

OPTIMIZING

VISION AND VISUALS

Lectures on Cameras, Displays and Perception

Koray Kavaklı, David Robert Walton, Nick Antipa,
Rafał Mantiuk, Douglas Lanman and Kaan Akşit

Association for Computing Machinery

SIGGRAPH 2022

2022

Preface

The evolution of the internet is underway, where immersive virtual 3D environments (commonly known as *metaverse* or *telelife*) will replace flat 2D interfaces. Crucial ingredients in this transformation are next-generation displays and cameras representing genuinely 3D visuals while meeting the human visual system's perceptual requirements.

This course will provide a fast-paced introduction to optimization methods for next-generation interfaces geared towards immersive virtual 3D environments. Firstly, we will introduce lensless cameras for high dimensional compressive sensing (e.g., single exposure capture to a video or one-shot 3D). Our audience will learn to process images from a lensless camera at the end. Secondly, we introduce holographic displays as a potential candidate for next-generation displays. By the end of this course, you will learn to create your 3D images that can be viewed using a standard holographic display. Lastly, we will introduce perceptual guidance that could be an integral part of the optimization routines of displays and cameras. Our audience will gather experience in integrating perception to display and camera optimizations.

This course targets a wide range of audiences, from domain experts to newcomers. To do so, examples from this course will be based on our in-house toolkit to be replicable for future use. The course material will provide example codes and a broad survey with crucial information on cameras, displays and perception.

Prerequisite

The material for this course assumes some basic assumptions about the course participants. The participants in the audience should have some familiarity with concepts such as *Metaverse*, *Telelife* [4], virtual reality and augmented reality. Participants can be from various technical backgrounds but are willing to advance their understanding of the optimization side of displays, cameras or perception. The course material will describe the base hardware used in the demonstrated optimizations. However, the lecturers will not cover how to build this hardware from the ground up. Instead, they will provide a quick overview and cite relevant resources from the most recent literature. The attendees are also expected to be knowledgeable or willing to learn Python programming language, modern machine learning libraries, signal theory, and optics. But most importantly, above-average interest in optimizing future's devices is a must.

The code material of this course is a result of various research works from the [Computational Light Laboratory](#) that was conducted in the passing one-year timeframe (2021-2022). These research works include new methods for foveated rendering and image statistics [5], learned techniques for holographic light transport [1], learned optimizations for multiplane holography [2], perceptually guided hologram generation routines in displays [6] and learned optimizations for lensless cameras [3].

Individuals willing to replicate the outcome of our materials from this course on their local machines have to install several pieces of software in their operating systems. Such individuals must be familiar with the Python programming language, and they should install [Python](#) and [Jupyter Notebooks](#) in their operating systems. In addition, these individuals will need to install [Torch](#) with its Python bindings, [Matplotlib](#) and [plotly](#) libraries for plotting purposes while using the provided Jupyter Notebooks. When we compiled this material, our production machines used the Python distribution 3.9.7, Torch distribution 1.9.0, Matplotlib distribution 3.3.4 and Jupyter Notebook distribution 6.2.0. As a final piece, please make sure to install our library using:

```
[1]: pip install odak
```

At the time of this course, Odak is at version 0.2.0. The participants willing to go beyond this course and learn more about relevant research are welcome to reach out to the lecturers via email. In addition, Computational Light Laboratory offers [seminars](#) from experts on relevant topics. Finally, Computational Light Laboratory also invites all attendees to a research hub formulated as a [Slack Group](#). This way, curious readers and attendees of our course can keep up-to-date and meet more folks in the relevant fields.

Lecturers

Koray Kavali

PhD Candidate, University College London, United Kingdom

E-mail: kkavakli@ku.edu.tr

Koray Kavali is a PhD student in Computational Light Laboratory at University College London. He has completed his M.Sc. degree in Optical Microsystems Laboratory at Koç University. He received his B.S. degrees from Koç University Electrical and Electronics Engineering and B.A. in Business Administration. During his M.Sc. studies, he worked as a research engineer in CY Vision, Turkey. He worked as a research engineer in CY Vision, Istanbul. His research includes developing computer-generated holographic displays with novel graphics pipelines, light transport modelling, and biomedical applications of the holographic displays.

David Robert Walton

Postdoctoral Researcher, University College London, United Kingdom

E-mail: david.walton.13@ucl.ac.uk

David Robert Walton is a postdoctoral researcher working in the Virtual Environments and Computer Graphics group at University College London (UCL) in the United Kingdom. He previously completed an EngD working with UCL and Imagination Technologies on computer vision and rendering techniques for improved lighting in augmented reality. His current research focuses on computer graphics, novel displays and human perception, particularly how properties of visual perception can aid in developing efficient graphics algorithms and novel display hardware.

Nick Antipa

Assistant Professor, University of California, San Diego, United States of America

E-mail: nantipa@eng.ucsd.edu

Nick Antipa is an assistant professor in Electrical and Computer engineering where his lab focuses on computational imaging. He received his PhD in Computational Imaging at UC Berkeley in the Electrical Engineering and Computer Sciences department with Laura Waller and Ren Ng. Prior to his time at Berkeley, Nick worked as an optical engineer at Lawrence Livermore National Lab, designing 3D metrology equipment in support of the National Ignition Facility. Nick received his

MS in Optics from the University of Rochester in 2009, and BS in Optical Science and Engineering at UC Davis in 2008.

Rafał Mantiuk

Professor, University of Cambridge, United Kingdom

E-mail: rafal.mantiuk@cl.cam.ac.uk

Rafał K. Mantiuk is a Professor of Graphics and Displays at the Department of Computer Science and Technology, the University of Cambridge in the United Kingdom. He received PhD from the Max-Planck Institute for Computer Science in Germany. His recent interests focus on computational displays, rendering and imaging algorithms that adapt to human visual performance and deliver the best image quality given limited resources, such as computation time or bandwidth. He contributed to early work on high dynamic range imaging, including quality metrics (HDR-VDP), video compression and tone-mapping.

Douglas Lanman

Director of Display Systems Research, Facebook Reality Labs, United States of America

E-mail: douglas.lanman@fb.com

Douglas Lanman is the Director of Display Systems Research at Reality Labs, where he leads investigations into advanced AR/VR display and imaging technologies. He received a B.S. in Applied Physics with Honors from Caltech in 2002 and M.S. and Ph.D. degrees in Electrical Engineering from Brown University in 2006 and 2010, respectively. He was a Senior Research Scientist at NVIDIA Research from 2012 to 2014, a Postdoctoral Associate at the MIT Media Lab from 2010 to 2012, and an Assistant Research Staff Member at MIT Lincoln Laboratory from 2002 to 2005.

Kaan Akşit

Associate Professor, University College London, United Kingdom

E-mail: k.aksit@ucl.ac.uk

Kaan Akşit is an Associate Professor in the computer science department at University College London, where he leads the Computational Light Laboratory. Kaan received his PhD degree in electrical engineering at Koç University, Turkey, in 2014. He received an M.Sc. degree in electrical power engineering from RWTH Aachen University, Germany, in 2010. He obtained a B.S. degree in electrical engineering from Istanbul Technical University, Turkey, in 2007. He worked as a research intern in Philips Research, the Netherlands, and Disney Research, Switzerland, in 2009 and 2013, respectively. In addition, he was a scientist at NVIDIA, the USA, between 2014 and 2020.

Course Overview

5 minutes: Welcome and Introductions

Kaan Akşit Welcoming the audience, Describing the aims and objectives of the course, Highlighting the target audience and Introducing lecturers.

20 minutes: Keynote on future interfaces

Douglas Lanman An analysis of issues in the current day vision and visuals, and a prime outlook on future's devices.

30 minutes: Next-generation Compressive Sensing: Lensless Cameras

Nick Antipa A survey on state-of-art research on lensless cameras with research highlights and their promises for future's interfaces. Basic concepts and descriptions will be provided related to lensless cameras. In the next step, an interactive coding session will be provided to demonstrate how to process images from a lensless camera using the Alternating Direction Method of Multipliers (ADMM), vanilla Gradient-Descent (GD) and Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) methods.

45 minutes: Next-generation Computational Displays: Holographic Displays

Kaan Akşit, Koray Kavaklı and David Robert Walton A survey on state-of-art research on holographic displays with research highlights and their promises for future's interfaces. Basic concepts and descriptions will be provided related to holographic displays. The survey is followed by an interactive coding session to demonstrate how to create computer-generated holograms for a standard holographic display using Gerchberg-Saxton (GS), Vanilla Gradient-Descent (GD) and Stochastic Gradient Descent (SGD). In addition, an interactive coding session to demonstrate how to derive a perceptually guided gaze-contingent loss function and how to use such functions on a standard holographic display will also be provided.

30 minutes: Practical models of Perception in Graphics

Rafał Mantiuk A brief tutorial on modelling the visual quality using psychophysical models of contrast sensitivity, luminance masking and contrast masking. The tutorial will be illustrated with examples and ready-to-use code.

Bibliography

- [1] K. Kavaklı, H. Urey, and K. Akşit. Learned holographic light transport. *Applied Optics*, 61(5): B50–B55, 2022.
- [2] K. Kavaklı, Y. Itoh, H. Urey, and K. Akşit. Realistic defocus blur for multiplane computer-generated holography, 2022. URL <https://arxiv.org/abs/2205.07030>.
- [3] O. Kingshott, N. Antipa, E. Bostan, and K. Akşit. Unrolled primal-dual networks for lensless cameras. *arXiv preprint arXiv:2203.04353*, 2022.
- [4] J. Orlosky, M. Sra, K. Bektaş, H. Peng, J. Kim, N. Kos'myna, T. Höllerer, A. Steed, K. Kiyokawa, and K. Akşit. Telelife: The future of remote living. *Frontiers in Virtual Reality*, 2, 2021. ISSN 2673-4192. doi: 10.3389/frvir.2021.763340. URL <https://www.frontiersin.org/article/10.3389/frvir.2021.763340>.
- [5] D. R. Walton, R. K. Dos Anjos, S. Friston, D. Swapp, K. Akşit, A. Steed, and T. Ritschel. Beyond blur: Real-time ventral metamers for foveated rendering. *ACM Transactions on Graphics*, 40(4): 1–14, 2021.
- [6] D. R. Walton, K. Kavaklı, R. K. d. Anjos, D. Swapp, T. Weyrich, H. Urey, A. Steed, T. Ritschel, and K. Akşit. Metameric varifocal holography. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022.



SIGGRAPH 2022
VANCOUVER+ 8-11 AUG

OPTIMIZING
VISION AND VISUALS

Lectures on Cameras, Displays and Perception

**Part II: Next-generation Computational Displays -
Computer-Generated Holography**

Kaan Akşit, David Robert Walton and Koray Kavaklı

University College London



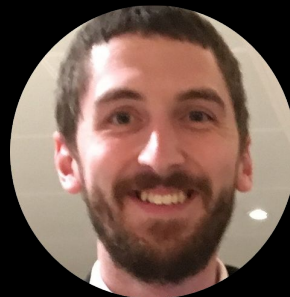
UCL

Lecturers

Computational
Light
Laboratory



Kaan Akşit
Associate Professor
University College London
<https://kaanaksit.com>




David Robert Walton
Post-Doctoral Researcher
University College London
<https://drwalton.github.io/>



Koray Kavaklı
Incoming PhD Student
University College London

For more on our activities: <https://complightlab.com>

For updates follow [@kaanaksit](#) and [@complightlab](#) on  Twitter

For our codebase follow [kunguz](#) and [complight](#) on  GitHub

PEOPLE.COM > HUMAN INTEREST

Average U.S. Adult Will Spend Equivalent of 44 Years of Their Life Staring at Screens: Poll

People believe less than half the time they spend on these devices is "productive," research shows



Jason Orlosky, Misha Sra, Kenan Bektaş, Huaishu Peng, Jeeun Kim, Nataliya Kos'myna, Tobias Hollerer, Anthony Steed, Kiyoshi Kiyokawa and **Kaan Akşit** (2021). *Telexlife: The Future of Remote Living*. *Frontiers in Virtual Reality*.



Image courtesy Interesting Engineering

Computational Displays at SIGGRAPH

Bimber, Oliver, Bernd Fröhlich, Dieter Schmalstieg, and L. Miguel Encarnacao. "The virtual showcase." In *ACM SIGGRAPH 2006 Courses*, pp. 9-es. 2006.

Mantiuk, Rafał, Scott Daly, and Louis Kerofsky. "Display adaptive tone mapping." In *ACM SIGGRAPH 2008 papers*, pp. 1-10. 2008.

Wetzstein, Gordon, Douglas R. Lanman, Matthew Waggner Hirsch, and Ramesh Raskar. "Tensor displays: compressive light field synthesis using multilayer displays with directional backlighting." (2012).

Lanman, Douglas, and David Luebke. "Near-eye light field displays." *ACM Transactions on Graphics (TOG)* 32, no. 6 (2013): 1-10.

Huang, Fu-Chung, David P. Luebke, and Gordon Wetzstein. "The light field stereoscope." In *SIGGRAPH Emerging Technologies*, pp. 24-1. 2015.

and many more...

Next in Computational Displays: Holographic Displays

Lee, Seungjae, Changwon Jang, Seokil Moon, Jaebum Cho, and ByoungHo Lee. "Additive light field displays: realization of augmented reality with holographic optical elements." *ACM Transactions on Graphics (TOG)* 35, no. 4 (2016): 1-13.

Akşit, Kaan, Ward Lopes, Jonghyun Kim, Peter Shirley, and David Luebke. "Near-eye varifocal augmented reality display using see-through screens." *ACM Transactions on Graphics (TOG)* 36, no. 6 (2017): 1-13.

Maimone, Andrew, Andreas Georgiou, and Joel S. Kollin. "Holographic near-eye displays for virtual and augmented reality." *ACM Transactions on Graphics (Tog)* 36, no. 4 (2017): 1-16.

Kim, Jonghyun, Youngmo Jeong, Michael Stengel, **Kaan Akşit**, Rachel Albert, Ben Boudaoud, Trey Greer et al. "Foveated AR: dynamically-foveated augmented reality display." *ACM Transactions on Graphics (TOG)* 38, no. 4 (2019): 1-15.

Shi, Liang, Fu-Chung Huang, Ward Lopes, Wojciech Matusik, and David Luebke. "Near-eye light field holographic rendering with spherical waves for wide field of view interactive 3D computer graphics." *ACM Transactions on Graphics (TOG)* 36, no. 6 (2017): 1-17.

Chakravarthula, Praneeth, Ethan Tseng, Tarun Srivastava, Henry Fuchs, and Felix Heide. "Learned hardware-in-the-loop phase retrieval for holographic near-eye displays." *ACM Transactions on Graphics (TOG)* 39, no. 6 (2020): 1-18.

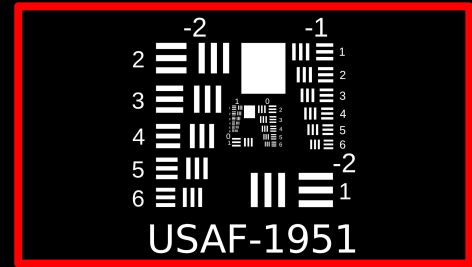
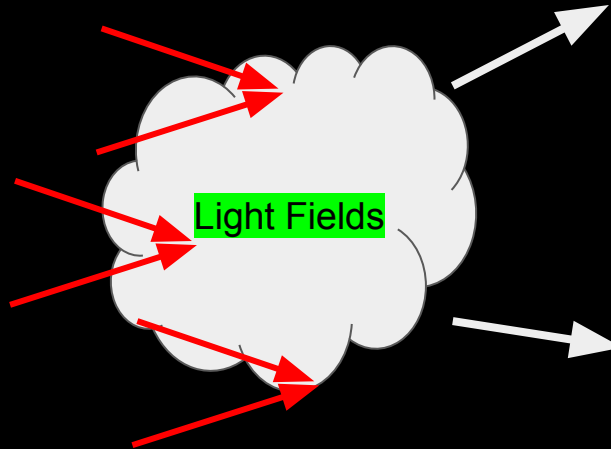
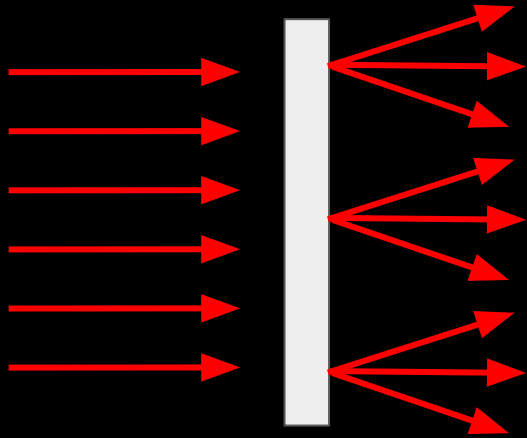


<https://github.com/bchao1/awesome-holography>

Why Holographic Displays? And why now?

A holographic display aims to produce a targeted light field using diffraction and interference phenomena.

Holographic display
showing a hologram



2D Image



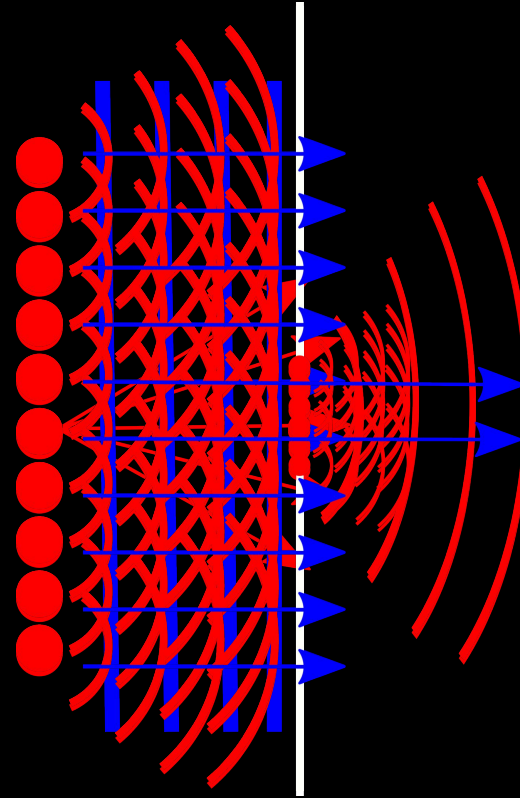
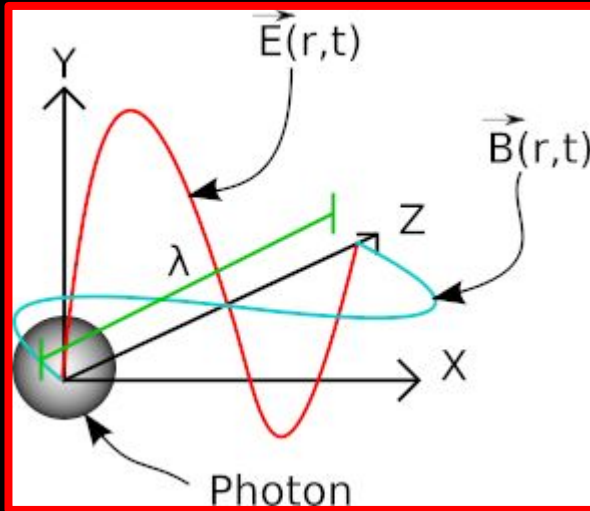
3D Image

Holographic displays can generate 2D or 3D images at any optical depth!

A holographic display aims to produce a targeted light field using **diffraction** and interference phenomena.

Diffraction

Light tends to bend around corners.

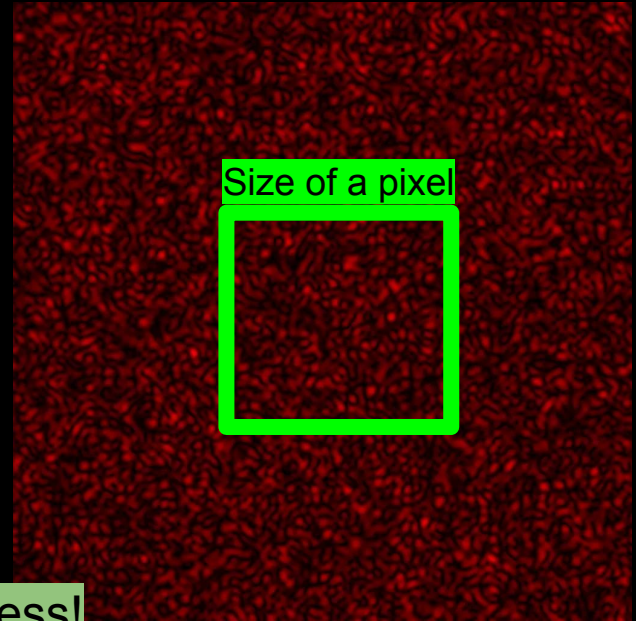
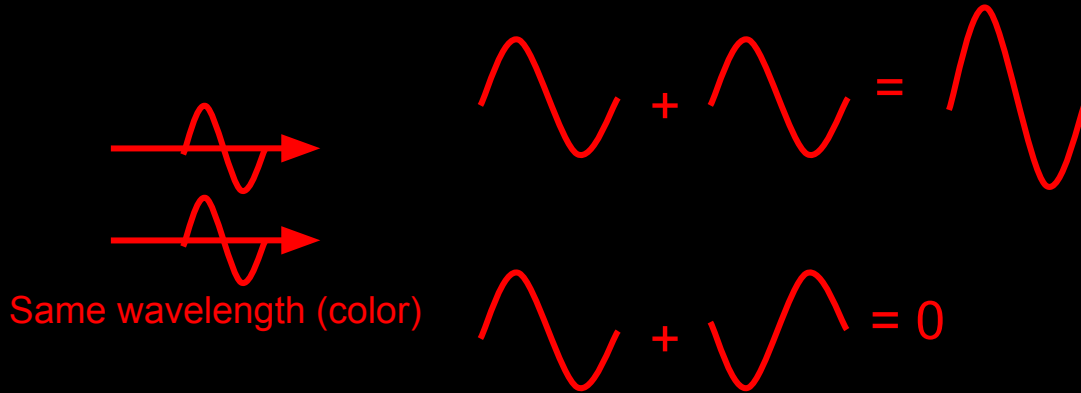


Inaccurate modeling in other techniques degrades effective resolution!

A holographic display aims to produce a targeted light field using diffraction and **interference** phenomena.

Interference

Light beams can amplify or cancel each other.



Boost in resolution and brightness!

A **holographic display** aims to produce a targeted light field using diffraction and interference phenomena.

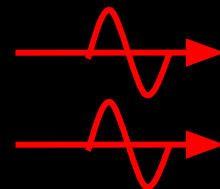
Hologram

Diffractive optics

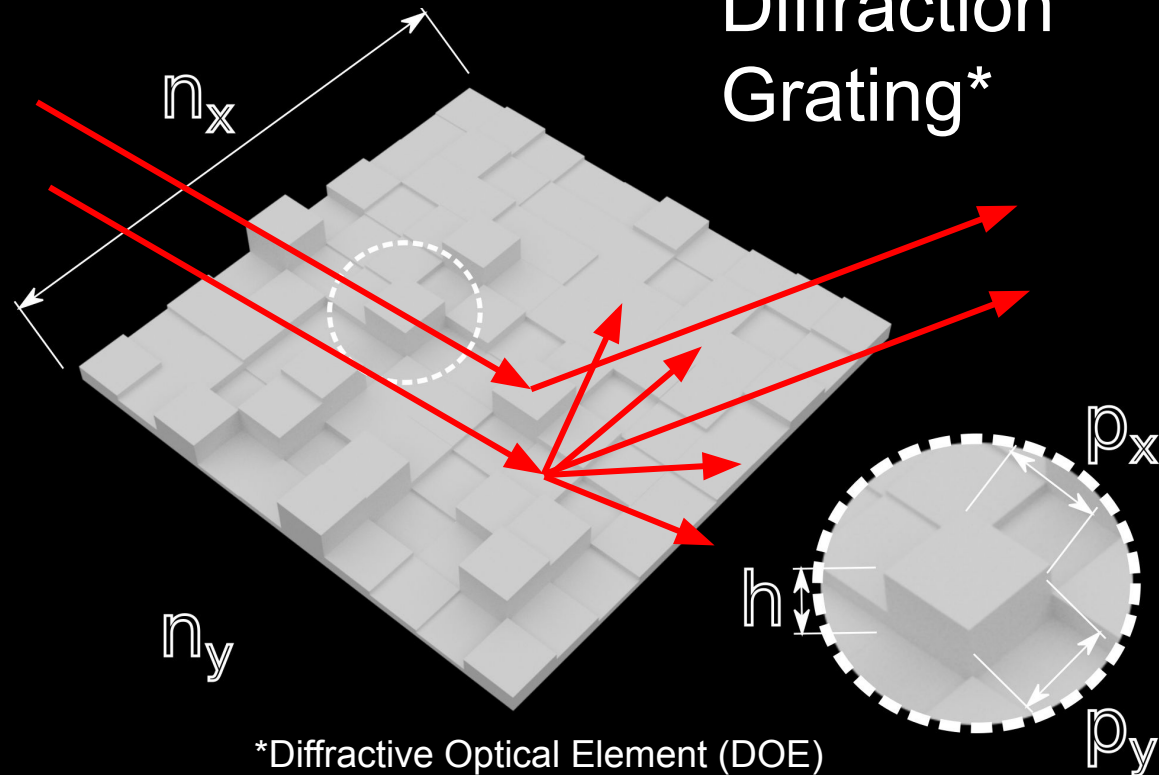
Diffraction Grating*



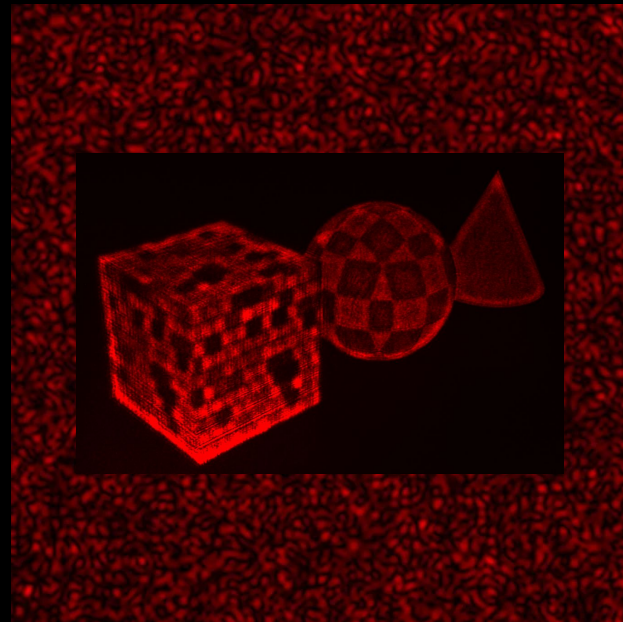
Collimated beam



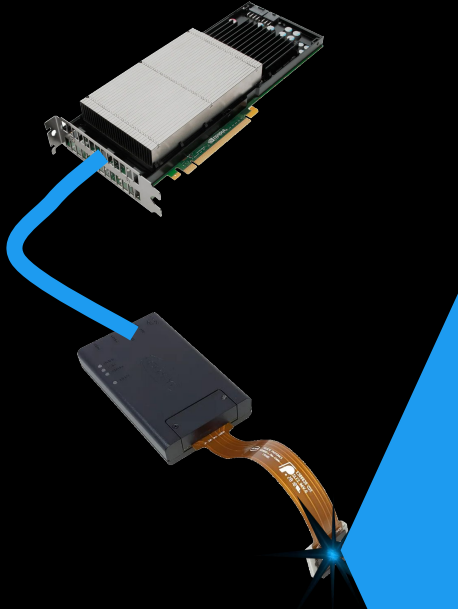
Same wavelength (colour)



*Diffractive Optical Element (DOE)



A holographic display aims to produce a targeted light field using diffraction and interference phenomena.



Texas Instruments
Thorlabs
Holoeye
JasperDisplay

* Consult with the lecturers to learn more about this specific algorithm to generate shown result.

A holographic display aims to produce a targeted light field using diffraction and interference phenomena.

Computer-generated holography promises unmatched resolution characteristics.

Holography can provide true dark levels and true dynamic range.

A large color gamut can be achieved with lasers.

Instant 2D visuals at any depth or true 3D visuals within the same display.

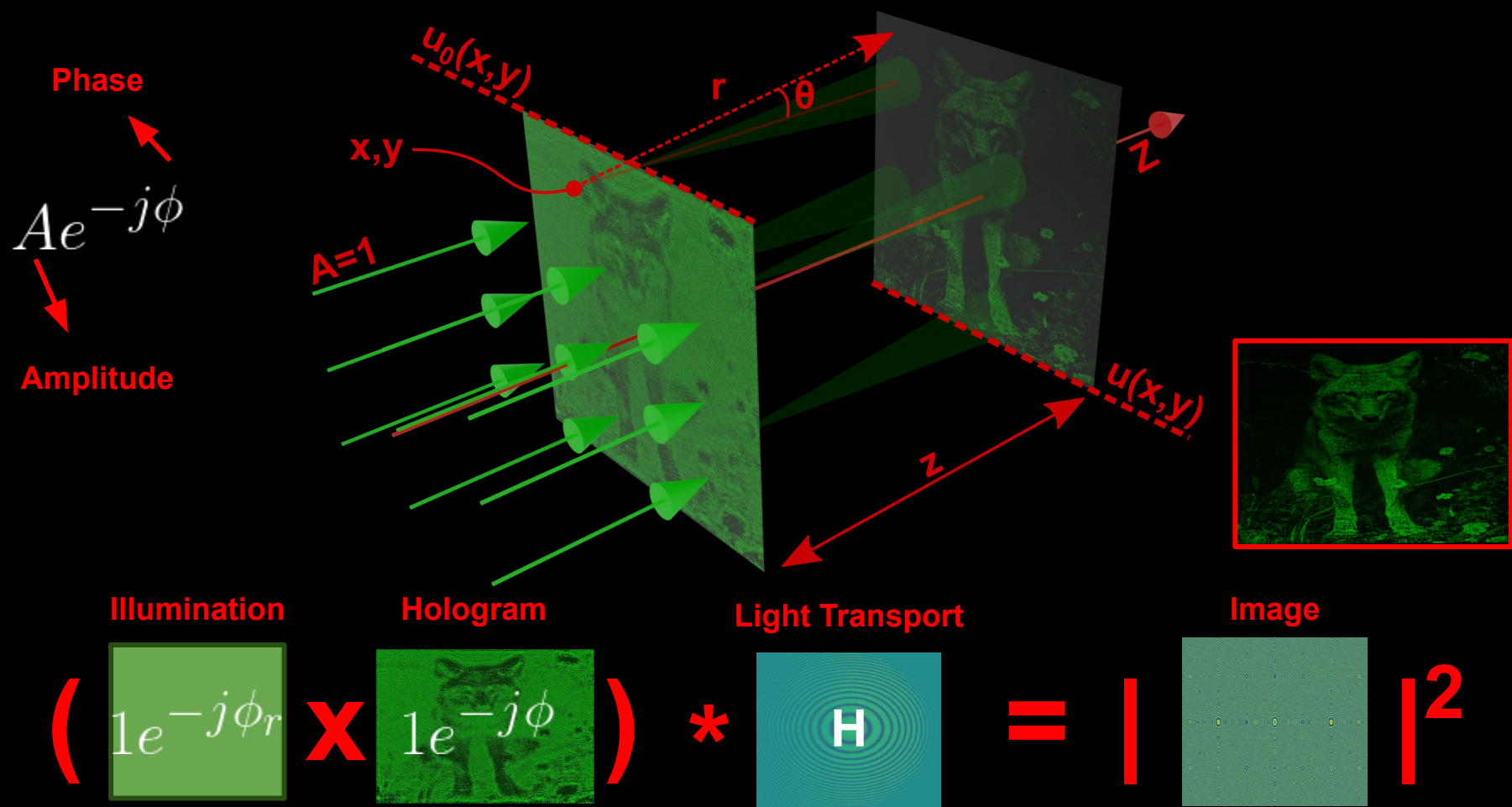
A dead pixel is an ancient relic.


COMPUTER-GENERATED HOLOGRAPHY: ACTIVE RESEARCH PROBLEM

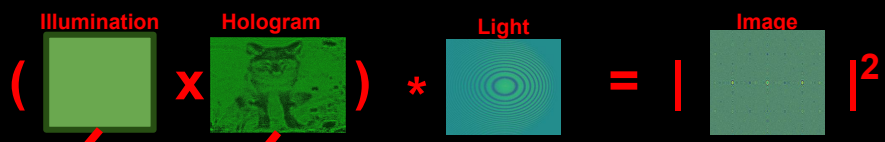
Full control with Holography: Polarization, Amplitude,
Phase, Interference, Diffraction



Light-efficient projection, Wide dynamic
range, High resolution, Solid-state steering



 <https://github.com/kunguz/odak>



Preparations

```
In [1]: import odak
In [2]: import torch
In [3]: device = torch.device('cuda')
In [4]: resolution = [1080, 1920]
In [5]: wavelength = 515e-9
In [6]: pixel_pitch = 8e-6
```

Illumination

```
In [7]: illumination_amplitude = torch.ones(resolution[0], resolution[1]).to(device)
In [8]: illumination_phase = torch.rand(resolution[0], resolution[1]).to(device)
In [9]: illumination = odak.learn.wave.generate_complex_field(illumination_amplitude,
...: illumination_phase)
```

Hologram

```
In [10]: hologram_phase = torch.rand(resolution[0], resolution[1]).detach().to(device)
...: .requires_grad_()
In [11]: hologram_amplitude = torch.ones_like(hologram_phase).to(device)
In [12]: hologram = odak.learn.wave.generate_complex_field(hologram_amplitude, hologram
...: _phase)
```

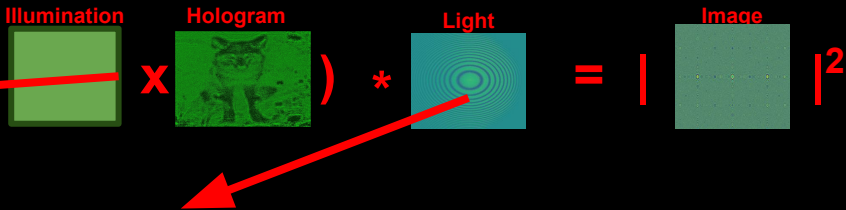
Multiply

```
In [13]: modulated_beam = illumination * hologram
```

```
In [14]: propagation_type = 'Bandlimited Angular Spectrum'
```

```
In [15]: def propagate(input_field, distance, pixel_pitch, wavelength, propagation_type):
...:     k = odak.learn.wave.wavenumber(wavelength)
...:     input_field_padded = odak.learn.tools.zero_pad(input_field)
...:     propagated_field_padded = odak.learn.wave.propagate_beam(input_field_padded, k, d
istance, pixel_pitch, wavelength, propagation_type=propagation_type)
...:     propagated_field = odak.learn.tools.crop_center(propagated_field_padded)
...:     return propagated_field
```

```
In [16]: image_plane = propagate(modulated_beam, 0.15, pixel_pitch, wavelength, propagation_ty
pe)
```



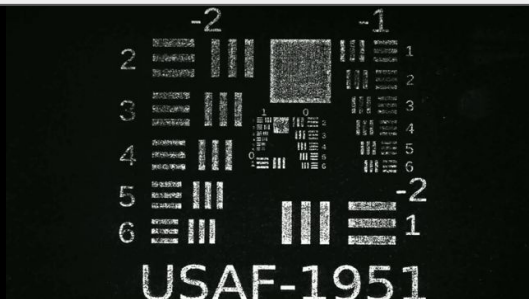
Light transport

Bandlimited Angular Spectrum, Transfer Function
Fresnel, Impulse Response Fresnel and Fraunhofer

See `Odak.learn.wave` and `Odak.wave` for more!



https://github.com/complight/realistic_holography



$$u(x, y) = \frac{1}{j\lambda} \iint u_0(x, y) \frac{e^{jkr}}{r} \cos(\theta) dx dy$$

Kavaklı, Koray, Hakan Urey, and Kaan Akşit. "Learned holographic light transport." *Applied Optics* (2021). (INVITED)

$$\left(\begin{array}{c} \text{Illumination} \\ \text{Hologram} \end{array} \right) \times \left(\begin{array}{c} \text{Hologram} \\ \text{Image} \end{array} \right) * \begin{array}{c} \text{Light} \\ \text{Image} \end{array} = \left| \begin{array}{c} \text{Image} \\ \text{Image} \end{array} \right|^2$$

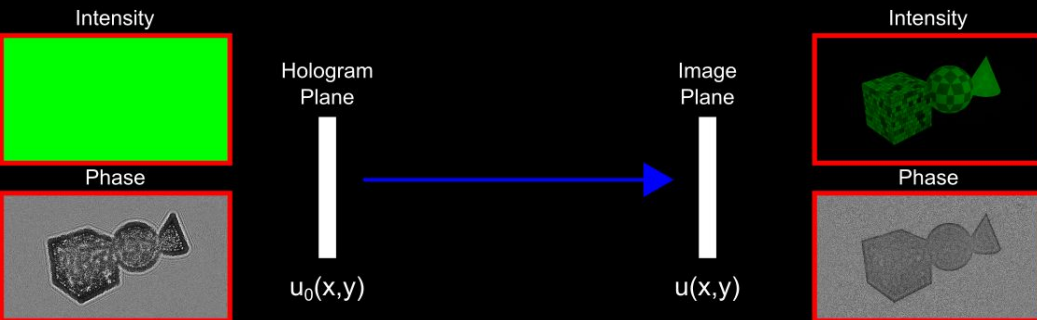
Image

```
In [17]: image = odak.learn.wave.calculate_amplitude(image_plane)**2
```



How do we optimize holograms with this model?

Gerchberg-Saxton Algorithm



odak.learn.wave.gerchberg_saxton

R. W. Gerchberg and W. O. Saxton. A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures. Optik, 1972.

```
import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber, propagate_beam,
set_amplitude, generate_complex_field
from odak.learn.tools import zero_pad, crop_center, save_image, load_image
from odak import np
from tqdm import tqdm

wavelength = 515e-9
k = wavenumber(wavelength)
dx = 0.000008
resolution = [1080,1920]
distance = 0.15
slm_range = 2*np.pi
dynamic_range = 255

phase = torch.rand(resolution[0], resolution[1])
target = load_image('target_image.png')[ :, :,1]/255.

amplitude = torch.ones_like(target)

hologram = generate_complex_field(amplitude,phase)

propagation_type = 'TR Fresnel'

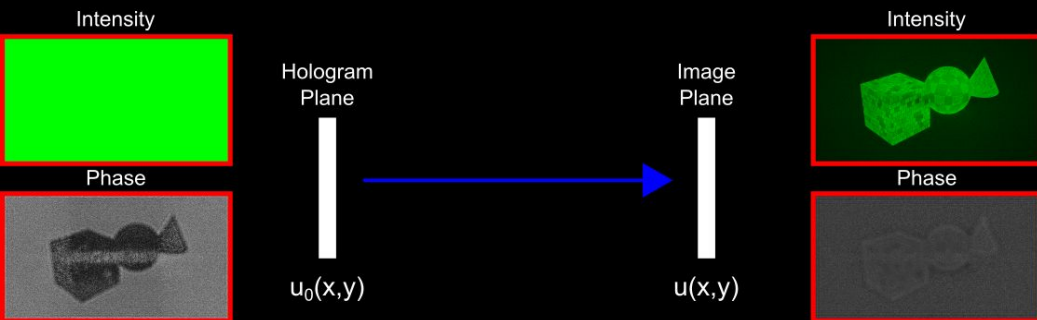
iteration_number = 100
t = tqdm(range(iteration_number), leave=False)
for i in t:
    hologram = zero_pad(hologram)
    recon_field = propagate_beam(hologram, k, distance, dx, wavelength, propagation_type)
    recon_field = crop_center(recon_field)
    recon_field = set_amplitude(recon_field,target)
    recon_field = zero_pad(recon_field)
    hologram = propagate_beam(recon_field, k, -distance, dx, wavelength,
propagation_type)
    hologram = crop_center(hologram)
    hologram = set_amplitude(hologram, amplitude)

hologram_padded = zero_pad(hologram)
reconstruction = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
reconstruction = crop_center(reconstruction)

reconstruction_amp = calculate_amplitude(reconstruction)
reconstruction_intensity = (reconstruction_amp/reconstruction_amp.max())**2
save_image('reconstructed_image.png',reconstruction_intensity,cmin=0.,cmax=1.)

phase_hologram = calculate_phase(hologram)
phase_only_hologram = (phase_hologram%slm_range)/slm_range*dynamic_range
save_image('phase_only_hologram.png',phase_only_hologram)
```

Gradient Descent Algorithm



odak.learn.wave.gradient_descent

```
import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber, propagate_beam,
set_amplitude, generate_complex_field
from odak.learn.tools import zero_pad, crop_center, save_image, load_image
from odak import np
from tqdm import tqdm
wavelength = 515e-9
k = wavenumber(wavelength)
dx = 0.000008
resolution = [1080,1920]
distance = 0.15
slm_range = 2*np.pi
dynamic_range = 255

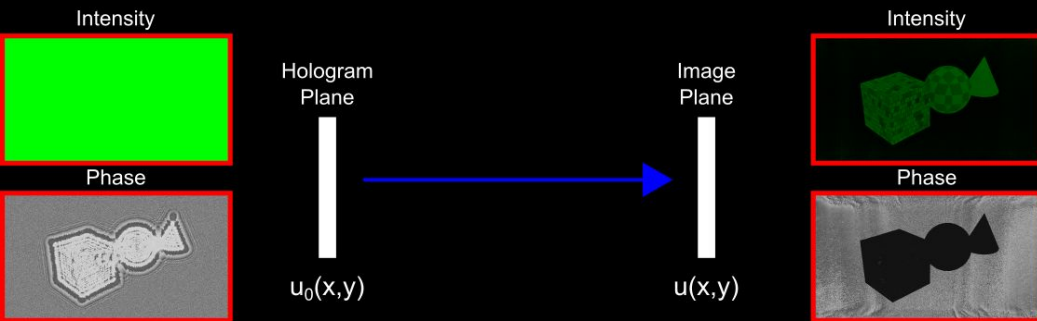
phase = torch.rand(resolution[0], resolution[1])
target = load_image('target_image.png')[ :, :, 1]/255.
amplitude = torch.ones_like(target)
hologram = generate_complex_field(amplitude, phase)
propagation_type = 'TR Fresnel'
loss_function = torch.nn.MSELoss(reduction='none')
alpha = 0.1
iteration_number = 100
t = tqdm(range(iteration_number), leave=False)
hologram_padded = zero_pad(hologram)
recon_field = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
recon_field_cropped = crop_center(recon_field)
recon_intensity = calculate_amplitude(recon_field_cropped)**2
loss = loss_function(recon_intensity, target)
loss_field = generate_complex_field(loss, calculate_phase(recon_field_cropped))
loss_field_padded = zero_pad(loss_field)
loss_propagated_padded = propagate_beam(loss_field_padded, k, -distance, dx, wavelength,
propagation_type)
loss_propagated = crop_center(loss_propagated_padded)
hologram_updated = hologram - alpha * loss_propagated
hologram_phase = calculate_phase(hologram_updated)
hologram = generate_complex_field(amplitude, hologram_phase)
t.set_description('Loss: {:.4f}'.format(torch.mean(loss)))

hologram_padded = zero_pad(hologram)
reconstruction = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
reconstruction = crop_center(reconstruction)

reconstruction_amp = calculate_amplitude(reconstruction)
reconstruction_intensity = (reconstruction_amp/reconstruction_amp.max())**2
save_image('reconstructed_image.png', reconstruction_intensity, cmin=0., cmax=1.)

phase_hologram = calculate_phase(hologram)
phase_only_hologram = (phase_hologram%slm_range)/slm_range*dynamic_range
save_image('phase_only_hologram.png', phase_only_hologram)
```


Double Phase Method



odak.learn.wave.shift_w_double_phase

C. K. Hsueh and A. A. Sawchuk. Computer-generated double-phase holograms. Applied Optics, 1978.

```
import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber, propagate_beam,
set_amplitude, generate_complex_field
from odak.learn.tools import zero_pad, crop_center, save_image, load_image
from odak import np
from tqdm import tqdm
wavelength = 515e-9
k = wavenumber(wavelength)
dx = 0.000008
resolution = [1080,1920]
distance = 0.30
slm_range = 2*np.pi
dynamic_range = 255
illumination_amplitude = torch.ones(resolution[0], resolution[1])
target = load_image('target_image.png')[ :, :, 1]/255.
reconstruction_phase = torch.rand(resolution[0], resolution[1])
reconstruction_amplitude = target**0.5
reconstruction = generate_complex_field(reconstruction_amplitude, reconstruction_phase)
propagation_type = 'TR Fresnel'
reconstruction_padded = zero_pad(reconstruction)
hologram_padded = propagate_beam(reconstruction_padded, k, -distance, dx, wavelength,
propagation_type)
hologram = crop_center(hologram_padded)

amplitudes = calculate_amplitude(hologram)
amplitudes = amplitudes / amplitudes.max()
phases = calculate_phase(hologram)
phase_zero_mean = phases - torch.mean(phases)
phase_offset = torch.arccos(amplitudes)
phase_low = phase_zero_mean - phase_offset
phase_high = phase_zero_mean + phase_offset
phase_only = torch.zeros_like(reconstruction_phase)
phase_only[0::2, 0::2] = phase_low[0::2, 0::2]
phase_only[0::2, 1::2] = phase_high[0::2, 1::2]
phase_only[1::2, 0::2] = phase_high[1::2, 0::2]
phase_only[1::2, 1::2] = phase_low[1::2, 1::2]

hologram = generate_complex_field(illumination_amplitude, phase_only)
hologram_padded = zero_pad(hologram)
reconstruction = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
reconstruction = crop_center(reconstruction)
reconstruction_amp = calculate_amplitude(reconstruction)
reconstruction_intensity = (reconstruction_amp/reconstruction_amp.max())**2
save_image('reconstructed_image.png', reconstruction_intensity, cmin=0., cmax=1.)

phase_hologram = calculate_phase(hologram)
phase_only_hologram = (phase_hologram%slm_range)/slm_range*dynamic_range
save_image('phase_only_hologram.png', phase_only_hologram)
```

Target Image with Foveated Blur



Target Image with Foveated Blur and Noise



Target Image Ventral Metamer



Optimising Metameric Loss

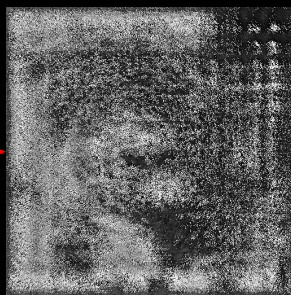


GD: Metameric Loss



Target Image

Optimization



Phase Map

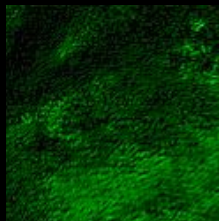
Reconstructed Intensities



Fovea



Periphery



```
import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber, propagate_beam,
set_amplitude, generate_complex_field
from odak.learn.tools import zero_pad, crop_center, save_image, load_image
from odak import np
from tqdm import tqdm
from odak.learn.perception import MetamericLoss
device = torch.device('cuda')
wavelength = 515e-9
k = wavenumber(wavelength)
dx = 0.000008
resolution = [512,512]
distance = 0.15
slm_range = 2*np.pi
dynamic_range = 255
propagation_type = 'TR Fresnel'

target = load_image('target_image.png')[ :, : ].to(device)/255.
amplitude = torch.ones_like(target)
phase = torch.rand_like(target, requires_grad=True)
optim = torch.optim.Adam(params=[phase], lr=0.1)
loss_function = MetamericLoss(device=device, alpha=0.08, \
    real_image_width=0.2, real_viewing_distance=0.7)

iteration_number = 100
t = tqdm(range(iteration_number), leave=False)
for i in t:
    optim.zero_grad()
    hologram = generate_complex_field(amplitude, phase)
    hologram_padded = zero_pad(hologram)
    recon_field = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
    recon_field_cropped = crop_center(recon_field)
    recon_intensity = calculate_amplitude(recon_field_cropped)**2
    loss = loss_function(recon_intensity[None,None,...], target[None,None,...], gaze=[0.5, 0.5])
    loss.backward()
    optim.step()
    t.set_description('Loss: {:.4f}'.format(torch.mean(loss)))

hologram_padded = zero_pad(hologram)
reconstruction = propagate_beam(hologram_padded, k, distance, dx, wavelength, propagation_type)
reconstruction = crop_center(reconstruction)
reconstruction_amp = calculate_amplitude(reconstruction)
reconstruction_intensity = (reconstruction_amp/reconstruction_amp.max())**2
save_image('reconstructed_image.png', reconstruction_intensity, cmin=0., cmax=1.)
phase_hologram = calculate_phase(hologram)
phase_only_hologram = (phase_hologram%slm_range)/slm_range*dynamic_range
save_image('phase_only_hologram.png', phase_only_hologram)
```

Extra materials



L^AT_EX



<https://github.com/complight/cameras-displays-perception-course>

Join the conversation



https://complightlab.com/research_hub/



Kaan Akşit

Associate Professor

University College London

kaanaksit@kaanaksit.com

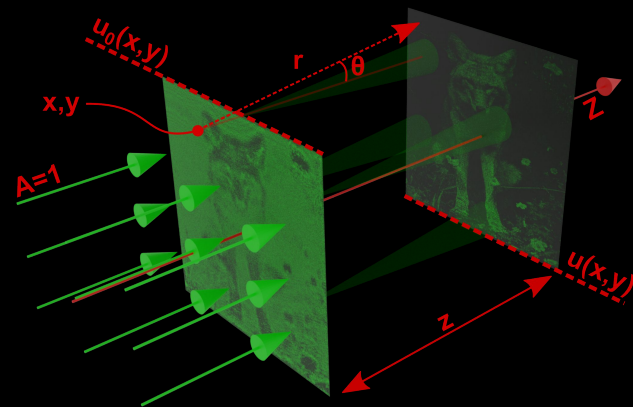
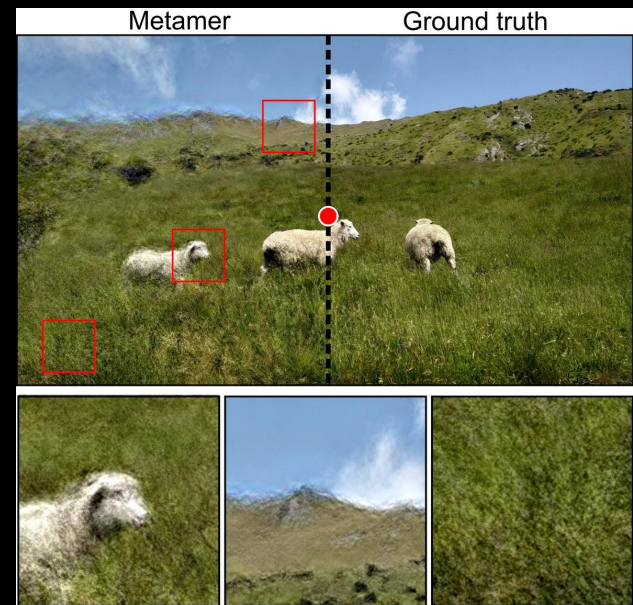
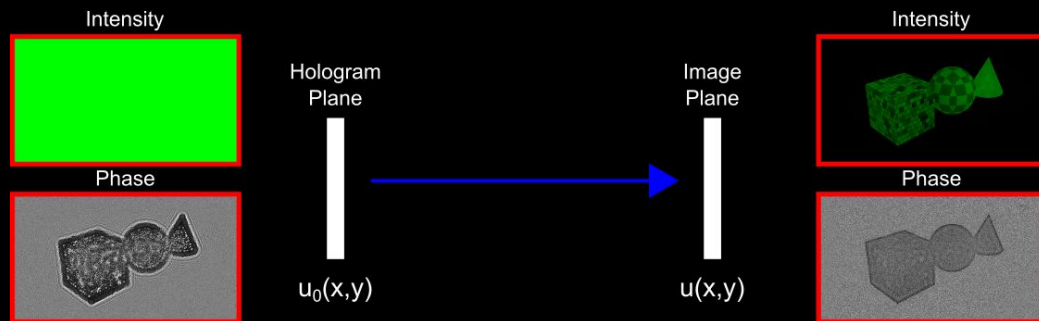


Image courtesy Interesting Engineering



Thank you for listening!

Practical models of perception in graphics

Rafał Mantiuk

UNIVERSITY OF
CAMBRIDGE
COMPUTER LABORATORY

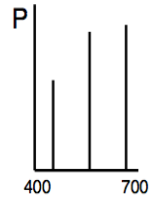


UNIVERSITY OF
CAMBRIDGE

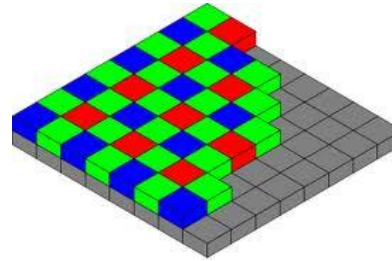
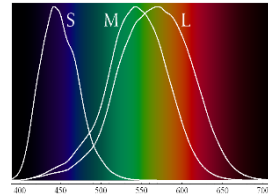


European
Research
Council

Perception in graphics, displays and cameras



*

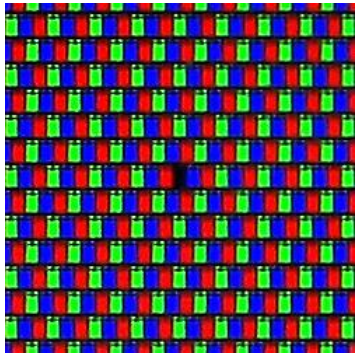


Camera's Bayer pattern

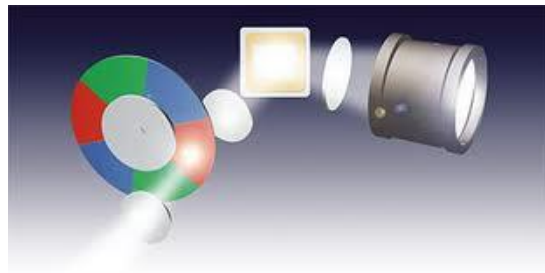
Display spectral emission - metamerism



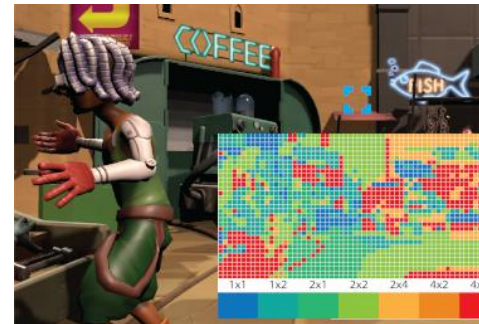
Foveated rendering



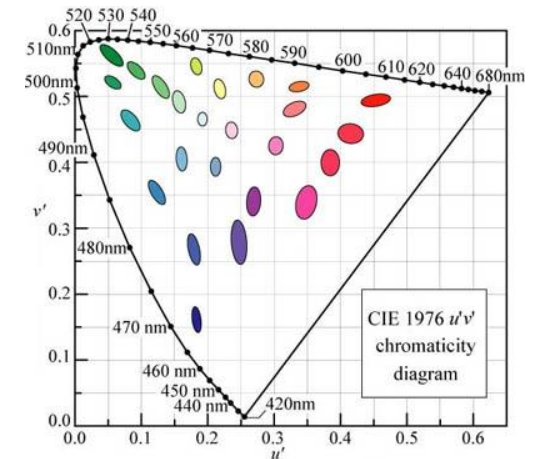
Display's subpixels



Color wheel in DLPs



Adaptive shading and refresh rate

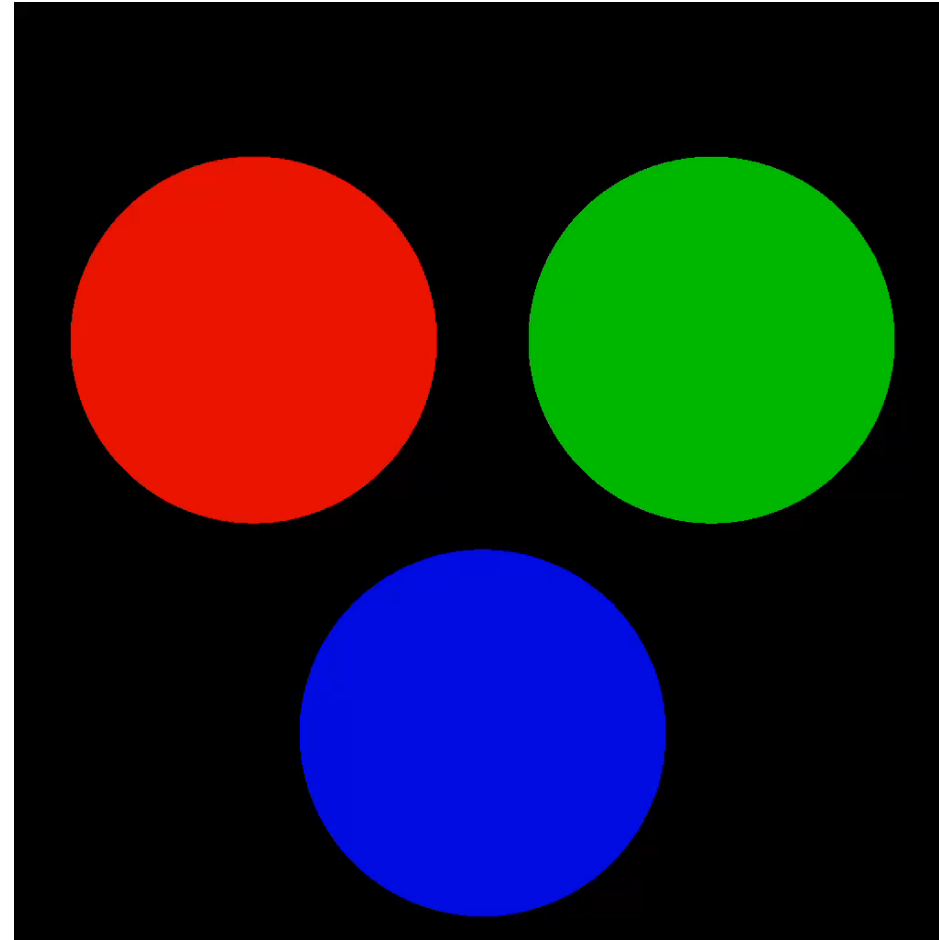


Uniform colour spaces

- If you cannot make it perfect, make it look good

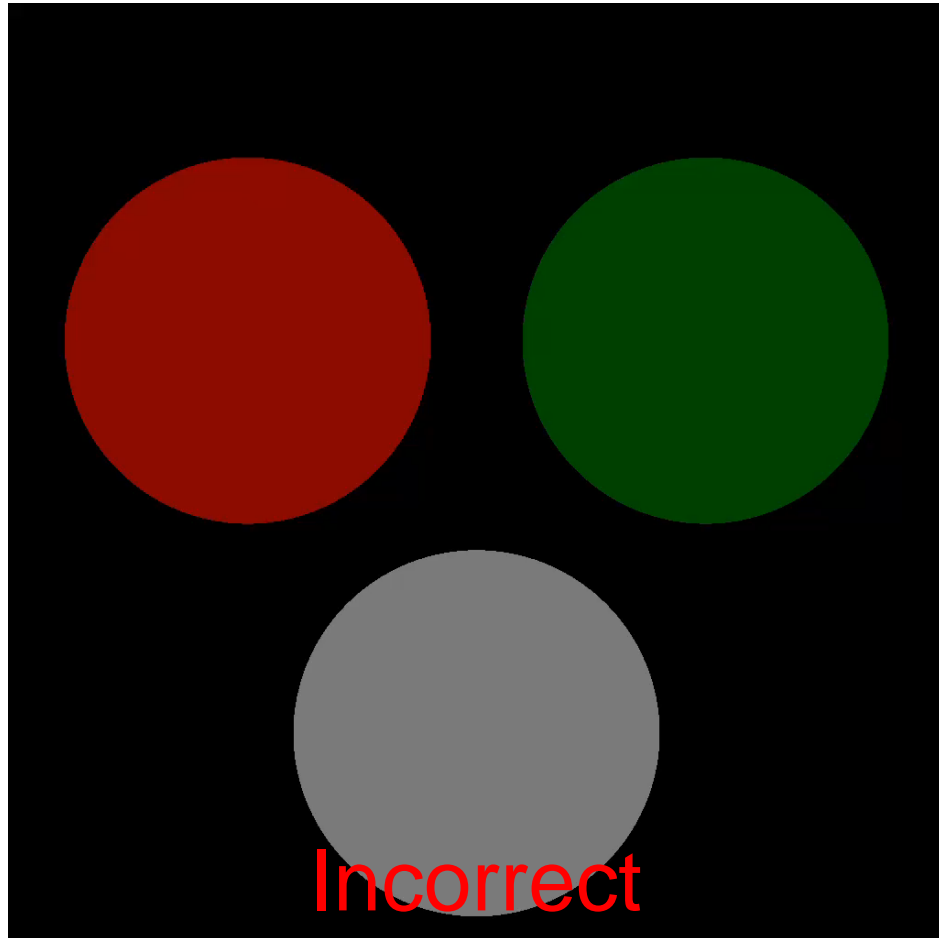
1. **Light and colour**
2. Sensitivity to luminance
3. Contrast sensitivity

Additive light mixture demonstration RGB

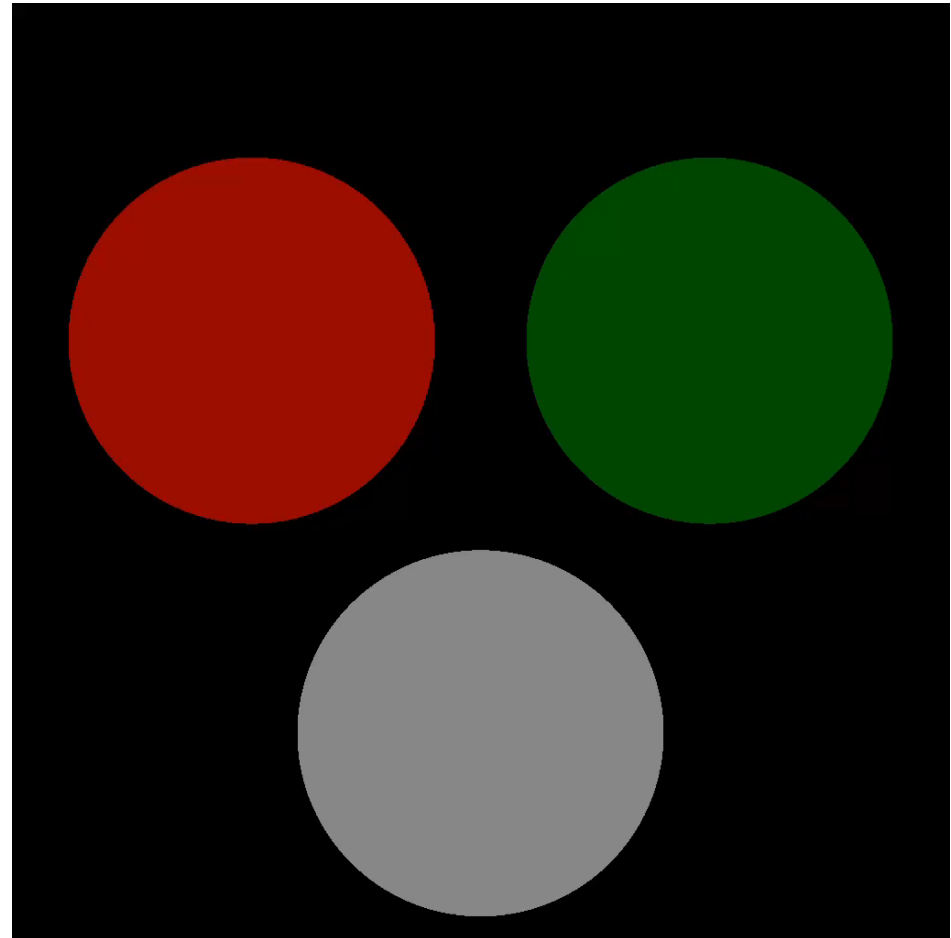


Additive light mixture demonstration RG + white

In display-encoded colour space



In linear colour space



Display encoded colour space

- Examples: sRGB, Adobe RGB, BT.2100 PQ RGB
- a.k.a. gamma-encoded or gamma-corrected (non HDR)
- The values sent to display (pixel values)
- Approximately perceptually uniform
- Can be efficiently encoded as integer

Linear colour space

- Examples: *linear* RGB, XYZ, LMS
- Used to model physics of light and **perception**
 - Can model **linear** mixtures of colours
 - Used in physically-based rendering
- Perceptually non-uniform
- Requires higher bit-depth (12-16 bits) or floating points

Linear RGB to display-encoded R'G'B'

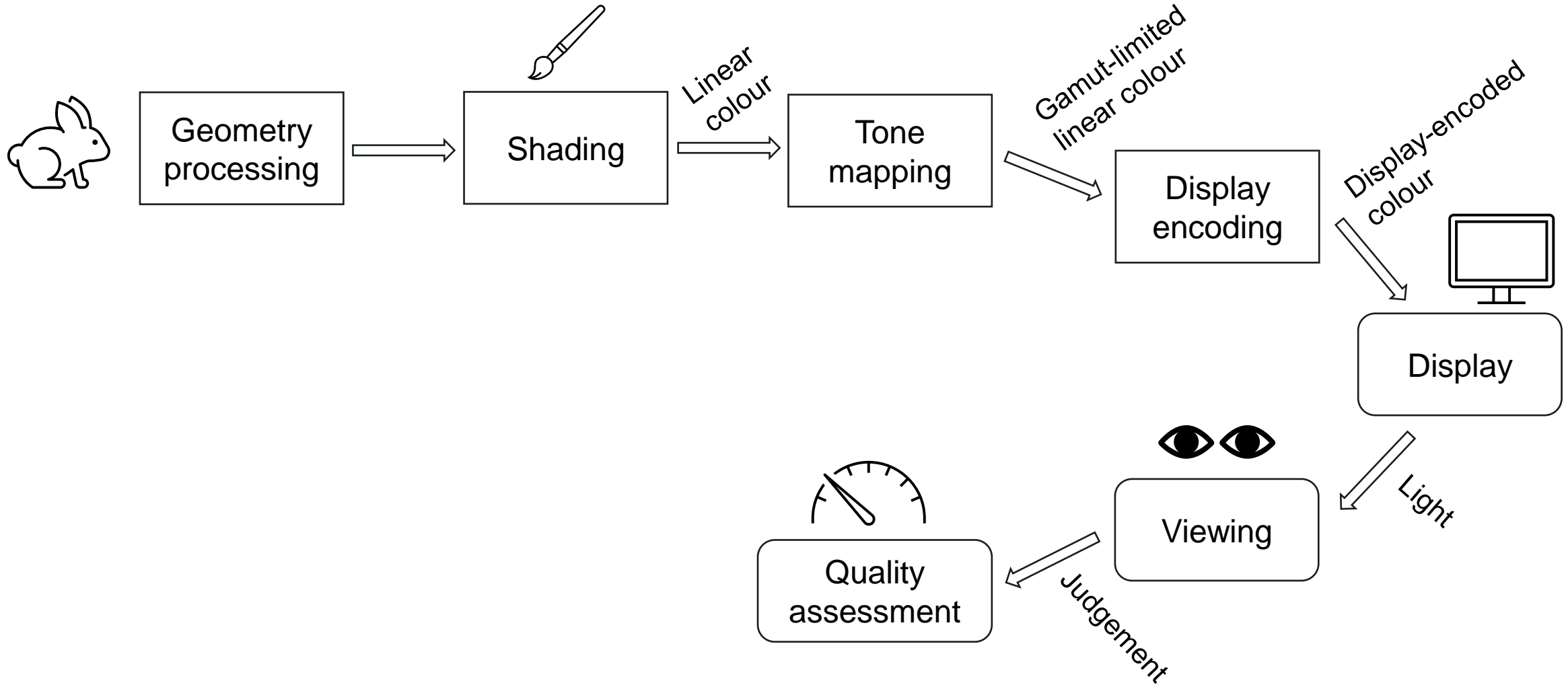
$$[R' \ G' \ B'] = [R^{1/2.2} \ G^{1/2.2} \ B^{1/2.2}]$$



Display-encoded R'G'B' to linear RGB

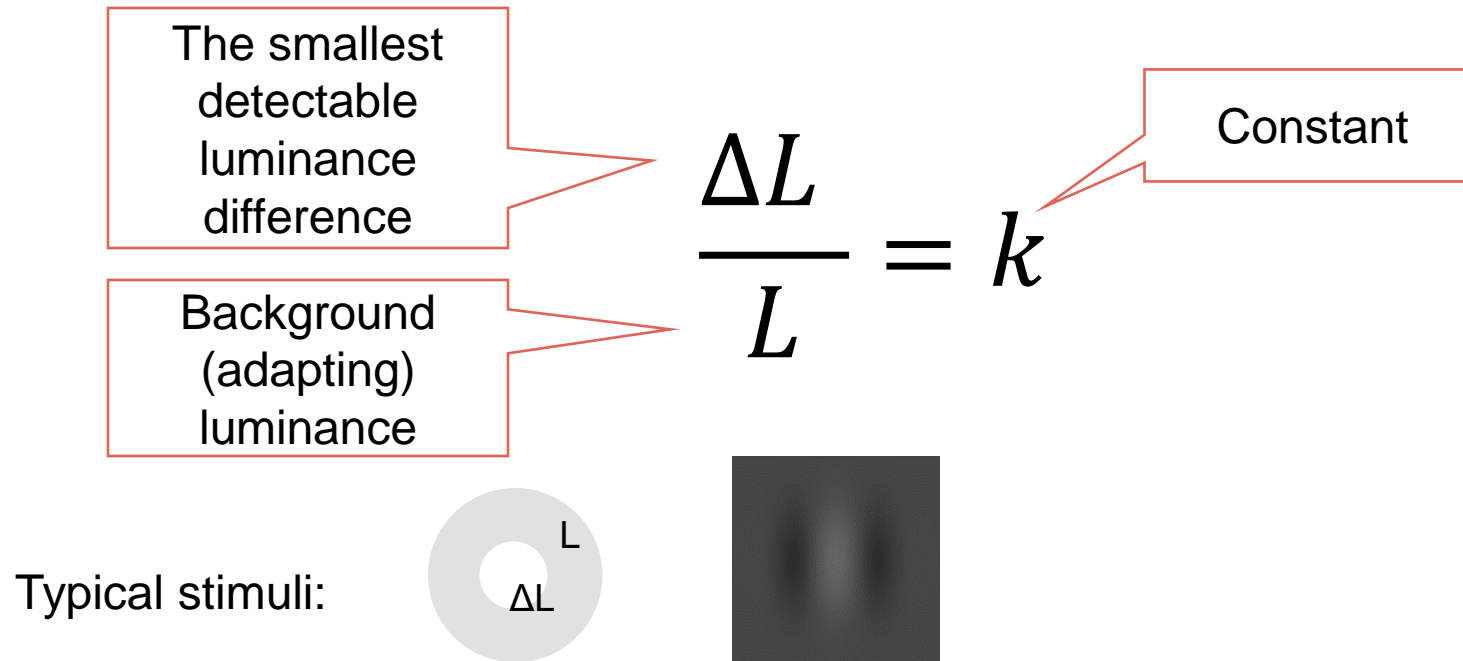
$$[R \ G \ B] = [R'^{2.2} \ G'^{2.2} \ B'^{2.2}]$$

Perceptual rendering pipeline



1. Light and colour
2. **Sensitivity to luminance**
3. Contrast sensitivity

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



Ernst Heinrich Weber
[From wikipedia]

- Smallest detectable difference in luminance

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

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

For k=1%

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

How to make luminance (more) perceptually uniform?

- Using “Fechnerian” integration

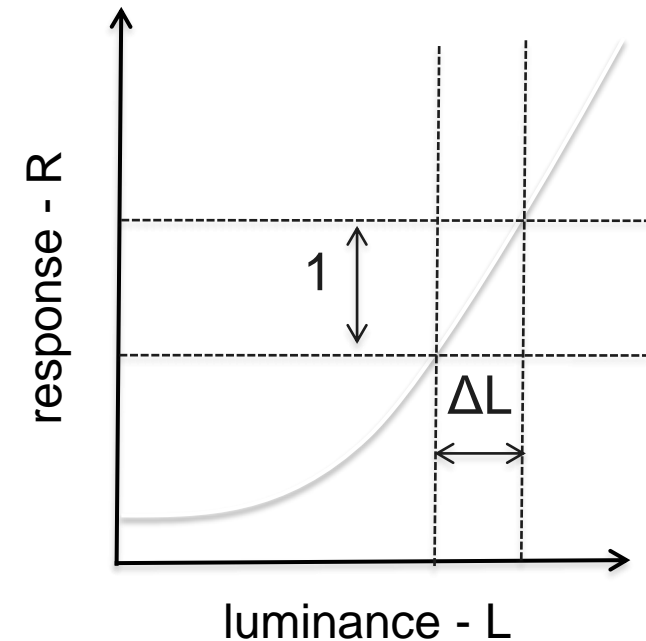
$$\frac{dR}{dl}(L) = \frac{1}{\Delta L(L)}$$

Derivative of
response

Detection
threshold

Luminance
transducer:

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



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

- and given the luminance transducer

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

- the response of the visual system to light is:

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

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

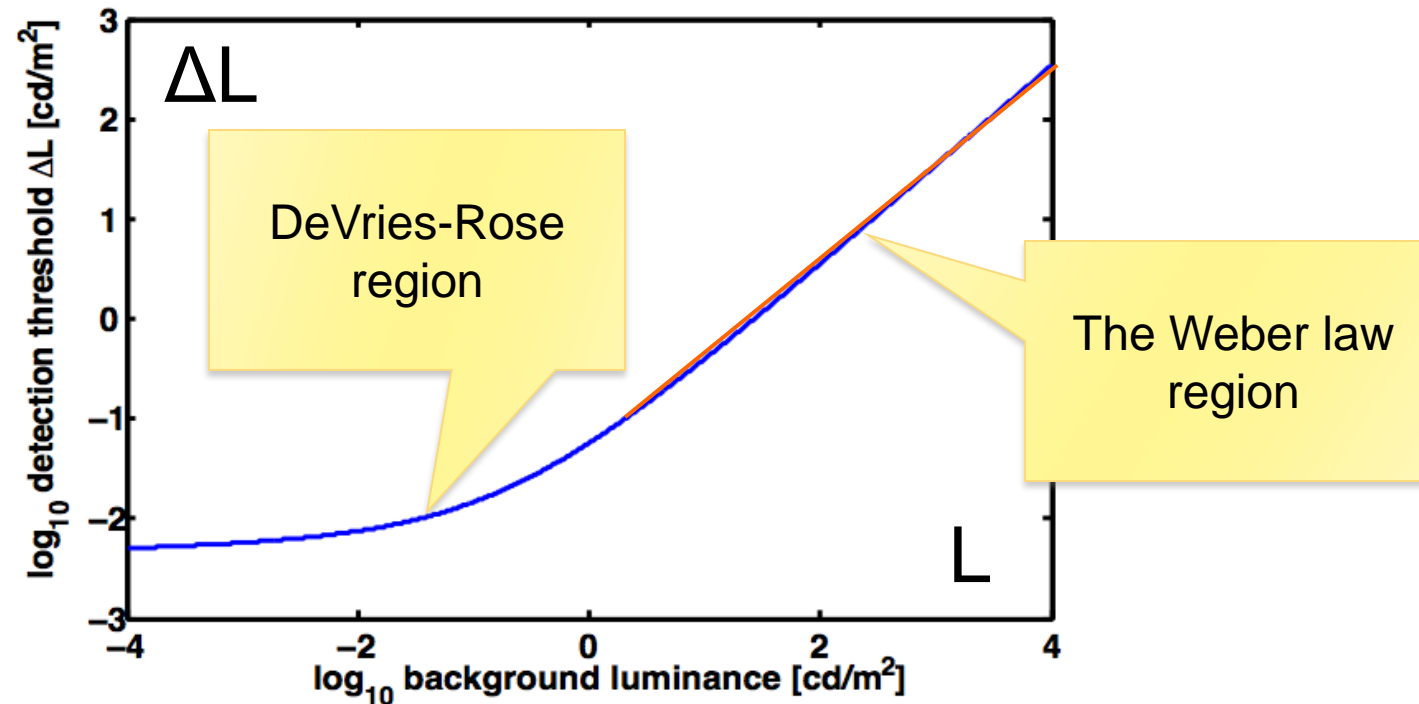
- Response of the visual system to luminance is **approximately** logarithmic
- This is a good 1st order approximation
 - Especially when only relative luminance values are known
- Luminance should be plotted only on the logarithmic scale



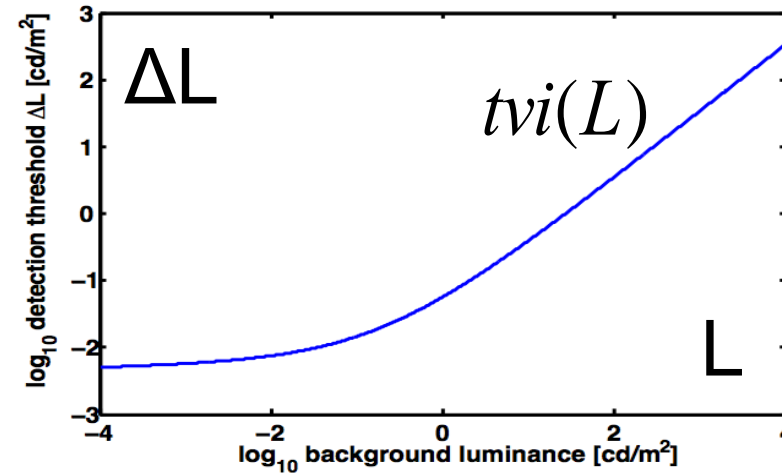
Gustav Fechner
[From Wikipedia]

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

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



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



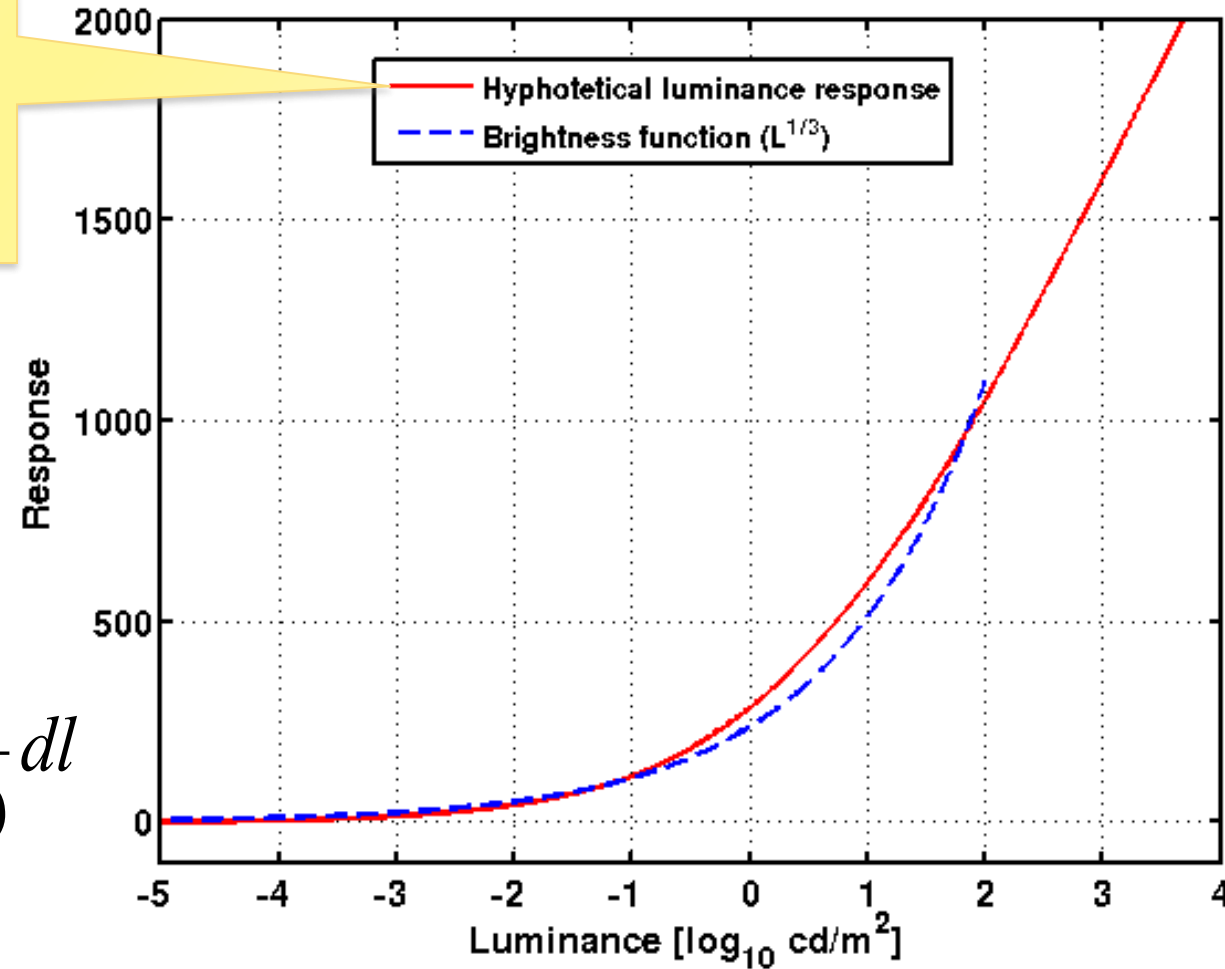
- we can get a more accurate estimate of the “response”:

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

Fechnerian integration and Stevens' law

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

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



- DICOM grayscale function
 - Function used to encode signal for medical 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
- HDR quality metrics
 - PU21-PSNR, HDR-VDP, HDR-VQM

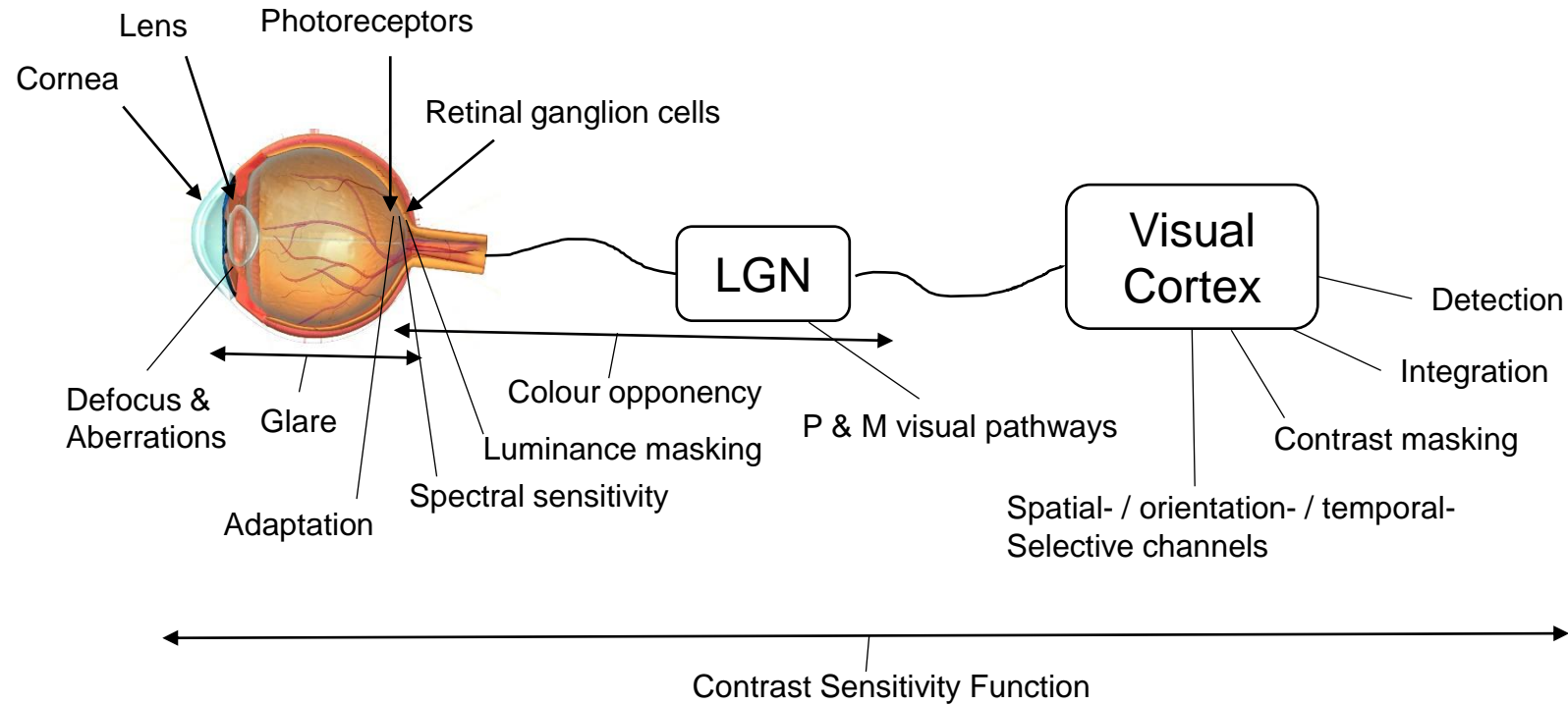


HDR 10

1. Light and colour
2. Sensitivity to luminance
3. **Contrast sensitivity**

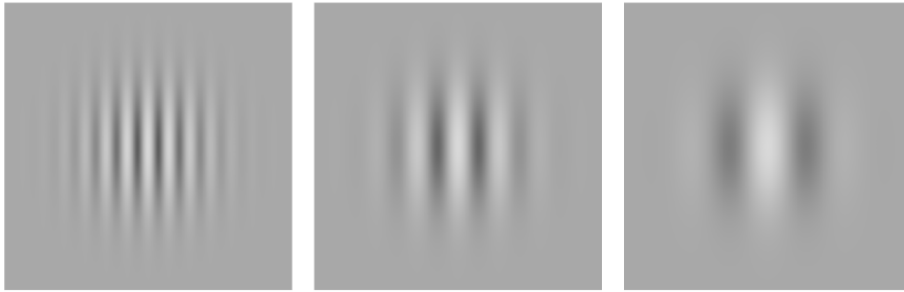
Contrast Sensitivity

Modeling contrast detection

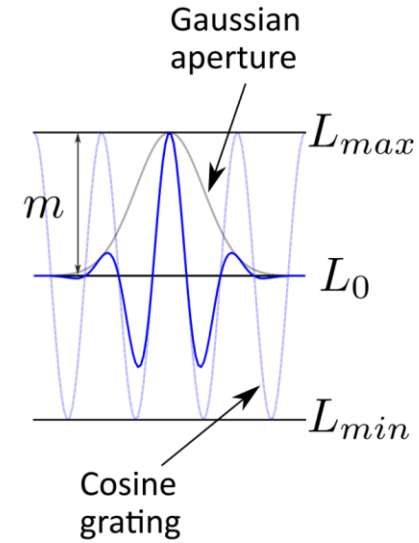
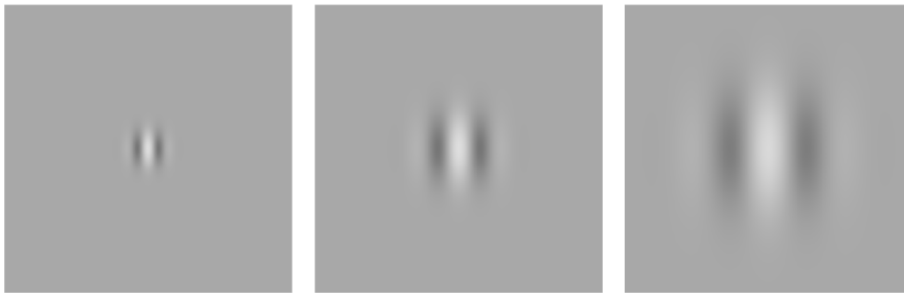


Gabor patches

Fixed size



Fixed cycles



$$s(x, t) = L_0 + m \cos 2\pi\rho x \cos 2\pi\omega t$$

Background
luminance

Modulation

Spatial
frequency

Gaussian
aperture

$$s'(x, y, t) = s(x, t) \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad [\text{cd/m}^2]$$

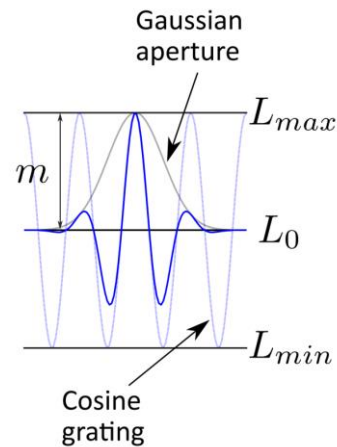
- 2-Alternative-Forced-Choice experiment

Does A or B contain the pattern?



- Michelson contrast

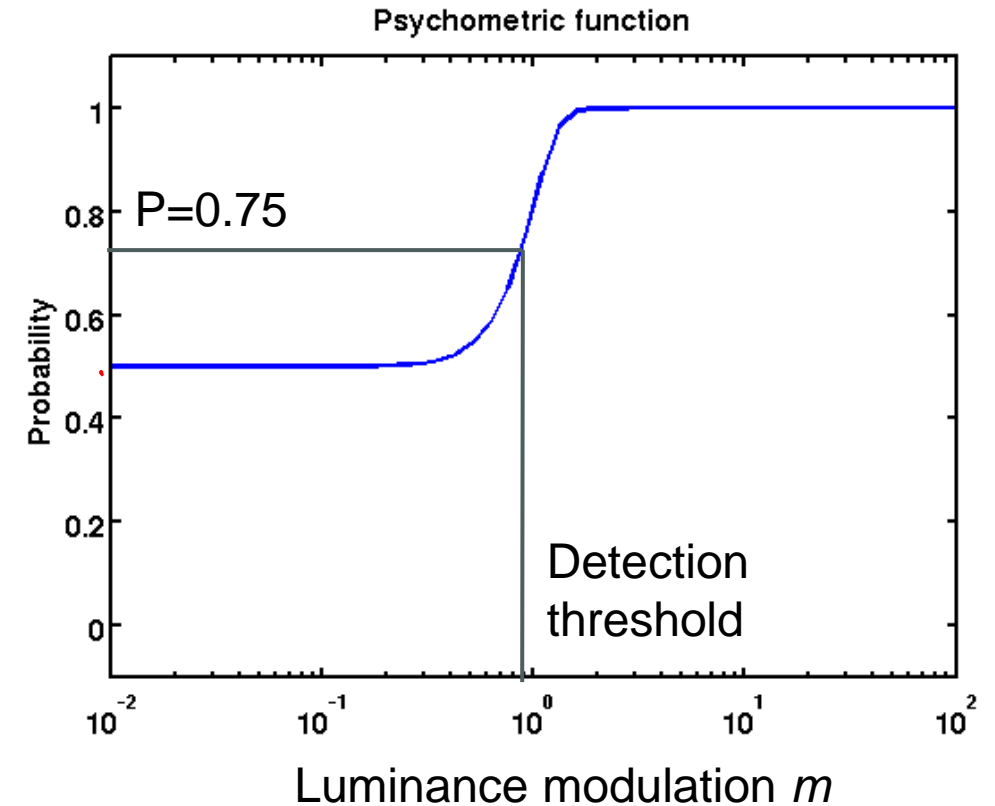
$$c = \frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}} = \frac{m}{L_0}$$

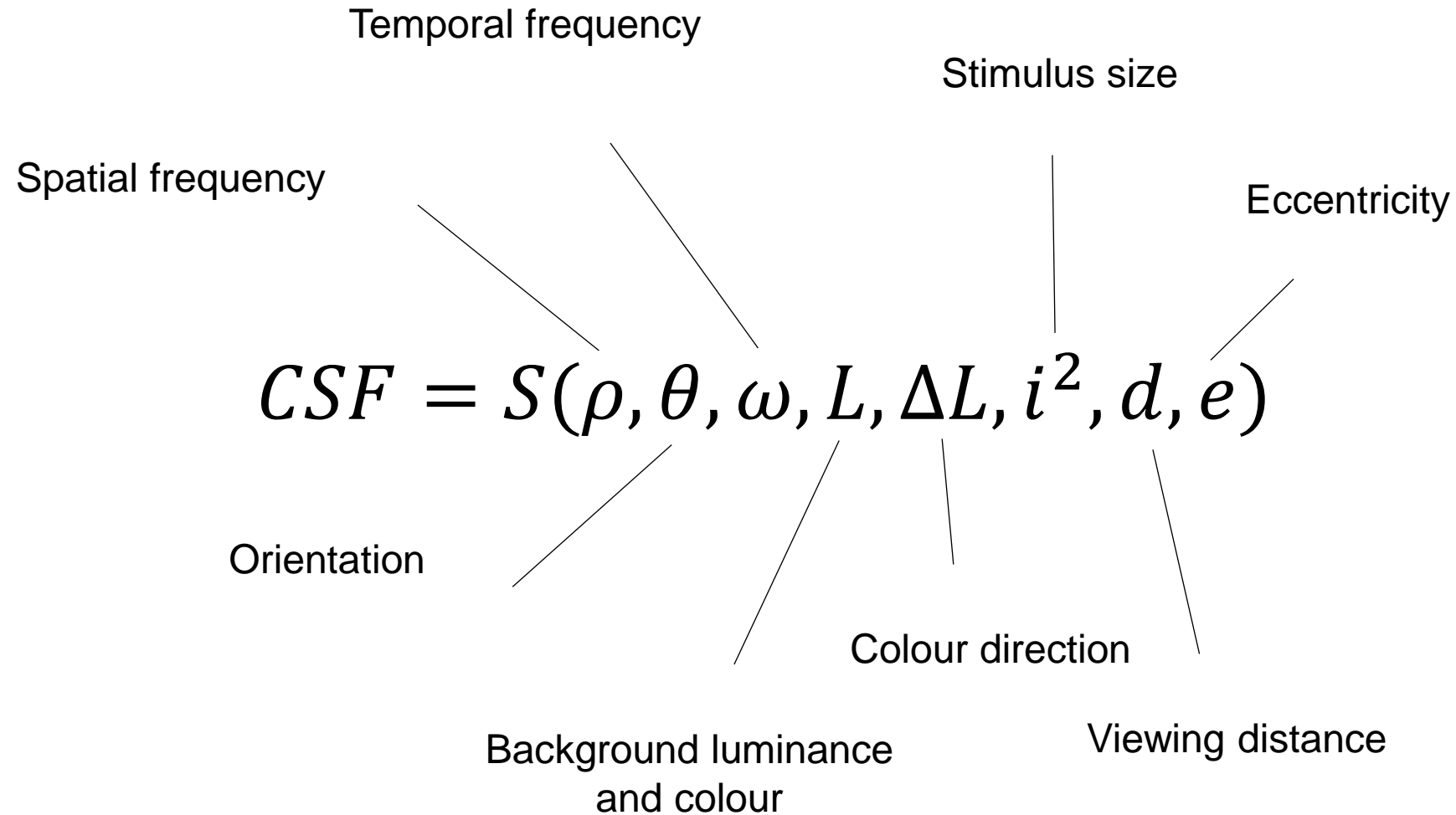


- Sensitivity

$$S = \frac{1}{c} = \frac{L_0}{m}$$

Contrast detected with probability 0.75





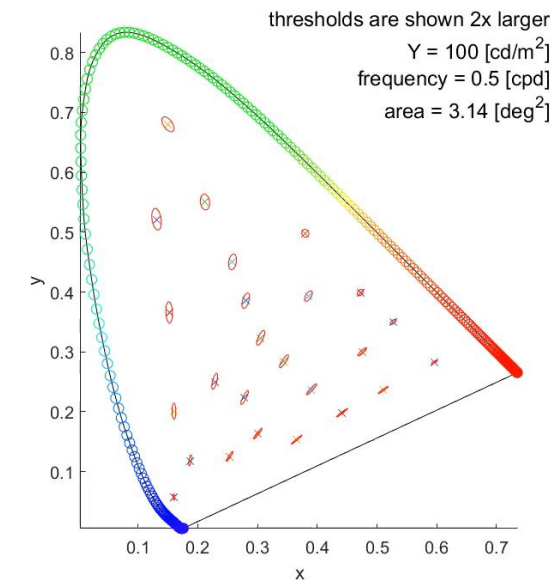
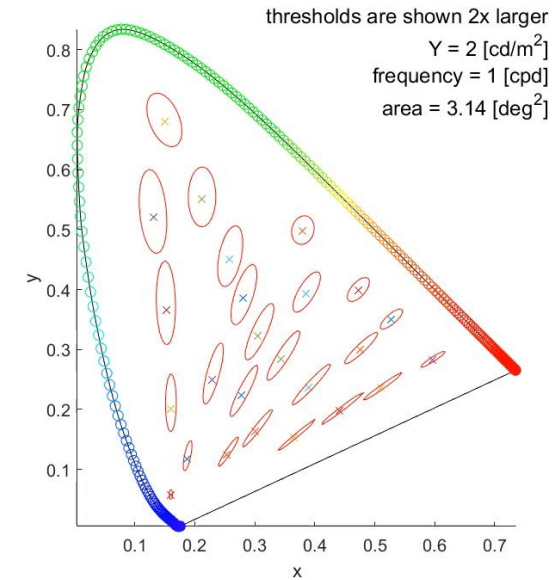
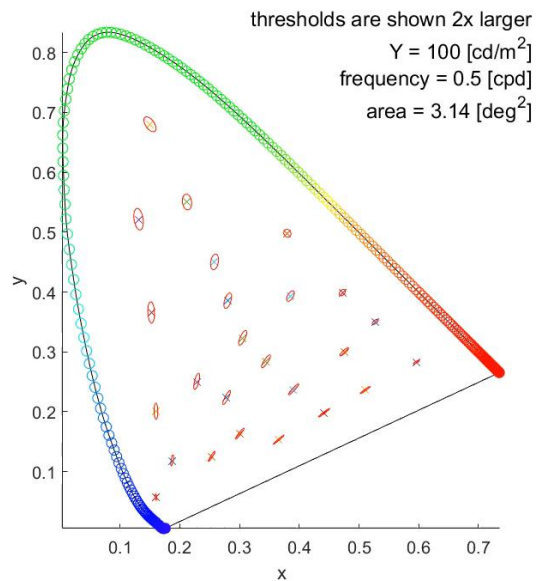
Contrast Sensitivity Models

	Spatial freq.	Temp. freq.	Luminance	Colour	Orientation	Eccentricity	Size
Barten's [1]	✓	✗	✓	✗	✓	✗	✓
scCSF [2]	✓	✗	✓	✓	✗	✗	✓
stelaCSF [3]	✓	✓	✓	✗	✗	✓	✓

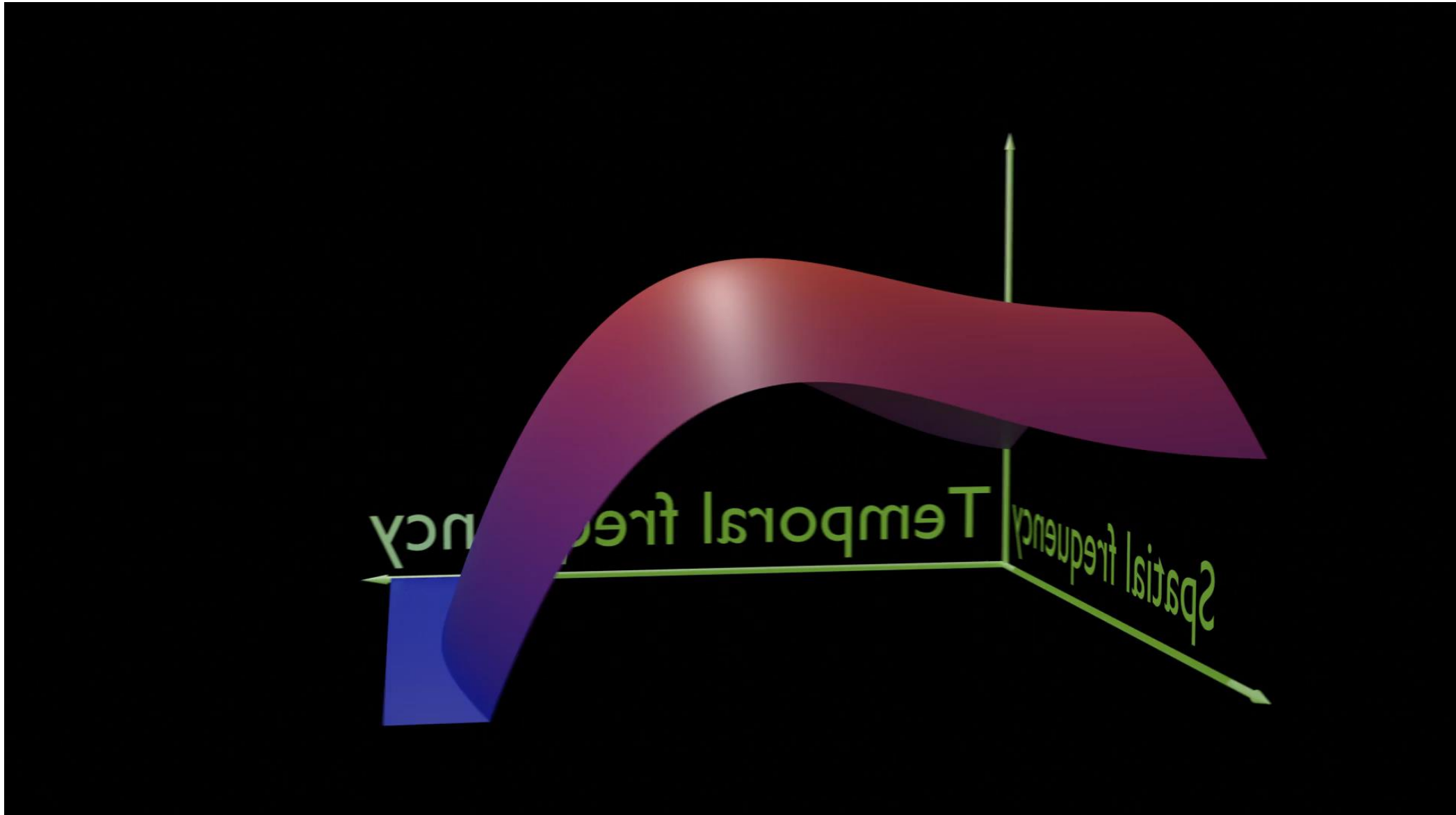
- [1] Barten, Peter G. J. *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality*. SPIE Press, 1999.
- [2] Mantiuk, Rafał K., Minjung Kim, Maliha Ashraf, Qiang Xu, M. Ronnier Luo, Jasna Martinovic, and Sophie Wuerger. "Practical Color Contrast Sensitivity Functions for Luminance Levels up to 10000 cd/m²." *Color and Imaging Conference 2020*, no. 28 (November 4, 2020): 1–6. <https://doi.org/10.2352/issn.2169-2629.2020.28.1>.
- [3] Mantiuk, Rafał K, Maliha Ashraf, and Alexandre Chapiro. "StelaCSF - A Unified Model of Contrast Sensitivity as the Function of Spatio-Temporal Frequency , Eccentricity , Luminance and Area." *ACM Transactions on Graphics* 41, no. 4 (2022): 145. <https://doi.org/10.1145/3528223.3530115>.

spatio-chromatic CSF [scCSF]

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



stelaCSF – Spatio-Temporal Eccentricity Luminance Area CSF

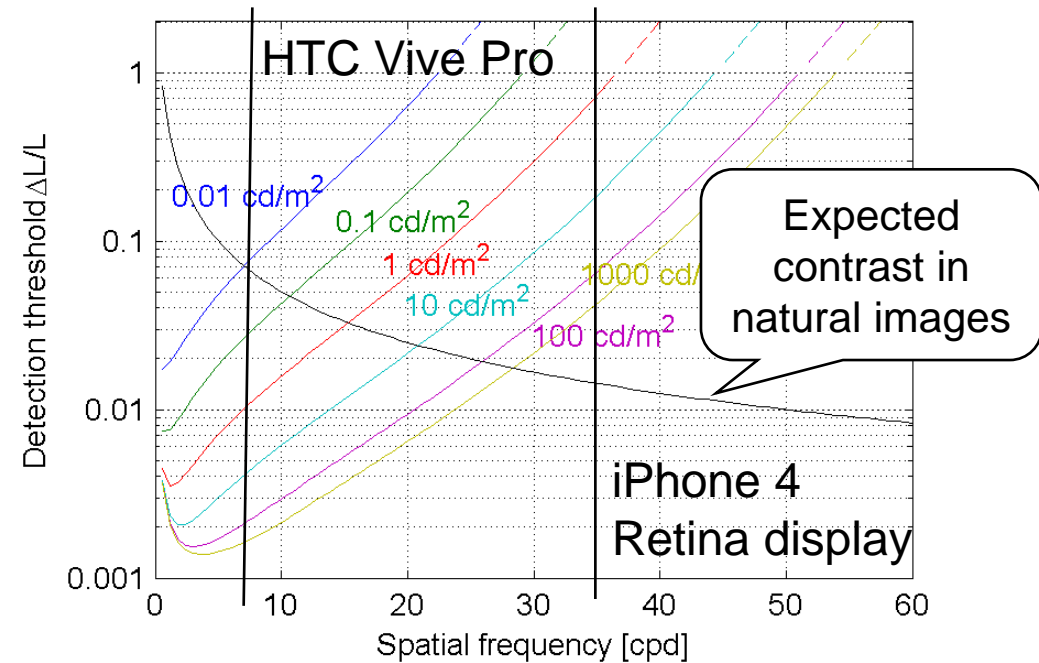


Example: CSF and the resolution

- CSF plotted as the detection contrast

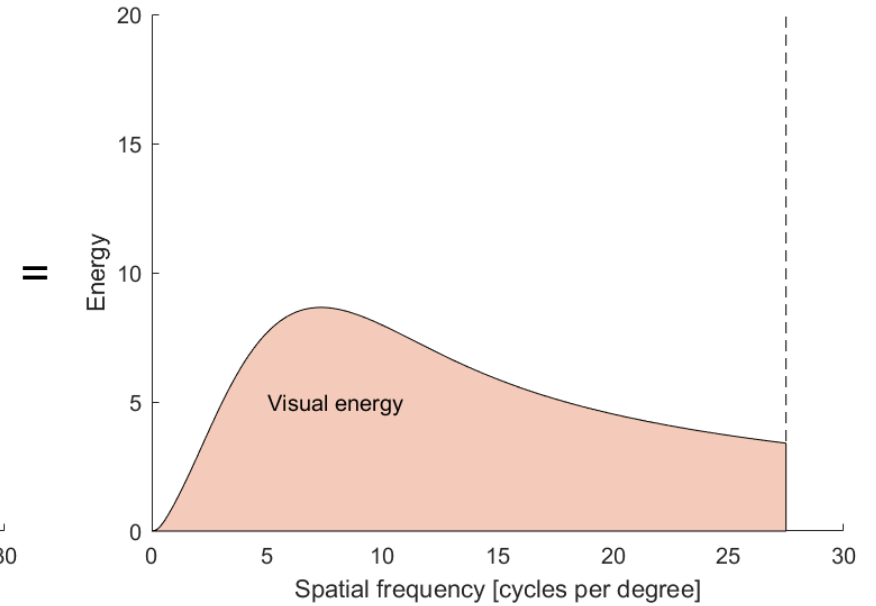
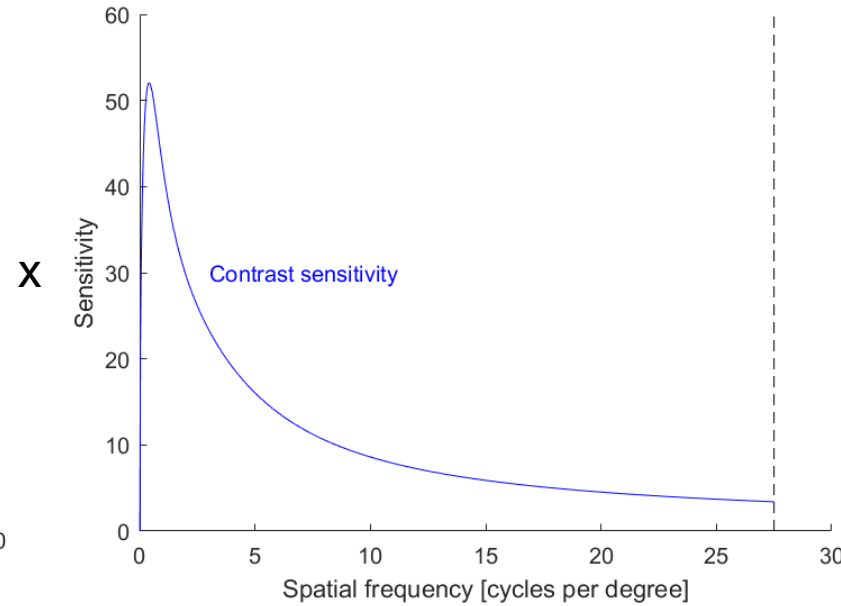
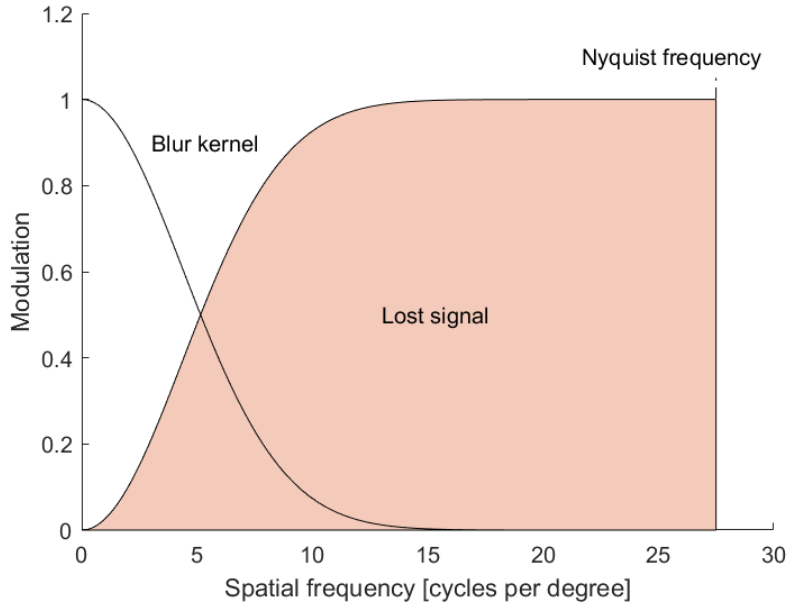
$$\frac{\Delta L}{L_b} = S^{-1}$$

- The contrast below each line is invisible
- Maximum perceivable resolution depends on luminance



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

Example: quality degradation due to blur



Quality degradation \propto Visual energy

1. Light and colour

- Perception and physics/optics is modelled in linear colour spaces

2. Sensitivity to luminance

- Logarithm of luminance accounts for Weber's law
- PU21 or PQ are better approximations – but rely on absolute values

3. Contrast sensitivity

- End-to-end model of contrast detection
- Sets the limits of visual system, helpful in quality prediction