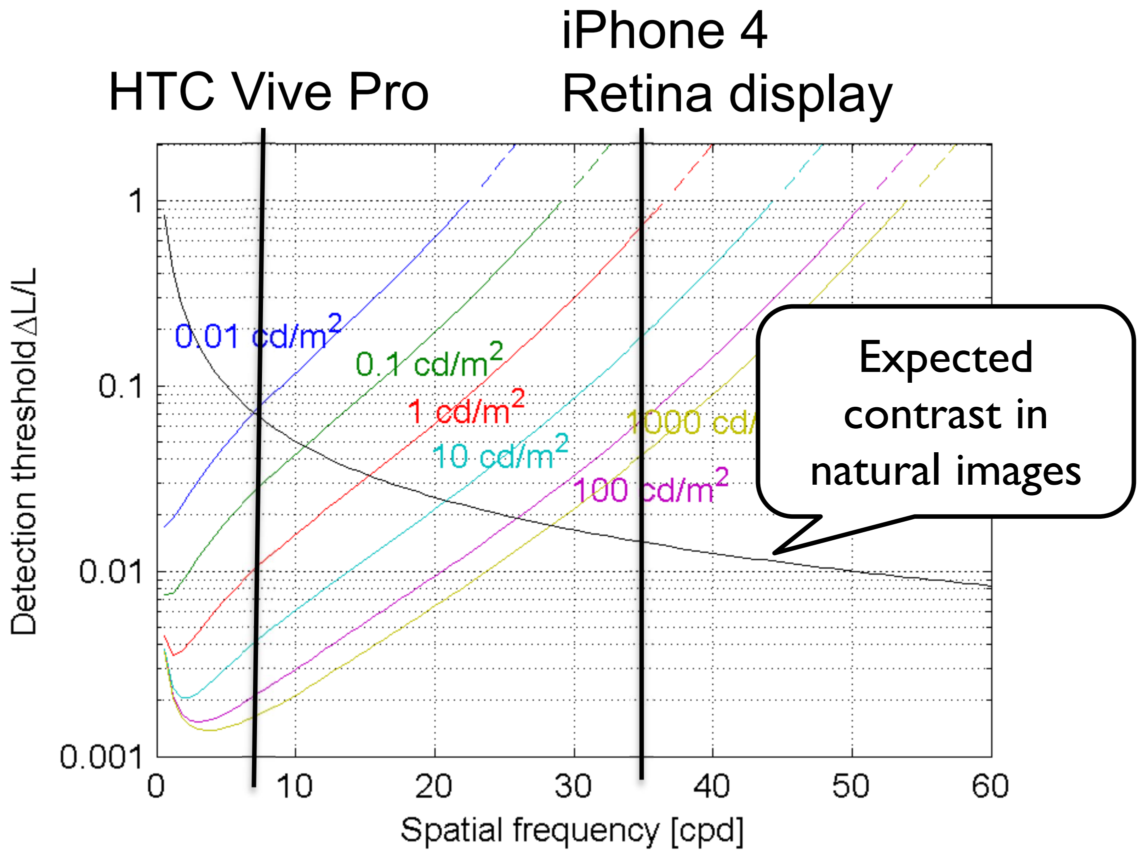


Contrast constancy

No CSF above the detection threshold

CSF and the resolution

- CSF plotted as the detection contrast
$$\frac{\Delta L}{L_b} = S^{-1}$$
- The contrast below each line is invisible
- Maximum perceivable resolution depends on luminance



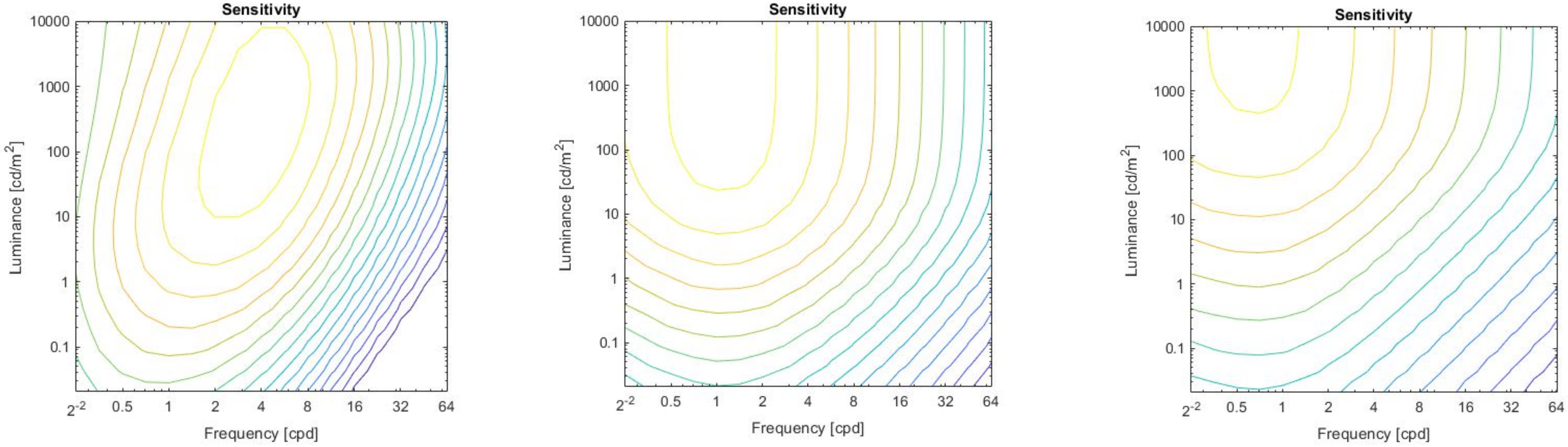
CSF models:
Barten, P. G. J. (2004).
https://doi.org/10.1117/12.537476

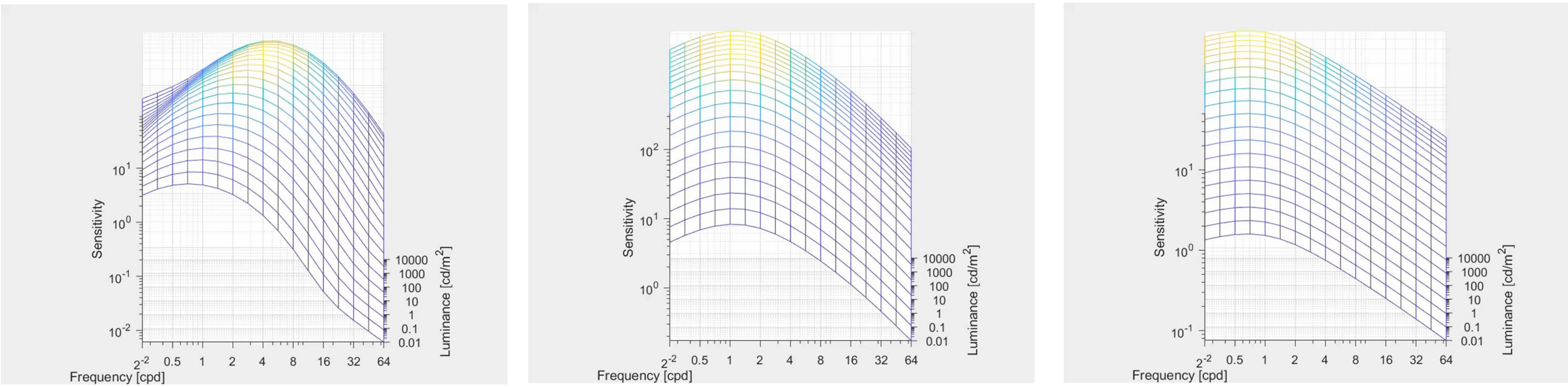
Spatio-chromatic CSF

Spatio-chromatic contrast sensitivity

- CSF as a function of **luminance** and **frequency**

Black-White | Red-Green | Violet-Yellow

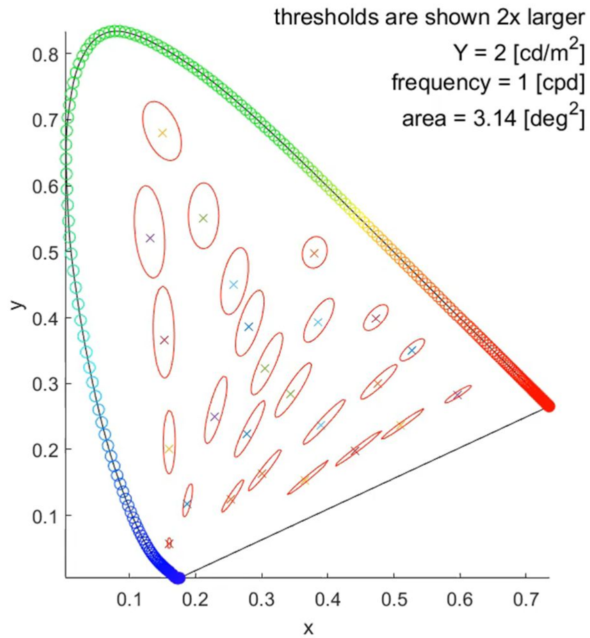


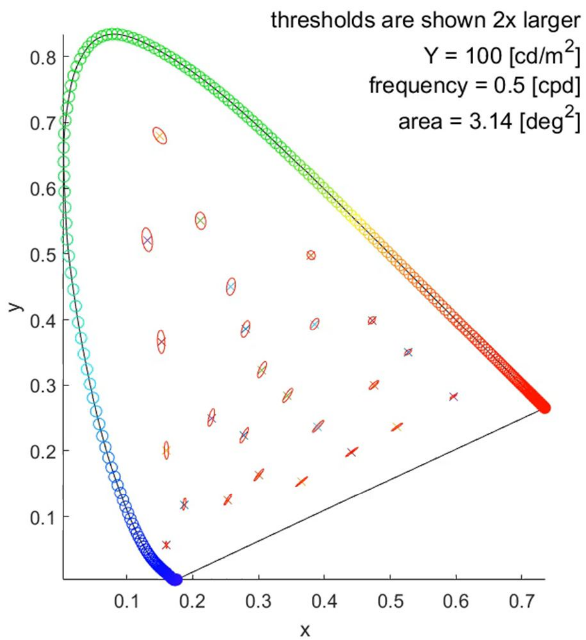


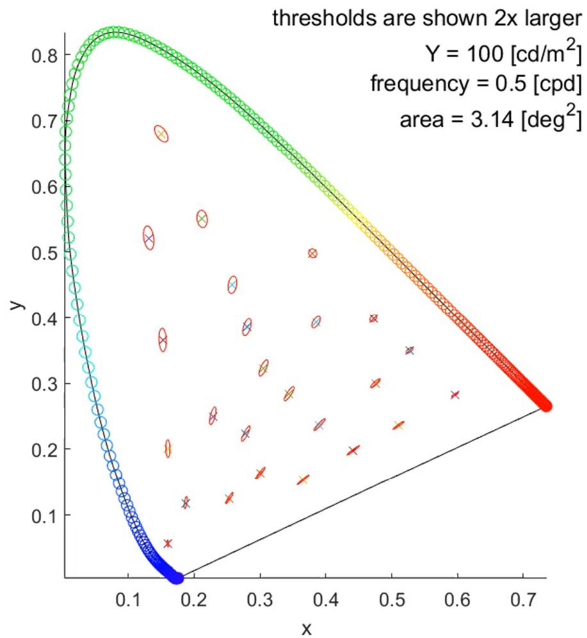
Rafał Mantiuk, University of Cambridge
http://dx.doi.org/10.2352/issn.2169-2629.2020.28.1

CSF and colour ellipses

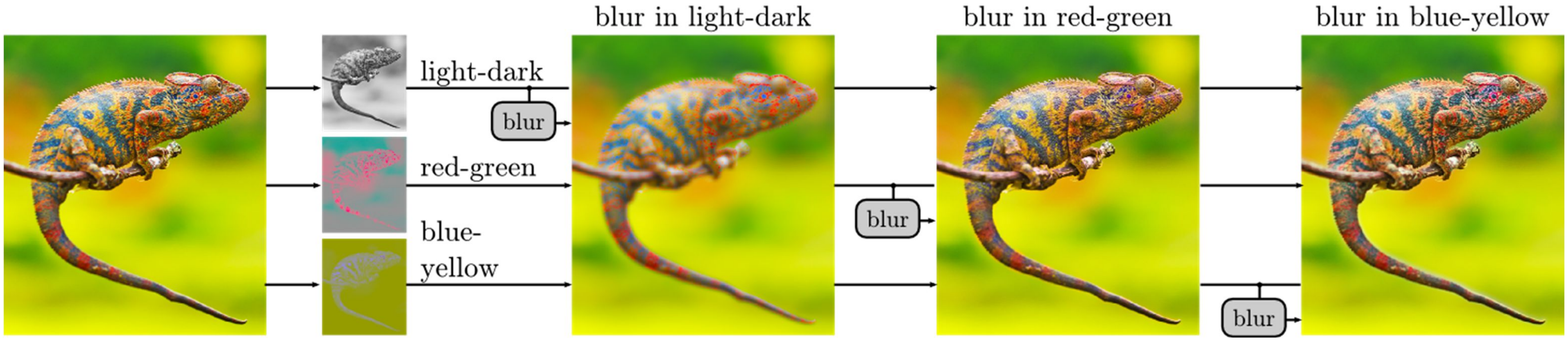
- Colour discrimination as a function of
  - Background colour and luminance [LMS]
  - Spatial frequency [cpd]
  - Size [deg]







Visibility of blur



- The same amount of blur was introduced into light-dark, red-green and blue-yellow colour opponent channels
- The blur is only visible in light-dark channel
- This property is used in image and video compression
  - Sub-sampling of colour channels (4:2:1)

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY
RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

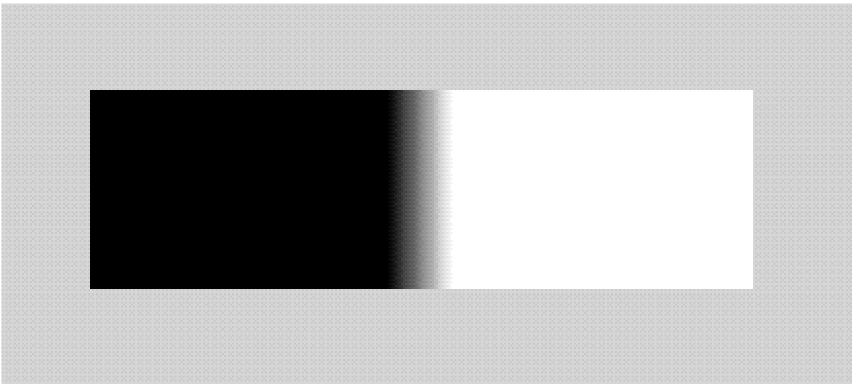
Models of early visual perception

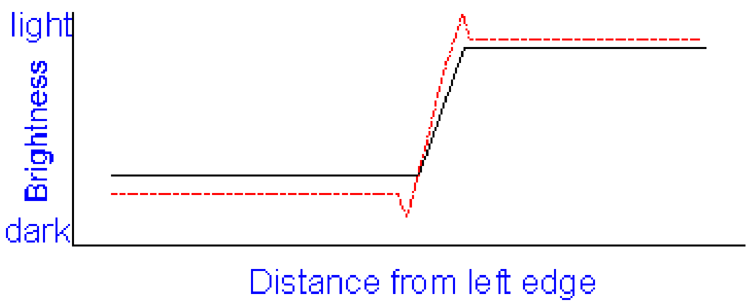
Part 4/6 – lateral inhibition and multi-resolution models

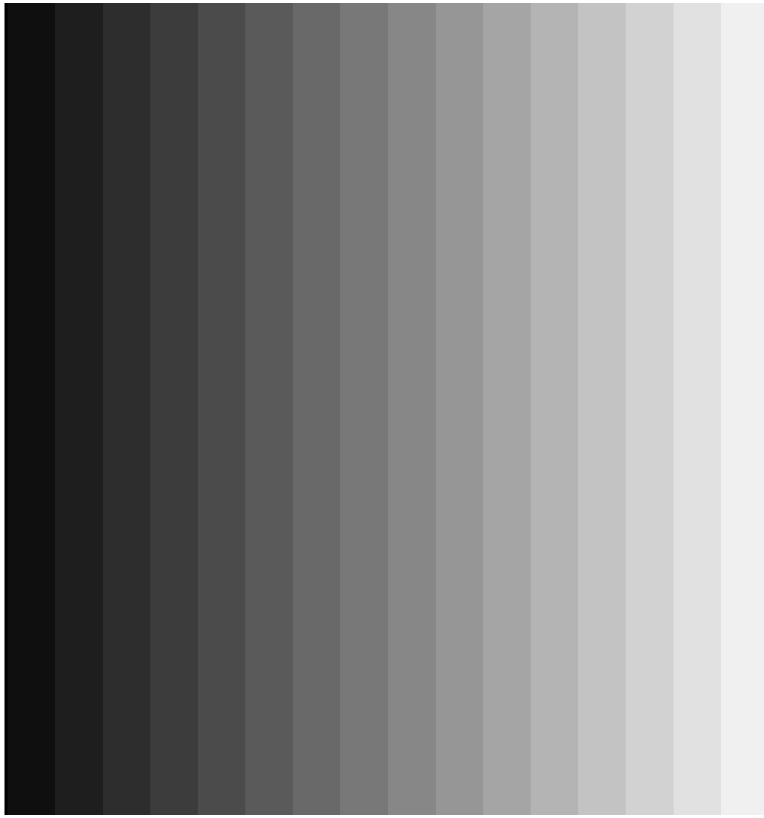
Rafal Mantiuk
Computer Laboratory, University of Cambridge

Mach Bands – evidence for band-pass visual processing

- "Overshooting" along edges
  - Extra-bright rims on bright sides
  - Extra-dark rims on dark sides
- Due to "Lateral Inhibition"



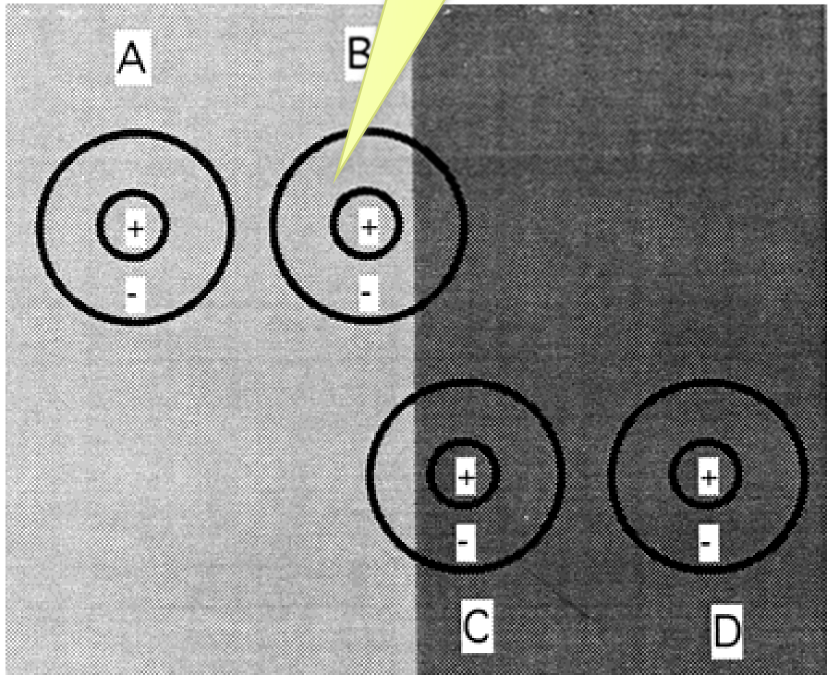


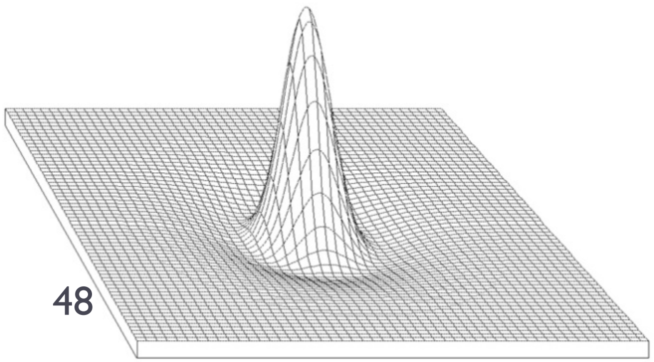


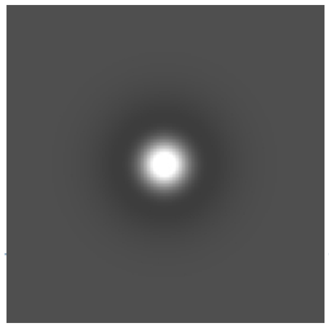
Centre-surround (Lateral Inhibition)

- "Pre-processing" step within the retina
  - Surrounding brightness level weighted negatively
    - A: high stimulus, maximal bright inhibition
    - B: high stimulus, reduced inhibition & stronger response
    - D: low stimulus, maximal inhibition
    - C: low stimulus, increased inhibition & weaker response

Center-surround receptive fields (groups of photoreceptors)

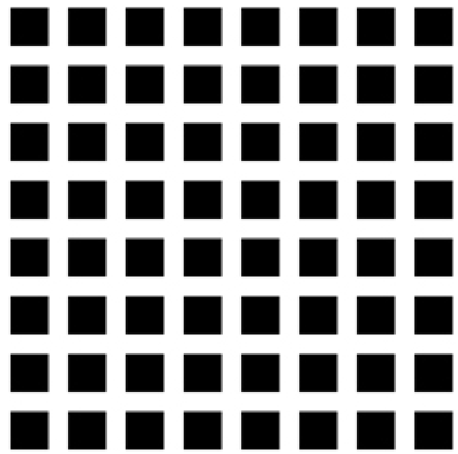


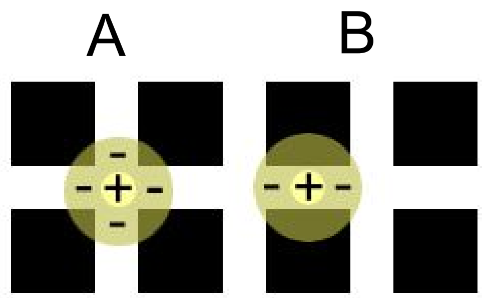




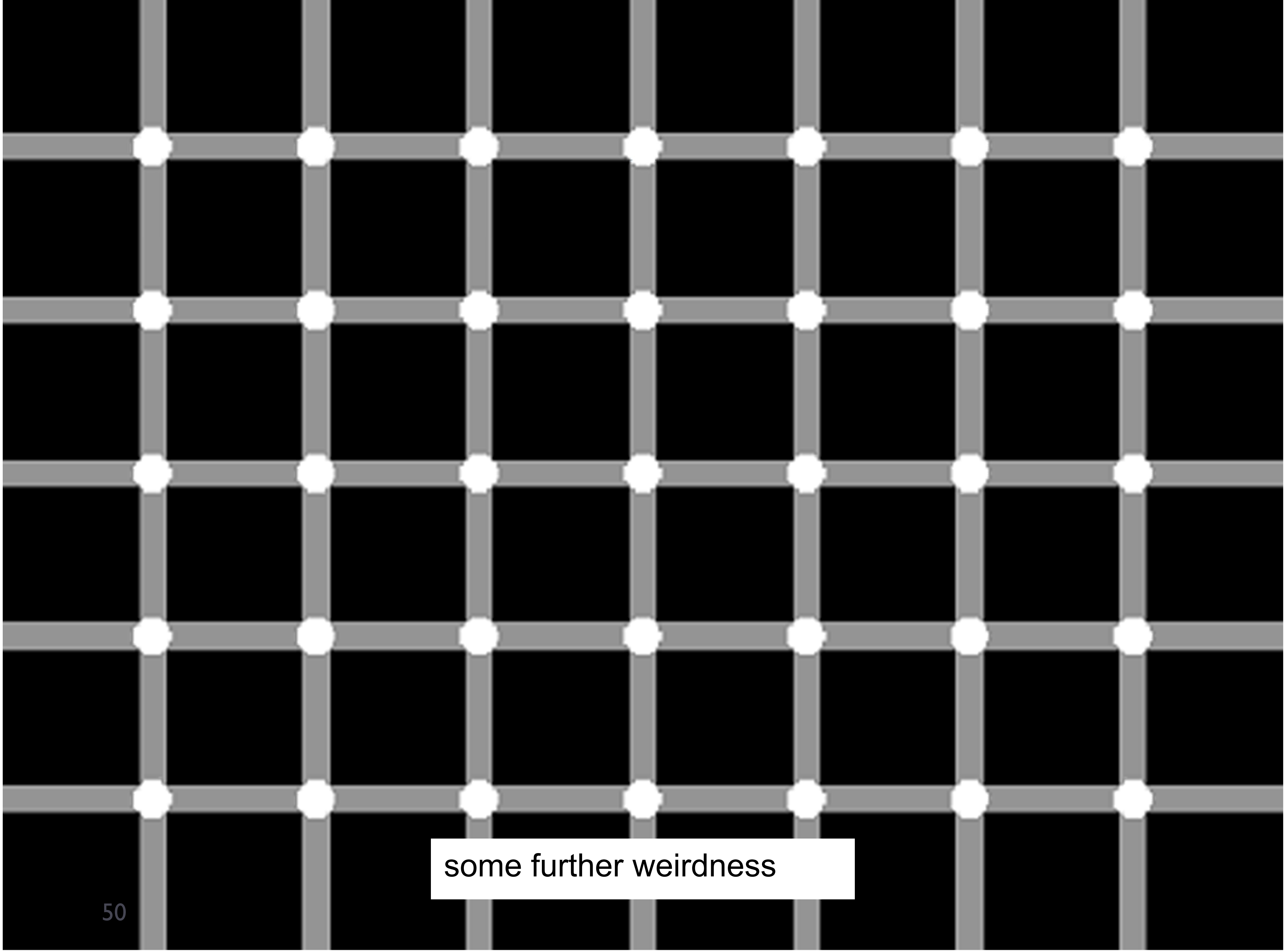
Centre-surround: Hermann Grid

- Dark dots at crossings
- Explanation
  - Crossings (A)
    - More surround stimulation (more bright area)
    - ⇒ Less inhibition
    - ⇒ Weaker response
  - Streets (B)
    - Less surround stimulation
    - ⇒ More inhibition
    - ⇒ Greater response
- Simulation
  - Darker at crossings, brighter in streets
  - Appears more steady
  - What if reversed ?





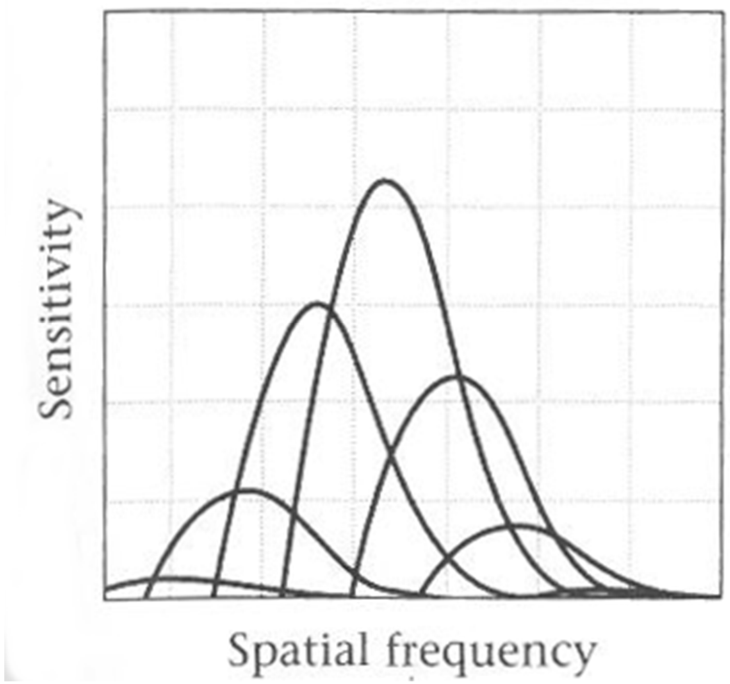
Simulation



some further weirdness

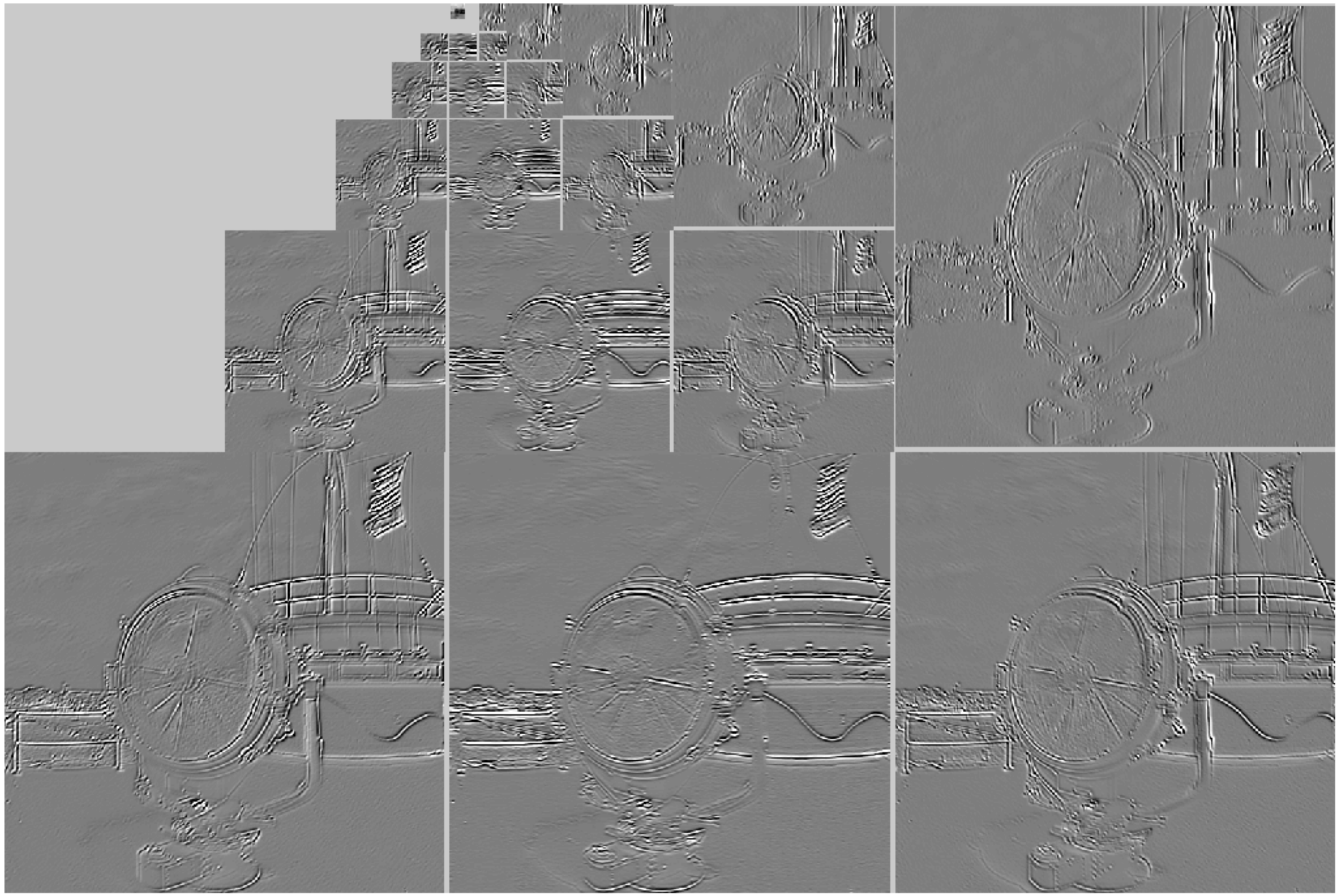
Spatial-frequency selective channels

- The visual information is decomposed in the visual cortex into multiple channels
  - The channels are selective to spatial frequency, temporal frequency and orientation
  - Each channel is affected by different „noise" level
  - The CSF is the net result of information being passed in noise-affected visual channels



From: Wandell, 1995

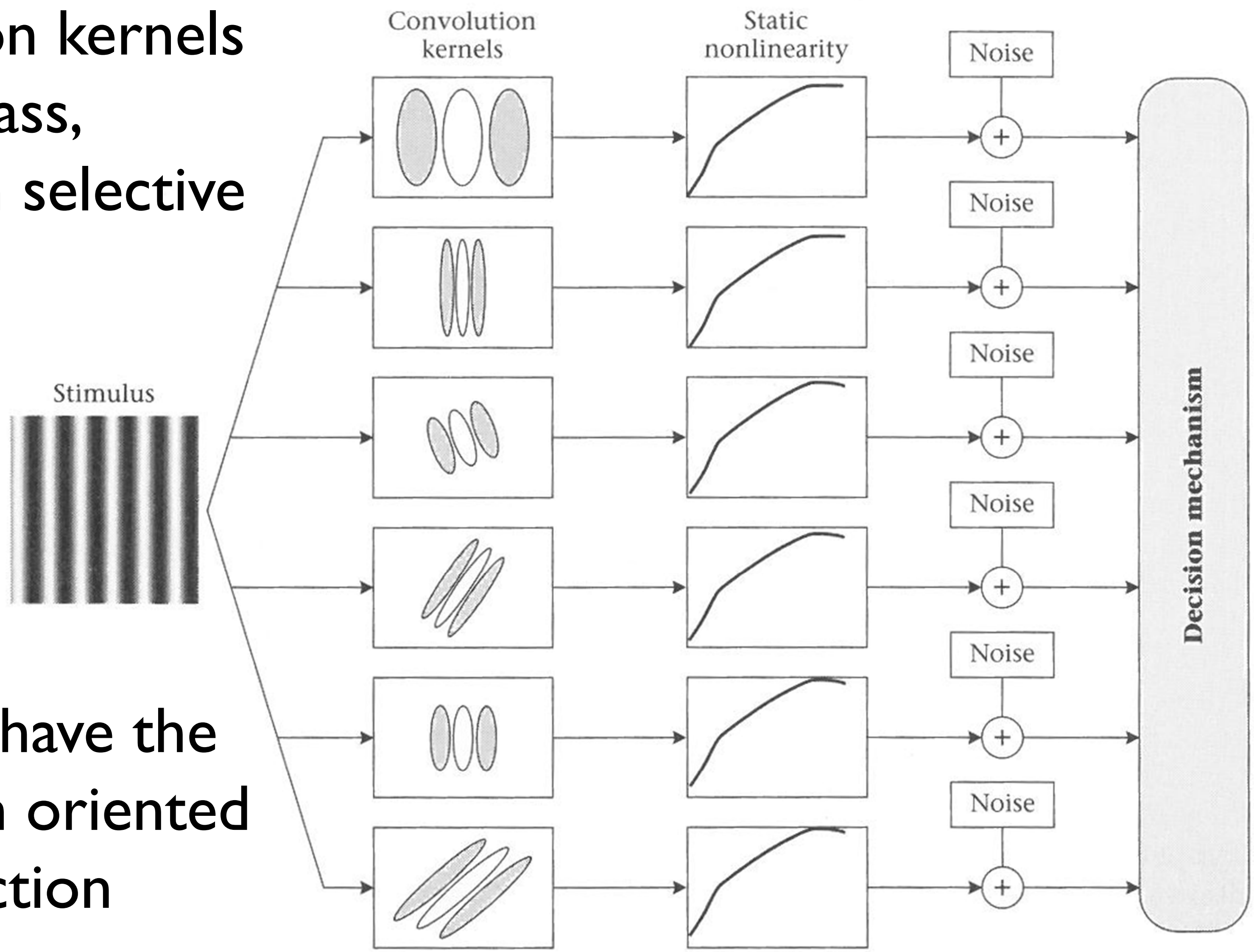
Multi-scale decomposition



Steerable pyramid decomposition

Multi-resolution visual model

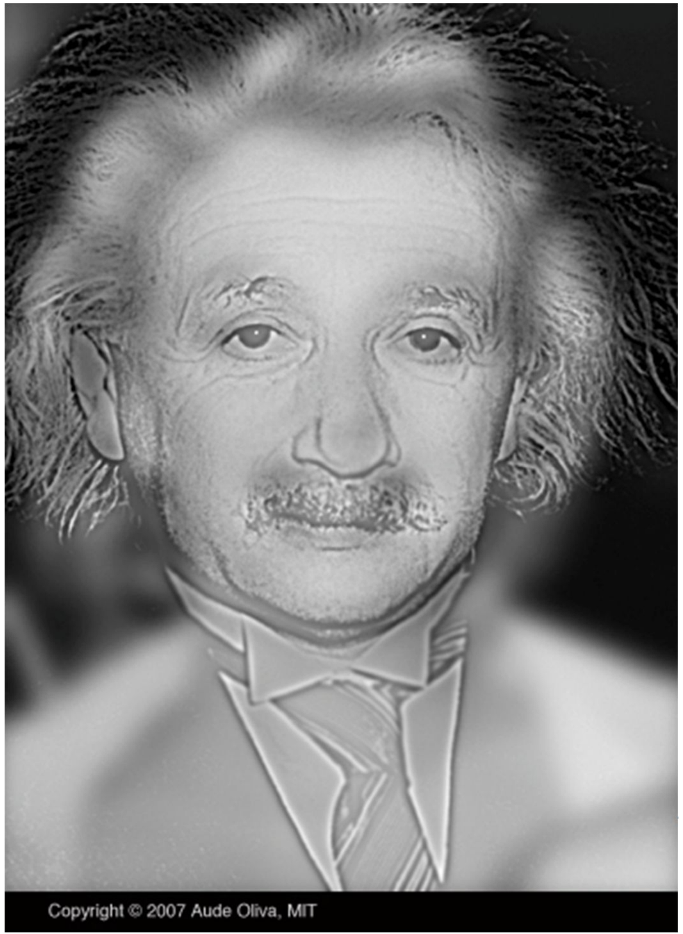
- Convolution kernels are band-pass, orientation selective filters
- The filters have the shape of an oriented Gabor function

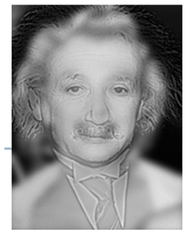


From: Wandell, 1995

Applications of multi-scale models

- JPEG2000
  - Wavelet decomposition
- JPEG / MPEG
  - Frequency transforms
- Image pyramids
  - Blending & stitching
  - Hybrid images





Hybrid Images by Aude Oliva
http://cvcl.mit.edu/hybrid_gallery

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

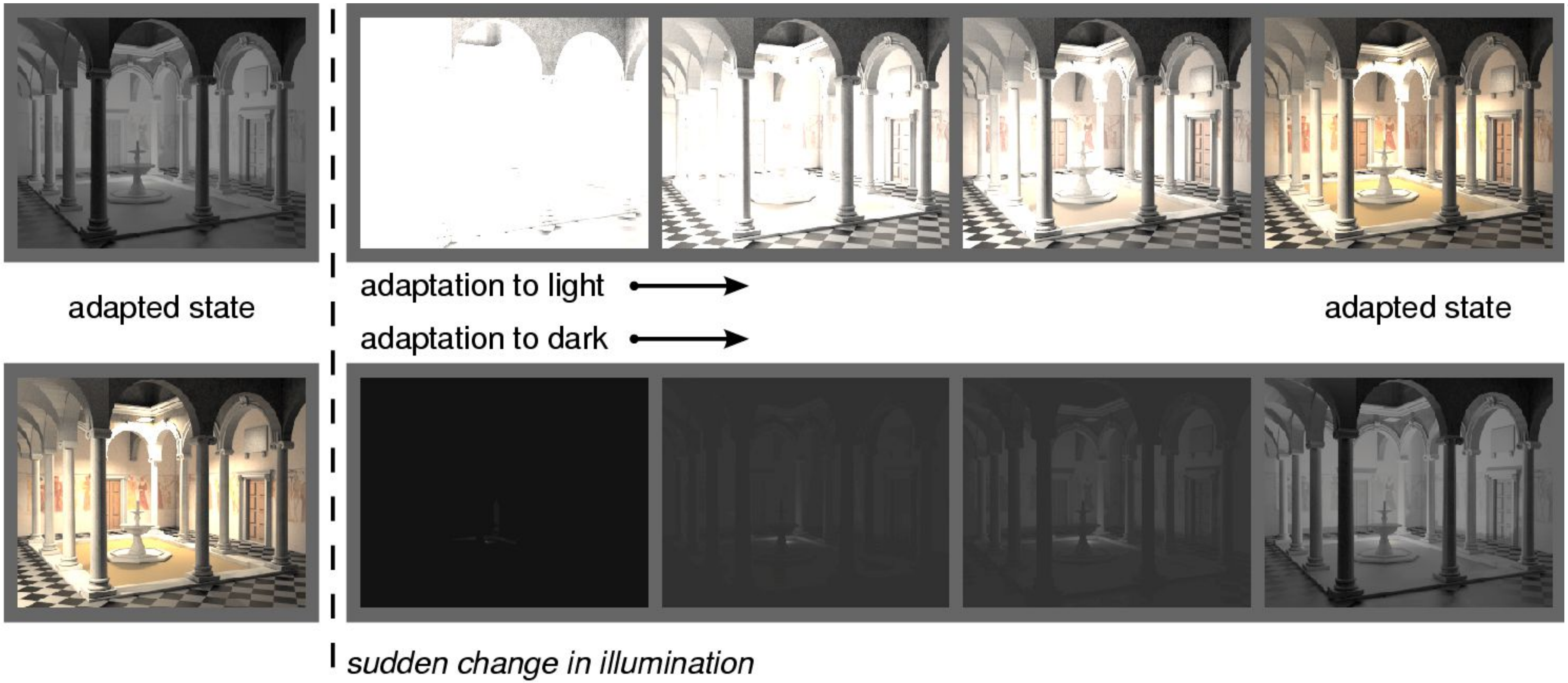
Advanced Graphics and Image Processing

Models of early visual perception

Part 5/6 – light and dark adaptation

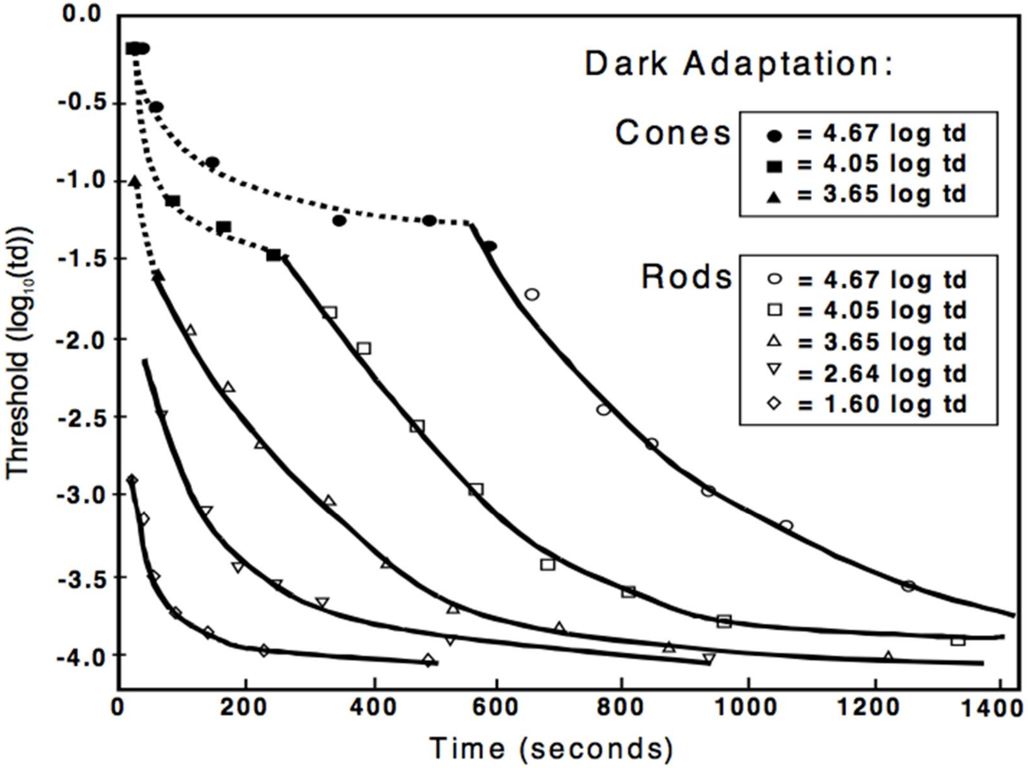
Rafal Mantiuk
Computer Laboratory, University of Cambridge

Light and dark adaptation

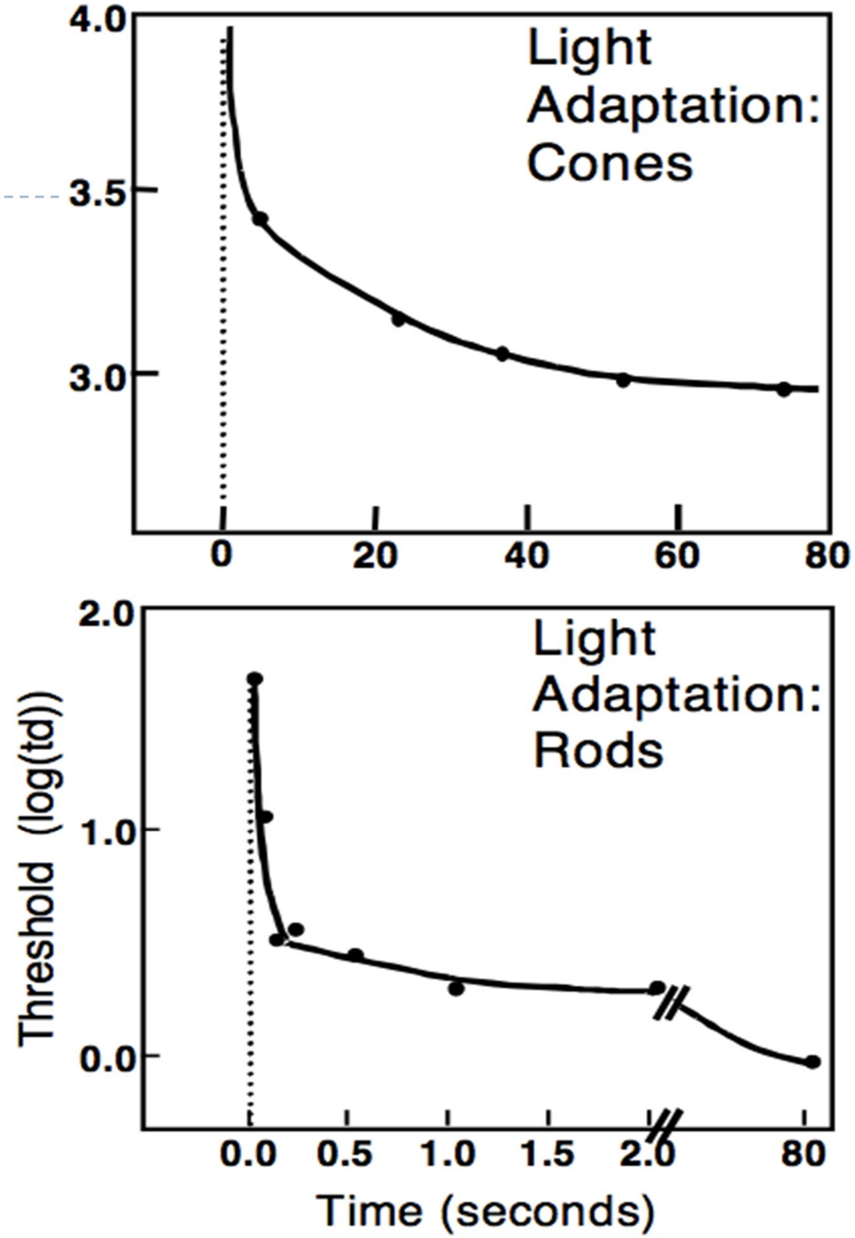


- Light adaptation: from dark to bright
- Dark adaptation: from bright to dark (much slower)

Time-course of adaptation



Bright -> Dark

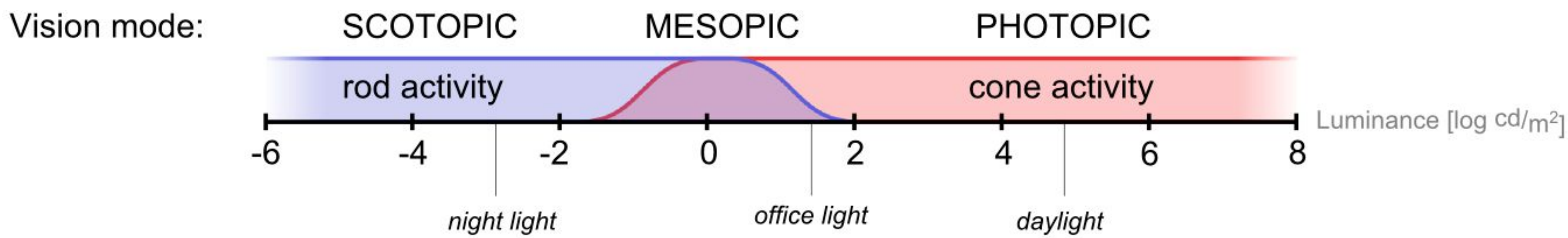


Dark -> Bright

Temporal adaptation mechanisms

- Bleaching & recovery of photopigment
  - Slow assymetric (light -> dark, dark -> light)
  - Reaction times (1-1000 sec)
  - Separate time-course for rods and cones
- Neural adaptation
  - Fast
  - Approx. symmetric reaction times (10-3000 ms)
- Pupil
  - Diameter varies between 3 and 8 mm
  - About 1:7 variation in retinal illumunation

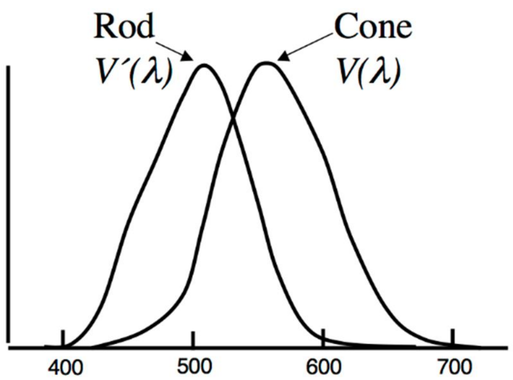
Night and daylight vision



Mode properties: monochromatic vision
limited visual acuity

good color perception
good visual acuity

Luminous efficiency



UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics and Image Processing

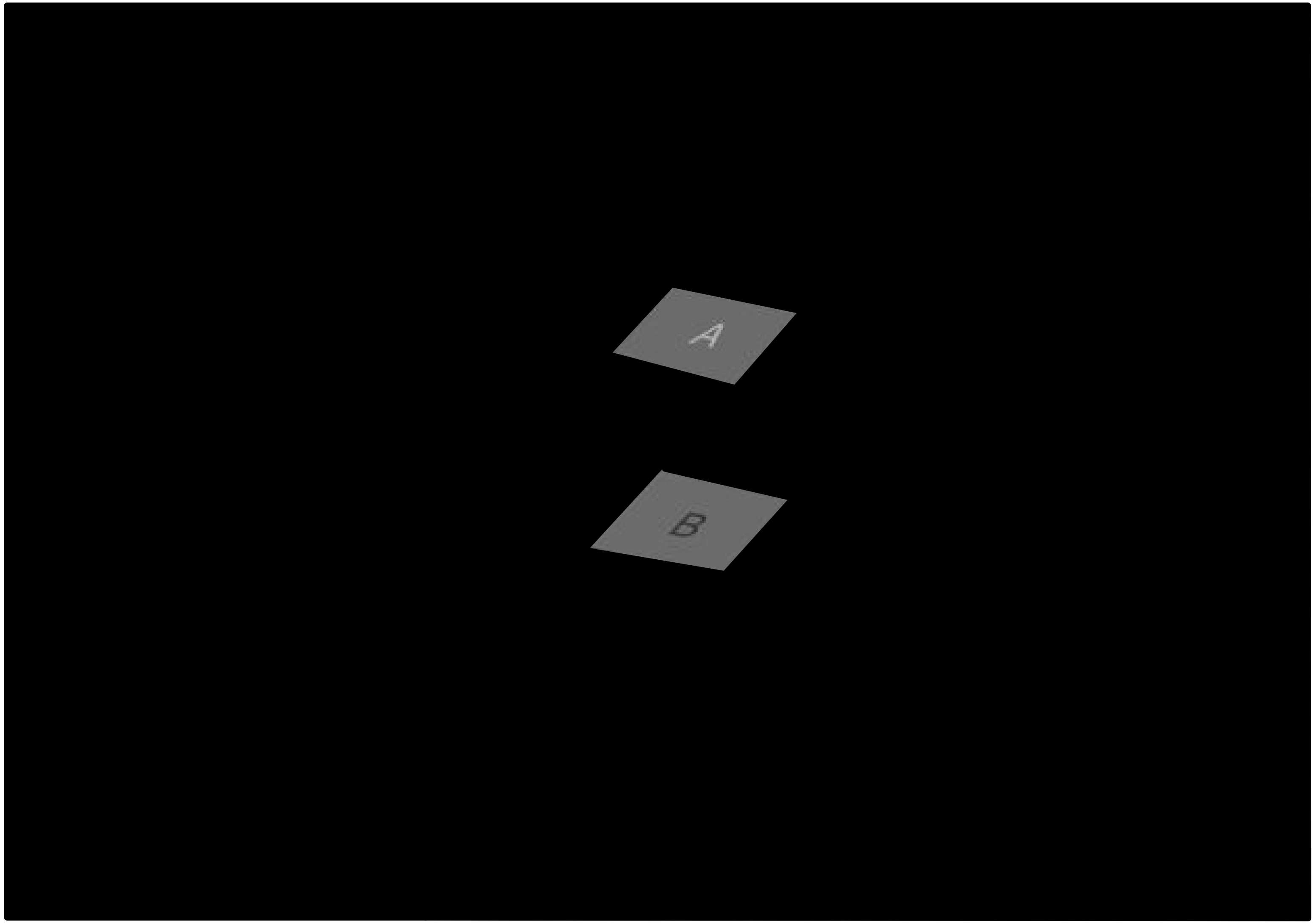
Models of early visual perception

Part 6/6 – high(er) level vision

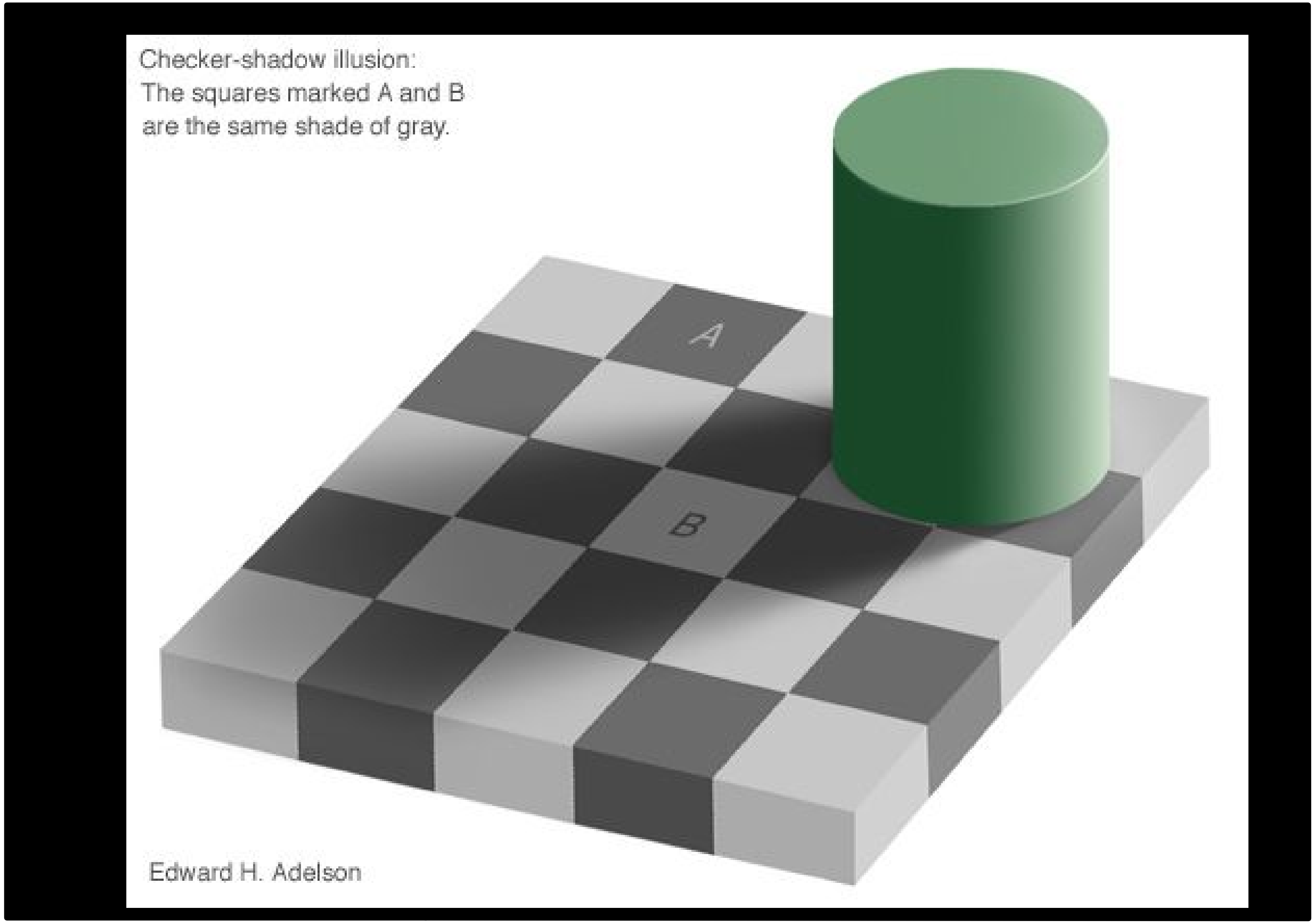
Rafal Mantiuk
Computer Laboratory, University of Cambridge

Simultaneous contrast

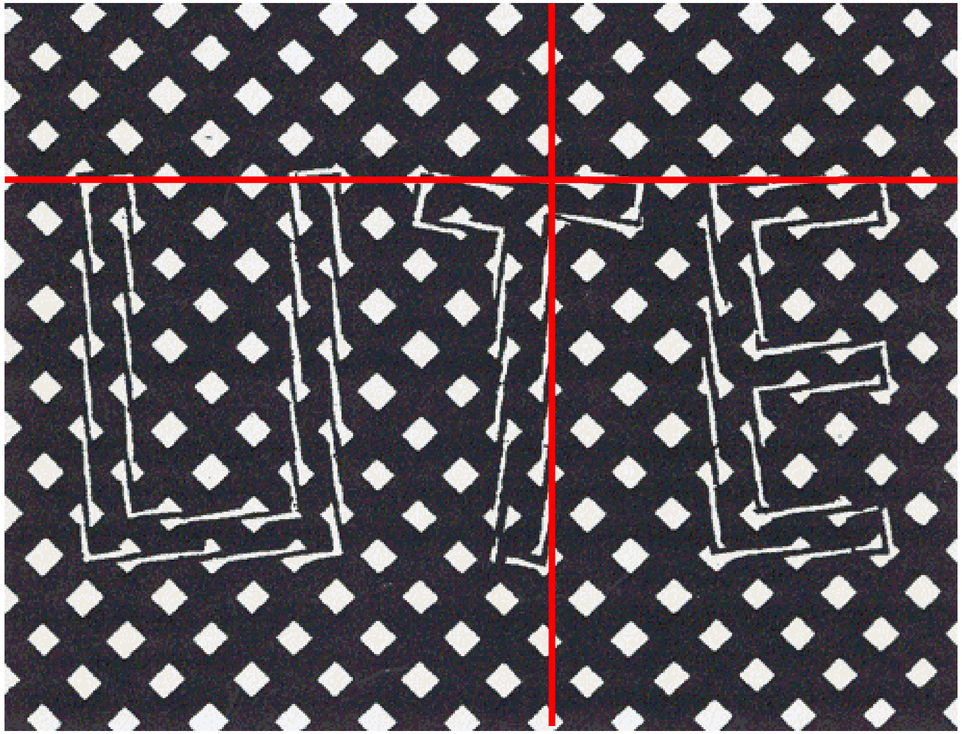
High-Level Contrast Processing



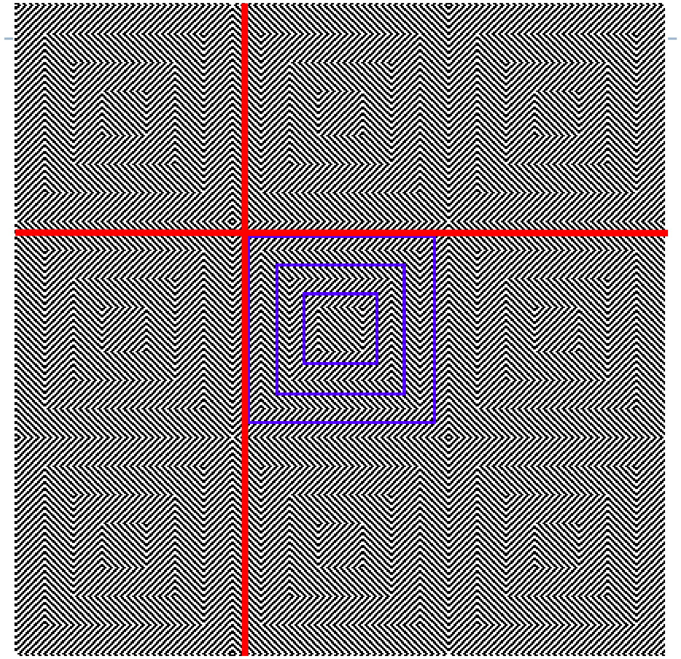
High-Level Contrast Processing

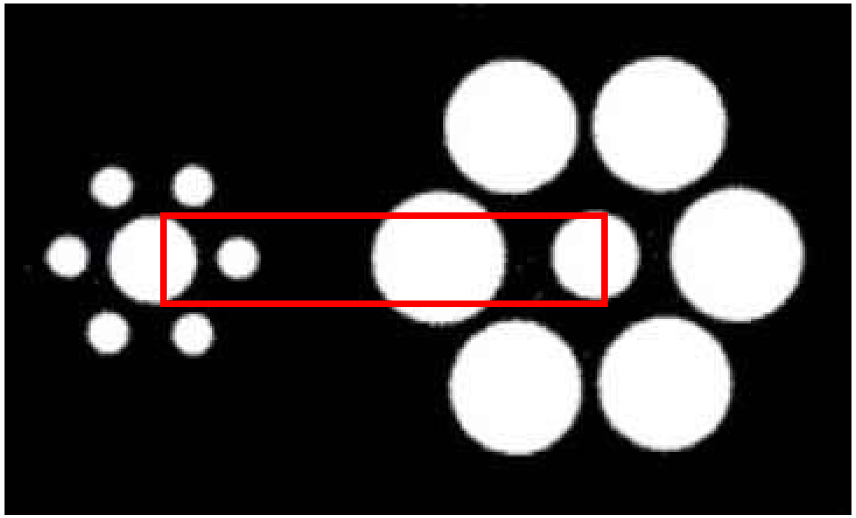


Shape Perception



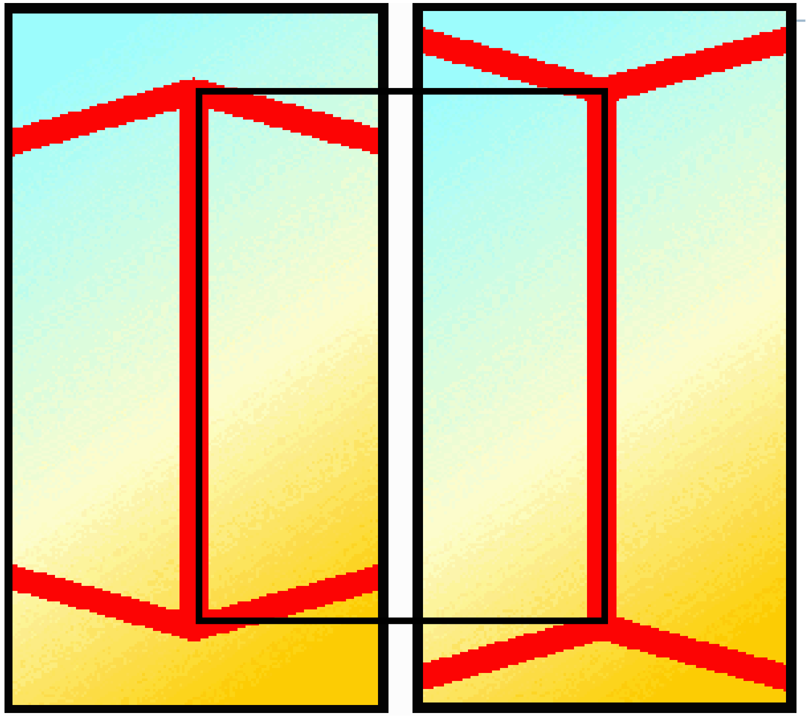
- Depends on surrounding primitives
  - Directional emphasis
  - Size emphasis

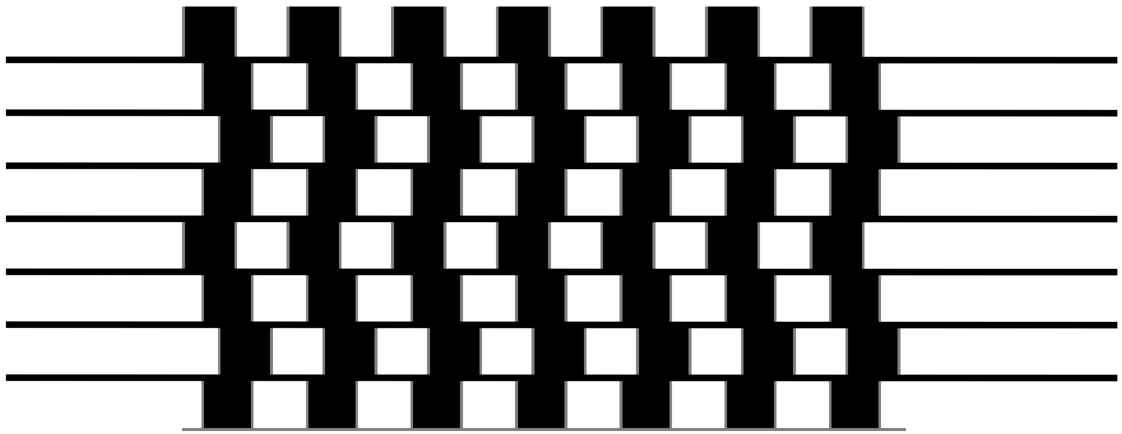




http://www.panoptikum.net/optischetaeuschungen/index.html

Shape Processing: Geometrical Clues



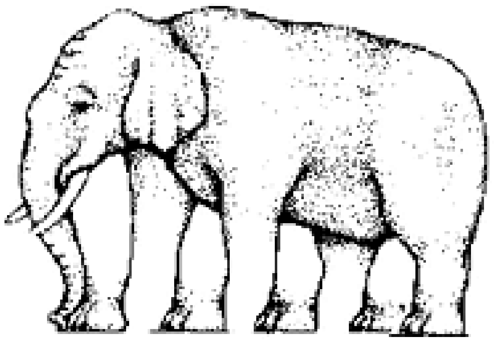


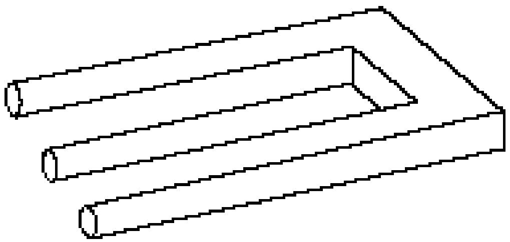
http://www.panoptikum.net/optischetaeuschungen/index.html

- Automatic geometrical interpretation
  - 3D perspective
  - Implicit scene depth

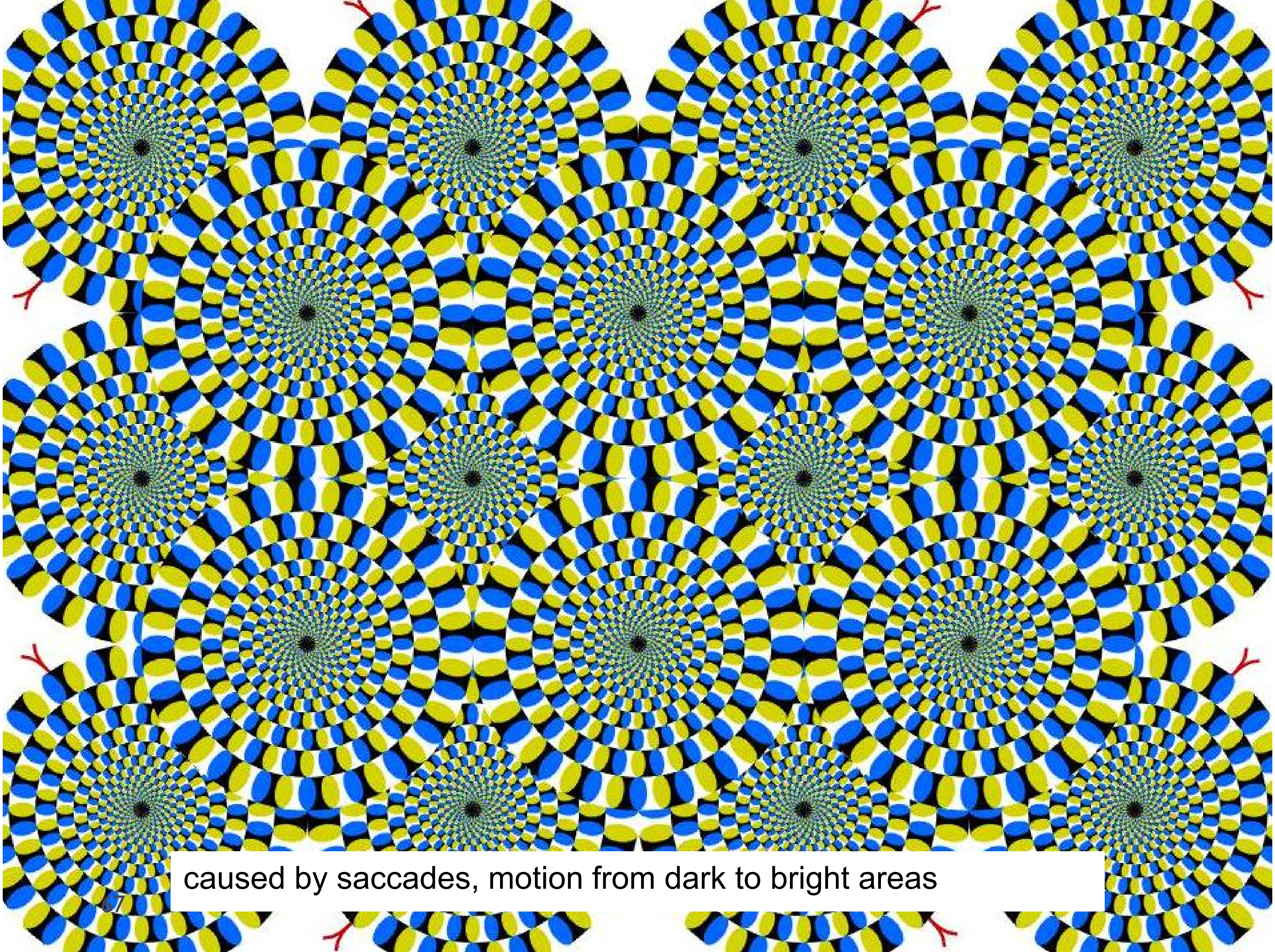
Impossible Scenes

- Escher et.al.
  - Confuse HVS by presenting contradicting visual clues
  - Local vs. global processing





http://www.panoptikum.net/optischetaeuschungen/index.html



caused by saccades, motion from dark to bright areas

Law of closure

References

- Wandell, B. A. (1995). *Foundations of vision*. Sinauer Associates.
  - Available online: https://foundationsofvision.stanford.edu/
- Mantiuk, R. K., Myszkowski, K., & Seidel, H. (2015). High Dynamic Range Imaging. In *Wiley Encyclopedia of Electrical and Electronics Engineering*. Wiley.
  - Section 2.4
  - Available online: http://www.cl.cam.ac.uk/~rkm38/hdri_book.html

Advanced Graphics & Image Processing

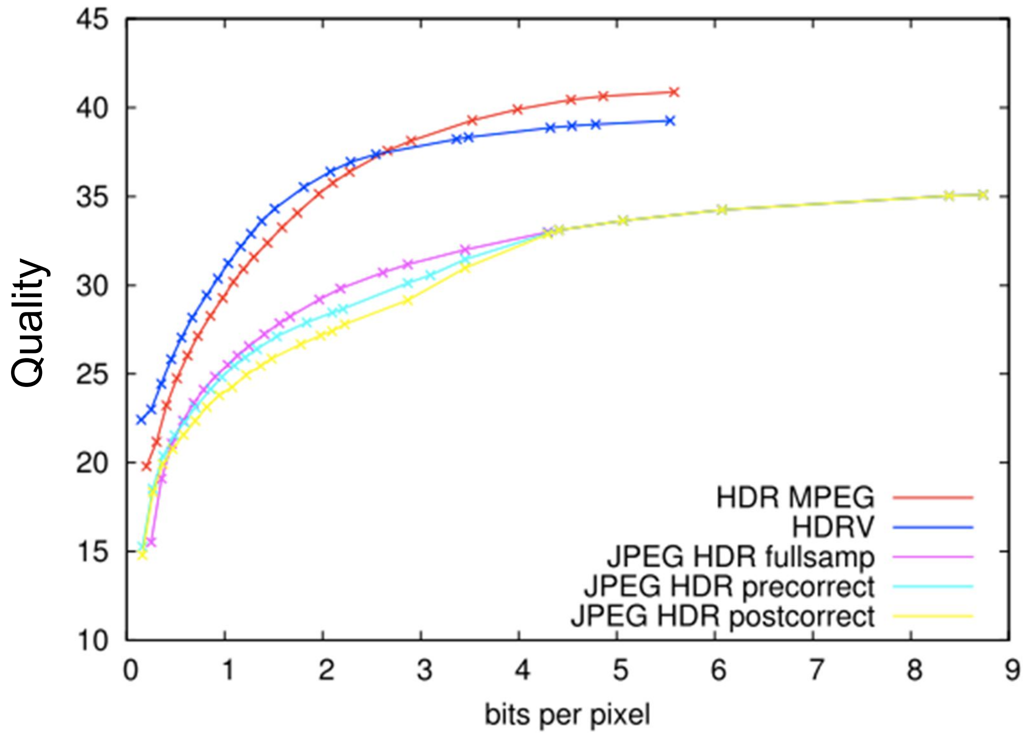
Assessing Image Quality

Rafał Mantiuk

Computer Laboratory, University of Cambridge

The purpose of image quality assessment

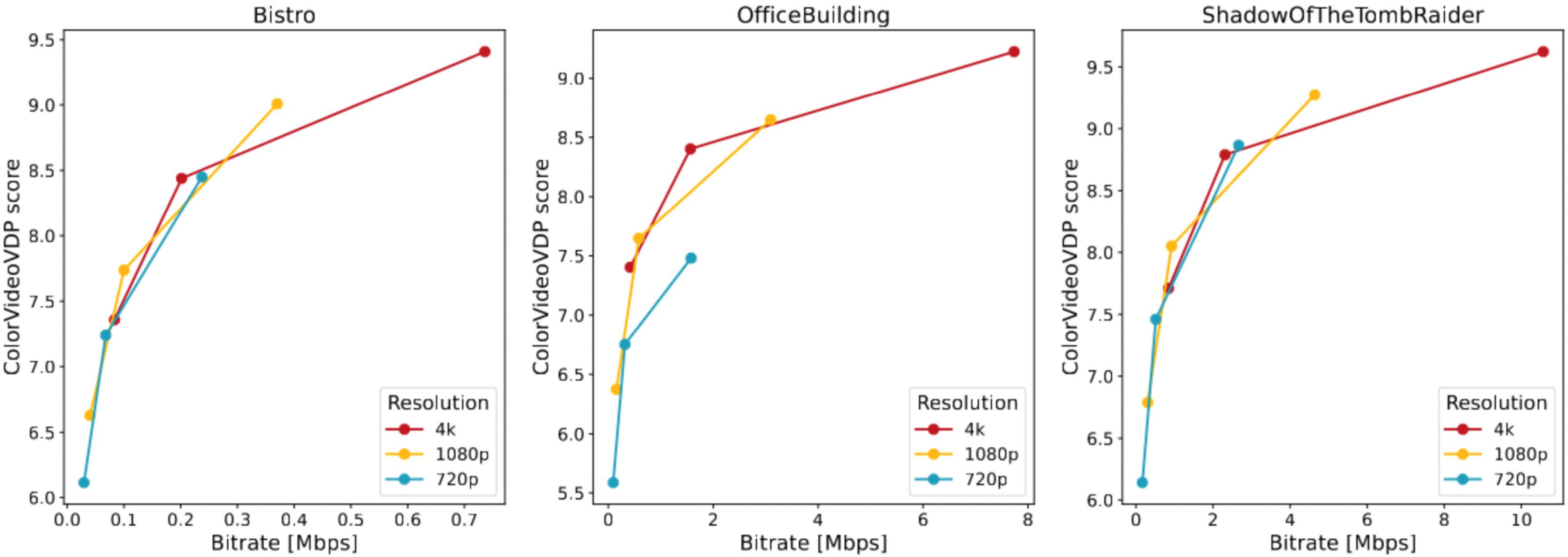
- To compare algorithms in terms of image or video quality



Rate-Distortion (RD) curves

The purpose of image quality assessment

- To optimize application parameters – e.g. resolution and bit-rate



The purpose of image quality assessment

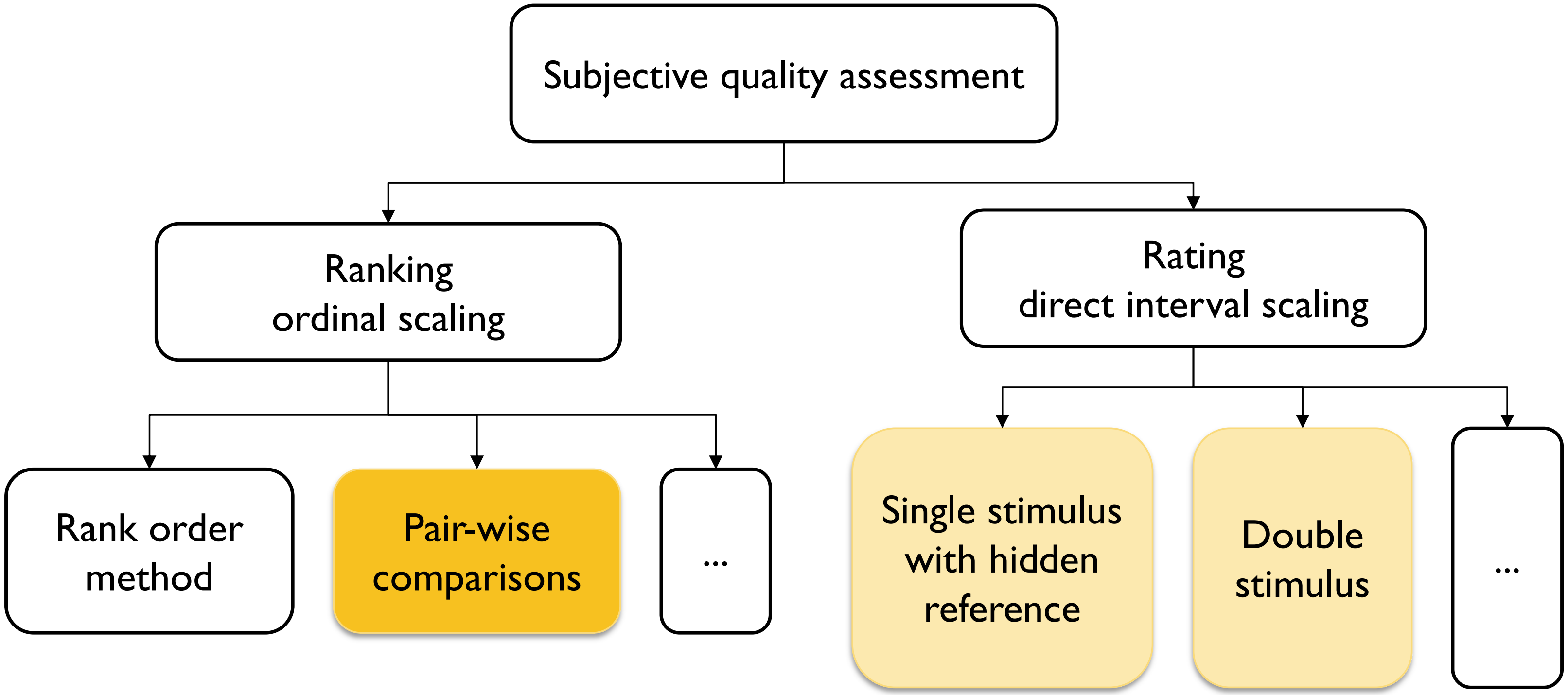
- To provide evidence of improvement over the state-of-the-art

Algorithm A Algorithm B Algorithm C

Other application domains

- Recommendation systems
  - Which movie to watch? (Netflix)
  - Which product to buy? (Amazon)
- Product acceptance / rating
  - Food
  - Clothing
  - Consumer electronics, …
- Similar techniques used for
  - Ranking of the players/gamers to match their skills in the game (TrueSkill on Xbox)

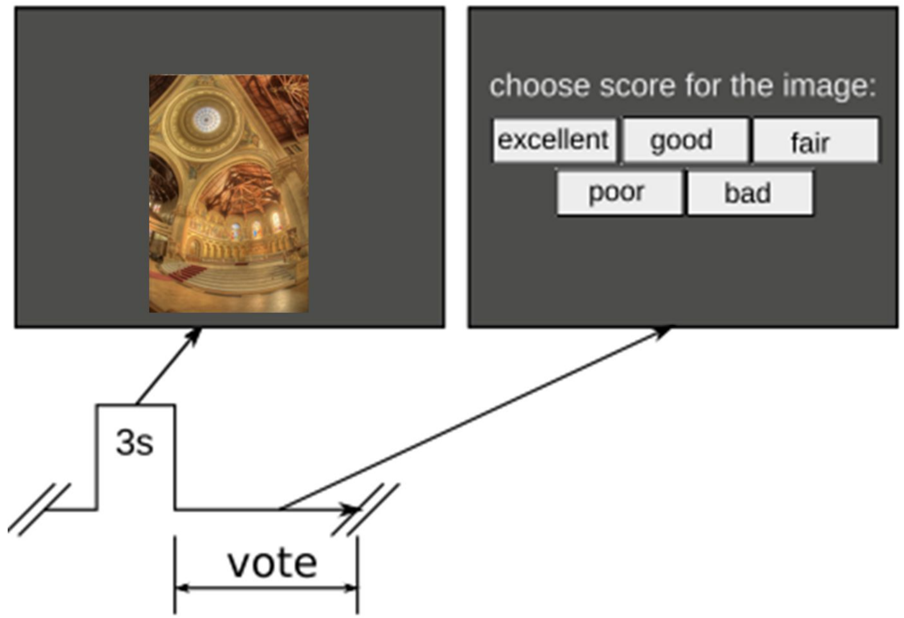
Subjective image/video quality assessment methods



Rating: Single stimulus + hidden reference

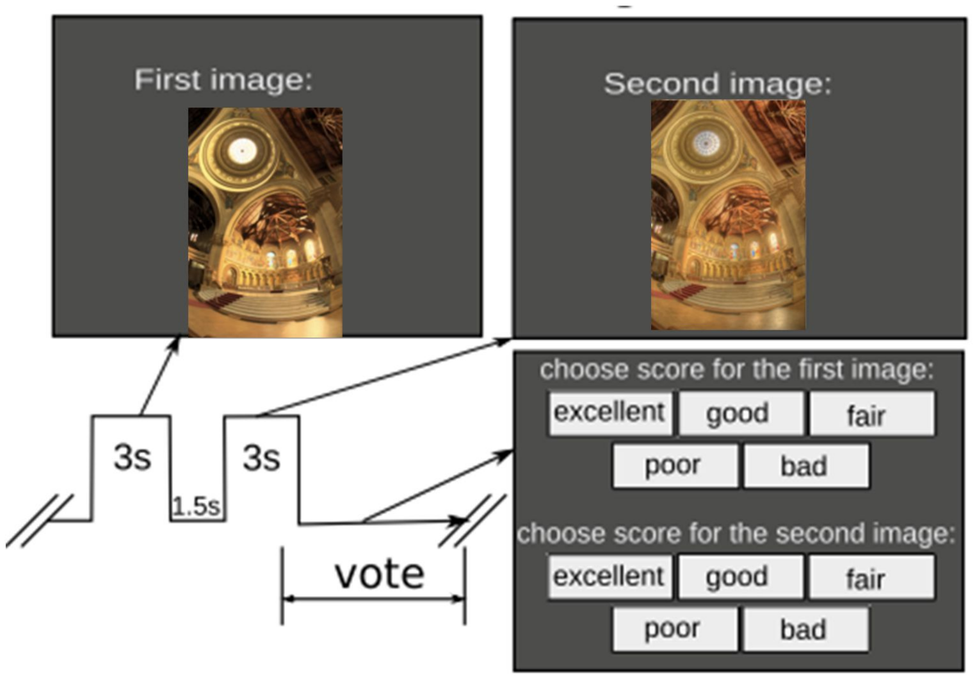
- With a hidden reference
- Task: **Rate** the quality of the image
- The categorical variables (excellent, good, …) are converted into scores 1-5
- Then those are averaged across all observers to get Mean-Opinion-Scores (MOS)
- To remove the effect of reference content, we often calculate DMOS:

$$Q_{DMOS} = Q_{MOS}^{reference} - Q_{MOS}^{test}$$



Rating: Double stimulus

- Task: Rate the quality of the first and the second image
- The second image is typically the reference
- Potentially better accuracy of DMOS
- But takes more time
  - The reference shown after each test image



Pair-wise comparison method

- Example: video quality
- Task: Select the video sequence that has a higher quality

Comparison matrix

- Results of pairwise comparisons can be stored in a comparison matrix

$$\begin{array}{cc} & \begin{array}{ccc} C1 & C2 & C3 \end{array} \\ C = & \begin{bmatrix} 0 & 3 & 1 \\ 3 & 0 & 2 \\ 5 & 4 & 0 \end{bmatrix} \end{array} \begin{array}{c} C1 \\ C2 \\ C3 \end{array}$$

- In this example: 3 compared conditions: C1, C2, C3
- $C_{ij} = n$ means that condition Ci was preferred over Cj n times

Full and reduced designs

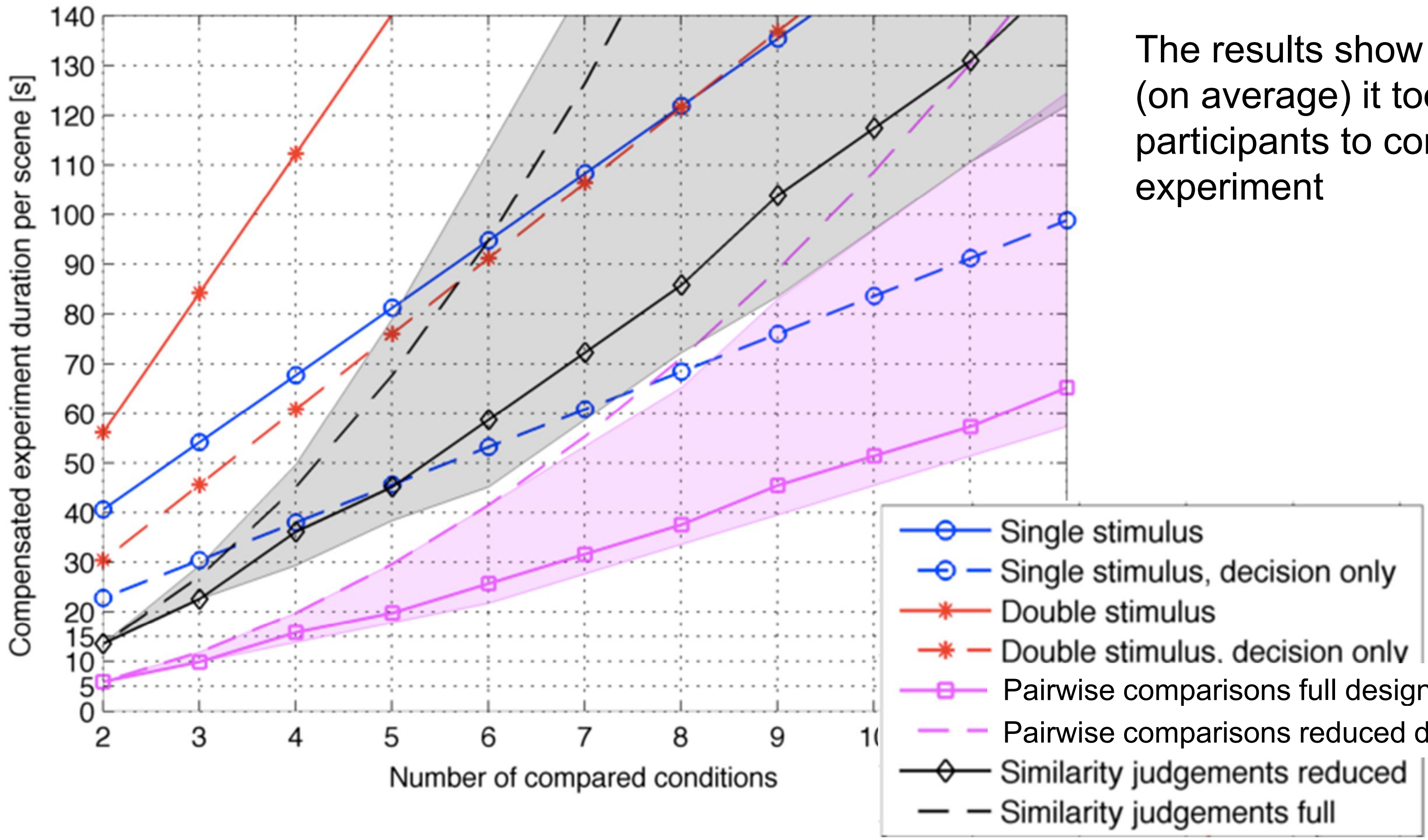
- Full design
  - Compare all pairs of conditions
  - This requires $\binom{n}{2} = \frac{n(n-1)}{2}$ comparisons for n conditions
  - Tedious if n is large
- Reduced design
  - We assume transitivity
    - If C1 > C2 and C2 > C3 then C1 > C3
      - no need to do all comparisons
  - There are numerous "block designs" (before computers)
  - But the task is also a sorting problem
    - The number comparison can be reduced to $n\log(n)$ for a "human quick-sort"
  - And many others: Swiss chess system, active sampling ...

$$C = \begin{array}{cc} & \begin{array}{ccc} C1 & C2 & C3 \end{array} \\ & \begin{bmatrix} 0 & 3 & 1 \\ 3 & 0 & 2 \\ 5 & 4 & 0 \end{bmatrix} \begin{array}{c} C1 \\ C2 \\ C3 \end{array} \end{array}$$

Pairwise comparisons vs. rating (e.g. single stimulus)

- The method of pairwise comparisons is **fast**
  - More comparisons, but
  - It takes less time to achieve the same sensitivity as for direct rating methods
- Has a higher sensitivity
  - Less "external" variance between and within observers
- Provides a unified quality scale
  - The scale (of JOD/JND) is transferrable between experiments
- Simple procedure
  - Training is much easier
  - Less affected by learnining effects
- Especially suitable for non-expert participants
  - E.g. Crowdsourcing experiments

Time-efficiency



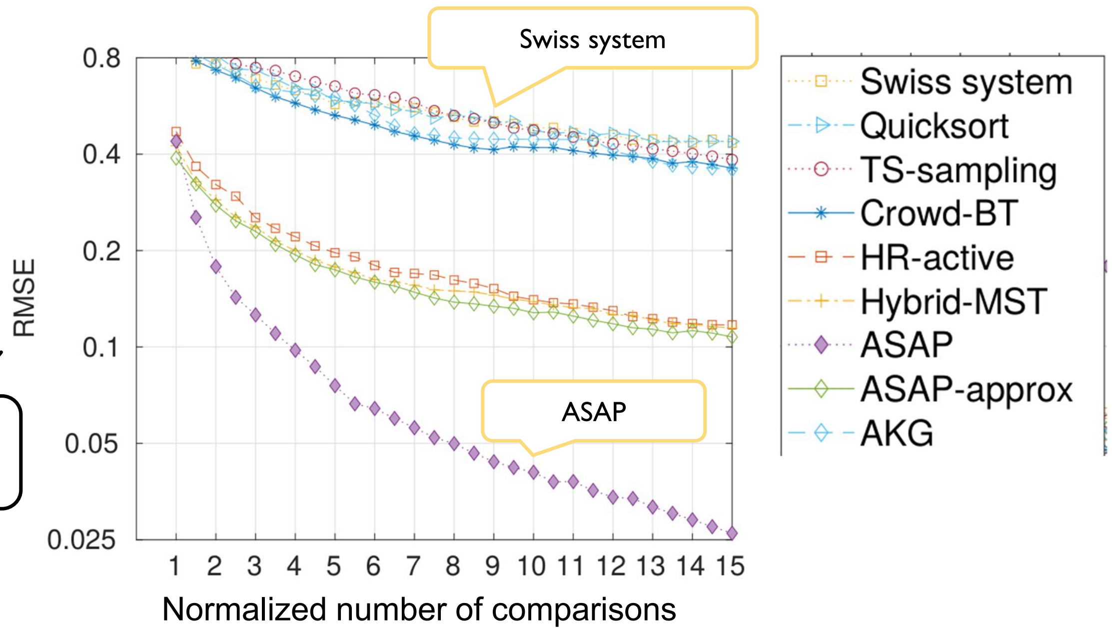
The results show how long (on average) it took participants to complete the experiment

From: Mantiuk et al. CGF 2012

Active sampling can make the experiments even faster

- Active sampling
  - For each trial, select a pair of conditions that maximizes the information gain
  - Information gain is the DK-divergence between the prior and posterior distributions

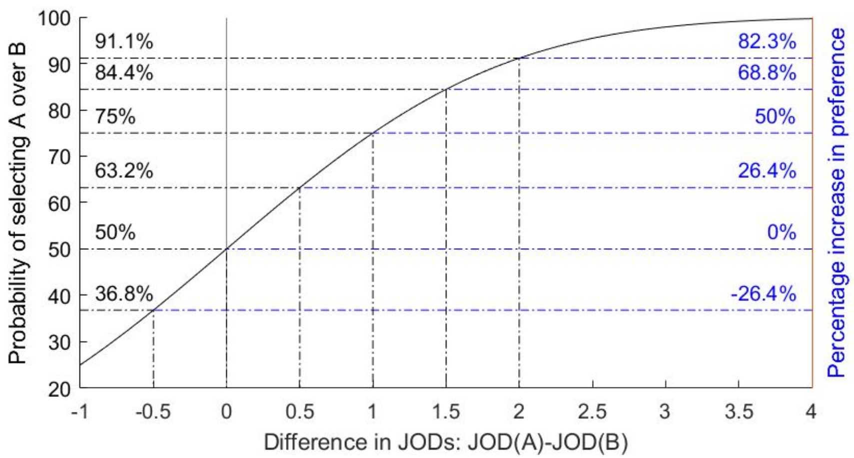
Estimation error

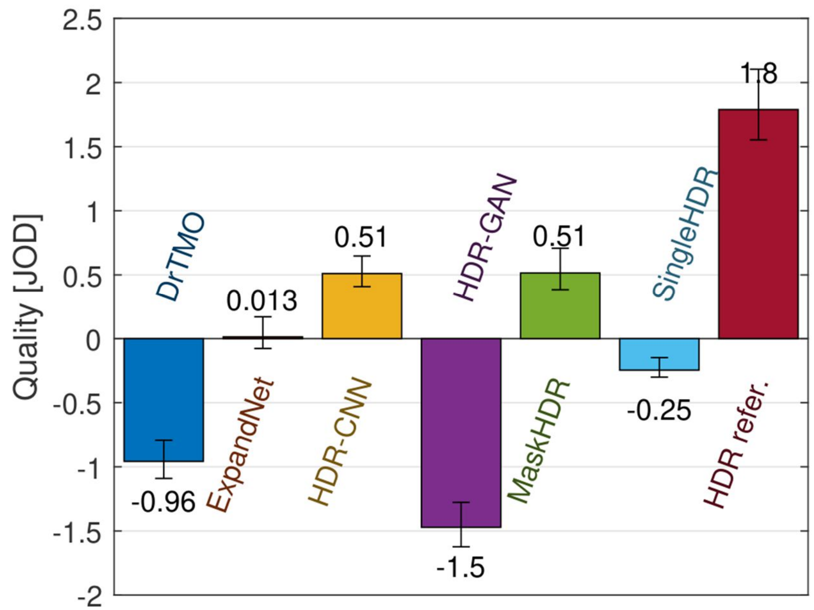


- Mikhailiuk, A., C. Wilmot, M. Perez-Ortiz, D. Yue, and R.K. Mantiuk. "ASAP: Active Sampling for Pairwise Comparisons via Approximate Message Passing and Information Gain Maximization." In *International Conference on Patter Recognition*, 2020.

Practical significance - scaling

- Scaling: to map user judgments into meaningful interval scale
- Typically that scale is in just-noticeable-difference units
  - The difference of 1 JND means that 75% of observers would choose one condition over another
  - Useful to show "practical" significance





Scaling pairwise comparison data

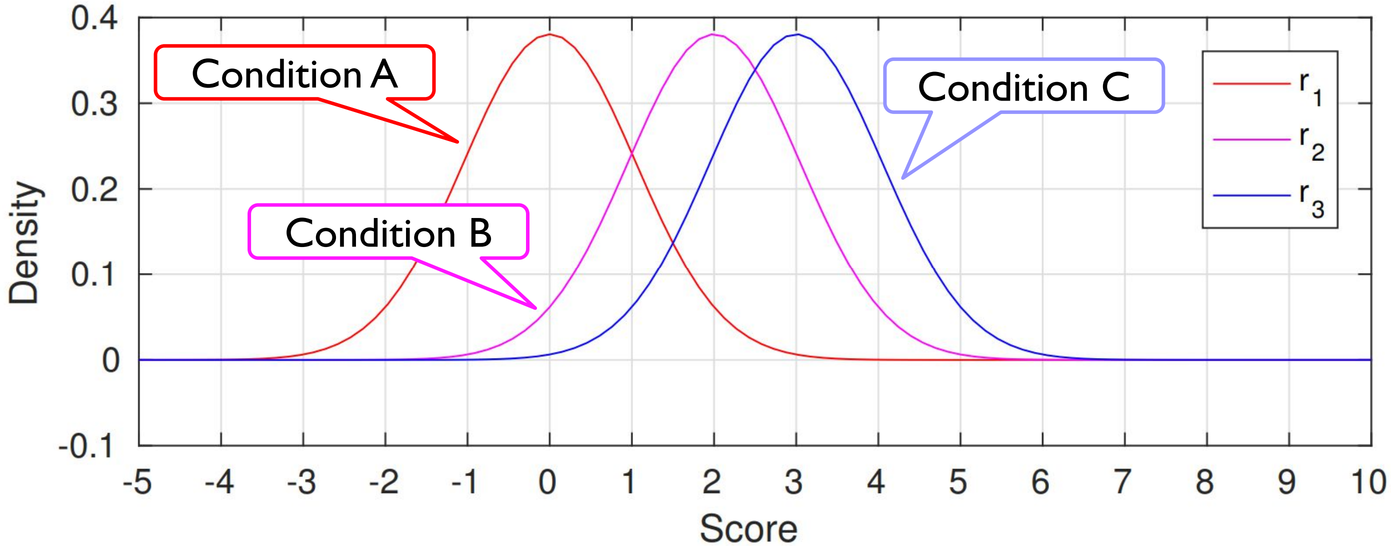
- Given a matrix of comparisons, for example

$$\mathbf{C} = \begin{bmatrix} 0 & 3 & 0 \\ 27 & 0 & 7 \\ 30 & 23 & 0 \end{bmatrix}$$

- Infer the quality scores for all compared conditions
  - Using Maximum Likelihood Estimation (MLE)
- We start from an observer model, then link it to the observations

Thurstone (observer) model - Case V

- Two assumptions:
  - Quality scores for a given condition are normally distributed across the population
  - The variance of that distribution is the same for each condition and the judgements are independent



From the observer model to probabilities

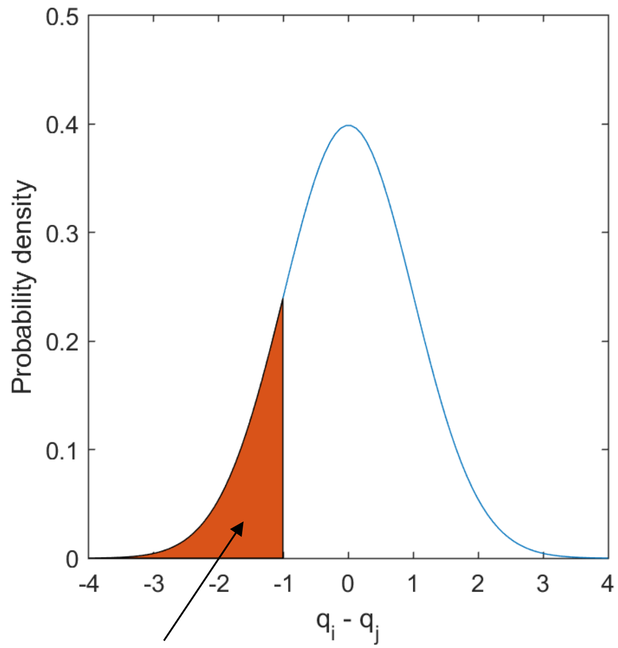
- Given the observer model for two conditions:

$$r_i = N(q_i, \sigma^2) \qquad r_j = N(q_j, \sigma^2)$$

- The difference between two quality scores is:

$$r_i - r_j = N(q_i - q_j, 2\sigma^2)$$

- Then, the probability of the judgment is explained by the cumulative normal distribution



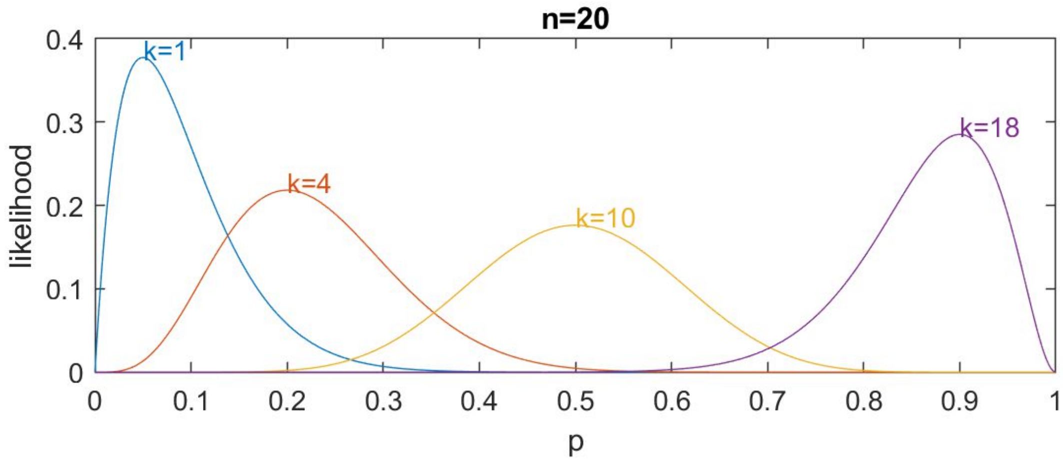
$P(r_i > r_j | q_i - q_j = -1)$

$$P(r_i > r_j) \quad = P(r_i - r_j > 0) = \Phi\left(\frac{q_i - q_j}{\sigma_{ij}}\right)$$

$$= \frac{1}{\sigma_{ij}\sqrt{2\,\pi}} \int_{-\infty}^{q_i - q_j} e^{\left(\frac{-x^2}{2\,\sigma_{ij}{}^2}\right)} dx \,. \quad \text{where } \sigma_{ij} = \sqrt{2}\sigma$$

Binomial distribution

- Given that *k* out of *n* observers selected A over B, what is the probability distribution *of* selecting A over B



$$P(r_i > r_j | n, k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Maximum Likelihood Estimation

- Given our observations (comparison matrix) what is the likelihood of the quality values q_i:

$$L(\hat{q}_i - \hat{q}_j | c_{ij}, n_{ij}) = \binom{n_{ij}}{c_{ij}} P(r_i > r_j)^{c_{ij}} \left(1 - P(r_i > r_j)\right)^{n_{ij} - c_{ij}}$$

$$= \binom{n_{ij}}{c_{ij}} \Phi\left(\frac{\hat{q}_i - \hat{q}_j}{\sigma_{ij}}\right)^{c_{ij}} \left(1 - \Phi\left(\frac{\hat{q}_i - \hat{q}_j}{\sigma_{ij}}\right)\right)^{n_{ij} - c_{ij}}$$

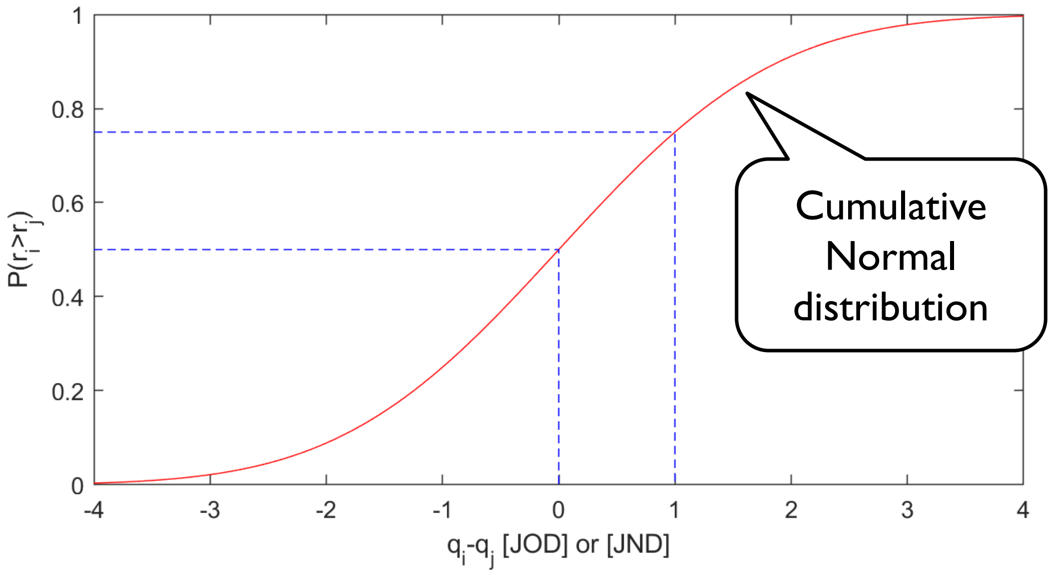
Cumulative Normal

  - where $n_{ij} = c_{ij} + c_{ji}$
- To estimate the values of q_i, we maximize:

$$\arg\max_{\hat{q}_2, \ldots, \hat{q}_n} \prod_{i,j \in \Omega} L(\hat{q}_i - \hat{q}_j | c_{ij}, n_{ij})$$

JND/JOD = 1

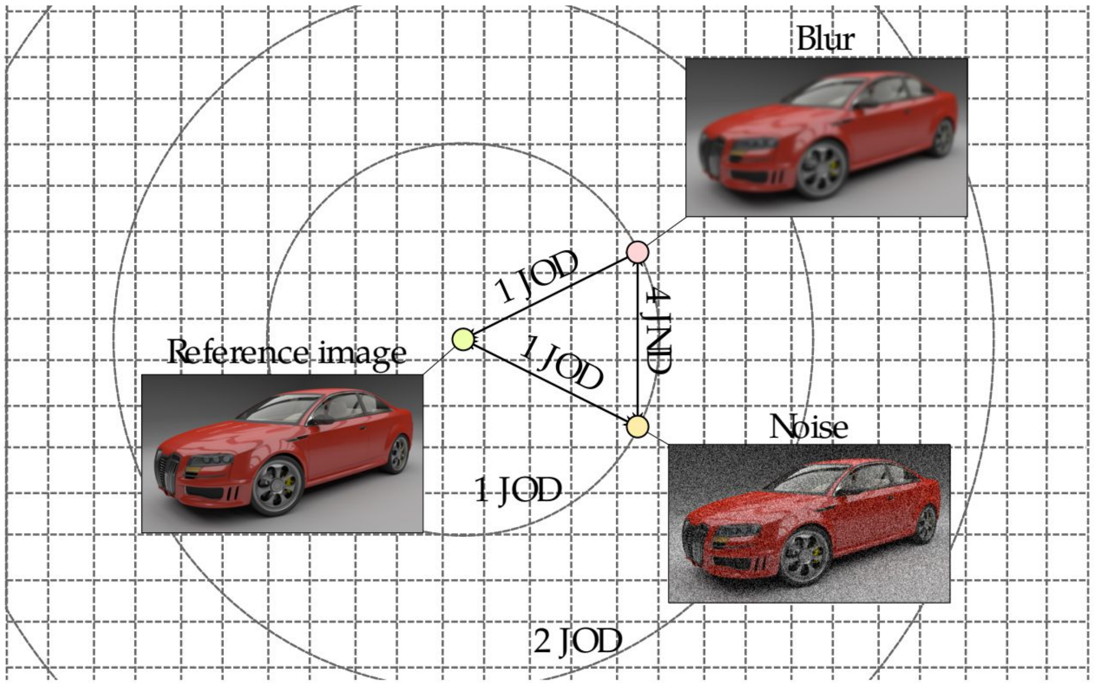
- Just Noticeable Differences
- Just Objectionable Differences
- We want $q_i - q_j = 1$ when 75% of observers prefer condition "i" over "j"



  - This happens when $\sigma_{ij} = 1.4826$
  - This is arbitrary selected scaling, made for easier interpretation of the results

JND vs JOD

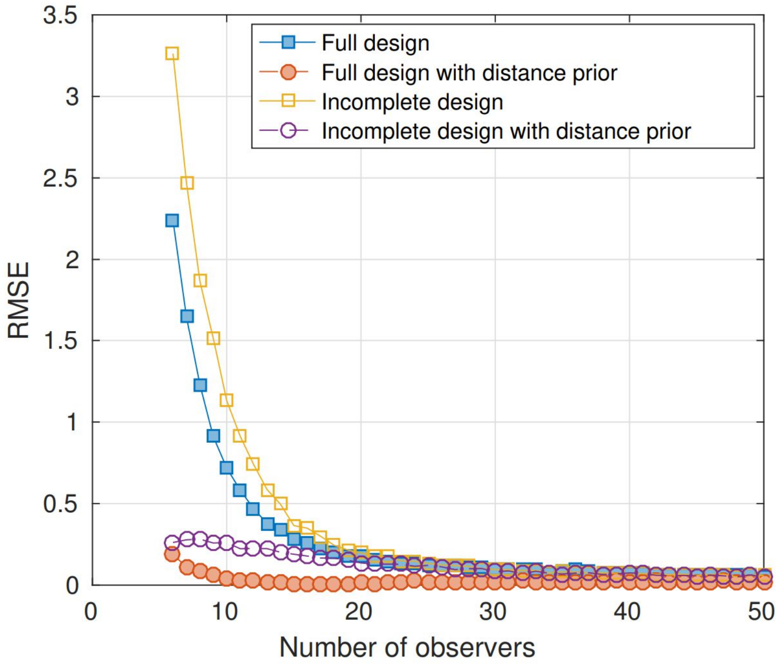
- Just Noticeable Differences
- Just Objectionable Differences

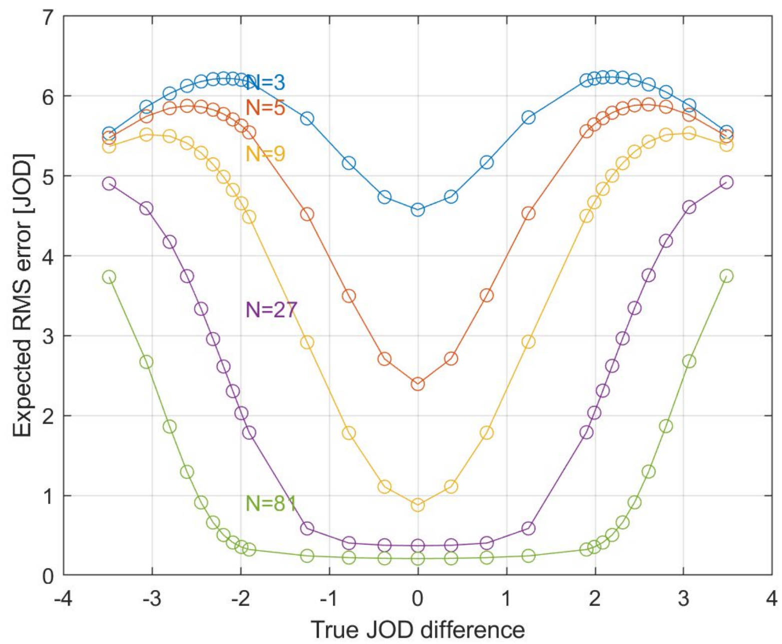


- JND – is one visually different from another
- JOD – is the **quality** of one different from the quality of another (relative to the reference)

Practicalities of MLE scaling

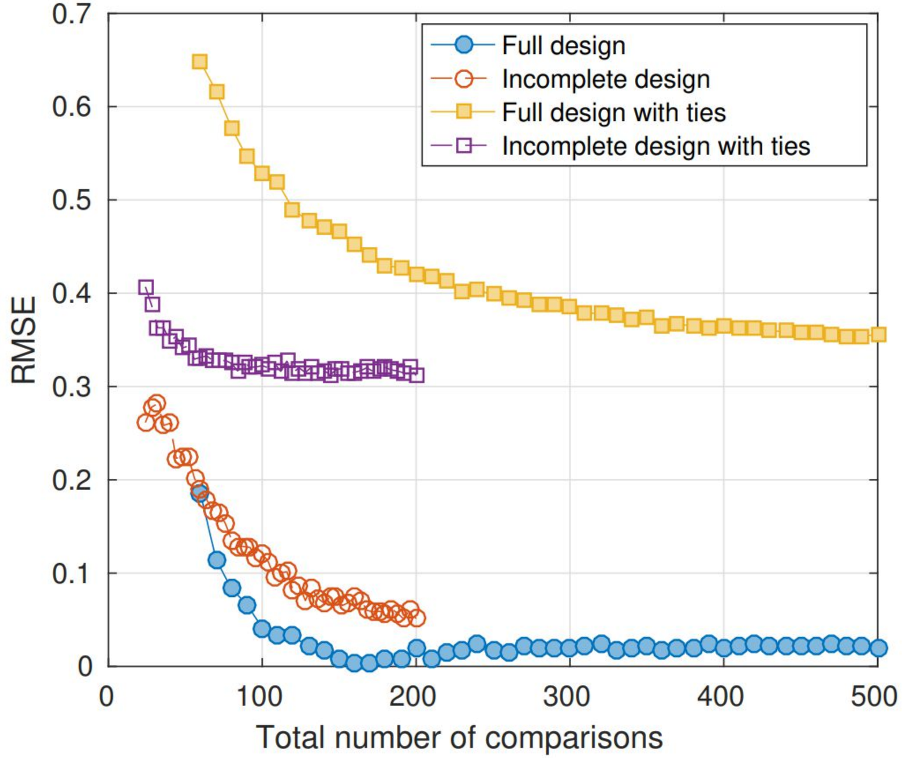
- At least 15-20 comparisons per each pair are needed to obtain stable results (prior helps)





Forced choice vs. comparison with ties

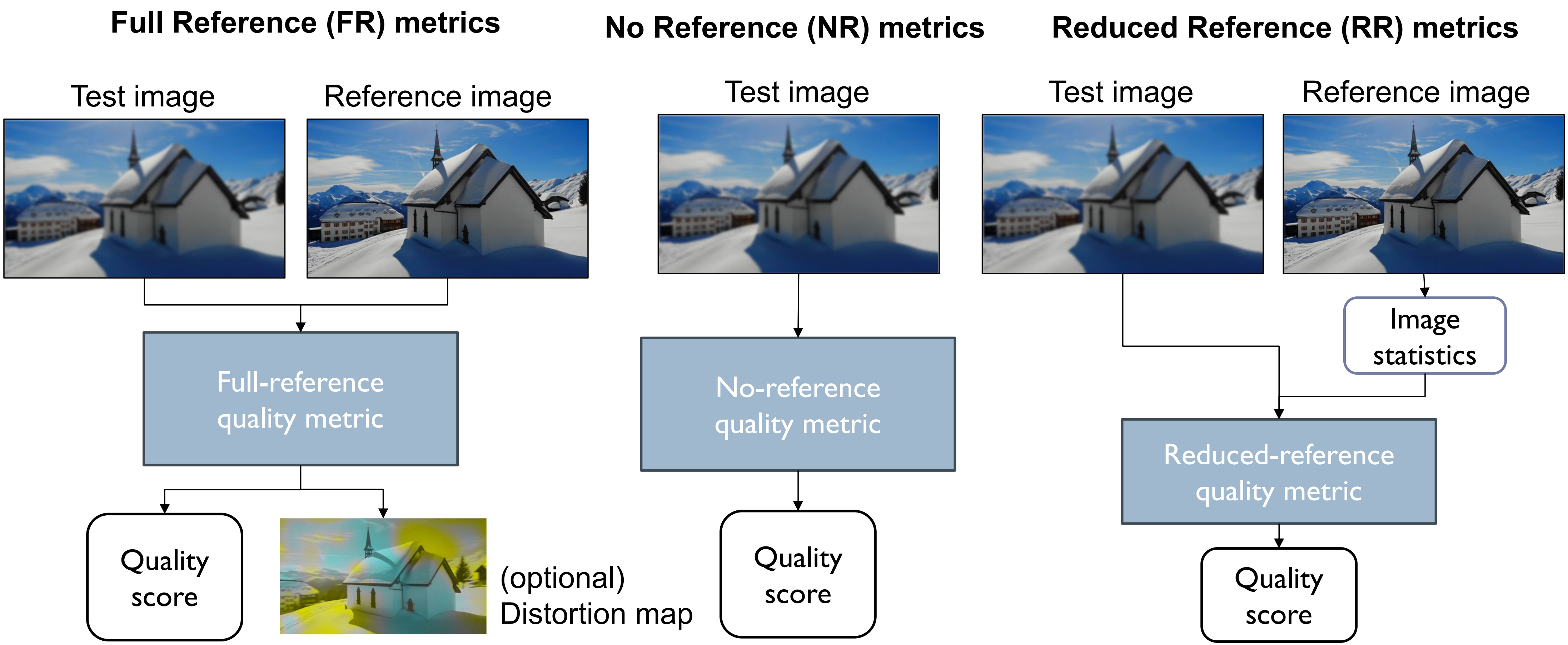
- Giving a "tie" option is usually a bad idea



- Scaling the results with ties requires a more complex observer model with more parameters to estimate

Objective (image/video) quality metrics

Types of objective (image/video) quality metrics



Main use cases of objective quality metrics

(I) Evaluation

Which method is the best?

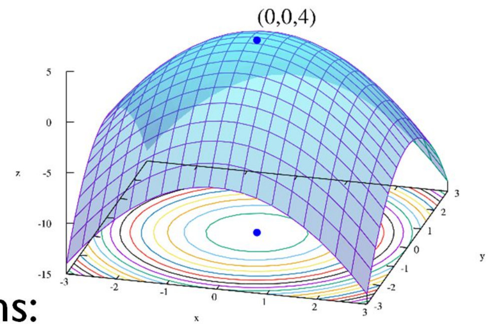
Dataset	Scale	Bicubic	A+ [27]	SRCNN [4]	VDSR [11]
Set5	×2	33.66 / 0.9299	36.54 / 0.9544	36.66 / 0.9542	37.53 / 0.9587
	×3	30.39 / 0.8682	32.58 / 0.9088	32.75 / 0.9090	33.66 / 0.9213
	×4	28.42 / 0.8104	30.28 / 0.8603	30.48 / 0.8628	31.35 / 0.8838
Set14	×2	30.24 / 0.8688	32.28 / 0.9056	32.42 / 0.9063	33.03 / 0.9124
	×3	27.55 / 0.7742	29.13 / 0.8188	29.28 / 0.8209	29.77 / 0.8314
	×4	26.00 / 0.7027	27.32 / 0.7491	27.49 / 0.7503	28.01 / 0.7674
B100	×2	29.56 / 0.8431	31.21 / 0.8863	31.36 / 0.8879	31.90 / 0.8960
	×3	27.21 / 0.7385	28.29 / 0.7835	28.41 / 0.7863	28.82 / 0.7976
	×4	25.96 / 0.6675	26.82 / 0.7087	26.90 / 0.7101	27.29 / 0.7251
Urban100	×2	26.88 / 0.8403	29.20 / 0.8938	29.50 / 0.8946	30.76 / 0.9140
	×3	24.46 / 0.7349	26.03 / 0.7973	26.24 / 0.7989	27.14 / 0.8279
	×4	23.14 / 0.6577	24.32 / 0.7183	24.52 / 0.7221	25.18 / 0.7524

Aims:

- To demonstrate the difference in quality
- To replace subjective experiments

(II) Optimization

What are the best parameter values?



Aims:

- To replace manual parameter tweaking
- Especially in multi-dimensional problems

Pixel-wise quality metrics

- Root Mean Square Error (RMSE)

$$E_{RMSE} = \sqrt{\frac{1}{w \cdot h} \sum_{x,y} \left(t(x,y) - r(x,y)\right)^2}$$

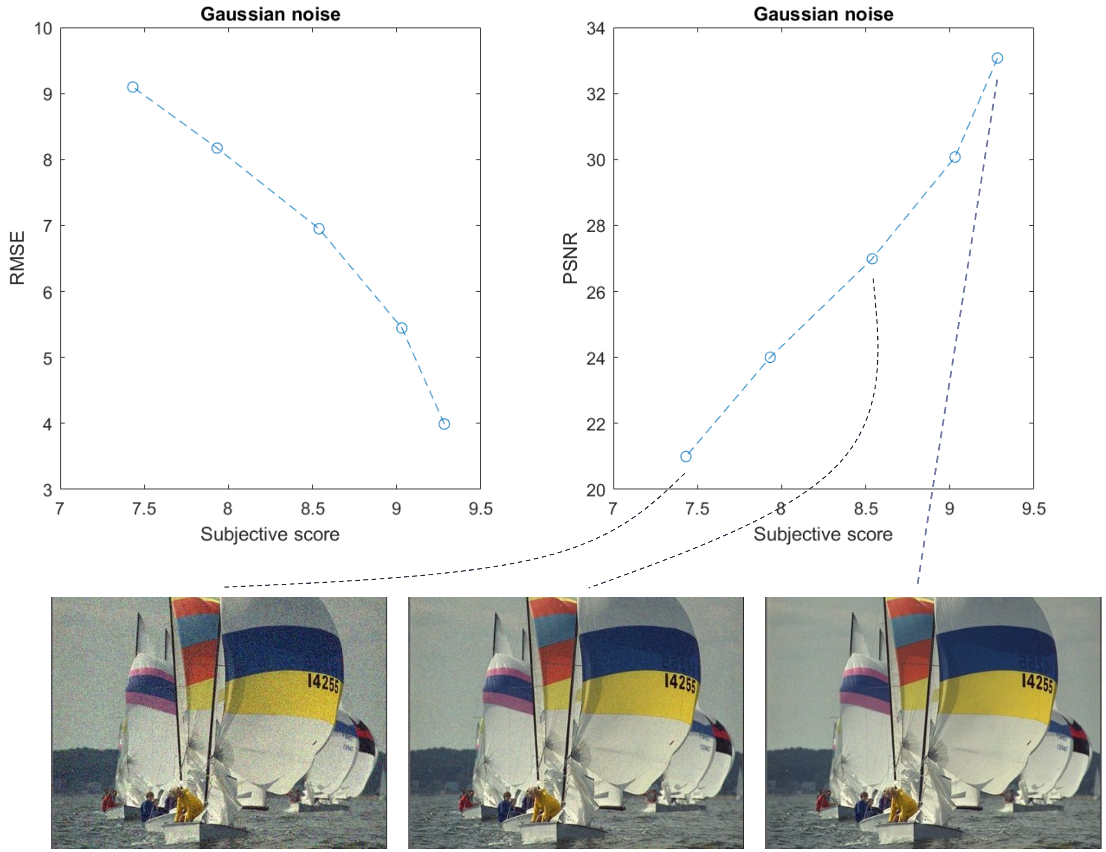
Test image

Reference image

- Peak Signal to Noise Ratio

$$E_{PSNR} = 20 \frac{I_{peak}}{E_{RMSE}} \; [dB]$$

  - I_{peak} - the peak pixel value (e.g. 255 or 1)
  - If the error is normally distributed and its mean is 0, E_{RMSE} is the standard deviation of the distortion (noise)



The shortcomings of pixel-wise metrics

Reference

JPEG-encoded
PSNR=24.7

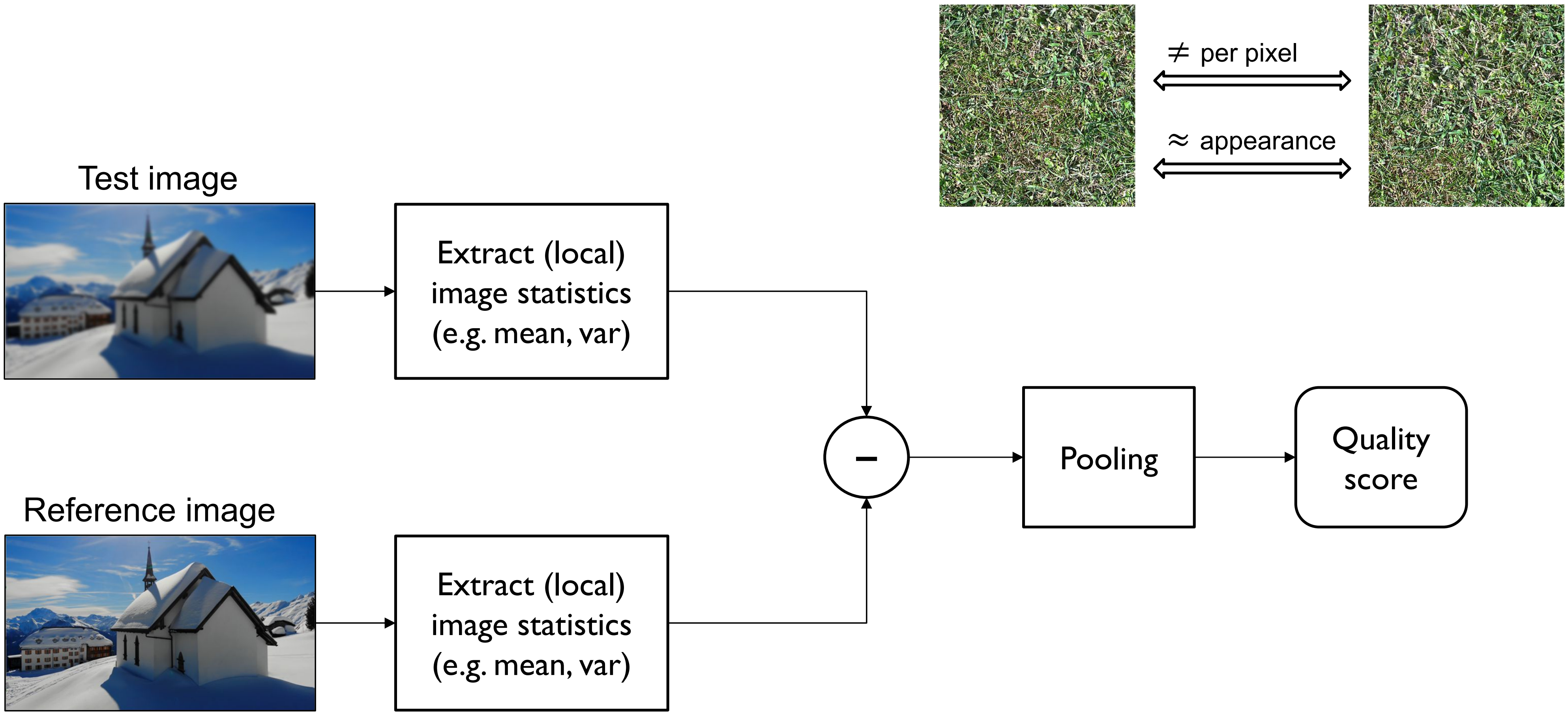
Blur
PSNR=24.8

Noise
PSNR=24.8

Rotation (1.3 deg)
PSNR=23.4

[Examples from: 10.1109/TIP.2008.926161]

Texture quality metrics

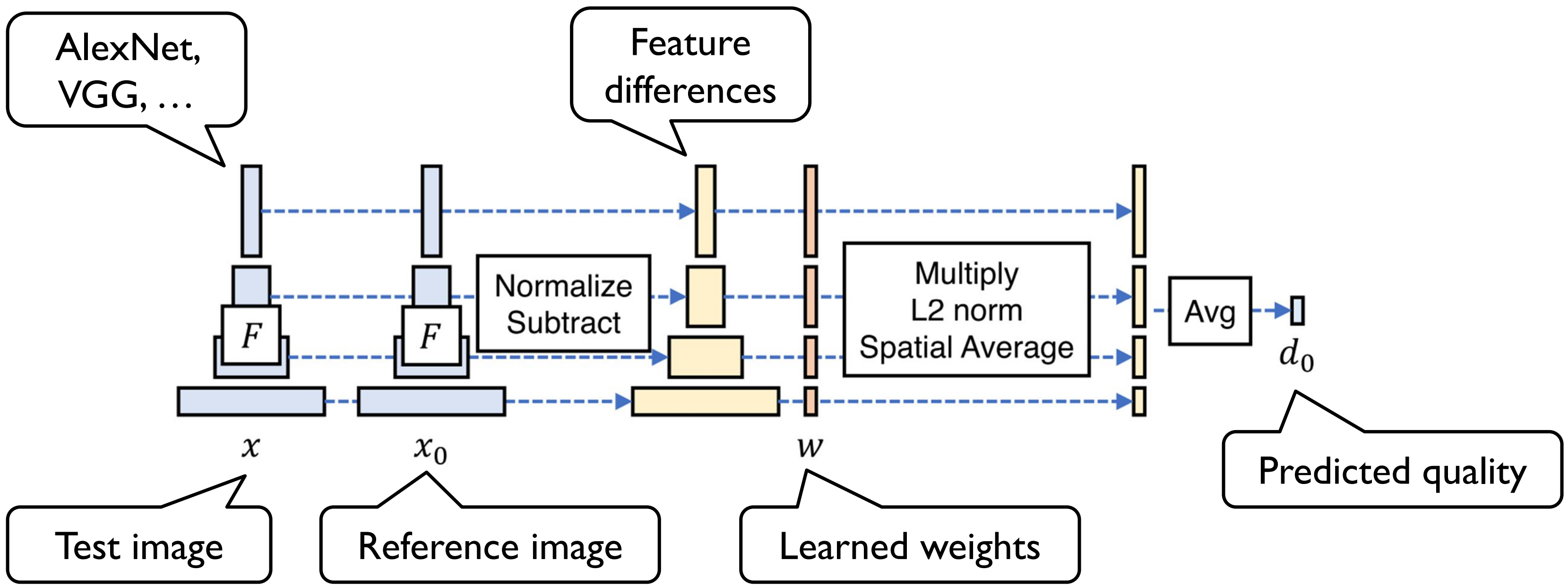


Structural Similarity Index (SSIM)

- Split test and reference images into 11×11 px overlapping patches
- For each patch, calculate mean μ_T, μ_R, std $\sigma_T \sigma_R$ and covariance σ_{TR}
  - of each patch, weighted by a Gaussian window
- Calculate three terms (per patch)
  - "Luminance": $\mathrm{l}_{\boldsymbol{x}} = \frac{2\mu_T\mu_R + C_0}{\mu_T^2 + \mu_R^2 + C_0}$
  - Contrast: $c_{\boldsymbol{x}} = \frac{2\sigma_T\sigma_R + C_1}{\sigma_T^2 + \sigma_R^2 + C_1}$
  - Structure: $s_{\boldsymbol{x}} = \frac{\sigma_{TR} + C_2}{\sigma_T\sigma_R + C_2}$ (cross-correlation)
- Multiply them together: $q_{\boldsymbol{x}} = l_{\boldsymbol{x}} \cdot \mathrm{c}_{\boldsymbol{x}} \cdot \mathrm{s}_{\boldsymbol{x}}$
- And pool: $q_{SSIM} = \frac{1}{N}\sum_{\boldsymbol{x}} q_{\boldsymbol{x}}$

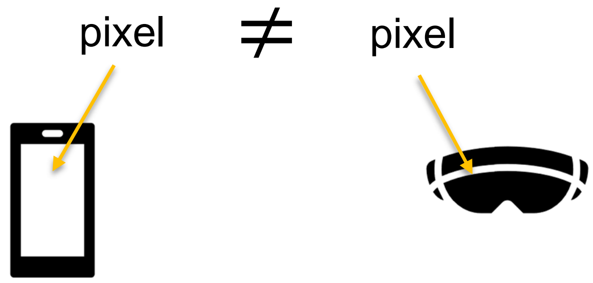
Learned Perceptual Image Patch Similarity (LPIPS)

- Use a pre-trained CNN as a feature extractor

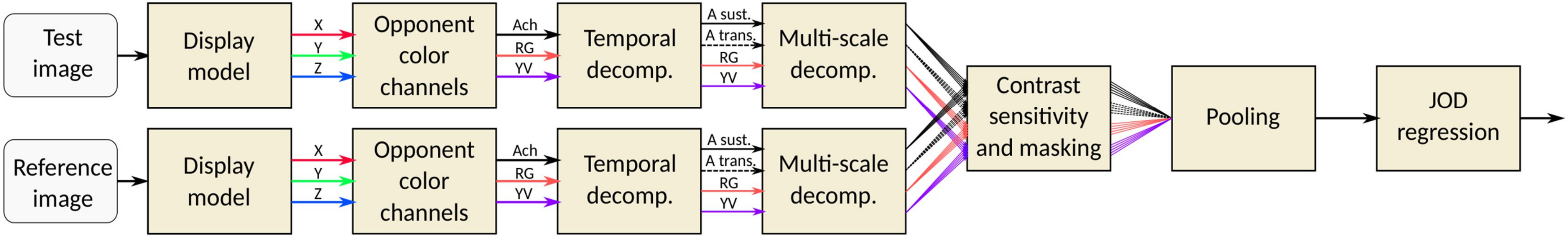


Metrics and viewing conditions

- Majority of image/video metrics disregard viewing conditions
  - Display size
  - Display resolution
  - Viewing distance
  - Display peak luminance
  - Colour gamut
- PSNR, SSIM, LPIPS operate on 0-255 pixel values
  - Cannot handle HDR images/video
- To account for the viewing conditions, we need metrics based on psychophysical models
  - known as visual difference predictors (VDPs)

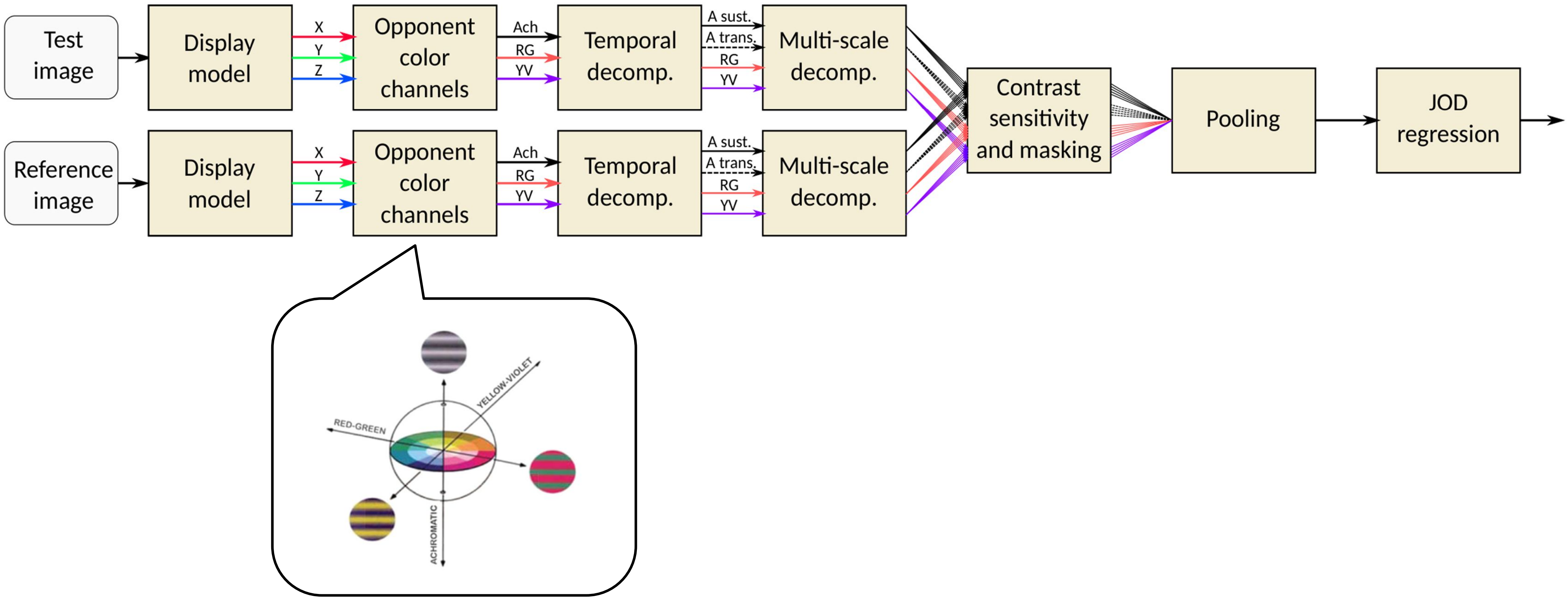


Perceptual metrics (Visual Difference Predictors)

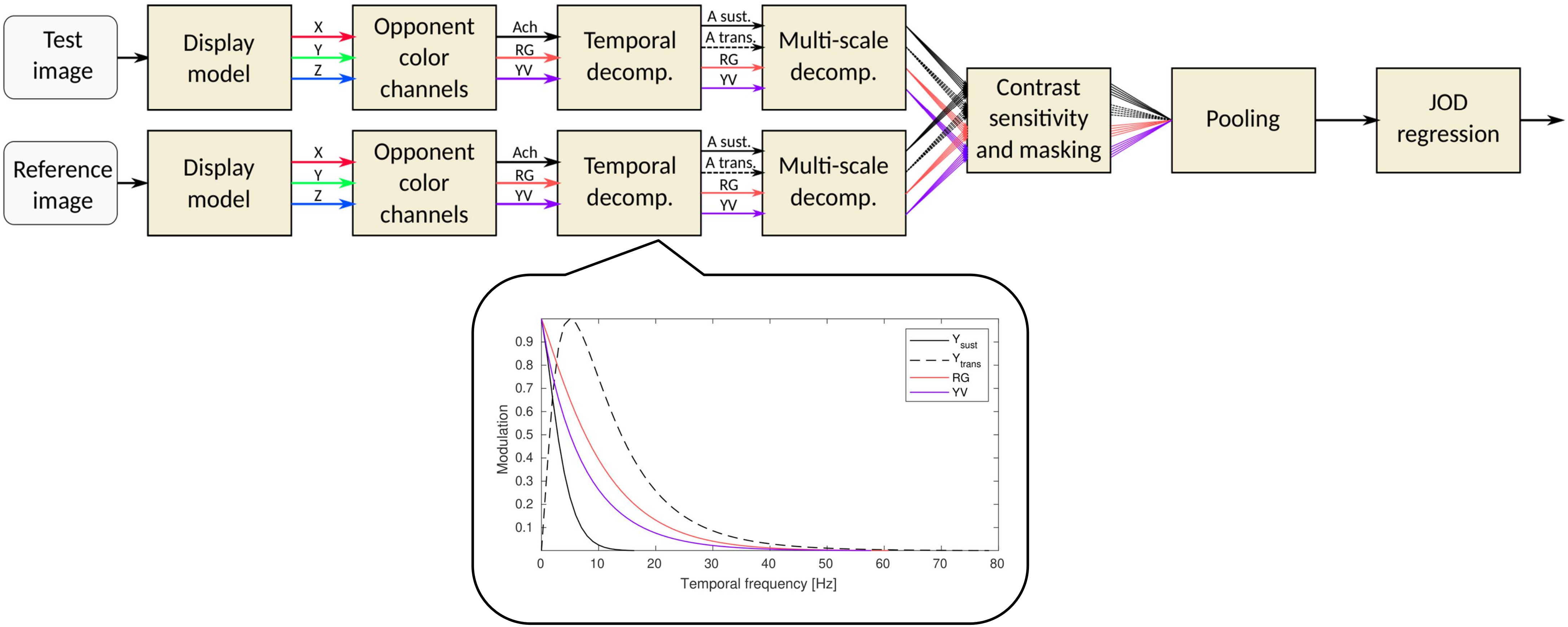



```
"standard_4k": {
  "resolution": [3840, 2160],
  "viewing_distance_meters":  0.7472,
  "diagonal_size_inches": 30,
  "max_luminance": 200,
  "contrast": 1000,
  "E_ambient": 250,
}
```

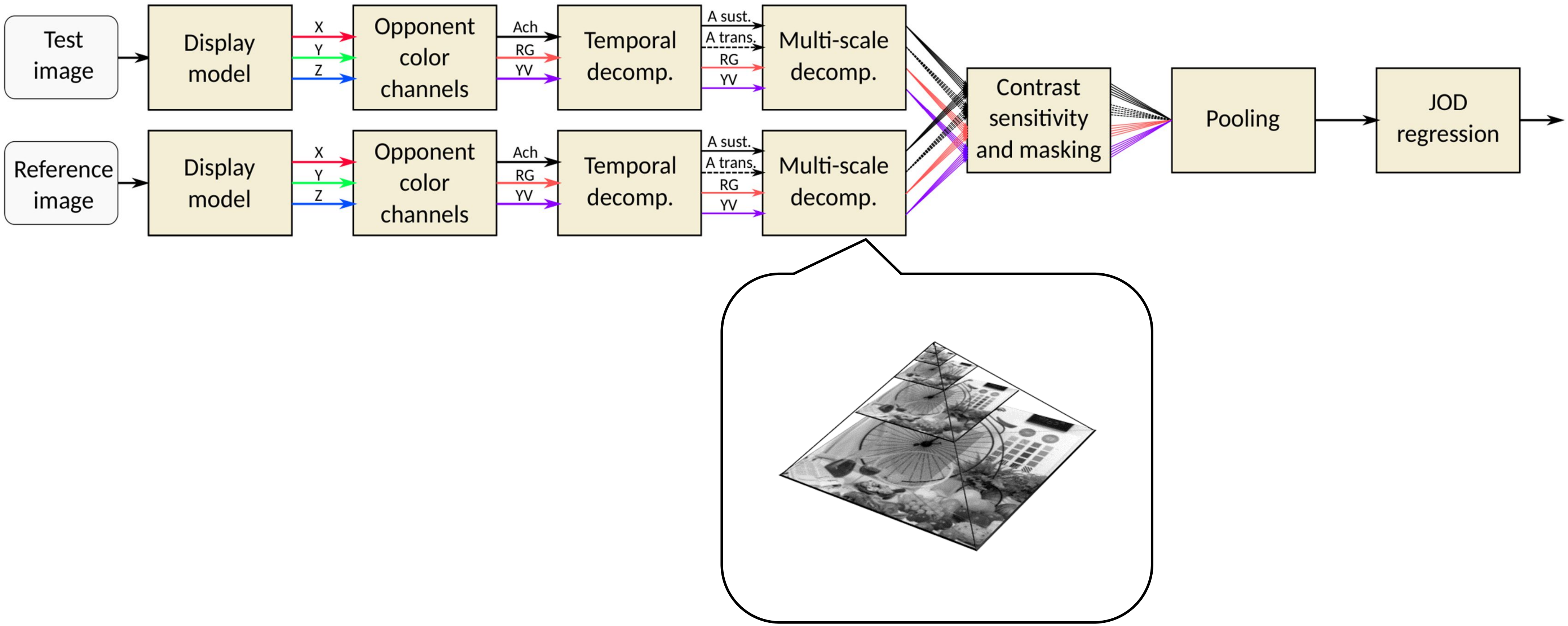
Perceptual metrics (Visual Difference Predictors)



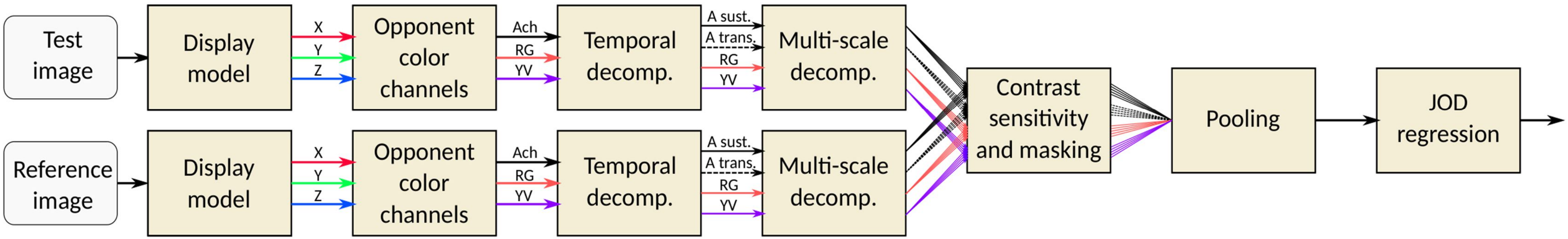
Perceptual metrics (Visual Difference Predictors)



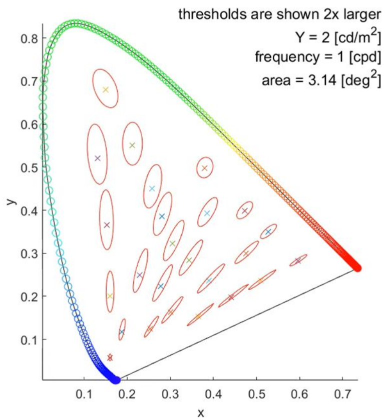
Perceptual metrics (Visual Difference Predictors)



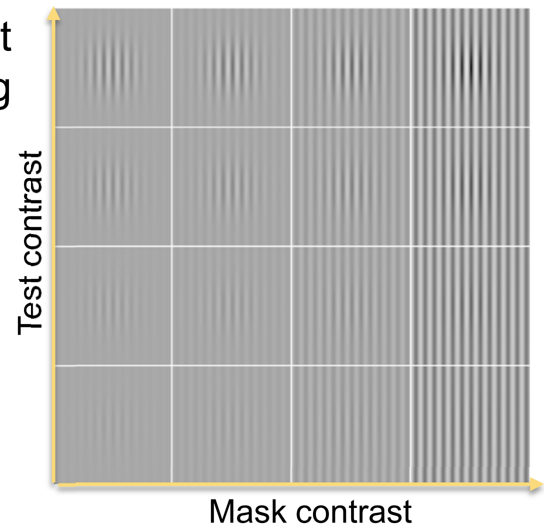
Perceptual metrics (Visual Difference Predictors)



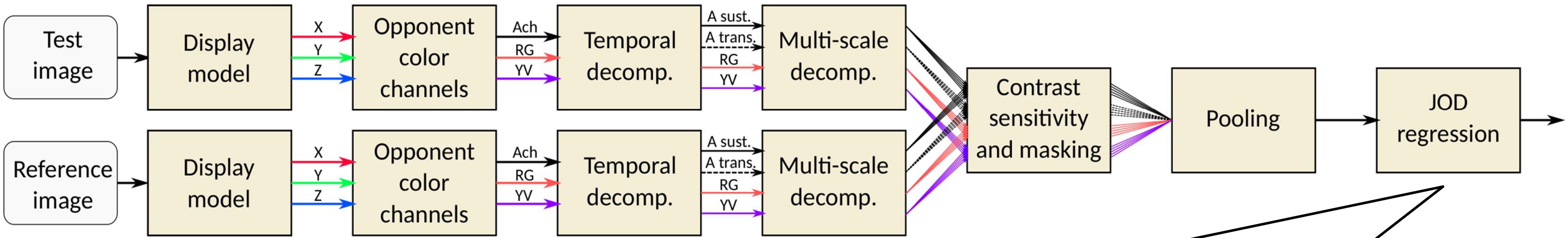
castle**CSF**
minimum
detectable
contrast
difference



Contrast
masking

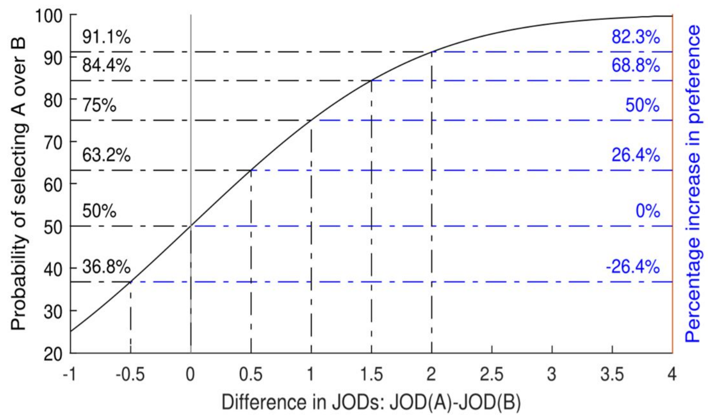


Perceptual metrics (Visual Difference Predictors)

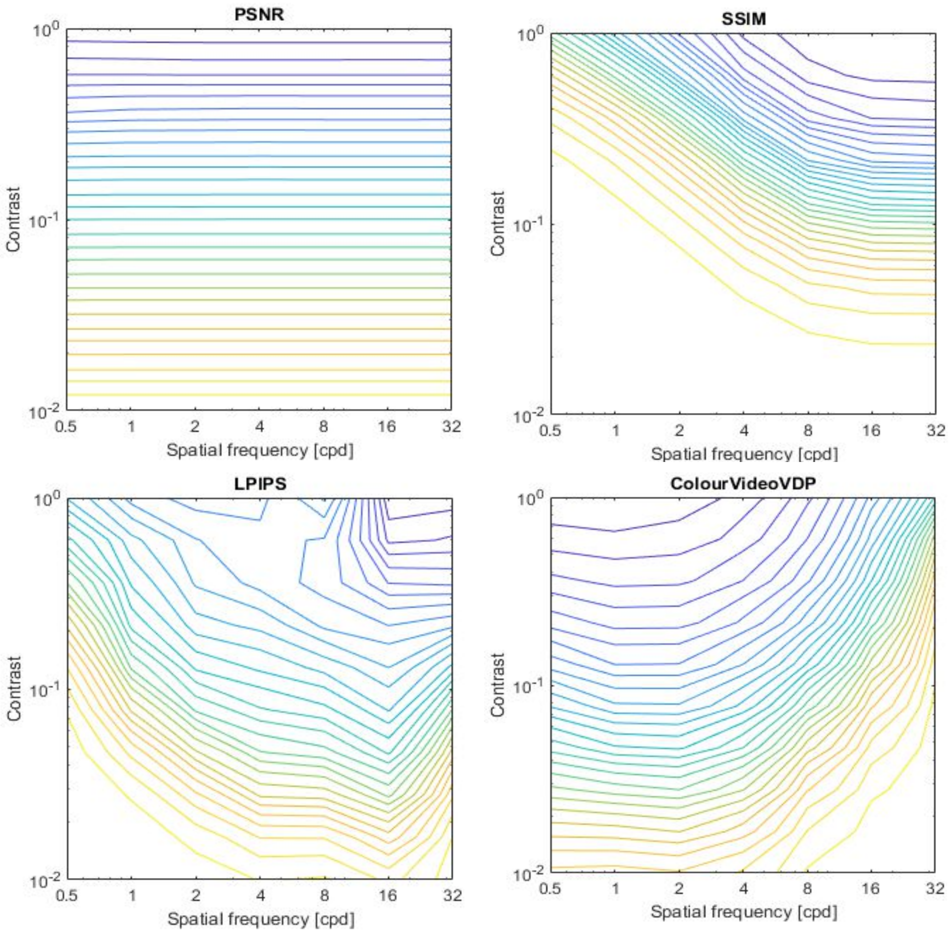


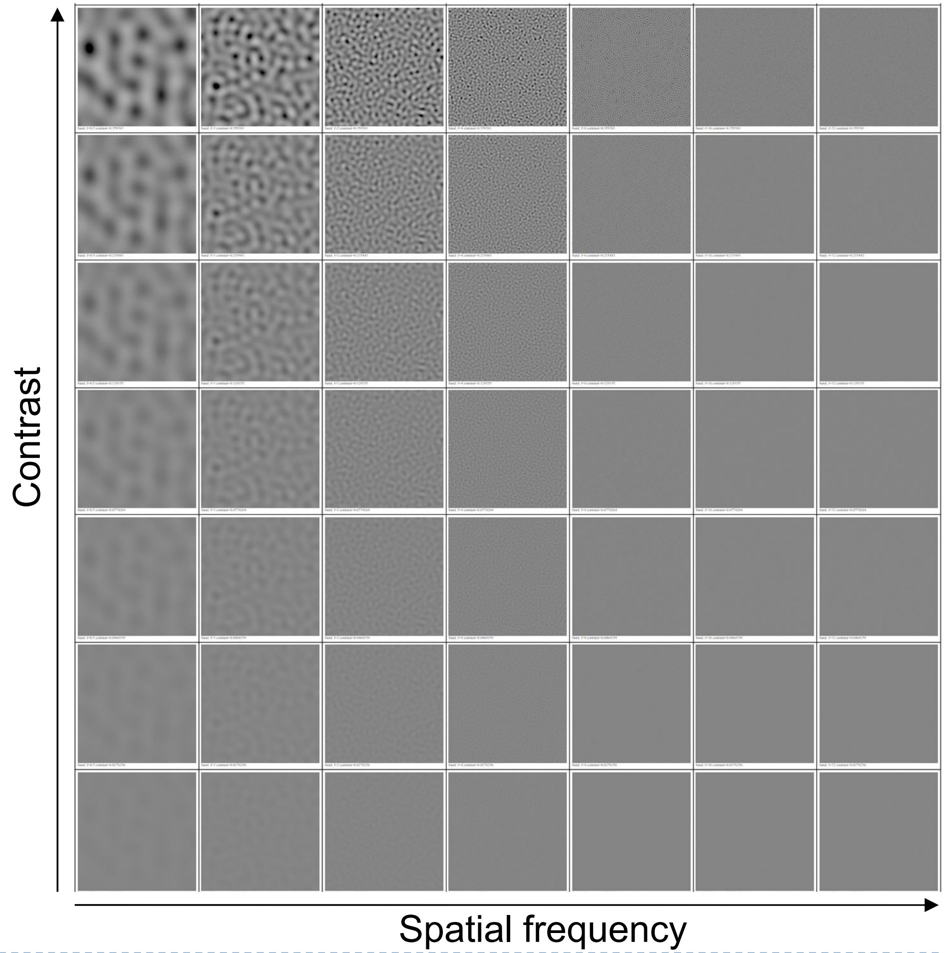
The quality is scaled in the units of
Just Objectionable Differences [JOD]
1 JOD difference ≈ 50% increase in preference

Can express supra-threshold (well-visible) differences



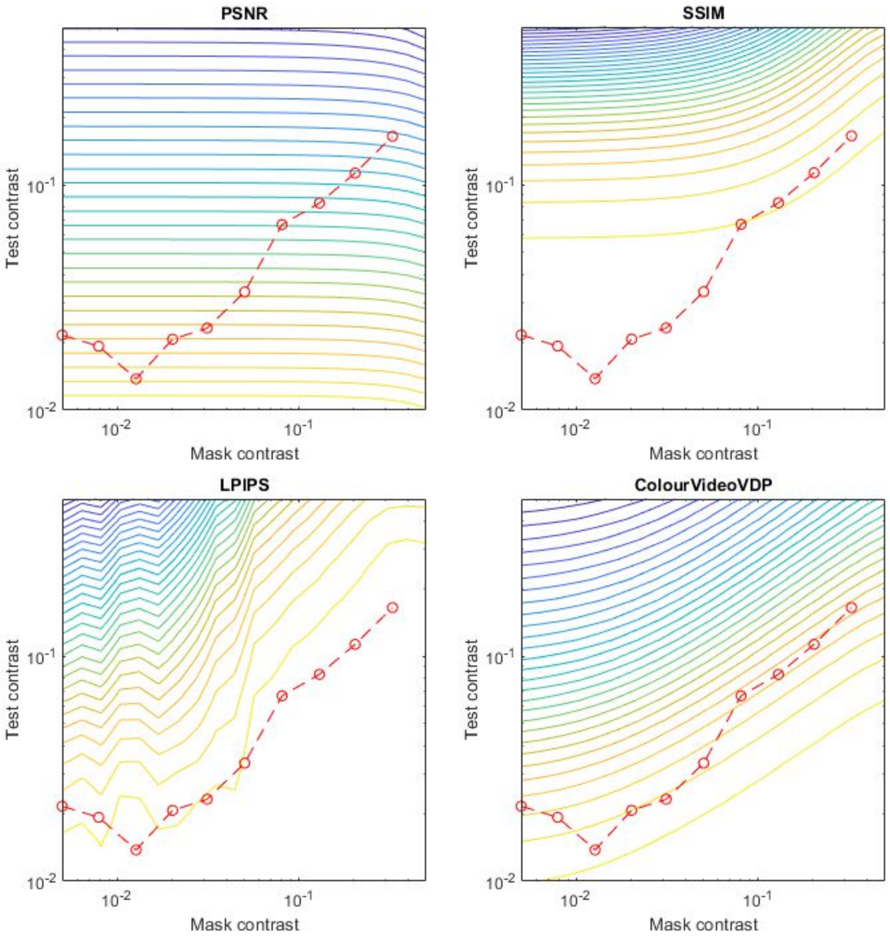
Metric performance on band-limited noise

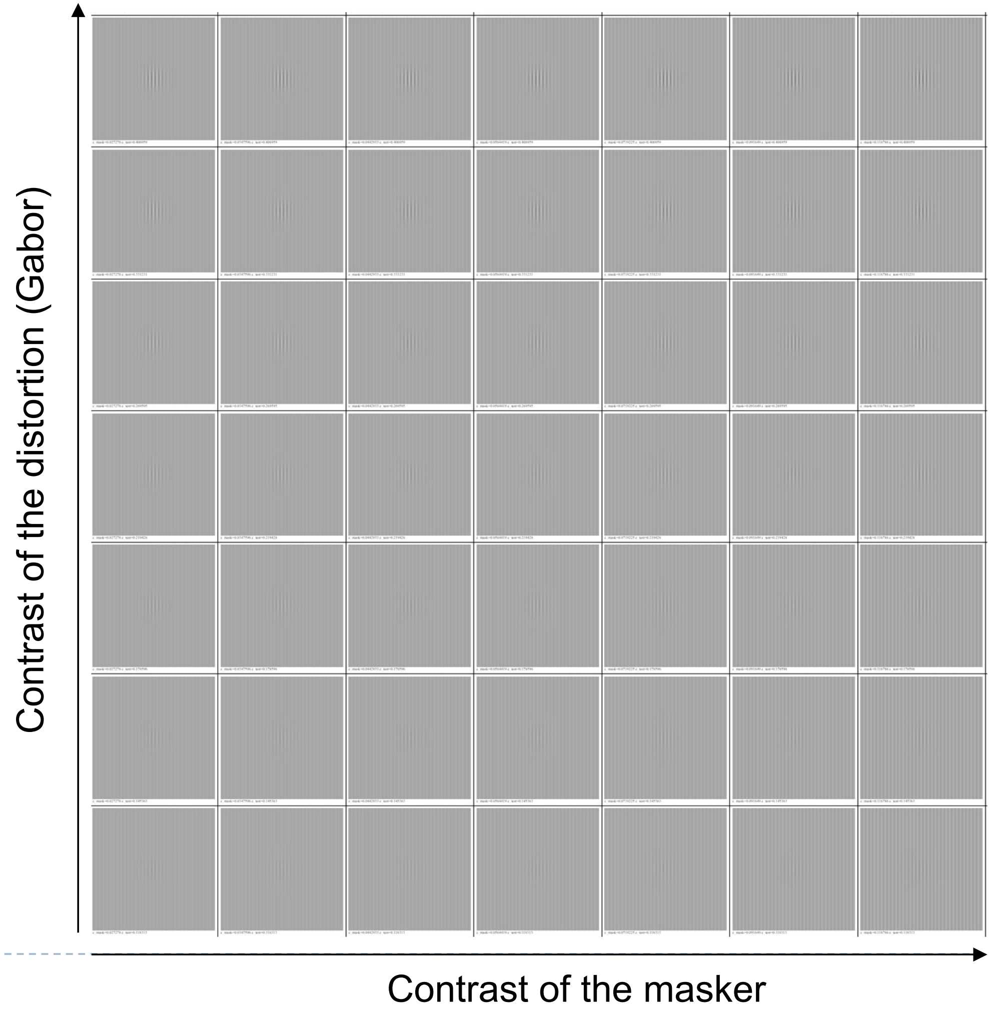




 Violet – large difference; Orange – small difference

Metric performance on masking patterns





 Violet – large difference; Orange – small difference

References

- Scaling of pairwise comparison data
  - pwcmp - https://github.com/mantiuk/pwcmp
  - A practical guide and software for analysing pairwise comparison experiments - https://arxiv.org/abs/1712.03686
- Active sampling
  - ASAP - https://github.com/gfxdisp/asap
- SSIM
  - A Hitchhiker's Guide to Structural Similarity - https://doi.org/10.1109/ACCESS.2021.3056504
- VDP metrics
  - HDR-VDP – https://hdrvdp.sourceforge.net/
  - FovVideoVDP - https://github.com/gfxdisp/FovVideoVDP
  - ColorVideoVDP - https://github.com/gfxdisp/ColorVideoVDP

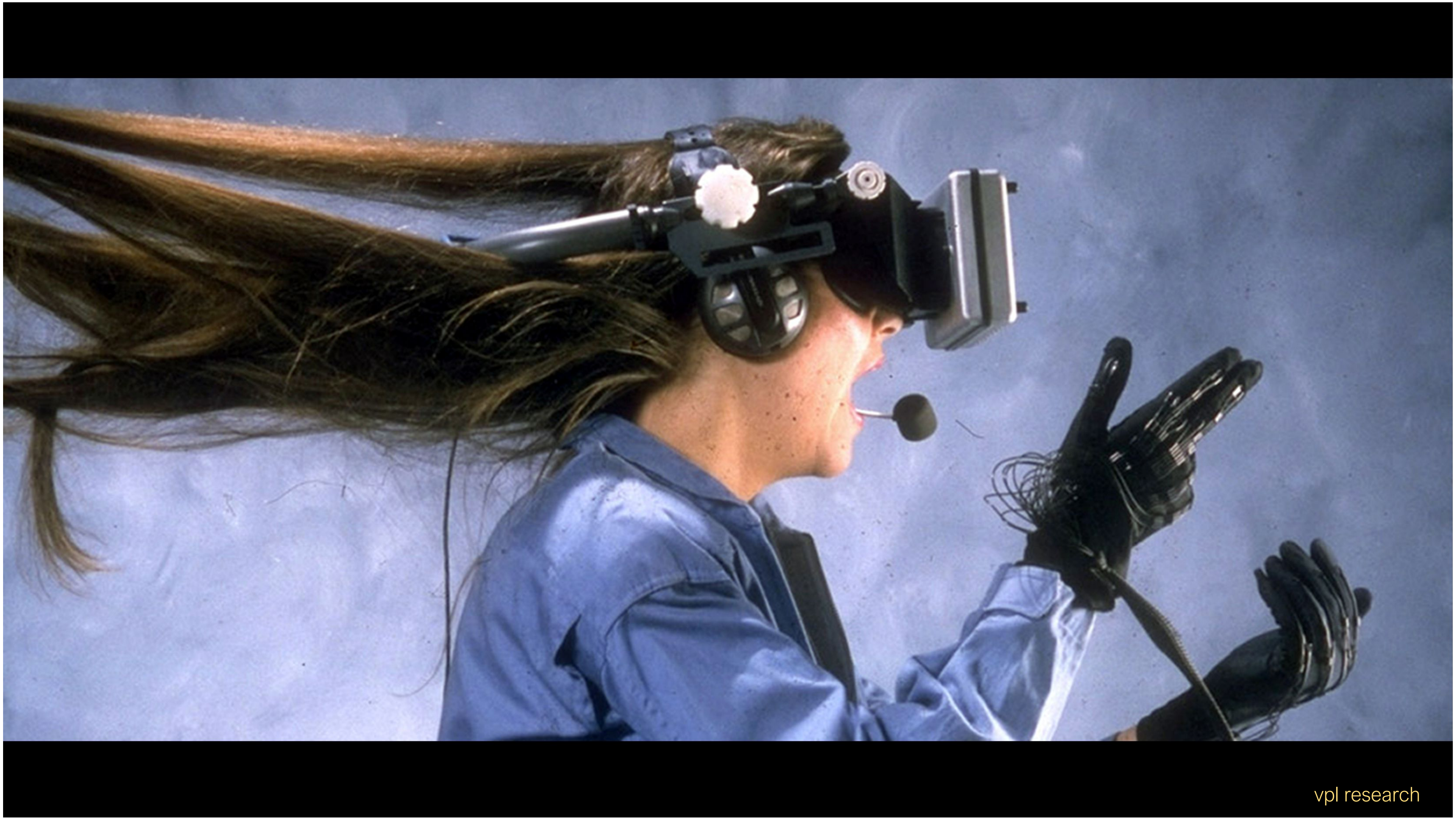
Advanced Graphics & Image Processing

Virtual and Augmented Reality

Part 1/4 – virtual reality

Rafał Mantiuk

Dept. of Computer Science and Technology, University of Cambridge


vpl research

simulation & training

visualization & entertainment

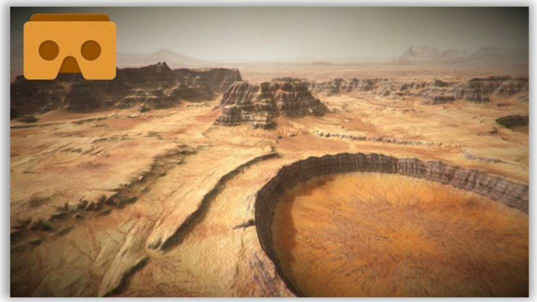
remote control of vehicles, e.g. drones

gaming

robotic surgery

architecture walkthroughs

education

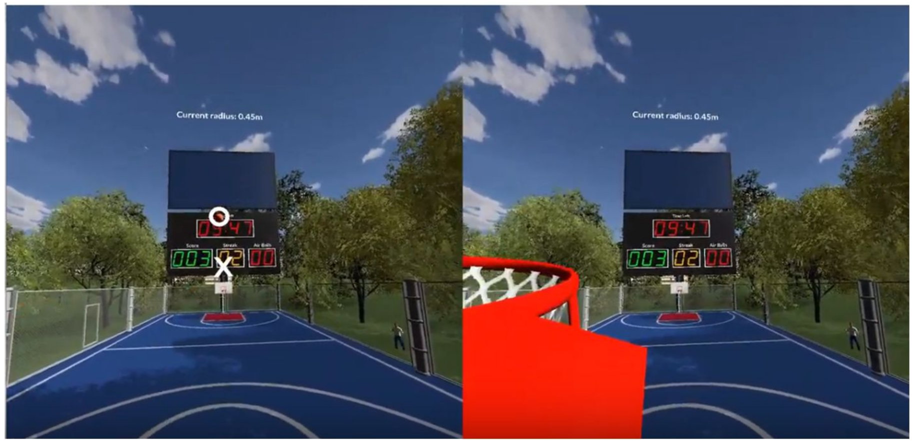


virtual travel

a trip down the rabbit hole

Vision treatment in VR

- Treatment of amblyopia
  - Training the brain to use the "lazy" eye



Images courtesy of VIVID VISION

Exciting Engineering Aspects of VR/AR

- cloud computing
- shared experiences

- CPU, GPU
- IPU, DPU?

- compression, streaming

- sensors & imaging
- computer vision
- scene understanding

images by microsoft, facebook

- VR cameras

- photonics / waveguides
- human perception
- displays: visual, auditory, vestibular, haptic, …

- HCI
- applications

Where We Want It To Be

image by ray ban

Personal Computer
e.g. Commodore PET 1983

Laptop
e.g. Apple MacBook

Smartphone
e.g. Google Pixel

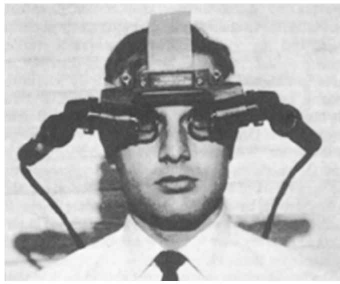
AR/VR
e.g. Microsoft Hololens

???

A Brief History of Virtual Reality

Stereoscopes
Wheatstone, Brewster, …

VR & AR
Ivan Sutherland



Nintendo
Virtual Boy

VR explosion
Oculus/Meta, Sony, HTC, …

???

1838

1968

1995

2012-2022

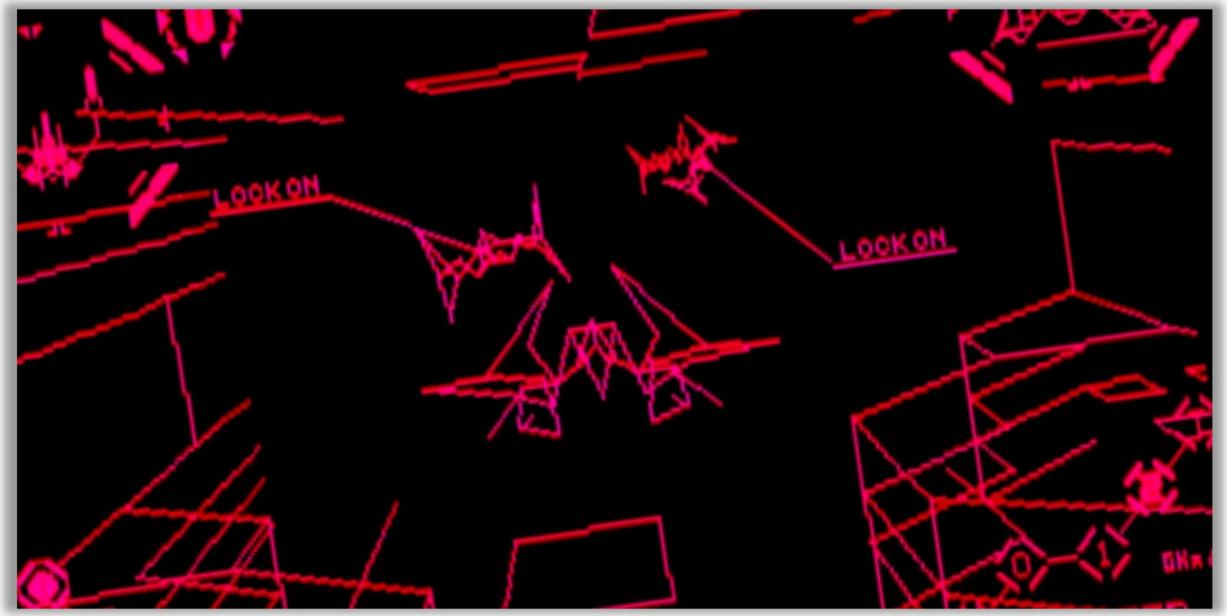
Ivan Sutherland's HMD

- optical see-through AR, including:
  - displays (2x 1" CRTs)
  - rendering
  - head tracking
  - interaction
  - model generation
- computer graphics
- human-computer interaction

I. Sutherland "A head-mounted three-dimensional display", Fall Joint Computer Conference 1968

Nintendo Virtual Boy

- computer graphics & GPUs were not ready yet!

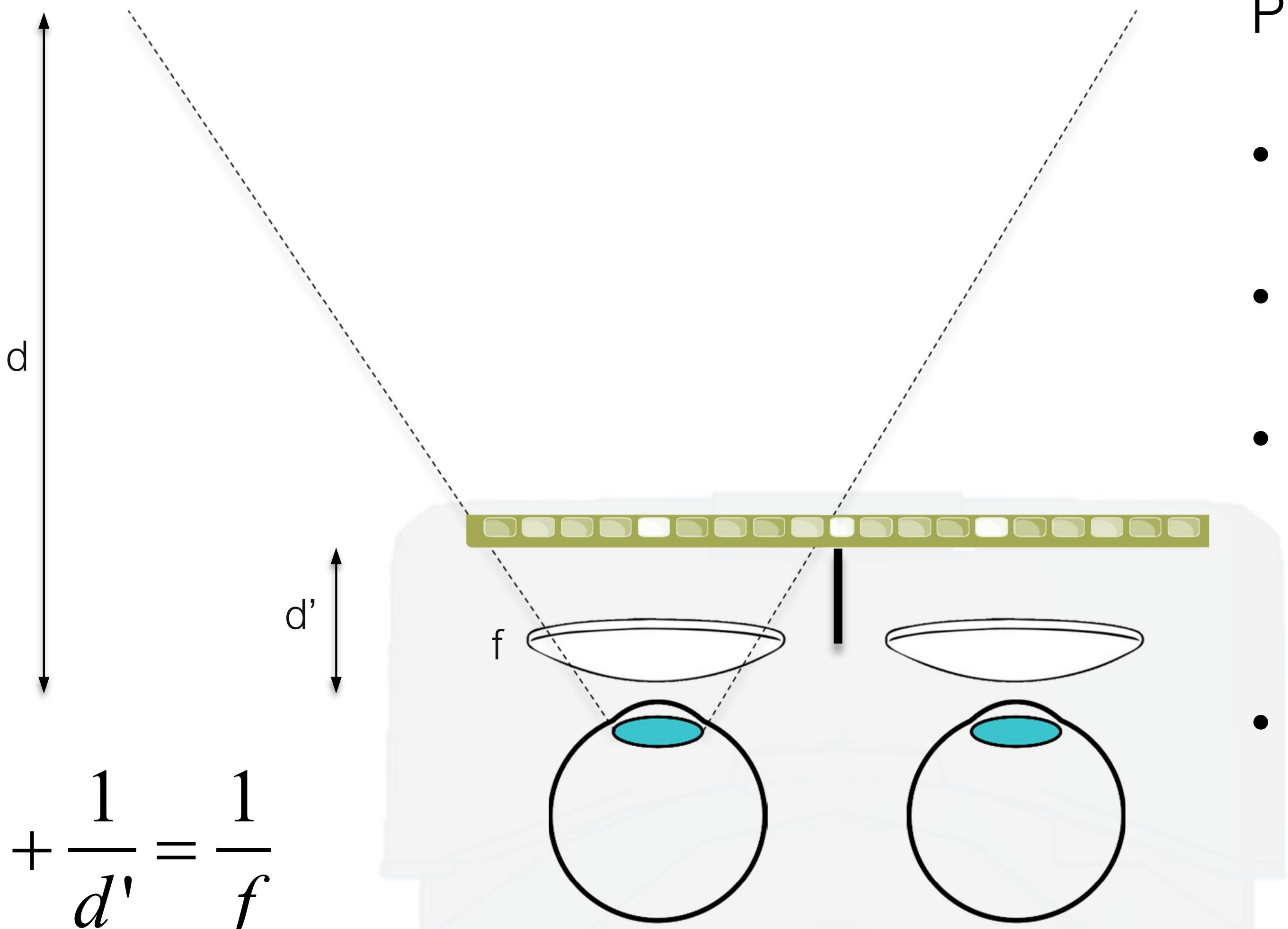


Game: Red Alarm

Where we are now

IFIXIT teardown

Virtual Image

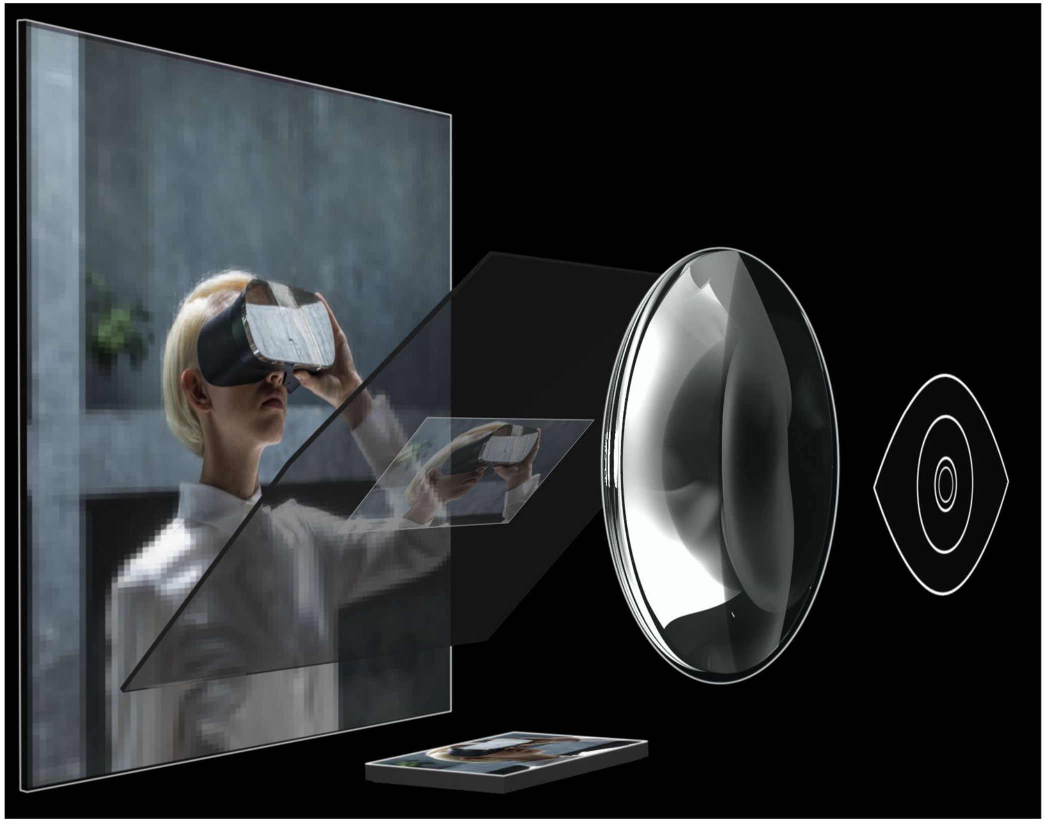


$$\frac{1}{d}+\frac{1}{d'}=\frac{1}{f}$$

Problems:

- fixed focal plane
- no focus cues ☹
- cannot drive accommodation with rendering!
- limited resolution

A dual-resolution display



- High resolution image in the centre, low resolution fills wide field-of-view
- Two displays combined using a beam-splitter
  - Image from: https://varjo.com/bionic-display/

Advanced Graphics & Image Processing

Virtual and Augmented Reality

Part 2/4 – augmented reality

Rafał Mantiuk

Dept. of Computer Science and Technology, University of Cambridge

The slides used in this lecture are the courtesy of Gordon Wetzstein.
From Virtual Reality course: http://stanford.edu/class/ee267/

Pepper's Ghost 1862

Optical see-through AR / head-up displays

Magic Leap 2

Microsoft Hololens 2

Lumus Maximums

Meta 2
(not the current Meta/Facebook)

Intel Vaunt

Google Glass

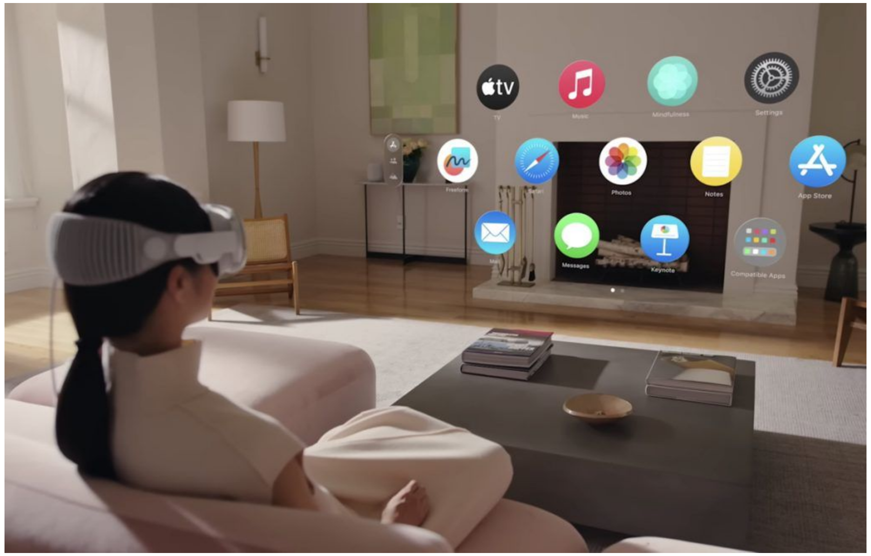
(Some) challenges of optical see-through AR

- Transparency, lack of opacity
  - Display light is mixed with environment light
- Resolution and field-of-view
- Eye-box
  - The volume in which the pupil needs to see the image
- Brightness and contrast
- Blocked vision – forward and periphery (safety)
- Power efficiency
- Size, weight and weight distribution
  - 50 grams are comfortable for long periods
- Social issues, price, vision correction, individual variability…

More resources: https://kguttag.com/

Video pass-through AR

Meta Quest 3



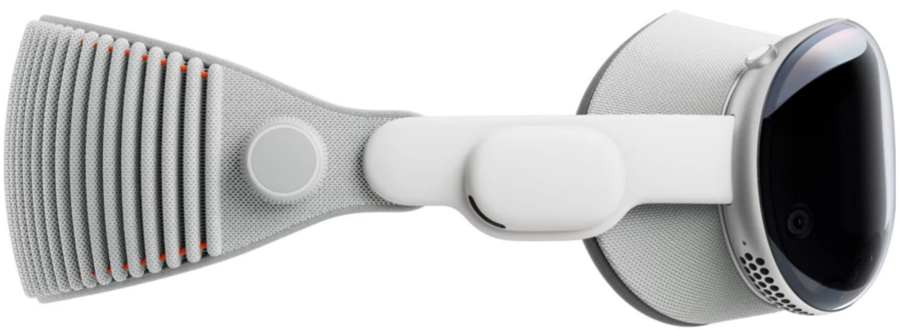
Apple Vision Pro

- Also for smartphones and tablets
- APIs
  - ARCore (by Google, Android/iOS)
  - ARKit (by Apple, iOS)
  - ARToolKit (OpenSource, Multiplatform) - http://www.artoolkitx.org/

Video pass-through AR

Pros:

- Better virtual image quality
- Occlusions are easy
- Simpler, less expensive optics
- Virtual image not affected by ambient light
- AR/VR in one device



Apple Vision Pro

Cons:

- Vergence-accommodation conflict (see the next part)
- Lower brightness, dynamic range and resolution than real-world
- Motion to photon delay
- Real-world images must be warped for the eye position (artifacts)
- Peripheral vision is occluded
  - Or display if affected by ambient light

VR/AR challenges

- Latency (next lecture)
- Tracking
- 3D Image quality and resolution
- Reproduction of depth cues (last lecture)
- Rendering & bandwidth
- Simulation/cyber sickness
- Content creation
  - Game engines
  - Image-Based-Rendering

Simulation sickness

- Conflict between vestibular and visual systems
  - When camera motion inconsistent with head motion
  - Frame of reference (e.g. cockpit) helps
  - Worse with larger FOV
  - Worse with high luminance and flicker

UNIVERSITY OF CAMBRIDGE COMPUTER LABORATORY
RAINBOW GRAPHICS GROUP

Advanced Graphics & Image Processing

Virtual and Augmented Reality

Part 3/4 – depth perception

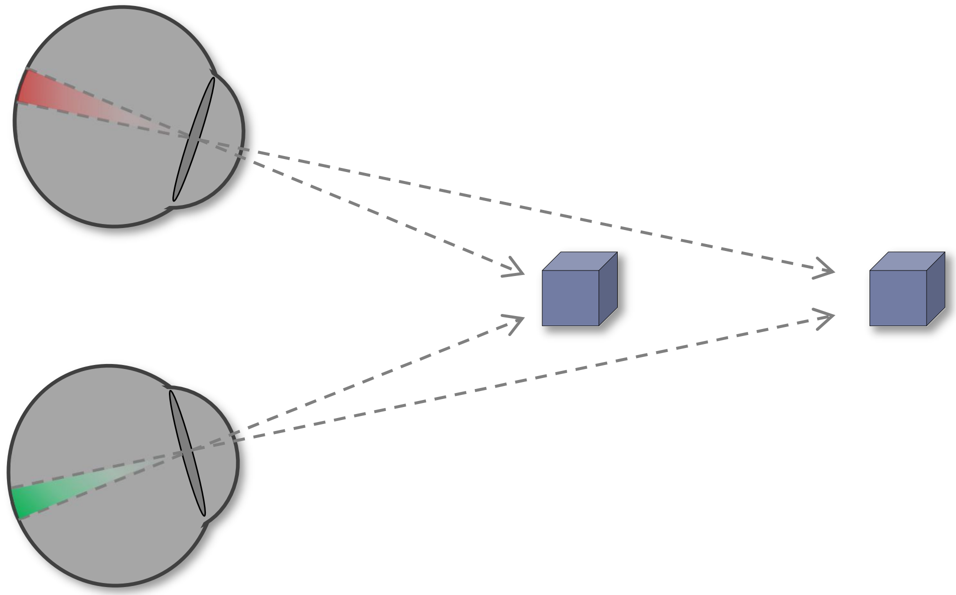
Rafał Mantiuk
Dept. of Computer Science and Technology, University of Cambridge

Depth perception

We see depth due to depth cues.

Stereoscopic depth cues:

binocular disparity



The slides in this section are the courtesy of Piotr Didyk (http://people.mpi-inf.mpg.de/~pdidyk/)

Depth perception

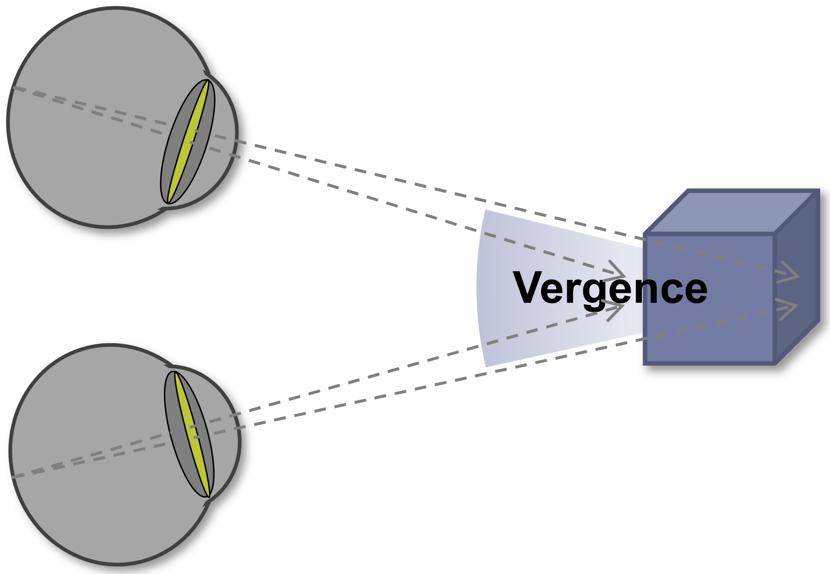
We see depth due to depth cues.

Stereoscopic depth cues:

binocular disparity

Ocular depth cues:

accommodation, vergence



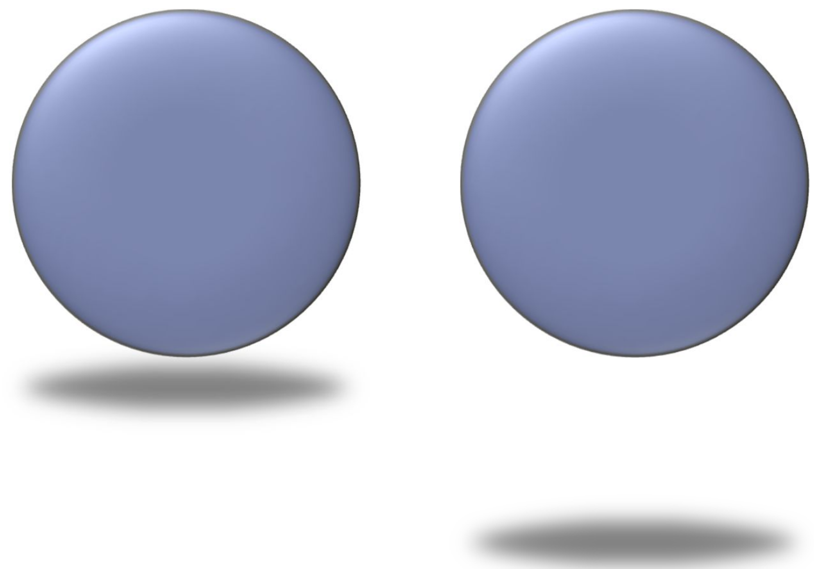
Depth perception

We see depth due to depth cues.

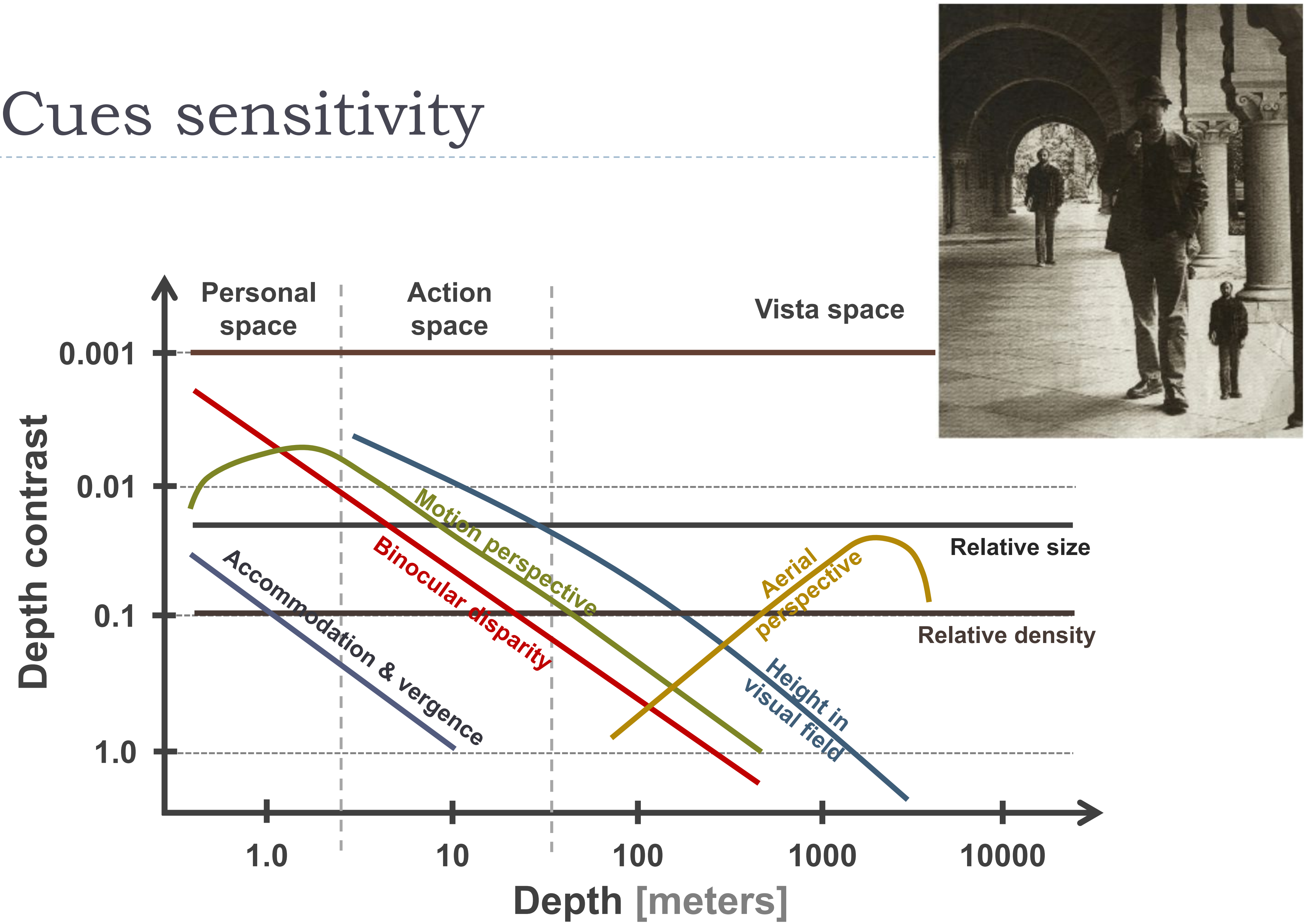
Stereoscopic depth cues:
binocular disparity

Ocular depth cues:
accommodation, vergence

Pictorial depth cues:
occlusion, size, shadows…



Cues sensitivity



"Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth"
by Cutting and Vishton [1995]

Depth perception

We see depth due to depth cues.

Stereoscopic depth cues:
binocular disparity

Ocular depth cues:
accommodation, vergence

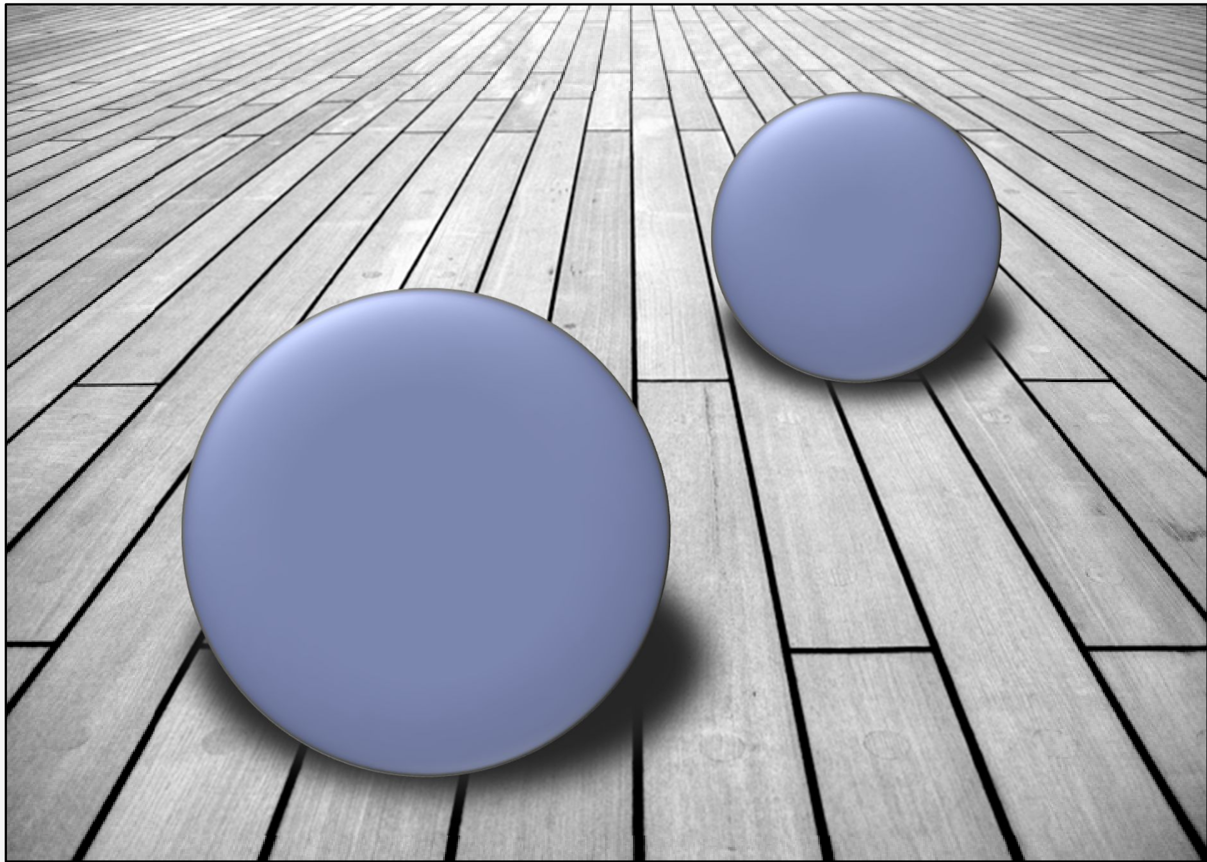
Pictorial depth cues:
occlusion, size, shadows…

Challenge:
Consistency is required!

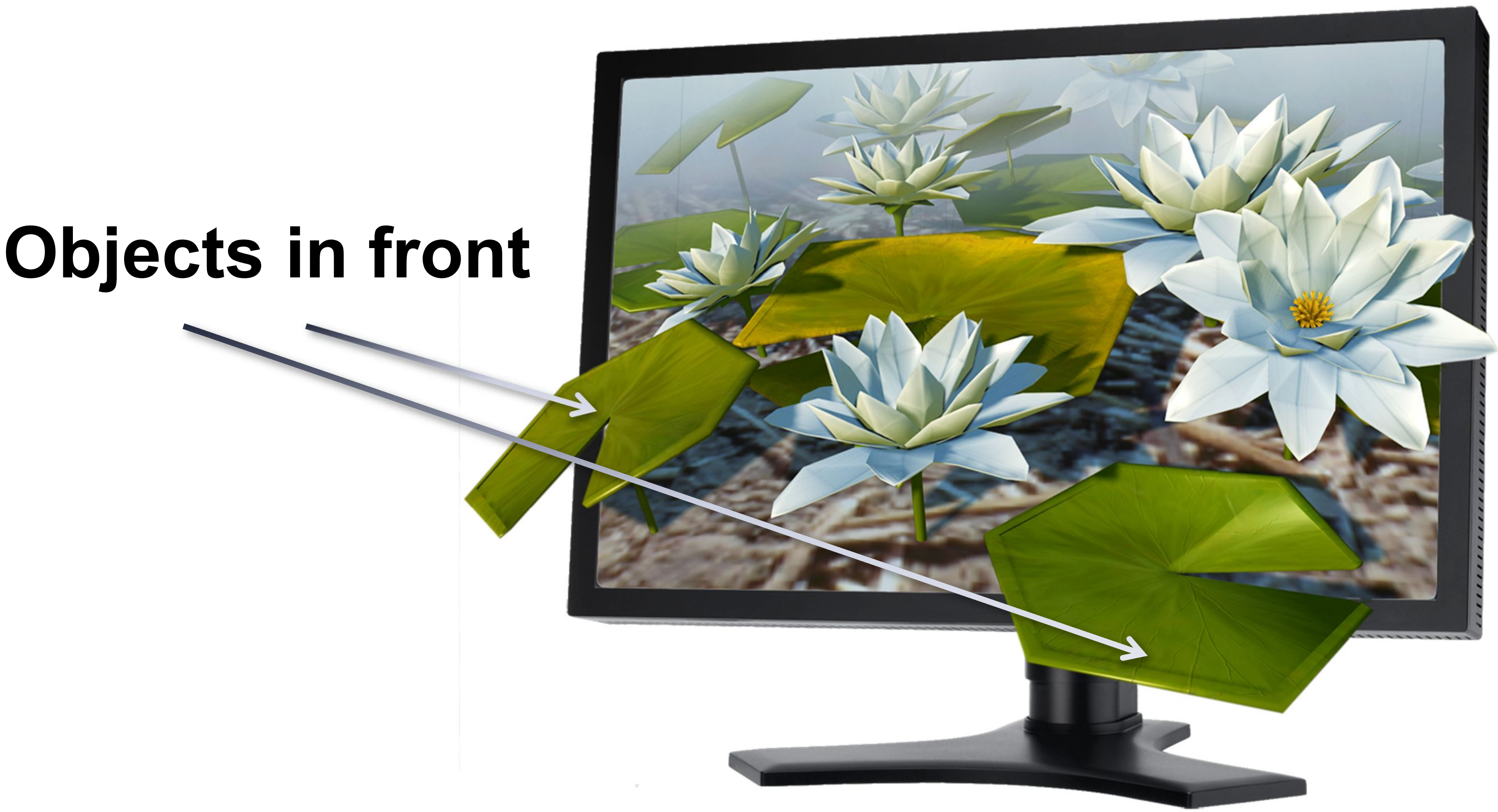
Simple conflict example

Present cues:

- Size
- Shadows
- Perspective
- **Occlusion**



Disparity & occlusion conflict



Disparity & occlusion conflict

Disparity & occlusion conflict

Depth perception

We see depth due to depth cues.

Stereoscopic depth cues:
binocular disparity

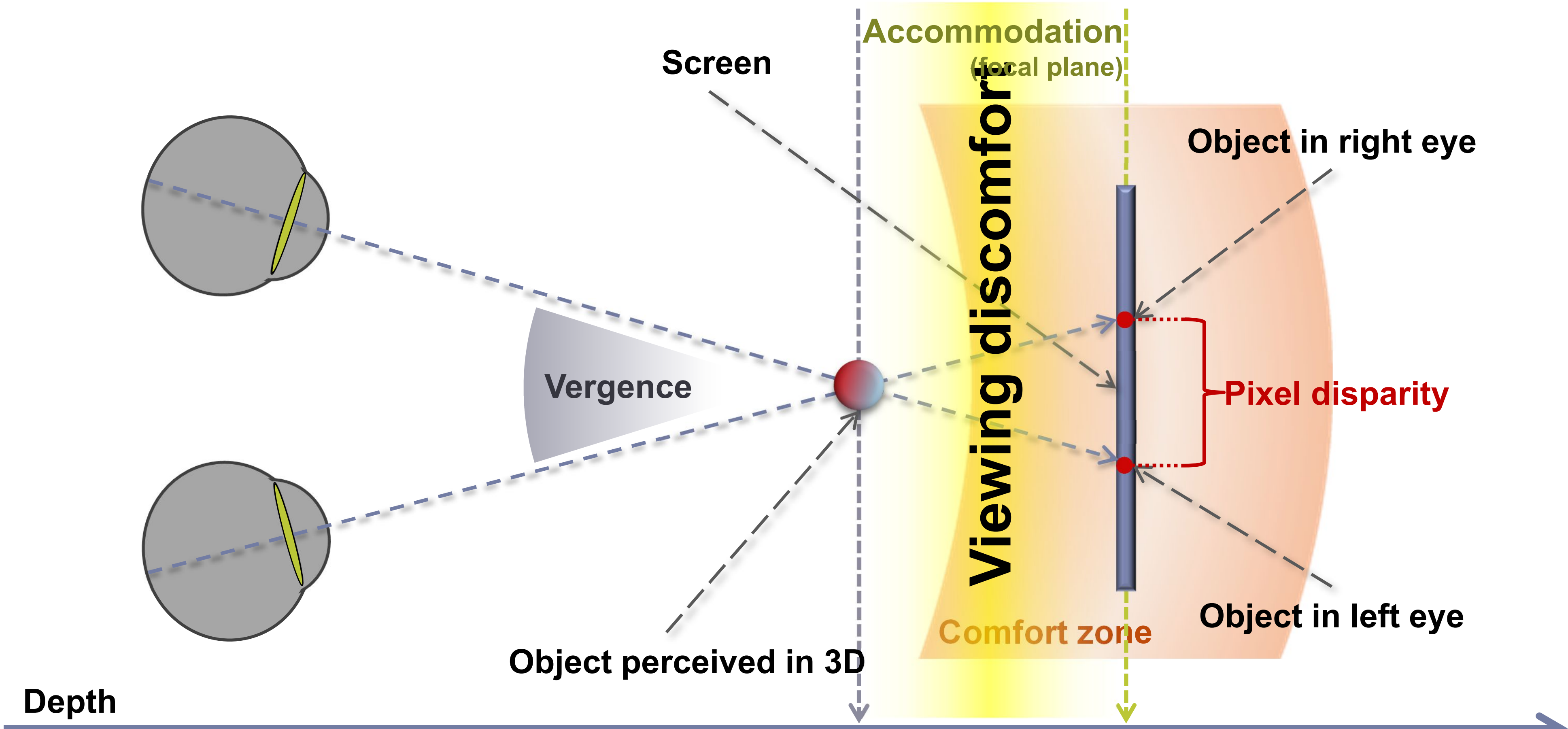
Ocular depth cues:
accommodation, vergence

→ **Require 3D space**
We cheat our Visual System!

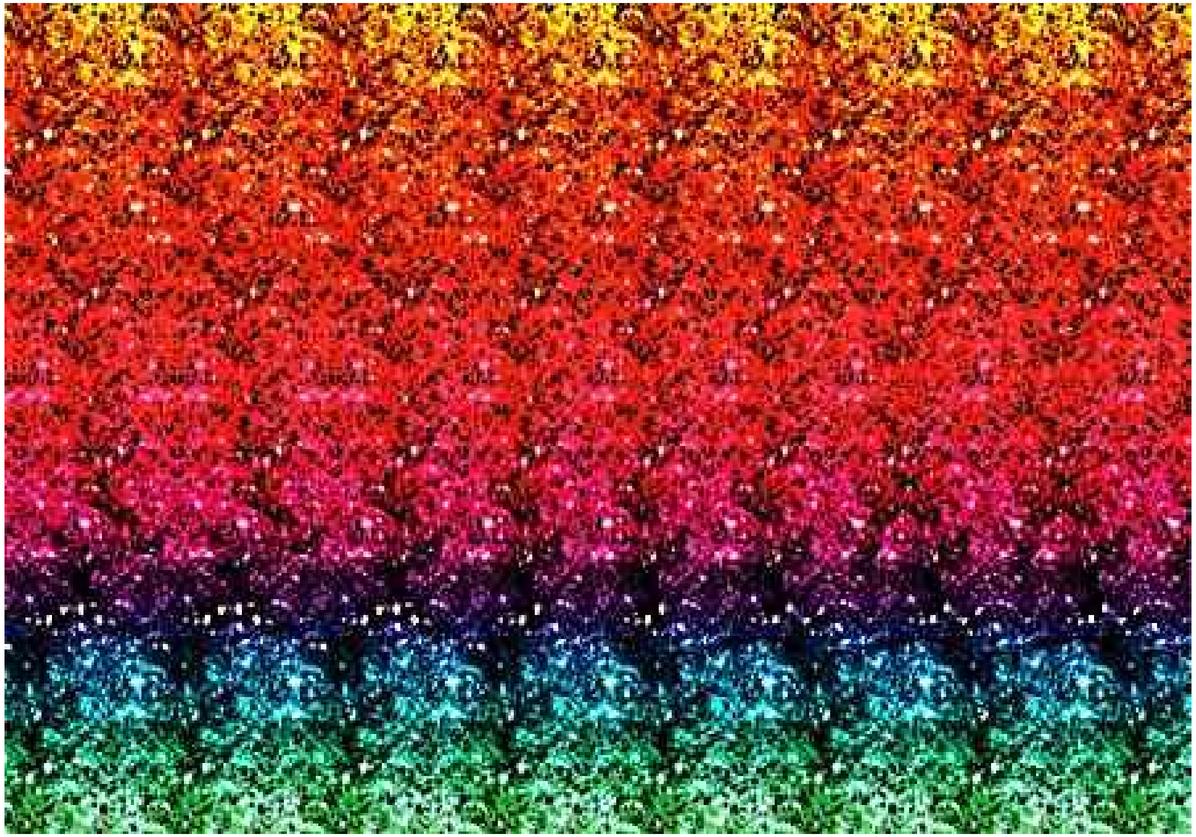
Pictorial depth cues:
occlusion, size, shadows…

→ **Reproducible on a flat displays**

Cheating our HVS

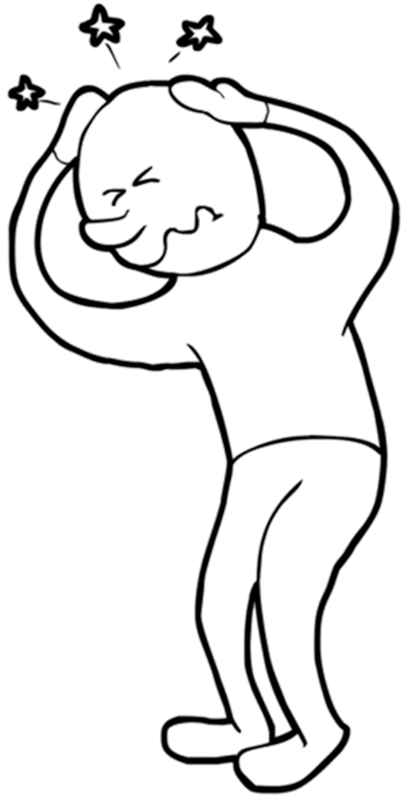


Single Image Random Dot Stereograms



- Fight the vergence vs. accommodation conflict to see the hidden image

Viewing discomfort

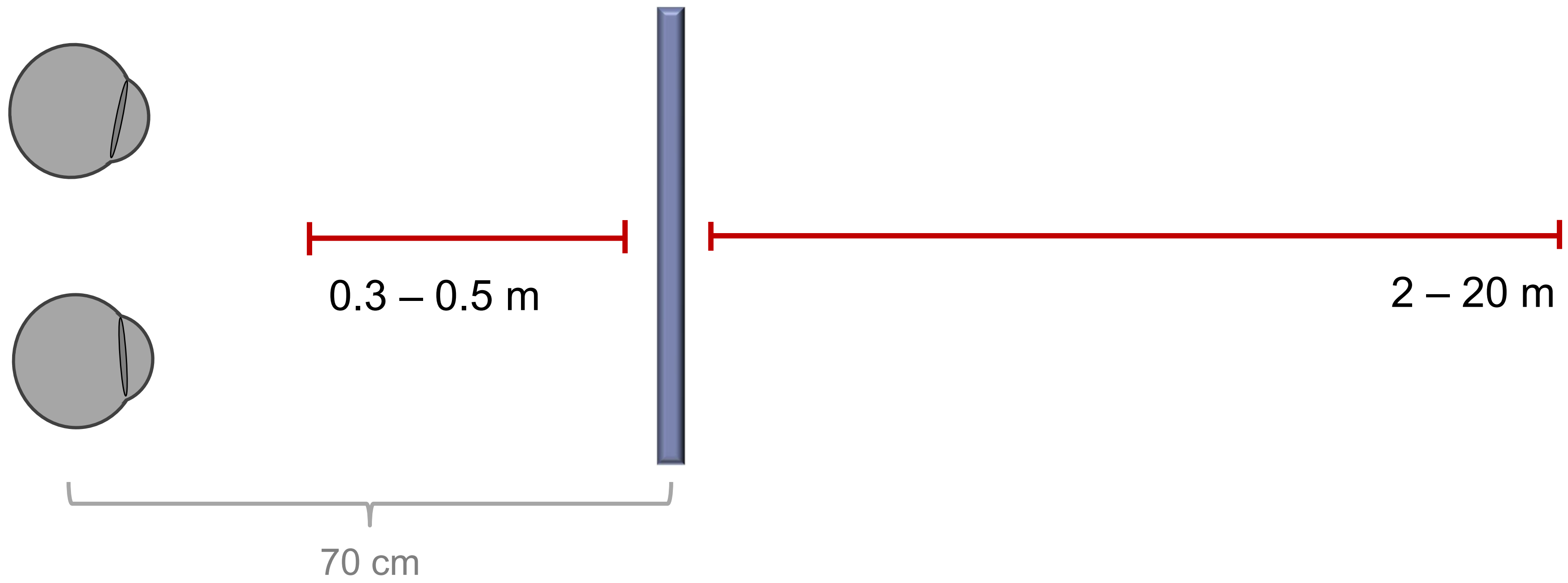


Comfort zones

Comfort zone size depends on:

- Presented content
- Viewing condition

Simple scene



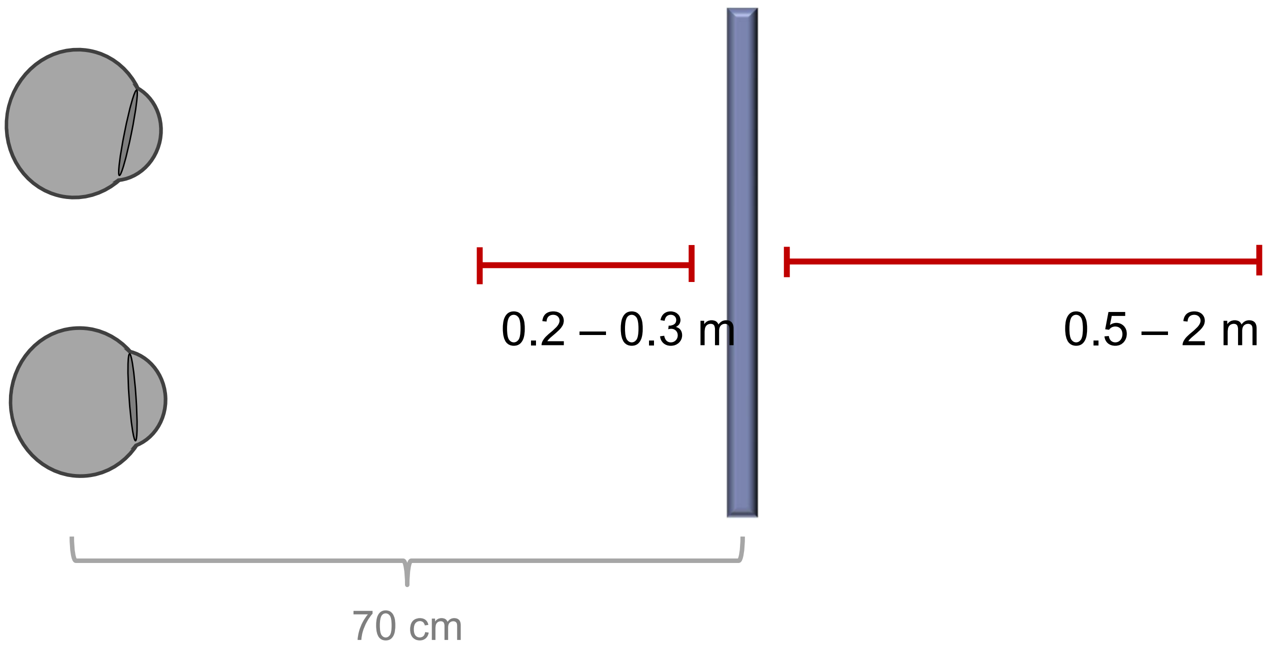
"Controlling Perceived Depth in Stereoscopic Images" by Jones et al. 2001

Comfort zones

Comfort zone size depends on:

- Presented content
- Viewing condition

Simple scene, user allowed to look away from screen



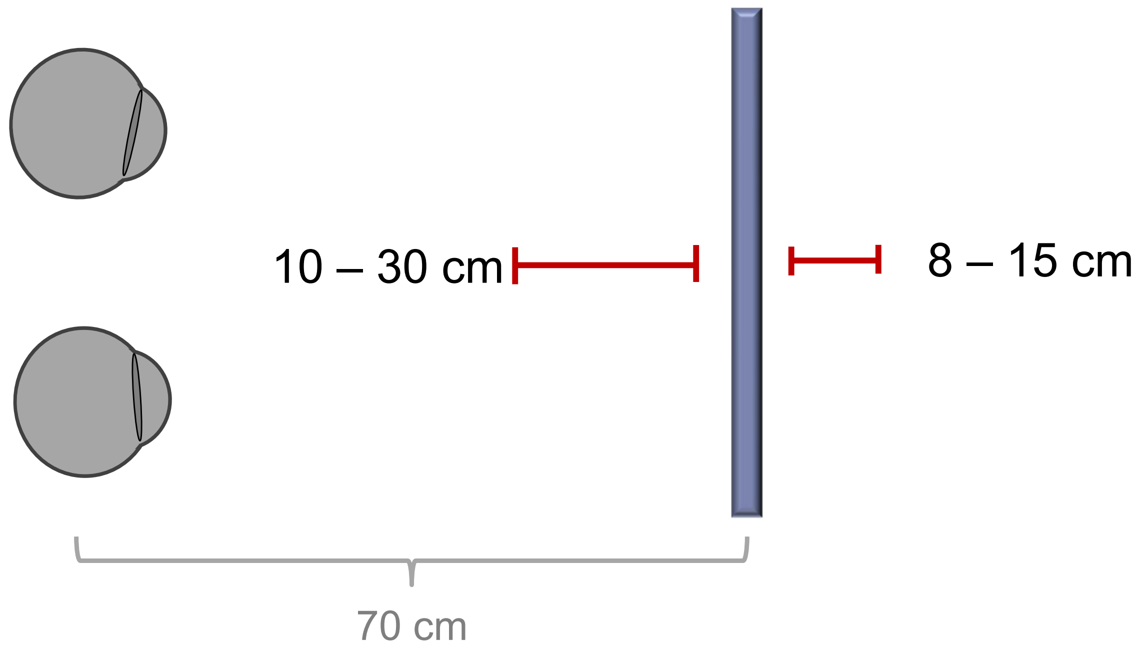
"Controlling Perceived Depth in Stereoscopic Images" by Jones et al. 2001

Comfort zones

Comfort zone size depends on:

- Presented content
- Viewing condition

Difficult scene



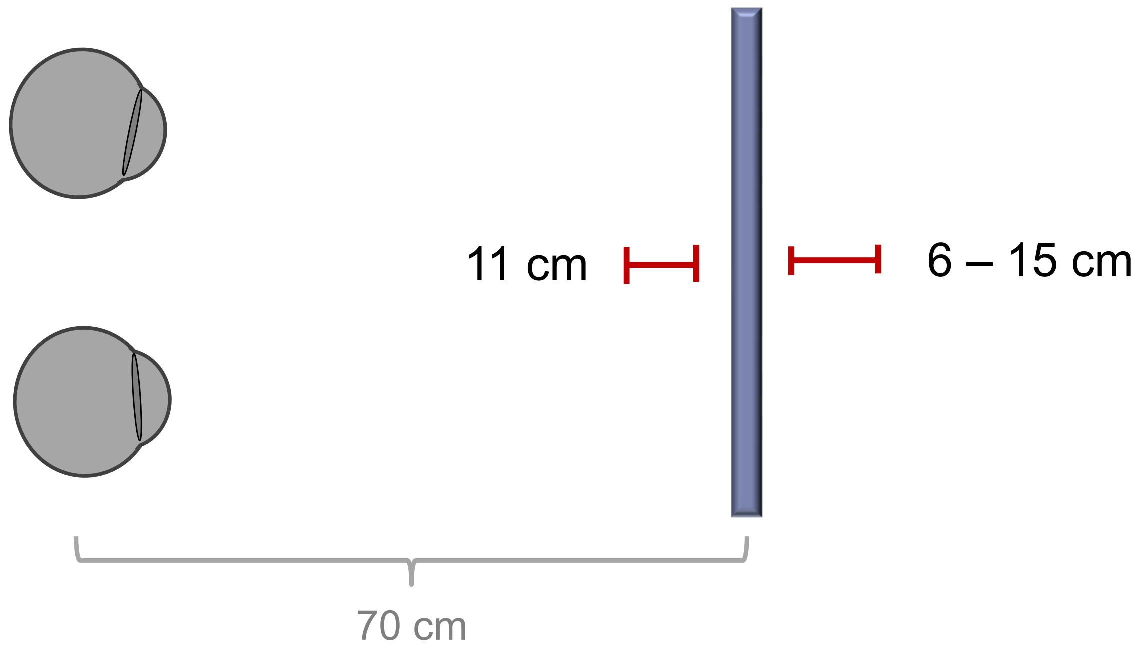
"Controlling Perceived Depth in Stereoscopic Images" by Jones et al. 2001

Comfort zones

Comfort zone size depends on:

- Presented content
- Viewing condition

Difficult scene, user allowed to look away from screen



"Controlling Perceived Depth in Stereoscopic Images" by Jones et al. 2001

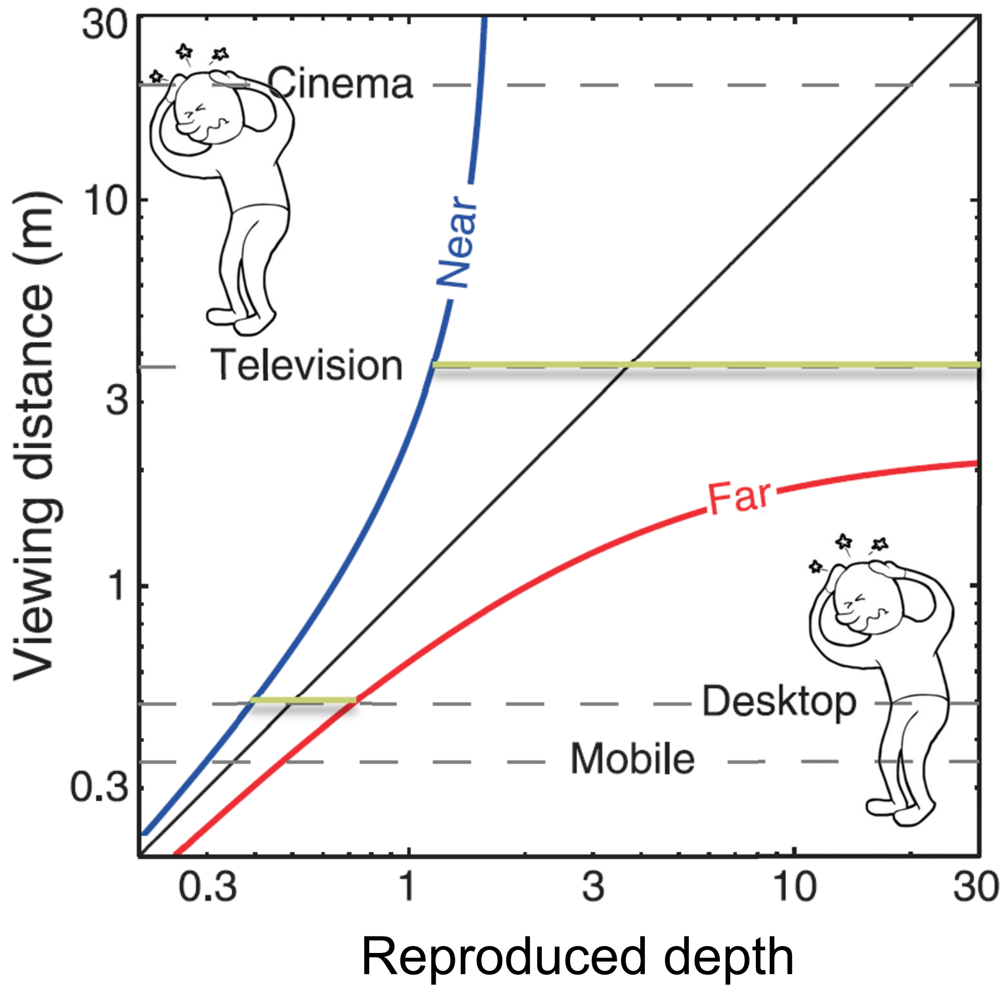
Comfort zones

Comfort zone size depends on:

- Presented content
- Viewing condition
- Screen distance

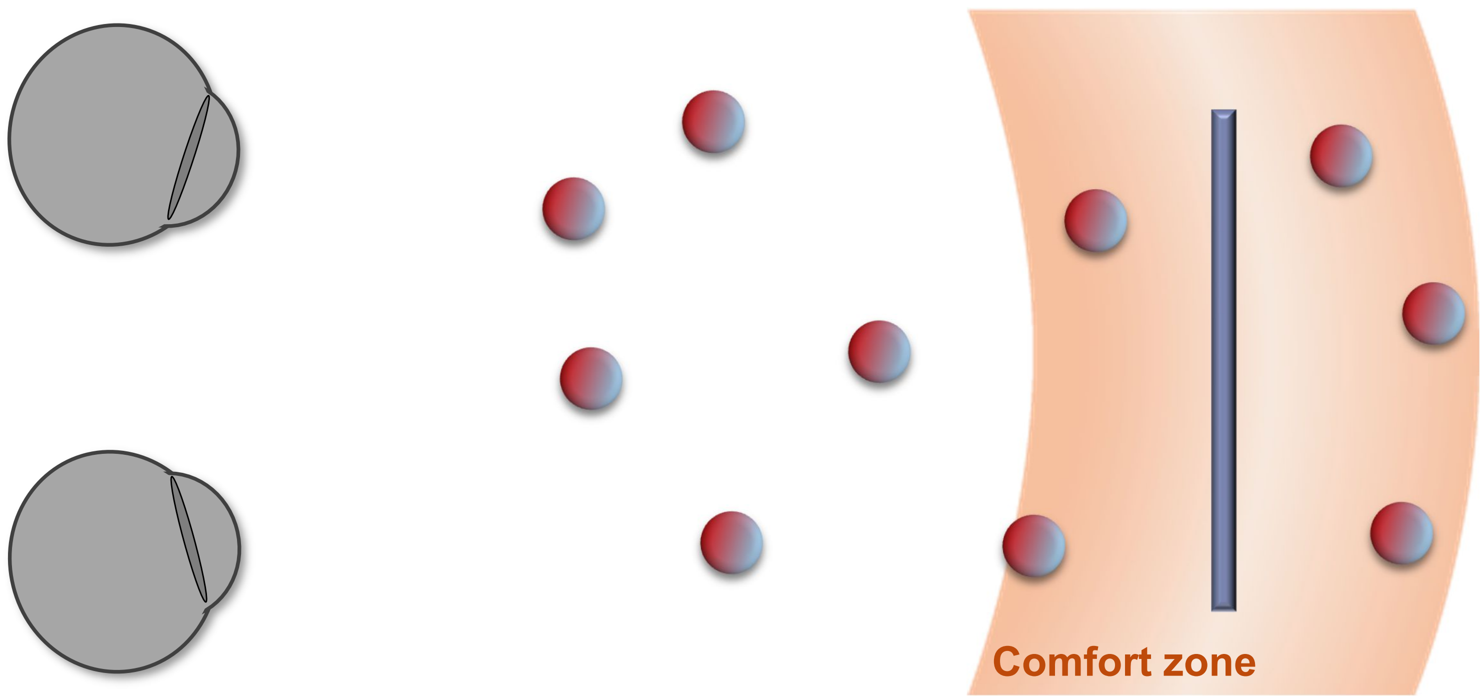
Other factors:

- Distance between eyes
- Depth of field
- Temporal coherence



"The zone of comfort: Predicting visual discomfort with stereo displays" by Shibata et al. 2011

Depth manipulation



Viewing discomfort $\xrightarrow{\text{Scene manipulation}}$ **Viewing comfort**

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Advanced Graphics & Image Processing

Virtual and Augmented Reality
Part 4/4 – stereo rendering

Rafał Mantiuk
Dept. of Computer Science and Technology, University of Cambridge

Put on Your 3D Glasses Now!

The slides used in this section are the courtesy of Gordon Wetzstein.
From Virtual Reality course: http://stanford.edu/class/ee267/

pintrest.com

Anaglyph Stereo - Monochrome

- render L & R images, convert to grayscale
- merge into red-cyan anaglyph by assigning I(r)=L, I(g,b)=R (I is anaglyph)

from movie “Bick Buck Bunny”

Anaglyph Stereo – Full Color

- render L & R images, do not convert to grayscale
- merge into red-cyan anaglyph by assigning I(r)=L(r), I(g,b)=R(g,b) (I is anaglyph)

from movie “Bick Buck Bunny”

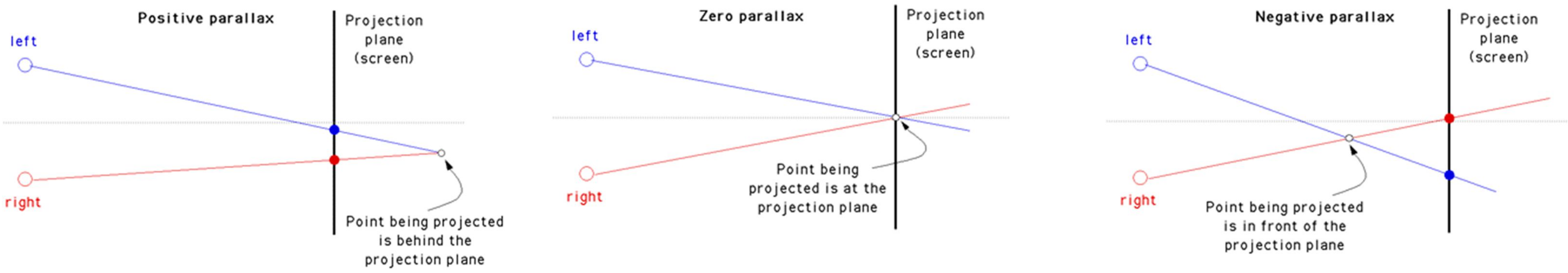
Open Source Movie: Big Buck Bunny

Rendered with Blender (Open Source 3D Modeling Program)

http://bbb3d.renderfarming.net/download.html

Parallax

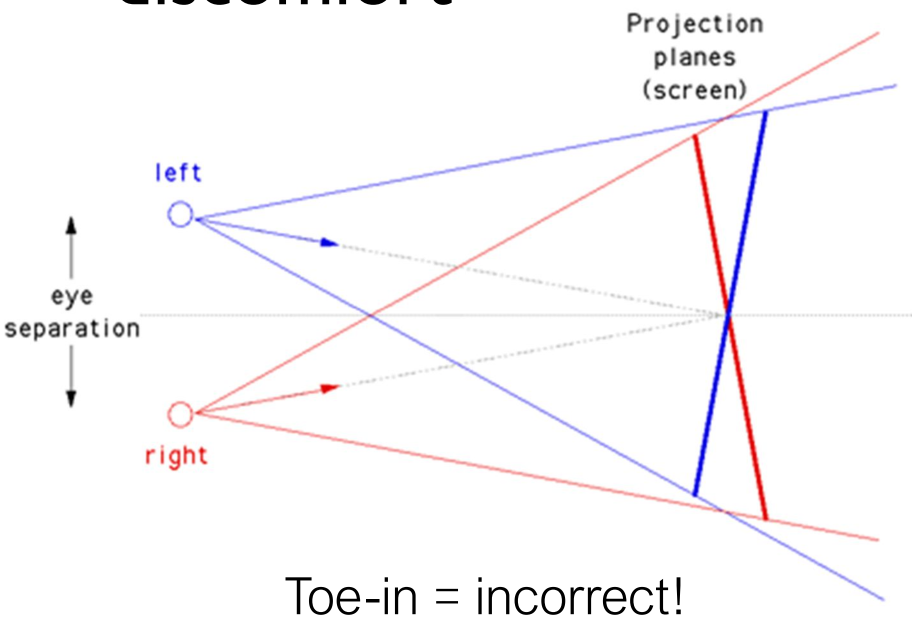
- Parallax is the relative distance of a 3D point projected into the 2 stereo images



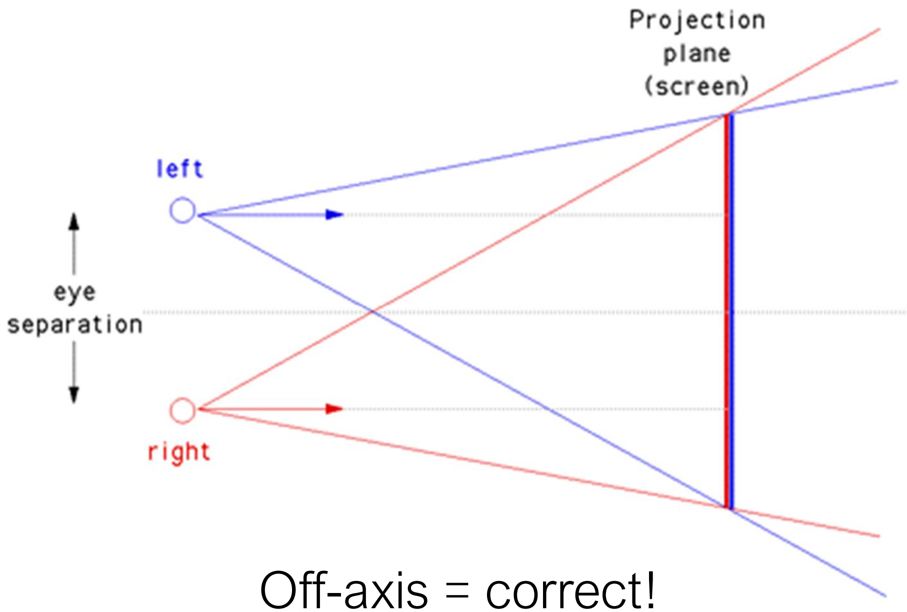
case 1 case 2 case 3

Parallax

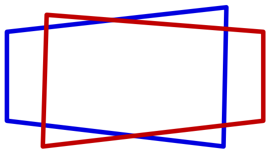
- visual system only uses horizontal parallax, no vertical parallax!
- naïve toe-in method creates vertical parallax and visual discomfort



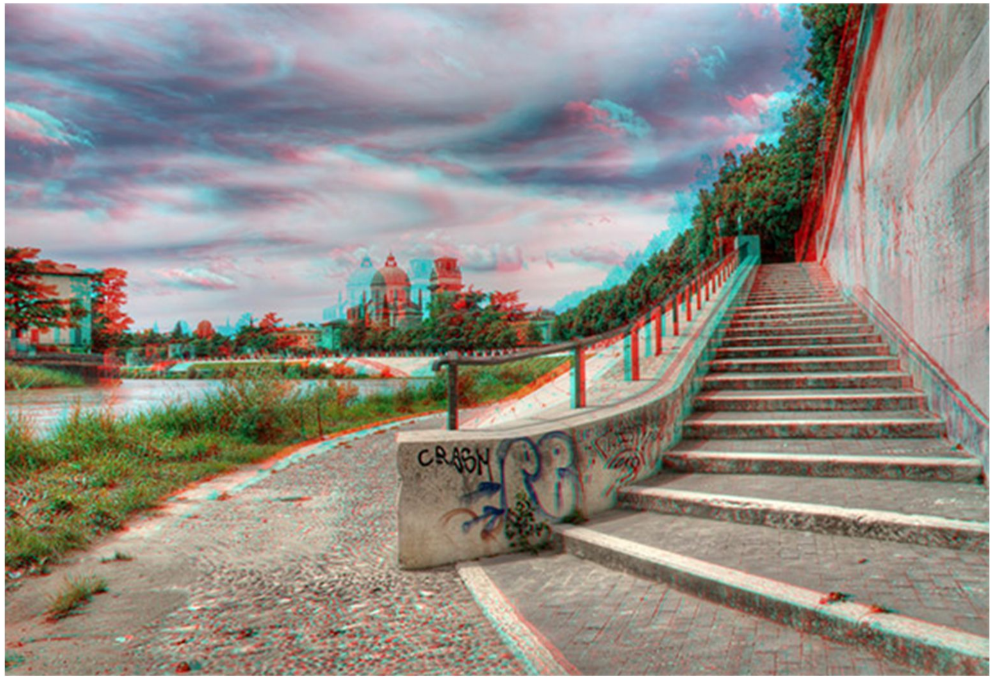
Toe-in = incorrect!



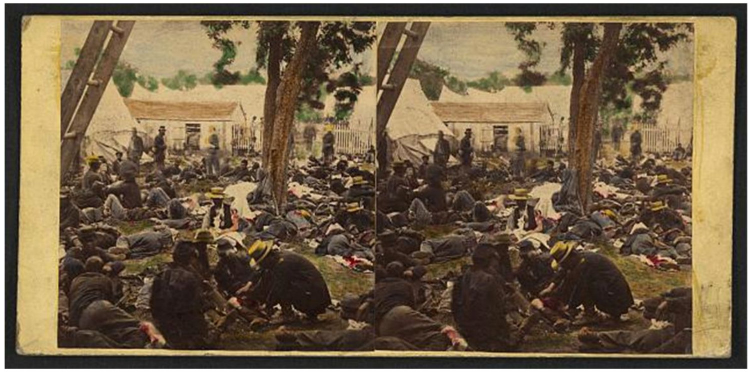
Off-axis = correct!



Parallax – well done



Parallax – well done



1862
“Tending wounded Union soldiers at Savage's Station, Virginia, during the Peninsular Campaign”,
Library of Congress Prints and Photographs Division

Parallax – not well done (vertical parallax = unnatural)

References

- LaValle "Virtual Reality", Cambridge University Press, 2016
  - http://vr.cs.uiuc.edu/
- Virtual Reality course from the Stanford Computational Imaging group
  - http://stanford.edu/class/ee267/
- KGOnTech blog
  - https://kguttag.com/
- The selected slides used in this lecture are the courtesy of Gordon Wetzstein (Virtual Reality course: http://stanford.edu/class/ee267/)

UNIVERSITY OF CAMBRIDGE
COMPUTER LABORATORY

RAINBOW GRAPHICS GROUP

Display Technologies

Advanced Graphics and Image Processing

Rafał Mantiuk
Computer Laboratory, University of Cambridge

Overview

- Temporal aspects
  - Latency in VR
  - Eye-movement
  - Hold-type blur
- 2D displays
  - 2D spatial light modulators
  - High dynamic range displays

Latency in VR

- Sources of latency in VR
  - IMU ~1 ms
    - Inertial Measurement Unit
  - sensor fusion, data transfer
  - rendering: depends on complexity of scene & GPU – a few ms
  - data transfer again
  - Display
    - 60 Hz = 16.6 ms;
    - 70 Hz = 11.1 ms;
    - 120 Hz = 8.3 ms.

- Target latency
  - Maximum acceptable: 20ms
  - Much smaller (5ms) desired for interactive applications

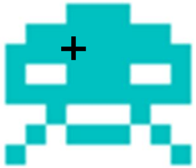
- Example
  - 16 ms (display) + 16 ms (rendering) + 4 ms (orientation tracking) = 36 ms latency total
  - At 60 deg/s head motion, 1Kx1K, 100deg fov display:
    - 19 pixels error
    - Too much

Post-rendering image warp (time warp)

- To minimize end-to-end latency
- The method:
  - get current camera pose
  - render into a larger raster than the screen buffer
  - get new camera pose
  - warp rendered image using the latest pose, send to the display
    - 2D image translation
    - 2D image warp
    - 3D image warp
- Original paper from Mark et al. 1997, also Darsa et al. 1997
  - Meta: Asynchronous Time Warp

Eye movement - basics

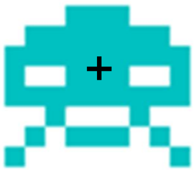
Fixation



Drift: 0.15-0.8 deg/s

Eye movement - basics

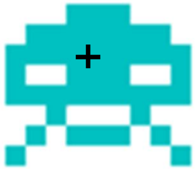
Saccade



160-300 deg/s

Eye movement - basics

Smooth Pursuit Eye Motion (SPEM)

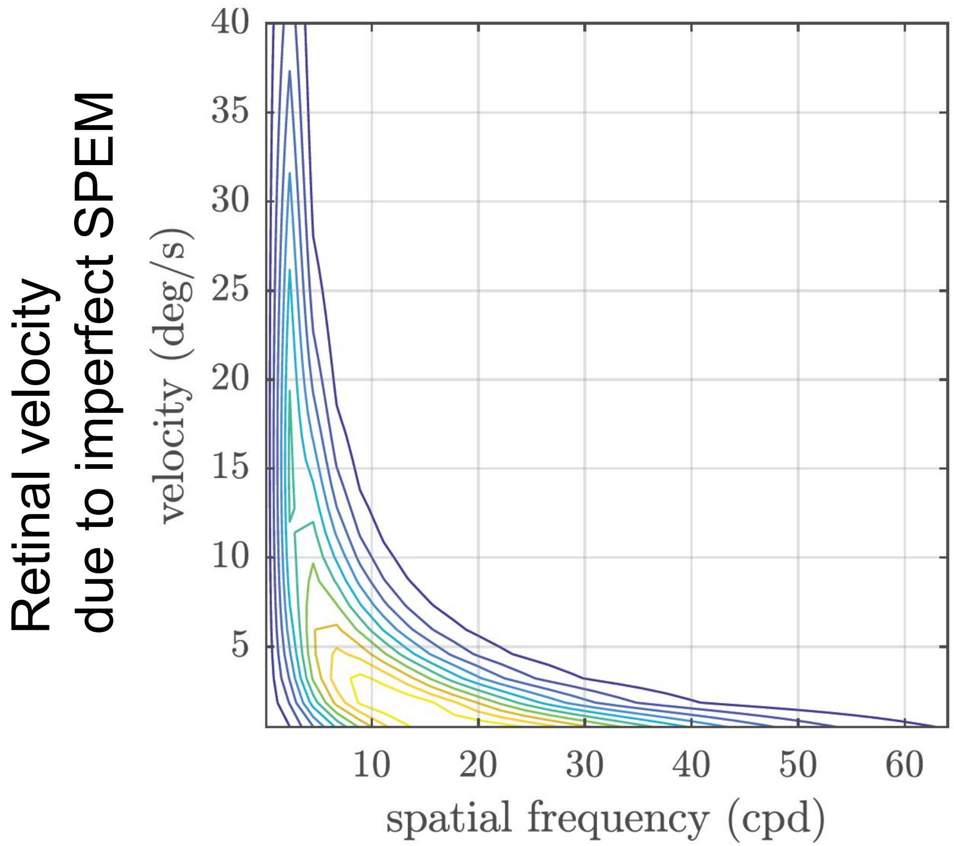


Up to 80 deg/s
The tracking is imperfect
- especially at higher velocities
- and for unpredictable motion

Retinal velocity

- The eye tracks moving objects
  - Smooth Pursuit Eye Motion (SPEM) stabilizes images on the retina
  - But SPEM is imperfect
- Loss of sensitivity mostly caused by imperfect SPEM
  - SPEM worse at high velocities

Spatio-velocity contrast sensitivity



Kelly's model [1979]

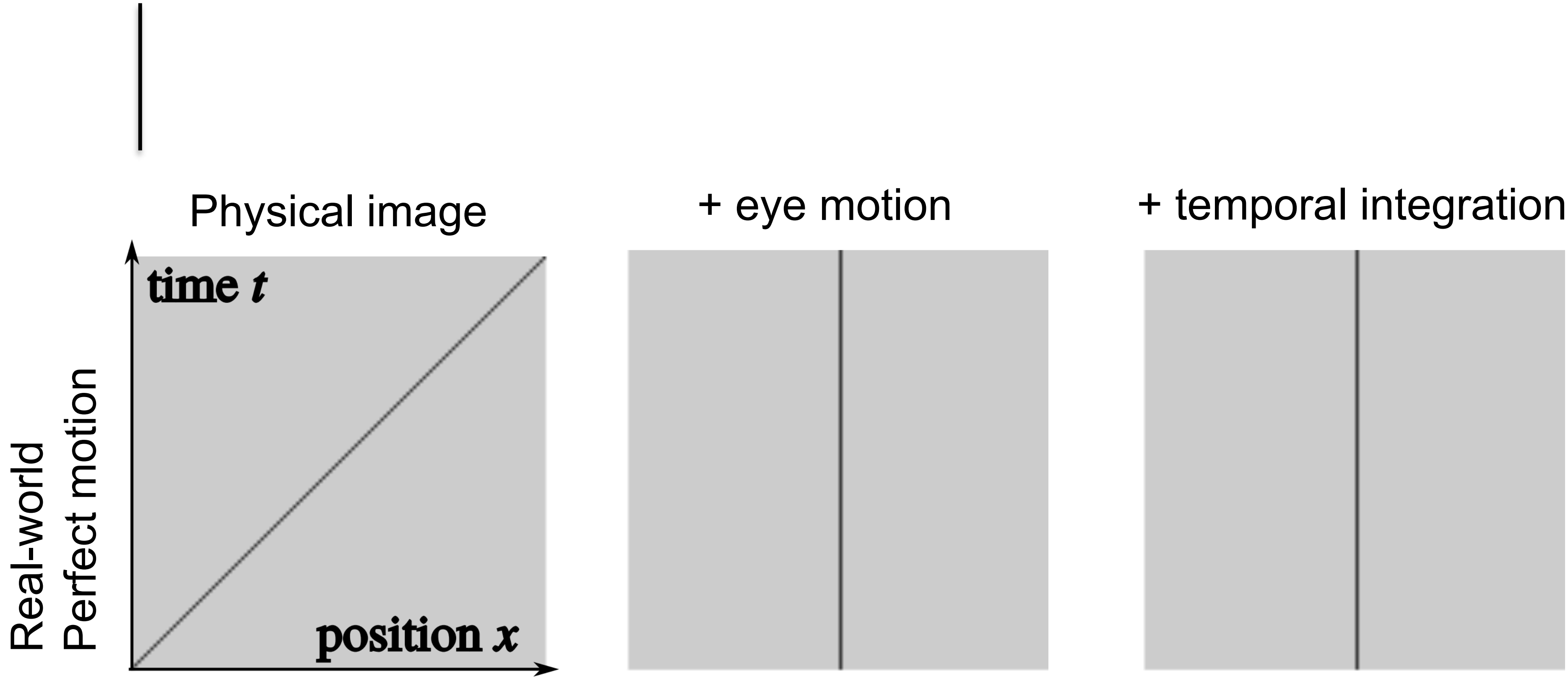
Motion sharpening

- The visual system "sharpens" objects moving at speeds of 6 deg/s or more

  - Potentially a reason why VR appears sharper than it actually is

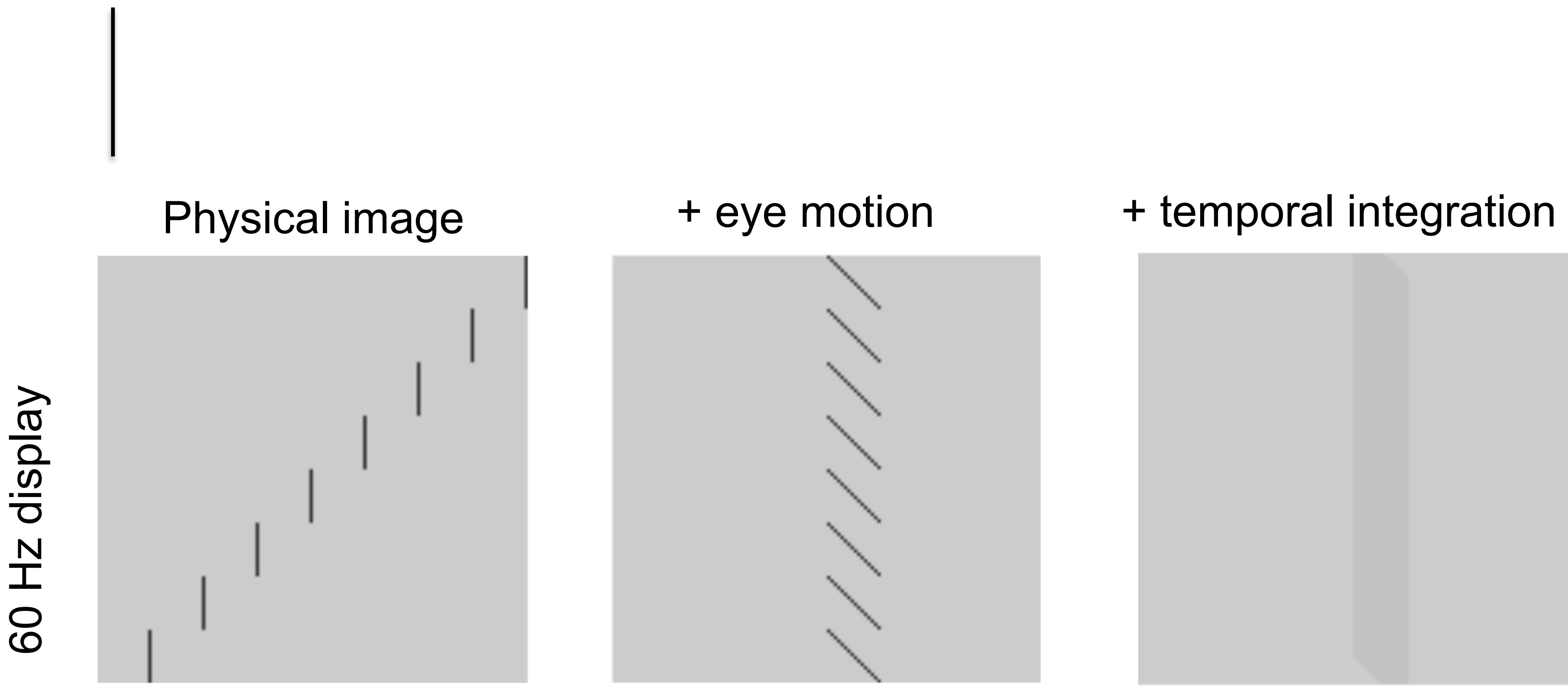
Hold-type blur

- The eye smoothly follows a moving object
- But the image on the display is "frozen" for 1/60th of a second



Hold-type blur

- The eye smoothly follows a moving object
- But the image on the display is "frozen" for 1/60th of a second

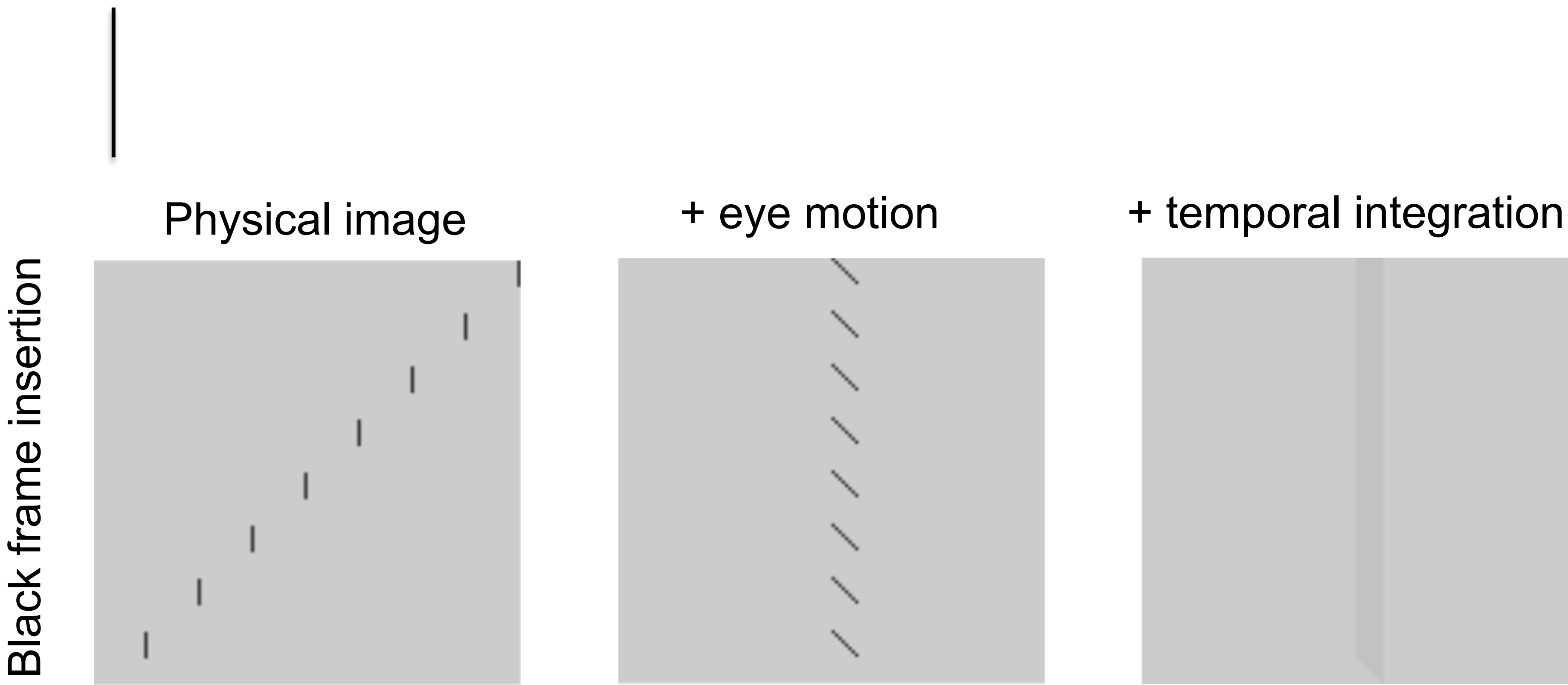


Original scene

With hold-type blur

Hold-type blur

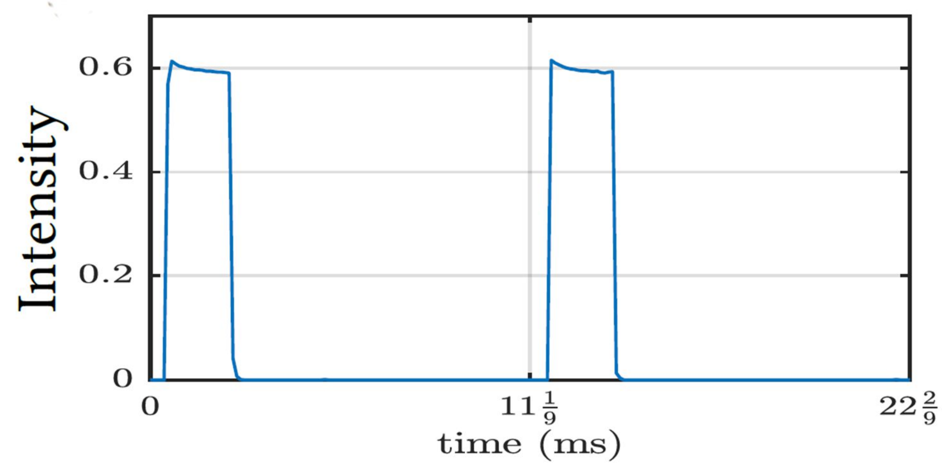
- The eye smoothly follows a moving object
- But the image on the display is "frozen" for 1/60th of a second



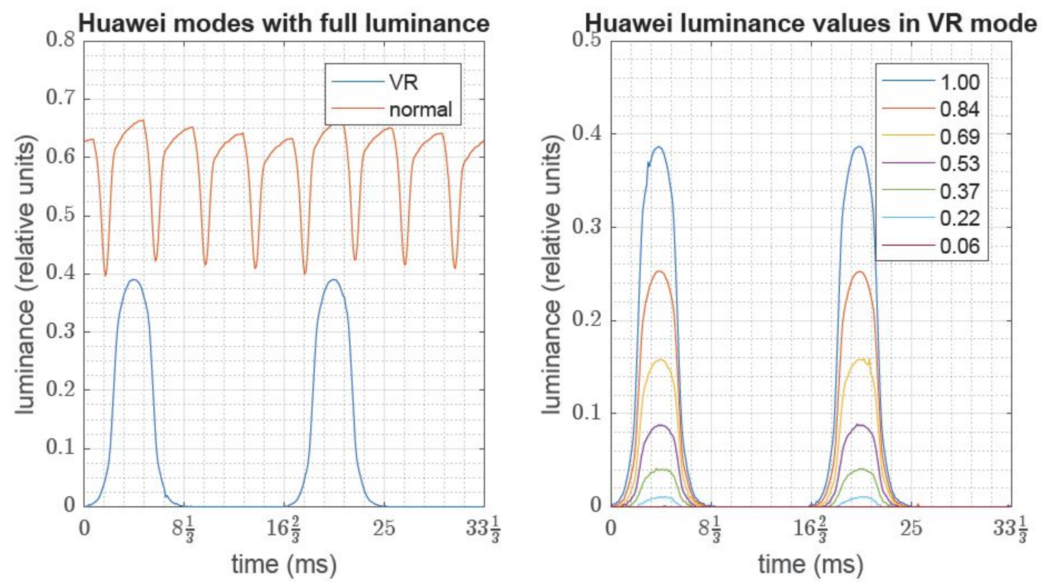
Low persistence displays

- Most VR displays flash an image for a fraction of frame duration
- This reduces hold-type blur
- And also reduces the perceived lag of the rendering

HTC Vive



Mate 9 Pro + DayDream



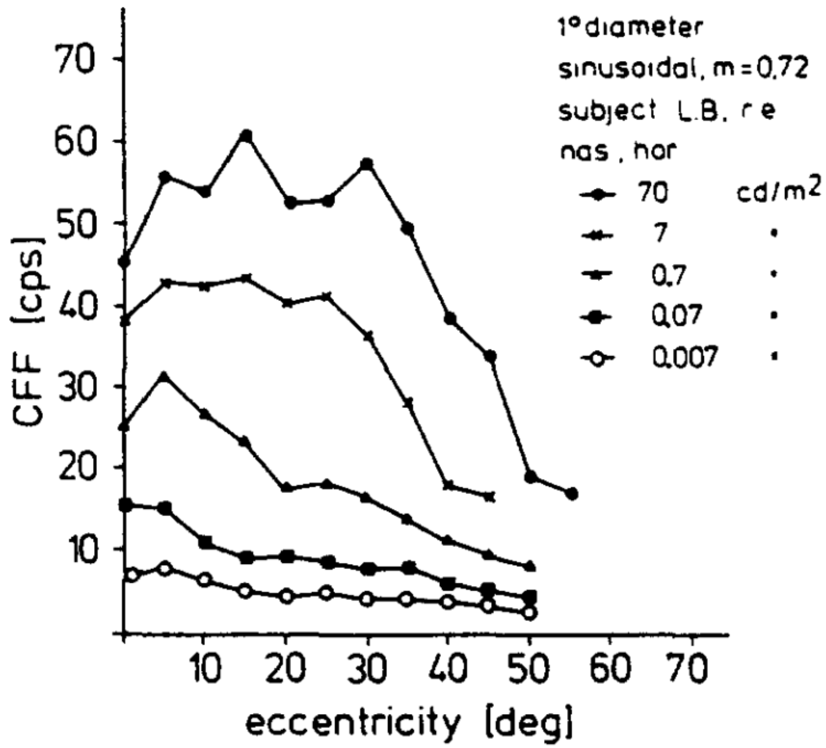
Black frame insertion

- Which invader appears sharper?

- A similar idea to low-persistence displays in VR
- Reduces hold-type blur

Flicker

- Critical Flicker Frequency
  - The lowest frequency at which flickering stimulus appears as a steady field
  - Measured for full-on / off presentation
  - Strongly depends on luminance – big issue for HDR VR headsets
  - Varies with eccentricity and stimulus size
  - It is possible to detect flicker even at 2kHz
    - For saccadic eye motion

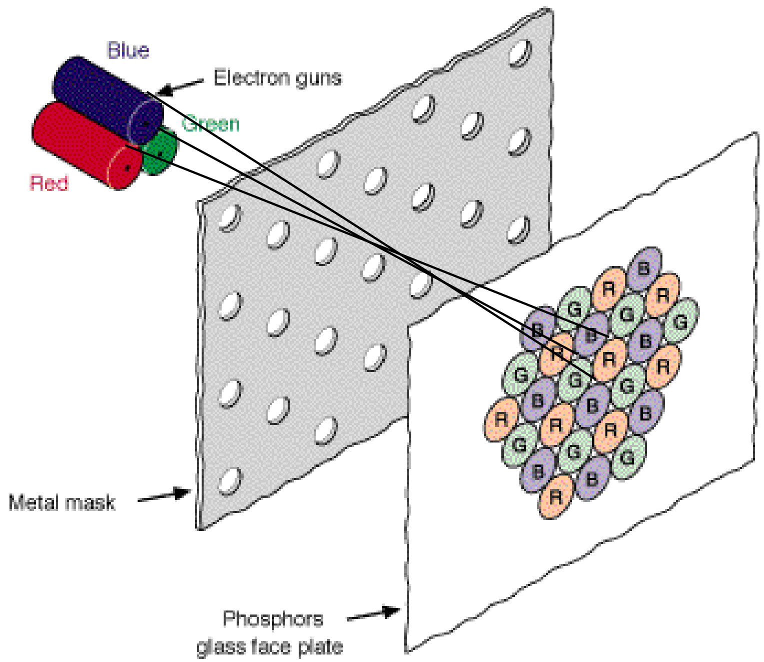


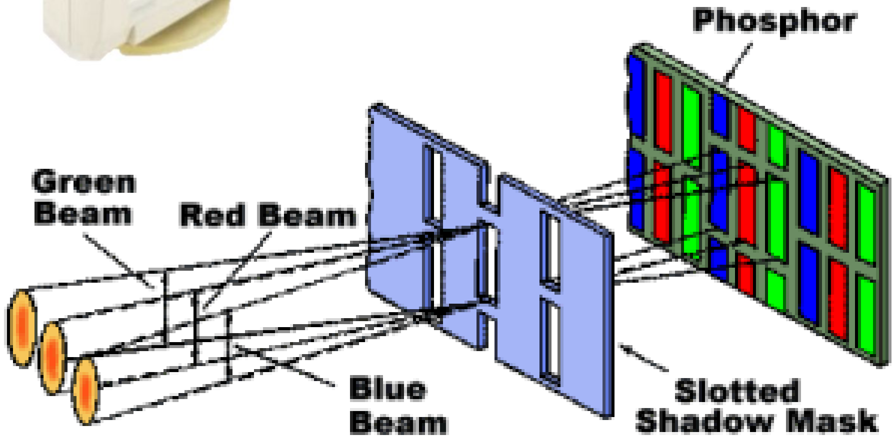
[Hartmann et al. 1979]

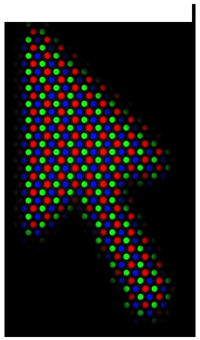
Overview

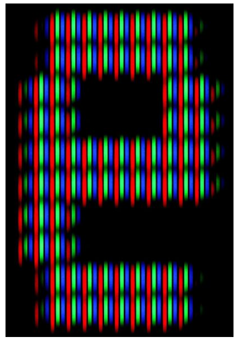
- Temporal aspects
  - Latency in VR
  - Eye-movement
  - Hold-type blur
- 2D displays
  - 2D spatial light modulators
  - High dynamic range displays

Cathode Ray Tube





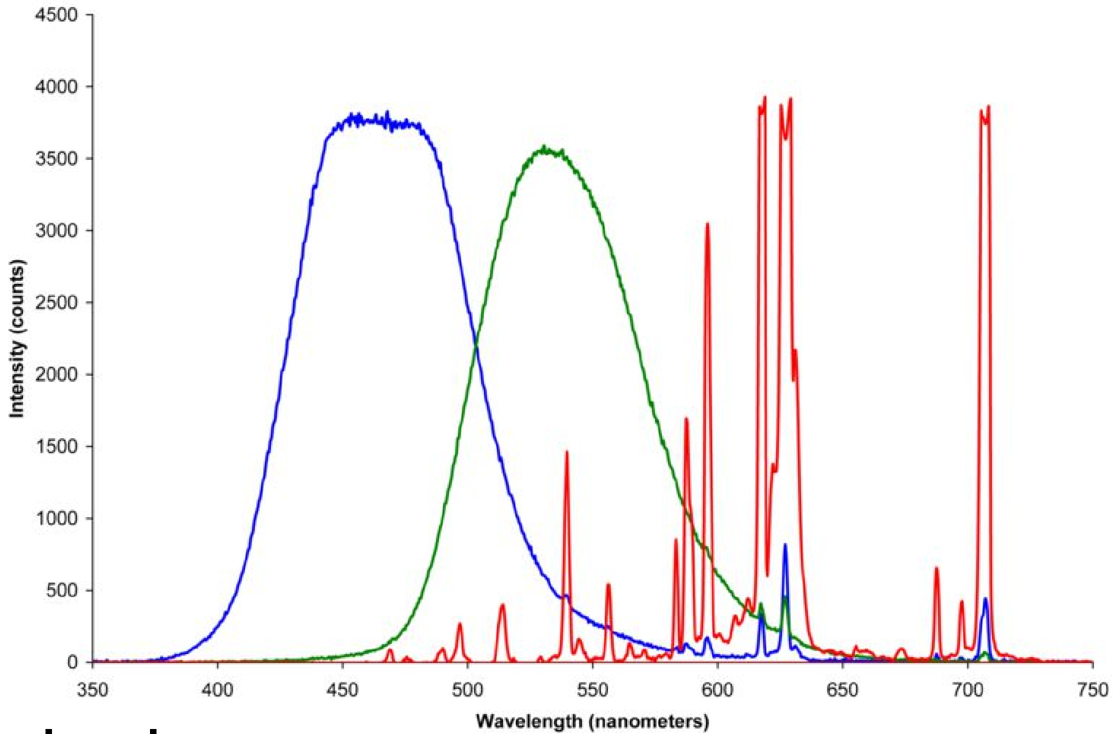




[from wikipedia]

Spectral Composition

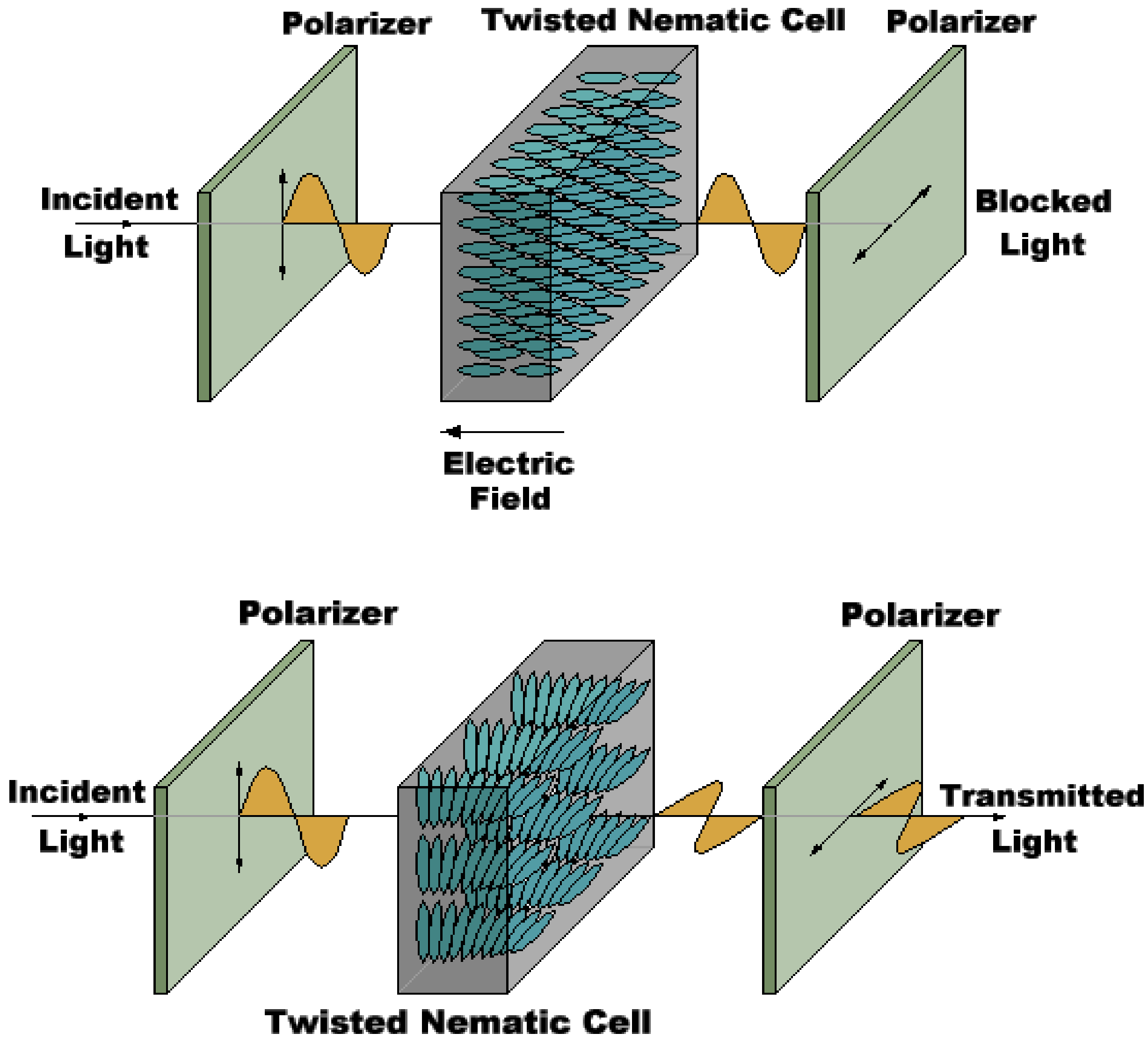
- three different phosphors



- saturated and natural colors
- inexpensive
- high contrast and brightness

[from wikipedia]

Liquid Chrystal Displays (LCD)



From: http://computer.howstuffworks.com/monitor5.htm

Twisted neumatic LC cell

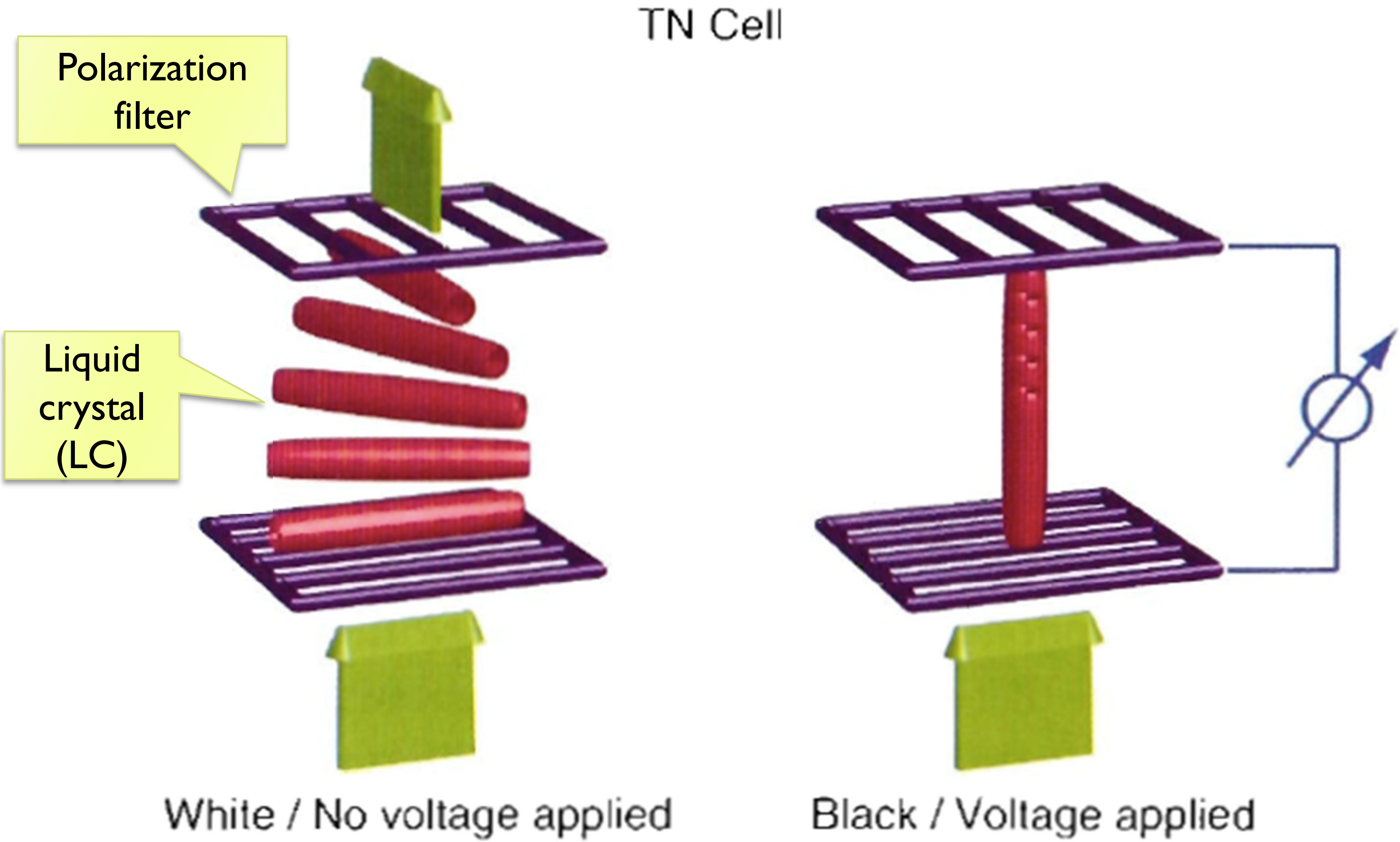


Figure from: High Dynamic Range Imaging by E. Reinhard et al.

In-plane switching cell (IPS)

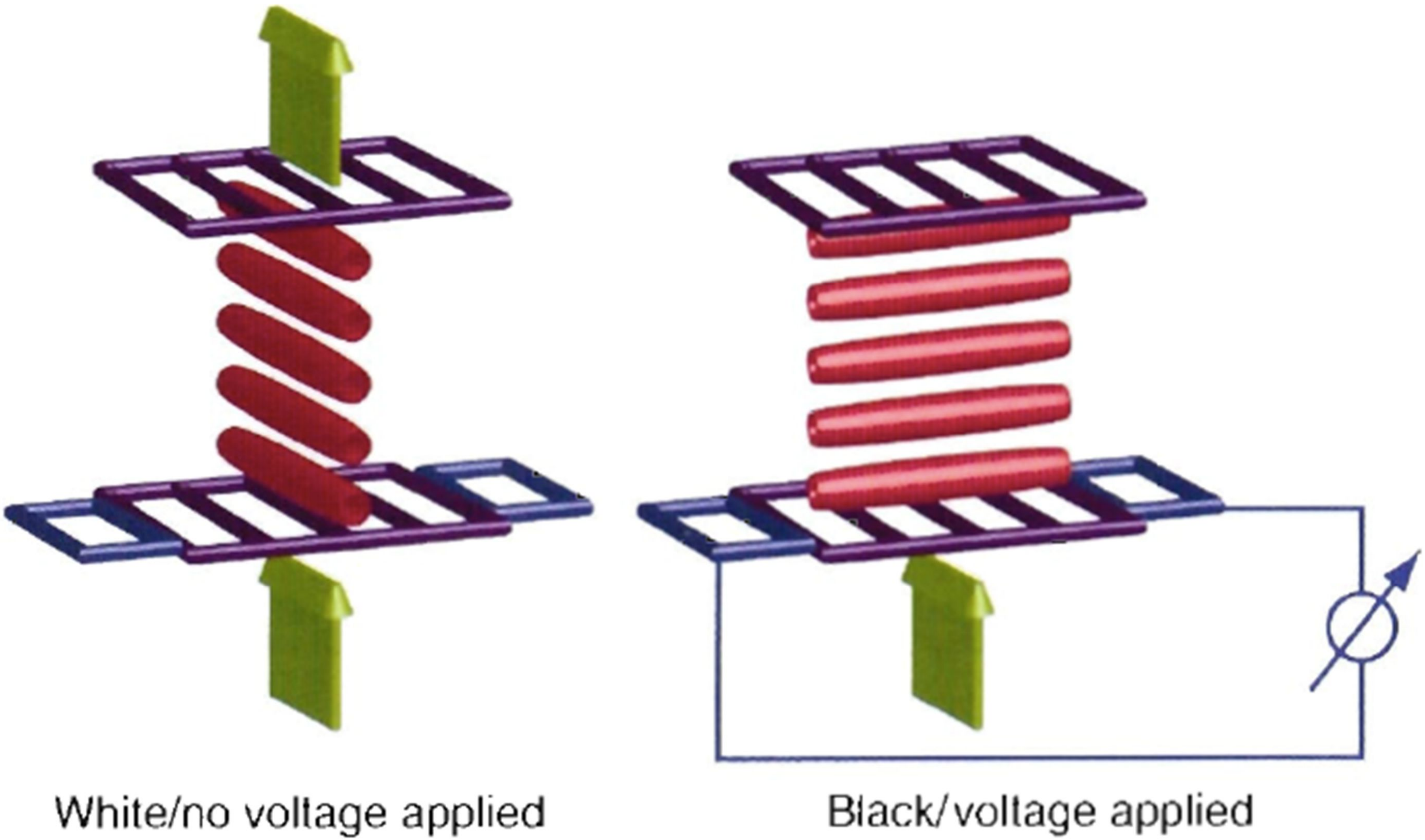
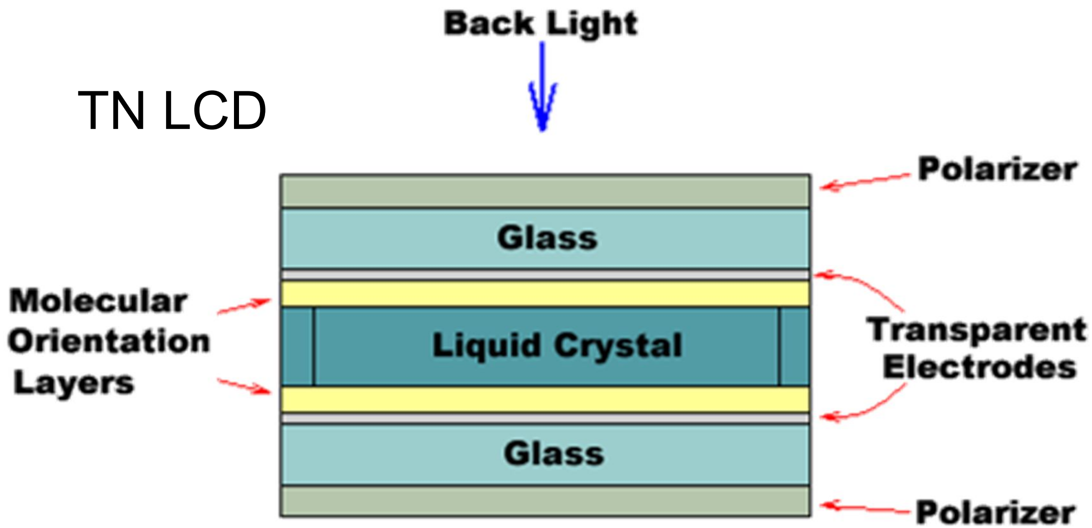
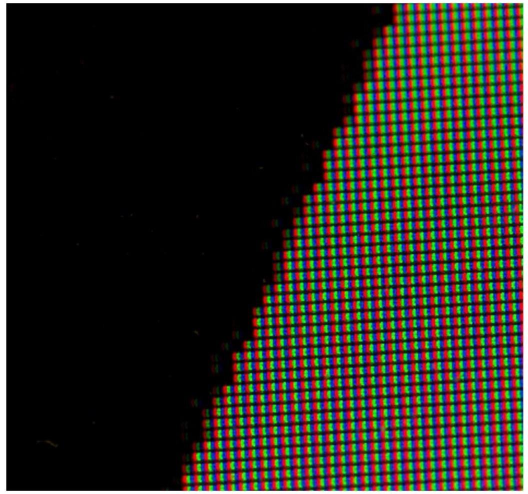


Figure from: High Dynamic Range Imaging by E. Reinhard et al.

LCD

TN LCD



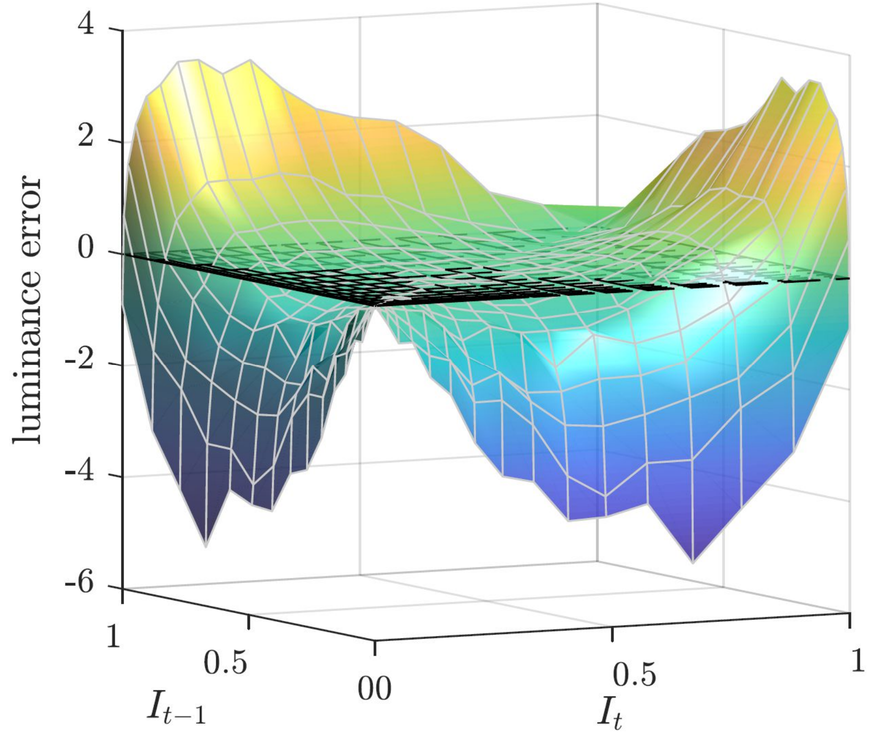


A white slanted edge (photograph)

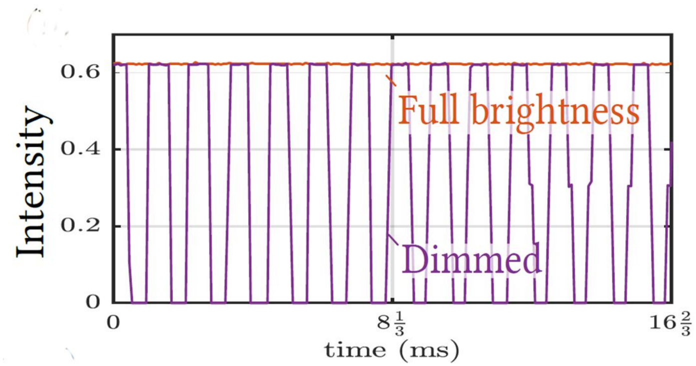
- color may change with the viewing angle
- contrast up to 3000:1
- higher resolution results in smaller fill-factor
- color LCD transmits only up to 8% (more often close to 4-5%) light when set to full white

LCD temporal response

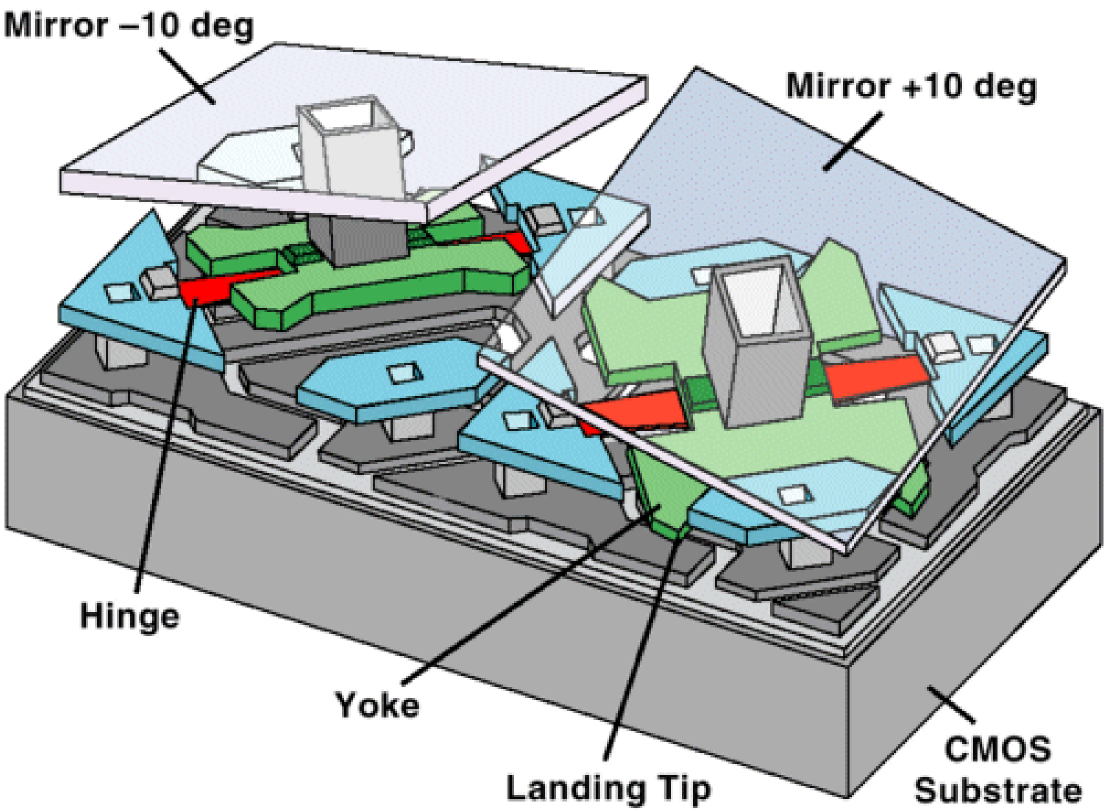
- Experiment on an IPS LCD screen
- We rapidly switched between two intensity levels at 120Hz
- Measured luminance integrated over 1s
- The top plot shows the difference between expected ($\frac{I_{t-1}+I_t}{2}$) and measured luminance

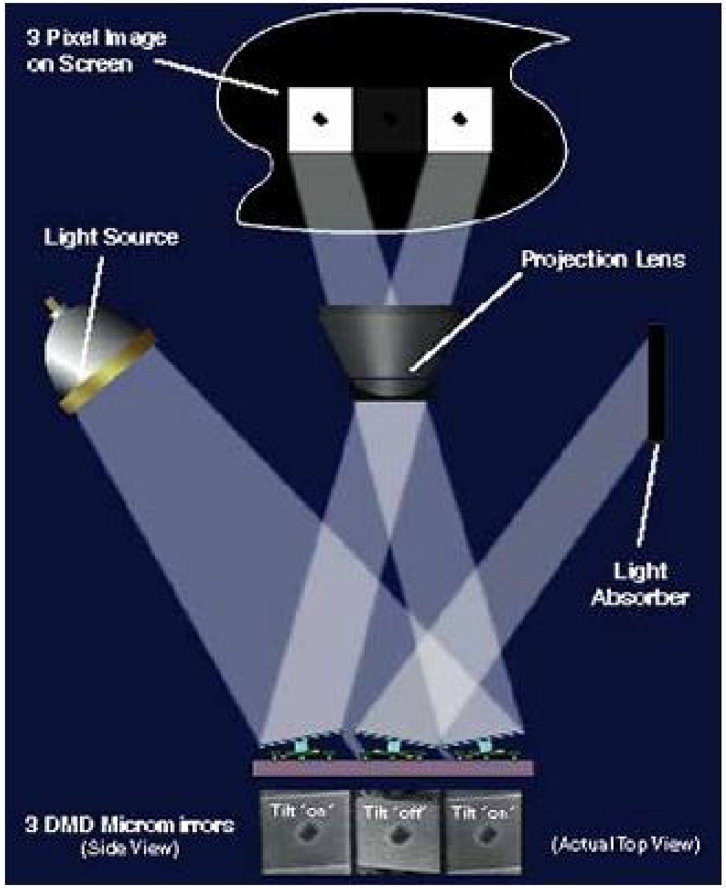


- The bottom plot: intensity measurement for the full brightness and half-brightness display settings
  - Pulse-Width Modulation controls brightness of the backlight



Digital Micromirror Devices (DMDs/DLP)

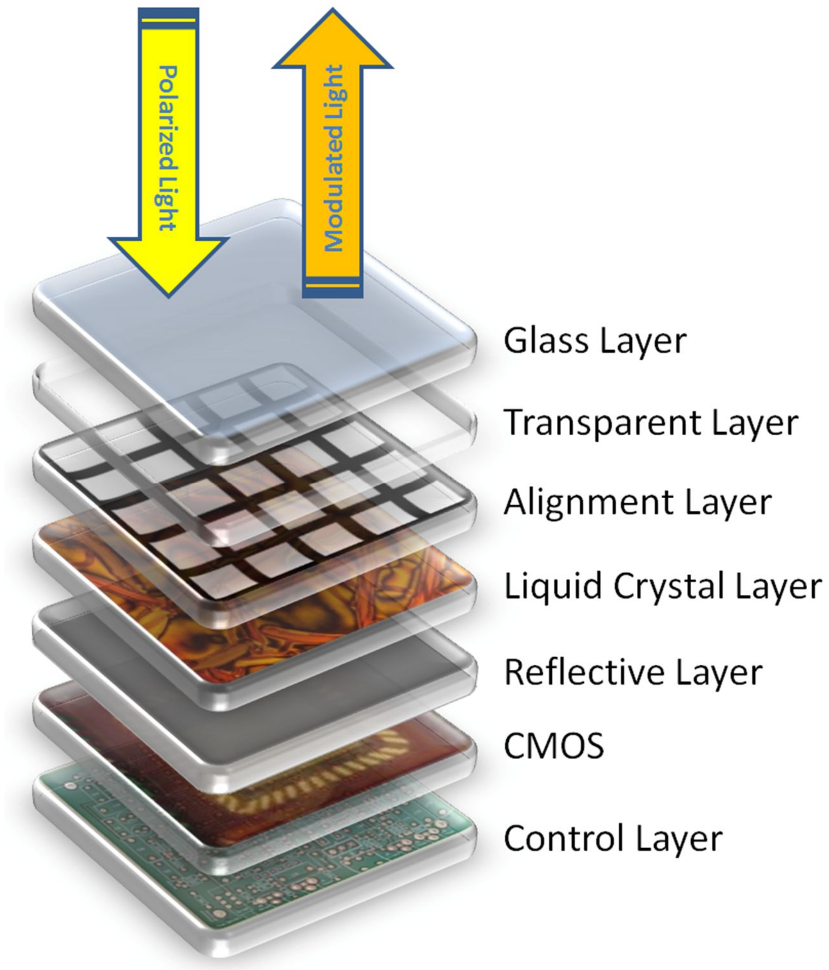




Texas Instruments

- 2-D array of mirrors
- Truly digital pixels
- Grey levels via Pulse-Width Modulation

Liquid Crystal on Silicon (LCoS)

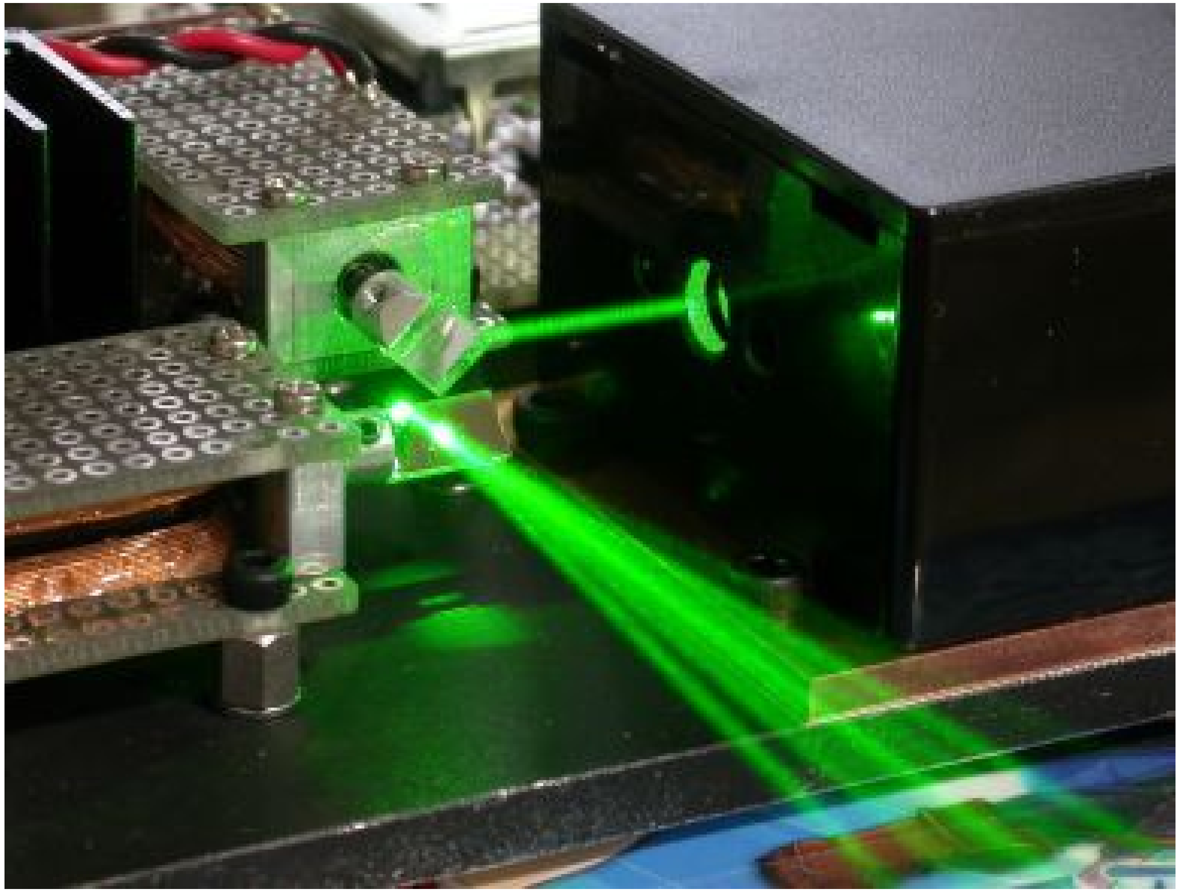


- basically a reflective LCD
- standard component in projectors and head mounted displays
- used e.g. in Google Glass

Scanning Laser Projector

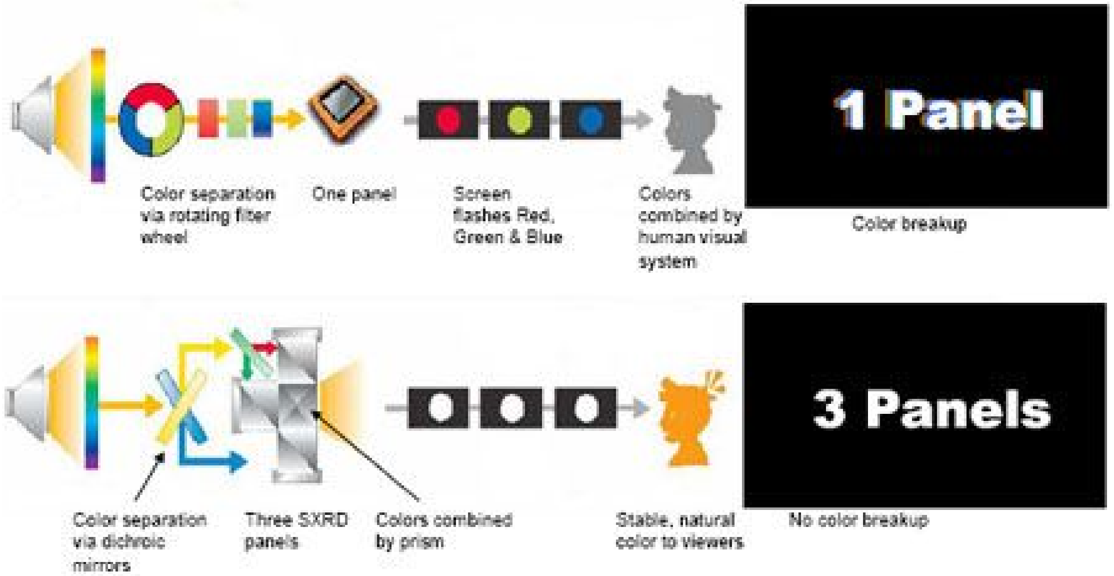
- maximum contrast
- scanning rays

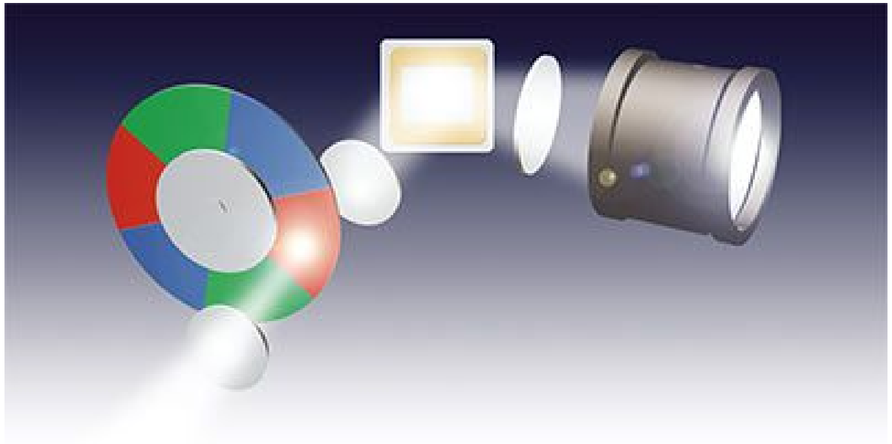
- very high power lasers needed for high brightness



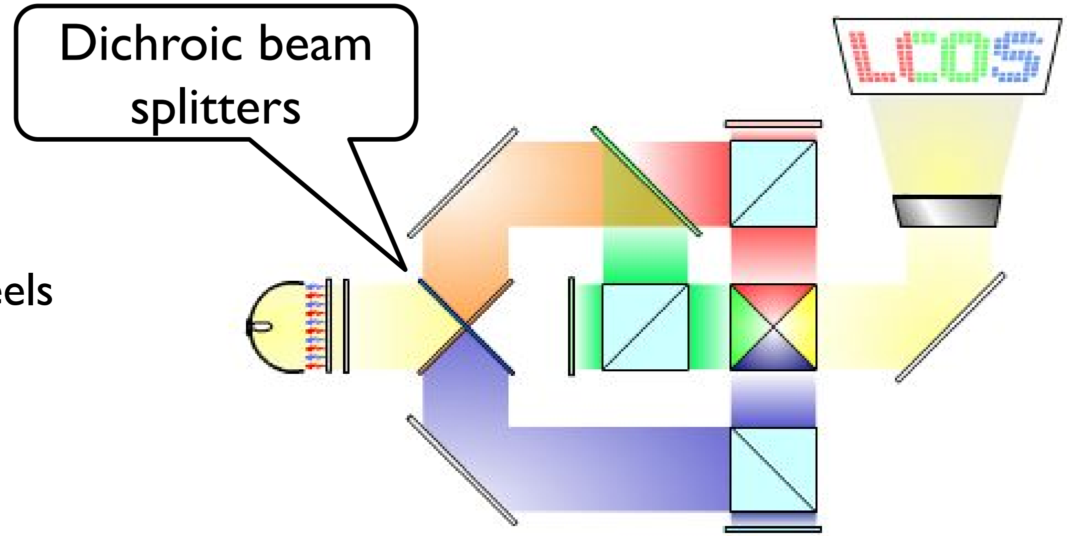
http://elm-chan.org/works/vlp/report_e.html

3-chip vs. Color Wheel Display



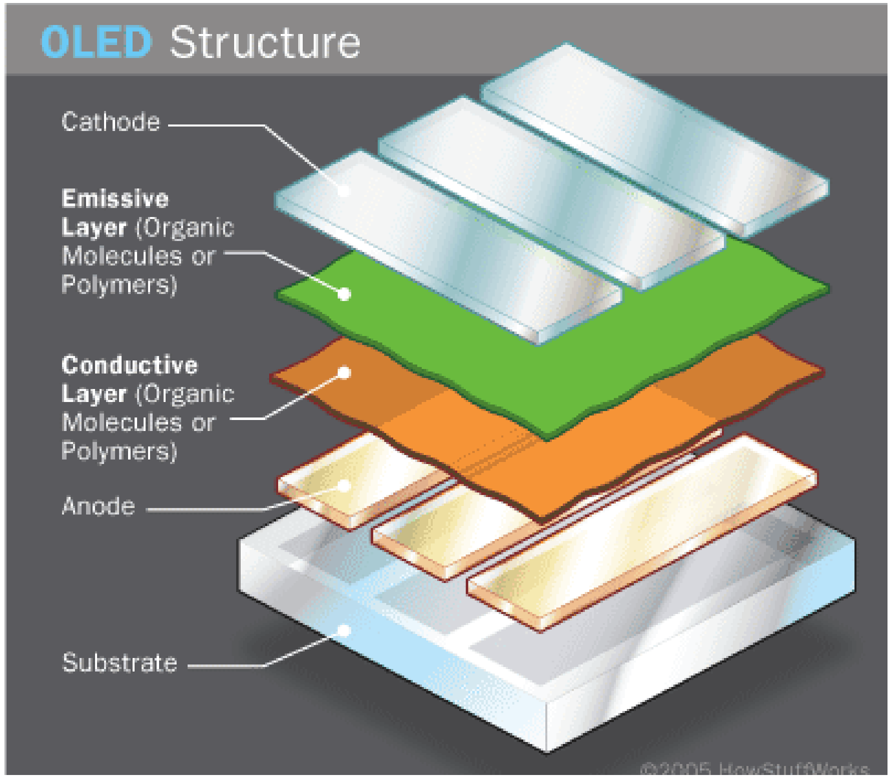


- color wheel
  - cheap
  - time sequenced colors
  - color fringes with motion/video
    - mitigated with advanced colour wheels
- 3-chip
  - complicated setup
  - no color fringes



OLED

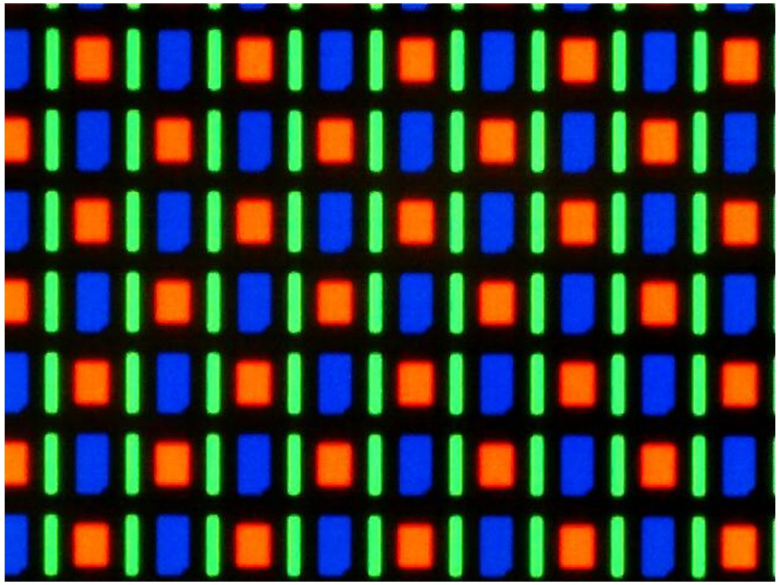
- based on electrophosphorescence
- large viewing angle
- the power consumption varies with the brightness of the image
- fast (< 1 microsec)
- arbitrary sizes



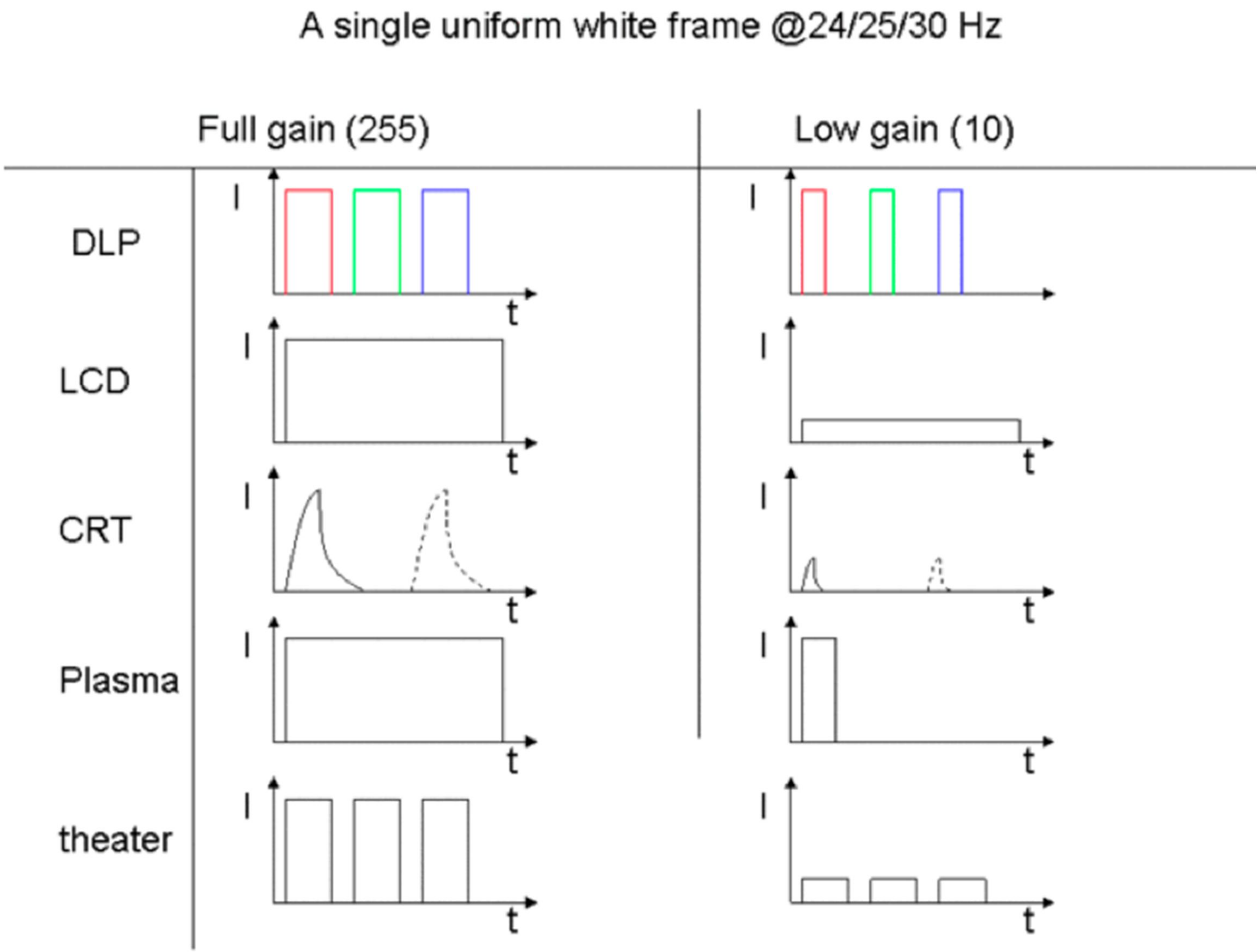
- life-span can be short
  - Worst for blue OLEDs

Active matrix OLED

- Commonly used in mobile phones (AMOLED)
- Very good contrast
  - But the screen more affected by glare than LCD
- But limited brightness
  - The brighter is OLED, the shorter is its live-span

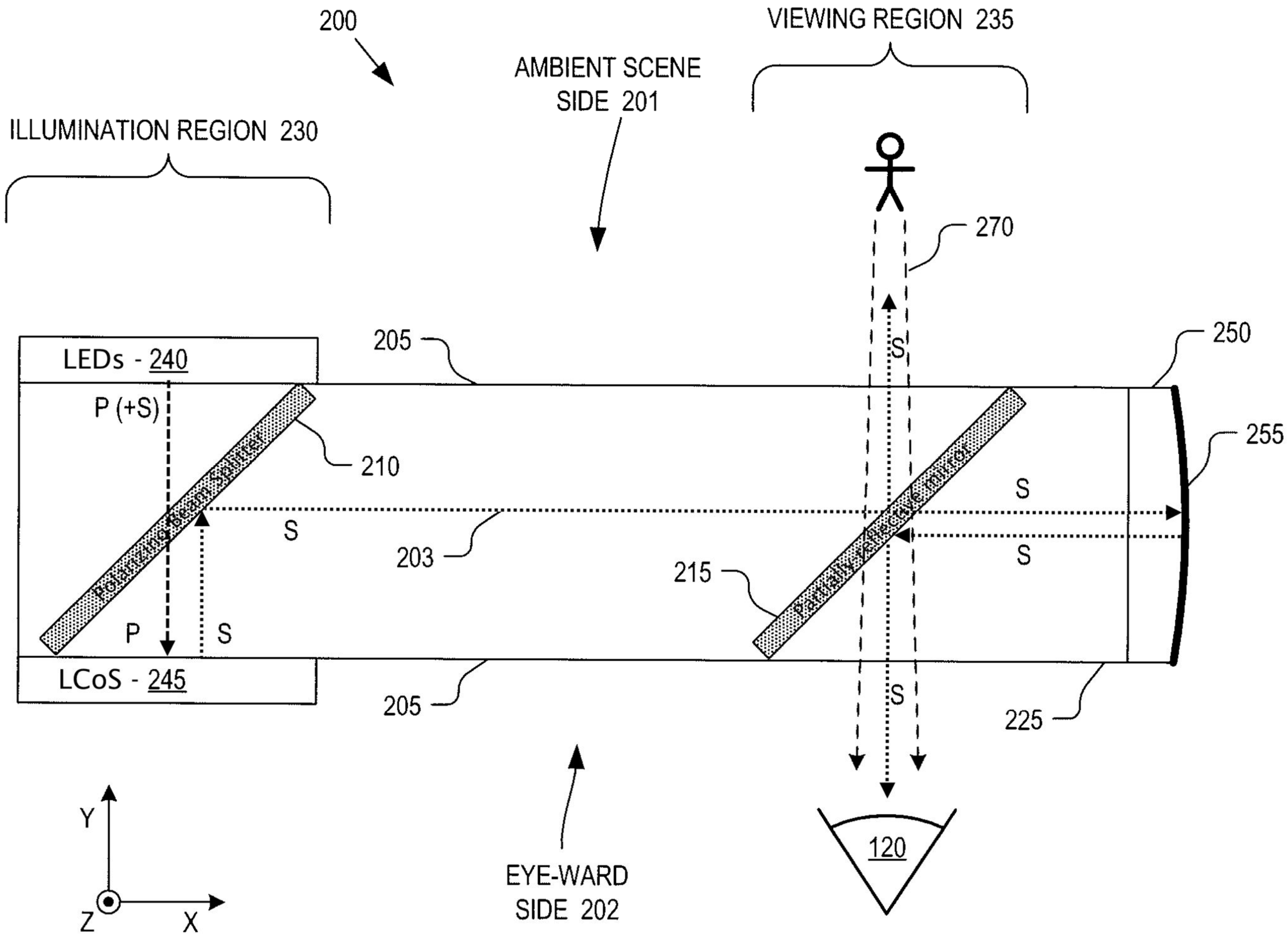


Temporal characteristic



From: http://en.wikipedia.org/wiki/Comparison_of_display_technology

Bird-bath optics for near-eye displays



Google Glass

Pros:

- Simple, efficient design

Cons:

- Cannot be scaled up easily

More reading: https://kguttag.com/2017/03/03/near-eye-bird-bath-optics-pros-and-cons-and-immys-different-approach/

Diffractive waveguides

US 2016/0116739

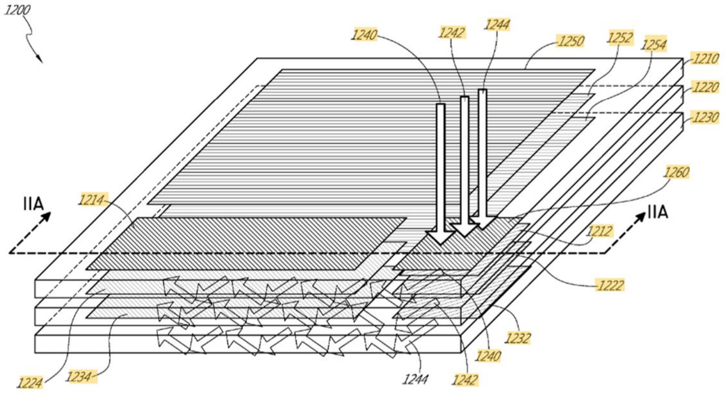
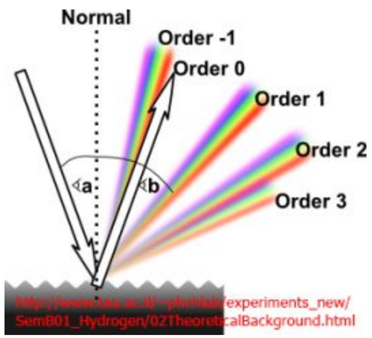
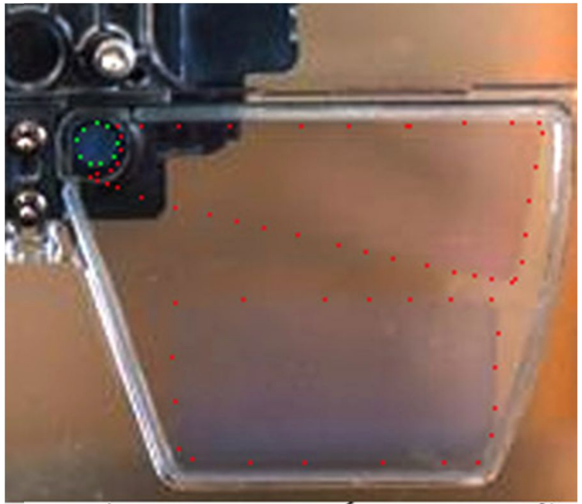
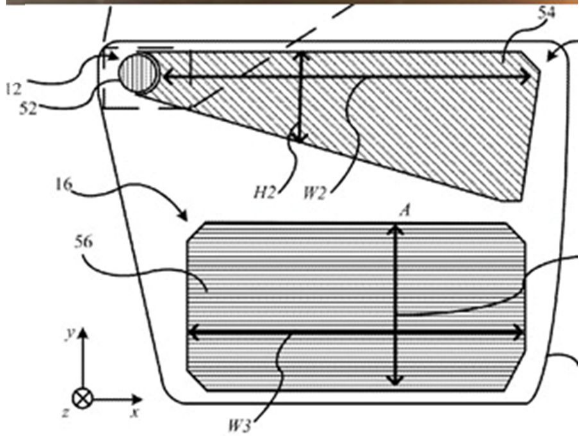


FIG. IIB

Magic Leap



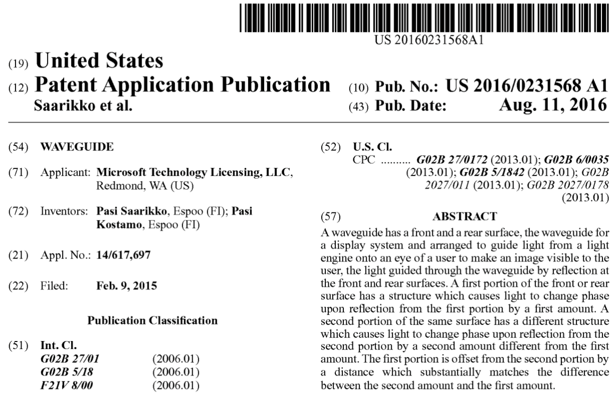




US 2016/0231568

Fig. 3B

Microsoft Hololens


US 20160231568A1

(19) **United States**
(12) **Patent Application Publication** (10) **Pub. No.: US 2016/0231568 A1**
Saarikko et al. (43) **Pub. Date: Aug. 11, 2016**

(54) **WAVEGUIDE**

(71) Applicant: **Microsoft Technology Licensing, LLC**, Redmond, WA (US)

(72) Inventors: **Pasi Saarikko**, Espoo (FI); **Pasi Kostamo**, Espoo (FI)

(21) Appl. No.: **14/617,697**

(22) Filed: **Feb. 9, 2015**

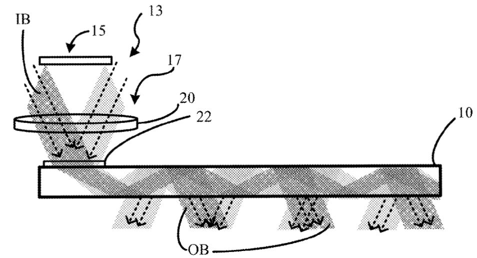
Publication Classification

(51) **Int. Cl.**
G02B 27/01 (2006.01)
G02B 5/18 (2006.01)
F21V 8/00 (2006.01)

(52) **U.S. Cl.**
CPC ***G02B 27/0172*** (2013.01); ***G02B 6/0035*** (2013.01); ***G02B 5/1842*** (2013.01); *G02B 2027/011* (2013.01); *G02B 2027/0178* (2013.01)

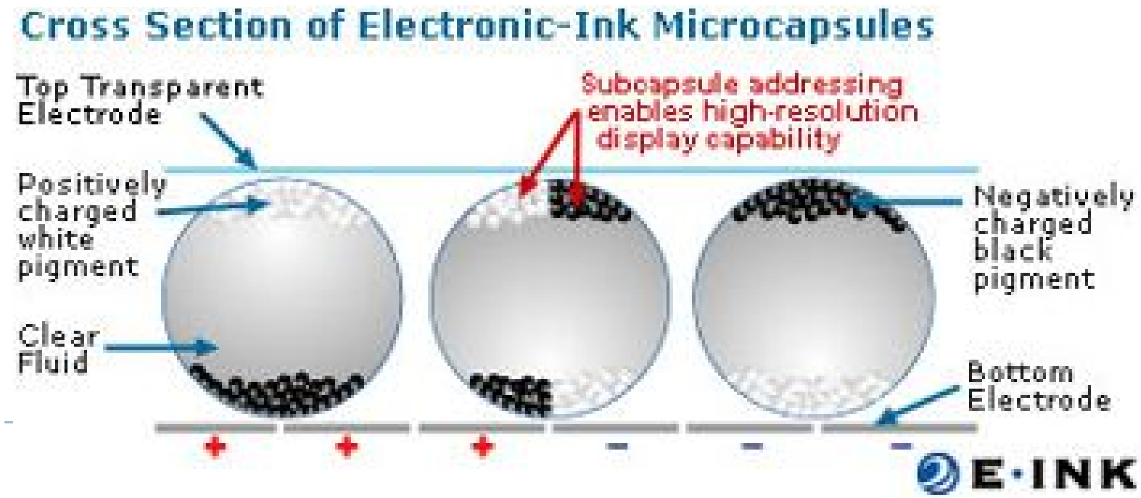
(57) **ABSTRACT**

A waveguide has a front and a rear surface, the waveguide for a display system and arranged to guide light from a light engine onto an eye of a user to make an image visible to the user, the light guided through the waveguide by reflection at the front and rear surfaces. A first portion of the front or rear surface has a structure which causes light to change phase upon reflection from the first portion by a first amount. A second portion of the same surface has a different structure which causes light to change phase upon reflection from the second portion by a second amount different from the first amount. The first portion is offset from the second portion by a distance which substantially matches the difference between the second amount and the first amount.

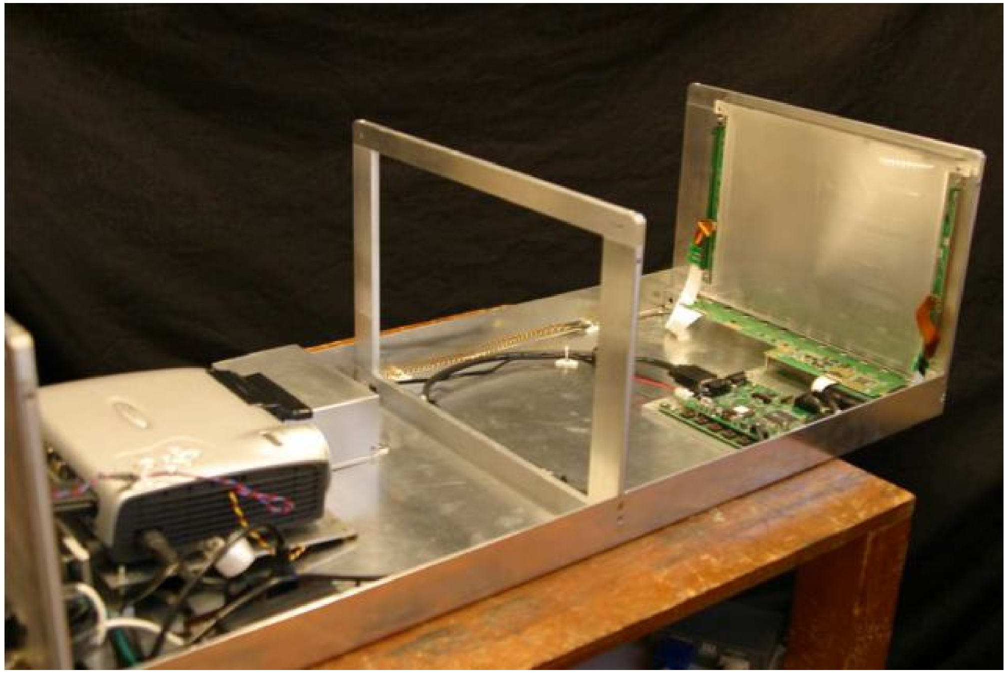


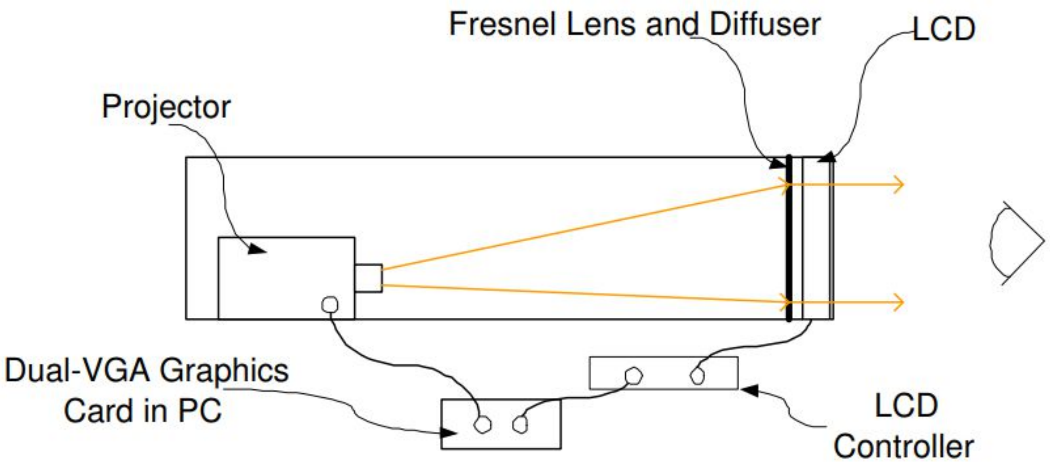
Electronic Paper

www.eink.com



Prototype HDR display (2004)

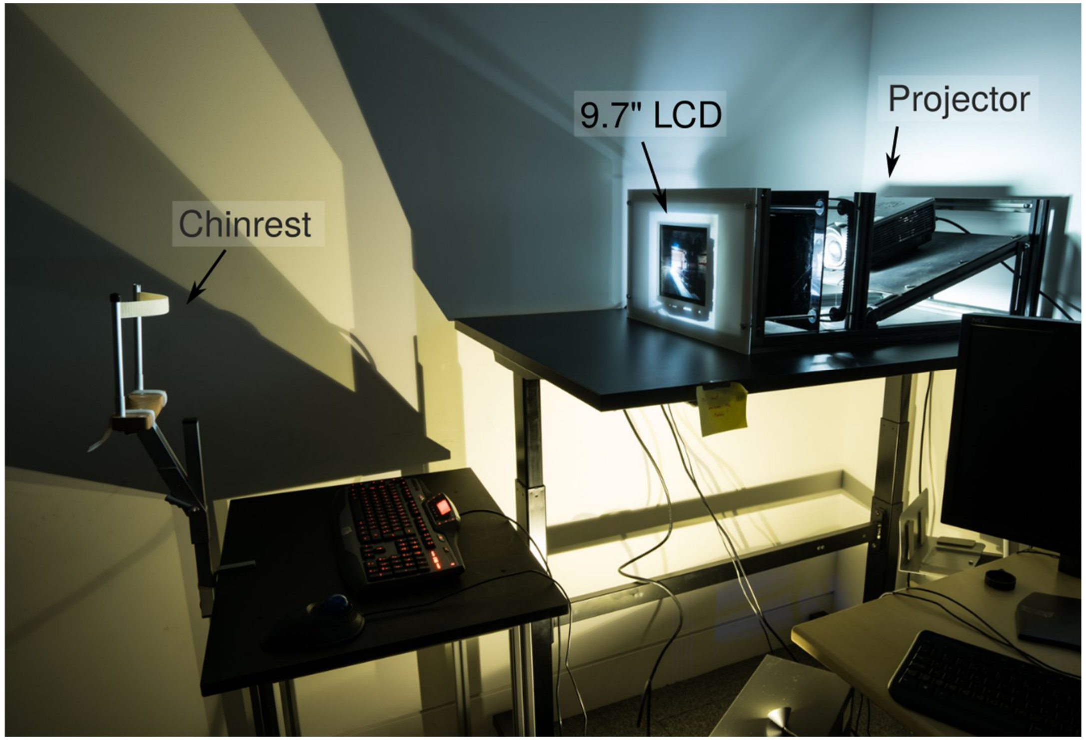


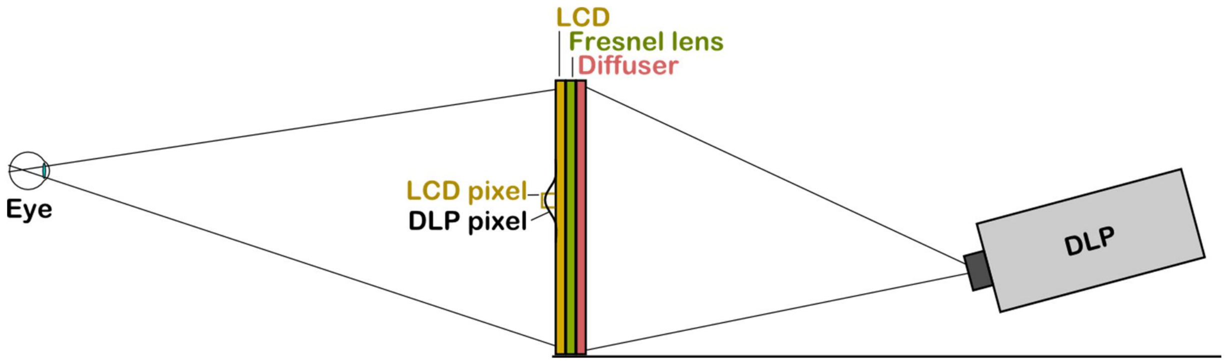


From [Seetzen et al. SIGGRAPH 2004]

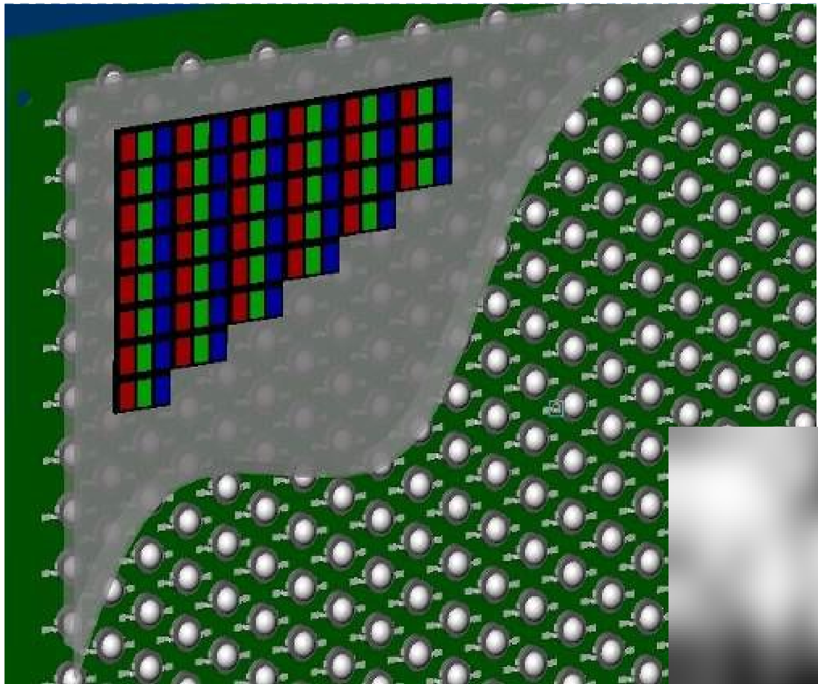
Cambridge experimental HDR display

- 35,000 cd/m^2 peak luminance
- 0.01 cd/m^2 black level
- LCD resolution: 2048x1536
- Backlight (DLP) resolution: 1024x768
- Geometric-calibration with a DSLR camera
- Display uniformity compensation
- Bit-depth of DLP and LCD extended to 10 bits using spatio-temporal dithering





Modern HDR displays



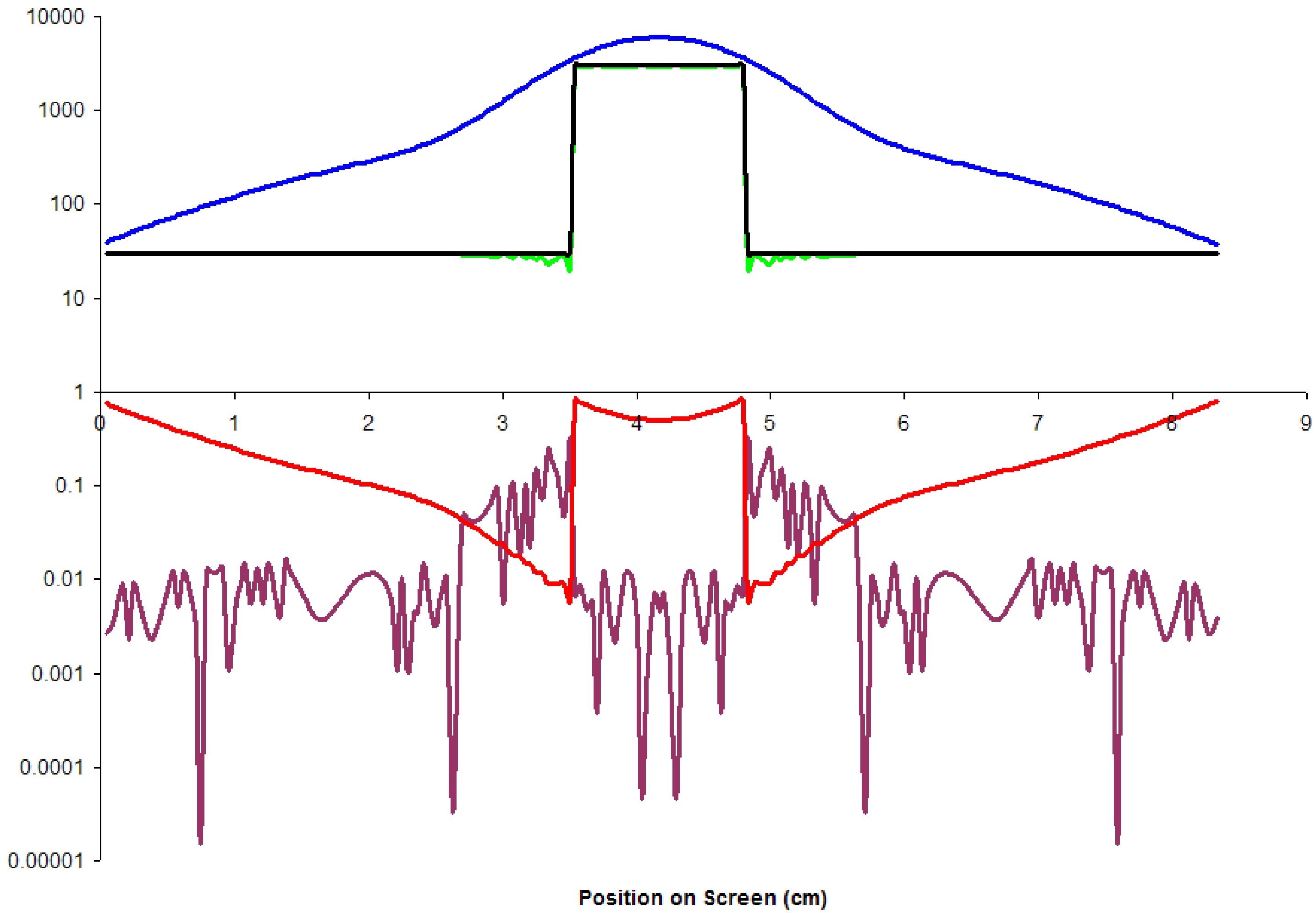
- Modulated LED array
- Conventional LCD
- Image compensation

Low resolution LED Array x High resolution Colour Image = High Dynamic Range Display

HDR Display

- Two spatial modulators
  - 1st modulator contrast 1000:1
  - 2nd modulator contrast 1000:1
  - Combined contrast 1000,000:1
- Idea: Replace constant backlight of LCD panels with an array of LEDs
  - Very few (about 1000) LEDs sufficient
  - Every LED intensity can be set individually
  - Very flat form factor (fits in standard LCD housing)
- Issue:
  - LEDs larger than LCD pixels
  - This limits maximum local contrast

Veiling Luminance



Receive Image

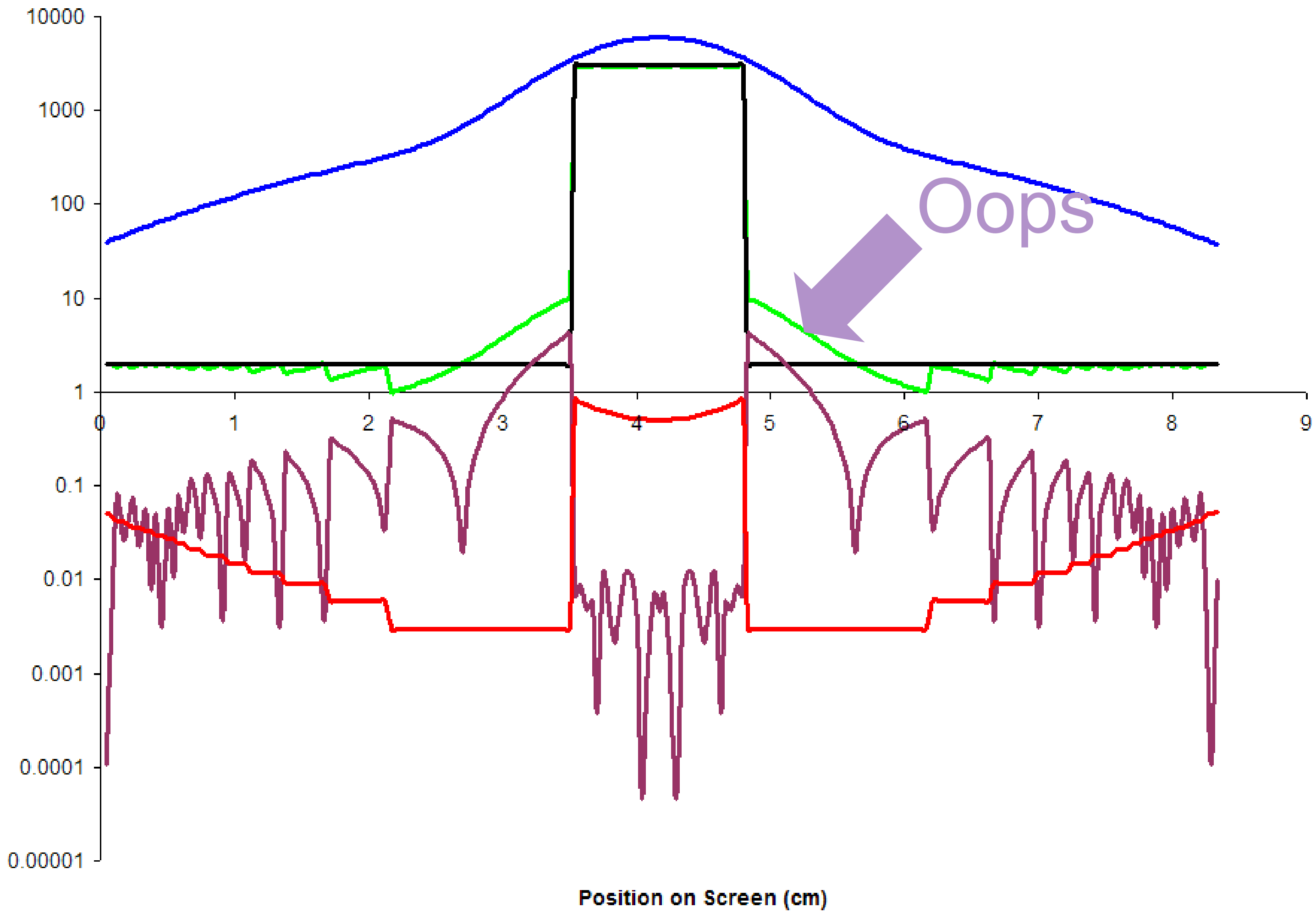
Drive LED

Divide Image by LED light field to obtain LCD values

Output Luminance is the product of LED light field and LCD transmission

(modest error)

Veiling Luminance



Receive Image

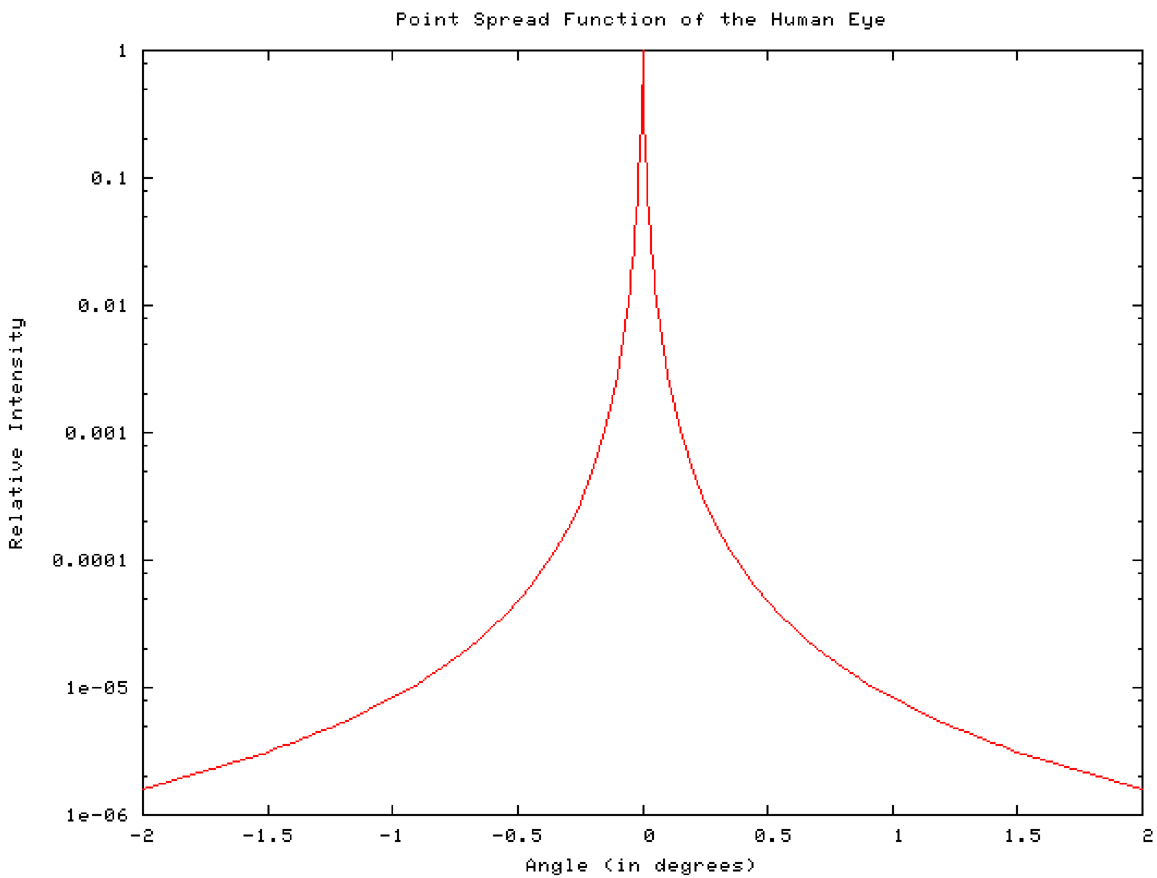
Drive LED

Divide Image by LED light field to obtain LCD values

Output Luminance is the product of LED light field and LCD transmission (Problematic error)

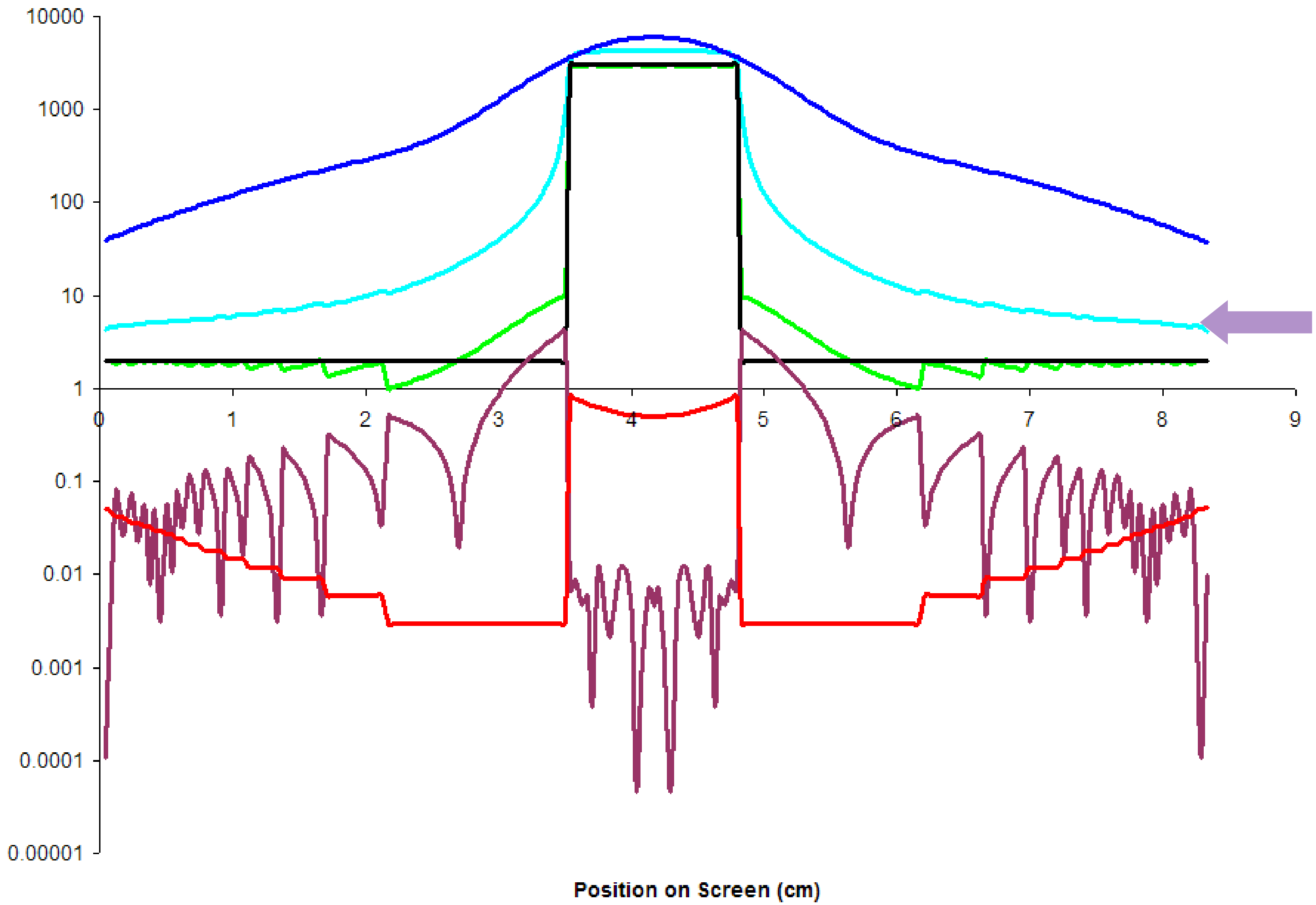
Veiling Luminance

- Maximum perceivable contrast
  - Globally very high (5-6 orders of magnitude)
    - That is why we create these displays!
  - Locally can be low: 150:1
- Point-spread function of human eye
  - Refer to „HDR and tone mapping" lecture
  - Consequence: high contrast edges cannot be perceived at full contrast



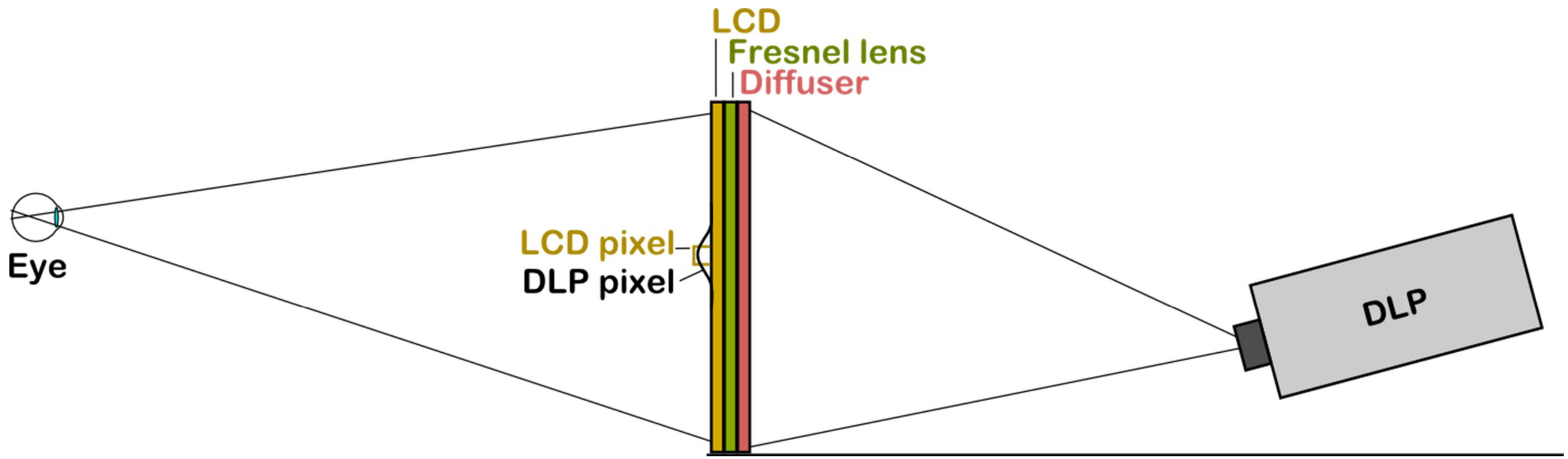
Veiling Glare (Camera)

Veiling Luminance



Veiling Luminance masks imperfection

HDR rendering algorithm - high level



Desired image

DLP blur (PSF)

$$\underset{L,D}{\operatorname{argmin}} \left|\left| I(x,y) - g * D(x,y)L(x,y) \right|\right|_2$$

DLP image

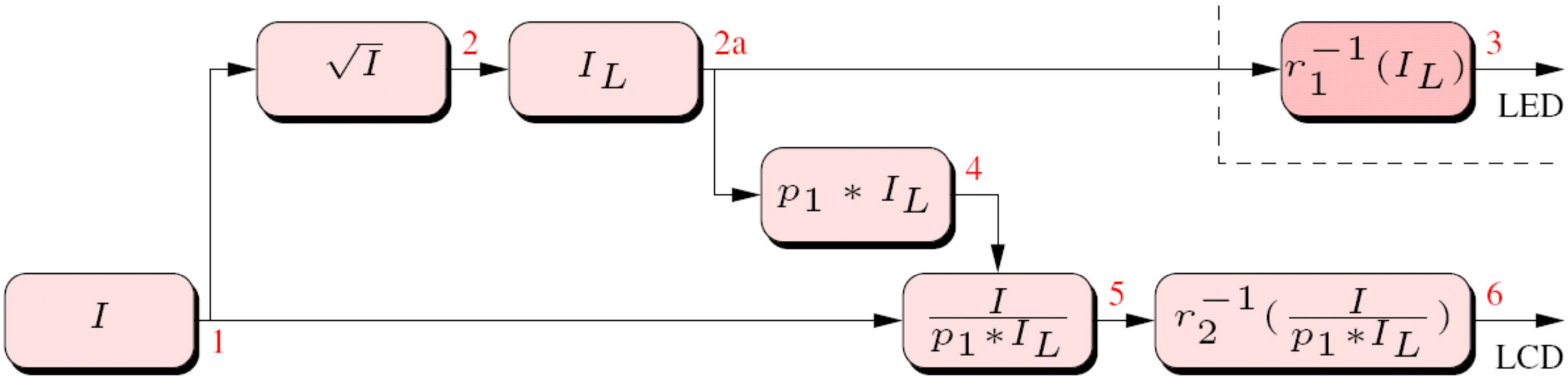
LCD image

Subject to:

$$\forall(x,y) \quad L_{min} \leq L(x,y) \leq L_{max}$$

$$\forall(x,y) \quad D_{min} \leq D(x,y) \leq D_{max}$$

Simplified HDR rendering algorithm



Rendering Algorithm

References

- HAINICH, R.R. AND BIMBER, O. 2011. *Displays: Fundamentals and Applications*. CRC Press.
- SEETZEN, H., HEIDRICH, W., STUERZLINGER, W., ET AL. 2004. High dynamic range display systems. *ACM Transactions on Graphics 23*, 3, 760.
- Visual motion test for high-frame-rate monitors:
  - https://www.testufo.com/

Advanced Graphics and Image Processing - Lecture notes

Rafał Mantiuk

Michaelmas term 2022/23

1 Contrast- and gradient-based methods

Many problems in image processing are easier to solve or produce better results if operations are not peformed directly on image pixel values but on differences between pixels. Instead of altering pixels, we can transform an image into gradient field and then edit the values in the gradient field. Once we are done with editing, we need to reconstruct an image from the modified gradient field.

A few examples of gradient-based methods are shown in Figures 1 and 2.

In one common case such differences between pixels represent gradients: for image I, a gradient at a pixel location (x, y) is computed as:

$$\nabla I_{x,y} = \begin{bmatrix} I_{x+1,y} - I_{x,y} \\ I_{x,y+1} - I_{x,y} \end{bmatrix}. \tag{1}$$

The equation above is obviously a discrete approximation of a gradient, as we are dealing with discrete pixel values rather than a continous function. This particular approximation is called forward difference, as we take the difference between the next and current pixel. Other choices include backward differences (current minus previous pixel) or central differences (next minus previous pixel).

Once a gradient field is computed, we can start modifying it. Usually better effects are achieved if the magnitude of gradients is modified and the orientation of each gradient remains unchanged. This can be achieved by

(a) Original image

(b) Details enhanced

(c) Cartoonized image

Figure 1: Two examples of gradient-based processing. Texture details in the original image were enhanced to produce the result shown in (b). Contrast was removed everywhere except at the edges to produced a cartoonized image in (c).

multiplying gradients by the gradient editing function $f()$:

$$G_{x,y} = \nabla I_{x,y} \cdot \frac{f\left(||\nabla I_{x,y}||\right)}{||\nabla I_{x,y}|| + \epsilon} \tag{2}$$

where $||\cdot||$ operator computes the magnitude (norm) of the gradient and ϵ is a small constant that prevents division by 0.

We try to reconstruct pixel values, which would result in a gradient field that is the closest to our modifed gradient field $G = [G^{(x)} \quad G^{(y)}]'$. In particular, we can try to minimize the squared differences between gradients in actual image and modified gradients:

$$\arg\min_I \sum_{x,y} \left[\left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y}\right)^2 + \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y}\right)^2 \right], \tag{3}$$

(a) Naive image copy & paste (b) Gradient-domain copy & paste

Figure 2: Comparison of naive and gradient domain image copy & paste.

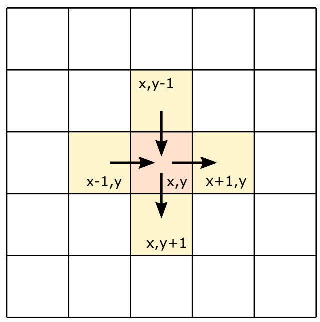


Figure 3: When using forward-differences, a pixel with the coordinates (x, y) is referred to in at most four partial derivates, two along x-axis and two along y-axis.

where the summation is over the entire image. To minimize the function above, we need to equate its partial derivatives to 0. As we optimze for pixel values, we need to compute partial derivates with respect to $I_{x,y}$. Fortunately, most terms in the sum will become 0 after differentiation, as they do not contain the differentiated variable $I_{x,y}$. For a given pixel (x, y), we need to consider only 4 partial derivates: two belonging to the pixel (x, y), x-derivative for the pixel on the left $(x-1, y)$ and y-derivative for the pixel in the top $(x, y-1)$, as shown in Figure 3. This gives us:

$$\frac{\delta F}{\delta I_{x,y}} = -2(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y}) - 2(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y}) + \tag{4}$$

$$2(I_{x,y} - I_{x-1,y} - G^{(x)}_{x-1,y}) + 2(I_{x,y} - I_{x,y-1} - G^{(y)}_{x,y-1})\,. \tag{5}$$

After rearanging the terms and equating $\frac{\delta F}{\delta I_{x,y}}$ to 0, we get:

$$I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y} = G^{(x)}_{x,y} - G^{(x)}_{x-1,y} + G^{(y)}_{x,y} - G^{(y)}_{x,y-1} \,. \quad (6)$$

In these few steps we derived a discrete Poisson equation, which can be found in many engineering problems. The Poisson equation is often written as:

$$\nabla^2 I = \mathrm{div} G \,, \quad (7)$$

where $\nabla^2 I$ is the discrete Laplace operator:

$$\nabla^2 I_{x,y} = I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y} \,, \quad (8)$$

and $\mathrm{div}G$ is the divergence of the vector field:

$$\mathrm{div} G_{x,y} = G^{(x)}_{x,y} - G^{(x)}_{x-1,y} + G^{(y)}_{x,y} - G^{(y)}_{x,y-1} \,. \quad (9)$$

We can also write the equation using discrete convolution operators:

$$I * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = G^{(x)} * \begin{bmatrix} 0 & 1 & -1 \end{bmatrix} + G^{(y)} * \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} . \quad (10)$$

Note that the covolution flips the order of elements in the kernel, thus the row and column vectors on the right hand side are also flipped.

When equation 6 is satisfied for every pixel, it forms a system of linear equations:

$$A \cdot \begin{bmatrix} I_{1,1} \\ I_{2,1} \\ \cdots \\ I_{N,M} \end{bmatrix} = b \quad (11)$$

Here we represent an image as a very large column vector, in which image pixels are stacked column-after-column (in an equivalent manner they can be stacked row-after-row). Every row of matrix A contains the Laplace operator for a corresponding pixel. But the matrix also needs to account for the boundary conditions, that is handle pixels that are at the image edge and therefore do not contain neighbour on one of the sides. Matrix A for a tiny

3x3 image looks like this:

$$A = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix} \tag{12}$$

Obviously, the matrix is enormous for normal size images. However, most matrix elements are 0, so it can be easily stored using a sparse matrix representation. Note that only the pixel in the center of the image (5th row) contains the full Laplace operator; all other pixels are missing neighbours so the operator is adjusted accordingly. Accounting for all boundary cases is probably the most difficult and error-prone part in formulating gradient-field reconstruction problem. The column vector b corresponds to the right hand side of equation 6.

2 Solving linear system

There is a large number of methods and software libraries, which can solve a sparse linear problem given in Equation 11. The Poisson equation is typically solved using multi-grid methods, which iteratively update the solution at different scales. Those, however, are rarther difficult to implement and tailored to one particular shape of a matrix. Alternatively, the solution can be readily found after transformation to the frequency domain (discrete cosine transform). However, such a method does not allow introducing weights, importance of which will be discussed in the next section. Finally, conjugate gradient and biconjugate gradient [1, sec. 2.7] methods provide a fast-converging iterative method for solving sparse systems, which can be very memory efficient. Those methods require providing only a way to compute multiplication of the matrix A and its transpose with an arbitrary vector. Such operation can be realized in an arbitrary way without the need to store the sparse matrix (which can be very large even if it is sparse). The conjugate gradient requires fewer operations than the biconjugate gradient method, but

(a) Uniform weights (b) Higher weights at low contrast

Figure 4: The solution of gradient field reconstruction often contain "pinching" artefacts, such as shown in figure (a). The artefacts can be avoided if small gradient magnitudes are weighted more than large magnitudes.

it should be used only with positive definite matrices. Matrix A is not positive definite so in principle the biconjugate gradient method should be used. However, in practice, conjugate gradient method converges equally well.

3 Weighted reconstruction

An image resulting from solving Equation 11 often contains undesirable "pinching" artefacts, such as those shown in Figure 4a. Those artefacts are inherent to the nature of gradient field reconstruction — the solution is just the best approximation of the desired gradient field but it hardly ever exactly matches the desired gradient field. As we minimize squared differences, tiny inaccuracies for many pixels introduce less error than large inaccuracies for few pixels. This in turn introduces smooth gradients in the areas, where the desired gradient field is inconsistent (cannot form an image). Such gradients produce "pinching" artefacts.

The problem is that the error in reconstructed gradients is penalized the same regardless of whether the value of the gradient is small or large. This is opposite to how the visual system perceives differences in color values: we are more likely to spot tiny difference between two similar pixel values than the same tiny difference between two very different pixel values. We could account for that effect by introducing some form of non-linear metric, however, that would make our problem non-linear and non-linear problems are in general much slower to solve. However, the same can be achieved by introducing weights to our objective function:

$$\arg\min_I \sum_{x,y} \left[w^{(x)}_{x,y} \left(I_{x+1,y} - I_{x,y} - G^{(x)}_{x,y} \right)^2 + w^{(y)}_{x,y} \left(I_{x,y+1} - I_{x,y} - G^{(y)}_{x,y} \right)^2 \right] , \tag{13}$$

where $w^{(x)}_{x,y}$ and $w^{(y)}_{x,y}$ are the weights or importance we assign to each gradient, for horizontal and vertical partial derivatives respectively. Usually the weights are kept the same for both orientations, i.e. $w^{(x)}_{x,y} = w^{(y)}_{x,y}$. To account for the contrast perception of the visual system, we need to assign a higher weight to small gradient magnitudes. For example, we could use the weight:

$$w^{(x)}_{x,y} = w^{(y)}_{x,y} = \frac{1}{||G_{x,y}|| + \epsilon} \tag{14}$$

where $||G_{x,y}||$ is the magnitude of the desired (target) gradient at pixel (x, y) and ϵ is a small constant (0.0001), which prevents division by 0.

4 Matrix notation

We could follow the same procedure as in the previous section and differentiate Equation 13 to find the linear system that minimizes our objective. However, the process starts to be tedious and error-prone. As the objective functions gets more and more complex, it is worth switching to the matrix notation. Let us consider first our original problem without the weights $w_{x,y}$, which we will add later. Equation 3 in the matrix notation can be written as:

$$\arg\min_I \left|\left| \begin{bmatrix} \nabla_x \\ \nabla_y \end{bmatrix} I - \begin{bmatrix} G^{(x)} \\ G^{(y)} \end{bmatrix} \right|\right|^2 . \tag{15}$$

In the equation I, $G^{(x)}$ and $G^{(y)}$ are stacked column vectors, representing columns of the resulting image or desired gradient field. The square brackets denote vertical concatenation of the matrices or vectors. Operator $||\cdot||^2$ is the L_2-norm, which squares and sums the elements of the resulting column vector. ∇_x and ∇_y are differential operators, which are represented as $N \times N$ matrices, where N is the number of pixels. Each row of those sparse matrices tells us which pixels need to be subtracted from one another to compute forward gradients along horizontal and vertical directions. For a tiny 3×3 pixel image those operators are:

$$\nabla_x = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

$$\nabla_y = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

Note that the rows contain all zeros for pixels on the boundary, for which no gradient can be computed: the last column of pixels for ∇_x and the last row of pixels for ∇_y.

Equation 15 is in the format $||Ax - b||^2$, which can be directly solved by some sparse matrix libraries, such as `SciPy.sparse` or the "\" operator in matlab Matlab. However, to reduce the size of the sparse matrix and to speed-up computation, it is worth taking one more step and transform the least-square optimization into a linear problem. For overdetermined systems, such as ours, the solution of the optimization problem:

$$\arg\min_x ||Ax - b||^2 \tag{18}$$

can be found by solving a linear system:

$$A'Ax = A'b\,. \tag{19}$$

Note that $'$ denotes a matrix transpose and $A'A$ is a square matrix. If we replace A and b with the corresponding operators and gradient values from our problem, we get the following linear system:

$$\begin{bmatrix}\nabla'_x & \nabla'_y\end{bmatrix}\begin{bmatrix}\nabla_x\\ \nabla_y\end{bmatrix} I = \begin{bmatrix}\nabla'_x & \nabla'_y\end{bmatrix}\begin{bmatrix}G^{(x)}\\ G^{(y)}\end{bmatrix}, \tag{20}$$

which, after multiplying stacked matrices, gives us:

$$\left(\nabla'_x\,\nabla_x + \nabla'_y\,\nabla_y\right)\,I = \nabla'_x\,G^{(x)} + \nabla'_y\,G^{(y)}\,. \tag{21}$$

Weights can be added to such a system by inserting a sparse diagonal matrix W. For simplicity we use the same weights for vertical and horizontal derivatives:

$$\left(\nabla'_x\,W\,\nabla_x + \nabla'_y\,W\,\nabla_y\right)\,I = \nabla'_x\,W\,G^{(x)} + \nabla'_y\,W\,G^{(y)}\,. \tag{22}$$

The above operations can be performed using a sparse matrix library (or SciPy/Matlab), thus saving us effort of computing operators by hand.

There is still one problem remaining: our equation does not have a unique solution. This is because the target gradient field contains relative information about differences between pixels, but it does not say what the absolute value of the pixels should be. For that reason, we need to constrain the absolute value, for example by ensuring that a value of a first reconstructed pixel is the same as in the source image (I_{src}):

$$\begin{bmatrix}1 & 0 & ... & 0\end{bmatrix}\,I = I_{src}(1,1)\,. \tag{23}$$

If we denote the vector on the left-hand side of the equation as C, the final linear problem can be written as:

$$\left(\nabla'_x\,W\,\nabla_x + \nabla'_y\,W\,\nabla_y + C'\,C\right)\,I = \nabla'_x\,W\,G^{(x)} + \nabla'_y\,W\,G^{(y)} + C'\,I_{src}(1,1)\,. \tag{24}$$

The resulting equation can be solved using a sparse solver in Python or Matlab.

References

[1] S. A. Teukolsky, B. P. Flannery, W. H. Press, and W. T. Vetterling. *Numerical recipes in C*. Cambridge University Press, Cambridge, vol. 2 edition, 1992.

Advanced Graphics and Image Processing — Lecture notes

Rafał Mantiuk

Michaelmas term 2022/23

1 Light field rendering using homographic transformation

This section explains how to render a light field for a novel view position using a parametrization with a focal plane. The method is explained on a rather high level in [1]. These notes are meant to provide a practical guide on how to perform the required calculations and in particular how to find a homographic transformation between the virtual and array cameras.

The scenario and selected symbols are illustrated in Figure 1. We want to render our light field "as seen" by camera K. We have N images captured by N cameras in the array (only 4 shown in the figure), all of which have their apertures on the camera array plane C. We further assume that our array cameras are pin-hole cameras to simplify the explanation. The novel view is rendered assuming focal plane F. The focal plane has a similar function as the focus distance in a regular camera: objects on the focal plane will be rendered sharp, while objects that and in front or behind that plane will appear blurry (in practice they will appear ghosted because of the limited number of cameras). The focal plane F does not need to be parallel to the camera plane; it can be titled, unlike in a traditional camera with a regular lens. Because we have a limited number of cameras, we need to use reconstruction functions A_0, ..., A_1 (only two shown) for each camera. The functions shown contain the weights in the range 0-1 that are used to interpolate between two neighboring views.

To intuitively understand how light field rendering is performed, imagine the following hypothetical scenario. Each camera in the array captures the

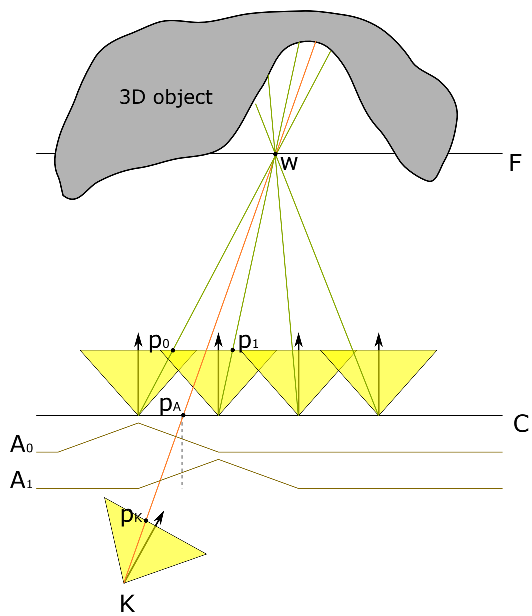


Figure 1: Light field rendering for the novel view represented by camera K. The pixels P_K in the rendered image is the weighted average of the pixels values p_1, ..., p_N from the images captured by the camera array.

image of the scene. Then, all objects in the scene are removed and you put a large projection screen where the focal plane F should be. Then, you swap all cameras for projectors, which project the captured images on the projection screen F. Finally, you put a new camera K at the desired location and capture the image of the projection screen. The projection screen (focal plane) is needed to form an image. Ideally, to obtain a sharp image, we would like to project the camera array images on a geometry. However, such a geometry is not readily available when capturing scenes with a camera array. In this situation a single plane is often a good-enough proxy, which has its analogy in physical cameras (focal distance). More advanced light field rendering methods attempt to reconstruct a more accurate proxy geometry using multi-view stereo algorithms and then project camera images on that geometry [3].

This simplified scenario misses one step, which is modulating each projected image by the reconstruction function A, as such modulation has no physical counterpart. However, this scenario should give you a good idea what operations need to be performed in order to render a light field from a

Data: Camera array images J_1, J_2, ..., J_N
Result: Rendered image I
for *each pixel at the coordinates* $\boldsymbol{p}_K$ *in the novel view* **do**
  $I(\boldsymbol{p}_K) \leftarrow 0$;
  $w(\boldsymbol{p}_K) \leftarrow 0$;
  for *each camera* i *in the array* **do**
    Find the coordinates $\boldsymbol{p}_i$ in the i-th camera image corresponding to the pixel $\boldsymbol{p}_K$;
    Find the coordinates $\boldsymbol{p}_A$ on the aperture plane A corresponding to the pixel $\boldsymbol{p}_K$;
    $I(\boldsymbol{p}_K) \leftarrow I(\boldsymbol{p}_K) + A(\boldsymbol{p}_A)\, J_i(\boldsymbol{p}_i)$;
    $W(\boldsymbol{p}_K) \leftarrow W(\boldsymbol{p}_K) + A(\boldsymbol{p}_A)$;
  end
  $I(\boldsymbol{p}_K) \leftarrow I(\boldsymbol{p}_K)/W(\boldsymbol{p}_K)$;
end

Algorithm 1: Light field rendering algorithm

novel view position.

Now let us see how we can turn such a high-level explanation into a practical algorithm. One way to render a light field is shown in Algorithm 1. The algorithm iterates over all pixels in the rendered image, then for each pixel it iterates over all cameras in the array. The resulting image is the weighted average of the camera images that are warped using homographic transformations. The weights are determined by the reconstruction functions A_i. The algorithm is straightforward, except for the mapping from pixel coordinates in the novel view p_K to coordinates in each camera image p_i and the coordinates on the aperture plane p_A. The following paragraphs explain how to find such transformations.

1.1 Homographic transformation between the virtual and array cameras

The text below assumes that you are familiar with homogeneous coordinates and geometric transformations, both commonly used in computer graphics and computer vision. If these topics are still unclear, refer to Section 2.1 in [4] (this book is available online) or Chapter 6 in [2].

We assume that we know the position and pose of each camera in the array, so that homogeneous 3D coordinates of a point in the 3D word coordinate space w can be mapped to the 2D pixel coordinates p_i of camera i:

$$\boldsymbol{p}_i = \boldsymbol{K}\,\boldsymbol{P}\,\boldsymbol{V}_i\,\boldsymbol{w}\,. \tag{1}$$

where $\boldsymbol{V}$ is the view transformation, $\boldsymbol{P}$ is the projection matrix and $\boldsymbol{K}$ is the intrinsic camera matrix. Note that we will use bold lower case symbols to denote vectors, uppercase bold symbols for matrices and a regular font for scalars. The operation is easier to understand if the coordinates and matrices are expanded:

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2}$$

The view matrix $\boldsymbol{V}$ translates and rotates the 3D coordinates of the 3D point $\boldsymbol{w}$ so that the origin of the new coordinate system is at the camera centre, and camera's optical axis is aligned with the z-axis (as the view matrix in computer graphics). This matrix can be computed using a *LookAt* function, often available in graphics matrix libraries.

The projection matrix $\boldsymbol{P}$ may look like an odd version of an identity matrix, but it actually drops one dimension (projects from 3D to 2D) and copies the value of Z coordinate into the additional homogeneous coordinate w_i. Note that to compute Cartesian coordinates of the point from the homogeneous coordinates, we divide x_i/w_i and y_i/w_i. As w_i is now equal to the depth in the camera coordinates, this operation is equivalent to a perspective projection (you can refer to slides 88–92 in the Introduction to Graphics Course).

The intrinsic camera matrix $\boldsymbol{K}$ maps the projected 3D coordinates into pixel coordinates. f_x and f_y are focal lengths and c_x and c_y are the coordinates of optical center expressed in pixel coordinates. We assume that the intrinsic matrix is the same for all the cameras in the array.

Besides having all matrices for all cameras in the array, we also have a similar transformation for our virtual camera K, which represents the currently rendered view:

$$\boldsymbol{p}_K = \boldsymbol{K}_K\,\boldsymbol{P}\,\boldsymbol{V}_K\,\boldsymbol{w}\,. \tag{3}$$

Our first task is to find transformation matrices that could transform from pixel coordinates $\boldsymbol{p}_K$ in the virtual camera image into pixel coordinates $\boldsymbol{p}_i$

for each camera i. This is normally achieved by inverting the transformation matrix for the novel view and combining it with the camera array transformation. However, the problem is that the product of $\boldsymbol{K}_K \boldsymbol{P} \boldsymbol{V}_K$ is not a square matrix that can be inverted — it is missing one dimension. The dimension is missing because we are projecting from 3D to 2D and one dimension (depth) is lost.

Therefore, to map both coordinates, we need to reintroduce missing information. This is achieved by assuming that the 3D point lies on the focal plane F. Note that the plane equation can be expressed as $\boldsymbol{N} \cdot (\boldsymbol{w} - \boldsymbol{w}_F) = 0$, where $\boldsymbol{N}$ is the plane normal, and $\boldsymbol{w}_F$ specifies the position of the plane in the 3D space. Operator $\cdot$ is the dot product. If the homogeneous coordinates of the point $\boldsymbol{w}$ are $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}$, the plane equation can be expressed as

$$d = \begin{bmatrix} n_x & n_y & n_z & -\boldsymbol{N} \cdot \boldsymbol{w}_F \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{4}$$

where d is the distance to the plane and $\boldsymbol{N} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}$. We can introduce the plane equation into the projection matrix from Equation 2:

$$\begin{bmatrix} x_i \\ y_i \\ d_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 & c_x \\ 0 & f_y & 0 & c_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ n_x^{(c)} & n_y^{(c)} & n_z^{(c)} & -\boldsymbol{N}^{(\boldsymbol{c})} \cdot \boldsymbol{w}_F^{(c)} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{31} & v_{32} & v_{33} & v_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{5}$$

The product of the matrices in above is a full 4×4 transformation matrix, which is not rank-deficient and can be inverted. Note that the pixel coordinates $\boldsymbol{p}_K$ and $\boldsymbol{p}_i$ now have an extra dimension d, which should be set to 0 (because we constrain 3D point w to lie on the focal plane).

It should be noted that the normal and the point in the plane equation have superscript $^{(c)}$, which denotes that the plane is given in the *camera* coordinate system, rather than in the world coordinate system. This is because the point $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}$ is transformed from the world to the camera coordinates by the view matrix $\boldsymbol{V_i}$ before it is multiplied by our modified projection matrix. This could be a desired behavior for the virtual camera, for example if we want the focal plane to follow the camera and be perpendicular to the camera's optical axis. But, if we simply want to specify the focal plane in the world coordinates, we have two options: either replace the third row in the final matrix (obtained after multiplying the three matrices in Equation 5) with our plane equation in the world coordinate system; or to transform the plane to the camera coordinates:

$$\boldsymbol{w}_F^{(c)} = \boldsymbol{V}_i \, \boldsymbol{w}_F \tag{6}$$

and

$$\boldsymbol{N}^{(c)} = \overline{\boldsymbol{V}}_i \, \boldsymbol{N} \, . \tag{7}$$

$\overline{\boldsymbol{V}}_i$ is the "normal" or direction transformation for the view matrix $\boldsymbol{V}_i$, which rotates the normal vector but it does not translate it. It is obtained by setting to zero the translation coefficients w_{14}, w_{24}, and w_{34}.

Now let us find the final mapping from the virtual camera coordinates $\hat{\boldsymbol{p}_K}$ to the array camera coordinates $\hat{\boldsymbol{p}}_i$. We will denote the extended coordinates (with extra d) in Equation 5 as $\hat{\boldsymbol{p}_K}$ and $\hat{\boldsymbol{p}}_i$. We will also denote our new projection and intrinsic matrices in Equation 5 as $\hat{\boldsymbol{P}}$ and $\hat{\boldsymbol{K}}$. Given that, the mapping from $\boldsymbol{p}_K$ to $\boldsymbol{p}_i$ can be expressed as:

$$\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{K}}_i \, \hat{\boldsymbol{P}} \, \boldsymbol{V}_i \, \boldsymbol{V}_K^{-1} \, \hat{\boldsymbol{P}}^{-1} \, \hat{\boldsymbol{K}}_K^{-1} \, \hat{\boldsymbol{p}_K}. \tag{8}$$

The position on the aperture plane $\boldsymbol{w}_A$ can be readily found from:

$$\boldsymbol{w}_A = \boldsymbol{V}_K^{-1} \hat{\boldsymbol{P}}_A^{-1} \hat{\boldsymbol{K}}_K^{-1} \hat{\boldsymbol{p}_K} \, , \tag{9}$$

where the projection matrix $\hat{\boldsymbol{P}}_A$ is modified to include the plane equation of the aperture plane, the same way as done in Equation 5.

1.2 Reconstruction functions

The choice of the reconstruction function A_i will determine how images from different cameras are mixed together. The functions shown in Figure 1 will perform bilinear-interpolation between the views. Although this could be a rational choice, it will result in ghosting for the parts of the scene that are further away from the focal plane F. Another choice is to simulate a wide-aperture camera and include all cameras in the generated view (set $A_i = 1$). This will produce an image with a very shallow depth of field. Another possibility is to use the nearest-neighbor strategy and a box-shaped reconstruction filter (the width of the boxes being equal to the distance between the cameras). This approach will avoid ghosting but will cause the views to jump sharply as the virtual camera moves over the scene. It is worth experimenting with different reconstruction startegies to choose the best for a given application but also for the given angular resolution of the light field (number of views).

References

[1] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *Proc of SIGGRAPH '00*, volume 7, pages 297–306, New York, New York, USA, 2000. ACM Press.

[2] Steve Marschner and Peter Shirley. *Fundamentals of Computer Graphics*. A K Peters/CRC Press, 4 edition edition, 2016.

[3] Ryan S. Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Transactions on Graphics*, 37(6):1–15, dec 2018.

[4] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York Inc, 2010.