

# Digital Image Processing

---

Daniel T. McGuiness, PhD

Version: γ.2024.5



MCI



# Table of Contents

1. Lecture Structure
2. Mathematical Fundamentals
3. Perception
4. Image Formats
5. Camera
6. Display
7. Noise
8. Histogram Operations
9. Morphological Operations
10. Appendix

## Lecture Structure

---



# Table of Contents

## First Steps

Introduction

Individual Assignment

Group Assignment

Point Distribution

Point Distribution

Resources



- The goal of this lecture is to give you the fundamentals of digital image processing and understanding of mathematical principles.
- This lecture is a total of **4 SWS** with a total of sixty (**60**) hours.
- There are two (**2**) assignments for this course
  - 1<sup>st</sup>** will be a pre-defined work which is individual based.
  - 2<sup>nd</sup>** will be group based.

You are to come up with a project that uses DIP using Python.

- You will work with a group of up to three (**3**) or two (**2**).
- You are to come up with a group and decide on your topic.



- The individual assignment focuses on understanding DIP principles.
- The assignment is uploaded to SAKAI for you to work on along with what is required of you for submission.
  - The assignment contains questions where applications of DIP will be needed.
- The deadline is the end day of **last lecture before presentations**.

## A Help in Colour

Due to the nature of the topic, some aspects are to be presented in a colour spectrum some student may not be able to perceive. In situations like this, please let me know if there are some diagrams or some colour choices making the lecture illegible via mail and I will send you a colour correct version based on the condition.



- For your project use Python.
- Some possible project ideas:
  - License plate detection,
  - Handwriting detection,
  - Signature verification,
  - Face detection,
  - Image to text conversion,
  - Barcode detection,
  - Convert sudoku drawings to computer code.
  - Book detection.

The use of AI/ML is allowed as long as clear explanation is given and its process is understood.



- The last three (3) appointments are reserved for group presentations.
- You will do a presentation in front of the class for 20 mins.
- The next 20 mins following your presentation will be the Q&A.
- The Q&A will involve two (2) questions from your relevant work.
- You are also to submit a report with your project detailing the work.

Each student needs to declare the part the student worked on.

- i.e., Student A has done the writing, edge detection
- i.e., Student B has done the data analysis, figure generation.
- You are to submit your reports and all relevant resources to SAKAI no later than 2 weeks before your assigned presentation.



Assessment Type	Overall Points	Breakdown	%
Homework	40		
		Report	20
		Solution(s)	60
		Code Analysis	20
Group Project	60		
		Report	40
		Presentation	40
		Q & A	20

**Table 1:** Assessment Grade breakdown for the lecture.



Covered Topic	Appointment
Mathematical Fundamentals	1
Perception	2
Camera	2-3
Display	4
Noise	4-5
Histogram Operations	6
Morphological Operations	7
Blurring Filters	8
Feature Analysis	9
Edge Detection	10
Neural Networks for Image Processing	11-12
Group Assignment Presentations	13-15

**Table 2:** Distribution of materials across the semester.



## Mathematical Fundamentals

- 2D Convolution,
- Discrete Fourier Transform,
- Sampling Theorem





## Perception

- Colour Blindness,
- Colour Standards,
- Colour Models





## Cameras

- Used sensors,
- Lenses,
- Sensitivity





## Displays

- Dithering,
- Interlacing,
- Display Technologies





## Noise

- Types of noises,
- Modelling Noises,
- Random Noise generation





## Histogram Operations

- Colour Channels,
- Masking,
- Dynamic Range





## Morphological Operations

- Opening,
- Closing,
- Erosion,
- Dilation.





## Blurring Filters

- Gaussian Blurring,
- Multivariate Distribution,
- Bilinear Filtering





## Feature Analysis

- ORB Feature Extractor,
- Adaptive Threshold,
- Scale Invariant Feature Transform.





## Edge Detection

- Defining an Edge to the computer,
- Types of Kernels,
- Canny Edge Detection.





## Neural Networks for Image Processing

- Defining ANNs,
- OCR,
- ResNet.





## Books

- Forsyth, Ponce "*Computer Vision: A Modern Approach*" Prentice-Hall, 2003.
- Young I. "*Fundamentals of Image Processing*" Delft 1998.
- Szeliski R. "*Computer Vision: Algorithms and Applications*" Springer 2022.
- Nixon M. et. al "*Feature Extraction and Image Processing for Computer Vision*" Academic press 2019.
- Gonzalez R. "*Digital Image Processing*" Pearson 2009



## White Papers

- Luminera "*Getting it Right: Selecting a Lens for a Vision System*",
- Luminera "*The Complete Guide to Industrial Camera Lenses*",
- Fowler B, et. al, *Read Noise Distribution Modeling for CMOS Image Sensors*.
- Oxford Instruments *Understanding Read Noise in sCMOS Cameras*.



## Lecture Notes

- Applied Multi-variable Statistical Analysis "*Lesson 4: Multivariate Normal Distribution*",
- Statistical Theory and Methods I "*Chapter 3: Multivariate Distributions*", Stephen M. Stigler
- The Discrete Fourier Transform "*Signal Processing & Filter Design*", Stephen Roberts.
- Procedural Generation: 2D Perlin Noise *Game Programming*, Mount .E, Eastman R.
- Foundations of computer vision: Lecture notes, Carreira-Perpinan M.
- Computer Vision, CMU School of Computer Science
- Computer Vision, University of Cambridge
- Computer Vision, NYU Computer Science



## Web Resources

- Scikit-image documentation
- OpenCV documentation
- Pillow (fork of PIL) documentation

# Mathematical Fundamentals

---



## Learning Outcomes

### Convolution

Introduction

2D Convolution Example

### Signal Sampling

### Nyquist Sampling Theorem

Statistical Properties

### Information Theory

Information and Entropy



## Learning Outcomes

- (LO1) An Overview of Mathematical Methods,
- (LO2) Description of Analogue and Digital,
- (LO3) Fourier Analysis Overview,
- (LO4) Convolution Introduction.





- Computer Vision encompasses multiple disciplines, including digital image processing, cameras and displays.
- To better prepare, it is important to refresh/learn some fundamental mathematical principles & concepts.

## Concepts and Principles

- Convolution
- Fourier Analysis
  - Properties
  - Discrete Fourier Transform
- Shannon-Nyquist Sampling Theorem
- A brief introduction to Information Theory
  - Entropy in Information Theory



- Convolution, mathematically is defined as:

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau.$$

In layman's terms convolution is just fancy multiplication.



## Example

Imagine you manage a hospital treating patients with a single disease.

You have:

**Treatment Plan** [3] Every patient gets 3 units of the cure on their first day.

**Patient List** [1, 2, 3, 4, 5] Your patient count for the week (1 person Monday, 2 people on Tuesday, etc.).

How much medicine do you use each day?



## Solution

The answer is a quick multiplication:

$$\text{Plan} \times \text{Patients} = \text{Daily Usage}$$

$$3 \times [1, 2, 3, 4, 5] = [3, 6, 9, 12, 15]$$

Multiplying the plan by the patient list gives usage for upcoming days:

$$[3, 6, 9, 12, 15]$$

Everyday multiplication of  $(3 \times 4)$  means using the plan with a single day of patients:

$$[3] \times [4] = [12]$$



## Example

Now the disease mutates and needs multi-day treatment. A new plan:

Plan: [3, 2, 1]

Meaning:

- 3 units of the cure on day one,
- 2 units on day two,
- 1 unit on day three.

Given the same patient schedule of:

Patient: [1, 2, 3, 4, 5]

what's our medicine usage each day?



## Solution

Let's see

- On day 1, 1 patient A comes in. It's their first day, so 3 units.
- On day 2, A gets 2 units (second day), but two new patients ( B1 & B2 ) arrive, who get 3 each ( $2 \times 3 = 6$ ).
  - The total is  $2 + (2 \times 3) = 8$  units.
- On Wednesday, it's trickier: The patient A finishes (1 unit, her last day), the B1 and B2 get 2 units ( $2 * 2$ ), and there are 3 new Wednesday people....

The patients are overlapping and it's hard to track. How can we organise this calculation?



## Solution

An idea worth considering is to **reverse the order** of the patient list:

New Patient List: [5, 4, 3, 2, 1]

Next, imagine we have 3 separate rooms where we apply the proper dose:

Rooms: [3, 2, 1]

On your first day, you walk into the first room and get 3 units of medicine. The next day, you walk into room #2 and get 2 units. On the last day, you walk into room #3 and get 1 unit. There's no rooms afterwards, and your treatment is done.



## Solution

To calculate the total medicine usage, line up the patients and walk them through the rooms:

Monday	
Rooms	-----
Patients	3 2 1
Usage	3

On Monday (our first day), we have a single patient in the first room. A gets 3 units, for a total usage of 3.

Makes sense, right?



## Solution

On Tuesday, everyone takes a step forward:

Tuesday

Rooms                    3 2 1  
Patients ->        5 4 3 2 1

Usage                    6 2        = 8

The first patient is now in the second room, and there's 2 new patients in the first room. We multiply each room's dose by the patient count, then combine.



## Solution

Wednesday

Rooms	3	2	1		
Patients ->	5	4	3	2	1
Usage	9	4	1	=	14

Thursday

Rooms	3	2	1		
Patients ->	5	4	3	2	1
Usage	12	6	2	=	20

Friday

Rooms	3	2	1		
Patients ->	5	4	3	2	1
Usage	15	8	3	=	26



## Solution

It's intricate, but we figured it out, right? We can find the usage for any day by reversing the list, sliding it to the desired day, and combining the doses.

The total day-by-day usage looks like this (don't forget Sat and Sun, since some patients began on Friday):

```

Plan      * Patient List   = Total Daily Usage
[3 2 1]  * [1 2 3 4 5]   = [3 8 14 20 26 14 5]
          M T W T F       M T W T F S S

```

This calculation is the convolution of the plan and patient list. It's a fancy multiplication between a list of input numbers and a "program".



## Example

Write a script which does convolution of the following two (2) arrays:

$$A = [1, 1, 2, 2, 1]$$

$$B = [1, 1, 1, 3]$$



## Solution

```
import numpy as np
def convolve_1d(signal, kernel):
    kernel = kernel[::-1]
    k = len(kernel)
    s = len(signal)
    signal = [0]*(k-1)+signal+[0]*(k-1)
    n = s+(k-1)
    res = []
    for i in range(s+k-1):
        res.append(np.dot(signal[i:(i+k)], kernel))
    return res
```

```
A = [1,1,2,2,1]
B = [1,1,1,3]

print(convolve_1d(A, B))
```



- An operation on two functions ( $f$  and  $g$ ) that produces  $f * g$ .

It expresses how the shape of one is modified by the other.

- There are several notations to indicate convolution with the most common is:

$$c = f(t) * g(t) = (f * g)(t),$$



- In 2D continuous space (i.e., **analogue**):

$$\begin{aligned} c(x, y) &= f(x, y) * g(x, y), \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\chi, \xi) g(x - \chi, y - \xi) d\chi d\xi. \end{aligned}$$

- In 2D discrete space (i.e., **digital**):

$$\begin{aligned} c[m, n] &= f[m, n] * g[m, n], \\ &= \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} f[j, k] g[m - j, n - k]. \end{aligned}$$



- It is the **single most important technique** in Digital Signal Processing.
- Using the strategy of impulse decomposition, systems are described by a signal called the impulse response.
- Convolution is important because it relates the three **(3)** signals of interest:
  1. Input signal,
  2. Output signal,
  3. Impulse response.



## Commutative

- The order in which we convolve two signals does not change the result:

$$f(t) * g(t) = g(t) * f(t)$$

## Distributive

- if there are three signals  $f(t), g(t), h(t)$ , then the convolution of  $f(t)$  is distributive:

$$f(t) * [g(t) + h(t)] = [f(t) * g(t)] + [f(t) * h(t)]$$



## Associative

- The way in which the signals are grouped in a convolution does not change the result:

$$f(t) * [g(t) * h(t)] = [f(t) * g(t)] * h(t)$$



## Shift Property

- The convolution of a signal with a time shifted signal results a shifted version of that signal.
- i.e.,

$$f(t) * g(t) = y(t)$$

- Then according to the shift property of convolution:

$$f(t) * f(t - T_0) = y(t - T_0)$$

- Similarly:

$$f(t - T_0) * f(t) = y(t - T_0)$$

- Therefore:

$$f(t - T_1) * f(t - T_2) = y(t - T_1 - T_2)$$



**Figure 1:** A visual representation of how convolution works in 2D.



- Converting from a continuous 2D data  $a(x, y)$  to its digital representation  $a[x, y]$  requires the process of **sampling**.
- An ideal sampling system is defined as the image  $a(x, y)$  multiplied by an ideal 2D impulse train  $\delta(x, y)$ :

$$\begin{aligned} b[m, n] &= a(x, y) \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \sum_{x=-\infty}^{+\infty} \delta(x - mX_0, y - nY_0) \\ &= \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} a(mX_0, nY_0) \delta(x - mX_0, y - nY_0). \end{aligned}$$

where  $X_0$  and  $Y_0$  are the sampling distance or intervals and  $\delta$  is the Dirac delta function.

- If you were to sample in square shapes  $X_0 = Y_0$  where you could think of each individual block a pixel



- To reconstruct a continuous analog signal from its sampled version accurately, the sampling rate must be at least **twice the highest frequency** present in the signal.
- This ensures that there are enough samples taken per unit of time to capture all the details of the original waveform without introducing aliasing, which can cause distortion or artifacts in the reconstructed signal.

$$f_s \geq 2f_m$$

where  $f_s$  is the signal frequency,  $f_m$  is the maximum sample frequency.

This is only a theoretical limit, not a practical one.



**Figure 2:** The effects of signal reconstruction on the sampling rate.

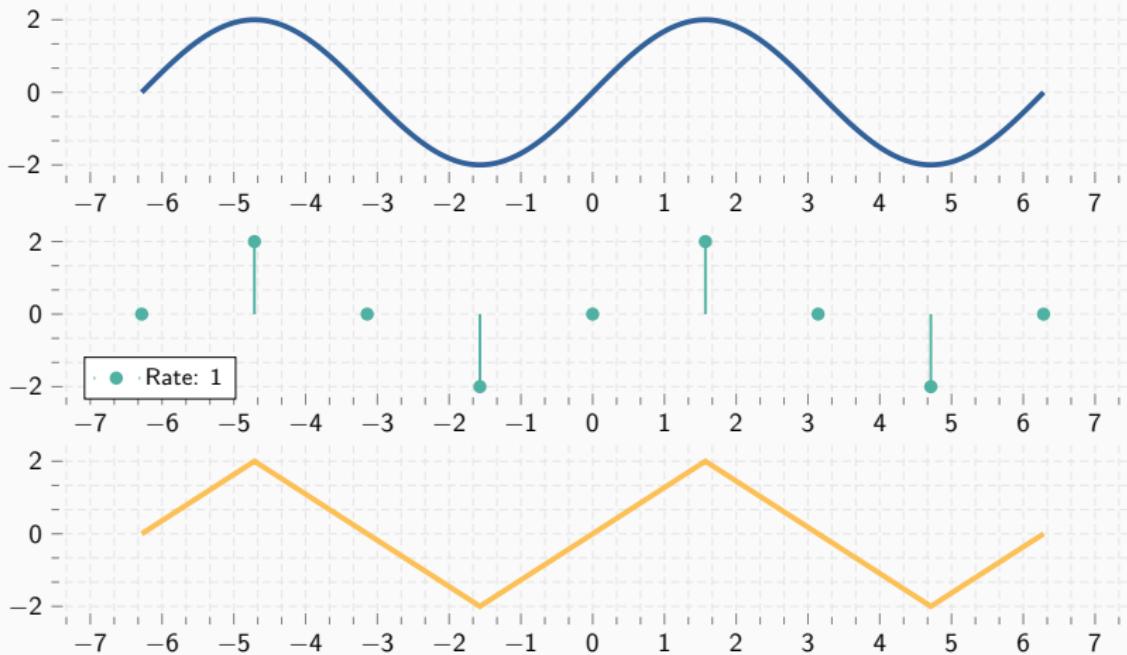


Figure 3: Reconstruction of the signal with 1 times the signal frequency.

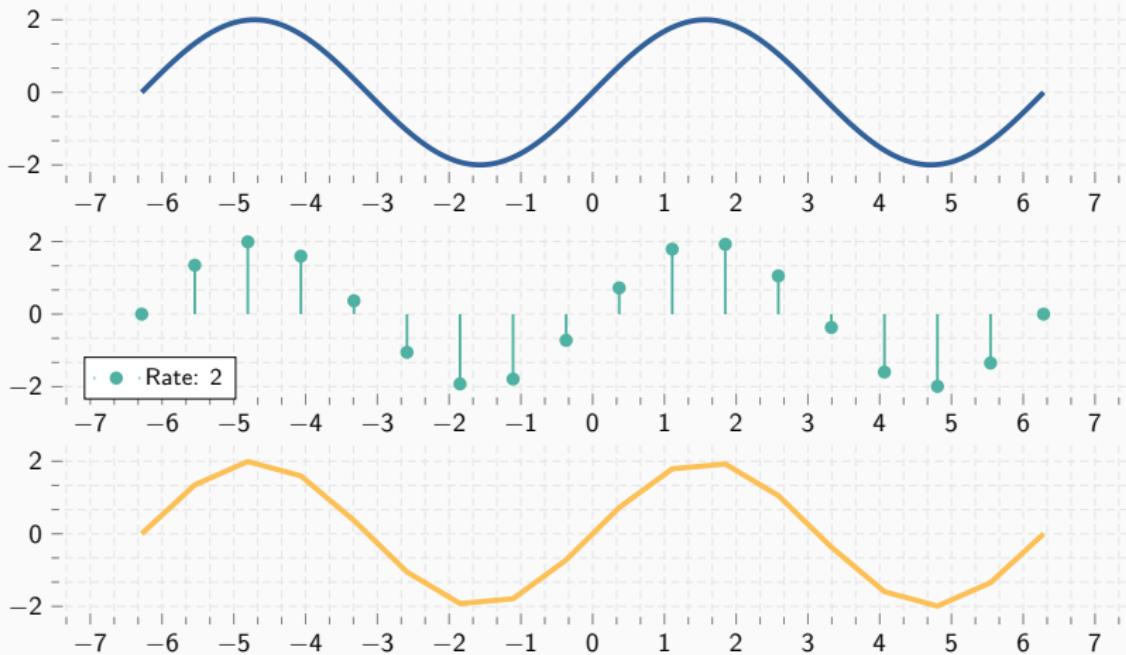


Figure 4: Reconstruction of the signal with 2 times the signal frequency.

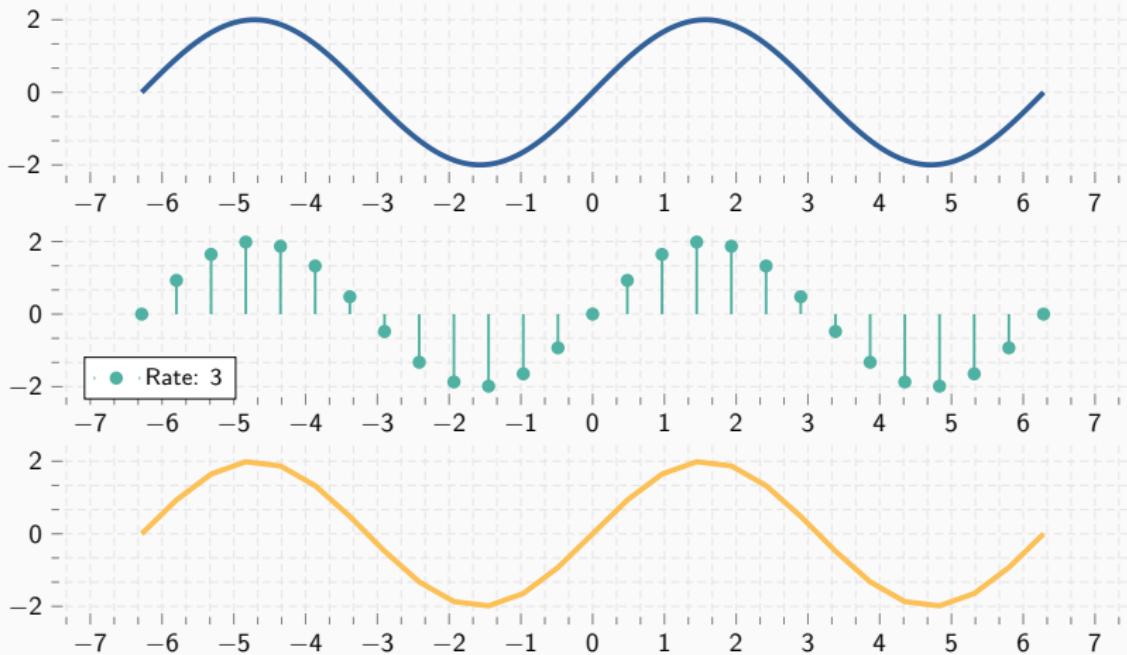


Figure 5: Reconstruction of the signal with 3 times the signal frequency.

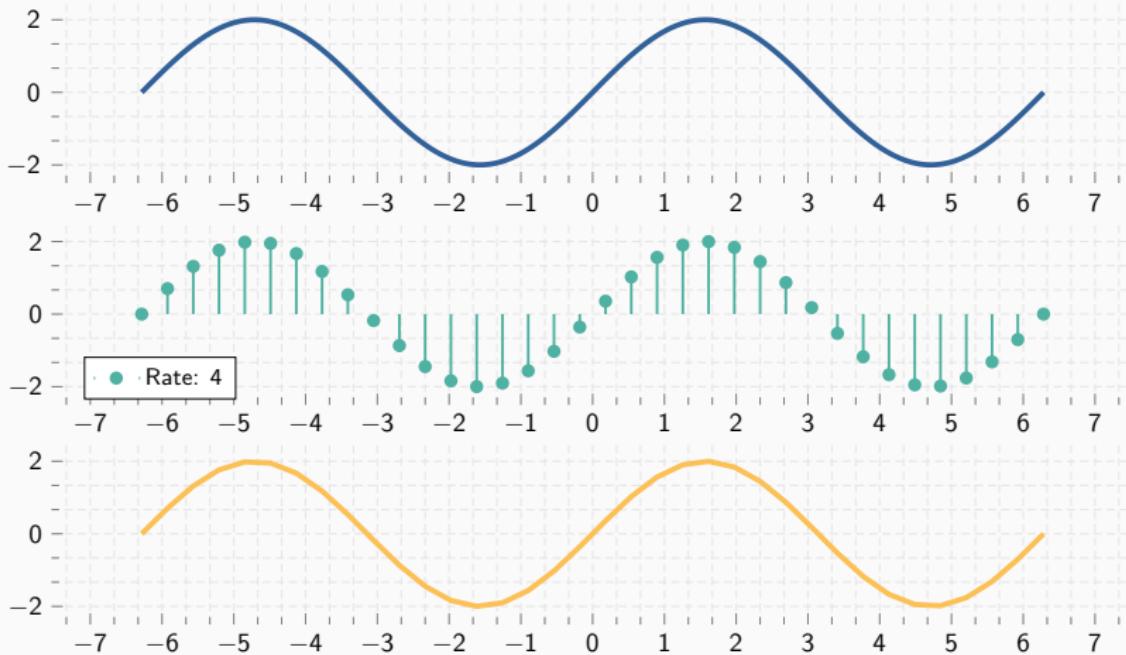


Figure 6: Reconstruction of the signal with 4 times the signal frequency.

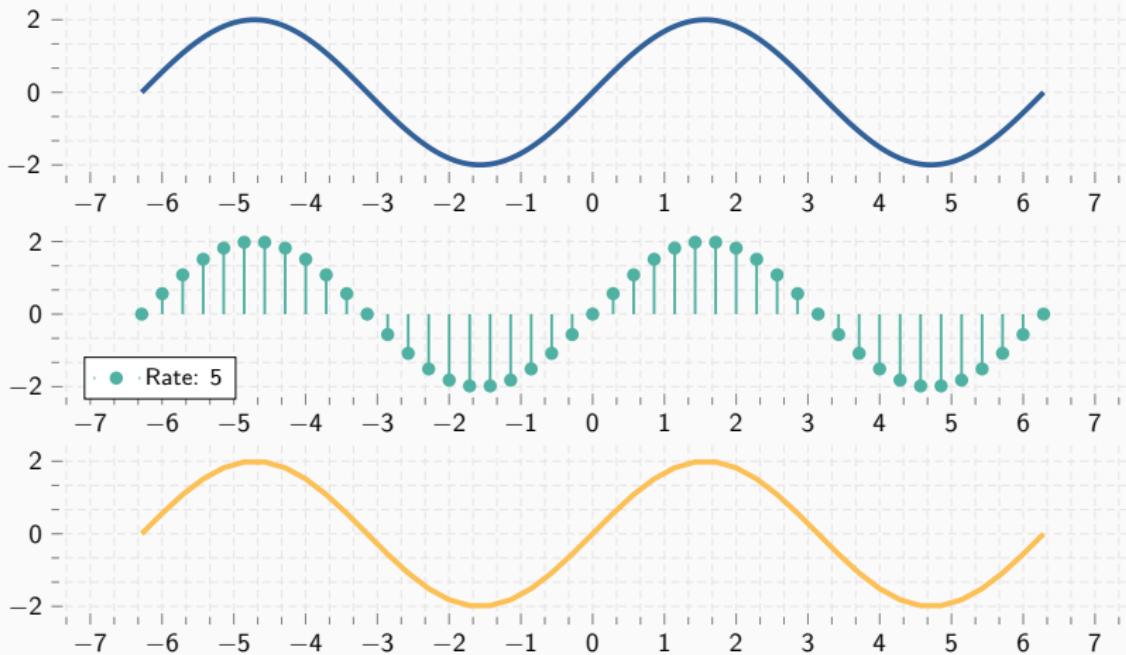
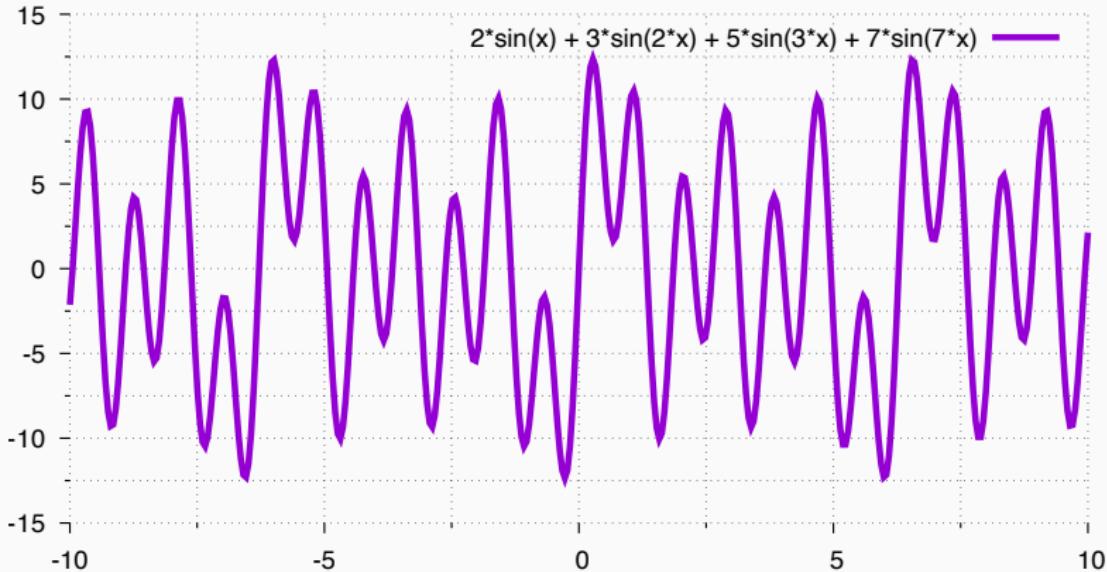


Figure 7: Reconstruction of the signal with 5 times the signal frequency.



## Reconstruction of an Audio Signal

- In practice, doubling frequency is not enough recreate the signal.
- Approaching Nyquist frequency will create a siren like sound, and reaching exact frequency will record a pulse-wave approximation of a sine wave at an amplitude that will vary based on phase.
- Even at 4 times the sampling will only reconstruct a triangle wave and shifting the phase will create tonal distortion.
- For practical cases at least 6 times sampling rate is needed to accurately reconstruct the sine wave



**Figure 8:** A sample signal with containing sample sine waves.



Figure 9: The FFT of the previous complex signal.



- Two (2) key problems arise when conducting spectral analysis of finite, discrete time series (not an infinite time series):

**Aliasing** we only resolve frequencies lower than the Nyquist frequency and frequencies higher than this get aliased to lower frequencies.

**Spectral Leakage** we assume that all waveforms stop and start at  $= 0$  and  $= \pi$ , but in the real world, many of these wave numbers may not complete a full integer number of cycles throughout the domain, causing spectral leakage to other wave numbers.



## Aliasing

- If the initial samples are not sufficiently closely spaced to represent high-frequency components present in the underlying function, then the DFT values will be corrupted by aliasing.
- The solution is either to increase the sampling rate (if possible) or to pre-filter the signal in order to minimise its high-frequency spectral content.



## Leakage

- The **continuous** Fourier transform of a periodic waveform requires the integration to be performed over the interval  $-\infty$  to  $+\infty$  or over an integer number of cycles of the waveform.
- If we attempt to complete the DFT over a non-integer number of cycles of the input signal, then we might expect the transform to be corrupted in some way.

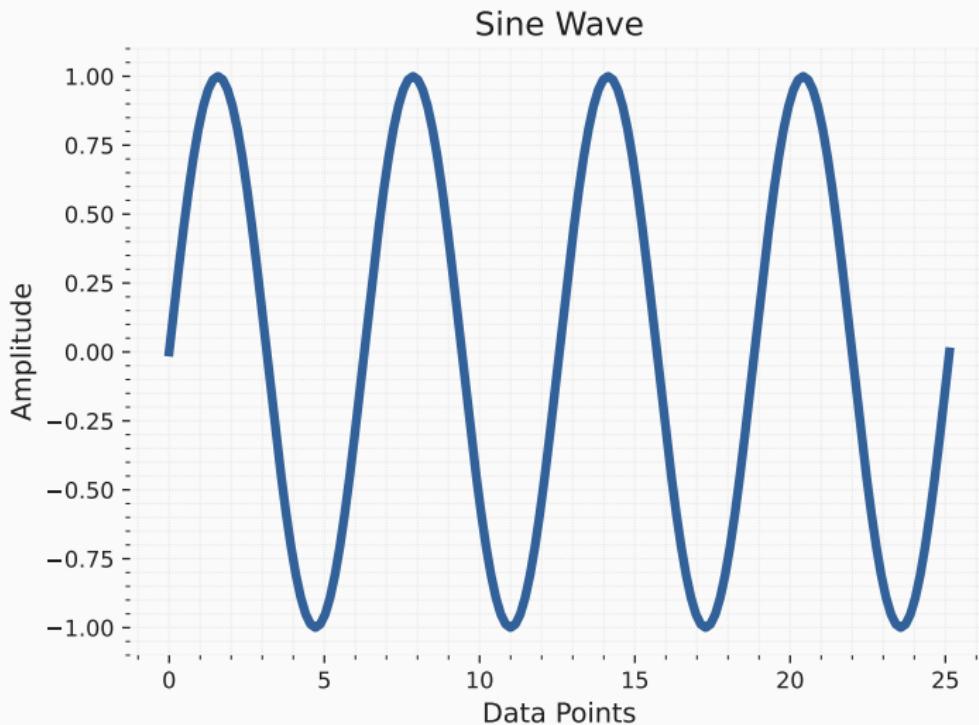
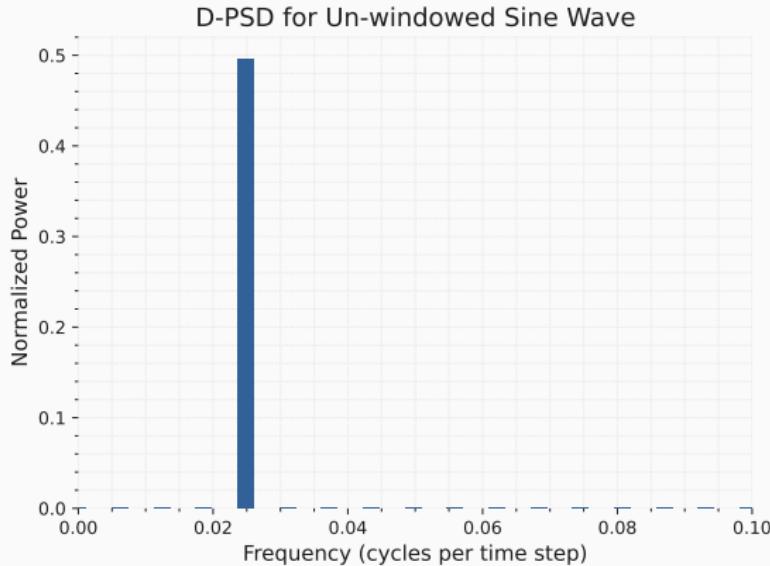


Figure 10: An example of a sine wave with four (4) complete cycles.



- Computing the discrete power spectrum gives:



**Figure 11:** The PSD of an un-windowed sine wave.



- As expected, a single spectral peak corresponding to the frequency of our sine wave.
- Let's see what happens if we apply a window to our sine wave that cuts off the sine wave such that the sine function does not complete an integer number of cycles within the time domain.

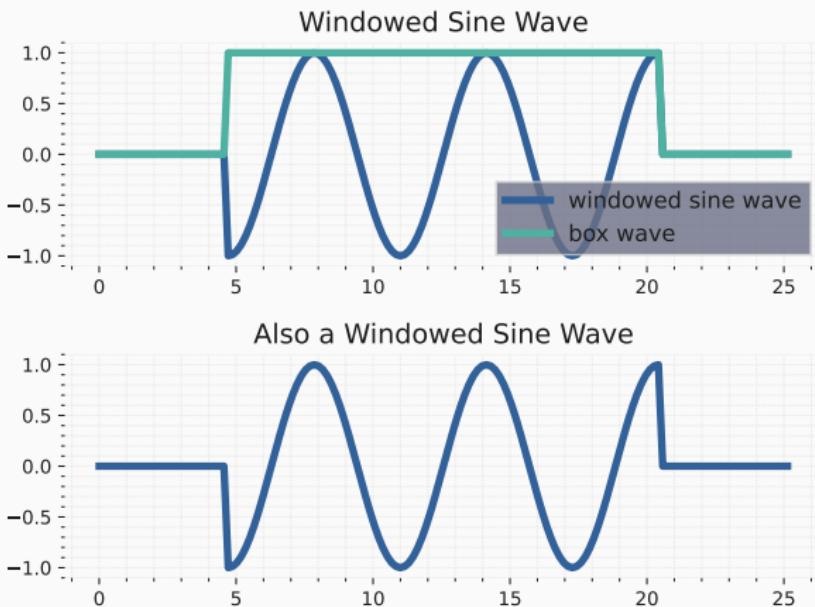


Figure 12: Windowed sine wave.



- To demonstrate what spectral leakage is, we will now compute the discrete power spectrum of the windowed sine wave to see what happens.

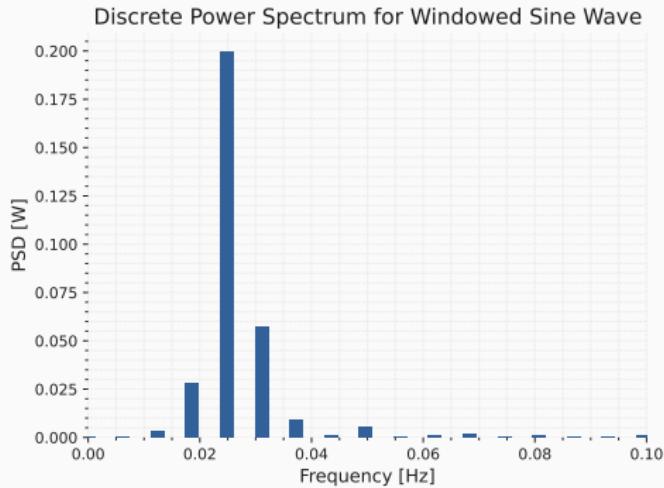


Figure 13: PSD of a windowed sine wave.



## Example

Below is a signal with 1 Hz, Amplitude of 1 and 8 Sampling points.

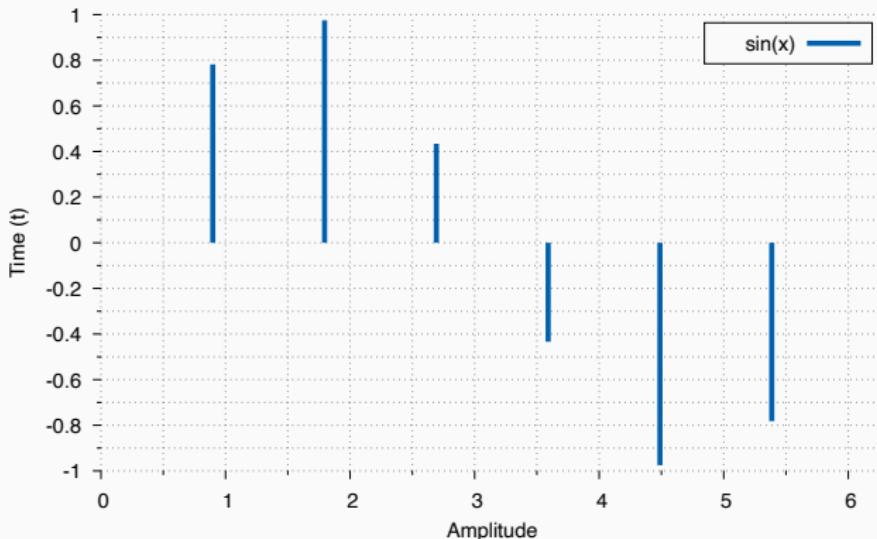


Figure 14: A Sampled Sine wave.



## Solution

As it is a single sine function with 1 Hz, we expect a single value of 1 in the frequency domain (at 1 Hz).

The sampling points will sample the signal and retrieve the following data points as shown in the array below:

$$x_k = [0, 0.707, 1, 0.707, 0, -0.707, -1, -0.707]$$

Once we have these sampling points ( $x_n$ ), we can turn our attention to the DFT formula:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-(j2\pi kn)/N}$$

where  $X_k$  is the  $k^{\text{th}}$  frequency bin.



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_1$ :

$$X_1 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (1) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 2\pi) / N} \\ e^{-(j 4\pi) / N} \\ e^{-(j 6\pi) / N} \\ e^{-(j 8\pi) / N} \\ e^{-(j 10\pi) / N} \\ e^{-(j 12\pi) / N} \\ e^{-(j 14\pi) / N} \end{bmatrix} = 0 - j 4 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_2$ :

$$X_2 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (2) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 4\pi) / N} \\ e^{-(j 8\pi) / N} \\ e^{-(j 12\pi) / N} \\ e^{-(j 16\pi) / N} \\ e^{-(j 20\pi) / N} \\ e^{-(j 24\pi) / N} \\ e^{-(j 28\pi) / N} \end{bmatrix} = 0 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_3$ :

$$X_3 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (3) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 6\pi) / N} \\ e^{-(j 12\pi) / N} \\ e^{-(j 18\pi) / N} \\ e^{-(j 24\pi) / N} \\ e^{-(j 30\pi) / N} \\ e^{-(j 36\pi) / N} \\ e^{-(j 42\pi) / N} \end{bmatrix} = 0 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_4$ :

$$X_4 = \sum_{n=0}^7 x_n \cdot e^{-(j) 2\pi (4) n / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j) 8\pi / N} \\ e^{-(j) 16\pi / N} \\ e^{-(j) 24\pi / N} \\ e^{-(j) 32\pi / N} \\ e^{-(j) 40\pi / N} \\ e^{-(j) 48\pi / N} \\ e^{-(j) 56\pi / N} \end{bmatrix} = 0 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_5$ :

$$X_5 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (5) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 10\pi) / N} \\ e^{-(j 20\pi) / N} \\ e^{-(j 30\pi) / N} \\ e^{-(j 40\pi) / N} \\ e^{-(j 50\pi) / N} \\ e^{-(j 60\pi) / N} \\ e^{-(j 70\pi) / N} \end{bmatrix} = 0 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_6$ :

$$X_6 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (6) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 12\pi) / N} \\ e^{-(j 24\pi) / N} \\ e^{-(j 36\pi) / N} \\ e^{-(j 48\pi) / N} \\ e^{-(j 60\pi) / N} \\ e^{-(j 72\pi) / N} \\ e^{-(j 84\pi) / N} \end{bmatrix} = 0 \quad \blacksquare$$



For the case of  $x_0 = 0$  the exponential part is removed and we are left with  $X_0 = 0$ .

For the cases of  $X_7$ :

$$X_7 = \sum_{n=0}^7 x_n \cdot e^{-(j 2\pi (7) n) / N}$$

$$= x \cdot \begin{bmatrix} 0 \\ e^{-(j 14\pi) / N} \\ e^{-(j 28\pi) / N} \\ e^{-(j 42\pi) / N} \\ e^{-(j 56\pi) / N} \\ e^{-(j 70\pi) / N} \\ e^{-(j 84\pi) / N} \\ e^{-(j 98\pi) / N} \end{bmatrix} = 0 + j 4 \quad \blacksquare$$



- Therefore the values are of the transform are:

$$X_k = [0, 0 - j4, 0, 0, 0, 0, 0, 0 + j4]$$

- We can see only the first and the seventh bins have values other than zero.
- Calculating the magnitudes of the bins, we arrive at 4.

$$|X_k| = [0, 4, 0, 0, 0, 0, 0, 4]$$

- The frequency resolution of the plot is the sampling frequency divided by the number of samples (i.e.,  $f S/N$ ).
- This means we can get values for every integer frequency values.



## Solution

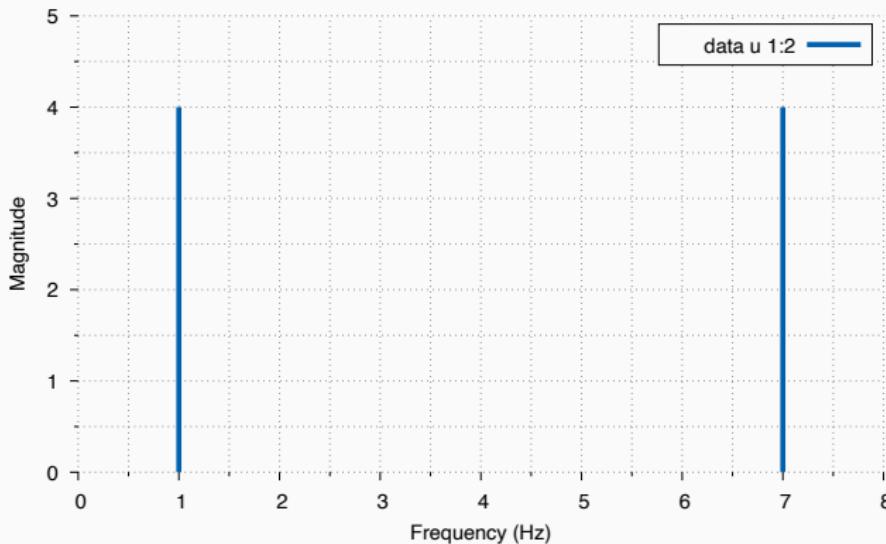


Figure 15: Sampled dataset of the original signal. There is still another step.



## Solution

- We can see we get a value for the first frequency bin (1 Hz) and it makes sense.
- The reason we get a frequency bin is due to the plot being a **two-sided frequency plot** where it shows the energy in both the positive and negative domains.
- The negative frequencies are always complex conjugate to the positive frequencies, so there is no additional information in the negative frequencies.



## Solution

- Therefore, to convert from a two-sided spectrum to a single-sided spectrum, discard the second half of the array and multiply every point except for DC by two.
- The last operation is to divide the magnitudes of the lower frequencies by the number of samples used in deriving these bins

$$|X_k| = [0, 8, 0, 0] \rightarrow |X_k| / N = [0, 1, 0, 0] \blacksquare$$



## Parseval's Theorem

- The sum (or integral) of the square of a function is equal to the sum (or integral) of the square of its transform.
- For continuous signals:

$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\omega)|^2 d\omega = \int_{-\infty}^{\infty} |F(2\pi f)|^2$$

- This signal energy is not to be confused with physical energy.



## Average Value ( $\mu$ )

- Defined as the sample mean of a given region.
- The equation is defined as below (it is also known as **expected value**):

$$\mu = \frac{1}{N} \sum_{i=0}^N x_N$$

## Standard Deviation ( $\sigma$ )

- The standard deviation is a measure of the amount of variation of the values of a variable about its mean ( $\mu$ ):

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$



## Mode

- The mode is the value appears most often in a set of data values.  
i.e., in a data pool of:

$$X = [1, 2, 2, 3, 4, 7, 9]$$

- The mode of is 2 as it is the most frequent value of the data set.
- whereas in:

$$X = [2, 4, 9, 6, 4, 6, 6, 2, 8, 2]$$

the mode is 2, 6 as there are two values with same frequency.



## Median

- The median is the middle value separating the greater and lesser halves of the data set.
- For a ordered data set  $X$  with  $n$  elements,
  - if  $n$  is odd,  $\text{med}(x) = x(n+1)/2$ ,
  - if  $n$  is even,  $\text{med}(x) = x(n/2) + x((n/2)+1)/2$
- i.e., in a ordered data set of:

$$X = [1, 2, 2, 3, 4, 7, 9],$$

- the median is 3 and in:

$$Y = [1, 2, 3, 4, 5, 6, 8, 9],$$

the median is 4.5.



- The signal-to-noise ratio (SNR) can have several definitions depending on the study field.
- Noise is characterised by its standard deviation,  $\sigma$ .
- The characterisation of the signal can differ.
- If the signal is known to lie between two boundaries,  $a_{\min} \leq a \leq a_{\max}$ , then the SNR is defined as:

$$\text{SNR} = 20 \log_{10} \left( \frac{a_{\max} - a_{\min}}{s_n} \right) \text{ dB.}$$

- If the signal is not bounded but has a statistical distribution then two other definitions are known:

$$\text{SNR} = 20 \log_{10} \left( \frac{\mu}{\sigma} \right) \text{ dB.}$$



- In 1948, Claude Shannon published a paper called **A Mathematical Theory of Communication**.
- This paper heralded a transformation in our understanding of information.
- Before Shannon's paper, information had been viewed as a kind of poorly defined ethereal concept.
- But after Shannon's paper, it became apparent that information is a well-defined and, above all, measurable quantity.



- Information theory defines definite, unbreachable limits on precisely how much information can be communicated between any two components of any system, whether this system is man-made or natural.
- The basic laws of information can be summarised as follows.
  1. there is a definite upper limit, the channel capacity, to the amount of information that can be communicated through that channel,
  2. this limit shrinks the amount of noise in the channel increases,
  3. this limit can very nearly be reached by judicious packaging, or encoding, of data.



## Bits are Not Binary Digits

- The word bit is derived from binary digit,
  - but a bit and a binary digit are fundamentally different types of quantities.
- A binary digit is the value of a binary variable, whereas a bit is an amount of information.



- Consider a coin which lands heads up 90% of the time:

$$p(x_h) = 0.9.$$

- When this coin is flipped, we expect it to land heads up ( $x = x_h$ ),
- When it does, we are less surprised than when it lands tails ( $x = x_t$ ).

The more improbable a particular outcome is, the more surprised we are to observe it.

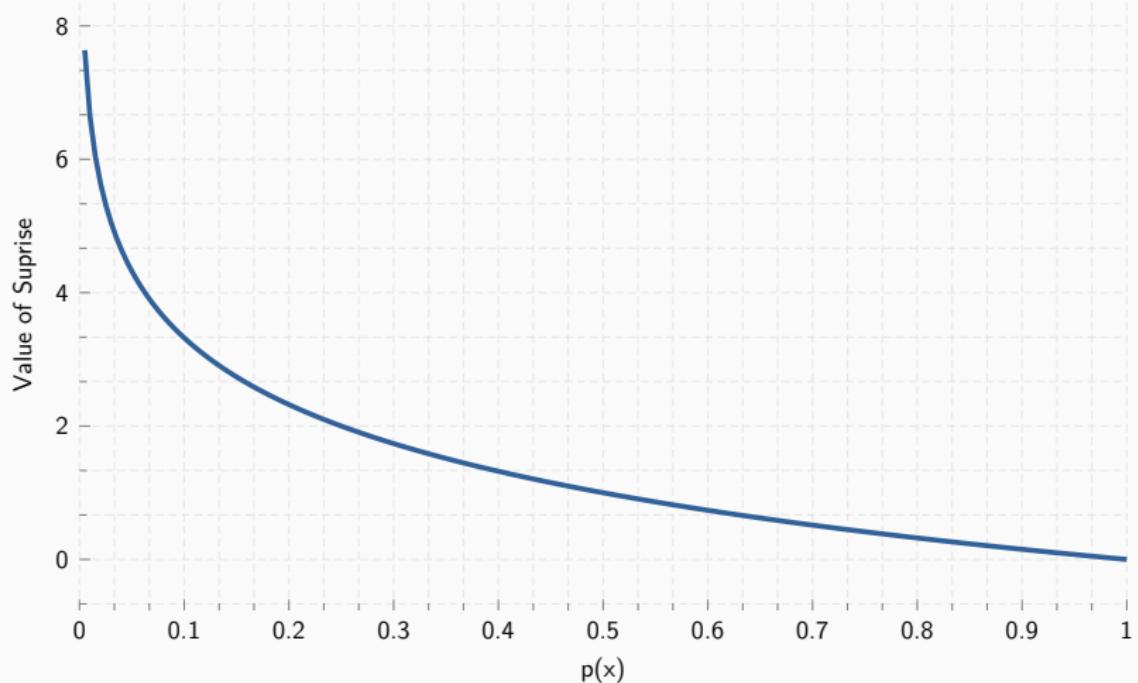
- If we use  $\log_2$  then the Shannon information or surprisal of each outcome is measured in bits.

$$\text{Shannon Information} = \log_2 \frac{1}{p(x_h)}$$



## Entropy is Average Shannon Information

- We can represent the outcome of a coin flip as the random variable  $x$ , such that a head is  $x = x_h$  and a tail is  $x = x_t$ .
- In practice, we are not usually interested in the surprise of a particular value of a random variable, but we are interested in how much surprise, on average, is associated with the entire set of possible values.
- The average surprise of a variable  $x$  is defined by its probability distribution  $p(x)$ , and is called the entropy of  $p(x)$ , represented as  $H(x)$ .



**Figure 16:** The quantifiable surprise with respect to increasing probability.



## Entropy of a Fair Coin

- If a coin is fair or unbiased then:

$$p_{x_h} = p_{x_t} = 0.5$$

- the Shannon information gained when a head or a tail is observed is:

$$\log 1/0.5 = 1 \text{ bit}$$

- The average Shannon information gained after each coin flip is also 1 bit.
- Because entropy is defined as average Shannon information, the entropy of a fair coin is  $H(x) = 1$  bit.



## Entropy of an Unfair Coin

- If a coin is biased such that the probability of a head is  $p(x_h) = 0.9$ .
  - it is easy to predict the result of each coin flip (i.e. with 90% accuracy if we predict a head for each flip)
- If the outcome is a head then the amount of Shannon information gained is  $\log(1/0.9) = 0.15$  bits.
- But if the outcome is a tail then the amount of Shannon information gained is  $\log(1/0.1) = 3.32$  bits.
- Notice that more information is associated with the more surprising outcome.



## Entropy of an Unfair Coin

- Given that the proportion of flips that yield a head is  $p(x_h)$ , and that the proportion of flips that yield a tail is  $p(x_t)$  (where  $p(x_h) + p(x_t) = 1$ ), the average surprise is

$$H(x) = p(x_h) \log \frac{1}{p(x_h)} + p(x_t) \log \frac{1}{p(x_t)},$$

- Which comes to 0.469bits.
- If we define a tail as  $x_1 = x_t$  and a head as  $x_2 = x_h$  then the above equation is written as:

$$H(x) = \sum_{i=1}^2 p(x_i) \log \frac{1}{p(x_i)} \text{ bits.}$$



- More generally, a random variable  $x$  with a probability distribution  $p(x) = p(x_1), \dots, p(x_m)$  has an entropy of

$$H(x) = \sum_{i=1}^m p(x_i) \log \frac{1}{p(x_i)} \text{ bits.}$$



- Entropy is a measure of **uncertainty**.
- When our uncertainty is reduced, we gain information,
  - so information and entropy are two sides of the same coin.
- However, information has a rather subtle interpretation, which can easily lead to confusion.
- Average information shares the same definition as entropy,
  - but whether we call a given quantity information or entropy depends on whether it is being **given to us or taken away**.
- For example, if a variable has high entropy the initial uncertainty of the variable is large and is, by definition, exactly equal to its entropy.
- If we are told the variable value, on average, we have been given information equal to the uncertainty (entropy) we had about its value.
- Thus, receiving an amount of information is equivalent to having exactly the same amount of entropy (uncertainty) taken away.

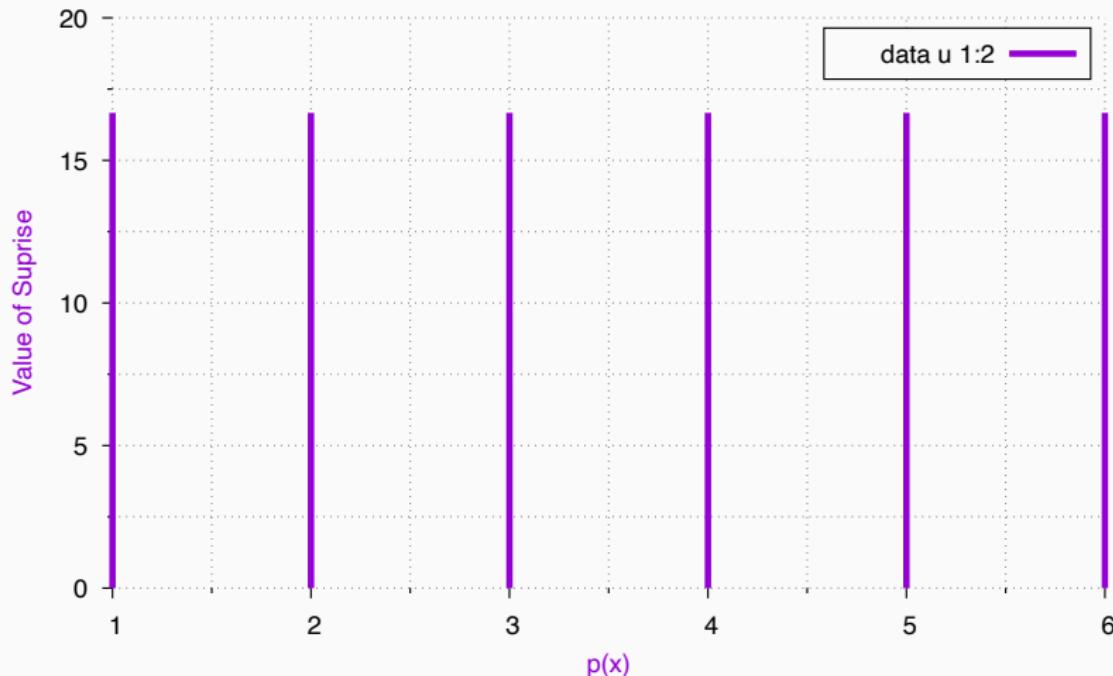


Figure 17: The probability distribution of 1 dice(s).

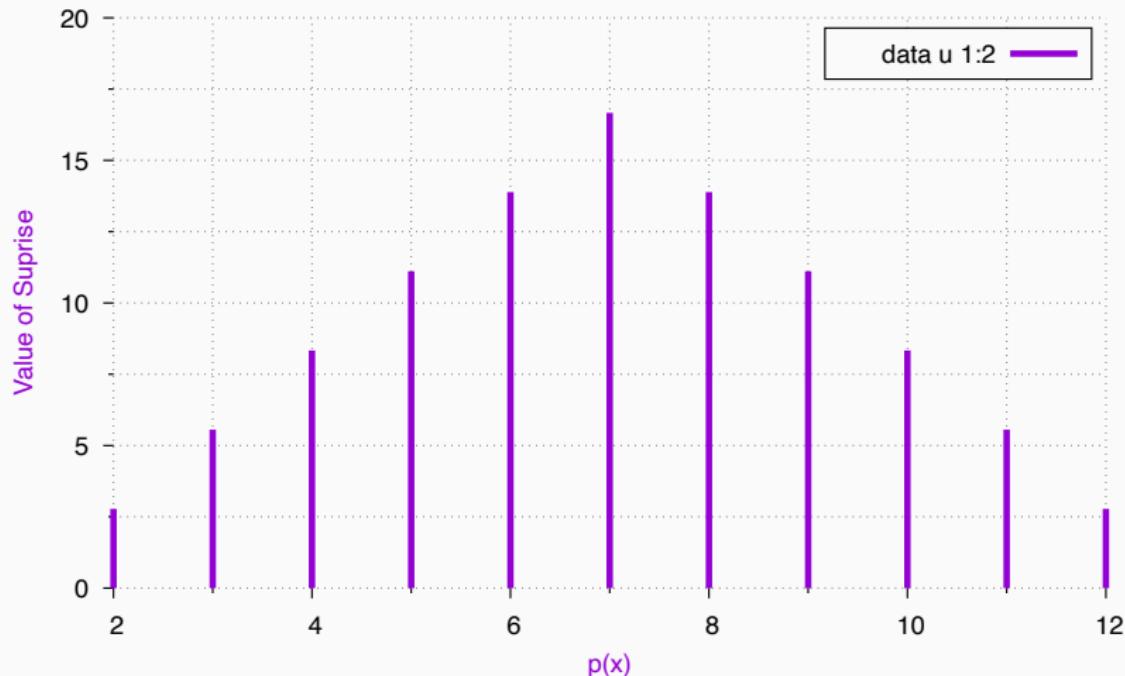


Figure 18: The probability distribution of 2 dice(s).

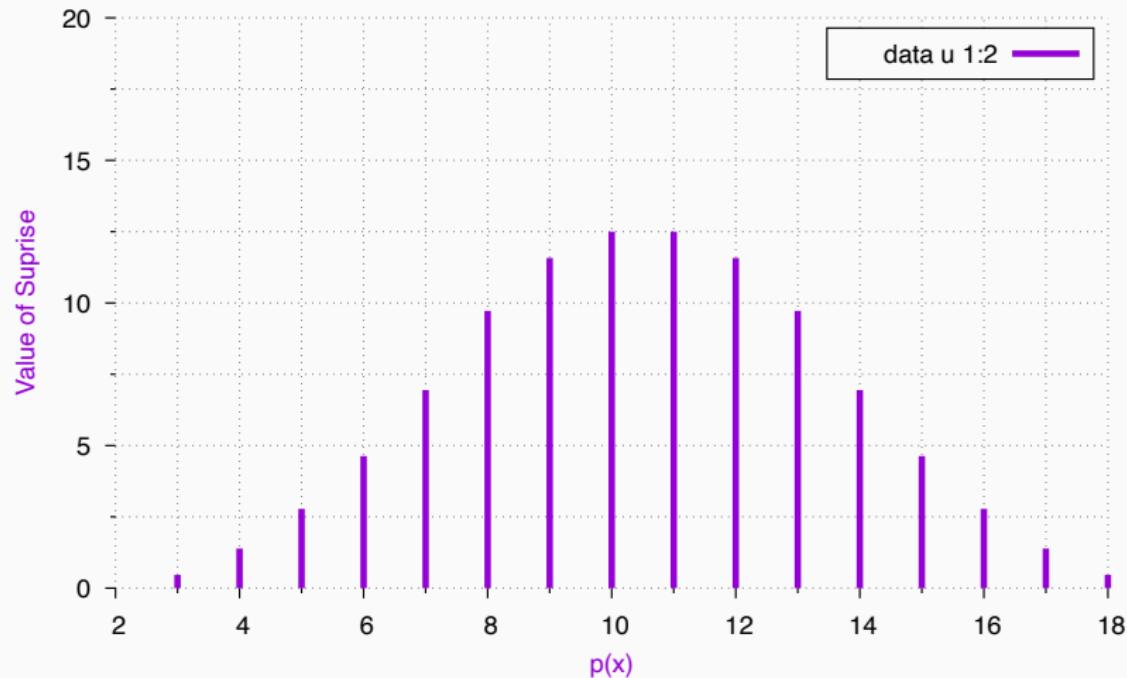


Figure 19: The probability distribution of 3 dice(s).

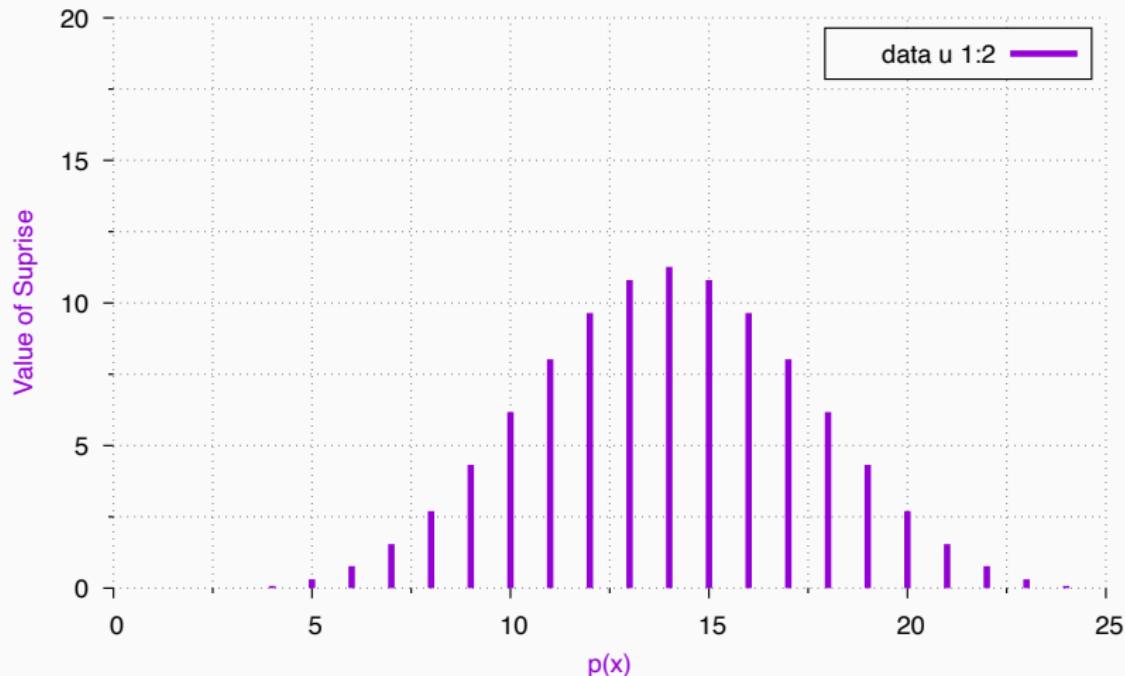


Figure 20: The probability distribution of 4 dice(s).



- Throwing a pair of 6-sided dice produces an outcome in the form of an ordered pair of numbers.
  - There are a total of 36 equiprobable outcomes,
- If we define an outcome value as the sum of this pair of numbers then there are  $m = 11$  possible outcome values:

$$A_x = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}.$$

Dividing the frequency of each outcome value by 36 yields the probability  $p$  of each outcome value.



- We can use these 11 probabilities to find the entropy.

$$\begin{aligned}H(x) &= p(x_1) \log \frac{1}{p(x_1)} + p(x_2) \log \frac{1}{p(x_2)} + \cdots + p(x_{11}) \log \frac{1}{p(x_{11})} \\&= 3.27 \text{ bits.}\end{aligned}$$

# Perception

---



<b>Learning Outcomes</b>	Wide Gamut RGB
<b>Introduction</b>	Prophoto RGB
Human Vision	Adobe RGB
Brightness Sensitivity	CIE Chromaticity Coordinates
Stimulus Sensitivity	Chromaticity
Colour Sensitivity	
<b>Colour Standards</b>	<b>Colour Models</b>
sRGB	CYMK Colour Model
	HSL and HLV Colour Model
	YCbCr



## Learning Outcomes

- (LO1) A Look into Human Vision,
- (LO2) Definition of Colour and Standardisation,
- (LO3) How Vision is Perceived,
- (LO4) Types of Colour-spaces.





- Many image processing applications are intended to produce images to be viewed by **humans**.
  - This is in contrast to automated industrial robots.
- It is important to understand the characteristics and limitations of the human visual system [12].
- At the outset it is important to realise:
  1. The human visual system is **not well understood** [12].
    - It is not easy to study the human visual system without directly measuring it.
  2. **No objective measure exists** for judging the quality of an image that corresponds to human assessment of image quality,
    - A colour you find fitting might be repugnant to someone.
  3. A typical human observer **does not exist**.

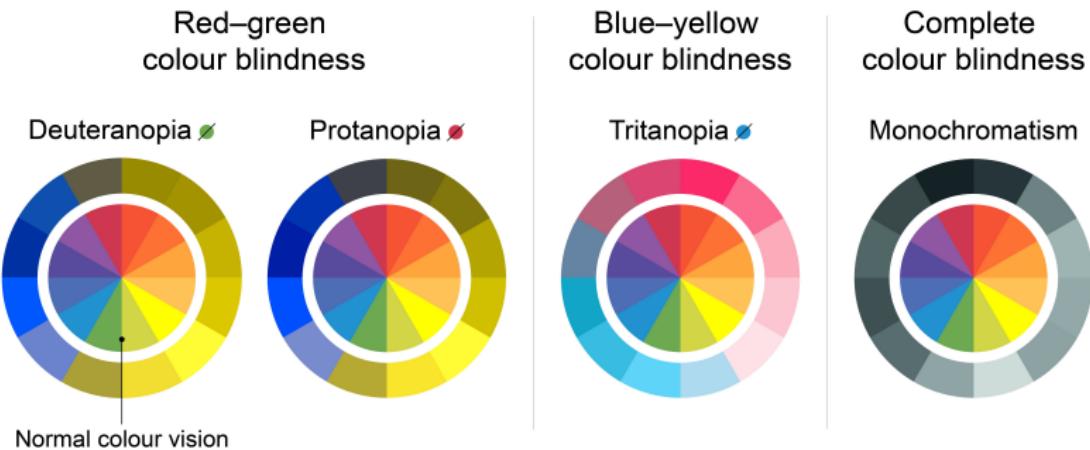
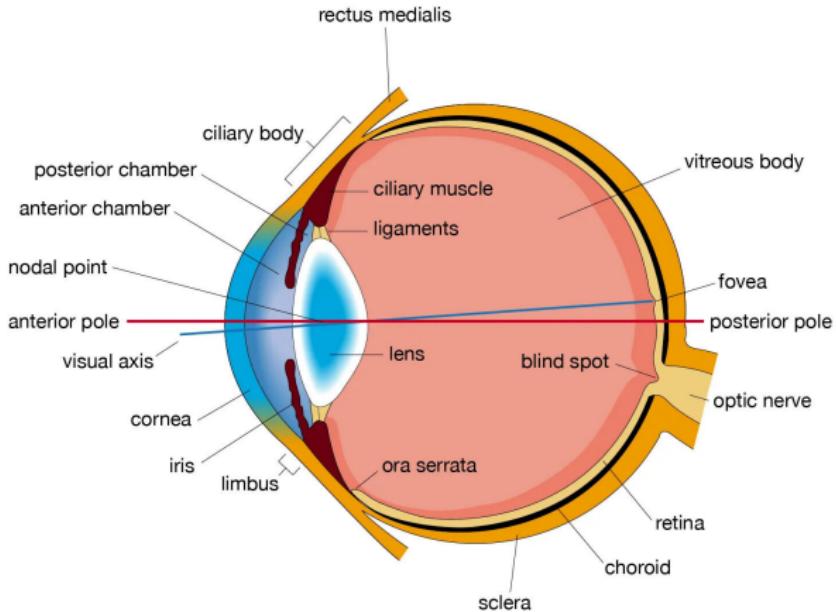


Figure 21: Colour-wheel of different kinds of blindness [29].



- Statistically 1 in 8 men and 1 in 200 women have a form of colour blindness [6].
- Men have one X chromosome and one Y chromosome, while women have two X chromosomes [5].
- To experience color blindness, the genetic mutation for colorblindness must be present on the X chromosome, but for women, this means it must be present on both X chromosomes.
- Men only need a mutation to be present on their singular X chromosome, making it much easier for them to inherit color blindness.



© 2013 Encyclopædia Britannica, Inc.

**Figure 22:** Horizontal section of the eye.



## Trichromacy

- Normal colour vision uses all three (3) types of cone cells which function correctly.
- Another term for normal colour vision is **trichromacy**.
- People with normal colour vision are known as trichromats.

## Anomalous Trichromacy

- People with **faulty** trichromatic vision will be colour blind to some extent and are known as anomalous trichromats [6].
- In people with this condition all of their three cone cell types are used to perceive light wavelengths but one type of cone cell perceives light slightly out of alignment.
- There are three (3) different types of effect produced depending upon which cone cell type is **faulty** and there are also different severities.



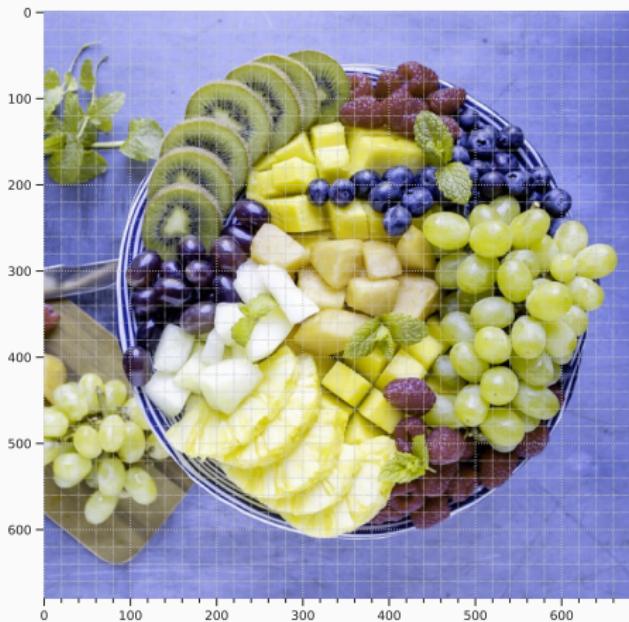
- The different anomalous condition types are [33]:
  - protanomaly** reduced sensitivity to red light,
  - deuteranomaly** reduced sensitivity to green light (most common),
  - tritanomaly** reduced sensitivity to blue light (most uncommon).

## Achromatopsia

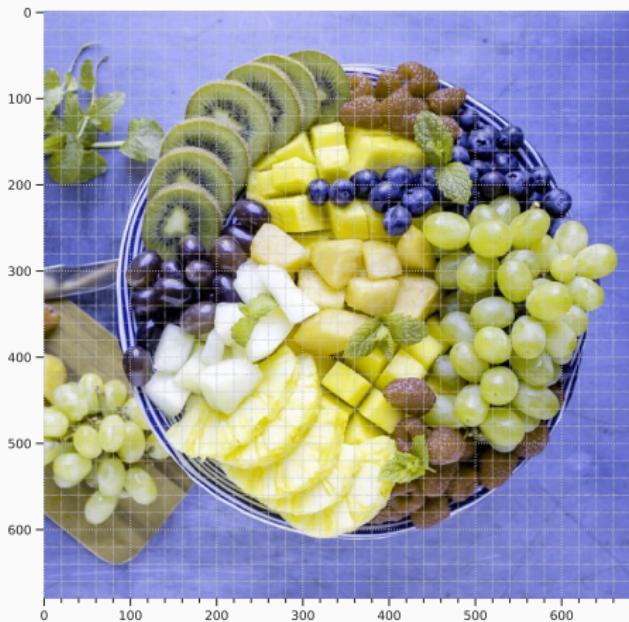
- Can see no colour at all and their world consists of different shades of grey ranging from black to white, rather like seeing the world on an old black and white television set [34].
- Achromatopsia is a specific eye condition in which people see in grey-scale.
- In rare cases, partial Achromatopsia can happen which is a **reduced** sensitivity to all three (3) cones [34].



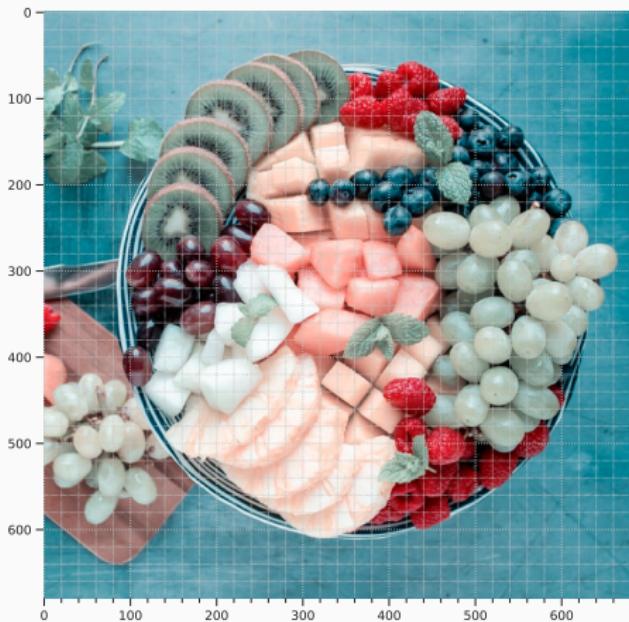
**Figure 23:** Image viewed by someone who has 3 cones.



**Figure 24:** Image viewed by someone who has protanomaly.



**Figure 25:** Image viewed by someone who has deuteranomaly.



**Figure 26:** Image viewed by someone who has tritanomaly.

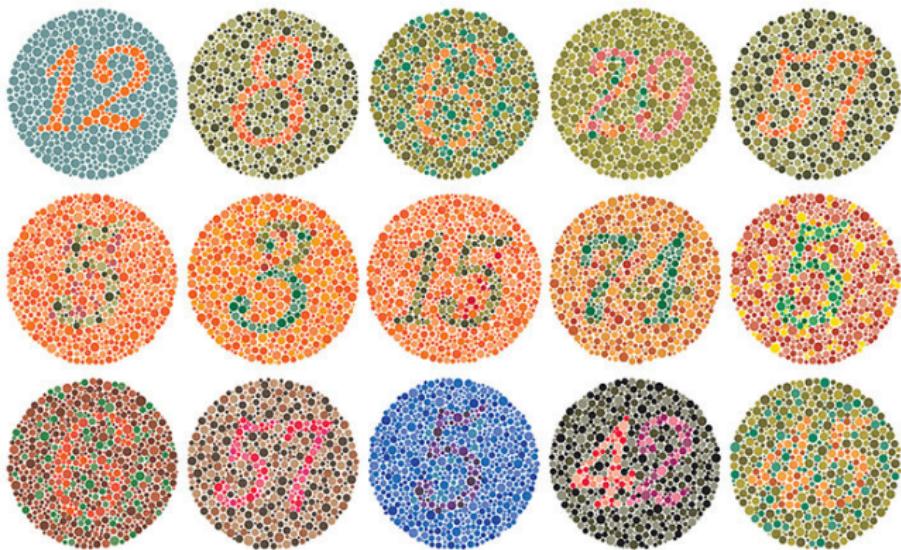


Figure 27: A colour blindness test issued to generally test before taking the driving license [4].



- There are ways to describe the sensitivity of human vision.
- Assume a homogeneous region in an image has an intensity as a function of wavelength (colour) given by  $I(\lambda)$  and assume that  $I(\lambda) = I_0$ , a constant.

## Wavelength Sensitivity

- The sensitivity of the human eye to light of a certain intensity varies strongly over wavelengths between 380 nm and 800 nm [27].
- Under daylight conditions, human eye is most sensitive at a wavelength of 555 nm, resulting in the fact that green light at produces the impression of highest “brightness” when compared to light at other wavelengths [27].



- The perceived intensity as a function of  $\lambda$ , the spectral sensitivity, for the **typical observer** is shown below.

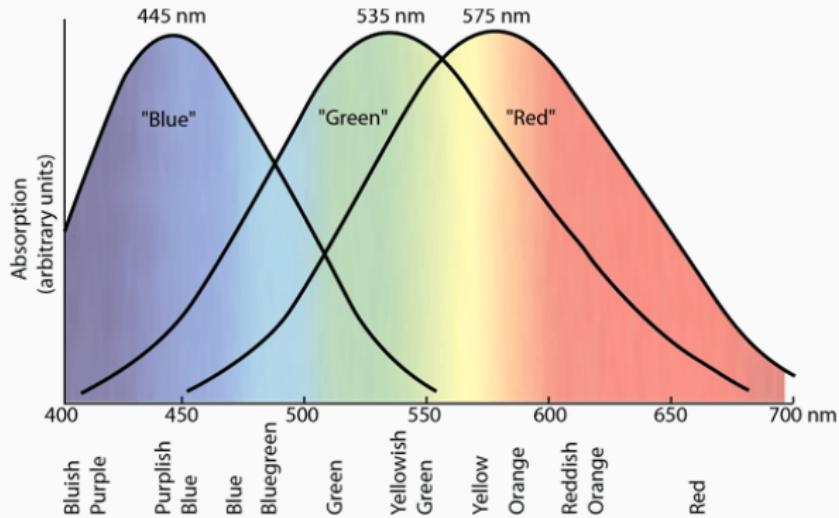
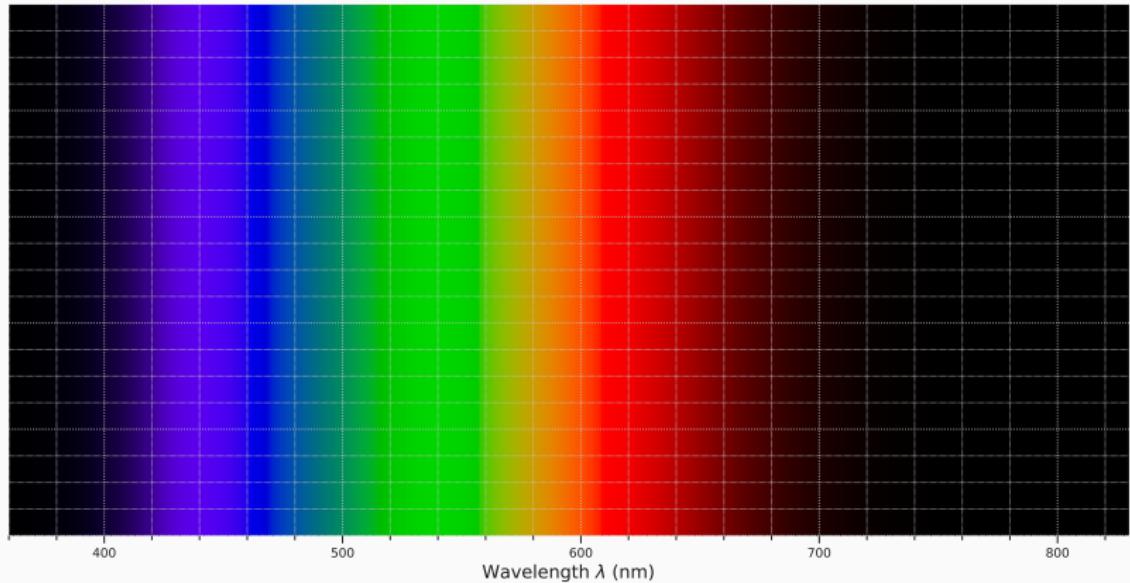


Figure 28: The colour sensitivity of the human eye [30].



The Visible Spectrum - CIE 1931 2° Standard Observer



**Figure 29:** The visible colour spectrum visible with the human eye.



- If the constant intensity (i.e., brightness)  $I_0$  is allowed to vary, then, to a good approximation, the visual response,  $R$ , is proportional to the logarithm of the intensity.
- This is known as the **Weber-Fechner law** [13].

The law relates to human perception, more specifically the relation between the actual change in a physical stimulus and the perceived change.

$$R = \log(I_0)$$

- This means, equal perceived steps in brightness,  $\Delta R = k$ , require the physical brightness (i.e., the stimulus) to increase exponentially.

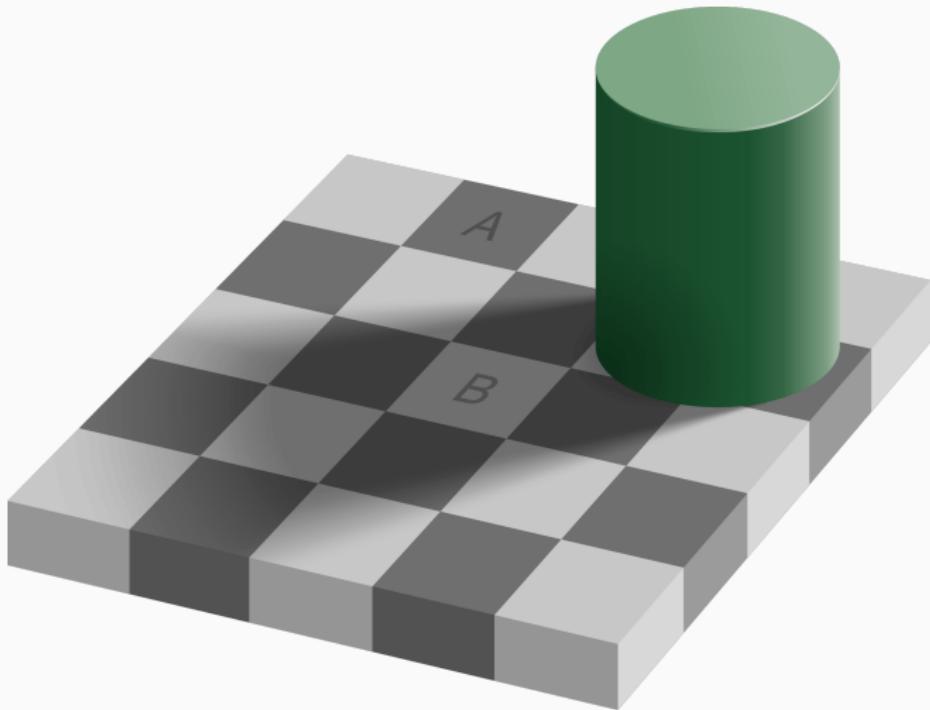
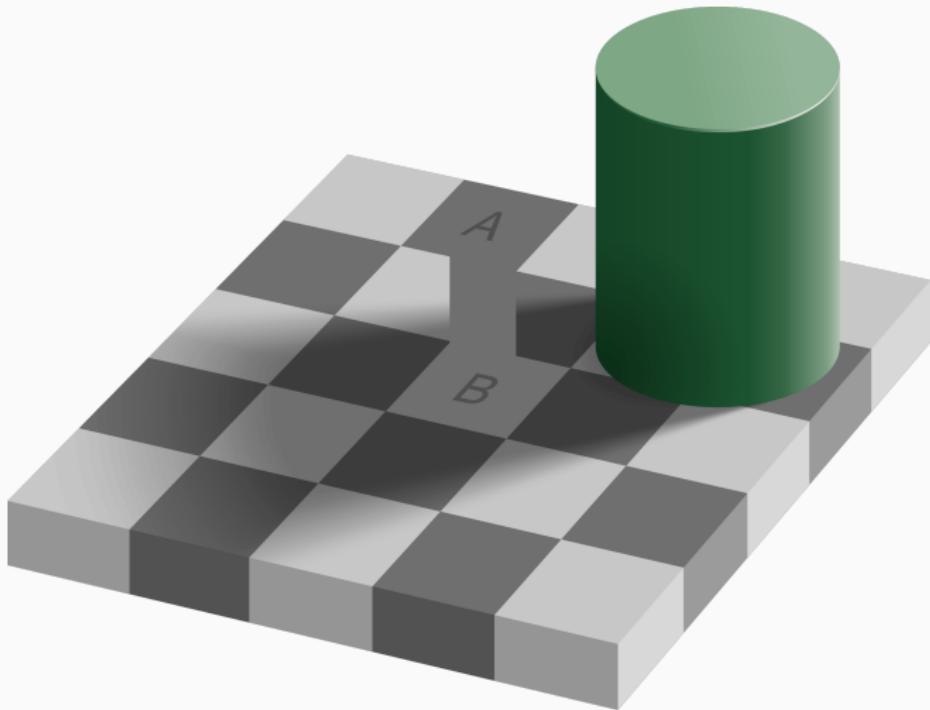


Figure 30: The checker shadow illusion [1].



**Figure 31:** A region of the same shade has been drawn connecting A and B.



Figure 32: A simple picture of a dress divided the internet.



**Figure 33:** A simple picture of a dress divided the internet [7].



- Human colour perception is complex,
  - therefore we can only present a brief introduction.
- **Standard Observer:** Based on psychophysical measurements, standard curves have been adopted by the CIE<sup>1</sup> as sensitivity curves for the typical observer for three pigments  $\bar{x}(\lambda)$ ,  $\bar{y}(\lambda)$ , and  $\bar{z}(\lambda)$ .

These are not the actual pigment absorption characteristics found in the standard human retina but rather sensitivity curves derived from actual data.

This standard is used by companies to produce monitors and software that are compatible with each other.

<sup>1</sup>Commission Internationale de l'Eclairage

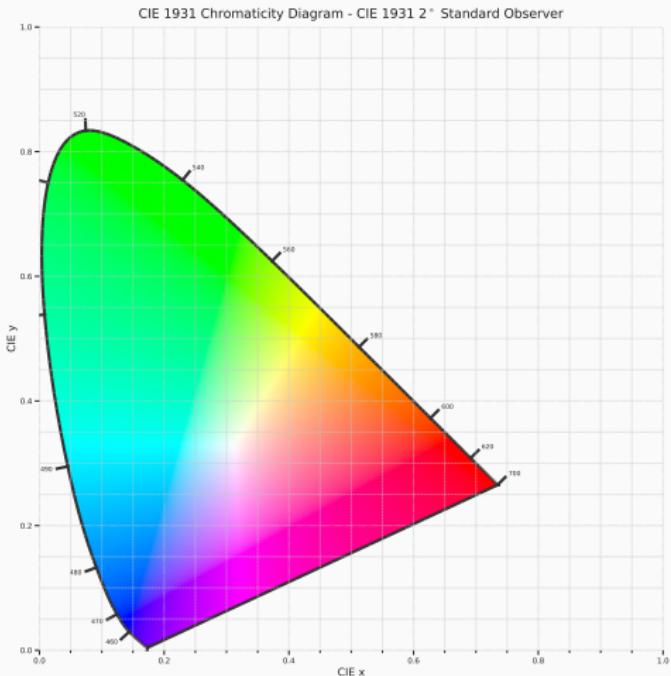


Figure 34: The colour gamut visible to the human eye, standardised by the CIE.



- sRGB is a standard RGB colour-space HP and Microsoft created cooperatively in 1996 to use on monitors, printers, and the World Wide Web [50].
- It was subsequently standardised by IEC as IEC 61966-2-1:1999 [41].
- sRGB is the current defined standard colour-space for the web, and it is usually the assumed colour-space for images that are neither tagged for a colour-space nor have an embedded color profile.
- It codifies the display specifications for the computer monitors in use at the time, which greatly aided its acceptance.
- sRGB uses the same colour primaries and white point as ITU-R BT.709 standard for HDTV, designed to match typical home and office viewing conditions.

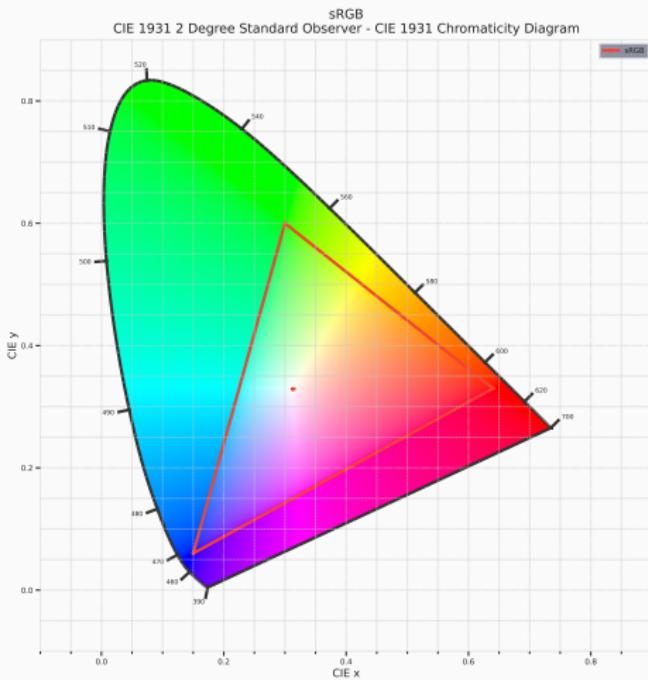


Figure 35: The sRGB colour-space superimposed to the CIE colour-gamut.



- Due to the standardisation of sRGB on the digital-space, and on printers, many low- to medium-end consumer digital cameras and scanners use sRGB as the **default** working colour-space [53].
- However, consumer-level CCDs<sup>1</sup> are typically **uncalibrated**, meaning that even though the image is being labeled as sRGB, one can not conclude that the image is color-accurate sRGB.

---

<sup>1</sup>Charge-Coupled Device



- The wide-gamut RGB colour-space (Adobe Wide Gamut RGB) is developed by Adobe, which offers a large gamut by using pure spectral primary colours [47].
- It is able to store a wider range of colour than sRGB or Adobe RGB.

For comparison, the wide-gamut RGB colour-space encompasses 77.6% of the visible colours, while Adobe RGB covers 52.1% and sRGB only 35.9% [56].



Figure 36: The wide gamut RGB colour-space superimposed to the CIE colour-gamut.



- The ProPhoto RGB colour space, a.k.a. ROMM RGB (Reference Output Medium Metric), is an output referred RGB color space developed by Kodak [49].
- Offers an especially **large gamut** designed for use with photographic output in mind.
- The gamut encompasses over 90% of possible color space, and 100% of likely occurring real-world surface colours making ProPhoto even larger than the Wide-gamut RGB color space [61].
- The ProPhoto RGB primaries were also chosen in order to minimise hue rotations associated with non-linear tone scale operations.

One of the downsides is that approximately 13% of the visible colours are imaginary colors that do not exist and are **impossible colour**.

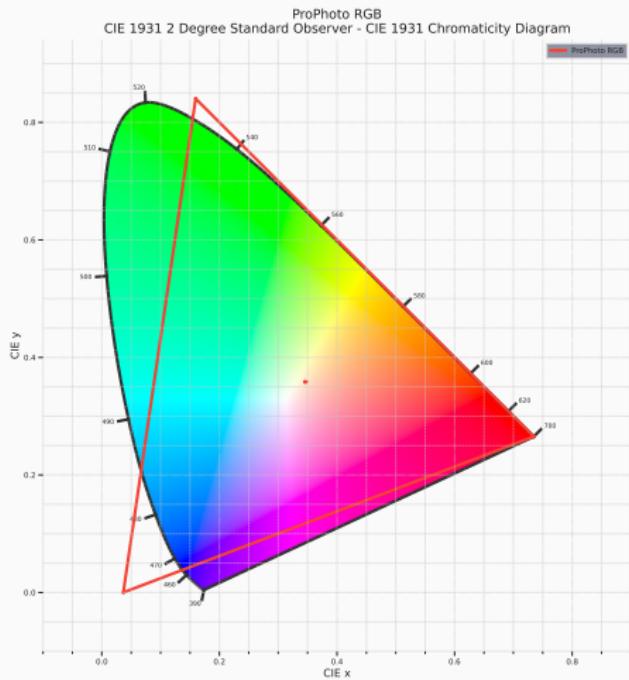


Figure 37: The ProPhoto colour-space superimposed to the CIE colour-gamut.



- The **Adobe RGB (1998)** or **opRGB** is a color space developed by Adobe Inc. in 1998.
- It was designed to encompass most of the colors achievable on CMYK color printers, but by using RGB primary colors on a device such as a computer display.
- The Adobe RGB (1998) color space encompasses roughly 30% of the visible colors specified by the CIE – improving upon the gamut of the sRGB color space, primarily in cyan-green hues.
- It was subsequently standardised by the IEC as IEC 61966-2-5:1999 with a name opRGB (optional RGB color space) and is used in HDMI [42].



- For an arbitrary homogeneous region in an image that has an intensity as a function of wavelength (colour) given by  $I(\lambda)$ , the three responses are called the **tristimulus values**:

$$A = \int_{-\infty}^{+\infty} I(\lambda) \bar{a}(\lambda) d\lambda \quad \text{where} \quad A = \{X, Y, Z\}, \quad a = \{x, y, z\}.$$

- The **chromaticity coordinates** which describe the perceived colour information are defined as:

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{y}{X + Y + Z}, \quad z = 1 - (x + y).$$

- The tristimulus values are linear in  $I(\lambda)$  and thus the absolute intensity information has been lost in the calculation of the chromaticity coordinates  $\{x, y\}$ .
- All colour distributions,  $I(\lambda)$ , that appear to an observer as having the same colour will have the same chromaticity coordinates.



- The formulas for converting from the tristimulus values ( $X, Y, Z$ ) to **RGB** colours ( $R, G, B$ ) and back are given by:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.19107 & -0.5326 & -0.2883 \\ -0.9843 & 1.9984 & -0.0283 \\ 0.0583 & -0.1185 & 0.8986 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

and:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.6067 & 0.1736 & 0.2001 \\ 0.2988 & 0.5868 & 0.1143 \\ 0.0000 & 0.0661 & 1.1149 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

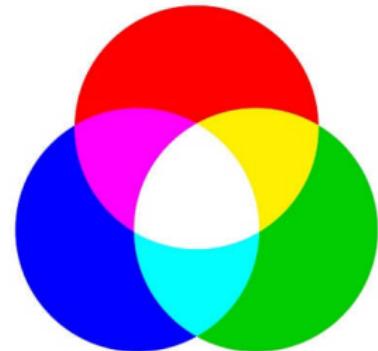
- Before we end our look on to these colour spaces, lets have a look at two (2) more standards.



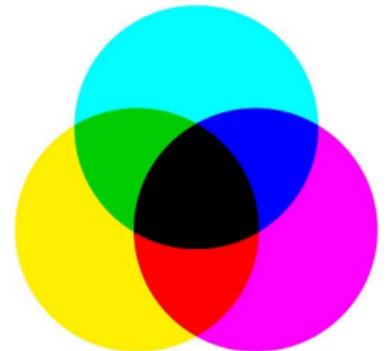
- The **CMYK** model is a **subtractive model** used in colour printing, and describing the printing process itself.
- The abbreviation CMYK refers to the four inks used:  
**cyan**, **magenta**, **yellow**, and **key** (black).
- Works by partially or entirely masking colours on a lighter, usually white, background.
- The ink limits the **reflected light**.
- Such a model is called subtractive because inks **subtract** the colours red, green and blue from white light.
- White light minus red leaves cyan, white light minus green leaves magenta, and white light minus blue leaves yellow.



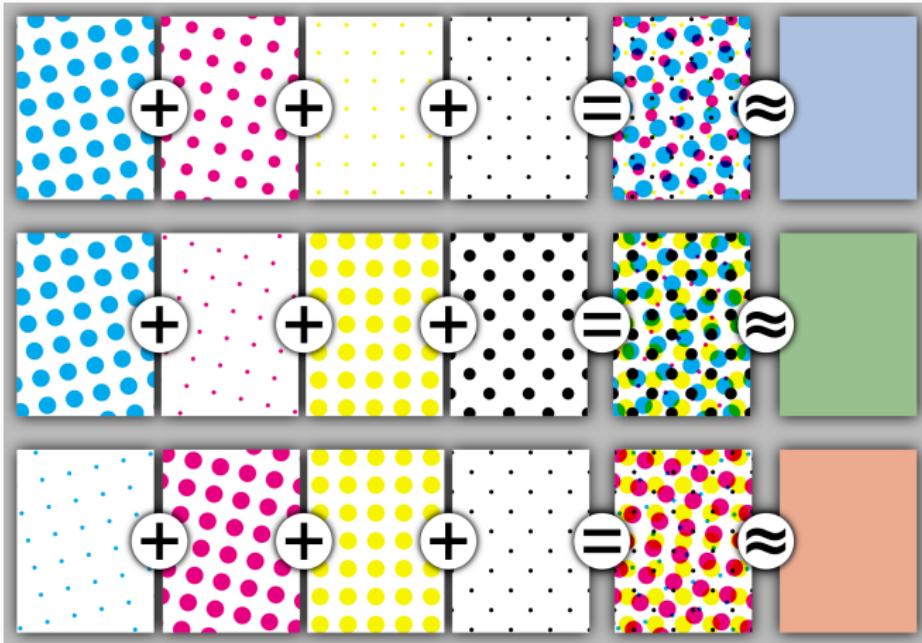
**RGB**



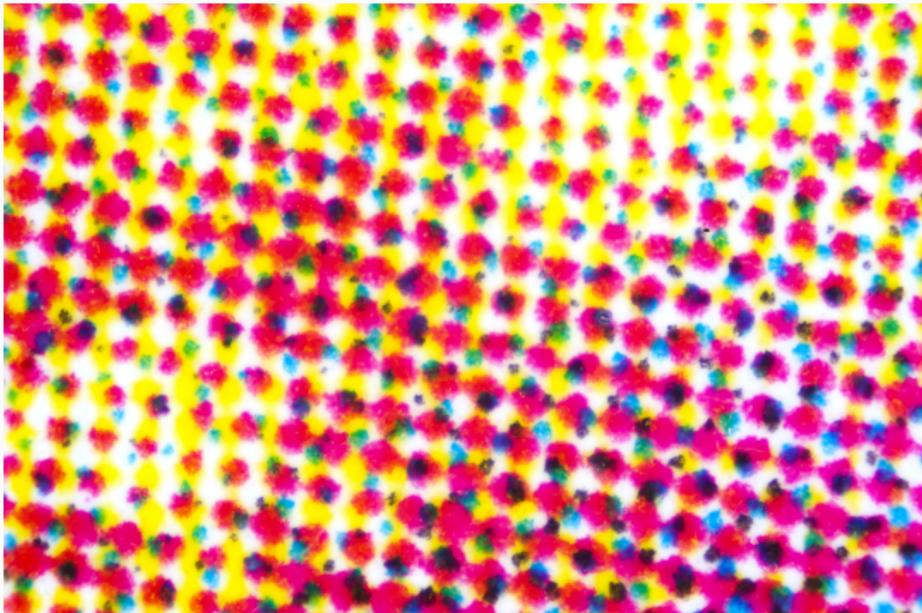
**CMYK**



**Figure 38:** The differences between RGB and CMYK colours [54].



**Figure 39:** Three examples of color halftoning with CMYK separations, as well as the combined halftone pattern and how the human eye would observe the combined halftone pattern from a sufficient distance [59].



**Figure 40:** A printer creates any colour by combining dots in particular places relative to the other dots.



- Two most common cylindrical-coordinate representations of points in an RGB color model.
- The two representations rearrange the geometry of RGB in an attempt to be more intuitive and perceptually relevant than the cartesian (cube) representation.
- Developed in the 1970s for computer graphics applications, are used in color pickers, in image editing software, and less commonly in image analysis and computer vision.

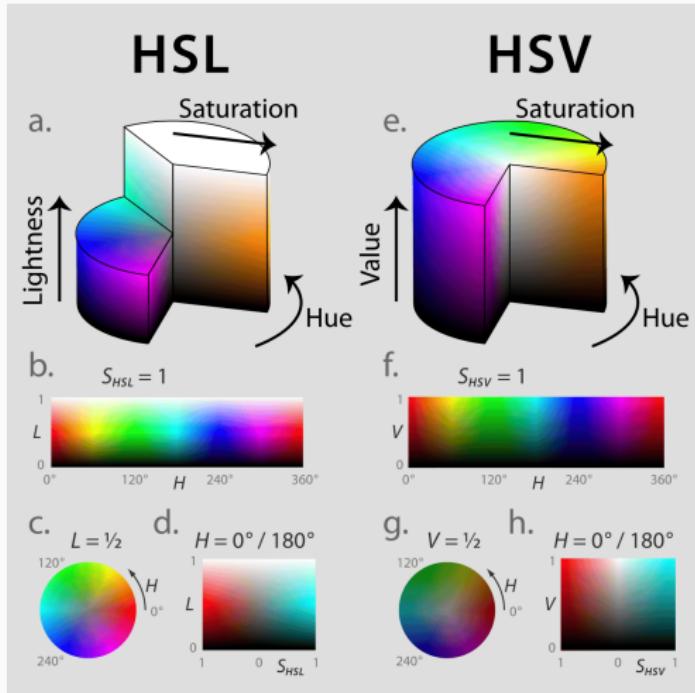


Figure 41: HSL and HSV models.



- $YC_bC_r$  is a family of colour spaces used as a part of the color image pipeline in video and digital photography systems.
- $Y$  is the luma (i.e., brightness) component and  $CB$  and  $CR$  are the blue-difference and red-difference chroma components.
- $Y$  (with prime) is distinguished from  $Y$ , which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.



- CRT<sup>1</sup> uses RGB signals, but they are not the best solution for storing information as they have a lot of redundancy.
- $YC_bC_r$  is a practical approximation, where the primary colours corresponding roughly to red, green and blue are processed into **perceptually meaningful** information.
- $Y'C_bC_r$  is used to separate out a luma signal ( $Y'$ ) that can be stored with high resolution or transmitted at high bandwidth, and two chroma components (CB and CR) that can be bandwidth-reduced, subsampled, compressed, or otherwise treated separately for improved system efficiency.

One practical example would be decreasing the bandwidth or resolution allocated to "color" compared to "black and white", since humans are more sensitive to the black-and-white information (see image example to the right). This is called chroma subsampling.

<sup>1</sup>Cathode Ray Tube

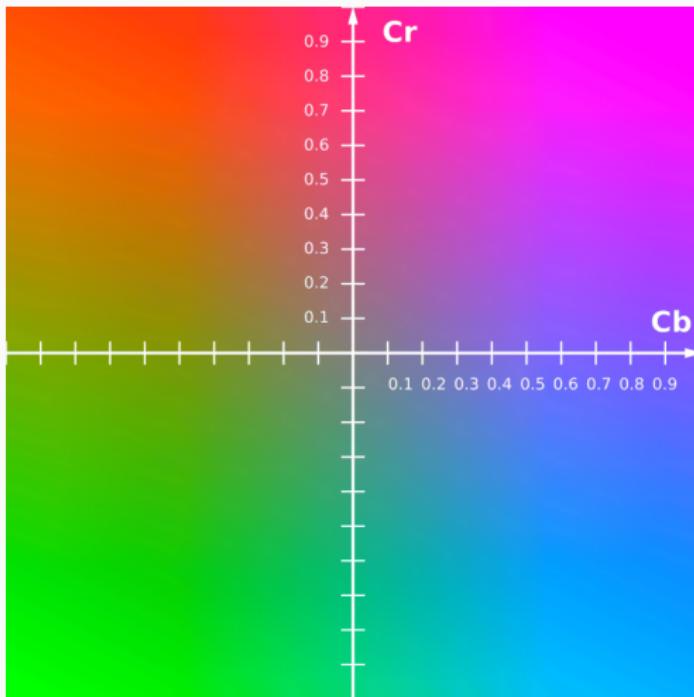


Figure 42: The  $Y'CbCr$  plane at constant luma  $Y = 0.5$  [18].

# Image Formats

---



<b>Learning Outcomes</b>	JPEG
<b>Introduction</b>	<b>JPEG Compression</b>
File Formats	Down-sampling
A General Overview	Isolate the Colour Information
<b>Compression Methods</b>	Throw Away some Colour Information
Lossy Compression	Convert Image to Frequency Domain
<b>Important File Formats</b>	Quantisation
RAW File	Lossless Data Compression



## Learning Outcomes

- (LO1) Types of File Formats used,
- (LO2) Defining Compression and its Types,
- (LO3) A Look into Selective File Types.
- (LO4) A Deep Dive into JPEG Compression.



# Image Formats

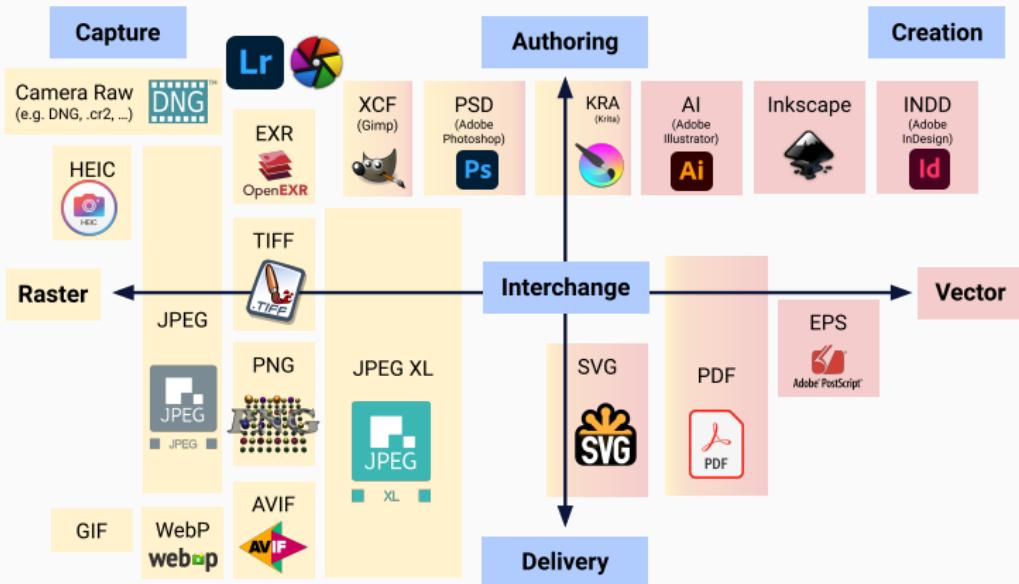


Figure 43: Categorization of image formats by scope which are used in commercial, industrial and personal use [60].



## TIFF( .tif , .tiff ) Tagged Image File Format

- Store image data without losing any data.
  - It does **not perform any compression** on images,
  - High-quality image is obtained however the image size is large,
- Good for printing, and professional work.

## JPEG ( .jpg , .jpeg ) Joint Photographic Experts Group

- It is a loss-prone (lossy) format.
  - Some **data is lost to reduce the image size**.
  - Due to compression, some data is lost but that loss is very less
- Used in digital cameras, non-professional prints, e-Mail, etc.



## GIF ( .gif ) GIF or Graphics Interchange Format

- Primarily used in web graphics (i.e., Reddit, WhatsApp.)
  - They can be animated and are limited to only 256 colours.
  - These images can also allow transparency.
- GIF files are typically small in size compared to other formats.

## PNG ( .png ) PNG or Portable Network Graphics

- It is a lossless image format.
  - It was designed to replace GIF as it supported 256 colours.
  - PNG, whereas, supports 16 million colours.



## Bitmap ( .bmp ) Bit Map Image

- It was developed by Microsoft for windows.
- It is same as TIFF due to lossless, no compression property.
- As it is a proprietary format, it is generally recommended to use TIFF.

## EPS ( .eps ) Encapsulated PostScript

- It is a vector file.
- EPS files can be opened in applications such as Adobe Illustrator.



## RAW Image Files ( .raw , .cr2 , .nef , .orf , .sr2 )

- Unprocessed and created by a camera or scanner.
- Many DSLR cameras can shoot in RAW.
- These images are the **equivalent of a digital negative**, meaning that they hold a lot of image information.
- It saves metadata and is used for photography.



- The class of data compression methods using **inexact** approximations and partial data discarding to represent the content.
- These techniques are used to reduce data size for storing, handling, and transmitting content.
- It can also remove metadata to save up on space [2].
- An example would be **JPEG** compression.

Data is permanently removed from the file.

Well-designed lossy compression technology often reduces file sizes significantly before degradation is noticed by the end-user.



- A class of data compression that allows the original data to be **perfectly** reconstructed from the compressed data with no loss of information.
- Lossless compression is possible because most real-world data exhibits statistical redundancy [57].
- In essence, lossless compression algorithms are needed in cases that require compression where we want the reconstruction to be identical to the original.
- Huffman Coding is a perfect example of lossless compression [39].



- A camera RAW image file contains unprocessed or minimally processed data from the image sensor of either a digital camera, a motion picture film scanner, or other image scanner.
- Named RAW as they are not yet processed, and contain large amounts of potentially **redundant data**.
- Normally, the image is processed by a RAW converter, in a wide-gamut internal color space where precise adjustments can be made before conversion to a viewable format.
- There are dozens of RAW formats in use by different manufacturers of digital image capture equipment.
  - i.e., Nikon uses `.nef` format, and Canon uses `.cr2` .



- It is a **lossy compression** method.
- Usually stored in the **.jfif** (JPEG File Interchange Format) or the **.exif** (Exchangeable image file format) file format.
- The JPEG filename extension is **.jpg** or **.jpeg**.
- Nearly every camera can save images in the **.jpeg**, which supports eight-bit grayscale images and 24-bit color images
  - Eight bits each for **red** , **green** , **blue** .
- Compression can result in a significant reduction of the file size.
- Applications can determine the degree of compression to apply.
- When not too high, compression does not affect or detract from the image's quality, but JPEG files suffer **generation loss** when repeatedly edited and saved.



**Figure 44:** A photo of a European wildcat with the compression rate, and associated losses, decreasing from left to right [25].



- For good lossy compression, throw away the unnecessary information.
- In the case of image compression, Start by understanding which parts of an image are important to human perception, and which aren't.
- Then you find a way to keep the important qualities and trash the rest.
- JPEG, uses two (2) psychovisual principles to compress the image.



1. Changes in brightness are more important than changes in colour:

Human retina contains about 120 million brightness-sensitive rod cells, but only about 6 million colour-sensitive cone cells.

2. Low-frequency changes are more important than high-frequency changes.

The human eye is good at judging low-frequency light changes, like the edges of objects. It is less accurate at judging high-frequency light changes, like the fine detail in a busy pattern or texture. Camouflage works in part because higher-frequency patterns disrupt the lower-frequency edges of the thing camouflaged.



- Each pixel is stored as three numbers:
  - Representing red, green and blue.

The problem, is that the image's brightness information is spread evenly across RGB.

Remember, brightness is more important than colour.

- We want to isolate brightness from the colour information so we can deal with it separately.
- To do this, JPEG converts the image from RGB to YCbCr.
- In YCbCr all brightness information in one channel ( $Y$ ) while splitting the colour information between the other two ( $Cb$  and  $Cr$ ).



- Before doing processing, JPEG throws away some of the colour information by scaling down just the Cb and Cr (colour) channels while keeping the important Y (brightness) channel full size.

This step is optional.

- You can keep all of the colour information, half of it, or a quarter.
- For images, most software will keep half of the colour information;
- for video it is usually a quarter and for this lets do quarter.
- We started with 3 full channels and now we have 1 full channel and  $2 \times 0.25$  channels, for a total of 1.5.
- We are already down to half of the information we started with.



- To use the second observation about perception, start by dividing each of the Y, Cb, and Cr channels up into 8-by-8 blocks of pixels.
- Transform each of these blocks from the spatial domain to the frequency domain.
- Consider just one of these 8-by-8 blocks from the Y channel.
- The spatial domain is what we have now:
  - the value in the upper-left corner represents the brightness (Y-value) of the pixel in the upper-left corner of that block. Likewise, the value in the lower-right corner represents the brightness of the pixel in the lower-right corner of that block. Hence the term spatial: position in the block represents position in the image.



- When we transform this block to the frequency domain, position in the block will instead represent a frequency band in that block of the image. The value in the upper-left corner of the block will represent the lowest-frequency information and the value in the lower-right corner of the block will represent the highest-frequency information.

# Image Formats



11	11	10	10	10	10	9	9
11	11	10	10	10	10	9	9
11	11	12	12	9	9	8	8
11	11	12	12	9	9	8	8
10	10	10	10	11	11	9	9
10	10	10	10	11	11	9	9
10	10	8	8	11	11	10	10
10	10	8	8	11	11	10	10

79.5	3.58	-1.39	1.89	0.0	-1.26	0.57	-0.71
0.64	3.91	-0.59	-2.43	0.0	1.62	0.25	-0.78
-0.46	-2.41	2.13	2.09	0.0	-1.4	-0.88	0.48
0.22	-1.61	-0.21	1.34	0.0	-0.9	0.09	0.32
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-0.15	1.08	0.14	-0.9	0.0	0.6	-0.06	-0.21
0.19	1.0	-0.88	-0.87	0.0	0.58	0.37	-0.2
-0.13	-0.78	0.12	0.48	0.0	-0.32	-0.05	0.15

**Figure 45:** On the left we have an input of 8x8 pixels, on the right we have DCT coefficients rounded to two decimal places.



- The next step is to selectively throw away some of the frequency information.
- If you have ever saved a JPEG image and chosen a quality value, this is where that choice comes into play.
- Start with two 8-by-8 tables of whole numbers, called the quantization tables.
- One table is for brightness information, and one is for colour information.
- You will use these numbers on each of the 8-by-8 blocks in the image data by dividing the frequency value in the image data by the corresponding number in its quantization table.



- So the upper-left corner of each  $8 \times 8$  block in the Y frequency channel will be divided by the number in the upper-left corner of the brightness quantization table, and so on.
- The result of each division is rounded to the nearest whole number and the fractional parts are thrown away.

The quantization tables are saved along with the image data in the JPEG file. They'll be needed to decode the image correctly.



- If you think carefully about what just happened, you will realize that even though we threw away some frequency information by tossing the decimal parts after division, we still have the same amount of data:
- one number for each pixel from each of the three channels.
- However, this data is now going to be compressed using traditional lossless compression.



- But wait, wasn't the whole reason we used lossy compression in the first place that lossless compression doesn't work well for images? Yes, but that quantization we just did is going to make the data more compressible by making it less noisy. To see why, compare these three number sequences:
- JPEG has one last trick for making the data more compressible: it lists the values for each  $8\times 8$  block in a zig-zag pattern that puts the numbers in order from lowest frequency to highest. That means that the most heavily quantized parts (with the largest divisors) are next to each other to make nice, repetitive patterns of small numbers.

# Camera

---



<b>Learning Outcomes</b>	Signal-to-Noise Ratio
<b>Camera Sensors</b>	Thermal Noise (Dark Current)
Charge Coupled Devices	Quantum Efficiency
Complementary MOS	ISO
CCD v. CMOS	<b>Camera Lenses</b>
Colour Filters	Introduction
<b>Camera Properties</b>	Aperture
Linearity	Types of Lenses
Sensitivity	Prime Lens



## Learning Outcomes

- (LO1) An Overview of Camera Sensors,
- (LO2) Camera as a Scientific Instrument,
- (LO3) Working with Colour Filters,
- (LO4) Lens Technologies.





- The cameras and recording media available for modern digital image processing applications are constantly changing.
- In this section we will focus on two (2) technologies:
  - CCD** Charge Coupled Device,
  - CMOS** Complementary Metal-Oxide-Semiconductor,
- The techniques that are used in these technologies are **universal**.
  
- There are also variations of this technologies which are more recent:
  - EMCCD** Electron-multiplying charge-coupled device [55],
  - sCMOS** Back-illuminated CMOS [46].



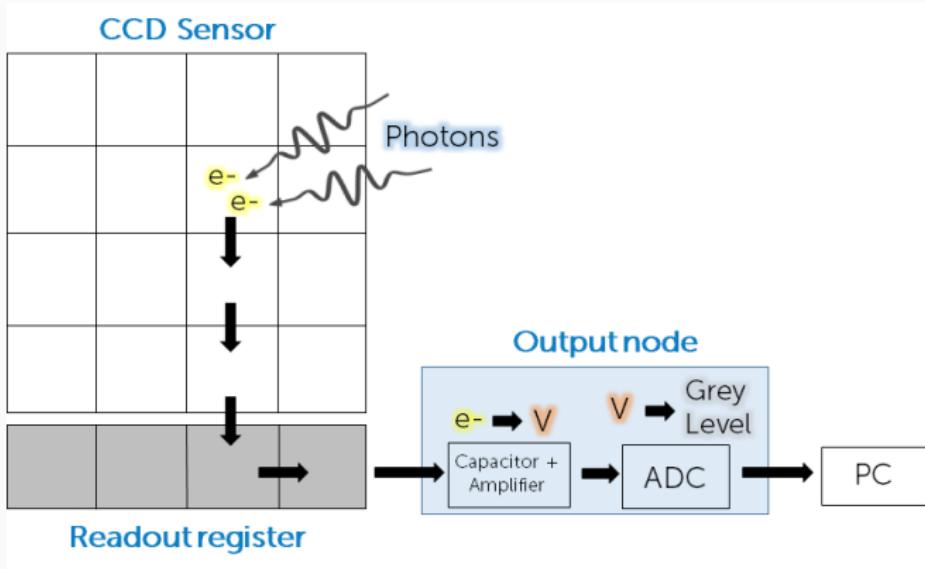
**Figure 46:** The first digital camera was invented in the 1970s by Kodak [17].



- First sensors used in digital cameras, available since 1970s [64].
- A Charge Coupled Device (CCD) is a **highly sensitive photon detector**.
- Divided up into a large number of light-sensitive small areas (known as pixels) which can be used to build up an image.
- A photon of light which falls within the area defined by one of the pixels will be converted into one (or more) electrons and the number of electrons collected will be directly proportional to the intensity of the scene at each pixel.

This allows the camera to be quantitative.

- When the CCD is clocked out, the number of electrons in each pixel are measured and the scene can be reconstructed.



**Figure 47:** How a CCD sensor works. Photons hit a pixel and are converted to electrons, which are then shuttled down the sensor to the readout register, and then to the output node, where they are converted to a voltage, then grey levels, and then displayed with a PC. [64]



## Limitations

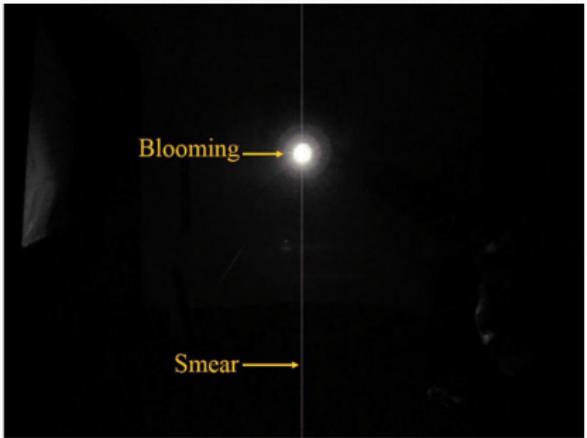
- The main issues with CCDs are their lack of speed and sensitivity, making it a challenge to perform low-light imaging or to capture dynamic moving samples.

**Sensor** millions of pixels of signal have to be shuttled through one node, creating a bottleneck and slowing the camera.

**Noise** to control the read noise from fast moving electrons, CCDs slows down the electrons.

**Refresh** Whole sensor needs to be cleared of the electron signal before the next frame can be exposed

- Small full-well capacity limits electron storage in each pixel.
- If a pixel becomes full and displayed the brightest signal, and blooming, where the pixel overflows and the excess signal is smeared down the sensor as the electrons are moved to the readout register.

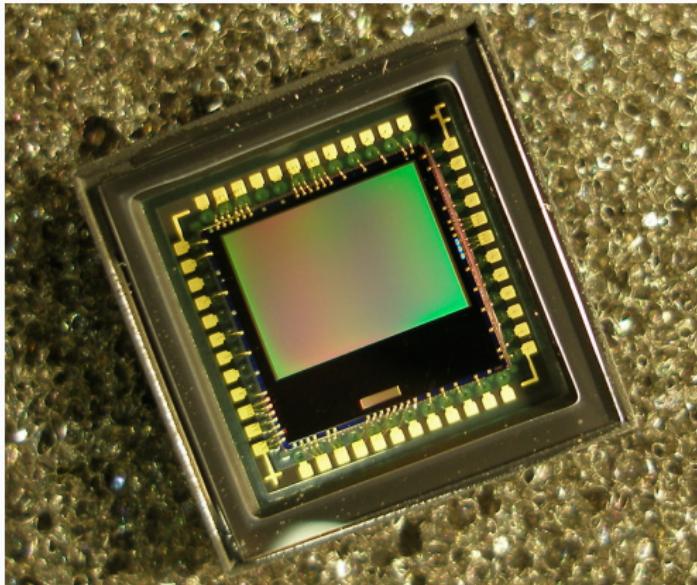


**Figure 48:** Examples of blooming caused by saturation of a CCD sensor pixel. Left) Picture of a sunset. The sun is so bright in the image that there is blooming on the sun itself, leaking into the surrounding pixels, and a vertical smear across the whole image. Right) A similar situation with the blooming and smear labeled [64].

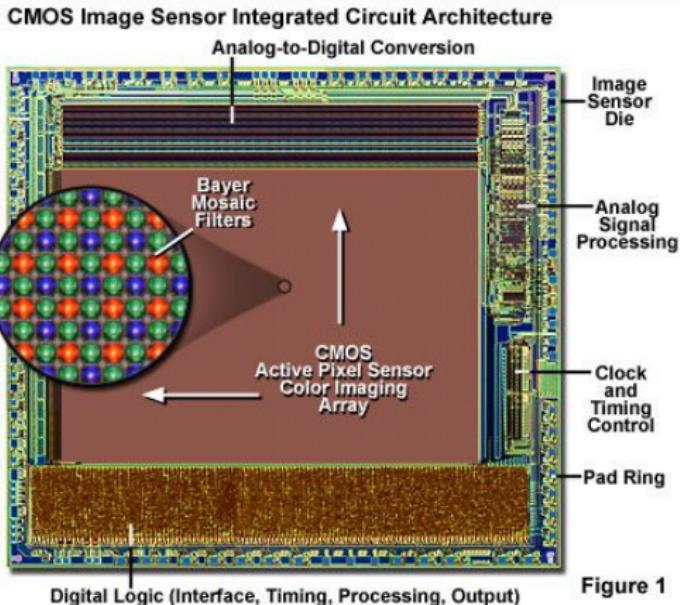


- The CMOS image sensor is made up of an array of tiny light-sensitive cells also known as pixels, each of which is connected to a transistor that acts as a switch [23].
- When light strikes the pixels, it is converted into an electrical charge which is amplified and read out by the sensor's readout electronics.
- The output is then converted into a digital signal, ready for further image processing or storage.

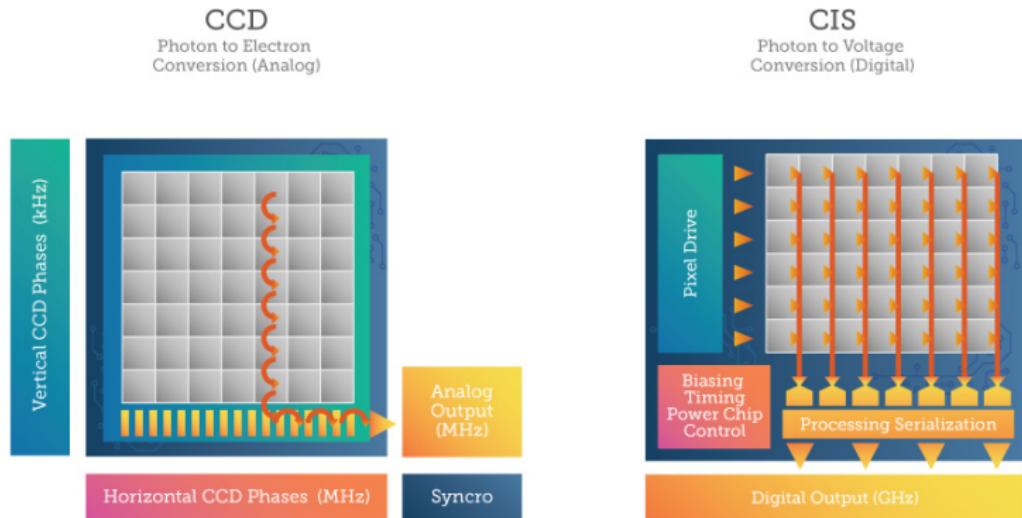
Emerged as an alternative to charge-coupled device (CCD) image sensors and eventually outsold them by the mid-2000s [22].



**Figure 49:** CMOS image sensor [20].



**Figure 50:** An integrated CMOS Image sensor [44].



**Figure 51:** A comparison on the working principle of two types of sensors: CCD and CMOS [24].



The first difference lies in the way the image data is captured/read.

- A CCD captures image data by moving charge packets in a linear sequence from one pixel to the next, which is referred to as **charge transfer**.
- This method results in improved image quality, but it also has drawbacks such as a slower readout speed (i.e., pixels per second).
- On the other hand, CMOS capture image data by reading out each pixel individually,
  - a process known as **parallel readout**.

This results in a faster readout time but generally lower image quality compared to CCD sensors.



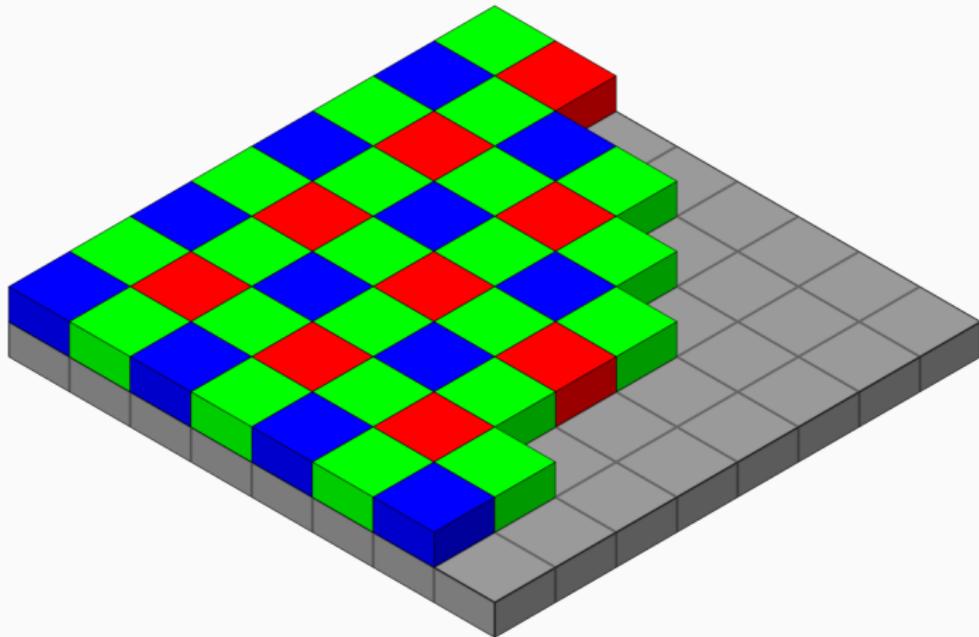
- CMOS is cheaper to produce than CCD due to their more complex electronic circuitry and manufacturing process.
- CCD is known to be more power-hungry as compared to CMOS.
  - CCD needs active power as opposed to the CMOS which don't.
- CMOS has a distinct advantage as readout circuits for each pixel is integrated on the same chip as the pixel.
  - This allows for a wide range of enhanced functions such as image processing and signal amplification.

CCDs are known for their high image quality, lower noise, and greater light sensitivity while CMOSs have the advantage of being faster, less power-hungry and more cost-effective.

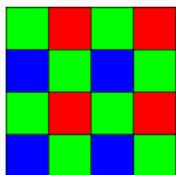


- A color filter array (CFA), or color filter mosaic (CFM), is a mosaic of tiny color filters placed over the pixel sensors of an image sensor to capture color information.
- Color filters are needed because the typical photosensors detect light intensity with little or no wavelength specificity and therefore cannot separate color information [43].

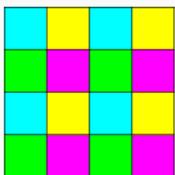
Diazonaphthoquinone (DNQ)-novolac photoresist is one material used as the carrier for making color filters from color dyes or pigments [38, 10].



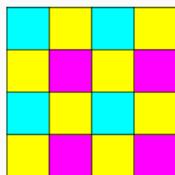
**Figure 52:** The Bayer color filter mosaic. Each two-by-two submosaic contains 2 green, 1 blue, and 1 red filter, each filter covering one pixel sensor. The reason as to why there are two greens to red and blue is to mimic the physiology of the human eye [8].



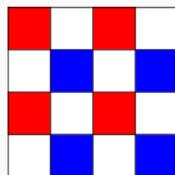
(a) Bayer Filter



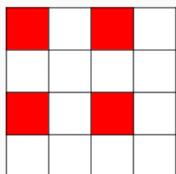
(b) CYGM Filter



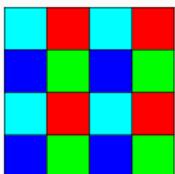
(c) CYYM Filter



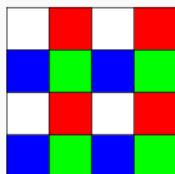
(d) RCCB Filter



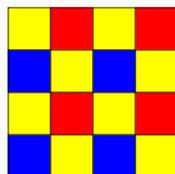
(e) RCCC Filter



(f) RGBE Filter



(g) RGBW Filter



(h) RYYB Filter

**Figure 53:** Types of colour filter used in commercial and industrial applications



**Figure 54:** Sony F828, produced in 2003 which uses an RGBE filter. Once the image is taken a demosaicing algorithm is used to reconstruct an image [21].



- It is generally desirable the relationship between the input physical signal and the output signal be linear [66].
- This means that if we have two images,  $a$  and  $b$ :

$$\mathcal{C} = \mathcal{R}\{w_1a + w_2b\} = w_1\mathcal{R}\{a\} + w_2\mathcal{R}\{b\}.$$

where  $\mathcal{R}(\cdot)$  is the camera response and  $\mathcal{C}$  is the camera output.

- In practice the relationship between  $a$  and  $\mathcal{C}$  is given by:

$$\mathcal{C} = \text{Gain} \cdot a^\gamma + \text{Offset}.$$

where  $\gamma$  is the gamma of the recording medium.

- A linear system must have  $\gamma = 1$  and Offset = 0.
- Unfortunately, Offset is never zero and must be compensated for if the intention is to extract intensity measurements.



- There are two (2) ways to describe the **sensitivity** of a camera.
  1. We can determine the minimum number of detectable photoelectrons called **absolute sensitivity**.
  2. We can describe the number of photoelectrons necessary to change from one digital brightness level to the next called **relative sensitivity**.



Figure 55: Types of camera lenses.



## Absolute Sensitivity

- We need a characterisation of the camera in terms of its **noise**.
- If total noise has a  $\sigma$  of, 100 photoelectrons, then to ensure detectability of a signal we could say that, at the  $3\sigma$  level (% 99.7), the minimum detectable signal (or absolute sensitivity) would be 300 photoelectrons.
- If all the noise sources, with the exception of photon noise, can be reduced to negligible levels, this means that an absolute sensitivity of less than 10 photoelectrons.
- This can be rephrase to the **number of photons needed to have signal equal to noise** [63].
- This quantity plays an important role in very-low light applications.



## Relative Sensitivity

- When coupled to the linear case, with  $\gamma = 1$ , leads to the result:

$$\text{Relative Sensitivity} = 1/\text{gain}.$$

The sensitivity or gain can be deduced in two (2) distinct ways.

- If  $a$  can be precisely controlled by either **shutter time** or intensity, the gain can be estimated by estimating the slope of the resulting straight-line curve.
  - To translate this to units, a standard source must be used emitting photons to the camera sensor and the quantum efficiency ( $\eta$ ) of the sensor must be known.
- If the limiting effect of the camera is only the **photon noise**, then:

$$S = \frac{\mu}{\sigma^2} = \frac{m_c}{s_c}$$



Camera	Pixels	Pixel Size	Temp.	S	Bits
Label		$\mu\text{m} \times \mu\text{m}$	K	$e^-/\text{ADU}$	
C-1	$1320 \times 1035$	$6.8 \times 6.8$	231	7.9	12
C-2	$578 \times 385$	$22.0 \times 22.0$	227	9.7	16
C-3	$1320 \times 1035$	$6.8 \times 6.8$	293	48.1	10
C-4	$576 \times 384$	$23.0 \times 23.0$	238	90.9	12
C-5	$756 \times 581$	$11.0 \times 5.5$	300	109.2	8

**Table 3:** Sensitivity measurements of sample cameras.

- In a scientific-grade CCD camera (C-1), only 8 photoelectrons (approximately 16 photons) separate two gray levels in the digital representation of the image.
- For a considerably less expensive video camera (C-5), only about 110 photoelectrons (approximately 220 photons) separate two gray levels.



- In a modern camera (CCD or CMOS), the noise is defined by:
  - Amplifier noise in the case of colour cameras.
    - This is more apparent in blue channel.
  - Thermal noise which, itself, is limited by the chip temperature  $K$  and the exposure time  $T$ ,
  - Photon noise which is limited by the photon production rate  $\rho$  and the exposure time  $T$ .

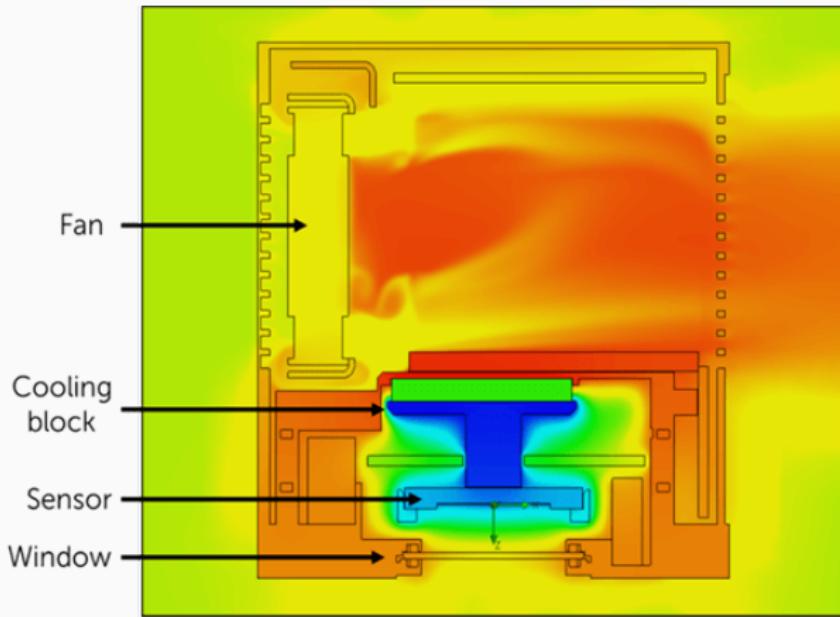
We will have a detailed look into these in our lecture called **Noise**.



- Caused by the thermal energy within the camera sensor [65].
- As sensor works, the heat from operation causes thermal energy generation.

These thermal electrons are independent of the photo-electrons.

- Cameras can't decide if the electrons are from the light or thermal.
- Any thermal electrons that accumulated in the sensor pixel wells are counted as signal upon readout, despite **not being part of the signal from the sample**.
- This is known as the dark current.



**Figure 56:** Scientific camera cooling and Citadel Chamber Technology. It shows typical temperature levels within a CMOS camera. The sensor is cooled absolutely and evenly, is an optimal distance from the window, and heat is effectively expelled (edited from [65]).



Camera	Dark Current	Data Sheet
Prime 95B sCMOS (Normal Operation)	0.55	
Prime 95B sCMOS (Air-Cooled)	0.3	
Retiga R6 CCD	0.00073	
Retiga E7 CMOS	0.001	
Nikon D5300 (ISO 200)	6.23	

**Table 4:** Values of Dark Current on Cameras.

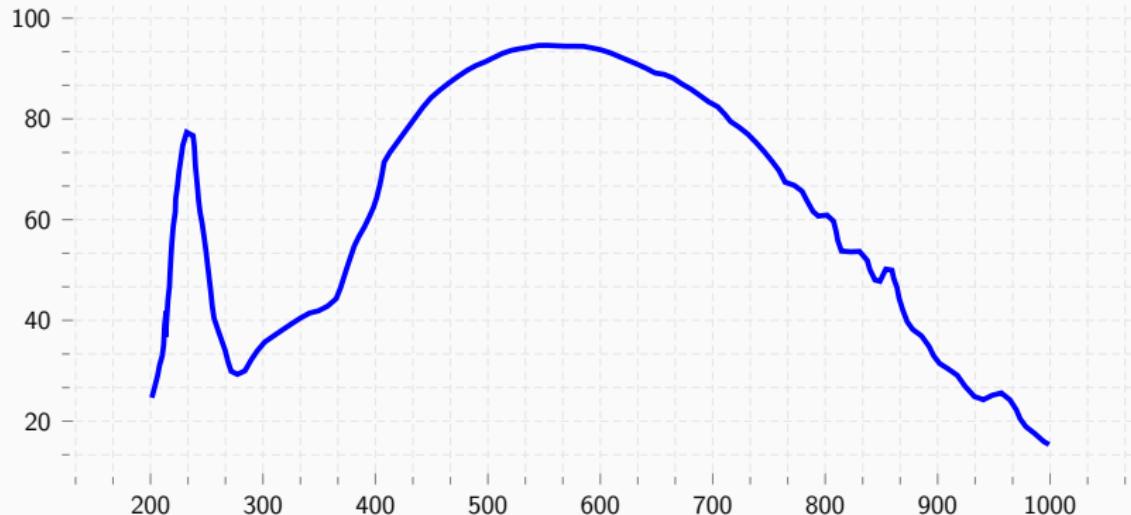


- Quantum efficiency (QE) is the measure of the **effectiveness** to convert incident photons into electrons.

For example, if a sensor had a QE of 100% and was exposed to 100 photons, it would produce 100 electrons of signal.

- In practice, sensors are never 100% efficient, and different sensor technologies have different QE values.

The highest-end scientific cameras can achieve up to 95% QE but this is dependent on the wavelength of light being detected.



**Figure 57:** The quantum efficiency (QE) of a 95% quantum efficient sensor at different photon wavelengths. 95% QE is possible at 500-600 nm wavelengths (green/yellow) but it is less efficient at shorter (violet, 300-400 nm) and longer (infrared, 800-1000 nm) wavelengths. This particular sensor also has a peak in QE at near-UV wavelengths, around 220-250 nm.



- ISO is a camera setting which brightens or darkens a photo.
- Increasing the ISO number, makes photos progressively brighter.
- For that reason, ISO can help you capture images in darker environments, or be more flexible about your aperture and shutter speed settings.

Raising your ISO has consequences. A photo taken at too high of an ISO will show a lot of grain, also known as noise, and might not be usable.

- Brightening a photo via ISO is always a trade-off.
- Only raise your ISO when you are unable to brighten the photo via shutter speed or aperture instead.



**Figure 58:** A comparison of different ISO numbers. The ISO controls how sensitive the camera is to the light and therefore a higher ISO number would indicate a brighter image [9].



- A typical lens used with an industrial camera is actually a lens system made of multiple types of optical lenses within an enclosure.
- This type of camera lens is technically called a “compound lens.”
- However, a camera lens is the colloquial term for a lens system when it is built with a mounting ring that can fit a variety of cameras.
- Choosing the correct camera lens for a vision system is critical for achieving a specific imaging result.

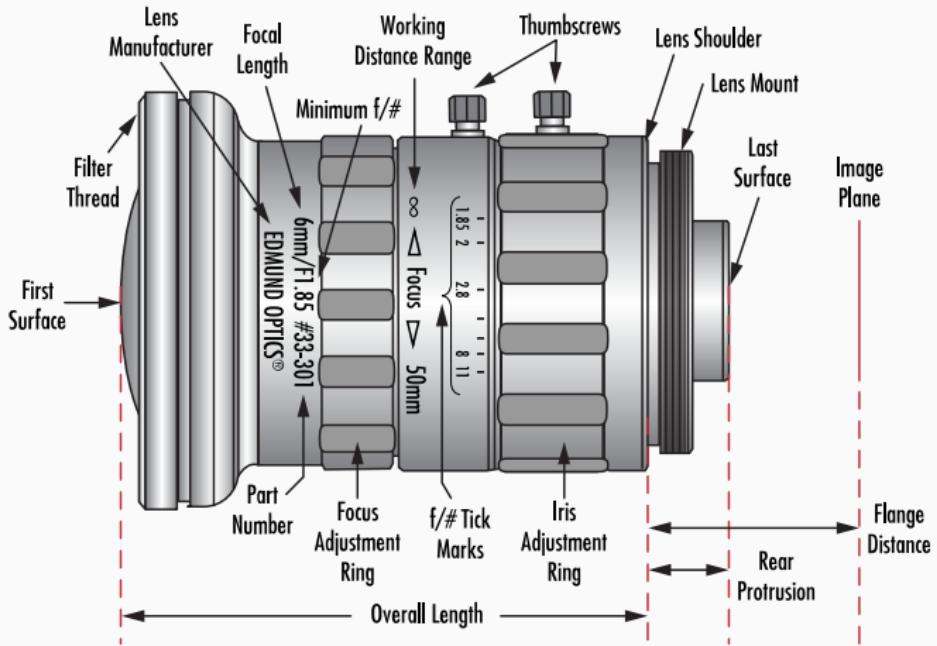


Figure 59: Anatomy of a lens [16].



$f/1.4$



$f/2.0$



$f/2.8$



$f/4.0$



$f/5.6$



$f/8.0$



Figure 60: Different apertures of a lens.



## APERTURE

LESS LIGHT



f/22

f/14

f/8

f/5.6

f/4

f/2.8

*"SLOWER"  
"SMALLER"*

*"FASTER"  
"WIDER"*

MORE LIGHT

Figure 61: The effects of aperture on the image.



**Figure 62:** The entrance pupil is typically about 4 mm in diameter, although it can range from as narrow as 2 mm ( $f/8.3$ ) in diameter in a brightly lit place to 8 mm ( $f/2.1$ )



- Choosing the proper lens needs careful consideration **focal length**.
- Focal length is the distance between the camera's sensor and where the light in the camera lens is focused.
- It is often measured in millimetres (mm) and provides plenty of information about what type of image will be captured,
  - such as the angle of view (often referred to as the field of view or FoV)
  - the magnification of the target within the image.



# FOCAL LENGTH AND ANGLE OF VIEW

TheDARKROOM.com

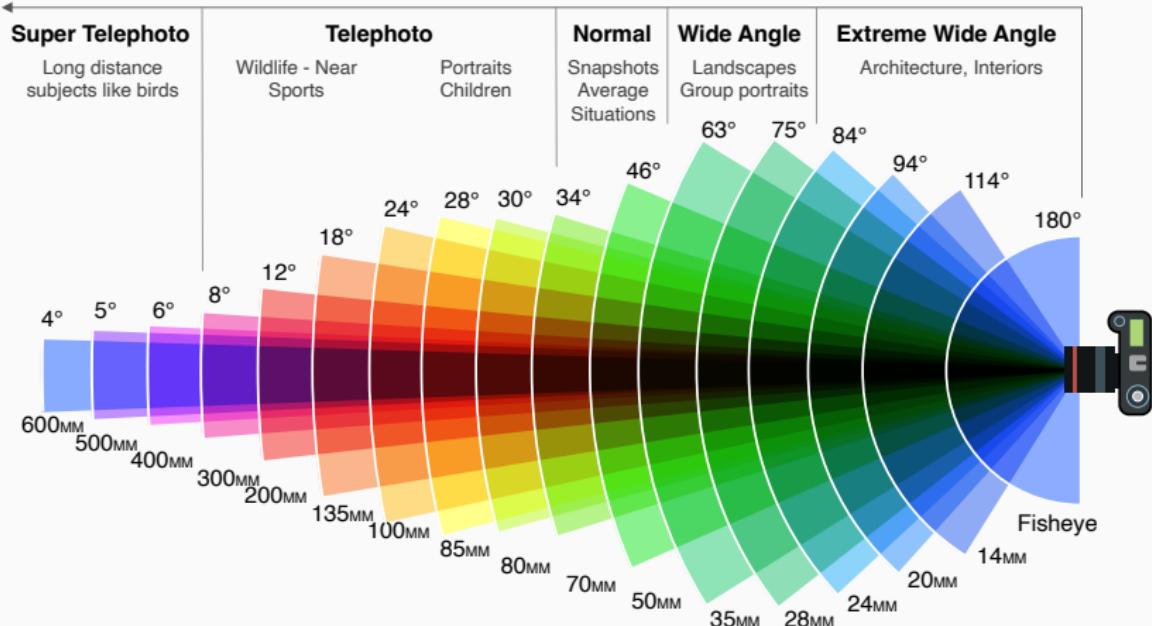


Figure 63: The effects of focal length and the angle of view [11].



- There are two (2) main types of camera lenses:
  1. Prime lens,
  2. Zoom lens (also called varifocal/kit)

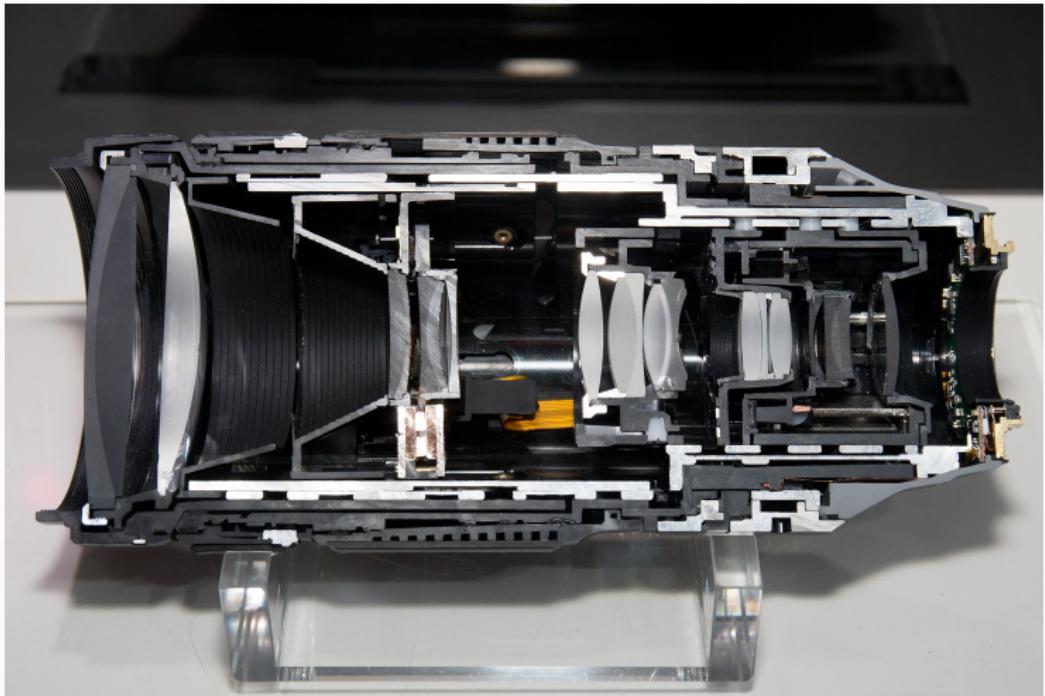
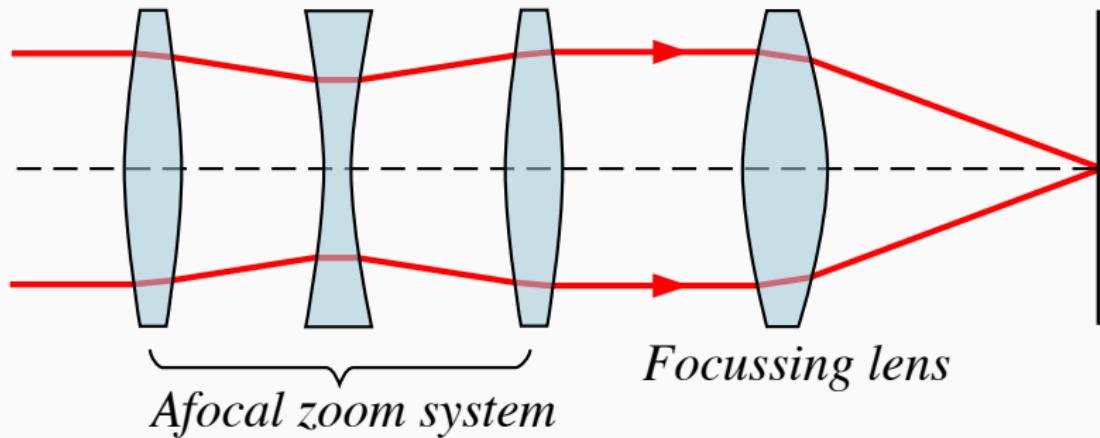


Figure 64: The cross-section of Fujinon XF100-400mm zoom lens [40].



**Figure 65:** A simple zoom lens system. The three lenses of the afocal system are [1], [2], [3] (from left). [1] and [2] can move to the left and right, changing the overall focal length of the system [15]



- An important issue in zoom lens design is the correction of optical aberrations (such as chromatic aberration and, in particular, field curvature) across the whole operating range of the lens;
- This is considerably harder in a zoom lens than a fixed lens, which needs only to correct the aberrations for one focal length.



**Figure 66:** Photographic example showing a high quality lens (top) compared to a lower quality one exhibiting transverse chromatic aberration [68].



- Prime lens is a fixed focal length photographic lens (as opposed to a zoom lens), typically with a maximum aperture from f2.8 to f1.2.
- While a prime lens of a given focal length is less versatile than a zoom lens, it is often of superior optical quality, wider maximum aperture, lighter weight, and smaller size.
- These advantages stem from having fewer moving parts, optical elements optimized for one particular focal length, and a less complicated lens systems that creates fewer optical aberration issues.
- Larger maximum aperture (smaller f-number) facilitates photography in lower light, and a shallower depth of field.

# Display

---

# Table of Contents



<b>Learning Outcomes</b>	Cathode Ray Tube Displays
<b>Introduction</b>	LED Displays
Refresh Rate	LCD Display
<b>Limits of Human Vision</b>	<b>Analog TV Formats</b>
Workings of the Eye	NTSC
<b>Interlacing</b>	PAL
Advantages	Worldwide Standards
Disadvantages	Test Card
	<b>Resolution</b>



## Learning Outcomes

- (LO1) Refresh Rate,
- (LO2) Display Technologies,
- (LO3) TV Standards,
- (LO4) Resolution.





- Defined as the number of complete images that are written to the screen per second.
- For video, TV refresh rate is either 25 (PAL)[67] or 29.97 (NTSC) images.
- For computer displays, the refresh rate can vary with common values being 30 Hz, 60 Hz and 144 Hz.
- At values above 60 Hz visual **flicker is negligible** at virtually all illumination levels.
- To prevent the appearance of visual flicker at refresh rates below 60 Hz, the display can be **interlaced**.
- Standard interlace for video systems is 2:1.
- Since interlacing is not necessary at refresh rates above 60 Hz, an interlace of 1:1 is used with such systems.



It's important to remember images are seen in the first place.

1. Light passes through the cornea at the front of your eye until it hits the lens.
2. The lens then focuses the light on a point at the very back of the eye in a place called the retina.
3. Then, photo-receptor cells at the back of your eye turn the light into electrical signals, while the cells known as rods and cones pick up on motion.
4. The optic nerve carries the electrical signals to your brain, which converts the signals into images.



What happens when watching something with high FPS rate.

- Are you actually seeing all those frames that flash by?
  - After all, your eye doesn't move as fast as 30 motions per second [3].

The short answer is that you may not be able to consciously register those frames, but your eyes and brain may be aware of them.



- 60 FPS rate is generally accepted as the uppermost limit.
- Some research suggests that your brain might actually be able to identify images that you see for a much shorter period of time than experts thought.
- For example, in [51], it was observed the brain can process an image that your eye sees for only 13 milliseconds.
- That's especially rapid when compared with the accepted 100 milliseconds that appears in earlier studies.

Higher FPS also has less stress (i.e., less blinking) on the eye [62].

Thirteen milliseconds translate into about 75 frames per second.



- Interlacing is a method used to create images on a display, like a TV or computer screen [28].
- In an interlaced display, the picture is made by scanning alternating lines [36].
- It scans **every other line** first, and then it fills in the missing lines in the next scan.
- This method lets the screen refresh faster and at a lower cost.
- A big downside is the picture might flicker or show visible lines.



**Figure 67:** An Example of Interleaving during a live-stream event [32].



## Faster Refresh Rate

- Interlacing allows higher refresh rate as it only needs to update half of the lines on the screen at any one time.
  - Images update quickly, making videos appear smoother.

## Reduced Bandwidth

- Interlacing transmits only half the image data at a time.
- Advantageous when streaming video content over slow internet connections as it can start displaying content more quickly,
  - even though the full quality might be achieved a bit later.

## Cost Efficiency

- Displays using interlacing are often less expensive to produce.

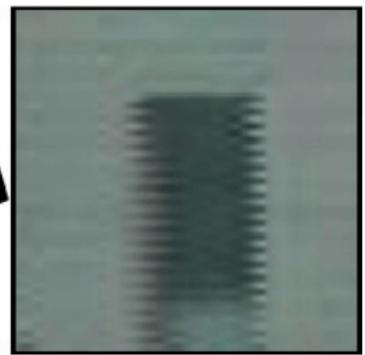


- A common problem is **flickering**, due to updating alternate lines instead of the whole image at once.
- This can make the picture seem unstable or shaky, noticeable when there are fast-moving scenes or during scrolling text.
- Another issue is the appearance of visible lines or **combing**.
- Occurs when the alternating lines don't blend perfectly, making it look like there are gaps or lines through moving objects.
- Artefacts can be distracting and reduces clarity and quality.
- As only half of the image is displayed at a time, interlaced videos can appear less sharp and detailed compared to progressive scan videos, where each frame shows the full image.

Makes Interlaced content seem outdated or lower quality, especially on modern high-resolution displays where every detail counts.



Screenshot



Combing artifacts

Figure 68: An example of the combing effect observed in an interlaced video [36].



**Figure 69:** A monochrome CRT as seen inside a Macintosh Plus computer [37].



- An LED emits light as a result of electric luminescence and is one of the most energy-efficient and power-saving ways to produce light.
- Consists of solid materials without movable parts and is often moulded into transparent plastic.
- An LED display consists of many closely-spaced LEDs.
  - By varying the brightness of each LED, the diodes jointly form an image.
- To create a bright colour image, additive colour mixing are used.
- By adjusting the intensity of the diodes, billions of colours can be formed.

When you look at the LED screen from a certain distance, the array of coloured pixels are seen as an image.

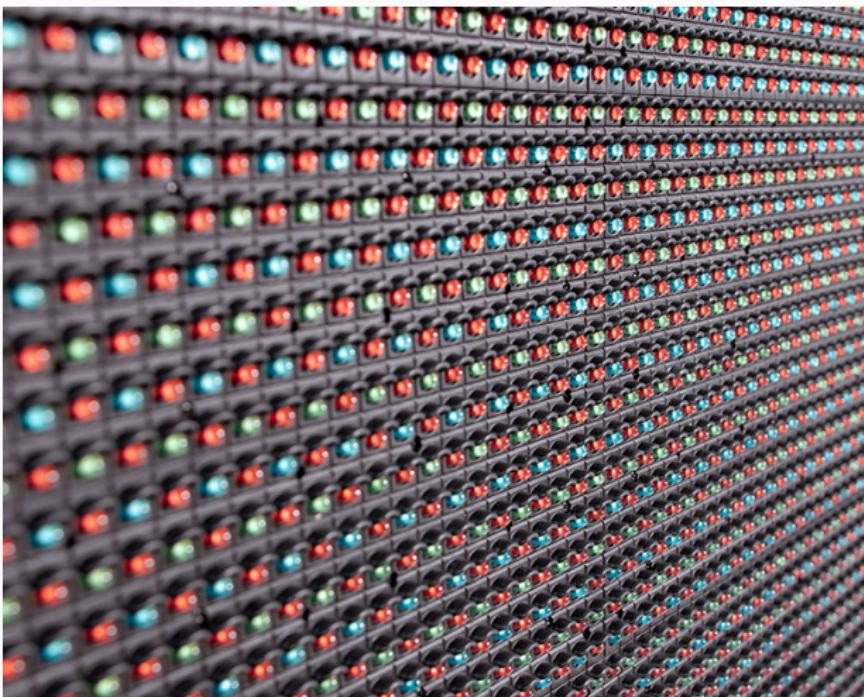


Figure 70: From a close-up it can be seen a display is made from RGB diodes [19].



- LCDs are lit by a backlight, and pixels are switched on and off electronically while using liquid crystals to rotate polarized light.
- A polarising glass filter is placed in front and behind all the pixels, the front filter is placed at 90 degrees.
- In between both filters are the liquid crystals, which can be electronically switched on and off.
- By controlling the electric source the orientation of the liquid crystals can be finely controlled which in turn allows specific light to pass through the second polarising filter.



## LCD

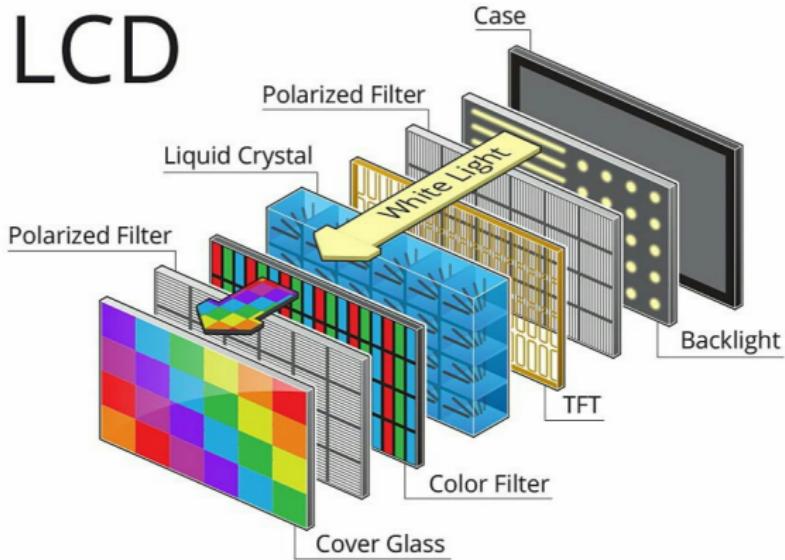


Figure 71: An exploded view of the LCD technology [35].



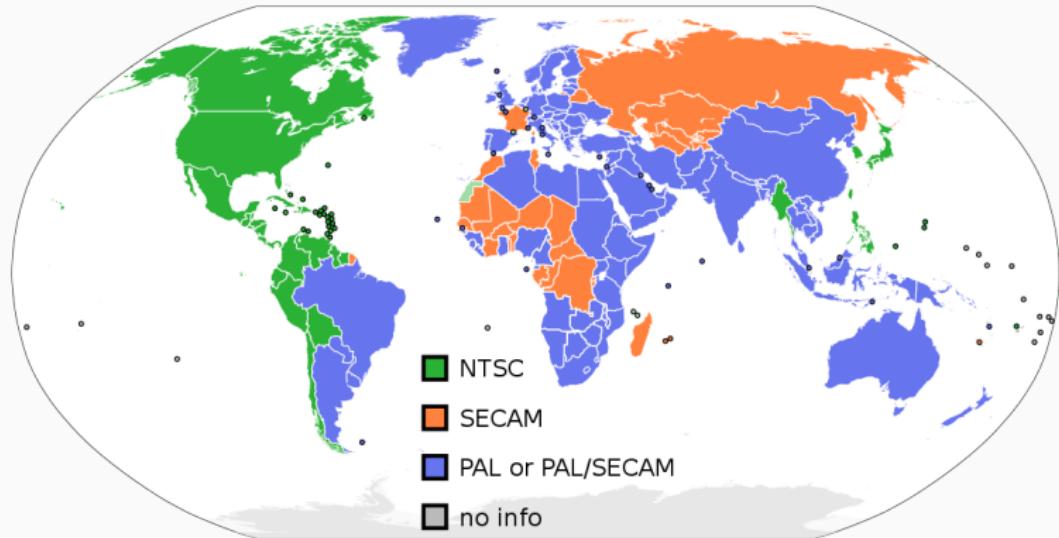
## NTSC (National Television Standard Committee)

- Analog colour-encoding video system used in DVD players and, until recently, television broadcasting in North America.
- In 1953 a new TV standard was introduced by the National Television System Committee and named “NTSC.”
- This format was developed with the intention to be compatible with most TV sets in the country, whether color or black-and-white.
- Even though modern television broadcasters switched to digital, the number of resolution lines and the frame rate they use are the same as established by the NTSC format.

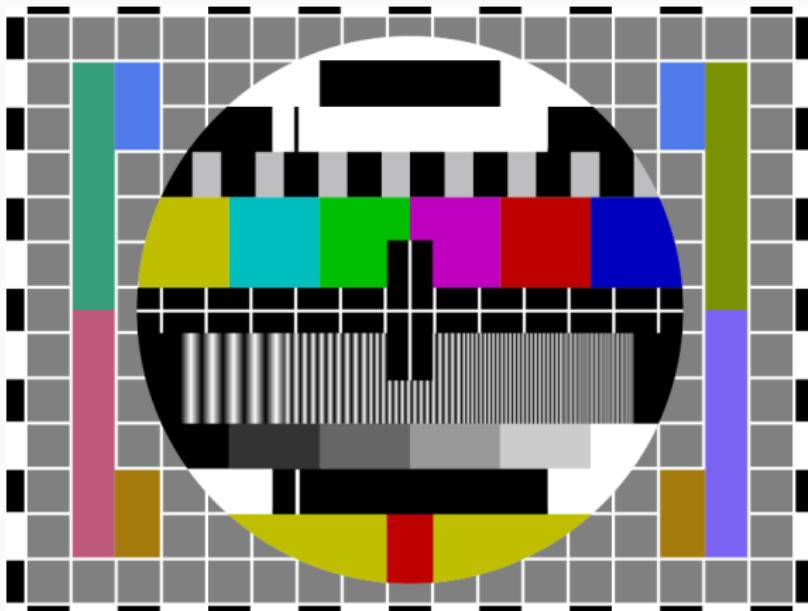


## PAL (Phase Alternating Line)

- Another video mode system for analog color television, also used in DVD and Blu-ray players.
- Designed in the late 1950s in Germany, the PAL format was supposed to deal with certain weaknesses of NTSC, including signal instability under poor weather conditions, which was especially relevant for European broadcasters.
- The new standard was to solve the problem by reversing every other line in a TV signal and thus eliminating errors.
- PAL also provided the locally required picture frequency – 50 Hz.
- This format, unlike NTSC, is still employed for broadcasting in the countries where it was adopted.



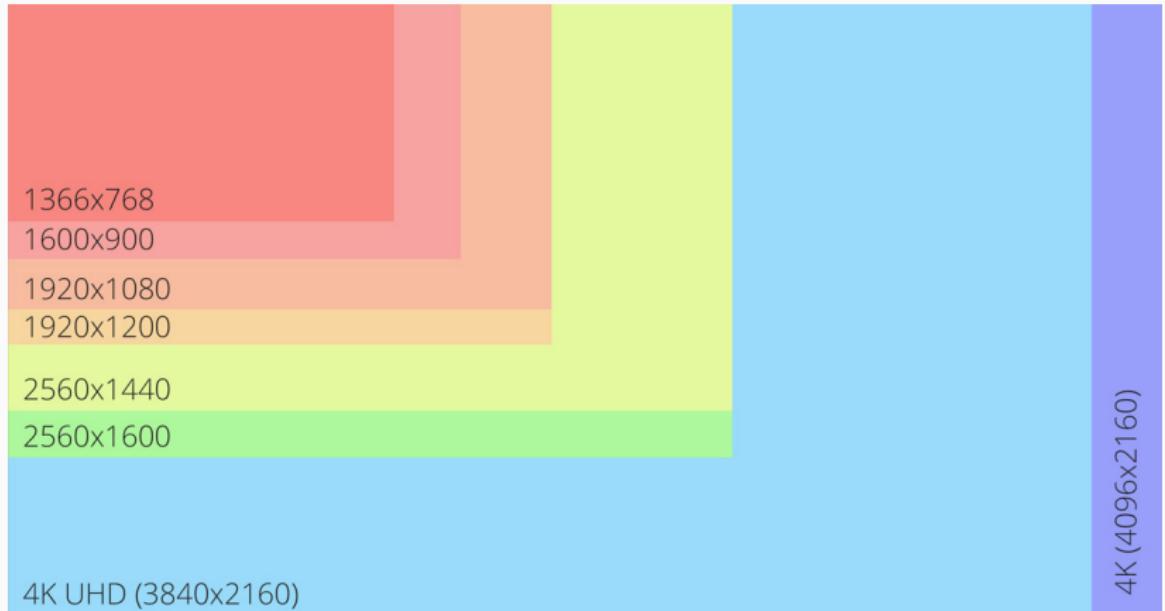
**Figure 72:** Analog television encoding systems by nation; NTSC (**green**), SECAM (Séquentiel de couleur à mémoire) (**orange**), and PAL (**blue**)



**Figure 73:** The Philips circle pattern was a physical card which a television camera was pointed, allowing for simple adjustments of picture quality. Such cards are still often used for calibration, alignment, and matching of cameras and camcorders [48].



- The pixels stored in computer memory, although they are derived from regions of finite area in the original scene, may be thought of as mathematical points having no physical extent.
- When displayed, the space between the points must be filled in.
- The brightness profile of a CRT spot is approximately Gaussian and the number of spots that can be resolved on the display depends on the quality of the system.
- It is relatively straightforward to obtain display systems with a resolution of 72 spots per inch (28.3 spots per cm.)
- This number corresponds to standard printing conventions.



**Figure 74:** A comparison of different resolution standards [31].

# Noise

---

# Table of Contents



<b>Introduction</b>	Photon Distribution
Definition	
<b>Colours of Noise</b>	<b>Thermal Noise</b>
White Noise	Definitions
Pink Noise	Dark Current
Brown Noise	
Violet Noise	<b>Read Noise</b>
Blue Noise	Introduction
<b>Photon Noise</b>	CMOS Read Noise
Introduction	Johnson-Nyquist Noise
Poisson Distribution	Quantisation Noise
	<b>Application</b>
	Perlin Noise

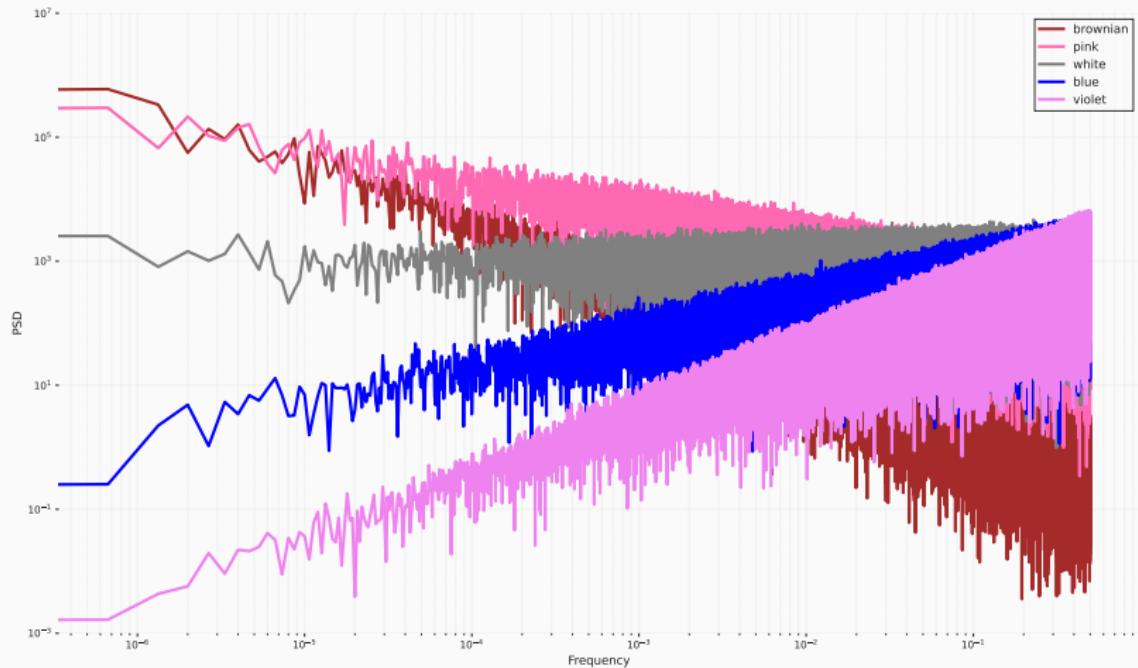


- Images acquired through modern sensors may be **contaminated** by a variety of noise sources.

Noise is of stochastic variations, opposed to being deterministic.

- We assume to be dealing with modern sensors (CCD, CMOS) where photons produce electrons or referred as photo-electrons.
- Nevertheless, most observations we make about noise and its various sources hold equally well for other imaging modalities.

Modern technology has reduced the noise levels associated with various electro-optical devices to almost negligible levels except one which **forms the absolute limiting case**.



**Figure 75:** A comparison of different colours of Noise and their power vs. frequency.



- White noise is a signal, named by analogy to **white light**, with a flat frequency spectrum.
- For example, with a white noise signal, the range of frequencies between 40 Hz and 60 Hz contains the **same amount of power** as the range between 400 Hz and 420 Hz,
  - As both these intervals are 20 Hz wide.



Let's generate some white noise. First is to define `noise_psd`

```
def noise_psd(N, psd = lambda f: 1):
    X_white = np.fft.rfft(np.random.randn(N));
    S = psd(np.fft.rfftfreq(N))
    # Normalize S
    S = S / np.sqrt(np.mean(S**2))
    X_shaped = X_white * S;
    return np.fft.irfft(X_shaped);
```

Continuing on define a template function which will be inherited.

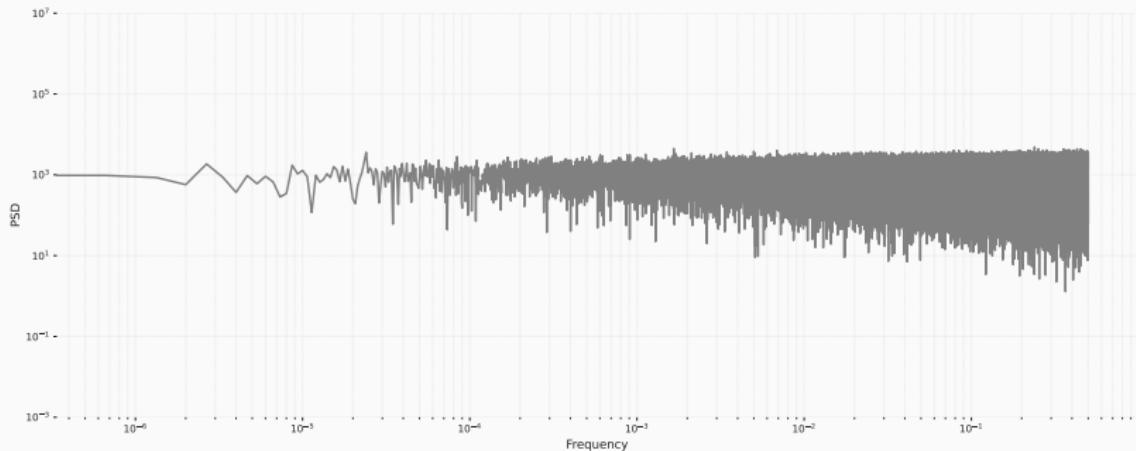
```
def PSDGenerator(f):
    return lambda N: noise_psd(N, f)
```

The aforementioned code uses `anonymous functions`.

# Noise



```
@PSDGenerator  
def white_noise(f):  
    return 1;
```



**Figure 76:** PSD of white noise.



- Pink noise's PSD decreases 3.01 dB per octave with **increasing frequency** (i.e.,  $\propto 1/f$ ).
- Each octave interval (halving or doubling in frequency) carries an **equal amount of noise energy**.

In audio form, Pink noise sounds like a waterfall.

- It is used in tuning loudspeaker systems in professional audio [26].
- It is often observed signals in **biological systems**.
  - i.e., fluctuations in tide and river heights, quasar light emissions, heart beat.

# Noise



```
@PSDGenerator
def pink_noise(f):
    return 1/np.where(f == 0, float('inf'), np.sqrt(f))
```

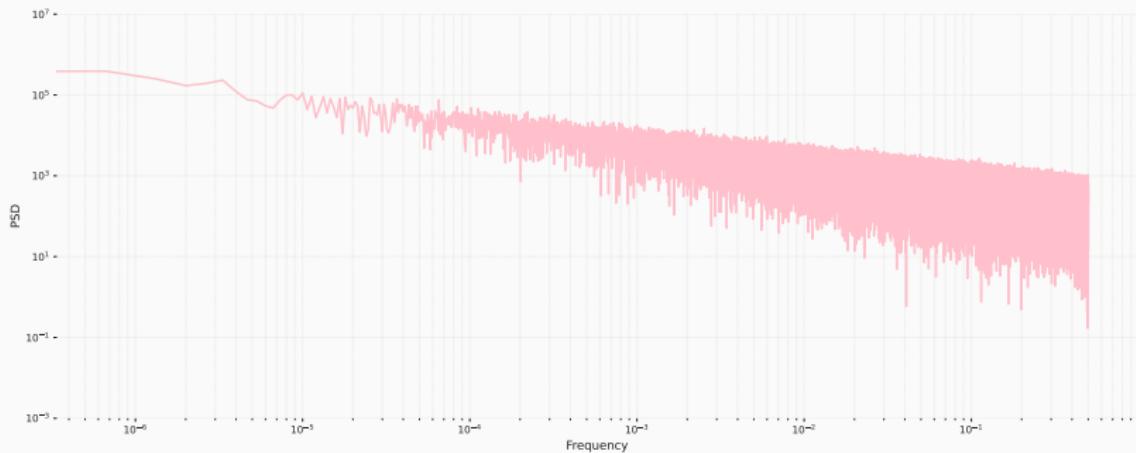


Figure 77: PSD of pink noise.



- Brown noise<sup>1</sup>, has a PSD which decreases 6.02 dB per octave with **increasing** frequency (i.e.,  $\propto 1/f_2$ ).
- i.e., a running washing machine, fan noise from an air conditioner or ventilation system.
- Brown noise can be produced by **integrating white noise**.

Brownian noise can also be computer-generated by first generating a white noise, applying Fourier-transforming, then dividing the amplitudes of the different frequency components by the frequency, or by the frequency squared.

---

<sup>1</sup>It is also known as Brownian Noise.



```
@PSDGenerator
def brownian_noise(f):
    return 1/np.where(f == 0, float('inf'), f)
```

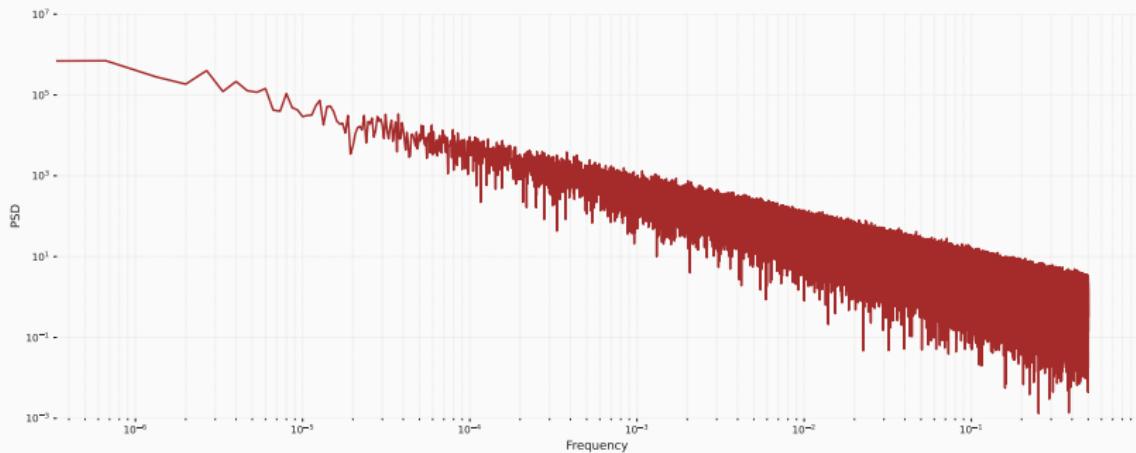


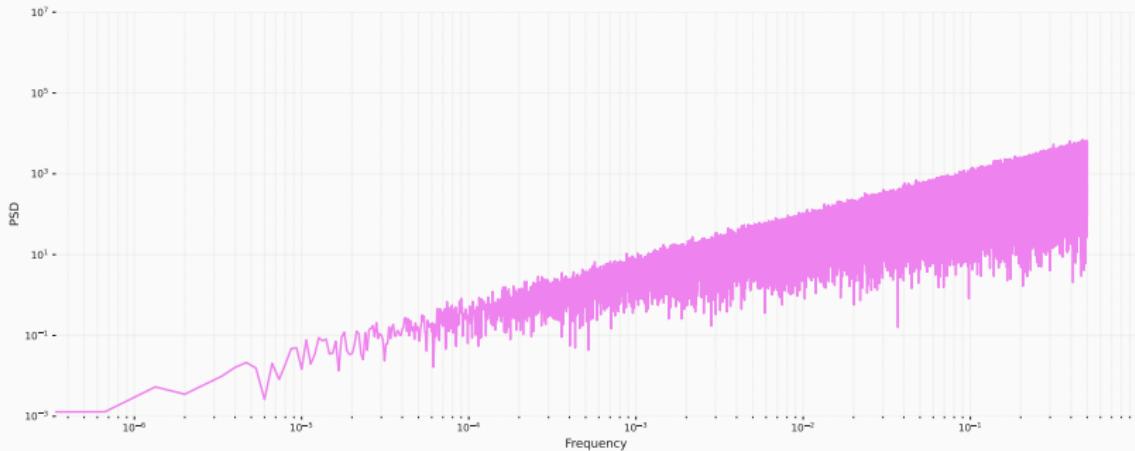
Figure 78: PSD of brown noise.



- Also called **purple** noise.
- It has a PSD which increases 6.02 dB per octave with **increasing** frequency (i.e.,  $\propto f^2$ ).
- GPS acceleration errors are an example of violet noise processes as they are dominated by high-frequency noise.
- It is also known as **differentiated white noise**, due to its being the result of the differentiation of a white noise signal.



```
@PSDGenerator
def violet_noise(f):
    return f;
```



**Figure 79:** PSD of violet noise.



- Also called **azure** noise.
- It has a PSD which increases 3.01 dB per octave with **increasing** frequency (i.e.,  $\propto f$ ).

In computer graphics, the term blue noise is sometimes used more loosely as any noise with minimal low frequency components and no concentrated spikes in energy which is used in for dithering.



```
@PSDGenerator  
def blue_noise(f):  
    return np.sqrt(f);
```

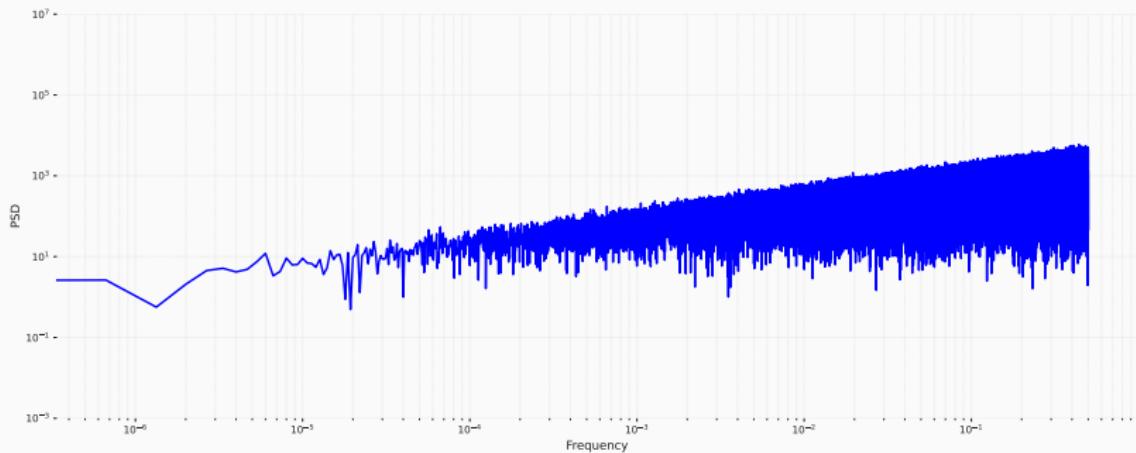
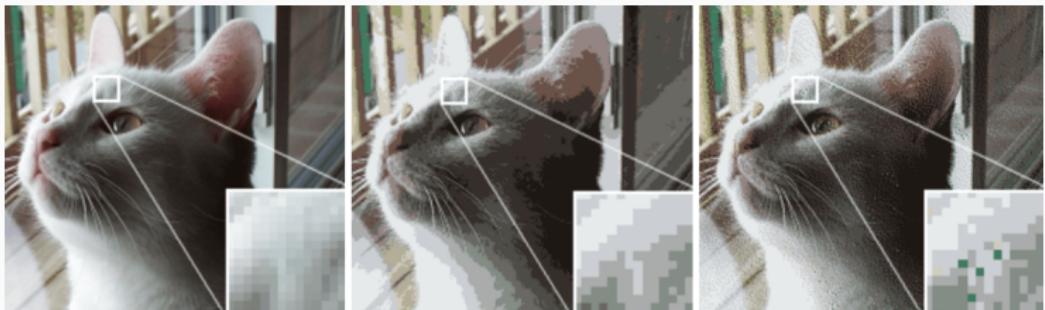


Figure 80: PSD of blue noise.



## Dithering

- An intentional noise to randomise quantization error.
- Used in processing of both digital audio and video data, and is often one of the last stages of mastering audio to a CD.
- A common use is converting a grey-scale image to black and white, so black dot density approximates the average grey level.



**Figure 81:** Image on left is original. Center image reduced to 16 colours. Right image also 16 colors, but dithered to reduce banding effect.



**Figure 82:** An example of colour banding, visible in the sky.



- When the signal is based upon light, then the **quantum nature of light plays a significant role.**
- A single photon at  $\lambda = 500 \text{ nm}$  carries an energy of:

$$E = h\nu = hc/\lambda = 3.97 \times 10^{-19} \text{ J.}$$

- Nowadays, CCDs cameras are able to count individual photons.

The problem comes from the statistical nature of photon.

We cannot assume that, in a given pixel for two consecutive but independent observation intervals of length  $T$ , the same number of photons will be counted.

- Photon production is governed by quantum physics, restricting us to talking about an average number of photons within a window.



## Poisson Process for Photon Noise

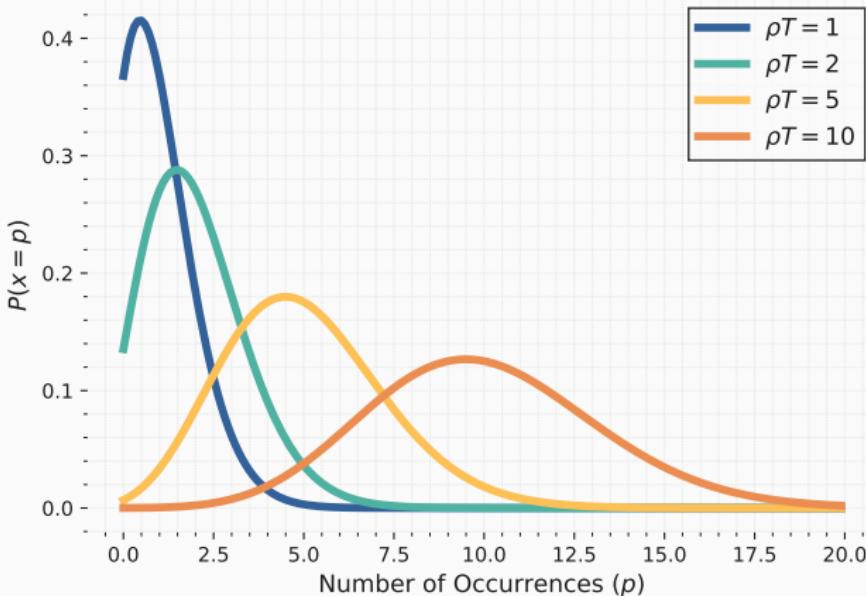


Figure 83: The distribution of the photon noise which is a Poisson process.



- The Poisson distribution is named after Simeon-Denis Poisson.
- Unlike a lot of distributions, it only has one (**1**) parameter:  $\theta$ .

This parameter must be positive.

- The formula for the probability density function (PDF) is:

$$P(X = x) = \frac{e^{-\theta} \theta^x}{x!}, \quad \text{for } x = 0, 1, 2, 3, \dots$$



## Example

Consider a computer system with Poisson job-arrival stream at an average of 2 per minute. Determine the probability that in any one-minute interval there will be:

- (i) 0 jobs,
- (ii) exactly 2 jobs,
- (iii) at most 3 arrivals,
- (iv) What is the maximum jobs that should arrive one minute with 90 % certainty

**Solution****(i) No Job Arrivals**

$$P(X = 0) = e^{-2} \approx .135 \quad \blacksquare$$

**(i) Exactly 3 Job Arrivals**

$$P(X = 3) = e^{-2} \frac{2^3}{3!} \approx .18 \quad \blacksquare$$



---

**Solution**

---

**(iii) At most 3 Arrivals**

$$\begin{aligned}P(X \leq 3) &= P(0) + P(1) + P(2) + P(3), \\&= e^{-2} + e^{-2} \frac{2}{1} + e^{-2} \frac{2^2}{2!} + e^{-2} \frac{2^3}{3!} \\&= .135 + .270 + .270 + .180 \\&= 0.857 \quad \blacksquare\end{aligned}$$

For more than 3 arrivals:

$$\begin{aligned}P(X > 3) &= 1 - P(X \leq 3) \\&= 1 - 0.857 \\&= 0.142 \quad \blacksquare\end{aligned}$$



- The probability distribution for  $p$  photons in an observation window of length  $T$  seconds is known to be Poisson:

$$P(p|\rho, T) = \frac{(\rho T)^p e^{-\rho T}}{p!}$$

where  $\rho$  is the rate of photons per second.

- Even if there were no other noise sources in the imaging chain, the statistical fluctuations of photon counting over a finite time interval  $T$  would still lead to a **finite signal-to-noise ratio (SNR)**.
- Rewriting the SNR, average ( $\mu$ ) value and standard deviation ( $\sigma$ ) are given by:

$$\mu = \rho T \quad \sigma = \sqrt{\rho T} \qquad \text{SNR} = 10 \log_{10} (\rho T) \quad \text{dB}$$



- Traditional assumptions about signal and noise **do not hold**:
  - photon noise is **not independent** of the signal,
  - photon noise is **not Gaussian**,
  - photon noise is **not additive**.
- For bright signals, where  $\rho T$  exceeds  $10^5$ , the noise fluctuations due to photon statistics can be ignored if the sensor has a sufficiently high saturation level.

If a pixel can be thought of as a well of electrons, saturation refers to the condition where the well becomes filled. The amount of charge that can be accumulated in a single pixel is determined largely by its area. However, due to the nature of the potential well, which holds charge within a pixel, there is less probability of trapping an electron within a well that is approaching saturation.



- Electrons can be freed from the CCD through **thermal vibration**.
- These freed electron are **indistinguishable** from true photoelectrons.
- Can be caused by quantum effects (i.e., quantum tunnelling)
- By cooling the CCD chip it is possible to significantly reduce the number of **thermal electrons** causing thermal noise or dark current.

Noise increases exponentially with temperature.



- A relatively small electric current flowing through photosensitive devices even when no photons enter the device.
- It consists of the charges generated in the detector when no outside radiation is entering the detector.
- It is referred to as reverse bias leakage current in non-optical devices and is present in all diodes.
- Physically, dark current is due to the random generation of electrons and holes within the depletion region of the device.



- Increasing the exposure<sup>1</sup> increases the number of thermal electrons.
- The probability distribution of thermal electrons is also a **Poisson process** where the rate parameter is an increasing function of temperature.
- There are techniques for **suppressing dark current** such as estimating the average dark current.
  - i.e., for the given exposure and then subtracting this value from the CCD pixel values before the A/D converter.
- While this does reduce the dark current average (i.e.,  $\mu$ ), it does not reduce the dark current standard deviation ( $\sigma$ ).
  - This also reduces the possible dynamic range of the signal.

---

<sup>1</sup>The amount of light entering the camera sensor.



- This noise originates from reading the signal from the sensor, in this case through the field effect transistor (FET) of a CCD chip.
- The general form of the power spectral density of readout noise is:

$$S_{\text{nm}}(\omega) \approx \begin{cases} \omega^{-\beta} & \omega < \omega_{\min} \quad \beta > 0, \\ k & \omega_{\min} < \omega < \omega_{\max}, \\ \omega^{\alpha} & \omega > \omega_{\max} \quad \alpha > 0. \end{cases}$$

where  $\alpha$ ,  $\beta$  are constants, and  $\omega$  is the radial frequency at which the signal is transferred.

- At very low readout rates ( $\omega < \omega_{\min}$ ) the noise is pink.
- Readout noise can be reduced to manageable levels by appropriate readout rates and proper electronics. At very low signal levels, readout noise can still become a significant component in the overall SNR.



- In the case of CCD sensors, all the pixels in a sensor pass through a common architecture and are essentially subject to the same sources of noise during the readout process.
- Therefore, the read noise of a CCD sensor can be described by a single readout noise value.
- The readout process for sCMOS however is different, sCMOS sensors are often referred to as **Active Pixel Sensors (APS)** since each pixel has its own amplifier circuit.



## Factors

- Read noise is inherent to the readout process of the sensor itself,
  - but cameras from different manufacturers that are based around the same sensor can have some significant differences in how the sensor has been implemented.
- sCMOS cameras are remarkably flexible imaging devices and have several settings that allow them to be optimised for different applications such as high speed, or for high dynamic range imaging.
- Different cameras will have different behaviours as settings are adjusted and this also includes how read noise is affected.



The process of a CMOS camera is as follows:

- Photons hit the sensor and generate charge (electrons),
- The photo-generated charge is converted to an analog voltage for each pixel amplifier,
- These pixel voltages are transferred to the column bus via a row select signal,
- The analog voltages are then **converted to digital signals** via columns of analog to digital (A/D) converters,
- The final digitised signals are then read out **sequentially** at a pixel readout speed which normally can be set at different speeds,



- Noise associated with the gate capacitor of an FET (Field Effect Transistor) is known as **Johnson-Nyquist** noise and can be non-negligible.
- The output RMS value of the noise voltage ( $v_{JN}$ ) is given by:

$$v_{JN} = \sqrt{kT/C} \quad V.$$

where  $C$  is gate switch capacitance,  $k_B$  is Boltzmann's constant, and  $T$  is the absolute temperature of the CCD chip in K.

Capacitance	Noise	Capacitance	Noise
1 fF	2 mV	10 pF	20 $\mu$ V
10 fF	640 $\mu$ V	100 pF	6.4 $\mu$ V
100 fF	200 $\mu$ V	1 nF	2 $\mu$ V

**Table 5:** Effect on the noise on the capacitor value.



- This happens in the amplitude quantization, occurring in the ADC<sup>2</sup>.
- The noise is **additive** and **independent** of the signal when the number of levels ( $L$ ) is less than  $2^4$  where  $B = 4$ .
- When a signal is converted to electricity, it has a minimum and maximum electrical value, due to quantisation.
- If the ADC is adjusted so that 0 corresponds to the minimum electrical value and  $2B - 1$  is the maximum value, SNR is:

$$SNR = 6B + 11 \text{ dB}$$

- For  $B \geq 8$  bits, this means a  $SNR \geq 59$  dB.
- This can usually be ignored as the total SNR is typically dominated by the smallest SNR. In CCD cameras this is **photon noise**.

---

<sup>2</sup>Analog to Digital Converter



- Perlin noise is a type of gradient noise developed by Ken Perlin in 1983.
- The need arose from the machine-looking textures used in CGI in the 80's.
- It has many uses, including but not limited to:
  - procedure generating terrain, applying pseudo-random changes to a variable, and assisting in the creation of image textures.
- It is most commonly implemented in two, three, or four dimensions, but can be defined for any number of dimensions.
- Used frequently to generate textures with extremely **limited** memory.
- Similar methods exists such as **fractal noise** and **simplex noise**



- First thing is to load all the necessary modules.
- The only novel module you may have not encountered is `noise`
  - A library including native-code implementations of Perlin “improved” noise and Perlin simplex noise.

```
import noise
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

- The only module worth mentioning is `PIL` which is an image library for python.



- Let's have a look at the parameters we can play with:

```
shape = (256,256)
scale = 200
octaves = 6
persistence = 0.5
lacunarity = 2.0
seed = 51
```

**shape** Dimensions of the image,

**scale** altitude in which to see the noise

**octaves** number of layers of the algorithm

**persistence** how much more each successive value brings

**lacunarity** level of detail per pass

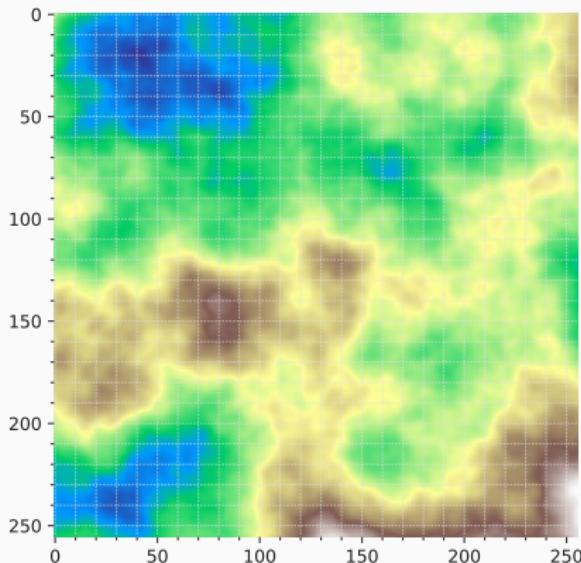
**seed** the initial value for the RNG.



Below is the main function for the perlin noise generation.

```
world = np.zeros(shape)
for i in range(shape[0]):
    for j in range(shape[1]):
        world[i][j] = noise.pnoise2(i/scale, j/scale,
                                      octaves=octaves,
                                      persistence=persistence,
                                      lacunarity=lacunarity,
                                      repeatx=repeatx, repeaty=repeaty,
                                      base=seed)
```

The above function generate the noise from calling the `pnoise2` function from the `noise` library and writes the results to the numpy array `world`.



**Figure 84:** 2D Perlin Noise plot. To give the aesthetics of a map, a colour map `cmap='terrain'` was used



- While this map is quite good we can make it more pop by introducing the height to the image to create a 3D plot.
- For plotting this in 3 dimensions, we must initialise 2 more arrays which will contain the x-y co-ordinates of our world.

```
from mpl_toolkits.mplot3d import axes3d
```

```
lin_x = np.linspace(0,1,shape[0],endpoint=False)
lin_y = np.linspace(0,1,shape[1],endpoint=False)
x,y = np.meshgrid(lin_x,lin_y)
```



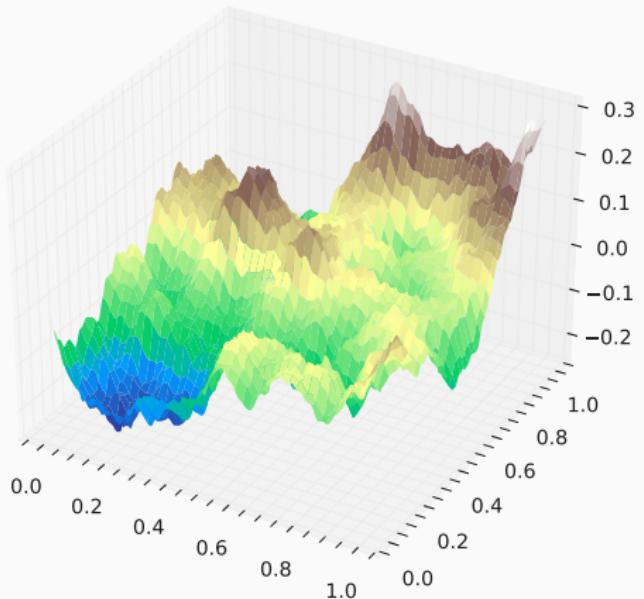
- Now we only need to write the code which will plot this 3D data.

```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.plot_surface(x,y,world,cmap='terrain')

for spine in ax.spines.values():
    spine.set_visible(False)

cp.store_fig("perlin-plot-3d-map",
            close = True)
```

- We use `cmap = 'terrain'` to make it look like a topological map.
- We also disable the axis with `spine.set_visible(False)`.



**Figure 85:** Our newly generated 3D terrain based on Perlin noise.

# Histogram Operations

---



# Table of Contents

Contrast Stretching

## Threshold Operations

Applying a Threshold

Bimodal Histogram

Otsu Threshold

Local Thresholding

## Channel Operations

RGB to Grayscale

Histogram Matching

## DIP Applications

Studying Metals

Steganography



- Frequently, an image is scanned in such a way that the resulting brightness values do not make **full use of the available dynamic range**.
- By stretching the histogram over the available dynamic range we can attempt to correct this range under-use.
- If the image is intended to go from brightness 0 to brightness  $2^B - 1$ , then one generally maps the 0 value to the value 0 and the 100 value to the value  $2^B - 1$ .
- The appropriate transformation is given by:

$$b[m, n] = (2^B - 1) \frac{a[m, n] - \text{minimum}}{\text{maximum} - \text{minimum}}.$$

- This formula, can sensitive to outliers and a less sensitive.



- A more general description is given by:

$$b[m, n] = \begin{cases} 0 & a[m, n] \leq p_{lo}\%, \\ (2^B - 1) \frac{a[m, n] - p_{lo}\%}{p_{hi}\% - p_{lo}\%} & p_{lo}\% < a[m, n] < p_{hi}\%, \\ (2^B - 1) & a[m, n] \geq p_{hi}\%. \end{cases}$$

- In this version one might choose the 1% and 99% values for  $p_{lo}\%$  and  $p_{hi}\%$ , respectively, instead of the 0% and 100% values.
- It is also possible to apply the contrast-stretching operation on a regional basis using the histogram from a region to determine the appropriate limits for the algorithm.
- It is possible to suppress the term  $2^B - 1$  and simply normalize the brightness range to  $0 \leq b[m, n] \leq 1$ .
- This means representing the final pixel brightnesses as reals instead of integers but modern computers can handle this with ease.



- Before we compare two images, it is always a good practice to normalise their histograms to a **standard** histogram.
- This is useful when images were retrieved from different sources.
- The most common technique is **histogram equalisation** where one attempts to change the histogram through the use of a function  $b = f(a)$  into a histogram that is constant for all brightness values.
- This would correspond to a brightness distribution where all values are equally probable.
- Unfortunately, for an arbitrary image, one can only approximate this result.



- For a suitable function  $f(\cdot)$  the relation between the input probability density function, the output probability density function, and the function  $f(\cdot)$  is given by:

$$p_b(b) db = p_a(a) da \rightarrow \frac{p_a(a) da}{p_b(b)}.$$

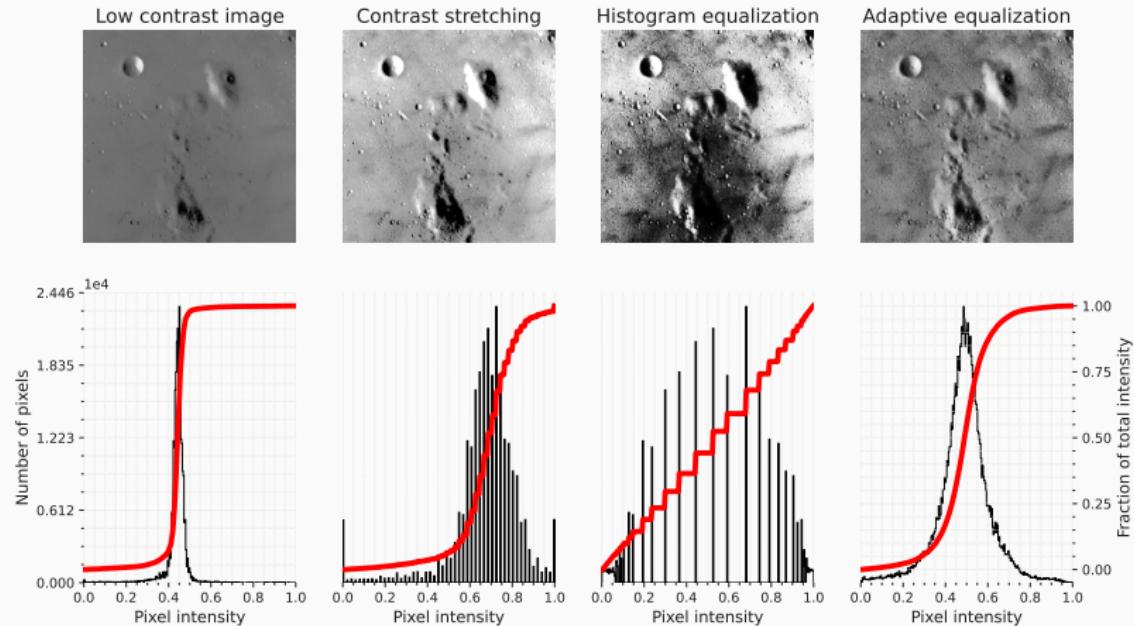
- Here we see that suitable means  $f(\cdot)$  is differentiable and that  $df/da \geq 0$ . For histogram equalization we desire that  $p_b(b) = \text{constant}$  and this means that:

$$f(a) = (2^B - 1) \cdot P(a)$$

where  $P(a)$  is the probability distribution function.

- In other words, the quantized probability distribution function normalized from 0 to  $2^B - 1$  is the look-up table required for histogram equalization.

# Histogram Operations



**Figure 86:** Different Types of contrast stretching methods.



# Histogram Operations

- Thresholding is used to create a binary image from a grayscale image.
- It is the simplest way to segment objects from a background.
- This study will use the `skimage` module which offers it in two (2) ways:
  - Histogram-based. The histogram of the pixels' intensity is used and certain assumptions are made on the properties of this histogram (e.g. bimodal).
  - Local. To process a pixel, only the neighboring pixels are used. These algorithms often require more computation time.



# Histogram Operations

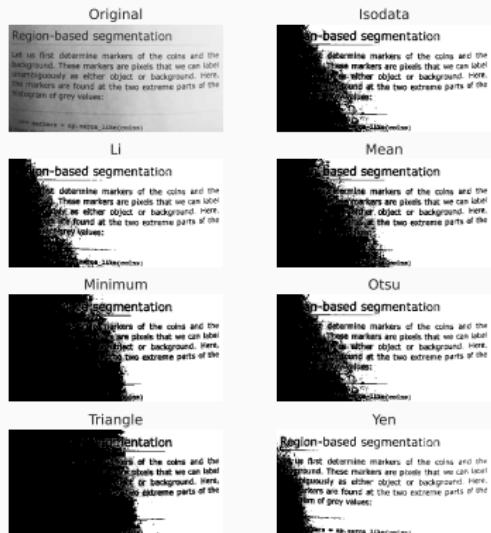


Figure 87



- illustrate how to apply one of these thresholding algorithms. This example uses the mean value of pixel intensities. It is a simple and naive threshold value, which is sometimes used as a guess value.

# Histogram Operations



```
from skimage.filters import threshold_mean

image = data.camera()
thresh = threshold_mean(image)
binary = image > thresh

fig, axes = plt.subplots(ncols=2, figsize=(8, 3))
ax = axes.ravel()

ax[0].imshow(image, cmap=plt.cm.gray)
ax[0].set_title('Original image')

ax[1].imshow(binary, cmap=plt.cm.gray)
ax[1].set_title('Result')

for a in ax:
    a.set_axis_off()

cp.store_fig("threshold-mean", close=True)
```

# Histogram Operations

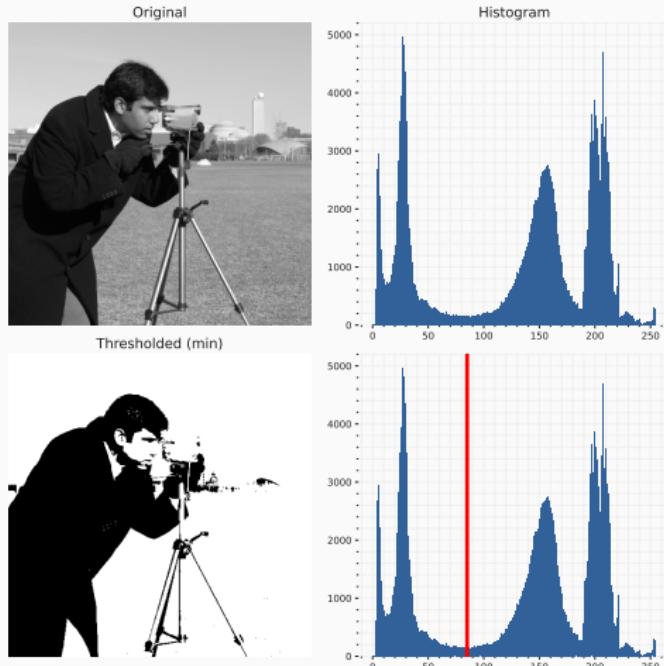


Figure 88



# Histogram Operations

- Named after Nobuyuki Otsu, is used to perform automatic image thresholding [58].
- In the simplest form, it returns a **single intensity threshold** separating pixels into two (2) classes:
  1. foreground,
  2. background,

Threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance [45].



- If background is relatively uniform, use a global threshold value.
  - such as Otsu's method.
- However, if there is large variation in the background intensity, adaptive thresholding (i.e., local or dynamic thresholding) can produce better results.

Note that local is much slower than global thresholding.

Here, we binarize the image using the `threshold_local` function, which calculates thresholds in regions with a characteristic size `block_size` surrounding each pixel. Each threshold value is the weighted mean of the local neighbourhood minus an offset value.

# Histogram Operations



```
from skimage.filters import threshold_otsu, threshold_local

image = data.page()
global_thresh = threshold_otsu(image)
binary_global = image > global_thresh
block_size = 35
local_thresh = threshold_local(image, block_size, offset=10)
binary_local = image > local_thresh
fig, axes = plt.subplots(ncols=3, figsize=(7, 8))
ax = axes.ravel()
ax[0].imshow(image)
ax[0].set_title('Original')
ax[1].imshow(binary_global)
ax[1].set_title('Global thresholding')
ax[2].imshow(binary_local)
ax[2].set_title('Local thresholding')

for a in ax:
    a.set_axis_off()

cp.store_fig("local-threshold", close=True)
```

# Histogram Operations



## Original

### Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.summe_like(coins)
```

## Global thresholding

### Region-based segmentation

Determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.summe_like(coins)
```

## Local thresholding

### Region-based segmentation

Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

```
>>> markers = np.summe_like(coins)
```

# Histogram Operations

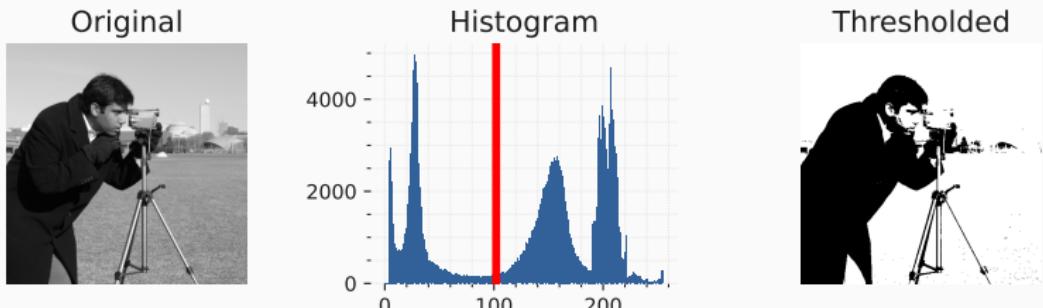


Figure 90

# Histogram Operations



**Figure 91:** A comparison of the original image with the one with a mean threshold applied.



- This example converts an image with RGB channels into an image with a single grayscale channel.
- The value of each grayscale pixel is calculated as the weighted sum of the corresponding red, green and blue pixels as:

$$Y = 0.2125 R + 0.7154 G + 0.0721 B$$

These weights are used by CRT phosphors as they better represent human perception of red, green and blue than equal weights.

# Histogram Operations



```
#+NAME: CHANNEL-A
#+begin_src python :session :results output
import matplotlib.pyplot as plt

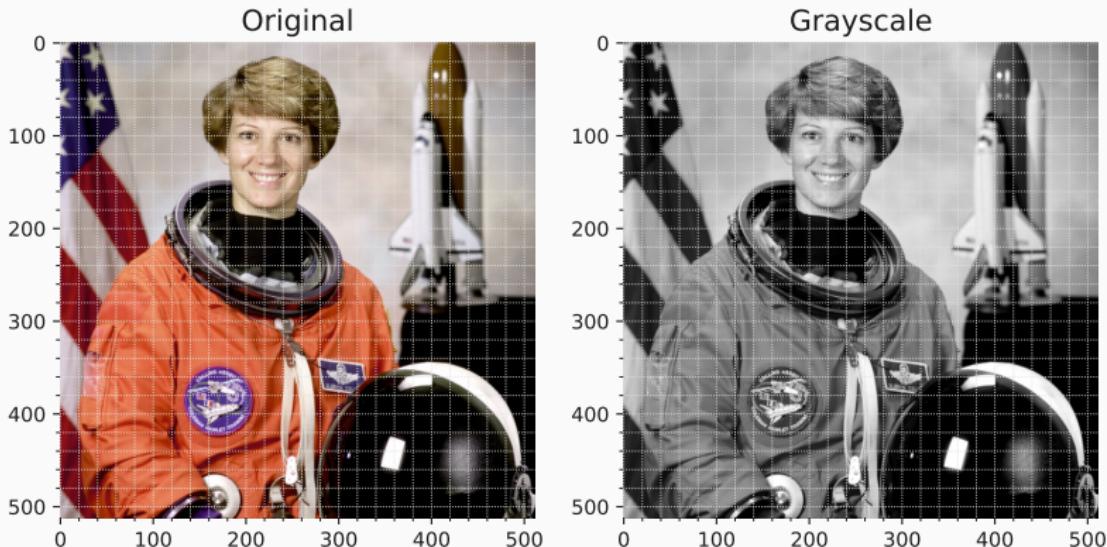
from skimage import data
from skimage.color import rgb2gray

original = data.astronaut()
grayscale = rgb2gray(original)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].imshow(original)
ax[0].set_title("Original")
ax[1].imshow(grayscale, cmap=plt.cm.gray)
```

# Histogram Operations



**Figure 92:** A comparison of an RGB image (left) v. its grey-scale version (right).



- Histogram matching manipulates the pixels of an input image so that its histogram matches the histogram of the reference image.
- If the images have multiple channels, the matching is done independently for each channel, as long as the number of channels is equal in the input image and the reference.
- Histogram matching can be used as a lightweight normalisation for image processing, such as feature matching, especially in circumstances where the images have been taken from different sources or in different conditions (i.e. lighting).

# Histogram Operations



```
#+NAME: HISTOGRAM-MATCHING
#+begin_src python :session :results output
import matplotlib.pyplot as plt
from skimage import data, exposure
from skimage.exposure import match_histograms

reference,image = data.coffee(), data.chelsea()
matched = match_histograms(image, reference, channel_axis=-1)

fig, (ax1, ax2, ax3) = plt.subplots(
    nrows=1, ncols=3, figsize=(8, 3), sharex=True, sharey=True
)
for aa in (ax1, ax2, ax3):
    aa.set_axis_off()

ax1.imshow(image); ax1.set_title('Source')
ax2.imshow(reference); ax2.set_title('Reference')
ax3.imshow(matched); ax3.set_title('Matched')
```

# Histogram Operations

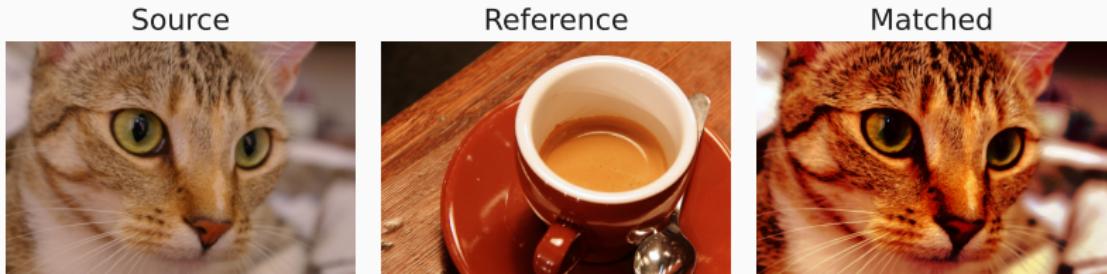


Figure 93



- Here, we identify and track the solid-liquid (S-L) interface in a nickel-based alloy undergoing solidification.
- Tracking the solidification over time enables the calculation of the solidification velocity.
- This is important to characterise the solidified structure of the sample and will be used to inform research into additive manufacturing of metals.

The image sequence was obtained by the Center for Advanced Non-Ferrous Structural Alloys (CANFSA) using synchrotron x-radiography at the Advanced Photon Source (APS) at Argonne National Laboratory (ANL).

# Histogram Operations



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.io

from skimage import filters, measure, restoration
from skimage.data import nickel_solidification

image_sequence = nickel_solidification()
y0, y1, x0, x1 = 0, 180, 100, 330
image_sequence = image_sequence[:, y0:y1, x0:x1]

print(f'shape: {image_sequence.shape}')
```

```
shape: (11, 180, 230)
```



## Compute Image Deltas

- The dataset is a 2D image stack with 11 frames (time points).
- We visualize and analyze it in a workflow where the first image processing steps are performed on the entire three-dimensional dataset (i.e., across space and time),
- the removal of localized, transient noise is favored as opposed to that of physical features (e.g., bubbles, splatters, etc.), which exist in roughly the same position from one frame to the next.

```
fig = px.imshow(  
    image_sequence,  
    animation_frame=0,  
    binary_string=True,  
    labels={'animation_frame': 'time point'},  
)  
plotly.io.show(fig)
```

# Histogram Operations

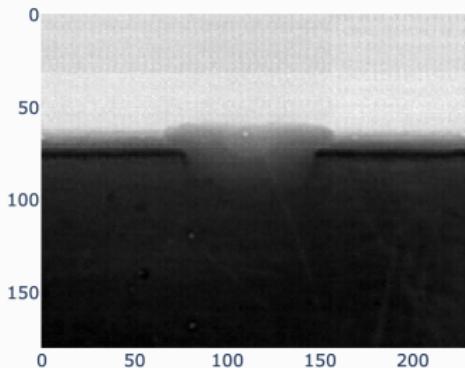


Figure 94



## Compute image deltas

- First, apply a low-pass to smooth the images and reduce noise.
- Next, compute the image deltas.
- To do this, subtract the image sequence ending at the second-to-last frame from the image sequence starting at the second frame.

```
#+NAME: METAL-C
#+begin_src python :session :results output
smoothed = filters.gaussian(image_sequence)
image_deltas = smoothed[1:, :, :] - smoothed[:-1, :, :]

fig = px.imshow(
    image_deltas,
    animation_frame=0,
    binary_string=True,
    labels={'animation_frame': 'time point'},
```

# Histogram Operations

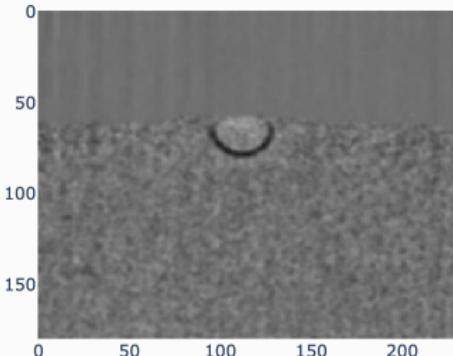


Figure 95



## Clip Lowest and Highest intensities

- We now calculate the 5th and 95th percentile intensities of `image_deltas`,
- We want to clip the intensities which lie below the 5th percentile intensity and above the 95th percentile intensity, while also rescaling the intensity values to [0, 1].

# Histogram Operations



```
#+NAME: METAL-D
#+begin_src python :session :results output
p_low, p_high = np.percentile(image_deltas, [5, 95])
clipped = image_deltas - p_low
clipped[clipped < 0.0] = 0.0
clipped = clipped / p_high
clipped[clipped > 1.0] = 1.0

fig = px.imshow(
    clipped,
    animation_frame=0,
    binary_string=True,
    labels={'animation_frame': 'time point'},
```

# Histogram Operations

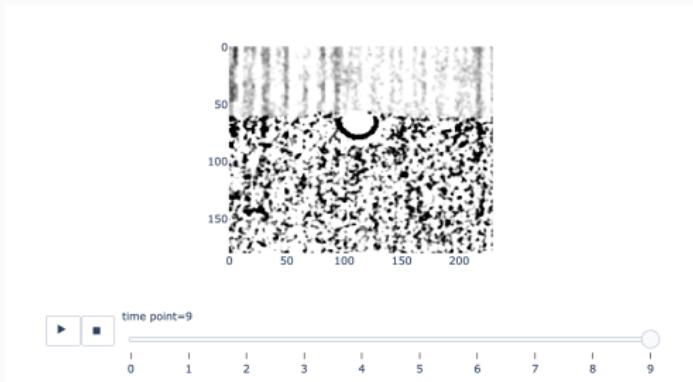


Figure 96



## Invert and De-noise

- We invert the `clipped` images so the regions of highest intensity will include the region we are interested in tracking (i.e., the S-L interface). Then, we apply a total variation denoising filter to reduce noise beyond the interface.

```
#+NAME: METAL-E
#+begin_src python :session :results output
inverted = 1 - clipped
denoised = restoration.denoise_tv_chambolle(inverted, weight=0.15)

fig = px.imshow(
    denoised,
    animation_frame=0,
    binary_string=True,
    labels={'animation_frame': 'time point'},
```

# Histogram Operations

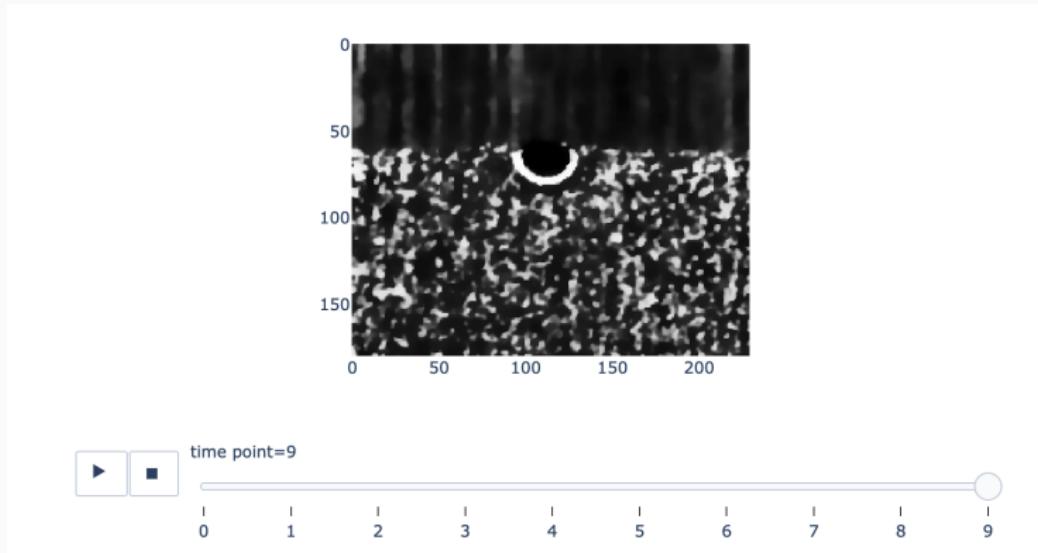


Figure 97



## Binarise

- Our next step is to create binary images, splitting the images into foreground and background.
- We want the S-L interface to be the most prominent feature in the foreground of each binary image, so that it can eventually be separated from the rest of the image.
- We need a threshold value `thresh_val` to create our binary images, binarized.

This can be set manually, but we shall use an automated minimum threshold method from the filters submodule of scikit-image.



## Select the Largest Region

- The S-L interface appears as the **largest region** of connected pixels.
- For this step of the workflow, operate on each 2D image **separately**, as opposed to the entire 3D dataset.

We are only interested in a single moment in time for each region.



- Apply `skimage.measure.label()` on the binary images so each region has its own label.
- Then, select the **largest region** in each image by computing region properties:
  - including the area property,
  - sorting by area values
- `skimage.measure.regionprops_table()` returns a table of region properties which can be read into a Pandas `DataFrame`.
- To begin with, consider the first image delta at this stage of the workflow, `binarized[0, :, :]`.

# Histogram Operations



```
#+NAME: METAL-G
#+begin_src python :session :results output
labeled_0 = measure.label(binarized[0, :, :])
props_0 = measure.regionprops_table(labeled_0,
                                    properties=('label', 'area',
                                                'bbox'))
props_0_df = pd.DataFrame(props_0)
props_0_df = props_0_df.sort_values('area', ascending=False)
```

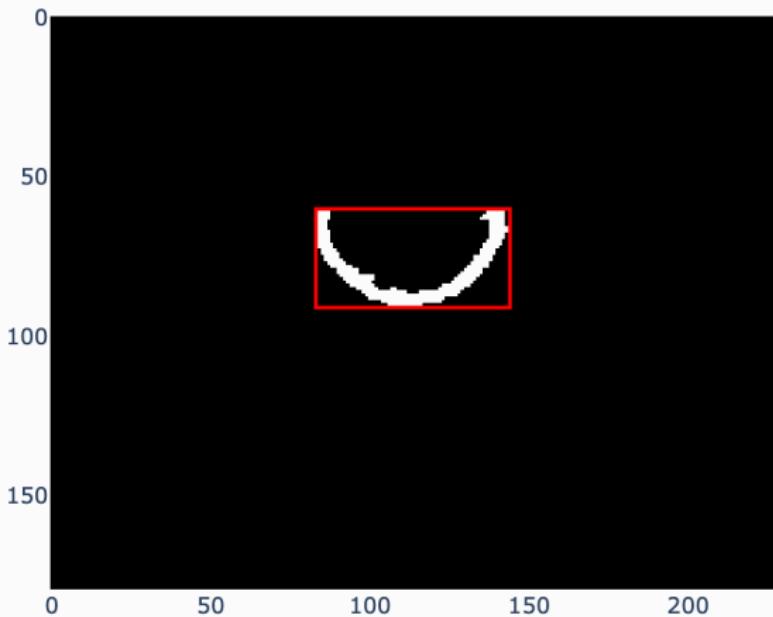
```
#+RESULTS: METAL-G
#+begin_example
label    area   bbox-0   bbox-1   bbox-2   bbox-3
1        2     417.0     60       83       91      144
198     199    235.0    141      141      179      165
11       12    136.0     62       164      87      180
```



- We can then select the largest region by matching its label with the one found in the first row of the **sorted** table.
- Let us visualize it, along with its bounding box ( `bbox` ) in red.

```
#+NAME: METAL-H
#+begin_src python :session :results output
largest_region_0 = labeled_0 == props_0_df.iloc[0]['label']
minr, minc, maxr, maxc = (props_0_df.iloc[0][f'bbox-{i}'] for i in
↪ range(4))
fig = px.imshow(largest_region_0, binary_string=True)
```

# Histogram Operations



**Figure 98:** Showcasing the bounding box over the largest connected pixel group.

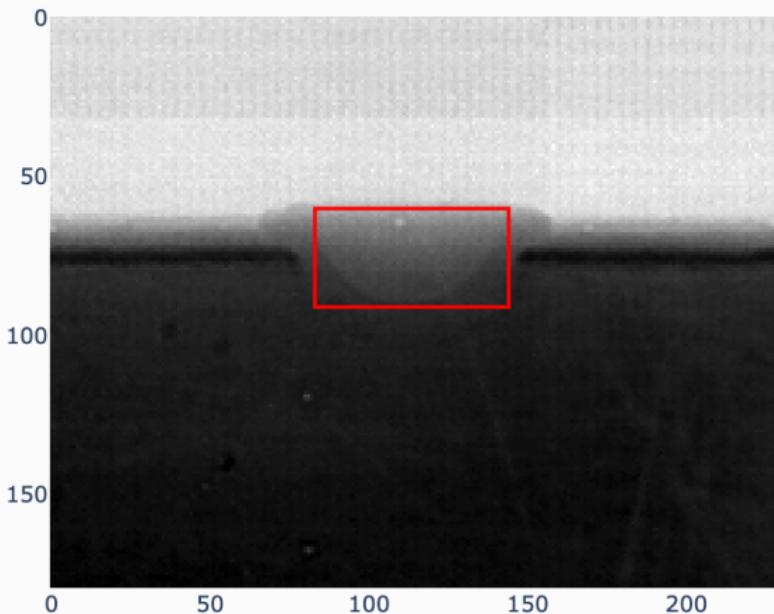


- We can see how the lower bounds of the box align with the bottom of the S-L interface by overlaying the same bounding box onto the 0th raw image.
- This bounding box was calculated from the image delta between the 0th and 1st images, but the bottom-most region of the box corresponds to the location of the interface earlier in time (0th image) because the interface is moving upward.

```
#+NAME: METAL-I  
#+begin_src python :session :results output  
fig = px.imshow(image_sequence[0, :, :], binary_string=True)
```



# Histogram Operations



**Figure 99:** The bounding box applied to an original frame instead of a delta.

# Histogram Operations

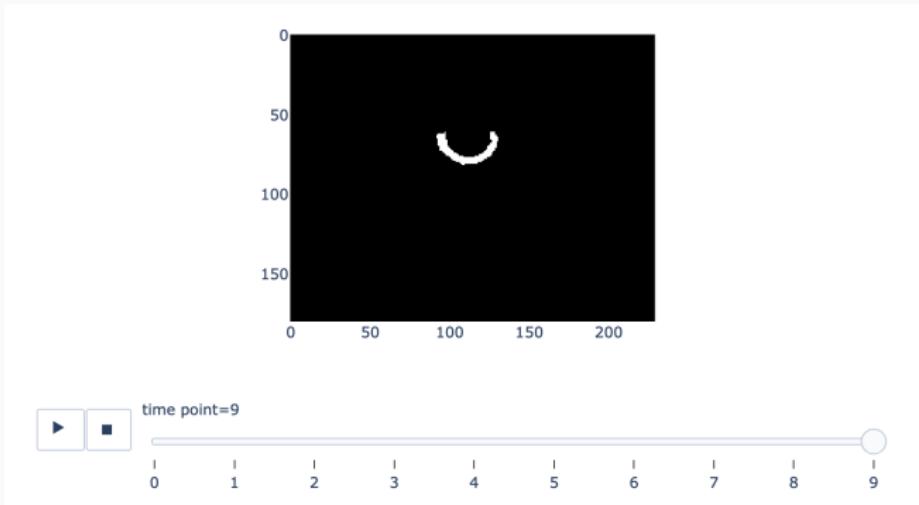


Figure 100



## Plot S-L v. time

- The final step is to plot the location of the **S-L interfaces** over time.
- This can be achieved by plotting `maxr` (third element in `bbox`) over time as this value shows the  $y$  location of the bottom of the interface.
- The pixel size in this experiment was 1.93 microns and the frame-rate was 80,000 frames per second, so these values are used to convert pixels and image number to physical units.
- We calculate the average solidification velocity by fitting a linear polynomial to the scatter plot.
- The velocity is the first-order coefficient.

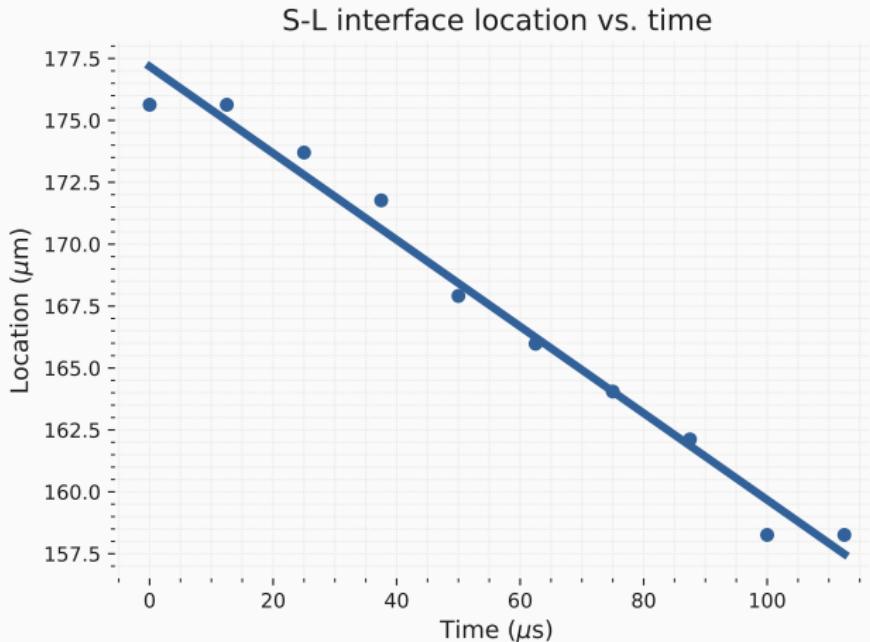
# Histogram Operations



```
#+NAME: METAL-J
#+begin_src python :session :results output
largest_region = np.empty_like(binarized)
bboxes = []

for i in range(binarized.shape[0]):
    labeled = measure.label(binarized[i, :, :])
    props = measure.regionprops_table(labeled, properties=('label',
        ↪ 'area', 'bbox'))
    props_df = pd.DataFrame(props)
    props_df = props_df.sort_values('area', ascending=False)
    largest_region[i, :, :] = labeled == props_df.iloc[0]['label']
    bboxes.append([props_df.iloc[0][f'bbox-{i}'] for i in
        ↪ range(4)])
fig = px.imshow(
    largest_region,
    animation_frame=0,
    binary_string=True,
    labels={'animation_frame': 'time point'},
)
)
```

# Histogram Operations

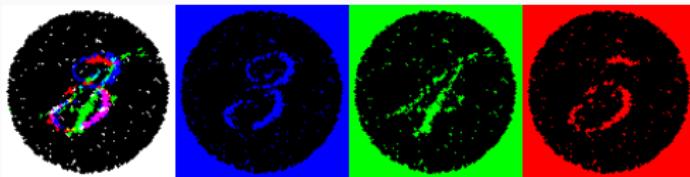


**Figure 101:** Using the image alone, it was possible to determine the metals S-L state transition.



Steganography is the practice of concealing a message within another message or a physical object.

- Today, we are going to do an implementation.
- We will make a program which writes text to images and reads it.
- We assign pixel values to each possible character.
- This is a proof of concept but anyone is welcome to improve on it.



**Figure 102:** The same image viewed by white, blue, green, and red lights reveals different hidden numbers.

# Histogram Operations



```
from PIL import Image # For reading images
import string # constants containing letters and digits.
```

```
# Open the image.
image = Image.open("Fruit.jpg")

# Get the image size and save it.
width, height = image.size
startingPixel = (10, 10, 255)
endingPixel = (255, 20, 20)
# Dictionary which hold the relations between pixels and characters
lettersToPixels = {}
pixelsToLetters = {}
# Making the relations using the string library
for i, lttr in enumerate(string.ascii_letters + string.digits + ' '
                           ''):
    lettersToPixels[lttr] = i
    pixelsToLetters[i] = lttr
```



- Let's get to hiding messages in images.
  - If the mode specified by the user is write, user wants to write.
- But we also check if the text was set.
- After that, assert if the text length is greater than the image width.
- Then, draw the starting pixel onto the image at position (0, 0), which is the top left.
- Loop until the message is written.

# Histogram Operations



```
def write(message):
    assert len(message) < width
    # Draw the starting Pixel
    image.putpixel((0, 0), startingPixel)

    # Draw a pixel for each letter in the test
    for index, letter in enumerate(message):

        # The Middle value (g = green) is the number in the
        # dictionary
        image.putpixel((index+1, 0), (11, lettersToPixels[letter],
        # 11))

        # Draw the ending Pixel
        image.putpixel((index+2, 0), endingPixel)

        # Save the image to the same place
        image.save("images/Histogram-Operations/encrypted-fruit.png")

write("hello")
```

# Histogram Operations



**Figure 103:** Encrypted image. Can you spot where the information is hidden?

# Histogram Operations



# Histogram Operations



```
def read():
    image =
        → Image.open("images/Histogram-Operations/encrypted-fruit.png")
    text = ''
    index = 1
    while True:
        # Loop through each pixel in the top row
        pixel = image.getpixel((index, 0))
        # if the pixel is the ending pixel, we break the loop
        if pixel == endingPixel:
            break
        try:
            # Get the letter from the dictionary with the g value.
            text += pixelsToLetters[pixel[1]]
        except:
            pass
        index += 1
        # Print out the text
    print(text)
read()
```

# Histogram Operations



hello

# Morphological Operations

---



# Table of Contents

## Introduction

Broad Definition

## Morphological Operations

Erosion

Dilation

Opening

Closing

Gradient

Top Hat

## Application

Logistic Mapping

Apply Segmented Rim as Mask



- In many areas of knowledge morphology deals with **form and structure**:
  - biology,
  - linguistics,
  - social studies,
- Mathematical morphology (which in turn in DIP) deals with set theory.
- Sets in Mathematical Morphology represents objects in an image.



# Morphological Operations

- Used to extract image components that are useful in the representation and description of region shape, such as:
  - Boundaries extraction;
  - Skeletons;
  - Convex hull;
  - Morphological filtering
  - Thinning
  - Pruning



- The applications of mathematical in DIP are:
  - Pre-processing:
  - Enhancing object structure
  - Segmentation
  - Quantitative description



- Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image.
- These operations rely only on the relative ordering of pixel values, not on their numerical values.
  - Therefore are especially suited to the processing of binary images.

Morphological operations can also be applied to greyscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest.

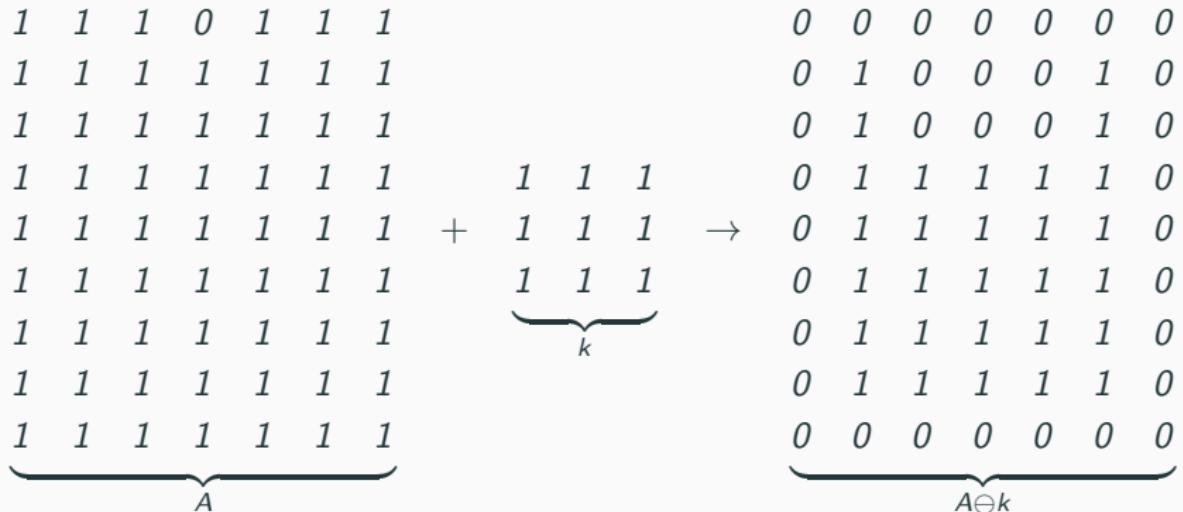


- The idea of erosion is similar to soil erosion
  - i.e., erodes away the boundaries of foreground object.
- The **kernel** slides through the image (as in 2D convolution).
- A pixel in the original image (either `1` or `0`) will be considered `1` only if all the pixels under the kernel is `1`, otherwise it is `0`.
- All pixels near boundary will be **discarded** based on kernel size.

Thickness or size of the foreground object decreases or simply white region decreases in the image.

- Used in removing small white noises, detaching connected objects ...
- Represented as  $A \ominus k$  where  $A$  is the image and  $k$  is the kernel.

# Morphological Operations



$$(A \ominus k)(x, y) = \min_{(i, j) \in k} f(x + i, y + j)$$

Figure 104: A visual representation on how the erosion process works on a 2D array on a binary image.

# Morphological Operations



(a)



(b)

**Figure 105:** The original image (a) and Erosion with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 106:** The original image (a) and Erosion with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 107:** The original image (a) and Erosion with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 108:** The original image (a) and Erosion with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 109:** The original image (a) and Erosion with 21-by-21 kernel applied (b).

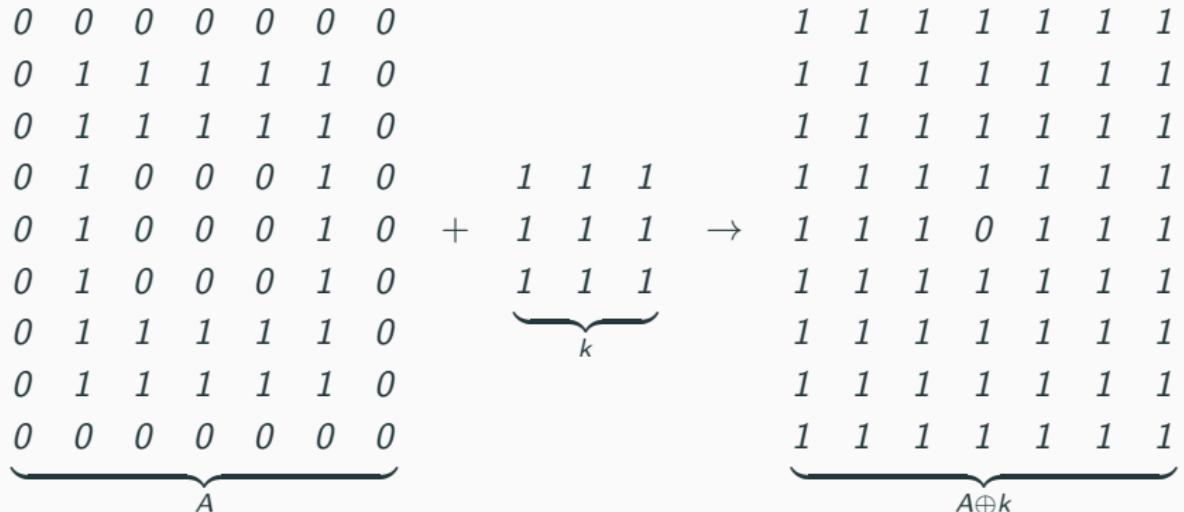


- Can be defined as the **opposite of erosion**.
- Here, a pixel element is 1 if at least one pixel under the kernel is 1.
- It increases the white region in the image or size of foreground object increases.

Normally, in cases like noise removal, erosion is followed by dilation as erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases.

- It is also useful in joining broken parts of an object.
- Mathematically it is represented as  $A \oplus k$  where  $A$  is the image and  $k$  is the kernel.

# Erosion Process



$$(A \oplus k)(x, y) = \max_{(i, j) \in k} f(x + i, y + j)$$

Figure 110: A visual representation on how the erosion process works on a 2D array.

# Morphological Operations



(a)



(b)

**Figure 111:** The original image (a) and Dilation with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 112:** The original image (a) and Dilation with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 113:** The original image (a) and Dilation with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 114:** The original image (a) and Dilation with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 115:** The original image (a) and Dilation with 21-by-21 kernel applied (b).



- Defined as **erosion followed by dilation**.

It is useful in removing noise.

- Represented as  $(A \ominus k) \oplus k$  where  $A$  is the image and  $k$  is the kernel.
- Opening removes small objects from the foreground.
  - Usually taken as the bright pixels of an image, placing them in the background

These can also be used to find specific shapes in an image.

- Opening can be used to find things into which a specific structuring element can fit (edges, corners, ...).

# Morphological Operations



(a)



(b)

**Figure 116:** The original image (a) and Opening with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 117:** The original image (a) and Opening with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 118:** The original image (a) and Opening with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 119:** The original image (a) and Opening with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 120: The original image (a) and Opening with 21-by-21 kernel applied (b).



- It is **dilation followed by erosion** and is useful in closing small holes.
- Represented as  $(A \oplus k) \ominus k$  where  $A$  is the image and  $k$  is the kernel.

Closing removes small holes in the foreground, changing small islands of background into foreground.

# Morphological Operations



(a)



(b)

**Figure 121:** The original image (a) and Closing with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 122: The original image (a) and Closing with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 123:** The original image (a) and Closing with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 124:** The original image (a) and Closing with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 125:** The original image (a) and Closing with 21-by-21 kernel applied (b).



- It is the **difference between dilation and erosion** which resembles an outline.
- It is  $(A \oplus k) - (A \ominus k)$  where  $A$  is the image and  $k$  is the kernel.

# Morphological Operations



(a)



(b)

**Figure 126:** The original image (a) and Gradient with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 127:** The original image (a) and Gradient with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 128: The original image (a) and Gradient with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 129: The original image (a) and Gradient with 17-by-17 kernel applied (b).

# Morphological Operations



Figure 130: The original image (a) and Gradient with 21-by-21 kernel applied (b).



- Top-hat extracts small elements and details from given images.
- There exist two types of top-hat transform:
  - White top-hat transform
  - Black top-hat transform
- white top-hat transform is the difference between the input image and its opening by some structuring element.
- black top-hat transform is defined dually as the difference between the closing and the input image.

Top-hat transforms are used for various image processing tasks, such as feature extraction, background equalization, image enhancement, and others

# Morphological Operations



(a)



(b)

Figure 131: The original image (a) and White-Hat with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 132: The original image (a) and White-Hat with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 133: The original image (a) and White-Hat with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 134: The original image (a) and White-Hat with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 135:** The original image (a) and White-Hat with 21-by-21 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 136:** The original image (a) and Black-Hat with 5-by-5 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 137:** The original image (a) and Black-Hat with 9-by-9 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 138: The original image (a) and Black-Hat with 13-by-13 kernel applied (b).

# Morphological Operations



(a)



(b)

Figure 139: The original image (a) and Black-Hat with 17-by-17 kernel applied (b).

# Morphological Operations



(a)



(b)

**Figure 140:** The original image (a) and Black-Hat with 21-by-21 kernel applied (b).



- For businesses dealing with logistics, delivery services, or transportation, understanding and analysing road networks is critical.
- Here, we will demonstrate how morphological operations can help such businesses by focusing on the road networks in maps.
- Let us use a map we are all familiar of:

# Morphological Operations

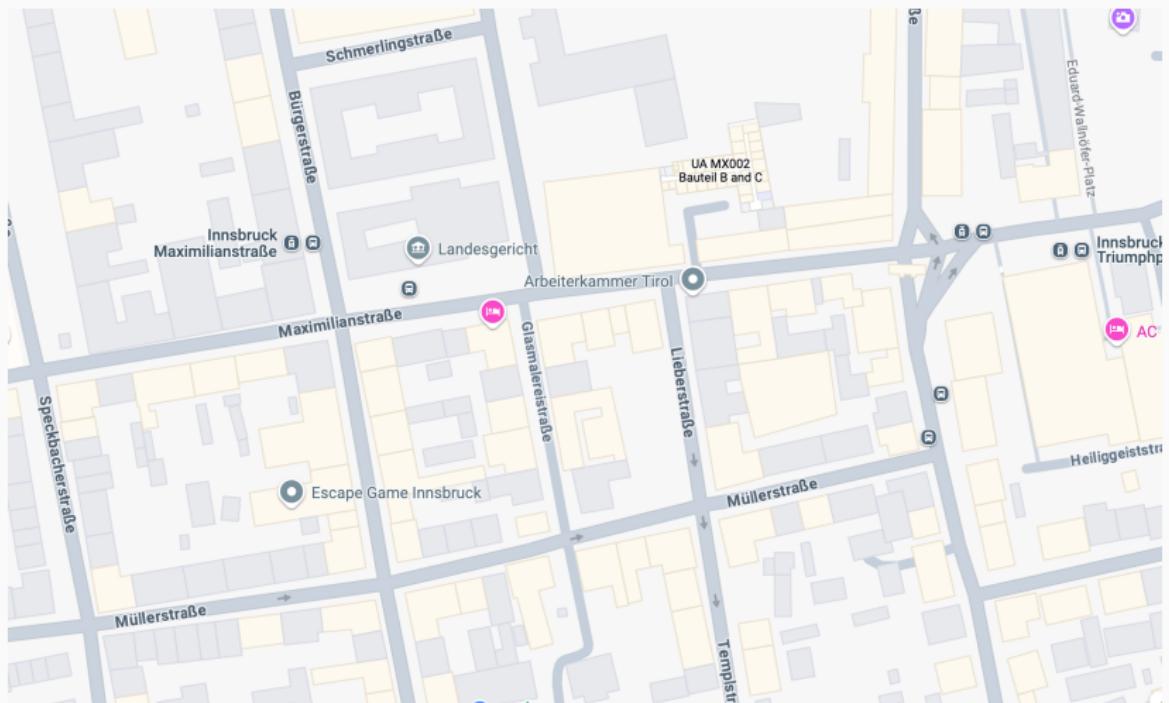


Figure 141: A Google Map view of central Innsbruck.



- We start with a grayscale version of the map image, then apply a manual threshold to binarize it:

transportation, understanding and analyzing road networks is  
→ critical.

Here, we will demonstrate how morphological operations can help  
→ such  
businesses by focusing on the road networks in maps. Let us use the  
**map** below:

We start with a grayscale version of the **map** image,  
then apply a manual threshold to binarise it:

# Morphological Operations

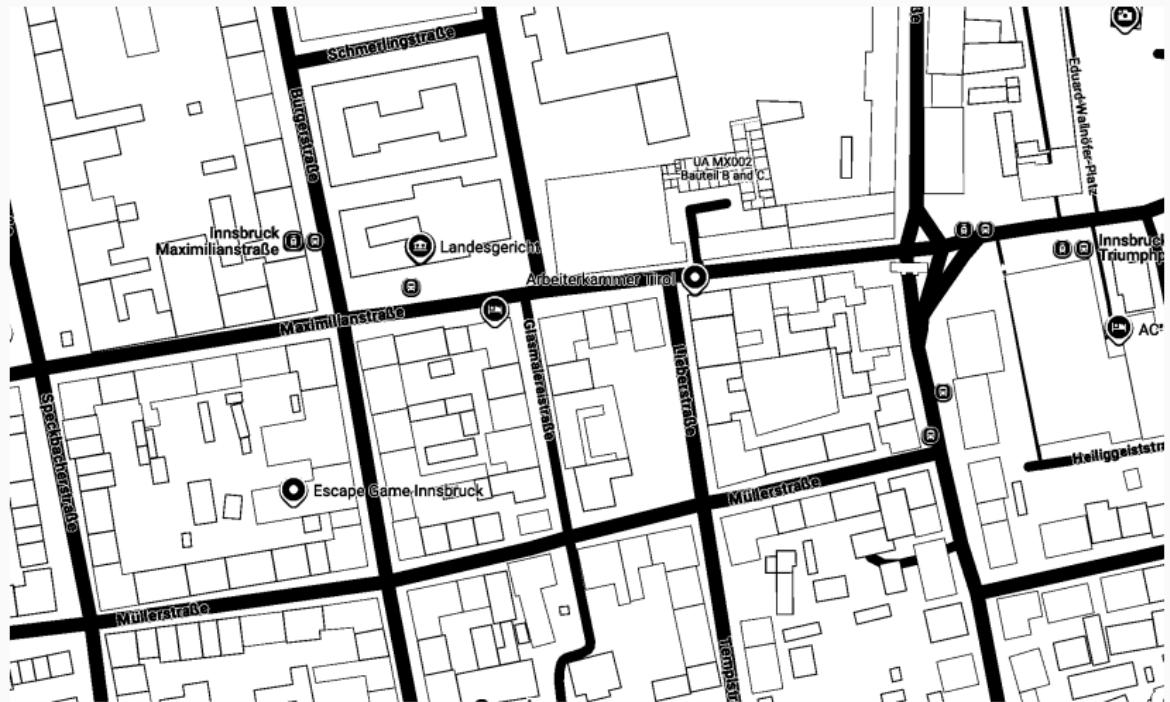


Figure 142: The map of Innsbruck once the binarisation is applied.

# Morphological Operations



- To filter out the roads, we apply a series of morphological operations.
- We first perform opening to remove small patches of white that are not part of the roads.
- Then, we apply closing to fill in gaps within the roads:

```
im_bw = im > 0.90
sk.io.imsave("images/Morphological-Operations/innsbruck-bw.png",im_bw)
#+end_src

To filter out the roads, we apply a series of morphological
→ operations.
We first perform opening to remove small patches of white that are
→ not
part of the roads. Then, we apply closing to fill in gaps within
→ the
roads:
```

# Morphological Operations



Figure 143: Operation of opening and closing are applied.



- The map now has an isolated representation of the road network.
- However, some roads are still not adequately represented.
- To refine this, we apply a sequence of erosion and dilation to further disconnect unrelated roads and widen the main roads:

# Morphological Operations



```
selem_temp = np.array([[-1, -1, -1, -1],  
                      [-1, 8.5, 8.5, -1],  
                      [-1, 8.5, 8.5, -1],  
                      [-1, -1, -1, -1]])  
  
map1 = closing(opening(im_bw, selem_temp), selem_temp)  
sk.io.imsave("images/Morphological-Operations/innsbruck-bw-2.png", map1)  
#+end_src  
  
#+NAME: LOGISTIC-MAPPING-C  
#+begin_src python :session logistic-mapping :results output  
from skimage.morphology import erosion, dilation  
from skimage.draw import disk  
  
def im_dilation(image, selem, n):
```

## Morphological Operations



**Figure 144:** Operation of erosion and dilation are applied.



- Finally, we skeletonize the image to get a single-pixel wide representation of each road, which can help businesses quantify the total length of the road network:

Skeletonization reduces binary objects to 1 pixel wide representations. This can be useful for feature extraction, and/or representing an object's topology.

It works by making successive passes of the image. On each pass, border pixels are identified and removed on the condition that they do not break the connectivity of the corresponding object.

# Morphological Operations



Figure 145: Skelotonise is applied to highlight the road network.



## Measure Fluorescence

- This exercise reproduces a well-established workflow in bioimage data analysis for measuring the fluorescence intensity localized to the nuclear envelope, in a time sequence of cell images.
  - Each with two channels and two spatial dimensions
- This shows a process of protein re-localization from the cytoplasmic area to the nuclear envelope.

# Morphological Operations



- First import the necessary modules:

```
** Application: Measure fluorescence intensity at the nuclear
→ envelope
```

This example reproduces a well-established workflow in bioimage data analysis for measuring the fluorescence intensity localized to the nuclear envelope, in a time sequence of cell images (each with two channels and two spatial dimensions) which shows a process of protein re-localization from the cytoplasmic area to the nuclear envelope.

- We start with a single cell/nucleus to construct the workflow.

```
from scipy import ndimage as ndi

from skimage import filters, measure, morphology, segmentation
```

# Morphological Operations

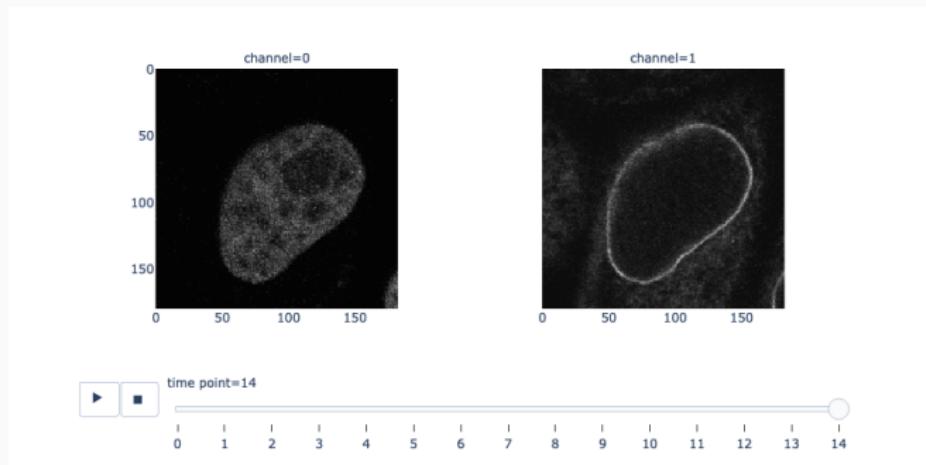


Figure 146

- To begin with, let us consider the first channel of the first image.

```
image_sequence,
```



## Segment the Nucleus Rim

- Let us apply a Gaussian low-pass filter to this image in order to smooth it.
- Next, we segment the nuclei, finding the threshold between the background and foreground with Otsu's method: We get a binary image.
- We then fill the holes in the objects.

```
channel of the first image.
```

```
#+NAME: FLORO-D
#+begin_src python :session :results none
image_t_0_channel_0 = image_sequence[0, 0, :, :]
#+end_src
```



- Following the original workflow, let us remove objects which touch the image border.
- Here, we can see that part of another nucleus was touching the bottom right-hand corner.

We then fill the holes in the objects.



- We compute both the morphological dilation of this binary image and its morphological erosion.

```
fill = ndi.binary_fill_holes(thresh)  
#+end_src
```

- Finally, we subtract the eroded from the dilated to get the nucleus rim.
- This is equivalent to selecting the pixels which are in `dilate`, but not in `erode`:

```
clear.dtype
```

- Let us visualize these processing steps in a sequence of subplots.

# Morphological Operations

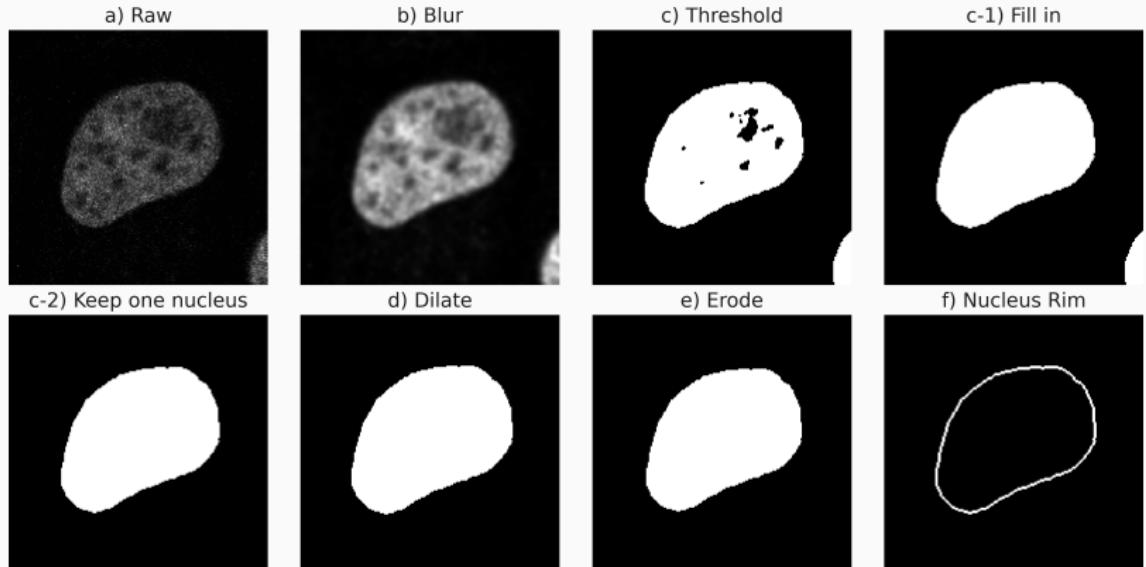


Figure 147: Step-by-step operations done on the original image.

# Morphological Operations



- Now that we have segmented the nuclear membrane in the first channel, we use it as a mask to measure the intensity in the second channel.

```
ax[1, 3].imshow(mask, cmap=plt.cm.gray)
ax[1, 3].set_title('f) Nucleus Rim')

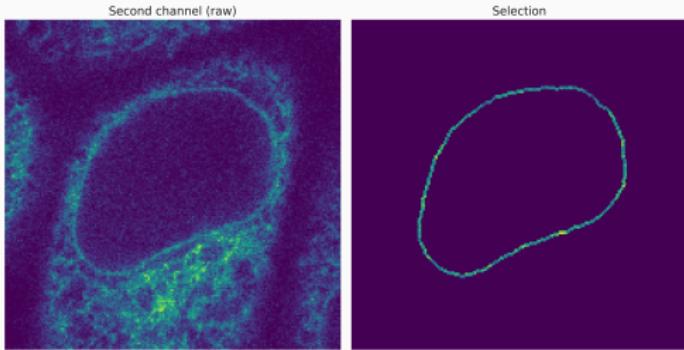
for a in ax.ravel():
    a.set_axis_off()

fig.tight_layout()
plt.savefig("images/Morphological-Operations/nucleus-comparison",
            bbox_inches='tight', dpi=300)
#+end_src

#+RESULTS: FLORO-I

*** Apply Segmented Rim as a Mask
```

# Morphological Operations



**Figure 148:** A comparison of the original image and the detection of the rim.

## Appendix

---



Go Back

- Univariate object is an expression, equation, function or polynomial involving only **one variable**.
- Objects involving more than one variable are multivariate.
  - In statistics, a univariate distribution characterizes one variable, although it can be applied in other ways as well.
    - i.e., in time series analysis, the whole time series is the "variable": a univariate time series is the series of values over time of a single quantity.



# Appendix

[Go Back](#)

In probability theory, the sample space (also called sample description space,[1] possibility space,[2] or outcome space[3]) of an experiment or random trial is the set of all possible outcomes or results of that experiment.[4] A sample space is usually denoted using set notation, and the possible ordered outcomes, or sample points,[5] are listed as elements in the set. It is common to refer to a sample space by the labels  $S$ ,  $\Omega$ , or  $U$  (for "universal set"). The elements of a sample space may be numbers, words, letters, or symbols. They can also be finite, countably infinite, or uncountably infinite.[6]

[Go Back](#)

Stands for red green blue alpha.

While it is sometimes described as a color space, it is actually a three-channel RGB color model supplemented with a fourth alpha channel.

Alpha indicates how opaque each pixel is and allows an image to be combined over others using alpha compositing, with transparent areas and anti-aliasing of the edges of opaque regions. Each pixel is a 4D vector.



[Go Back](#)

There are a wide variety of data types used in `opencv`. The most common are:

[Go Back](#)

Aperture of an optical system is a hole or an opening that primarily limits light propagated through the system.



Figure 149: Different apertures of a lens.

[Go Back](#)

The Gaussian function  $g(x)$ , is defined as:

$$f(x) = \exp(-ax^2)$$

The Fourier transform ( $\mathcal{F}_x$ ) if given by

$$\begin{aligned}\mathcal{F}_x(k) &= \int_{-\infty}^{\infty} \exp(-ax^2) \exp(-2\pi j kx) dx, \\ &= \int_{-\infty}^{\infty} \exp(-ax^2) [\cos 2\pi kx - j \sin 2\pi kx] dx \\ &= \int_{-\infty}^{+\infty} \exp(-ax^2) \cos 2\pi kx dx - j \int_{-\infty}^{+\infty} \exp(-ax^2) \sin 2\pi kx dx.\end{aligned}$$



Go Back

The second integrand is **odd**, so integration over a symmetrical range gives 0.

The value of the first integral is given by Abramowitz and Stegun, so:

$$\mathcal{F}_x(k) = \sqrt{\frac{\pi}{a}} \exp\left(\frac{-\pi^2 k^2}{a}\right),$$

therefore a Gaussian transforms to another Gaussian ■



## Appendix

[Go Back](#)

The Weierstrass transform of a function  $f : \mathbf{R} \rightarrow \mathbf{R}$ , is a smoothed version of  $f(x)$  obtained by averaging the values of  $f$ , weighted with a Gaussian centred at  $x$ .

The function  $F$  is defined by:

$$F(x) = \frac{1}{\sqrt{4\pi}} \int_{-\infty}^{\infty} f(y) e^{-\frac{(x-y)^2}{4}} dy = \frac{1}{\sqrt{4\pi}} \int_{-\infty}^{\infty} f(x-y) e^{-\frac{y^2}{4}} dy$$

the convolution of  $f$  with the Gaussian function

$$\frac{1}{\sqrt{4\pi}} e^{-x^2/4}$$

The factor  $\frac{1}{\sqrt{4\pi}}$  is chosen so that the Gaussian will have a total integral of 1, with the consequence that constant functions are not changed by the Weierstrass transform.



[Go Back](#)

Bivariate analysis is a form of quantitative analysis which involves the analysis of two variables, for the purpose of determining the empirical relationship between them.

Bivariate analysis can be helpful in testing simple hypotheses of association. Bivariate analysis can help determine to what extent it becomes easier to know and predict a value for one variable (possibly a dependent variable) if we know the value of the other variable (possibly the independent variable) (see also correlation and simple linear regression).



[Go Back](#)

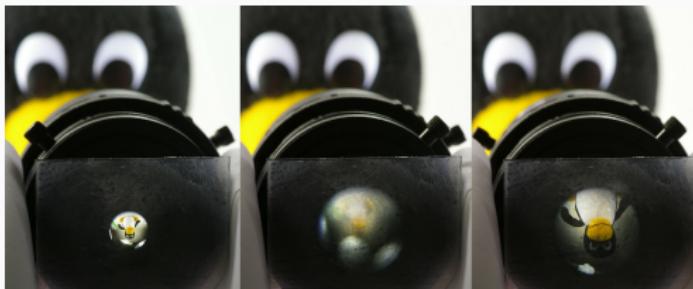
The central limit theorem (CLT) states that, under appropriate conditions, the distribution of a normalized version of the sample mean converges to a standard normal distribution. This holds even if the original variables themselves are not normally distributed. There are several versions of the CLT, each applying in the context of different conditions.

The theorem is a key concept in probability theory because it implies that probabilistic and statistical methods that work for normal distributions can be applicable to many problems involving other types of distributions.



[Go Back](#)

A varifocal lens is a camera lens with variable focal length in which focus changes as focal length (and magnification) changes, as compared to a parfocal ("true") zoom lens, which remains in focus as the lens zooms (focal length and magnification change).



**Figure 150:** A varifocal lens. Left image is at 2.8 mm, in focus. Middle image is at 12 mm with the focus left alone from 2.8 mm. Right image is at 12 mm refocused. The close knob is focal length and the far knob is focus.



# Appendix

[Go Back](#)

In photography and optics, vignetting (/vɪnɪtɪŋ/; vin-YET-ing) is a reduction of an image's brightness or saturation toward the periphery compared to the image center. The word vignette, from the same root as vine, originally referred to a decorative border in a book. Later, the word came to be used for a photographic portrait that is clear at the center and fades off toward the edges. A similar effect is visible in photographs of projected images or videos off a projection screen, resulting in a so-called "hotspot" effect.



[Go Back](#)

Impossible colors are colors that do not appear in ordinary vision.

Different color theories suggest different hypothetical colors that humans are incapable of perceiving for one reason or another, and fictional colors are routinely created in popular culture.

While some such colors have no basis in reality, phenomena such as cone cell fatigue enable colours to be perceived in certain circumstances that would not be otherwise.



[Go Back](#)

Generation loss is the loss of quality between subsequent copies or transcodes of data.

Anything that reduces the quality of the representation when copying, and would cause further reduction in quality on making a copy of the copy, can be considered a form of generation loss.

File size increases are a common result of generation loss, as the introduction of artifacts may actually increase the entropy of the data through each generation.

[Go Back](#)

Shutter speed or exposure time is the length of time that the film or digital sensor inside the camera is exposed to light (that is, when the camera's shutter is open) when taking a photograph [52]. The amount of light that reaches the film or image sensor is proportional to the exposure time.  $1/500$  of a second will let half as much light in as  $1/250$ .



**Figure 151:** Effects of shutter speed on image [14]. As shutter speed slows down the image gets blurrier.

[Go Back](#)

Baron Siméon Denis Poisson **FRS FRSE** was a French mathematician and physicist who worked on statistics, complex analysis, partial differential equations, the calculus of variations, analytical mechanics, electricity and magnetism, thermodynamics, elasticity, and fluid mechanics.

Moreover, he predicted the **Arago spot** in his attempt to disprove the wave theory of Augustin-Jean Fresnel.

21 June 1781 – 25 April 1840



Figure 152: Baron Siméon Denis Poisson.

[Go Back](#)

In optics, the Arago spot, Poisson spot, or Fresnel spot is a bright point that appears at the center of a circular object's shadow due to Fresnel diffraction.

This spot played an important role in the discovery of the wave nature of light and is a common way to demonstrate that light behaves as a wave.

[Go Back](#)

An anonymous function in Python is a function without a name. It can be immediately invoked or stored in a variable. Anonymous functions in Python are also known as lambda functions.



Slides were created using **GNU Emacs** version 29.1 with **AUCTeX** 14.0.7.

*"Emacs, is a family of text editors that are characterised by their extensibility. The manual for the most widely used variant, GNU Emacs, describes it as "the extensible, customizable, self-documenting, real-time display editor."*

*"AUCTeX is a package for writing and formatting TeX files in GNU Emacs."*

**Beamer** class was used as template with the **LuaTeX** engine.

*"Beamer is a LaTeX class for generating slides."*

*"LuaTeX is a TeX-based computer typesetting system which started as a version of pdfTeX with a Lua scripting engine embedded."*

All code presented in lectures are in **Python**, using version 3.9.13.

*"Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation."*



# Appendix

All non raster images were compressed using `pngquant` and `magick` CLI tools. Compression settings were kept at standard for `pngquant` and %80 compression were used for `magick`.

```
mogrify -quality 80% *.jpg  
pngquant -f --ext .png **/*.png
```



- [1] Edward H Adelson. "Checkershadow illusion. 1995". In: URL [http://web.mit.edu/persci/people/adelson/checkershadow\\_illusion.html](http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html) (2005).
- [2] Adobe. *Lossy vs Lossless Compression Differences and When to Use*. 2024. URL: <https://www.adobe.com/uk/creativecloud/photography/discover/lossy-vs-lossless.html>.
- [3] Amer Al-Rahayfeh and Miad Faezipour. "Enhanced frame rate for real-time eye tracking using circular hough transform". In: *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE. 2013, pp. 1–6.
- [4] American Optometric Association. *Color vision deficiency*. 2024. URL: <https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/color-vision-deficiency?sso=y>.
- [5] Colour Blind Awareness. *Inherited Colour Vision Deficiency*. 2024. URL: <https://www.colourblindawareness.org/colour-blindness/causes-of-colour-blindness/inherited-colour-vision-deficiency/>.
- [6] Colour Blind Awareness. *Types of Colour Blindness*. 2024. URL: <https://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/>.

## Appendix ii

- [7] BBC. *Optical illusion: Dress colour debate goes global*. 2015. URL: <https://www.bbc.com/news/uk-scotland-highlands-islands-31656935>.
- [8] Cmglee. *A Bayer pattern on a sensor in isometric perspective/projection*. 2020. URL: [https://commons.wikimedia.org/wiki/File:Bayer\\_pattern\\_on\\_sensor.svg](https://commons.wikimedia.org/wiki/File:Bayer_pattern_on_sensor.svg).
- [9] Crisp. *Getting out of Auto: Understanding ISO on your digital camera*. 2016. URL: <https://newatlas.com/photography-iso-range-setting-guide/44599/>.
- [10] Ralph Dammel. *Diazonaphthoquinone-based resists*. Vol. 11. SPIE press, 1993.
- [11] DarkRoom. *Effects of Focal Length and Angle*. 2024. URL: <https://thedarkroom.com/focal-length/>.
- [12] Michael F Deering. "The limits of human vision". In: *2nd international immersive projection technology workshop*. Vol. 2. 1. 1998.
- [13] Zoltan Derzsi and Robert Volcic. "Not only perception but also grasping actions can obey Weber's law". In: *Cognition* 237 (2023), p. 105465.
- [14] Dilmen. *Windflower and Shutter speed*. 2004. URL: <https://commons.wikimedia.org/wiki/File:Windflower-05237-nevit.JPG>.

## Appendix iii

- [15] DrBob. *Simple zoom lens*. 2005. URL:  
<https://commons.wikimedia.org/wiki/File:Zoomlens1.svg>.
- [16] EdmundOptics. *Anatomy of a Lens*. 2024. URL:  
<https://www.edmundoptics.com/knowledge-center/application-notes/imaging/the-anatomy-of-a-lens/>.
- [17] Estrin. *Kodak's First Digital Moment*. 2015. URL:  
<https://archive.nytimes.com/lens.blogs.nytimes.com/2015/08/12/kodaks-first-digital-moment/>.
- [18] S. A. Eugster. *CbCr plane of the YCbCr color space, with a Y value of 0.5, Cb on the horizontal x axis going from -1 to 1, Cr on the y axis going from -1 to 1 as well*. 2010. URL: [https://commons.wikimedia.org/wiki/File:YCbCr-CbCr\\_Scaled\\_Y50.png](https://commons.wikimedia.org/wiki/File:YCbCr-CbCr_Scaled_Y50.png).
- [19] Expromo. *Led Display Facts*. 2020. URL:  
<https://www.expmrmo.eu/en/led-display-facts/>.
- [20] Filya1. *CMOS image sensor*. 2009. URL:  
<https://en.m.wikipedia.org/wiki/File:Matrixw.jpg>.
- [21] FlickreviewR. *Sony F828 Camera*. 2009. URL:  
[https://commons.wikimedia.org/wiki/File:Sony\\_F828.jpg](https://commons.wikimedia.org/wiki/File:Sony_F828.jpg).

## Appendix iv

- [22] Eric R Fossum. "Active pixel sensors: Are CCDs dinosaurs?" In: *Charge-Coupled Devices and Solid State Optical Sensors III*. Vol. 1900. SPIE. 1993, pp. 2–14.
- [23] Eric R Fossum and Donald B Hondongwa. "A review of the pinned photodiode for CCD and CMOS image sensors". In: *IEEE Journal of the electron devices society* (2014).
- [24] Eric Fox. *CMOS TDI: A Game Changer in High-Fidelity Imaging*. 2019. URL: <https://possibility.teledyneimaging.com/cmos-tdi-a-game-changer-in-high-fidelity-imaging/>.
- [25] Michael Gabler. *Various image formats are categorized by scope in this diagram, along two axes: Raster vs Vector, and Authoring vs Delivery*. 2011. URL: [https://commons.wikimedia.org/wiki/File:Felis\\_silvestris\\_silvestris\\_small\\_gradual\\_decrease\\_of\\_quality.png](https://commons.wikimedia.org/wiki/File:Felis_silvestris_silvestris_small_gradual_decrease_of_quality.png).
- [26] Genelec. *Audio Test Signals*. 2024. URL: <https://www.genelec.com/audio-test-signals>.
- [27] Gigahertz-Optik. *Spectral Sensitivity of the Human Eye*. 2002. URL: <https://www.gigahertz-optik.com/en-us/service-and-support/knowledge-base/basics-light-measurement/light-color/spectr-sens-eye/>.
- [28] Rudolf F Graf. *Modern dictionary of electronics*. Elsevier, 1999.

## Appendix v

- [29] healthdirect. *Colour blindness*. 2024. URL: <https://www.healthdirect.gov.au/colour-blindness>.
- [30] Hyperphysics. *The Color-Sensitive Cones*. 2024. URL: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/colcon.html>.
- [31] Logical Increments. *Information About Screen Resolution*. 2023. URL: <https://www.logicalincrements.com/resolution>.
- [32] *Interlaced Video & Deinterlacing for streaming*. 2021. URL: <https://blog.video.ibm.com/streaming-video-tips/interlaced-video-deinterlacing-for-streaming/>.
- [33] Amanda Jefferies et al. “Do You See What I See?: Understanding the Challenges of Colour-Blindness in Online Learning”. In: *Proceedings of 10th European Conference for E-Learning*. Academic Conferences Ltd. 2011.
- [34] Susanne Kohl et al. “Achromatopsia”. In: (2018).
- [35] *LCD vs. LED: What is LCD & LED*. 2023. URL: <https://www.linsnled.com/difference-between-lcd-and-led.html>.
- [36] Ji-Won Lee et al. “Screenshot identification by analysis of directional inequality of interlaced video”. In: *EURASIP Journal on Image and Video Processing* 2012 (2012), pp. 1–15.

- [37] Logan. *A monochrome CRT as seen inside a Macintosh Plus computer*. 2018. URL: [https://commons.wikimedia.org/wiki/File:Macintosh\\_Plus\\_interior.jpg](https://commons.wikimedia.org/wiki/File:Macintosh_Plus_interior.jpg).
- [38] Harris R Miller. "Color filter array for CCD and CMOS image sensors using a chemically amplified thermally cured pre-dyed positive-tone photoresist for 365-nm lithography". In: *Advances in Resist Technology and Processing XVI*. Vol. 3678. SPIE. 1999, pp. 1083–1090.
- [39] Alistair Moffat. "Huffman coding". In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–35.
- [40] Morio. *Fujinon XF100-400mm F4.5-5.6 R LM OIS WR cutaway*. 2016. URL: [https://commons.wikimedia.org/wiki/File:Fujinon\\_XF100-400mm\\_F4.5-5.6\\_R\\_LM\\_OIS\\_WR\\_cutaway\\_2016\\_China\\_P%26E.jpg](https://commons.wikimedia.org/wiki/File:Fujinon_XF100-400mm_F4.5-5.6_R_LM_OIS_WR_cutaway_2016_China_P%26E.jpg).
- [41] *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB*. Standard. Geneva, CH: International Electrotechnical Commision, 1999.
- [42] *Multimedia systems and equipment - Colour measurement and management - Part 2-5: Colour management - Optional RGB colour space - opRGB*. Standard. Geneva, CH: International Electrotechnical Commision, 2007.

- [43] Junichi Nakamura. *Image sensors and signal processing for digital still cameras*. CRC press, 2017.
- [44] Olympus. *Introduction to CMOS Image Sensors*. 2024. URL: <https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/cmosimagesensors/>.
- [45] Nobuyuki Otsu et al. "A threshold selection method from gray-level histograms". In: *Automatica* 11.285-296 (1975), pp. 23–27.
- [46] Bedabrata Pain et al. "A back-illuminated megapixel CMOS image sensor". In: (2005).
- [47] Danny Pascale. "A review of rgb color spaces... from  $x_{yy}$  to  $r'g'b'$ ". In: *Babel Color* 18 (2003), pp. 136–152.
- [48] *Philips Test Card*. 2008. URL: [https://commons.wikimedia.org/wiki/File:Philips\\_PM5544.svg](https://commons.wikimedia.org/wiki/File:Philips_PM5544.svg).
- [49] *Photography and graphic technology — Extended colour encodings for digital image storage, manipulation and interchange*. Standard. Geneva, CH: International Organization for Standardization, 2013.

## Appendix viii

- [50] PolBr. *sRGB colors situated at calculated position in CIE 1931 chromaticity diagram (edited from CIExy1931\_sRGB.svg). Luminance Y sets so that R + G + B = 1 to avoid bright lines toward primaries' complementary colours..* 2021. URL: [https://commons.wikimedia.org/wiki/File:SRGB\\_chromaticity\\_CIE1931.svg](https://commons.wikimedia.org/wiki/File:SRGB_chromaticity_CIE1931.svg).
- [51] Mary C Potter et al. "Detecting meaning in RSVP at 13 ms per picture". In: *Attention, Perception, & Psychophysics* 76 (2014), pp. 270–279.
- [52] Sidney Ray. *Applied photographic optics*. Routledge, 2002.
- [53] R. Rehák, P. Bodrogi, and J. Schanda. "On the use of the sRGB colour space". In: *Displays* 20.4 (1999), pp. 165–170. ISSN: 0141-9382. DOI: [https://doi.org/10.1016/S0141-9382\(99\)00019-0](https://doi.org/10.1016/S0141-9382(99)00019-0). URL: <https://www.sciencedirect.com/science/article/pii/S0141938299000190>.
- [54] Inc. Reproductions. *Understanding the Differences Between RGB and CYMK Colors*. 2024. URL: <https://reproductionsinc.com/understanding-the-differences-between-rgb-and-cymk-colors/>.
- [55] Mark Stanford Robbins. "Electron-multiplying charge coupled devices—emccds". In: *Single-Photon Imaging* (2011), pp. 103–121.

- [56] Andrew Rodney. "The role of working spaces in Adobe applications". In: *Technical Paper. Adobe*. Retrieved (2008), pp. 05–09.
- [57] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.
- [58] Mehmet Sezgin and Bu" lent Sankur. "Survey over image thresholding techniques and quantitative performance evaluation". In: *Journal of Electronic imaging* 13.1 (2004), pp. 146–168.
- [59] Slippens. *Three examples of color halftoning with CMYK separations*. 2009. URL: <https://en.wikipedia.org/wiki/File:Halftoningcolor.svg>.
- [60] J. Sneyers. *Various image formats are categorized by scope in this diagram, along two axes: Raster vs Vector, and Authoring vs Delivery*. 2022. URL: [https://commons.wikimedia.org/wiki/File:Image\\_formats\\_by\\_scope.svg](https://commons.wikimedia.org/wiki/File:Image_formats_by_scope.svg).
- [61] Kevin E Spaulding, Geoffrey J Woolfe, and Edward J Giorgianni. "Reference input/output medium metric rgb color encodings". In: *Color Imaging Conference*. Vol. 1. 2000, p. 2.
- [62] Benjamin Tag et al. "In the eye of the beholder: The impact of frame rate on human eye blink". In: *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*. 2016, pp. 2321–2327.

## Appendix x

- [63] Teledyne. *How to Evaluate Camera Sensitivity*. 2021. URL: <https://www.flir.com/discover/iis/machine-vision/how-to-evaluate-camera-sensitivity/>.
- [64] Teledyne. *Types of Camera Sensor*. 2024. URL: <https://www.photometrics.com/learn/camera-basics/types-of-camera-sensor>.
- [65] TeledyneCurrent. *Dark Current*. 2024. URL: <https://www.photometrics.com/learn/advanced-imaging/dark-current>.
- [66] teledyneimage. *Linearity: Image Topics*. 2024. URL: <https://www.photometrics.com/learn/imaging-topics/linearity>.
- [67] European Broadcasting Union. "Enhanced 625-line Phased Alternate Line (PAL) television". In: (1997).
- [68] Zurek. *Chromatic aberration*. 2006. URL: [https://commons.wikimedia.org/wiki/File:Chromatic\\_aberration\\_\(comparison\).jpg](https://commons.wikimedia.org/wiki/File:Chromatic_aberration_(comparison).jpg).