

B.Sc

# Digital Image Processing

LectureSlide

D. T. McGuiness, Ph.D

WS.2025





1. Mathematical Fundamentals
2. Perception
3. Image Formats
4. Camera
5. Display
6. Noise
7. Histogram Operations
8. Morphological Opeations
9. Blurring Filters
10. Edge Detection
11. Processing Applications
12. Bibliography



Chapter

# Mathematical Fundamentals



## Introduction

Learning Outcomes

Image Processing

## Convolution

Mathematical Definition

A Hospital Visit (0)

2D Convolution

## Signal Sampling

### Nyquist Sampling Theorem

Mathematical Definition

Reconstruction of an Audio Signal

Aliasing

Leakage

Parseval's Theorem

Statistical Properties

# Introduction

Learning Outcomes

D. T. McGuiness, Ph.D

- (LO1) An Overview of mathematical methods,
- (LO2) A revisit on convolution.
- (LO3) Definitions of analogue and digital,
- (LO4) A mathematical look into Discrete Fourier Transform (DFT).



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

## Concepts and Principles

- Principle of convolution,
- Discrete Fourier analysis,
- Shannon-Nyquist Sampling Theorem,
- A brief introduction to Information Theory,
- The concept of information entropy.

## Convolution

Mathematical Definition

D. T. McGuiness, Ph.D

- Convolution, mathematically is defined as:

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

- Where  $f$  and  $g$  are arbitrary function and  $*$  is the convolution operator.

To put it simply, convolution is just fancy multiplication. But in principle, it has a strong relationship with Laplace transform and is used significantly in image processing.

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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?

Example

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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:

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

Solution

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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? Let's see

Solution

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

- 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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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

C.R. 1.1  
text

Monday

Rooms	3	2	1
Patients	5	4	3
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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

On Tuesday, everyone takes a step forward:

Tuesday

Rooms                    3 2 1

Patients ->        5 4 3 2 1

Usage                    6 2            = 8

C.R. 1.2  
text

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

# Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

C.R. 1.3  
text

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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):

C.R. 1.4  
python

$$\begin{array}{l} \text{Plan} \quad * \quad \text{Patient List} \quad = \text{Total Daily Usage} \\ [3 \ 2 \ 1] \quad * \quad [1 \ 2 \ 3 \ 4 \ 5] \quad = \ [3 \ 8 \ 14 \ 20 \ 26 \ 14 \ 5] \\ \text{M T W T F} \qquad \qquad \qquad \text{M T W T F S S} \end{array}$$

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".

Solution

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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

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

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

Example

# Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

C.R. 1.5  
python

```
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
```

Solution

# Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

```
A = [1,1,2,2,1]  
B = [1,1,1,3]  
  
print(convolve_1d(A, B))
```

C.R. 1.6  
python

Solution

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

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

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

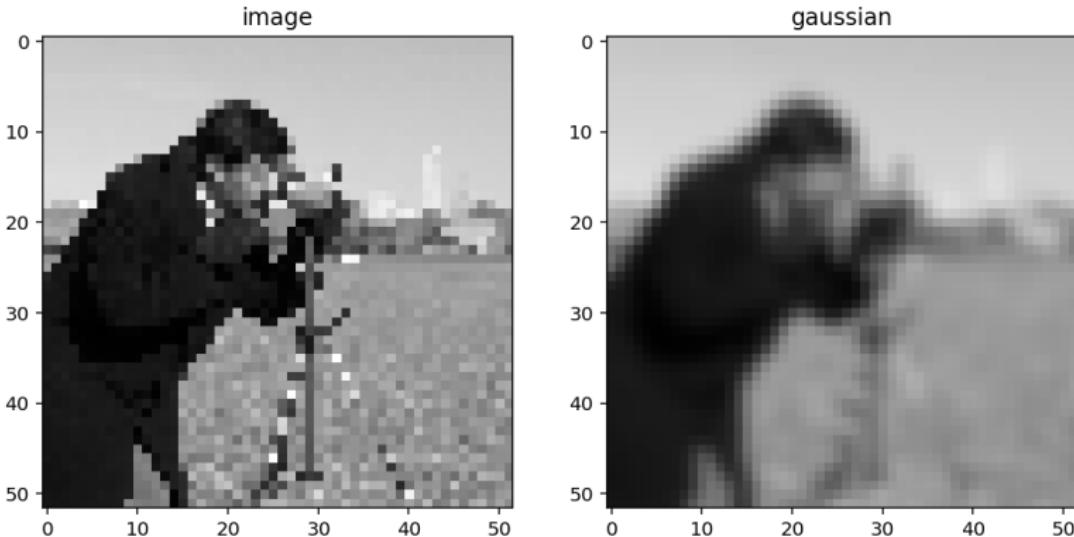
- There are several notations to indicate convolution with the most common is:

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

# Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D



**Figure:** An example of convolution used in image processing. Here a pixellated image is smoothed out using Gaussian Blur which relies on convolution.

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

- 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**):

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

- 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 as it relates the three **(3)** signals of interest:
  1. Input signal,
  2. Output signal,
  3. Impulse response.
- But now, let's look at some of it's properties:

**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 said to be distributive:

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

### Associative

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

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

### 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:

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

## Convolution

A Hospital Visit (0)

D. T. McGuiness, Ph.D

- Similarly:

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

- Therefore:

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

# Convolution

## 2D Convolution

D. T. McGuiness, Ph.D

**Figure:** 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  $\delta(x, y)$ :

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

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

## Nyquist Sampling Theorem

Mathematical Definition

D. T. McGuiness, Ph.D

- 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 and **NOT** a practical one.

# Nyquist Sampling Theorem

Mathematical Definition

D. T. McGuiness, Ph.D

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

# Nyquist Sampling Theorem

Mathematical Definition

D. T. McGuiness, Ph.D

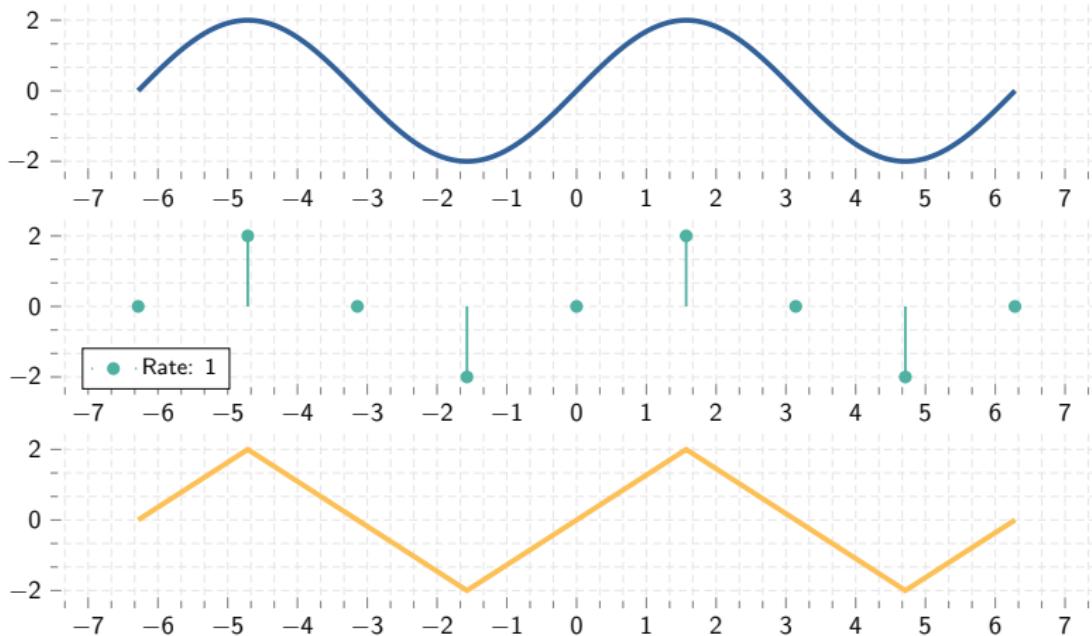
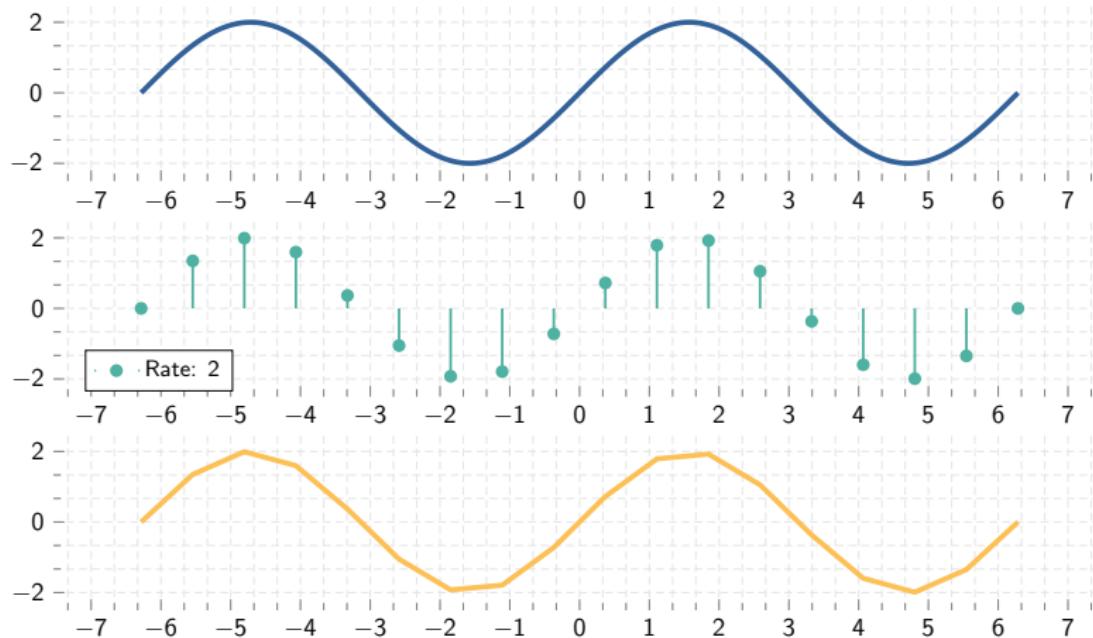


Figure: Reconstruction of the signal with 1 times the signal frequency.

## Nyquist Sampling Theorem

## Mathematical Definition

D. T. McGuiness, Ph.D

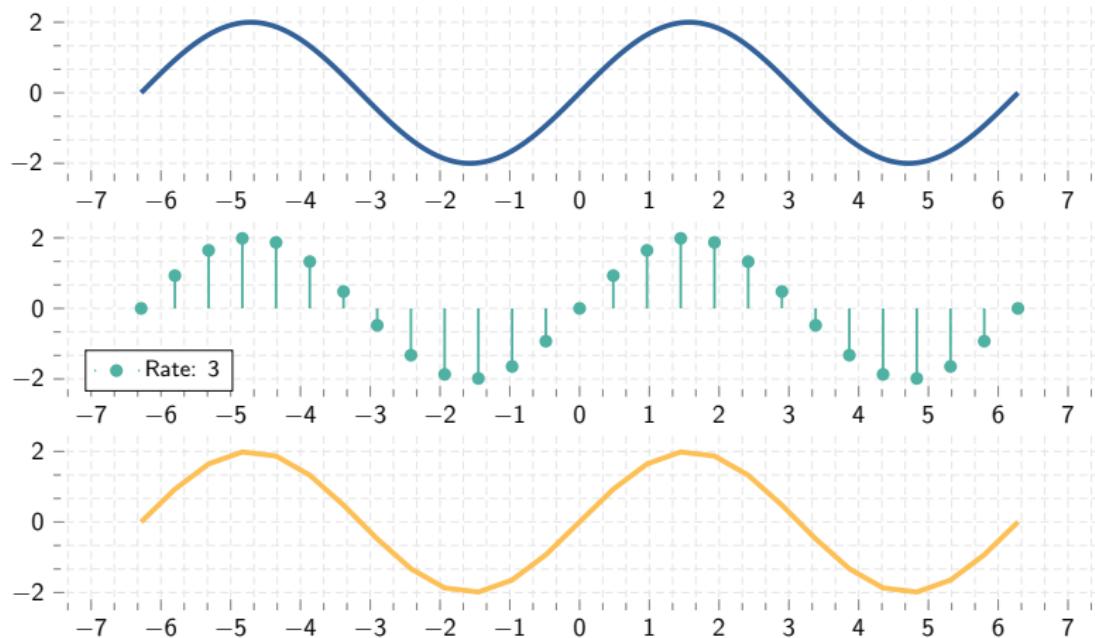


**Figure:** Reconstruction of the signal with 2 times the signal frequency.

## Nyquist Sampling Theorem

## Mathematical Definition

D. T. McGuiness, Ph.D



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

# Nyquist Sampling Theorem

Mathematical Definition

D. T. McGuiness, Ph.D

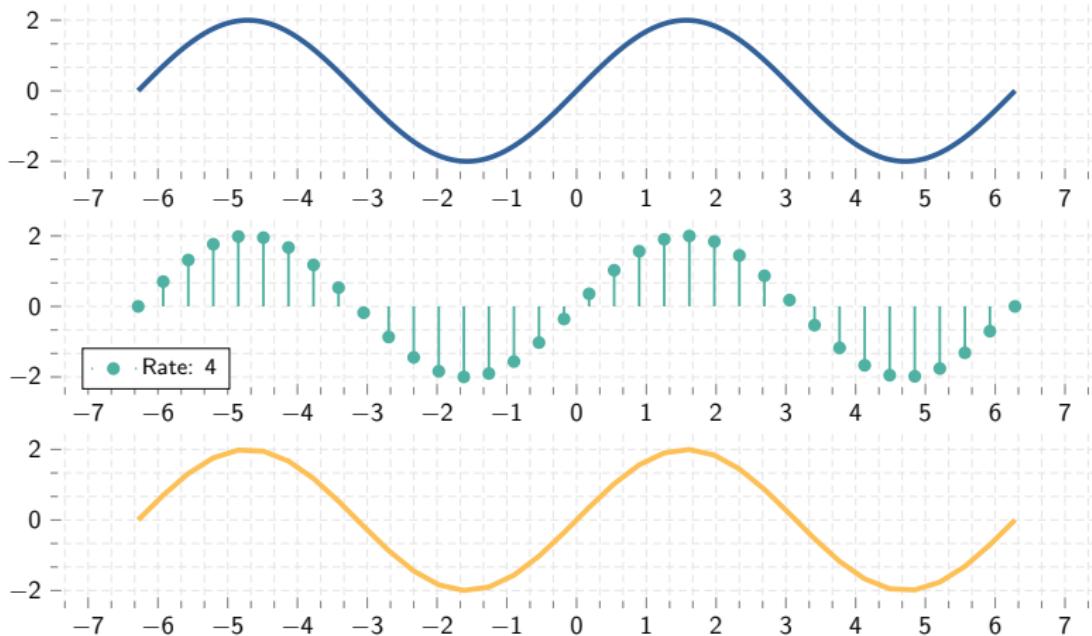
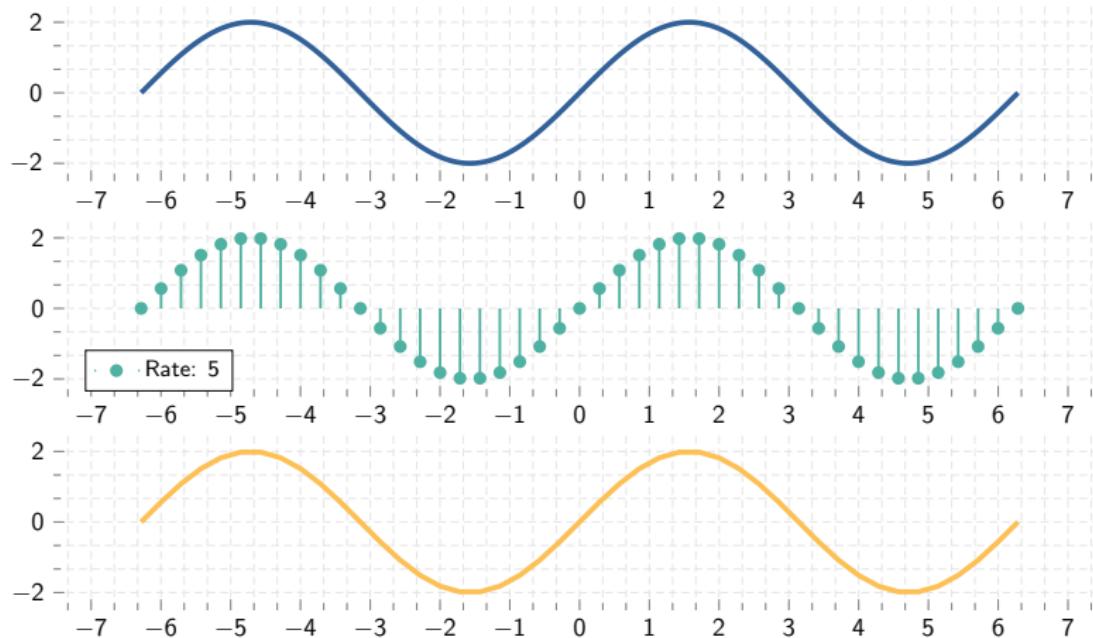


Figure: Reconstruction of the signal with 4 times the signal frequency.

## Nyquist Sampling Theorem

## Mathematical Definition

D. T. McGuiness, Ph.D



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

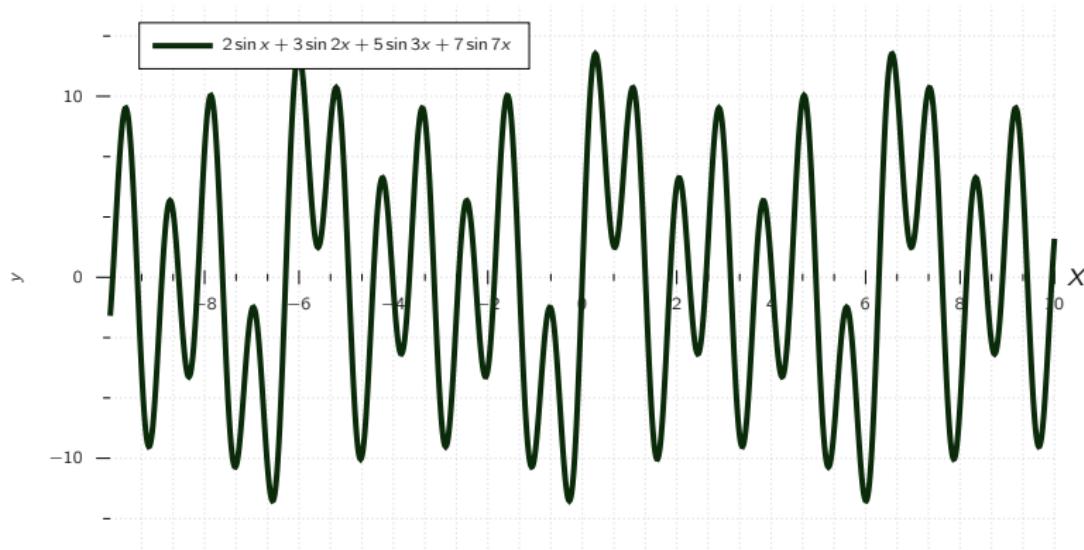
## Nyquist Sampling Theorem

Reconstruction of an Audio Signal

D. T. McGuiness, Ph.D

- 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 which will vary based on phase.
- Even 4 times 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.

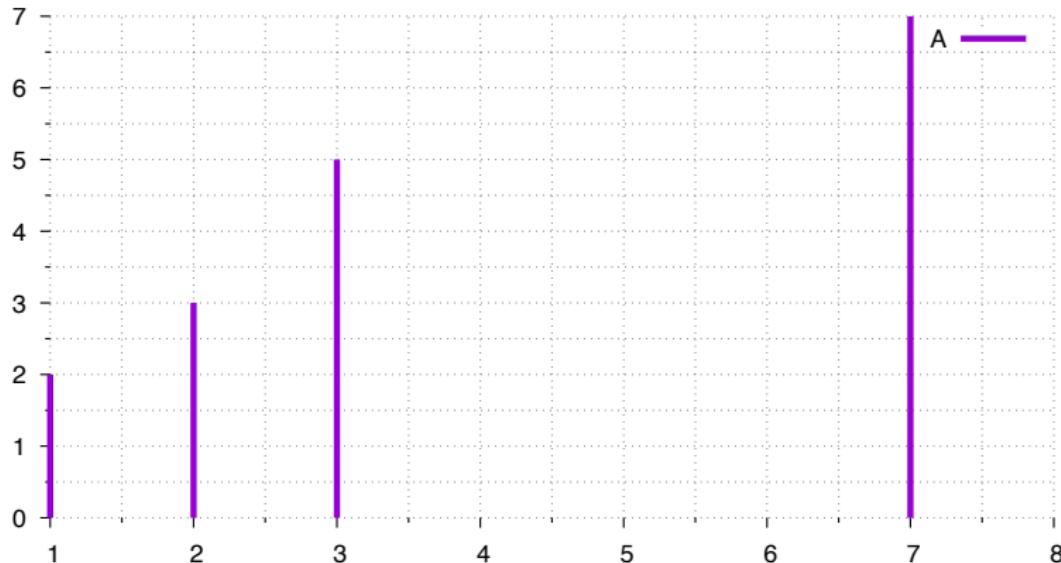


**Figure:** A sample signal with containing sample sine waves.

## Nyquist Sampling Theorem

## Reconstruction of an Audio Signal

D. T. McGuiness, Ph.D



**Figure:** The FFT of the previous complex signal.

# Nyquist Sampling Theorem

Reconstruction of an Audio Signal

D. T. McGuiness, Ph.D

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

**Aliasing** Only resolving frequencies lower than the *Nyquist frequency* and higher frequencies get aliased to lower frequencies.

**Spectral Leakage** we assume all wave-forms stop and start at 0 and end at  $n$ , but in the real world, many wave-numbers may not complete a full integer number of cycles throughout the domain, causing spectral leakage to other wave numbers.

## Nyquist Sampling Theorem

Aliasing

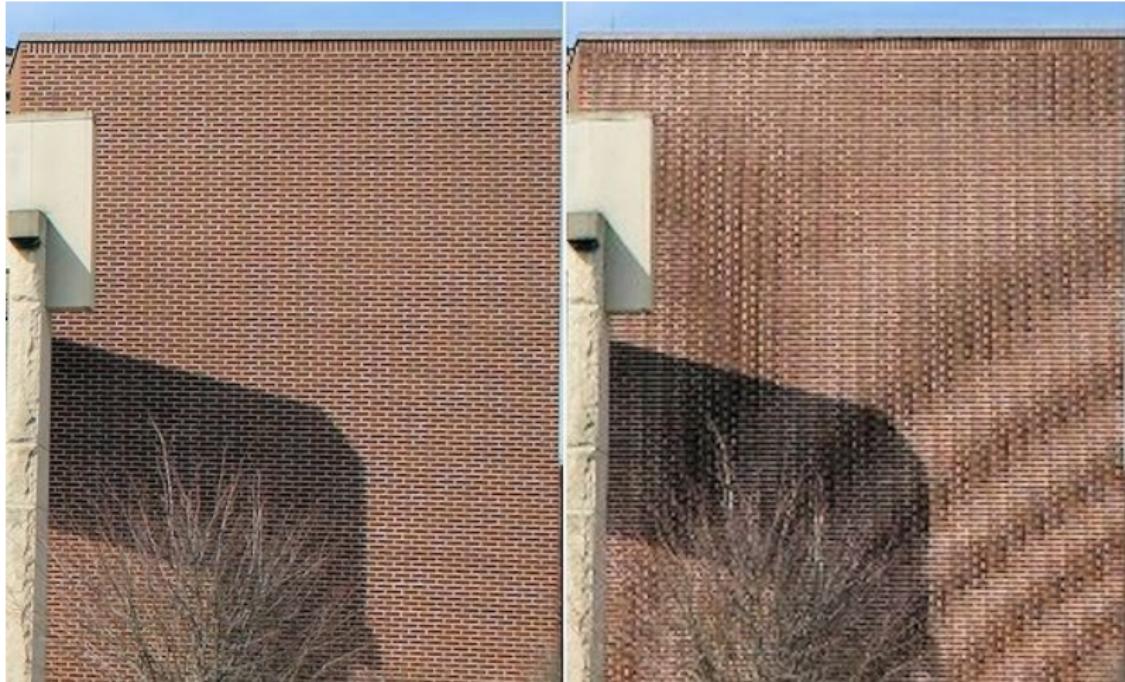
D. T. McGuiness, Ph.D

- 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.

# Nyquist Sampling Theorem

Aliasing

D. T. McGuiness, Ph.D



**Figure:** An example of under-sampling an image. Here aliasing produces non-real distortions of digitized images.

## Nyquist Sampling Theorem

Leakage

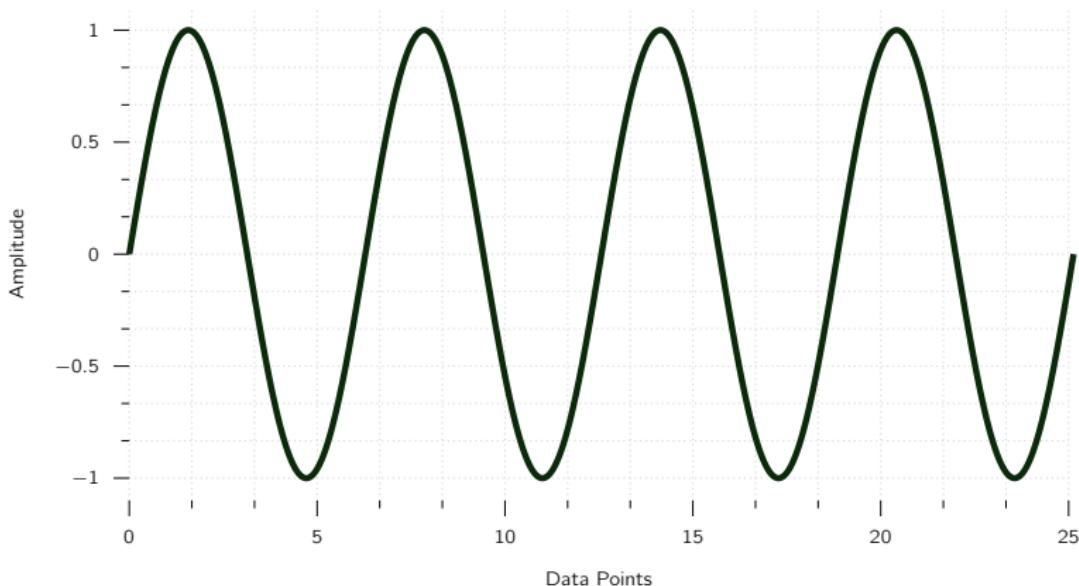
D. T. McGuiness, Ph.D

- 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, might cause the transform to be corrupted in some way.
- Let's start with looking at a simple sine-wave

# Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D



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

# Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

- Computing the discrete power spectrum gives:

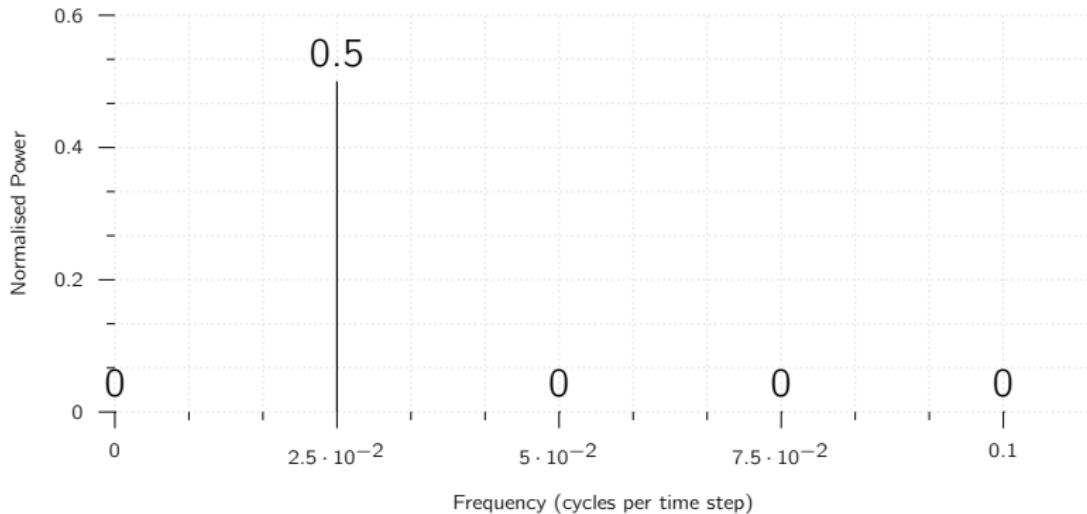


Figure: The PSD of an un-windowed sine wave.

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

- 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 which **cuts off** the sine wave such that the sine function does not complete an integer number of cycles within the time domain.

# Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

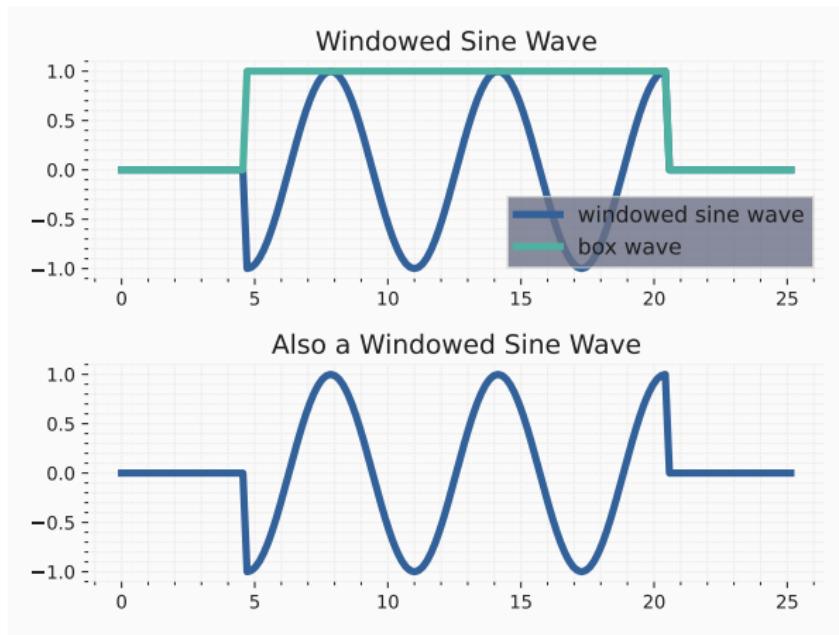


Figure: Windowed sine wave.

# Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

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

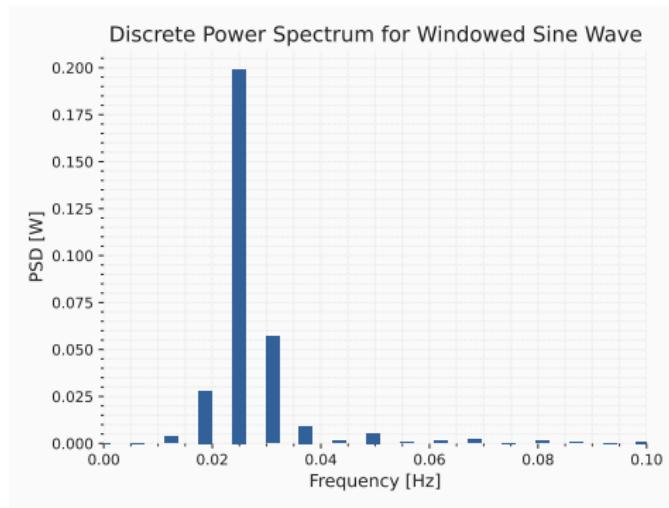


Figure: PSD of a windowed sine wave.

# Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

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

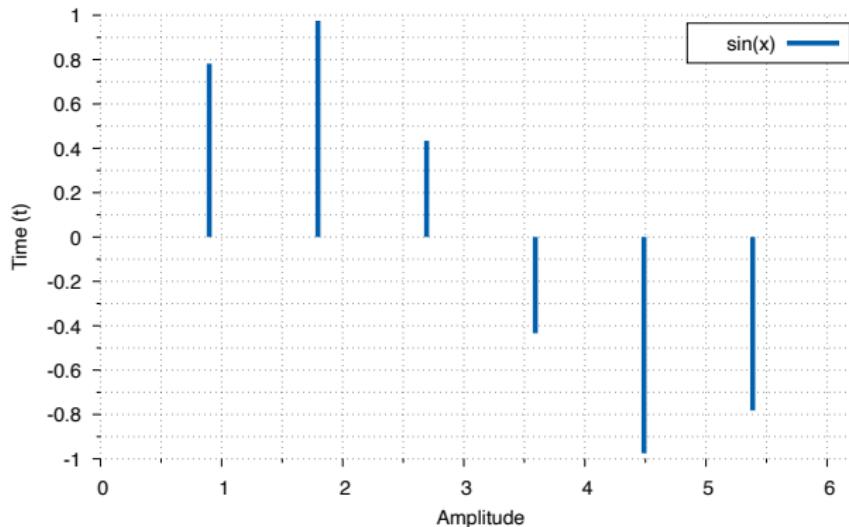


Figure: A Sampled Sine wave.

Example

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

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:

$$x_k = [0 \quad 0.707 \quad 1 \quad 0.707 \quad 0 \quad -0.707 \quad -1 \quad -0.707]$$

Solution

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

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

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

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

For  $x_0 = 0$  the exponential is removed and are left with  $X_0 = 0$ .

Solution

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

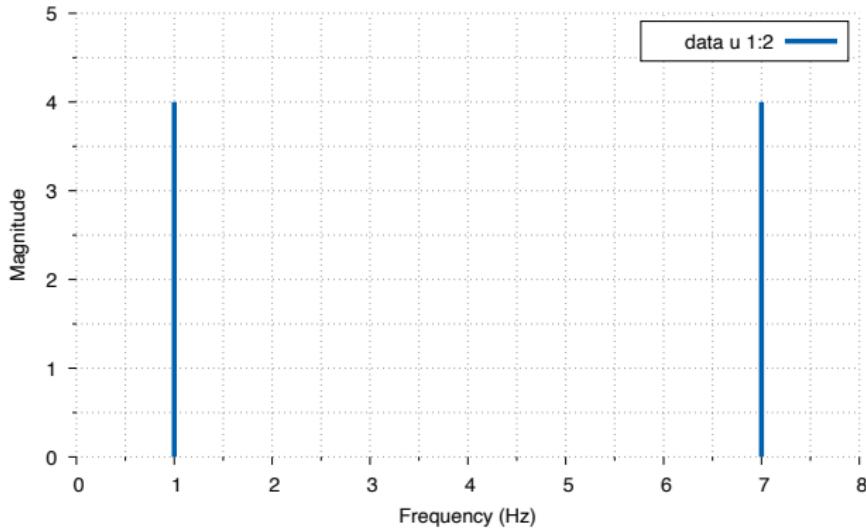
- Therefore the values are of the transform are:

$$X_k = \begin{bmatrix} 0 & 0 - j4 & 0 & 0 & 0 & 0 & 0 & 0 + j4 \end{bmatrix}$$

- 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.

$$|X_k| = \begin{bmatrix} 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

Solution



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

### Solution

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

- The frequency resolution of the plot is the sampling frequency divided by the number of samples:

$$\text{Resolution} = \frac{\text{Sampling Frequency}}{\text{Number of Samples}}$$

- This means we can get values for every integer frequency values.

Solution

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

- 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 frequency.

The negative frequencies are always complex conjugate to the positive frequencies, so there is no additional information in the negative frequencies.

Solution

## Nyquist Sampling Theorem

Leakage

D. T. McGuiness, Ph.D

- Therefore, to convert from a two-sided spectrum to a single-sided spectrum, we discard the second half of the array and multiply every point except for DC by two (2).
- The last operation is to divide the magnitudes of the lower frequencies by the number of samples used in deriving these bins:

$$\begin{aligned} \mathbf{X}_k &= \begin{bmatrix} 0 & 8 & 0 & 0 \end{bmatrix} \\ \mathbf{X}_k/N &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \quad \blacksquare \end{aligned}$$

Solution

## Nyquist Sampling Theorem

Parseval's Theorem

D. T. McGuiness, Ph.D

- 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.

# Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

## 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 ( $\mu$ ):

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

# Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

## Mode

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

$$X = [1 \ 2 \ 3 \ 4 \ 5 \ 2 \ 7 \ 2 \ 9]$$

- 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 (2) values with same frequency.

# Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

## 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:

$$\text{med}(x) = x \frac{n+1}{2},$$

- if  $n$  is even,

$$\text{med}(x) = x \left( \frac{n}{2} \right) + x \frac{\frac{n}{2} + 1}{2}$$

## Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

- i.e., in a ordered data set of:

$$X = \begin{bmatrix} 1 & 2 & 2 & 3 & 4 & 7 & 9 \end{bmatrix},$$

- the median is 3 and in:

$$Y = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 8 & 9 \end{bmatrix}.$$

the median is 4.5.

# Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

- The signal-to-noise ratio (SNR) can have several definitions depending on the field.
- Noise is characterised by its standard deviation,  $\sigma$ .
- The characterisation of the signal can differ.

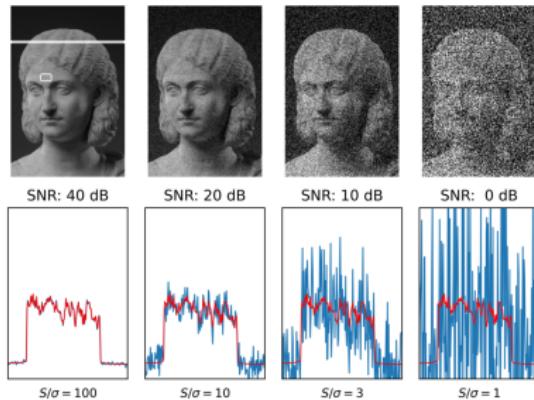


Figure: A gray-scale photography with different signal-to-noise ratios (SNRs).

# Nyquist Sampling Theorem

Statistical Properties

D. T. McGuiness, Ph.D

- If the signal is known to lie between two (2) 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:

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



Chapter

# Perception

## Learning Outcomes

### Introduction

Human Vision

Brightness Sensitivity

Stimulus Sensitivity

Colour Sensitivity

### Colour Standards

sRGB

Wide Gamut RGB

Prophoto RGB

Adobe RGB

CIE Chromaticity Coordinates

Chromaticity

### Colour Models

CYMK Colour Model

HSL and HLV Colour Model

YCbCr

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



## Introduction

Human Vision

D. T. McGuiness, Ph.D

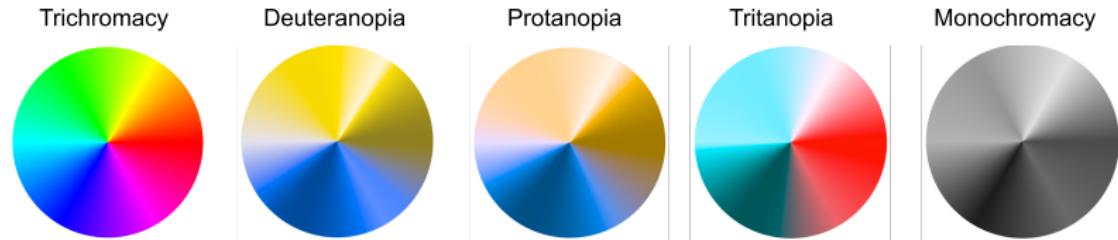
- Many image processing applications are intended to produce images to be viewed by **humans**.
  - This is in contrast to industrial robots.
- It is important to understand the characteristics and limitations of the human visual system [1].

# Introduction

Human Vision

D. T. McGuiness, Ph.D

- At the outset it is important to realise:
  1. The human visual system is **not well understood** [1].
    - 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:** A color wheel depicted approximately as it would be seen by a person with different kinds of color vision or color blindness: trichromacy (normal colour vision), deutanopia (red-green color blind), protanopia (red-green colour blind), tritanopia (blue-yellow color blind), and monochromacy (completely colour blind). [2].

## Introduction

Human Vision

D. T. McGuiness, Ph.D

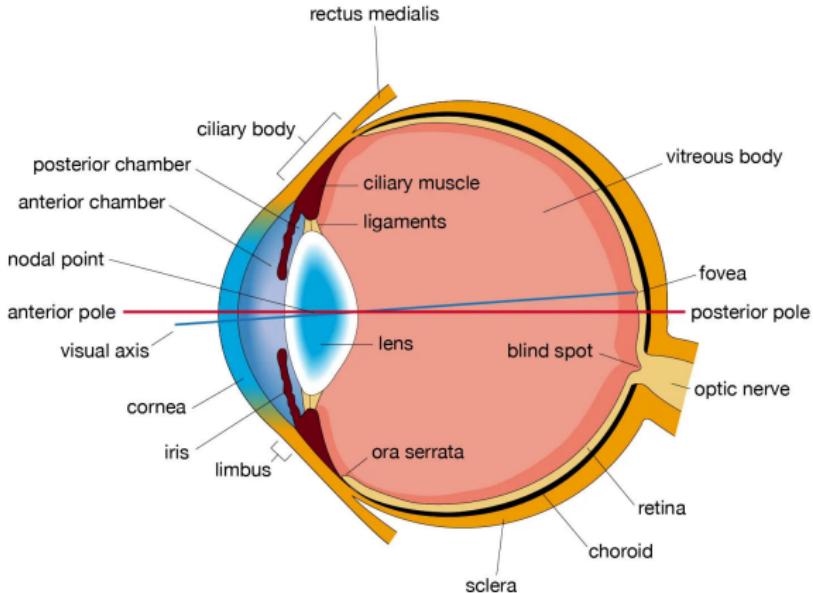
- Statistically 1 in 8 men and 1 in 200 women have a form of colour blindness [3].
- Men have one **X** chromosome and one **Y** chromosome, while women have two (**2**) **X** chromosomes [4].
- To experience color blindness, the genetic mutation for color-blindness must be present on the **X** chromosome, but for women, this means it must be present on both **X** chromosomes.

Men only need to mutation to be present on their singular **X** chromosome, making it much easier for them to inherit color blindness.

# Introduction

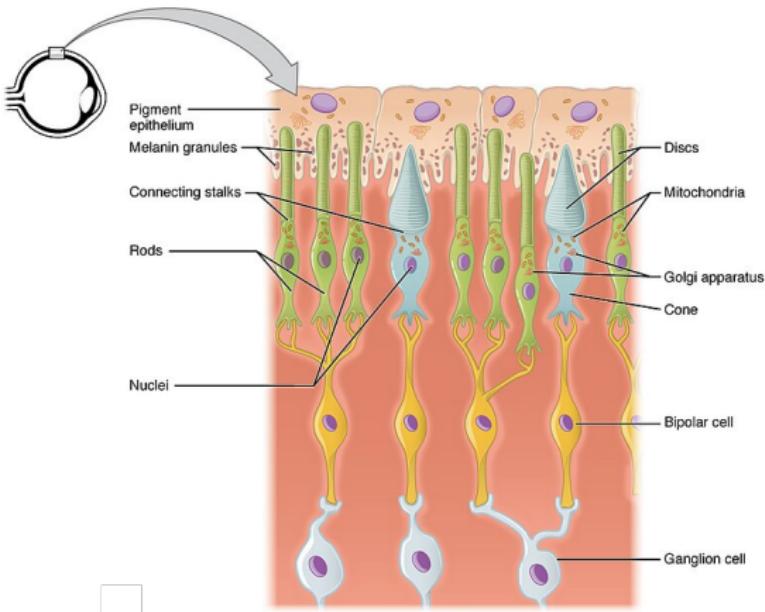
## Human Vision

D. T. McGuiness, Ph.D



© 2013 Encyclopædia Britannica, Inc.

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



**Figure:** Functional parts of the rods and cones, which are two of the three types of photosensitive cells in the retina.

## Introduction

Human Vision

D. T. McGuiness, Ph.D

### Trichromacy

- Normal colour vision uses all three (3) types of cone cells.
- 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 [3].
- In people with this condition all of their three (3) 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.

## Introduction

Human Vision

D. T. McGuiness, Ph.D

- The different anomalous condition types are [5]:

**protanomaly** reduced sensitivity to red light,

**deuteranomaly** reduced sensitivity to green light (most common),

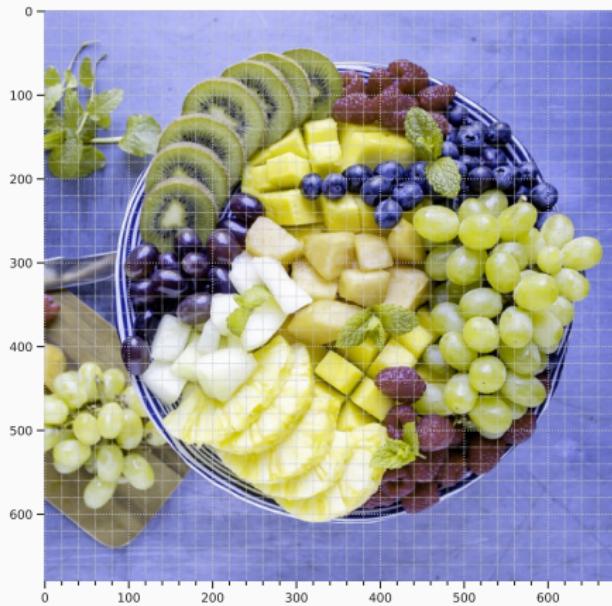
**tritanomaly** reduced sensitivity to blue light (most uncommon).

## Achromatopsia

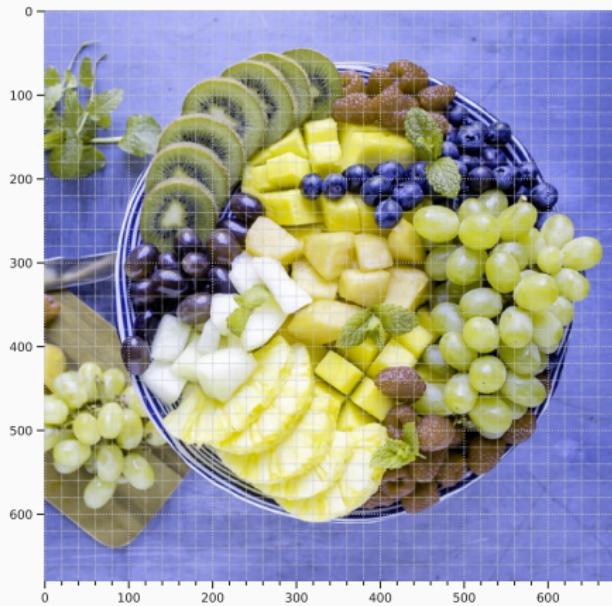
- Can see no colour at all and their world consists of different shades of grey ranging from black to white, rather like seeing the world on an old black and white television set [6].
- Achromatopsia is a specific eye condition in which people see in grey-scale.
- In rare cases, partial Achromatopsia can happen which is a **reduced** sensitivity to all three (3) cones [6].



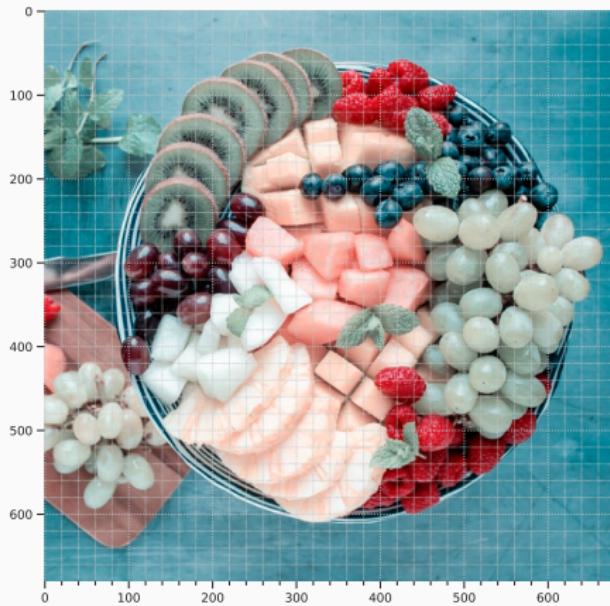
**Figure:** Image viewed by someone who has 3 cones.



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



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



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

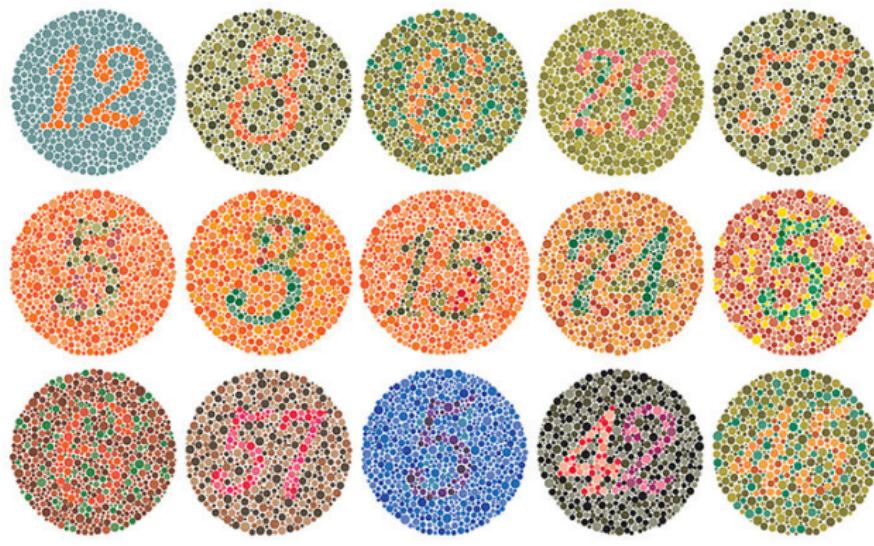


Figure: A colour blindness test issued to generally test before taking the driving license [7].

## Introduction

Brightness Sensitivity

D. T. McGuiness, Ph.D

- 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  $I(\lambda)$ .
  - assume  $I(\lambda) = I_0$  as a constant.

## Wavelength Sensitivity

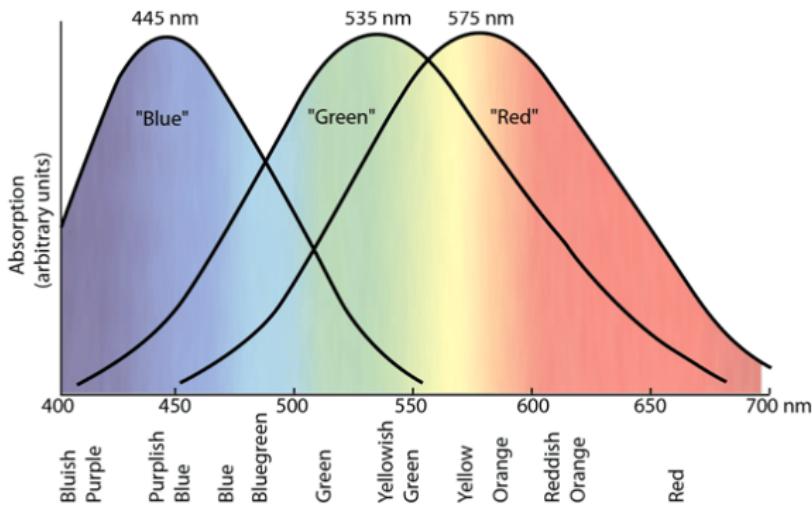
- The sensitivity of the human eye to light of a certain intensity varies strongly over wavelengths between 380 nm and 800 nm [8].
- Under daylight conditions, human eye is most sensitive at a wavelength of 555 nm, resulting in the fact that green light at produces the impression of highest “brightness” when compared to light at other wavelengths [8].

## Introduction

Brightness Sensitivity

D. T. McGuiness, Ph.D

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



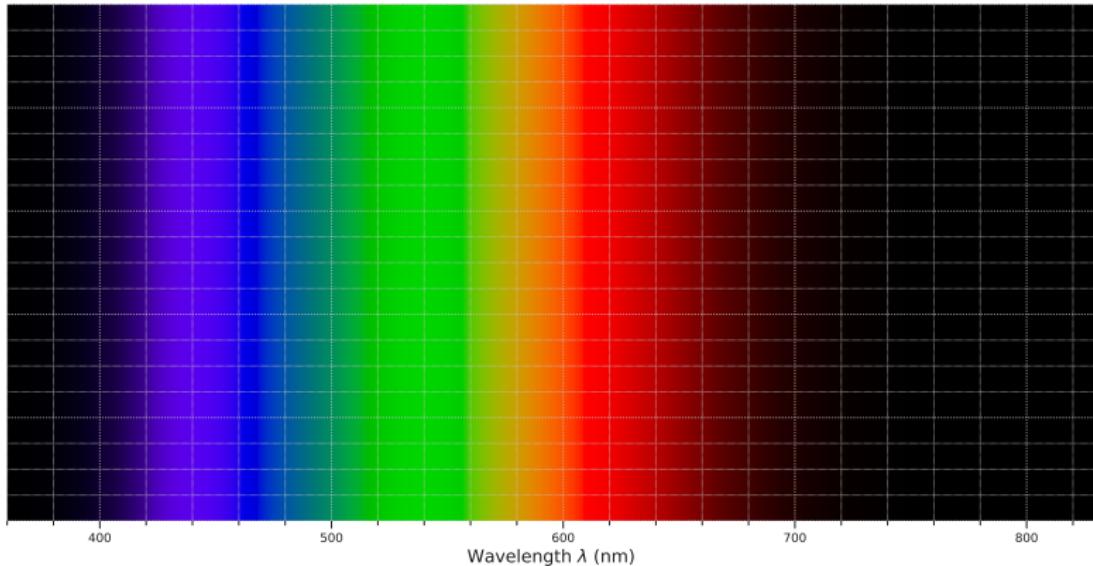
**Figure:** The colour sensitivity of the human eye [9].

# Introduction

Brightness Sensitivity

D. T. McGuiness, Ph.D

The Visible Spectrum - CIE 1931 2° Standard Observer



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

## Introduction

Stimulus Sensitivity

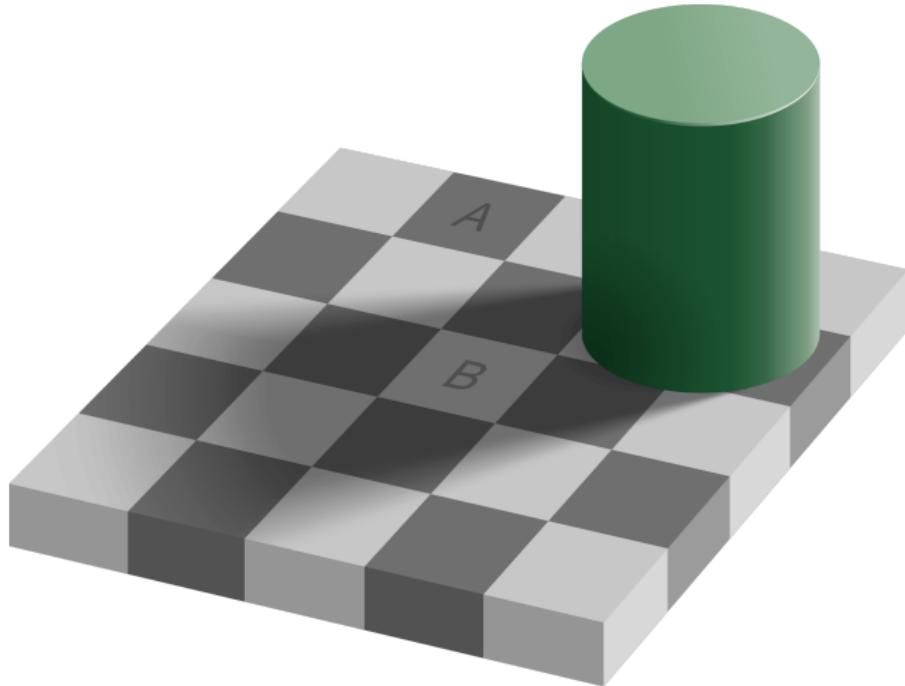
D. T. McGuiness, Ph.D

- 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 [10].

Relates to human perception, specifically the relation between the actual change in a physical stimulus and perceived change.

$$R = \log(I_0)$$

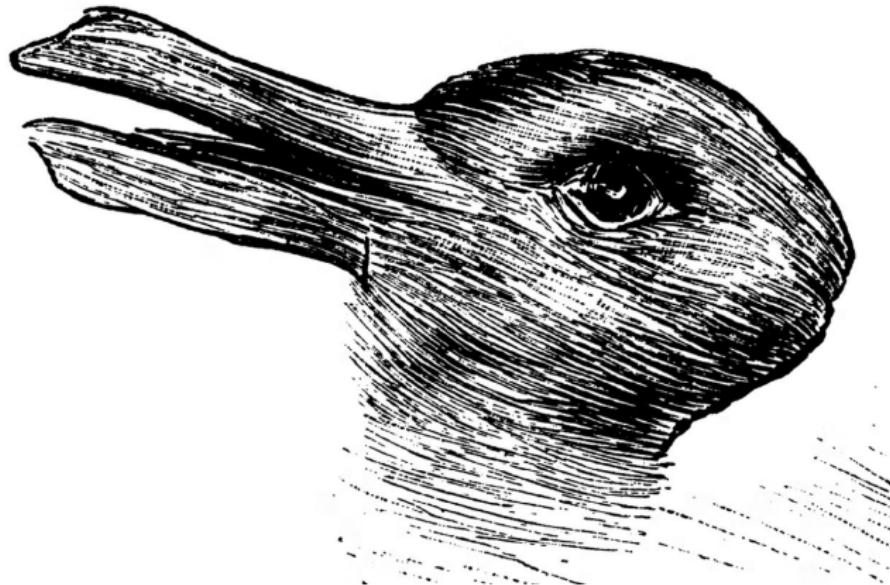
- This means, equal perceived steps in brightness,  $\Delta R = k$ , require the physical brightness (i.e., the stimulus) to increase exponentially.



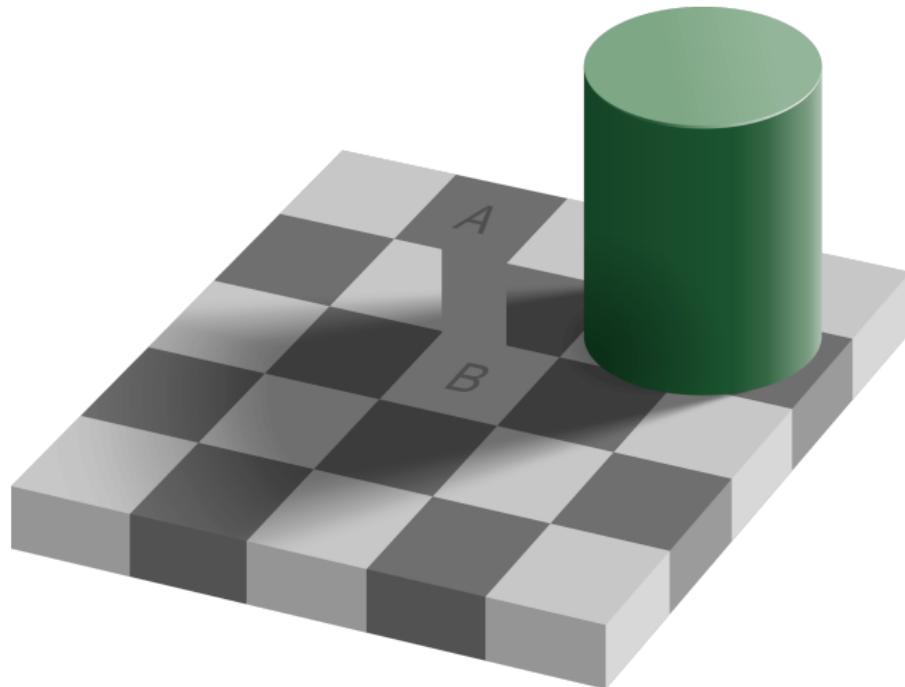
**Figure:** The checker shadow illusion [11].



**Figure:** You finally found someone. But who do you see? [12]



**Figure:** What do you see? A duck or a rabbit? [13]



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



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



**Figure:** A simple picture of a dress divided the internet [14].

## Introduction

Colour Sensitivity

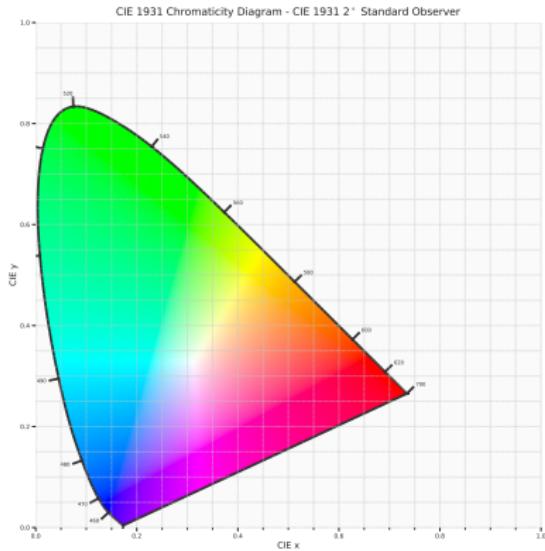
D. T. McGuiness, Ph.D

- Human colour perception is complex,
  - We can approximate its behaviour.
- **Standard Observer:** Based on psychophysical measurements, standard curves have been adopted by the CIE as sensitivity curves for the **typical** observer for three **pigments**:  $\bar{x}(\lambda)$ ,  $\bar{y}(\lambda)$ , and  $\bar{z}(\lambda)$ .

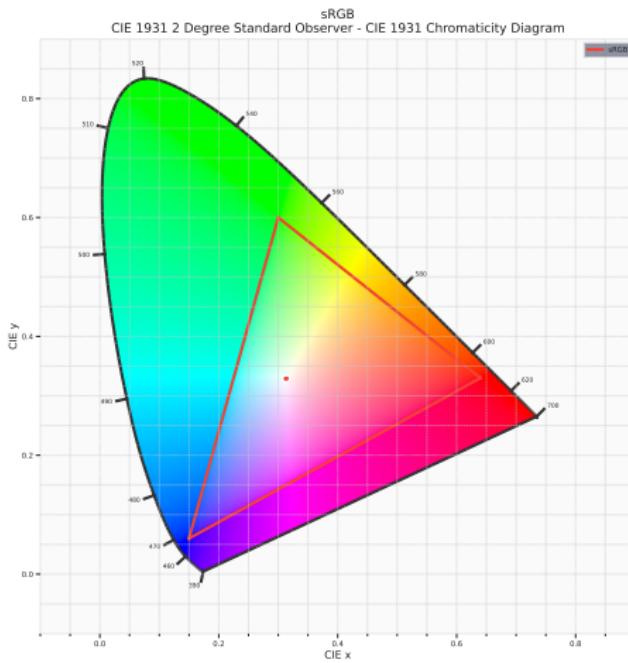
These are not pigment absorption characteristics found in human retina but rather sensitivity curves derived from actual data.

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

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



- A standard RGB colour-space defined by both HP and Microsoft in 1996 to use on monitors, printers, and the World Wide Web [15].
- It was subsequently standardised by IEC as IEC 61966-2-1:199 [16].
- 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:** The sRGB colour-space superimposed to the CIE colour-gamut.

## Colour Standards

sRGB

D. T. McGuiness, Ph.D

- 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 [17].
- However, consumer-level Charge Coupled Device (CCD)s 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.

## Colour Standards

Wide Gamut RGB

D. T. McGuiness, Ph.D

- 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 [18].
- 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% [19].



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

## Colour Standards

Prophoto RGB

D. T. McGuiness, Ph.D

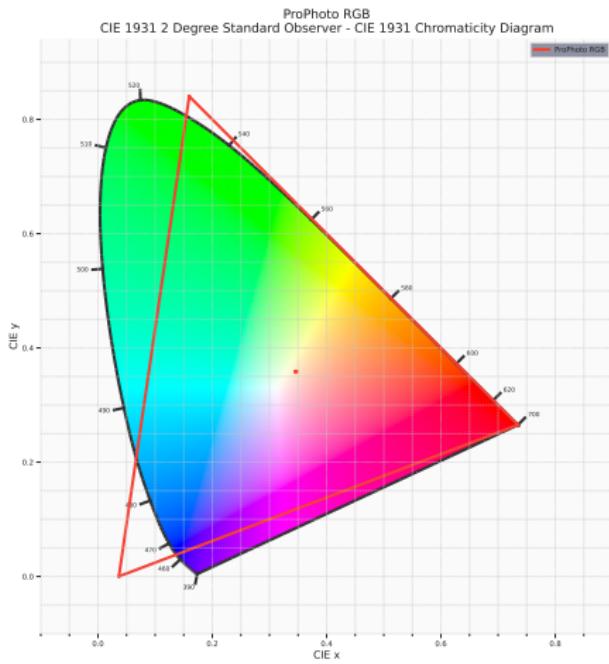
- The ProPhoto RGB colour space, a.k.a. ROMM RGB (Reference Output Medium Metric), is an output referred RGB color space developed by Kodak [20].
- Offers an especially **large gamut** designed for use with photographic output in mind.
- The gamut encompasses over 90% of possible color space, and 100% of likely occurring real-world surface colours making ProPhoto even larger than the Wide-gamut RGB color space [21].
- The ProPhoto RGB primaries were also chosen in order to minimise hue rotations associated with non-linear tone scale operations.

A downside is that approximately 13% of the visible colours are imaginary colors that do not exist and are impossible colour.

# Colour Standards

Prophoto RGB

D. T. McGuiness, Ph.D



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

## Colour Standards

Adobe RGB

D. T. McGuiness, Ph.D

- 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 then standardised by the IEC as IEC 61966-2-5:1999 with a name opRGB (optional RGB color space) and is used in HDMI [22].

# Colour Standards

CIE Chromaticity Coordinates

D. T. McGuiness, Ph.D

- 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:

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

- 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.

## Colour Standards

Chromaticity

D. T. McGuiness, Ph.D

- 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.

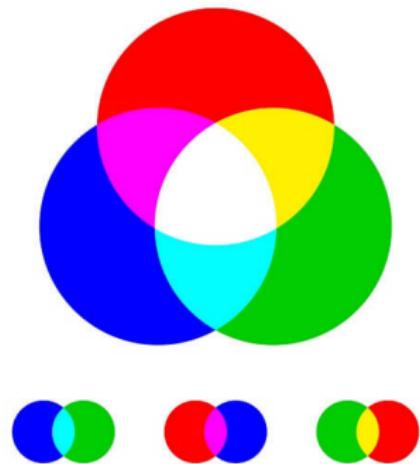
## Colour Models

### CYMK Colour Model

D. T. McGuiness, Ph.D

- 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

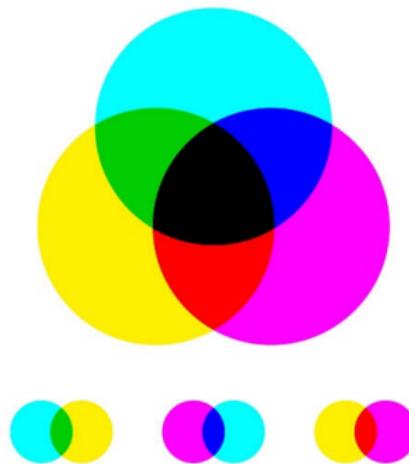
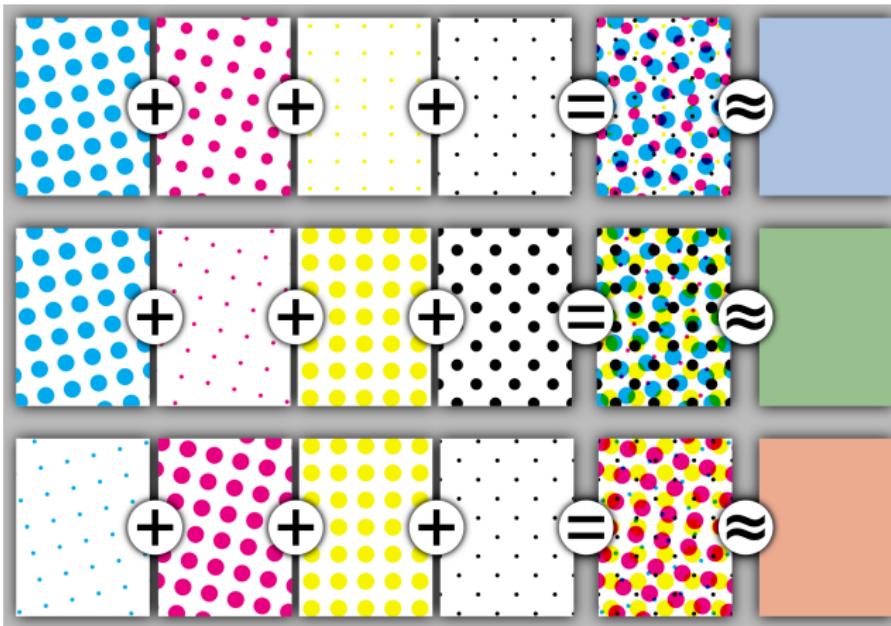


Figure: The differences between RGB and CMYK colours [23].

# Colour Models

## CYMK Colour Model

D. T. McGuiness, Ph.D

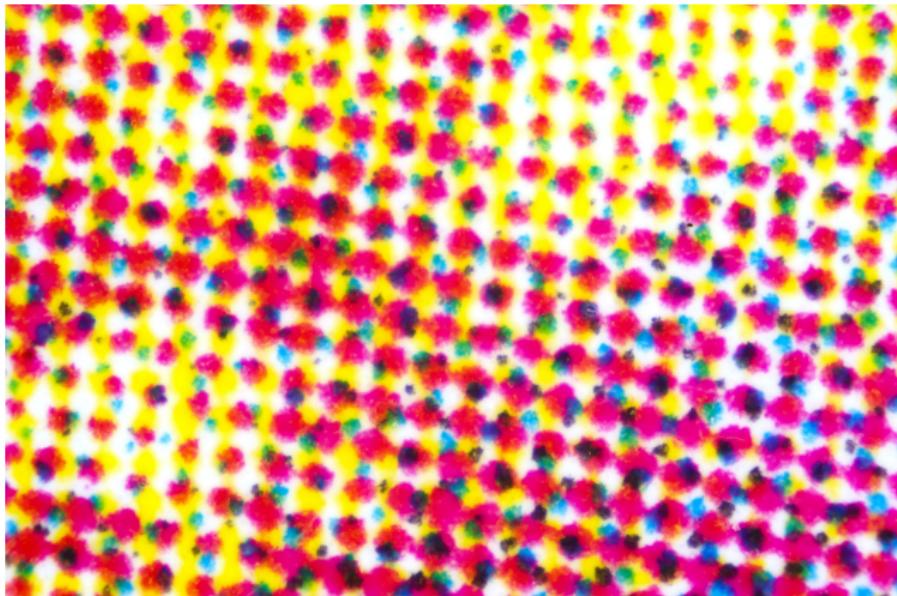


**Figure:** Three examples of color halftoning with CMYK separations, as well as the combined halftone pattern and how the human eye would observe the combined halftone pattern from a sufficient distance [24].

# Colour Models

CYMK Colour Model

D. T. McGuiness, Ph.D



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

## Colour Models

HSL and HLV Colour Model

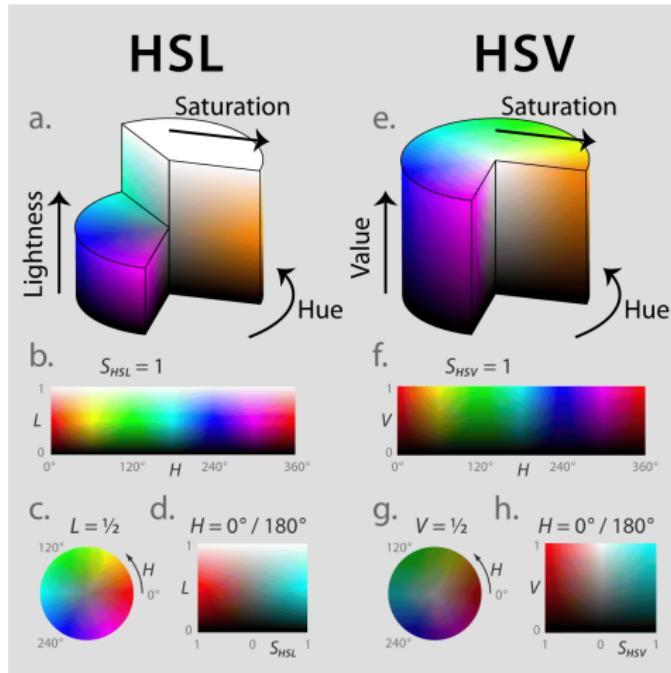
D. T. McGuiness, Ph.D

- 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.

# Colour Models

## HSL and HLV Colour Model

D. T. McGuiness, Ph.D



**Figure:** 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 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.

- $Y' C_b C_r$  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.

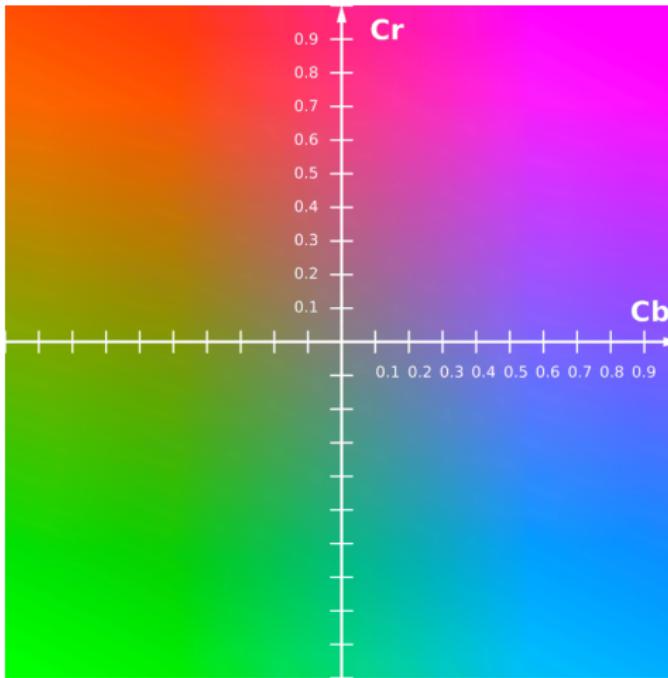


Figure: The  $Y'C_bC_r$  plane at constant luma  $Y=0.5$  [25].



Chapter

# Image Formats

## Learning Outcomes

### Introduction

File Formats

A General Overview

### Compression Methods

Lossy Compression

### Important File Formats

RAW File

JPEG

### JPEG Compression

Down-sampling

Isolate the Colour Information

Throw Away some Colour Information

Convert Image to Frequency Domain  
Quantisation

Lossless Data Compression

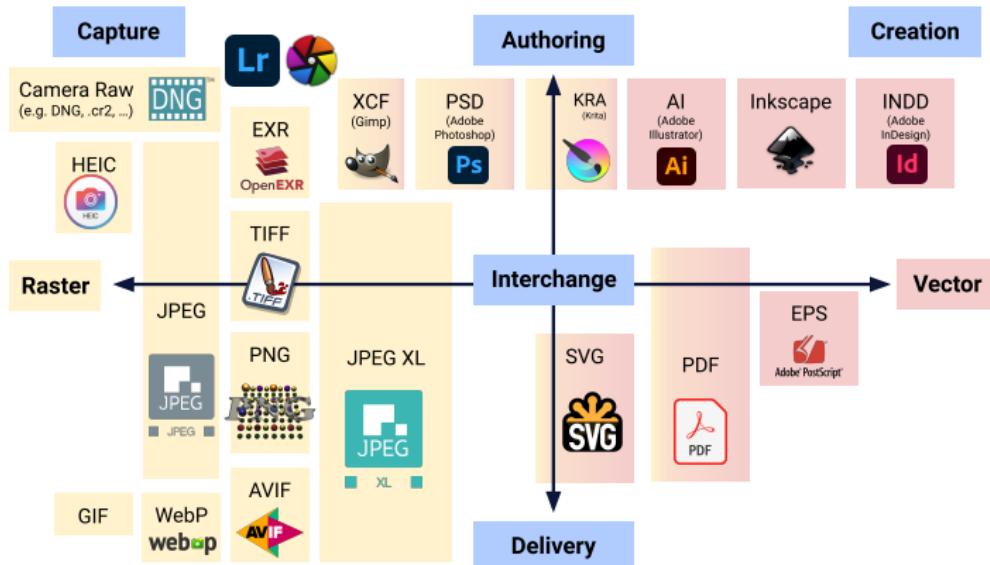
- (LO1) Types of File Formats used,
- (LO2) Defining Compression and its Types,
- (LO3) A Look into Selective File Types.
- (LO4) A Deep Dive into JPEG Compression.



# Introduction

File Formats

D. T. McGuiness, Ph.D



**Figure:** Categorization of image formats by scope which are used in commercial, industrial and personal use [26].

## 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.

## Compression Methods

### Lossy Compression

D. T. McGuiness, Ph.D

- 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 [27].
- 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.

## Compression Methods

### Lossy Compression

D. T. McGuiness, Ph.D

- 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 [28].
- 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 [29].

## Important File Formats

RAW File

D. T. McGuiness, Ph.D

- 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**.

## Important File Formats

JPEG

D. T. McGuiness, Ph.D

- 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.

## Important File Formats

JPEG

D. T. McGuiness, Ph.D



**Figure:** A photo of a European wildcat with the compression rate, and associated losses, decreasing from left to right [30].

## Important File Formats

JPEG

D. T. McGuiness, Ph.D

**Figure:** A photo of a European wildcat with the compression rate, and associated losses, decreasing from left to right [30].



## JPEG Compression

Down-sampling

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

Down-sampling

D. T. McGuiness, Ph.D

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.

# JPEG Compression

Isolate the Colour Information

D. T. McGuiness, Ph.D

- 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).

## JPEG Compression

Throw Away some Colour Information

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

Convert Image to Frequency Domain

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

Convert Image to Frequency Domain

D. T. McGuiness, Ph.D

- 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.

# JPEG Compression

Convert Image to Frequency Domain

D. T. McGuiness, Ph.D

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

## JPEG Compression

### Quantisation

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

### Quantisation

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

Lossless Data Compression

D. T. McGuiness, Ph.D

- 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.

## JPEG Compression

Lossless Data Compression

D. T. McGuiness, Ph.D

- 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.



# Chapter

# Camera

## Learning Outcomes

### Camera Sensors

- Charge Coupled Devices
- CCD Limitations
- Complementary MOS
- CCD v. CMOS
- Colour Filters

### Camera Properties

- Linearity

Sensitivity

Signal-to-Noise Ratio

Thermal Noise (Dark Current)

Quantum Efficiency

ISO

### Camera Lenses

Introduction

Aperture

Types of Lenses

Prime Lens

## Learning Outcomes

D. T. McGuiness, Ph.D

- (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 [31],

**sCMOS** Back-illuminated CMOS [32].



**Figure:** The first digital camera was invented in the 1970s by Kodak [33].

## Camera Sensors

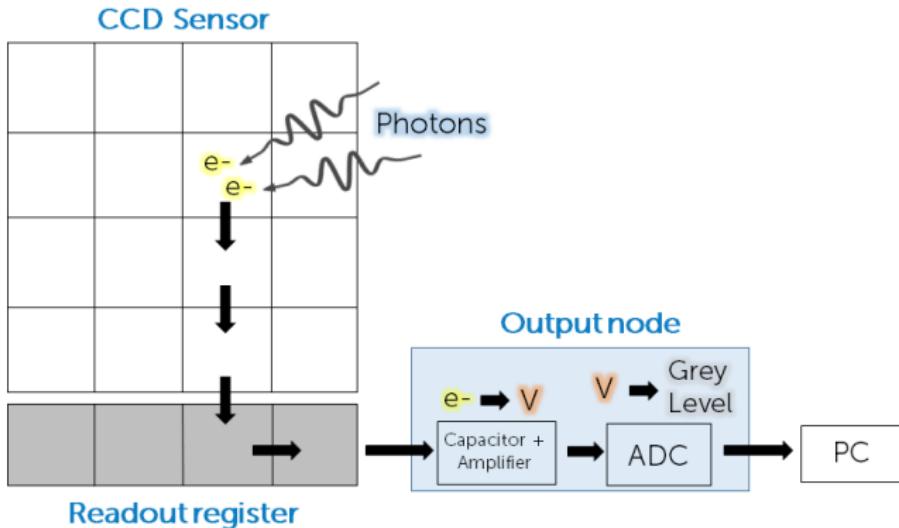
Charge Coupled Devices

D. T. McGuiness, Ph.D

- First sensors used in digital cameras, available since 1970s [34].
- 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:** 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.  
[34]

## Camera Sensors

### CCD Limitations

D. T. McGuiness, Ph.D

- 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.

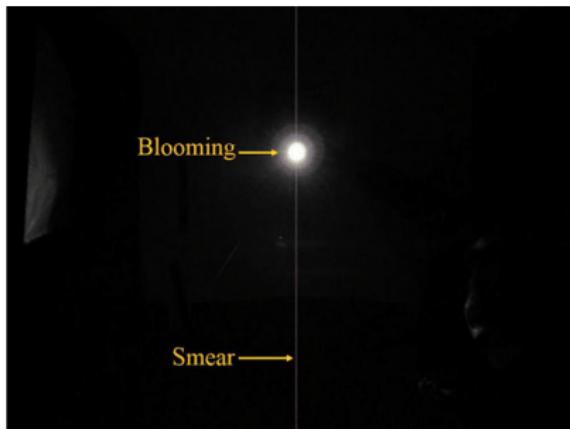
**Refresh** Whole sensor needs to be cleared of the electron signal before the next frame can be exposed

- 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.

## Camera Sensors

CCD Limitations

D. T. McGuiness, Ph.D



**Figure:** 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 [34].

## Camera Sensors

Complementary MOS

D. T. McGuiness, Ph.D

- 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 [35].
- 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 [36].

# Camera Sensors

Complementary MOS

D. T. McGuiness, Ph.D

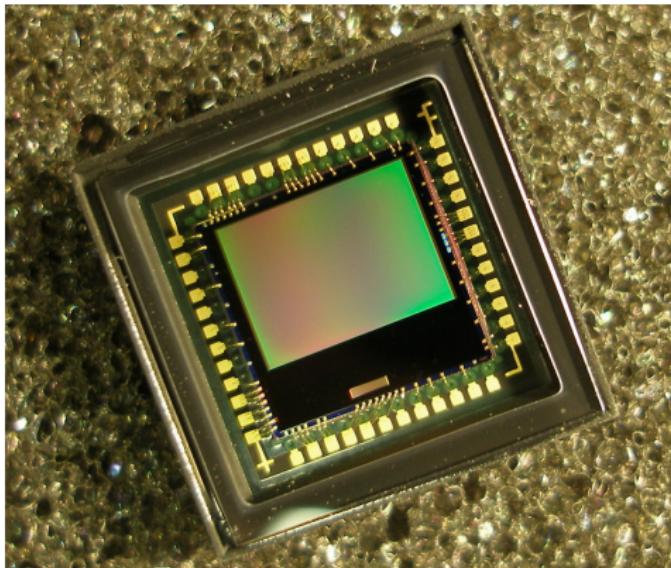
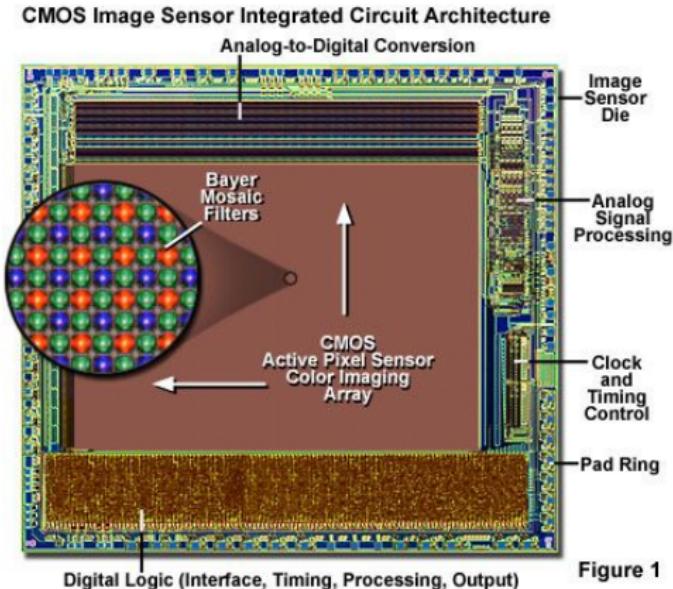


Figure: CMOS image sensor [37].



**Figure 1**

**Figure:** An integrated CMOS Image sensor [38].

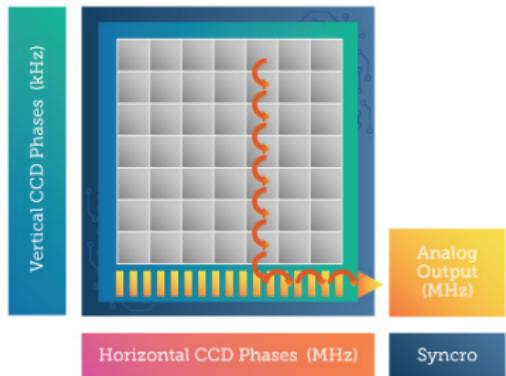
# Camera Sensors

Complementary MOS

D. T. McGuiness, Ph.D

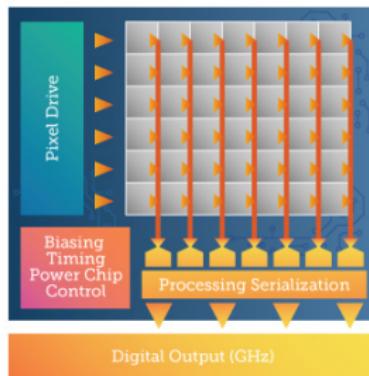
CCD

Photon to Electron  
Conversion (Analog)



CIS

Photon to Voltage  
Conversion (Digital)



**Figure:** A comparison on the working principle of two types of sensors: CCD and CMOS [39].

## Camera Sensors

CCD v. CMOS

D. T. McGuiness, Ph.D.

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.

## Camera Sensors

CCD v. CMOS

D. T. McGuiness, Ph.D.

- 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.

## Camera Sensors

### Colour Filters

D. T. McGuiness, Ph.D

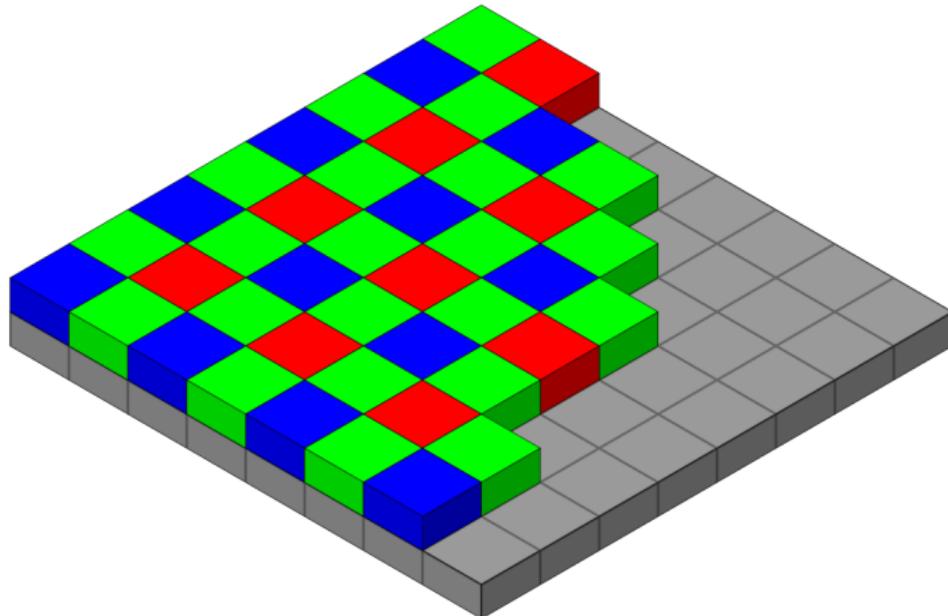
- A color filter array (CFA), or color filter mosaic (CFM), is a mosaic of tiny color filters placed over the pixel sensors of an image sensor to capture color information.
- Color filters are needed because the typical photosensors detect light intensity with little or no wavelength specificity and therefore cannot separate color information [40].

Diazonaphthoquinone (DNQ)-novolac photoresist is one material used as the carrier for making color filters from color dyes or pigments [41, 42].

# Camera Sensors

## Colour Filters

D. T. McGuiness, Ph.D

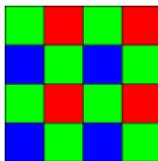


**Figure:** The Bayer color filter mosaic. Each two-by-two submosaic contains 2 green, 1 blue, and 1 red filter, each filter covering one pixel sensor. The reason as to why there are two greens to red and blue is to mimic the physiology of the human eye [43].

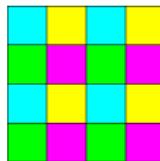
# Camera Sensors

## Colour Filters

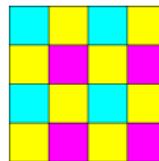
D. T. McGuiness, Ph.D



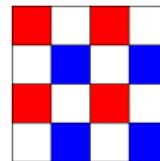
(a) Bayer Filter



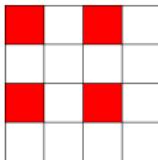
(b) CYGM Filter



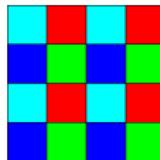
(c) CYYM Filter



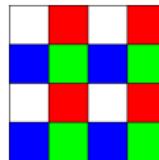
(d) RCCB Filter



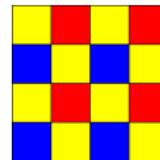
(e) RCCC Filter



(f) RGBE Filter



(g) RGBW Filter



(h) RYYB Filter

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

# Camera Sensors

Colour Filters

D. T. McGuiness, Ph.D



**Figure:** Sony F828, produced in 2003 which uses an RGBE filter. Once the image is taken a demosaicing algorithm is used to reconstruct an image [44].

## Camera Properties

### Linearity

D. T. McGuiness, Ph.D

- It is generally desirable the relationship between the input physical signal and the output signal be linear [45].
- 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.

## Camera Properties

Linearity

D. T. McGuiness, Ph.D

- 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.

## Camera Properties

Sensitivity

D. T. McGuiness, Ph.D

- 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: 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 [46].
- 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.

## Camera Properties

Sensitivity

D. T. McGuiness, Ph.D

1. 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.
2. If the limiting effect of the camera is only the **photon noise**, then:

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

## Camera Properties

Sensitivity

D. T. McGuiness, Ph.D.

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:** Sensitivity measurements of sample cameras.

## Camera Properties

Sensitivity

D. T. McGuiness, Ph.D

- 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.

## Camera Properties

Signal-to-Noise Ratio

D. T. McGuiness, Ph.D

- 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**.

## Camera Properties

### Thermal Noise (Dark Current)

D. T. McGuiness, Ph.D

- Caused by the thermal energy within the camera sensor [47].
- As sensor works, the heat from operation causes thermal energy generation.

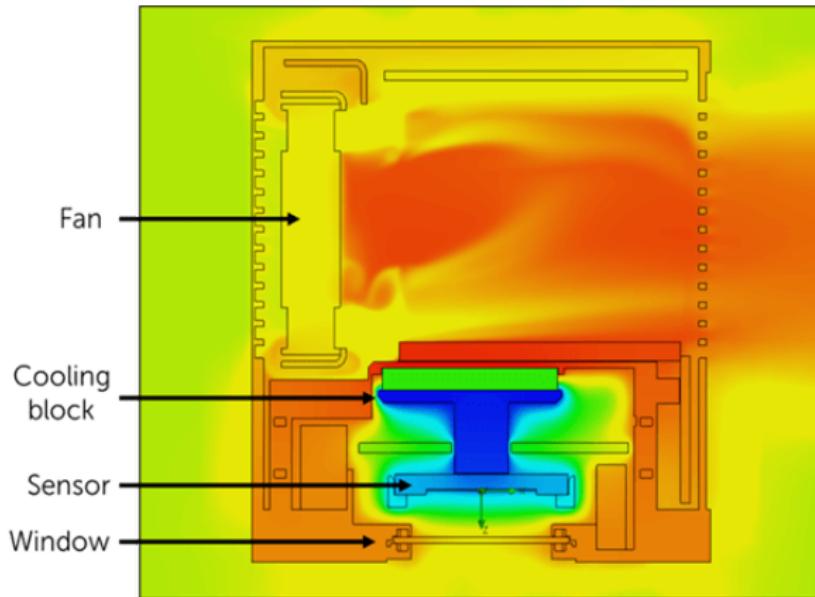
These thermal electrons are independent of the photo-electrons.

- 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.

## Camera Properties

Thermal Noise (Dark Current)

D. T. McGuiness, Ph.D



**Figure:** 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 [47]).

## Camera Properties

Thermal Noise (Dark Current)

D. T. McGuiness, Ph.D

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: Values of Dark Current on Cameras.

## Camera Properties

### Quantum Efficiency

D. T. McGuiness, Ph.D

- 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.

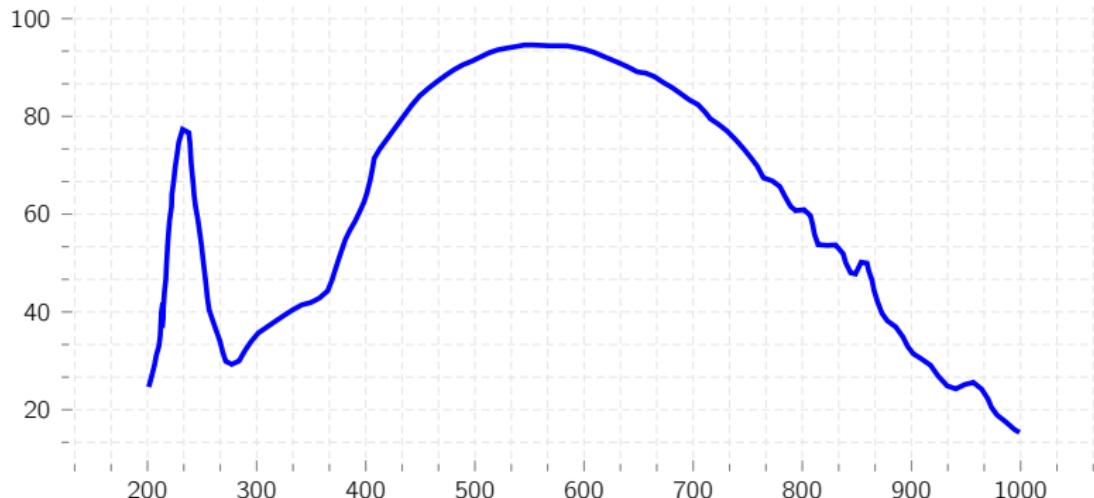
- In practice, sensors are never 100% efficient, and different sensor technologies have different QE values.

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

## Camera Properties

Quantum Efficiency

D. T. McGuiness, Ph.D



**Figure:** 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.

## Camera Properties

ISO

D. T. McGuiness, Ph.D



**Figure:** 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 [48].

## Camera Lenses

Introduction

D. T. McGuiness, Ph.D

- 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.

# Camera Lenses

Introduction

D. T. McGuiness, Ph.D

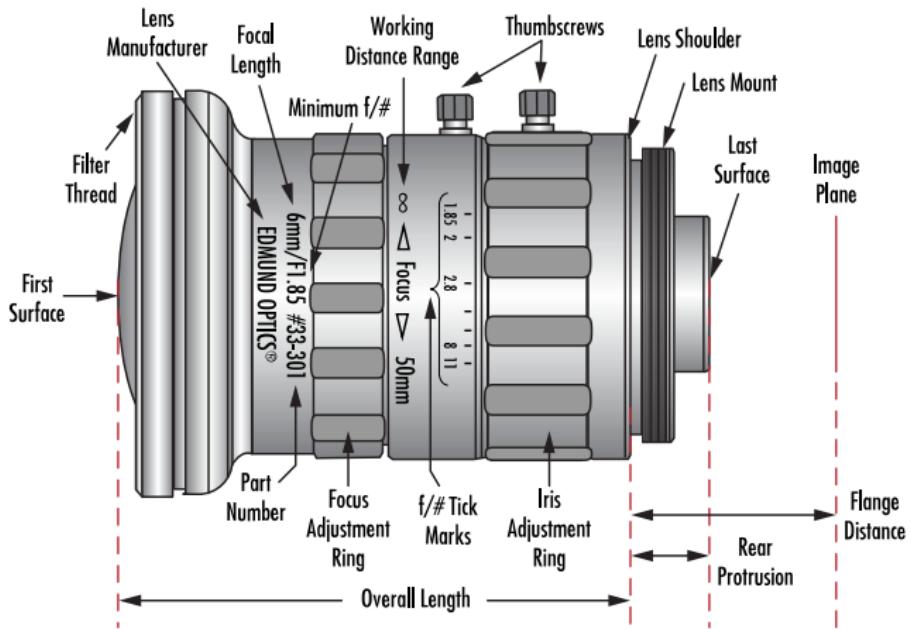


Figure: Anatomy of a lens [49].

# Camera Lenses

Aperture

D. T. McGuiness, Ph.D

$f/1.4$



$f/2.0$



$f/2.8$



$f/4.0$



$f/5.6$



$f/8.0$



**Figure:** Different apertures of a lens.

# APERTURE

LESS LIGHT



“SLOWER” ← → “FASTER”  
“SMALLER”                    “WIDER”

MORE LIGHT

Figure: The effects of aperture on the image.

## Camera Lenses

Aperture

D. T. McGuiness, Ph.D

**Figure:** 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 ( $f/2.1$ )

## Camera Lenses

### Types of Lenses

D. T. McGuiness, Ph.D

- 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.

# Camera Lenses

Types of Lenses

D. T. McGuiness, Ph.D

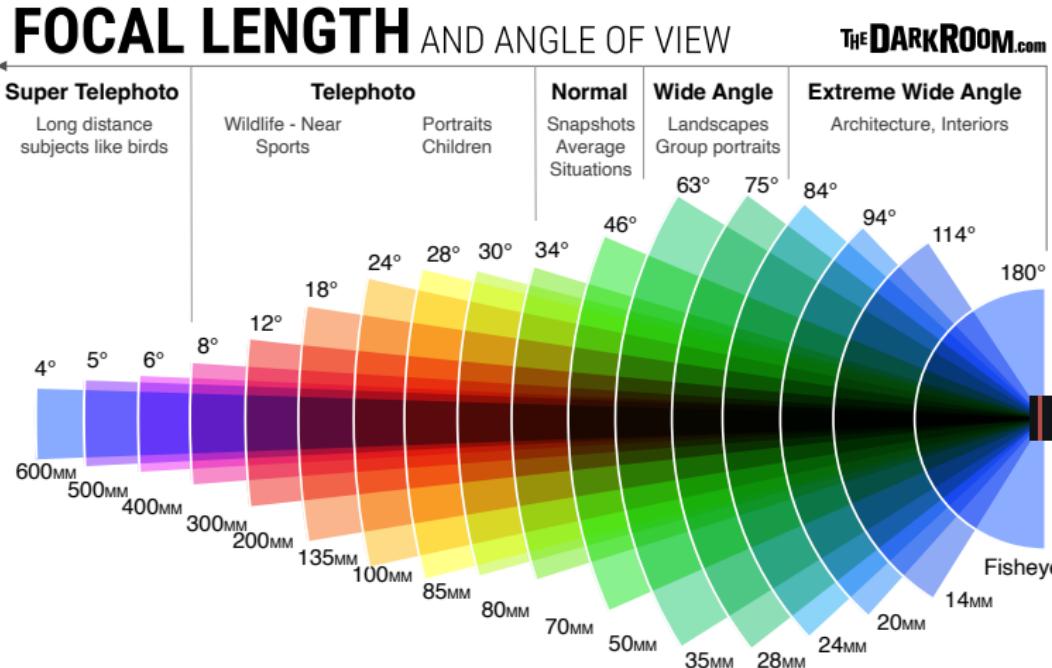


Figure: The effects of focal length and the angle of view [50].

## Camera Lenses

Types of Lenses

D. T. McGuiness, Ph.D

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

# Camera Lenses

## Types of Lenses

D. T. McGuiness, Ph.D

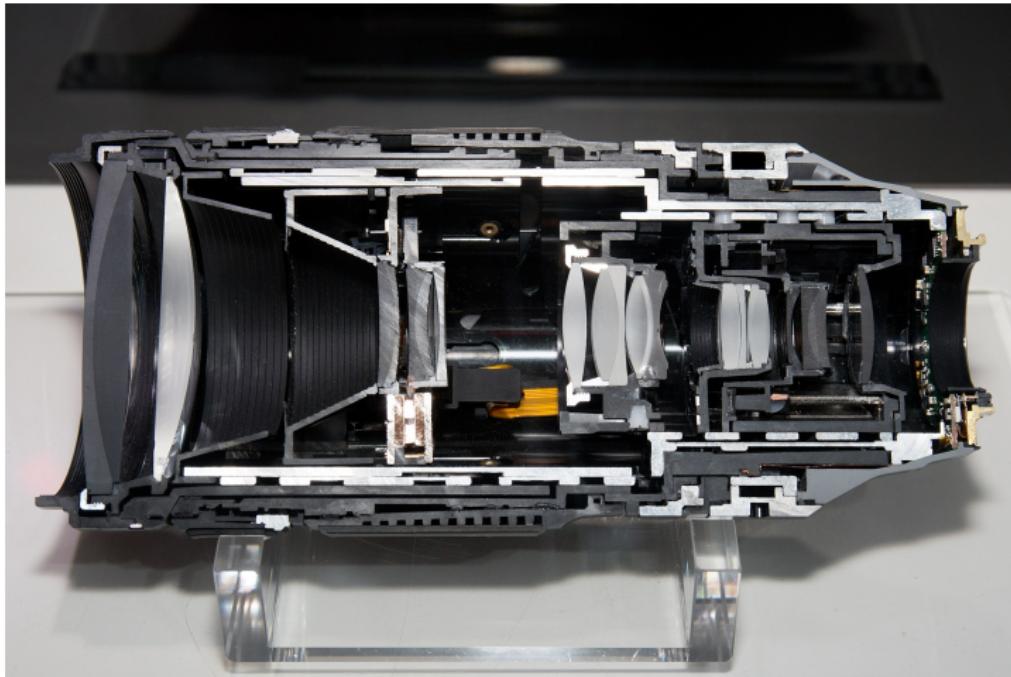
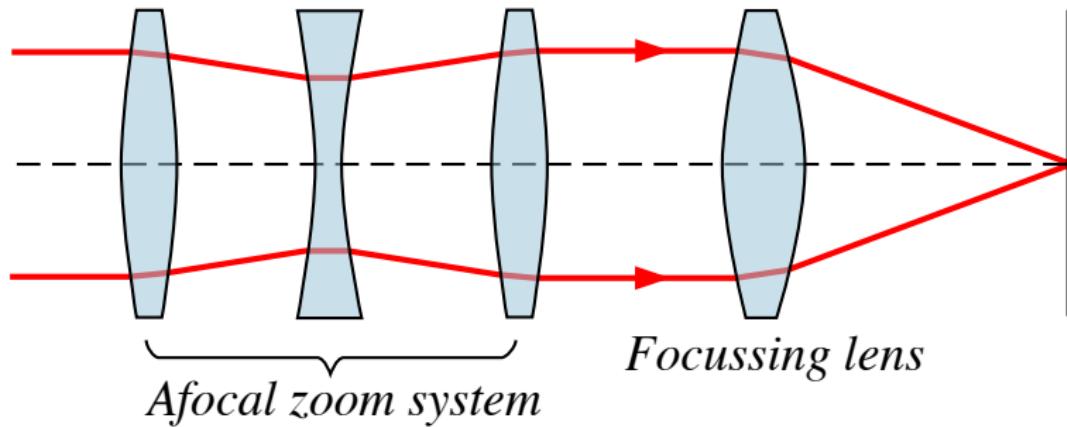


Figure: The cross-section of Fujinon XF100-400mm zoom lens [51].



**Figure:** A simple zoom lens system. The three lenses of the afocal system are  $L_1$ ,  $L_2$ ,  $L_3$  (from left).  $L_1$  and  $L_2$  can move to the left and right, changing the overall focal length of the system [52]

- 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:** Photographic example showing a high quality lens (top) compared to a lower quality one exhibiting transverse chromatic aberration [53].

## Camera Lenses

### Prime Lens

D. T. McGuiness, Ph.D

- 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.



Chapter  
**Display**

## Learning Outcomes

### Introduction

Refresh Rate

### Limits of Human Vision

Workings of the Eye

### Interlacing

Advantages

Disadvantages

Cathode Ray Tube Displays

LED Displays

LCD Display

### Analog TV Formats

NTSC

PAL

Worldwide Standards

Test Card

### Resolution

## Learning Outcomes

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



## Introduction

Refresh Rate

D. T. McGuiness, Ph.D

- Defined as the number of complete images that are written to the screen per second.
- For video, TV refresh rate is either 25 (PAL)[54] or 29.97 (NTSC) images.
- For computer displays, the refresh rate can vary with common values being 30 Hz, 60 Hz and 144 Hz.

## Introduction

Refresh Rate

D. T. McGuiness, Ph.D

- 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.

## Limits of Human Vision

Workings of the Eye

D. T. McGuiness, Ph.D

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.

## Limits of Human Vision

Workings of the Eye

D. T. McGuiness, Ph.D

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 [55].

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.

## Limits of Human Vision

Workings of the Eye

D. T. McGuiness, Ph.D

- 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 [56], 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 [57].

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 [58].
- In an interlaced display, the picture is made by scanning alternating lines [59].
- 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: An Example of Interleaving during a live-stream event [60].

## Interlacing

Advantages

D. T. McGuiness, Ph.D

### 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.

## Interlacing

Disadvantages

D. T. McGuiness, Ph.D

- 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.

## Interlacing

### Disadvantages

D. T. McGuiness, Ph.D

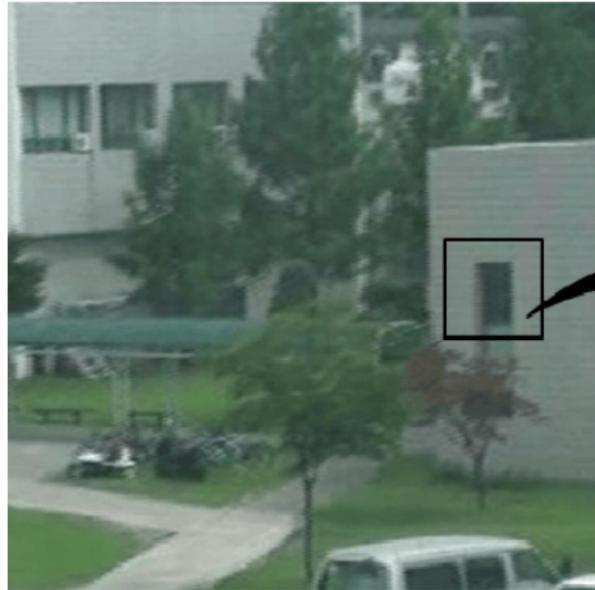
- Artefacts can be distracting and reduces clarity and quality.
- As only half of the image is displayed at a time, interlaced videos can appear less sharp and detailed compared to progressive scan videos, where each frame shows the full image.

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

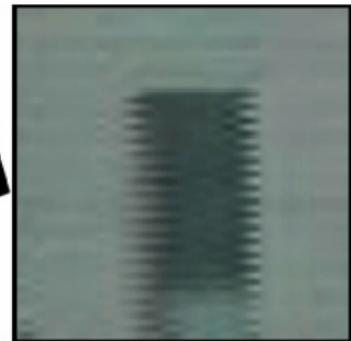
# Interlacing

Disadvantages

D. T. McGuiness, Ph.D



Screenshot



Combing artifacts

**Figure:** An example of the combing effect observed in an interlaced video [59].

# Interlacing

Cathode Ray Tube Displays

D. T. McGuiness, Ph.D



Figure: A monochrome CRT as seen inside a Macintosh Plus computer [61].

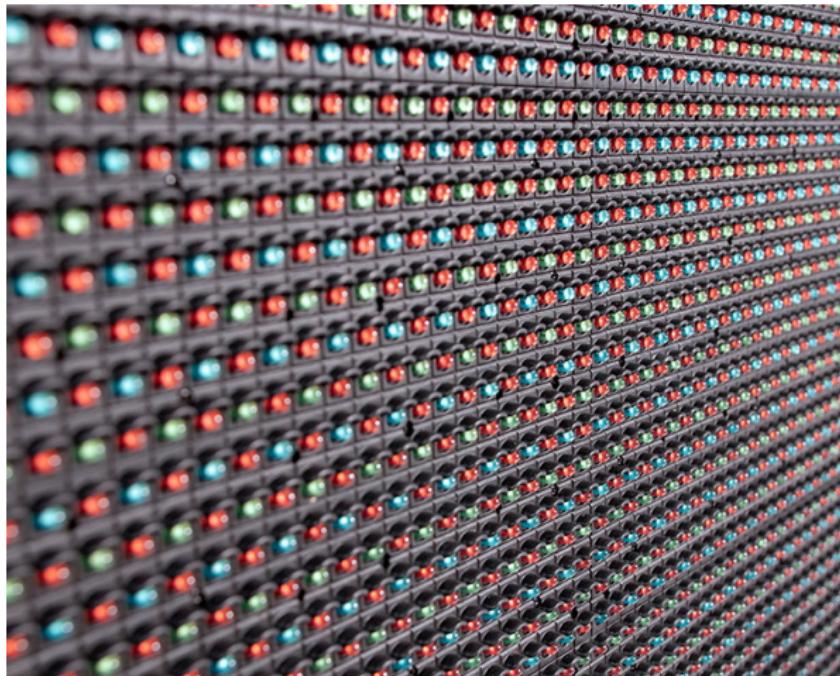
- An LED emits light as a result of electric luminescence and is one of the most energy-efficient and power-saving ways to produce light.
- Consists of solid materials without movable parts and is often moulded into transparent plastic.
- An LED display consists of many closely-spaced LEDs.
  - By varying 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.

# Interlacing

LED Displays

D. T. McGuiness, Ph.D



**Figure:** From a close-up it can be seen a display is made from RGB diodes [62].

## Interlacing

LCD Display

D. T. McGuiness, Ph.D

- 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.

# Interlacing

LCD Display

D. T. McGuiness, Ph.D

# LCD

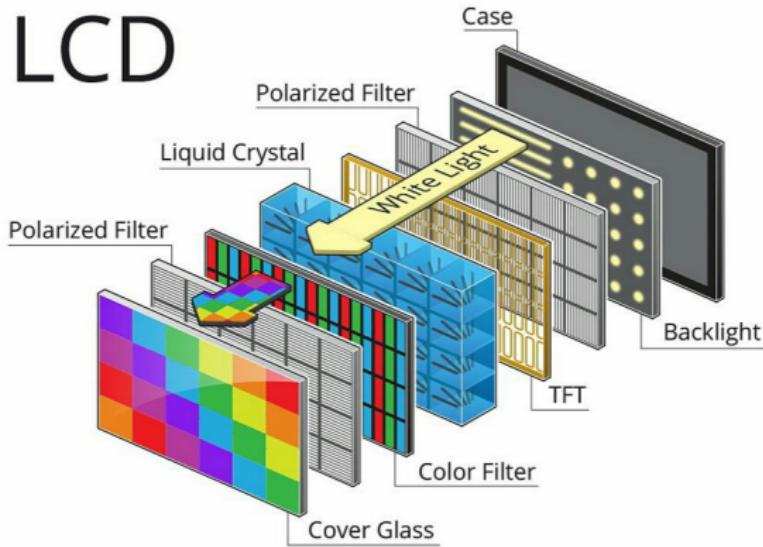


Figure: An exploded view of the LCD technology [63].

## 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.

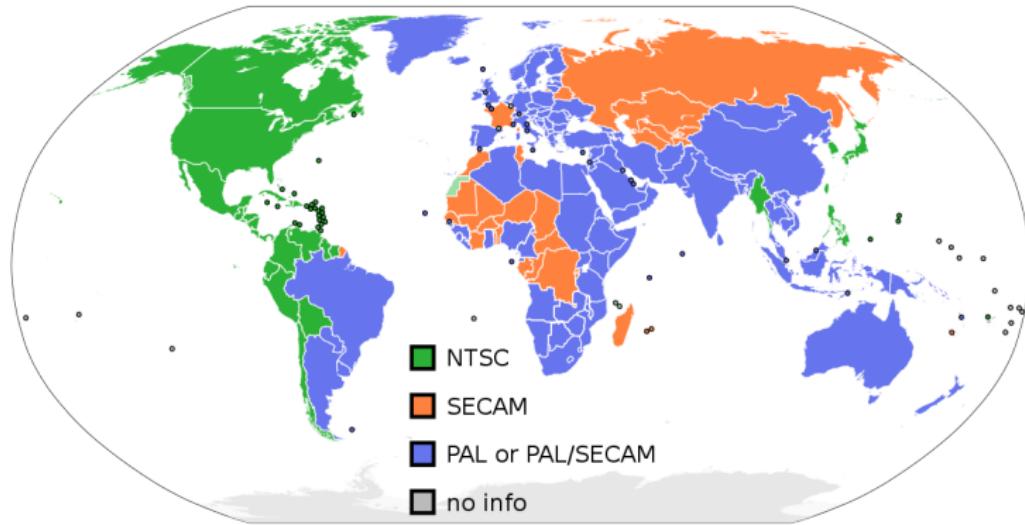
## PAL (Phase Alternating Line)

- 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.

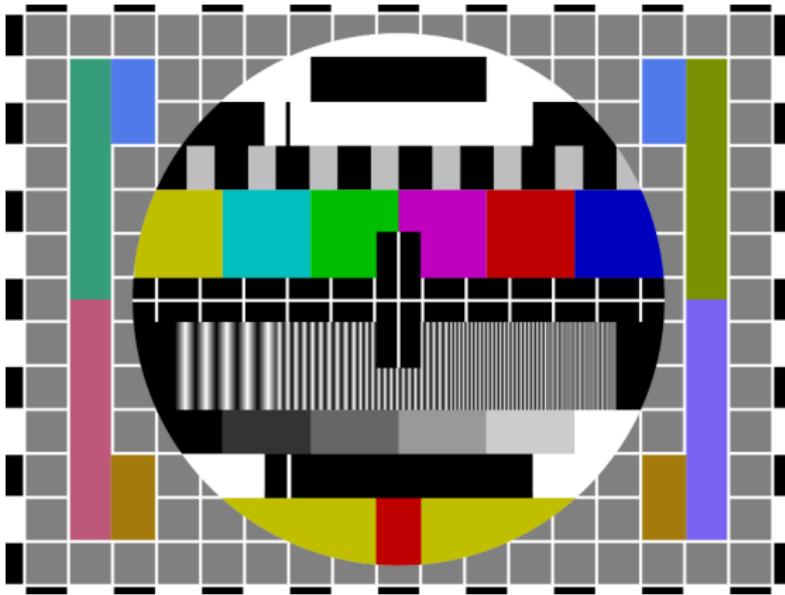
# Analog TV Formats

Worldwide Standards

D. T. McGuiness, Ph.D

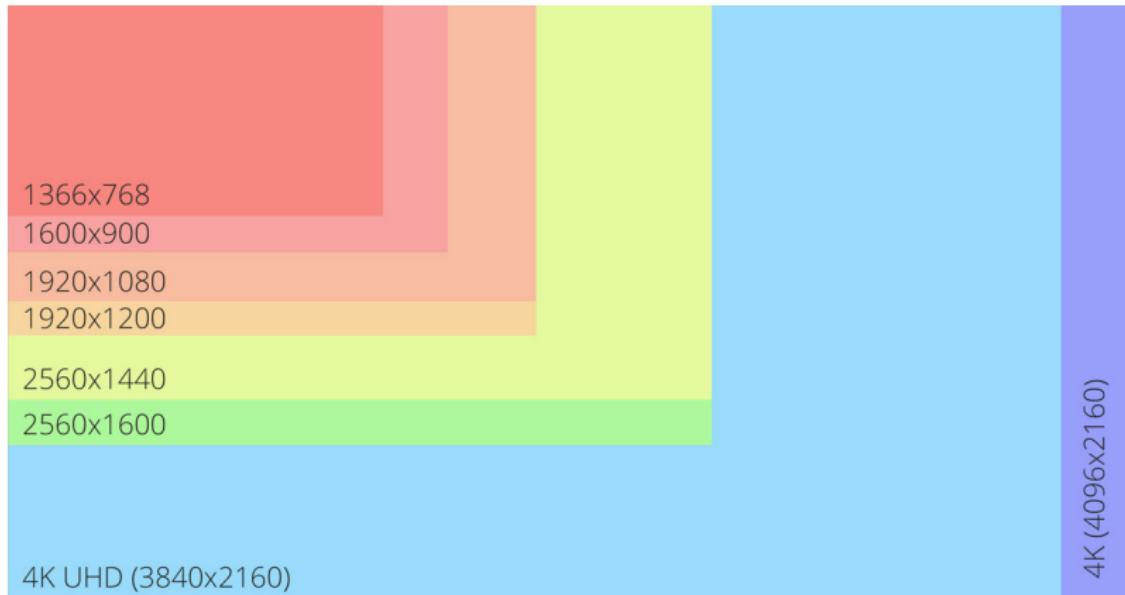


**Figure:** Analog television encoding systems by nation; NTSC (**green**), SECAM (Séquentiel de couleur à mémoire) (**orange**), and PAL (**blue**)



**Figure:** 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 [64].

- 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:** A comparison of different resolution standards [65].



Chapter  
**Noise**

## Introduction

Definition

## Colours of Noise

White Noise

Pink Noise

Brown Noise

Violet Noise

Blue Noise

## Photon Noise

Introduction

Poisson Distribution

Photon Distribution

## Thermal Noise

Definitions

Dark Current

## Read Noise

Introduction

CMOS Read Noise

Johnson-Nyquist Noise

Quantisation Noise

## Application

Perlin Noise

## Introduction

Definition

D. T. McGuiness, Ph.D

- 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**.

# Colours of Noise

D. T. McGuiness, Ph.D

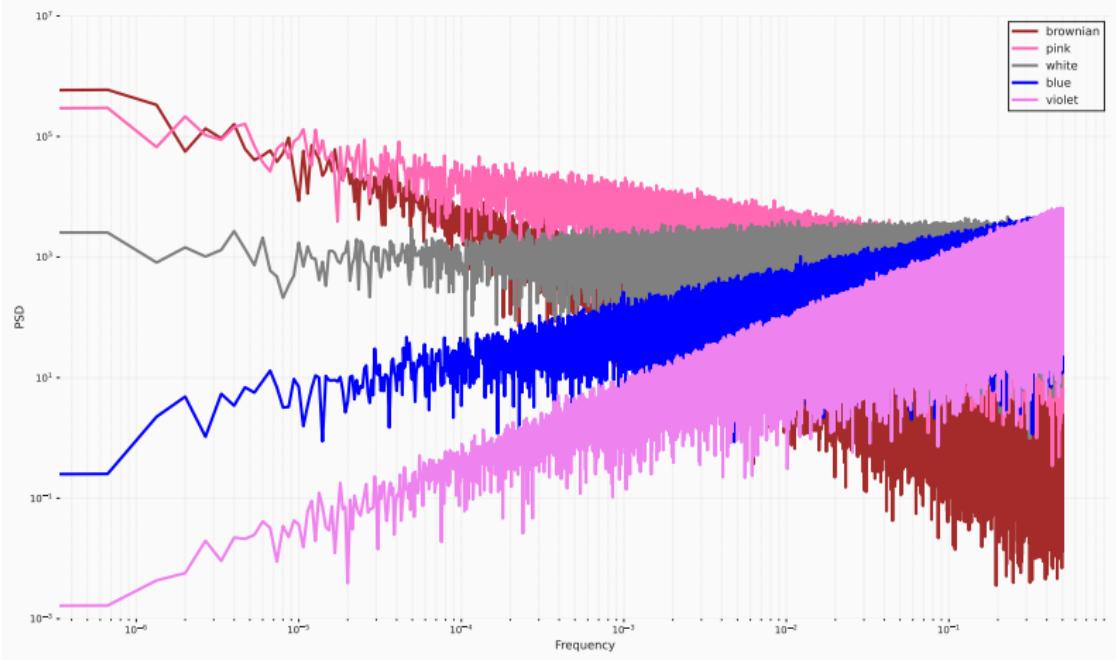


Figure: A comparison of different colours of Noise and their power vs. frequency.

## Colours of Noise

White Noise

D. T. McGuiness, Ph.D

- 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.

## Colours of Noise

White Noise

D. T. McGuiness, Ph.D

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);
```

C.R. 6.7  
python

Continuing on define a template function which will be inherited.

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

C.R. 6.8  
python

The aforementioned code uses **anonymous functions**.

# Colours of Noise

White Noise

D. T. McGuiness, Ph.D

C.R. 6.9  
python

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

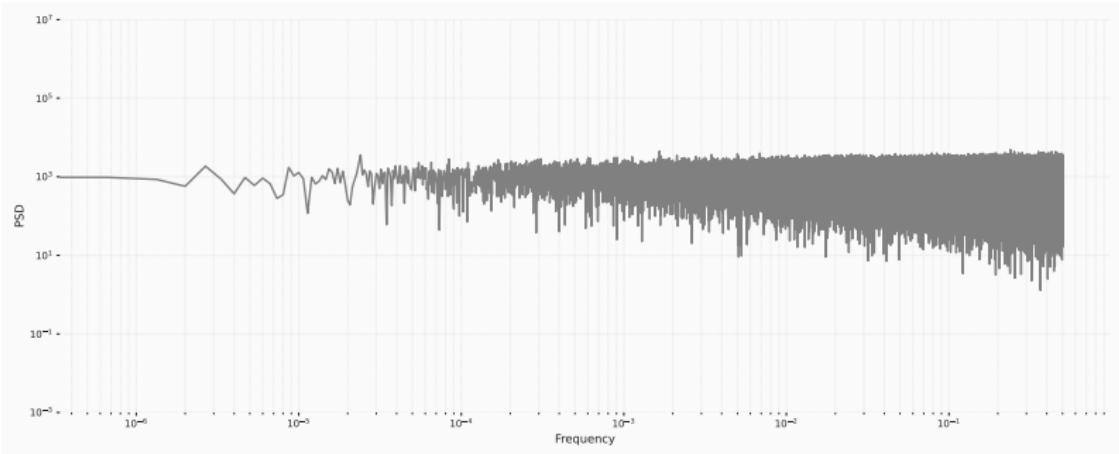


Figure: PSD of white noise.

## Colours of Noise

Pink Noise

D. T. McGuiness, Ph.D

- 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 [66].
- It is often observed signals in **biological systems**.
  - i.e., fluctuations in tide and river heights, quasar light emissions, heart beat.

# Colours of Noise

Pink Noise

D. T. McGuiness, Ph.D

C.R. 6.10  
python

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

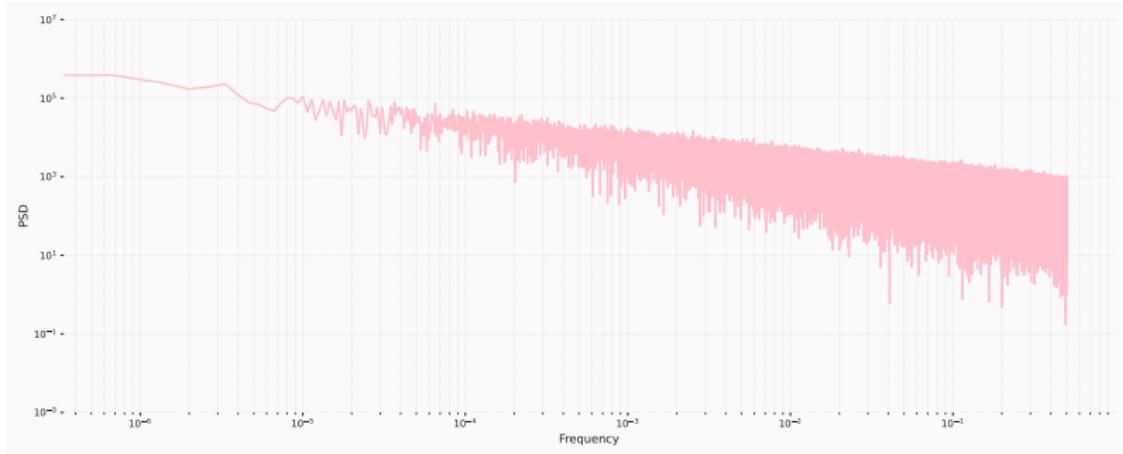


Figure: PSD of pink noise.

## Colours of Noise

Brown Noise

D. T. McGuiness, Ph.D

- 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.

# Colours of Noise

Brown Noise

D. T. McGuiness, Ph.D

C.R. 6.11

python

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

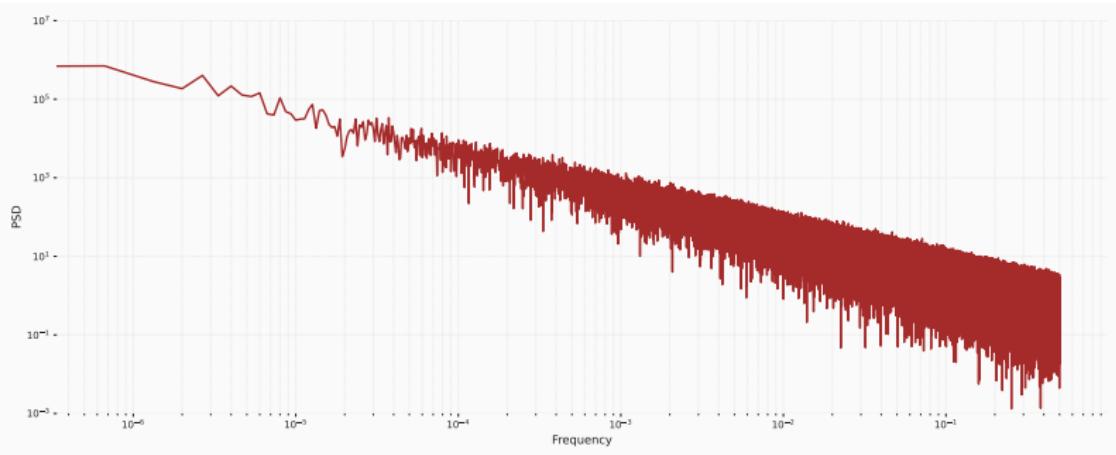


Figure: PSD of brown noise.

## Colours of Noise

Violet Noise

D. T. McGuiness, Ph.D

- 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.

# Colours of Noise

Violet Noise

D. T. McGuiness, Ph.D

C.R. 6.12  
python

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

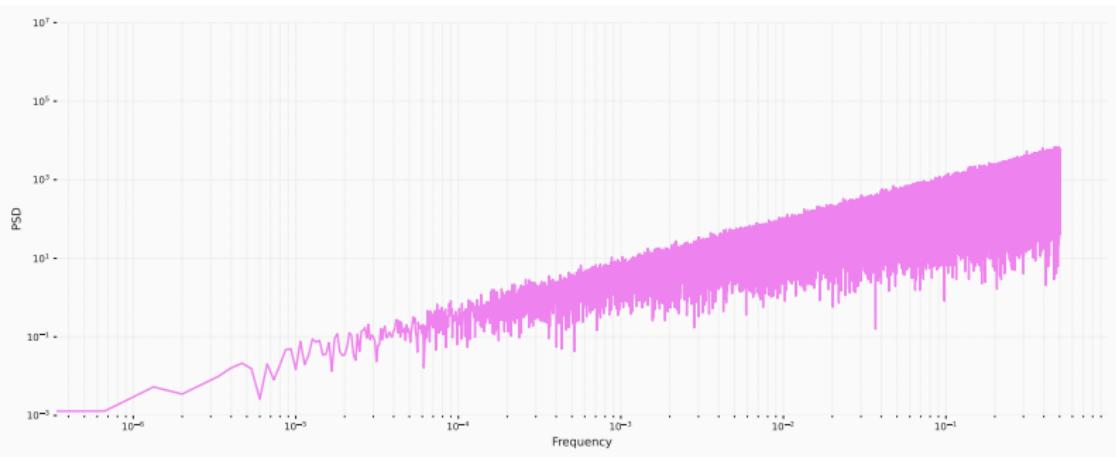


Figure: PSD of violet noise.

## Colours of Noise

Blue Noise

D. T. McGuiness, Ph.D

- 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.

# Colours of Noise

Blue Noise

D. T. McGuiness, Ph.D

C.R. 6.13  
python

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

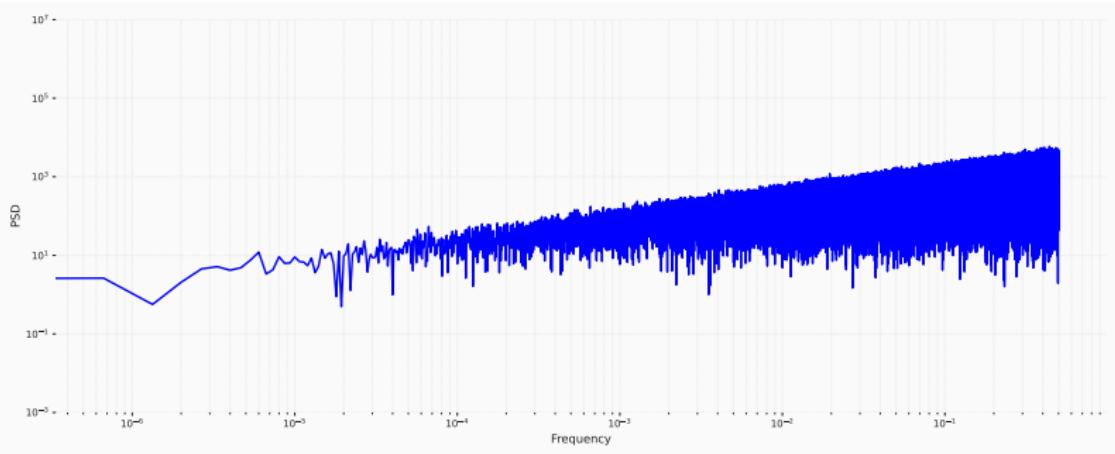


Figure: PSD of blue noise.

## Colours of Noise

Blue Noise

D. T. McGuiness, Ph.D

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

# Colours of Noise

Blue Noise

D. T. McGuiness, Ph.D



**Figure:** 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.

# Photon Noise

Introduction

D. T. McGuiness, Ph.D

Poisson Process for Photon Noise

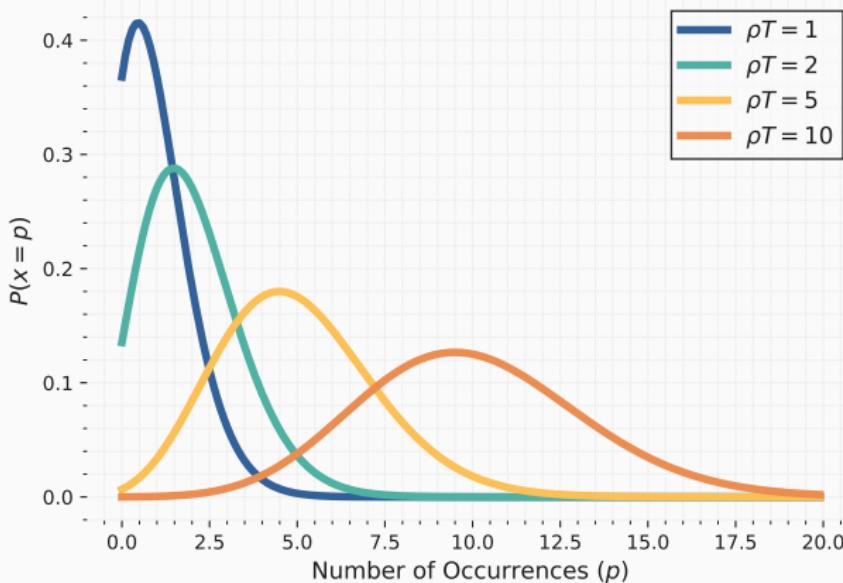


Figure: The distribution of the photon noise which is a Poisson process.

## Photon Noise

Poisson Distribution

D. T. McGuiness, Ph.D

- 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:

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

## Photon Noise

Poisson Distribution

D. T. McGuiness, Ph.D

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

Example

# Photon Noise

Poisson Distribution

D. T. McGuiness, Ph.D

## (i) No Job Arrivals

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

## (ii) 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:

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

Solution

## Photon Noise

### Photon Distribution

D. T. McGuiness, Ph.D

- 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:

$$\mu = \rho T \quad \sigma = \sqrt{\rho T}$$

$$\text{SNR} = 10 \log_{10} (\rho T) \quad \text{dB}$$

## Photon Noise

Photon Distribution

D. T. McGuiness, Ph.D

- 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.

## Thermal Noise

Definitions

D. T. McGuiness, Ph.D

- 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.

## Thermal Noise

Dark Current

D. T. McGuiness, Ph.D

- 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.

## Thermal Noise

Dark Current

D. T. McGuiness, Ph.D

- 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.

## Read Noise

Introduction

D. T. McGuiness, Ph.D

- 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.

## Read Noise

### CMOS Read Noise

D. T. McGuiness, Ph.D

- 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.

## Read Noise

CMOS Read Noise

D. T. McGuiness, Ph.D

### 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.

## Read Noise

CMOS Read Noise

D. T. McGuiness, Ph.D

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,

## Read Noise

Johnson-Nyquist Noise

D. T. McGuiness, Ph.D

- 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:

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

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

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: Effect on the noise on the capacitor value.

## Read Noise

Quantisation Noise

D. T. McGuiness, Ph.D

- 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

# Application

Perlin Noise

D. T. McGuiness, Ph.D

- 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**

# Application

Perlin Noise

D. T. McGuiness, Ph.D

- 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
```

C.R. 6.14  
python

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

# Application

Perlin Noise

D. T. McGuiness, Ph.D

- 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
```

C.R. 6.15  
python

**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.

## Application

Perlin Noise

D. T. McGuiness, Ph.D

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=1024, repeaty=1024,
                                      base=seed)
```

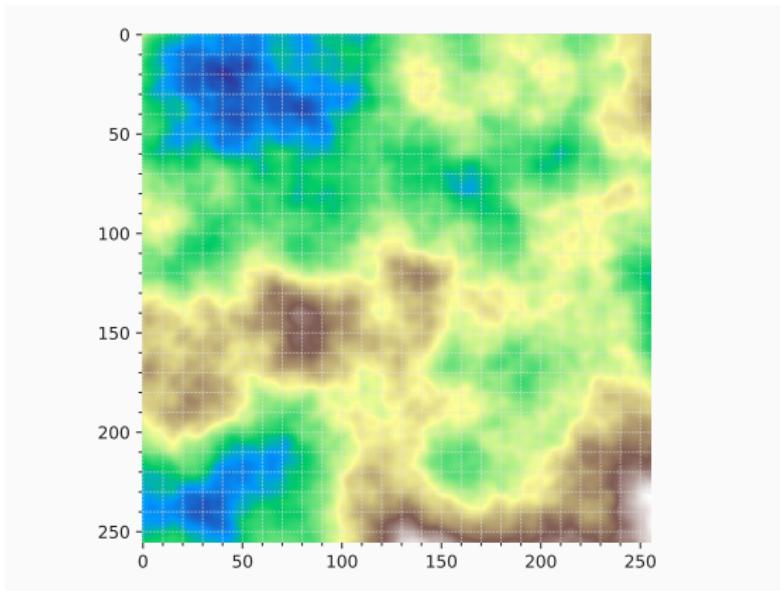
C.R. 6.16  
python

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

# Application

Perlin Noise

D. T. McGuiness, Ph.D



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

# Application

Perlin Noise

D. T. McGuiness, Ph.D

- 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
```

C.R. 6.17  
python

```
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)
```

C.R. 6.18  
python

# Application

Perlin Noise

D. T. McGuiness, Ph.D

- 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)
```

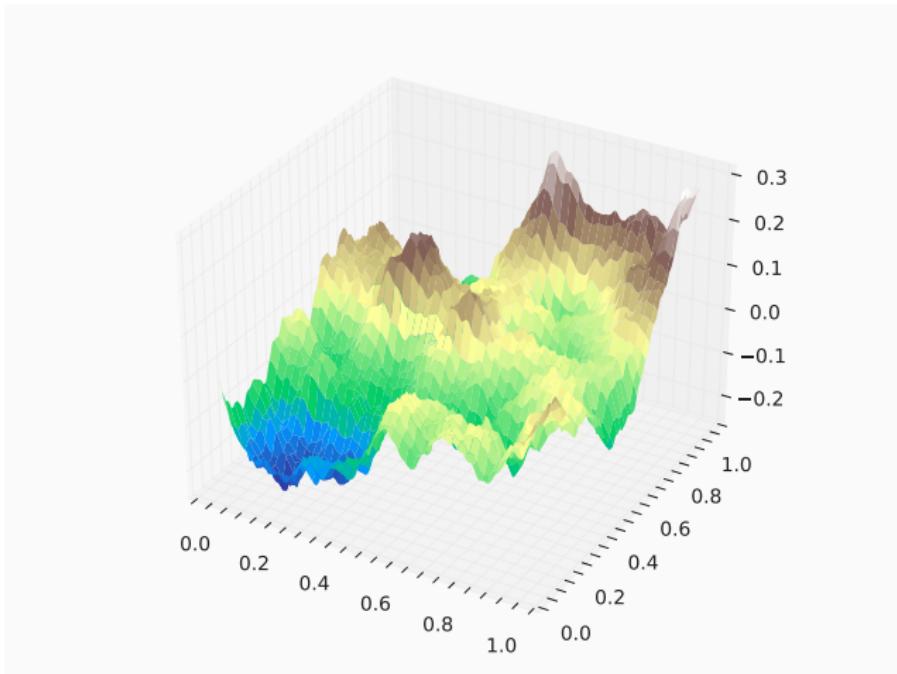
C.R. 6.19  
python

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

# Application

Perlin Noise

D. T. McGuiness, Ph.D



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



Chapter

# Histogram Operations

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 we 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:

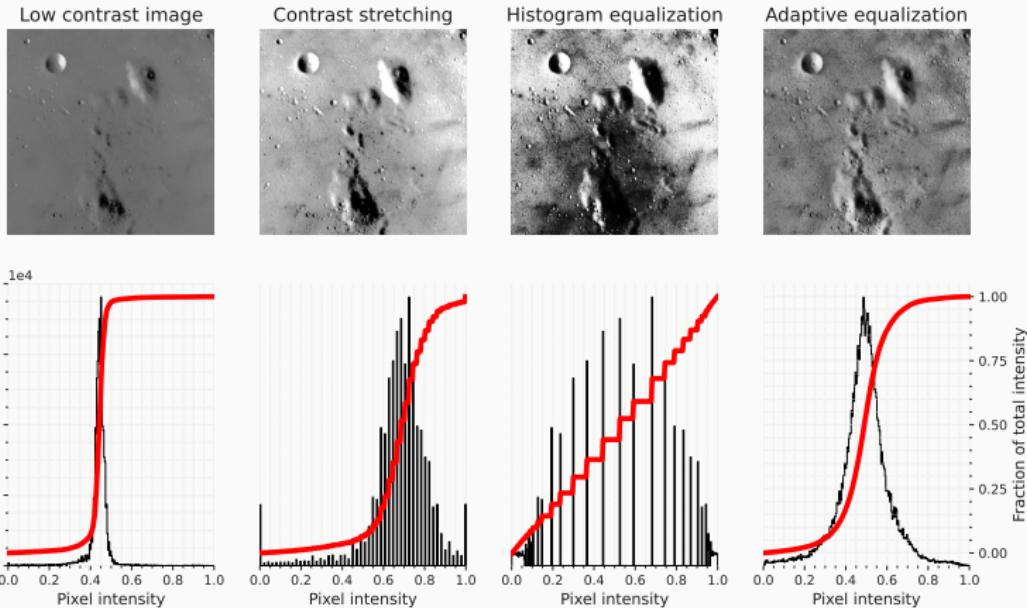
$$p_b(b) db = p_a(a) da \rightarrow \frac{p_a(a) da}{p_b(b)}.$$

- 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.



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

- 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.



## Figure

## Threshold Operations

### Applying a Threshold

D. T. McGuiness, Ph.D

- 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.

## Threshold Operations

Applying a Threshold

D. T. McGuiness, Ph.D

C.R. 7.20

python

```
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')

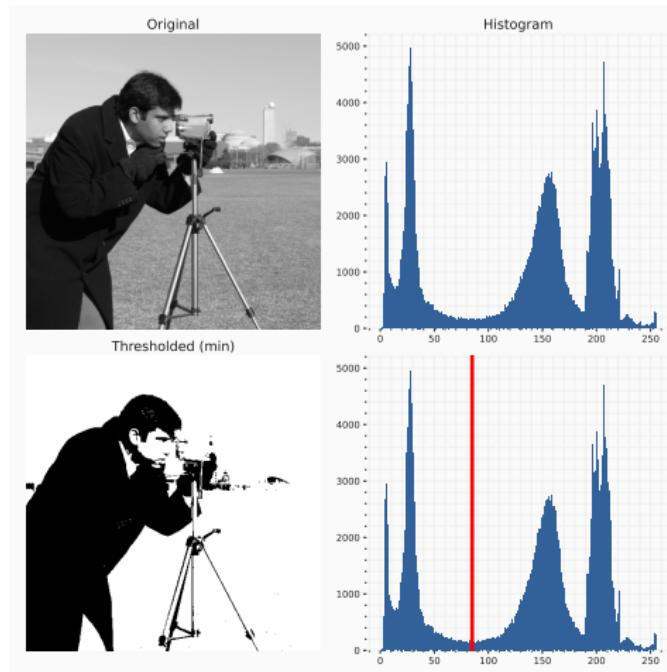
for a in ax:
    a.set_axis_off()

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

## Threshold Operations

## Bimodal Histogram

D. T. McGuiness, Ph.D



## Figure

## Threshold Operations

### Otsu Threshold

D. T. McGuiness, Ph.D

- Named after *Nobuyuki Otsu*, is used to perform automatic image thresholding [67].
- Simply, it returns **single intensity threshold** separating pixels to two (2) classes:

**Foreground** view nearest to the observer.

**Background** view farthest to the observer.

Threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance [68].

## Threshold Operations

### Local Thresholding

D. T. McGuiness, Ph.D

- 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.

## Threshold Operations

### Local Thresholding

D. T. McGuiness, Ph.D

```
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)
```

C.R. 7.21

python

# Threshold Operations

## Local Thresholding

D. T. McGuiness, Ph.D

### 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.zeros_like(coins)
```

### Global 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.zeros_like(coins)
```

### Local thresholding

#### Region-based segmentation

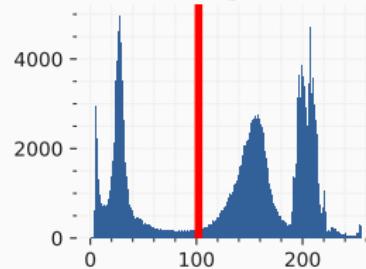
Let us first determine markers of the coins and the background. These markers are pixels that we can label unambiguously as either object or background. Here, the markers are found at the two extreme parts of the histogram of grey values:

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

## Original



## Histogram



### Thresholded



## Figure

# Threshold Operations

Local Thresholding

D. T. McGuiness, Ph.D

Original image



Result



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

## Channel Operations

RGB to Grayscale

D. T. McGuiness, Ph.D

- 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.

# Channel Operations

RGB to Grayscale

D. T. McGuiness, Ph.D

C.R. 7.22

python

```
import matplotlib.pyplot as plt

from skimage import data
from skimage.color import rgb2gray

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

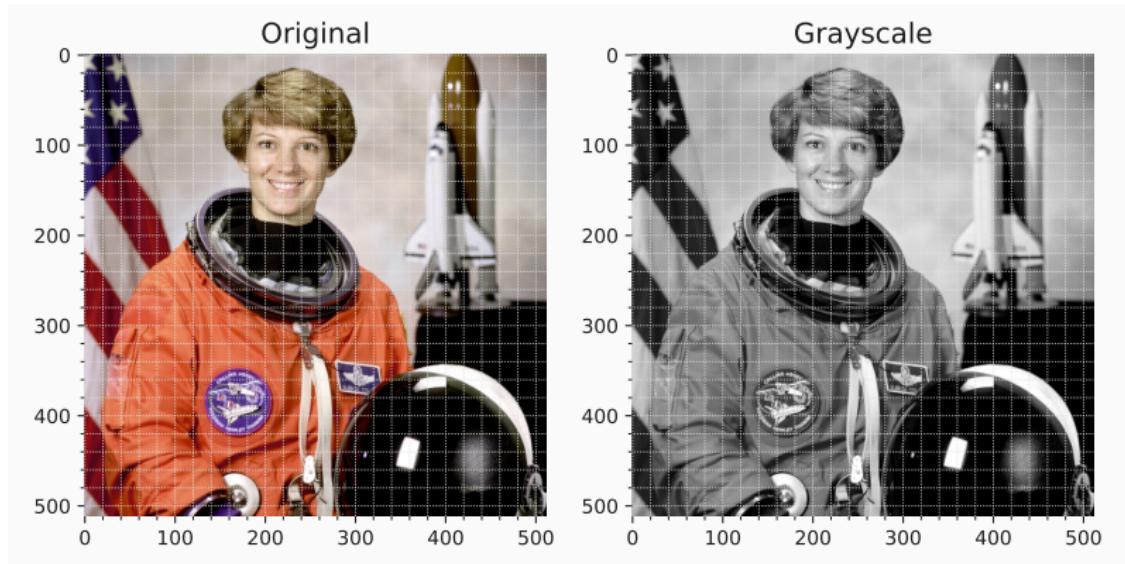
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()

ax[0].imshow(original)
ax[0].set_title("Original")
ax[1].imshow(grayscale, cmap=plt.cm.gray)
ax[1].set_title("Grayscale")
cp.store_fig("rgb-to-grayscale", close=True)
```

# Channel Operations

RGB to Grayscale

D. T. McGuiness, Ph.D.



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

## Channel Operations

### Histogram Matching

D. T. McGuiness, Ph.D

- 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).

# Channel Operations

## Histogram Matching

D. T. McGuiness, Ph.D

C.R. 7.23  
python

```
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()

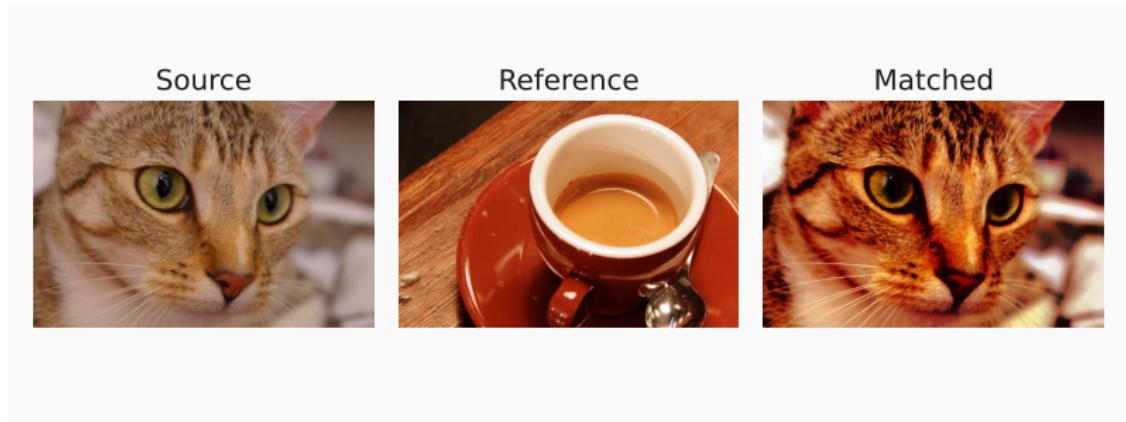
ax1.imshow(image); ax1.set_title('Source')
ax2.imshow(reference); ax2.set_title('Reference')
ax3.imshow(matched); ax3.set_title('Matched')

cp.store_fig("histogram-matching", close=True)
```

# Channel Operations

Histogram Matching

D. T. McGuiness, Ph.D



Figure

- 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).

Exercise

C.R. 7.24

python

```
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)

C.R. 7.25

text

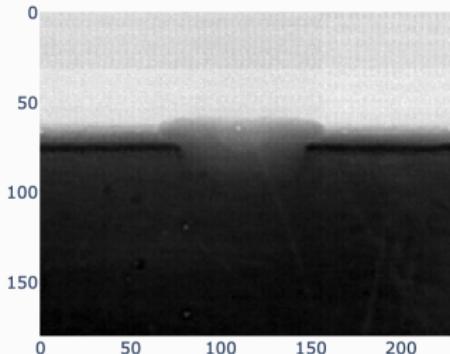
## Compute Image Deltas

- 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)
```

C.R. 7.26

python



Figure

## Compute image deltas

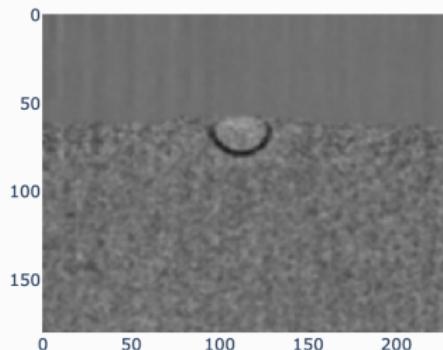
- First, apply a low-pass to smooth the images and reduce noise.
- Next, compute the image deltas.
- To do this, subtract the image sequence ending at the second-to-last frame from the image sequence starting at the second frame.

```
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'},
)
plotly.io.show(fig)
```

C.R. 7.27

python



Figure

## Clip Lowest and Highest intensities

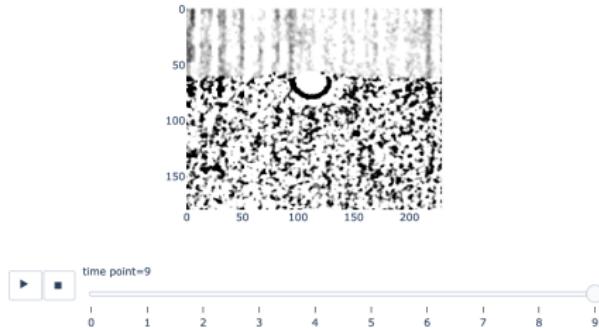
- We now calculate the 5th and 95th percentile intensities of `image_deltas`,
- We want to clip the intensities which lie below the 5th percentile intensity and above the 95th percentile intensity, while also rescaling the intensity values to  $[0, 1]$ .

C.R. 7.28

python

```
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'},
)
plotly.io.show(fig)
```



Figure

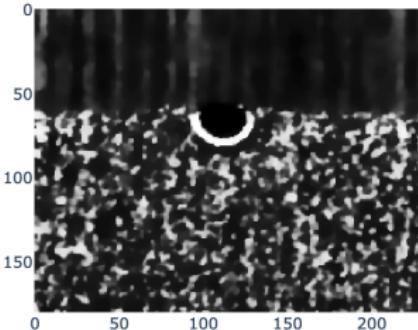
## Invert and De-noise

- 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.

```
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'},
)
plotly.io.show(fig)
```

C.R. 7.29  
python



Figure

## 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, :, :]`.

C.R. 7.30

python

```
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)
# Show top five rows
print(props_0_df.head())
```

C.R. 7.31

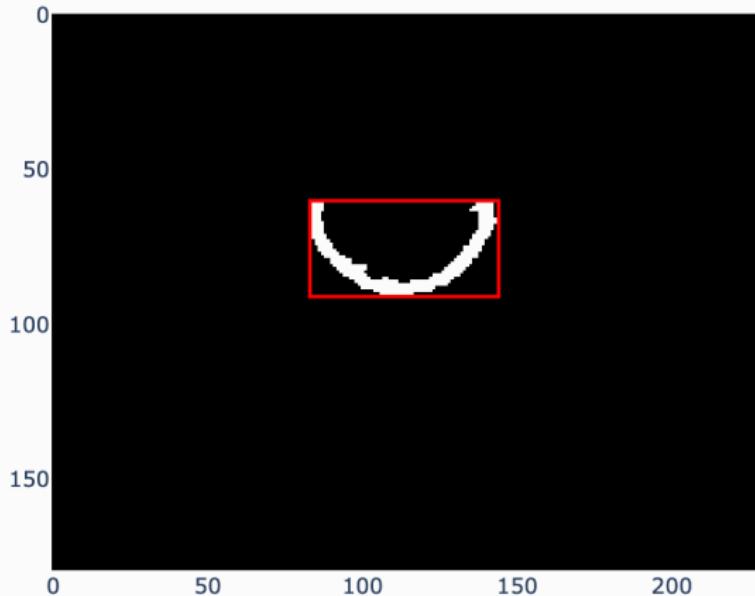
text

	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
9	10	122.0	61	208	79	229
8	9	114.0	61	183	83	198

- 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.

```
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)
fig.add_shape(type='rect', x0=minc, y0=minr, x1=maxc, y1=maxr,
↪ line=dict(color='Red'))
plotly.io.show(fig)
```

C.R. 7.32  
python



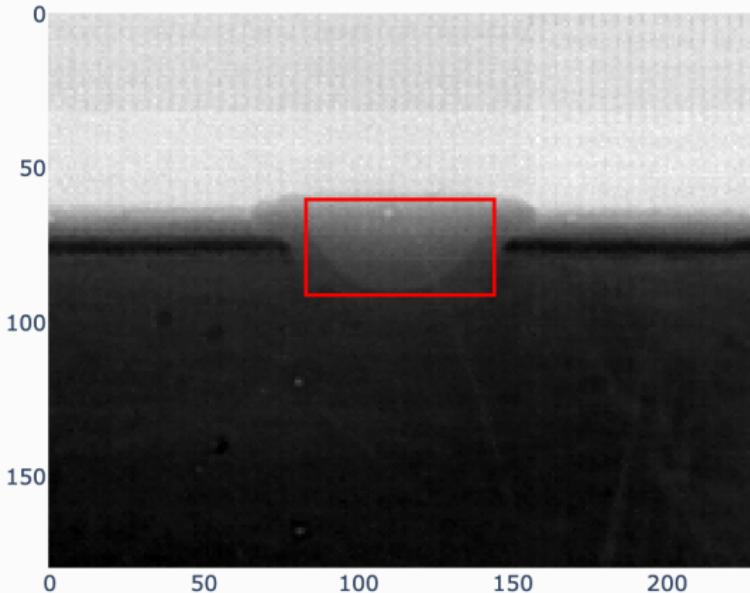
**Figure:** Showcasing the bounding box over the largest connected pixel group.

- 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.

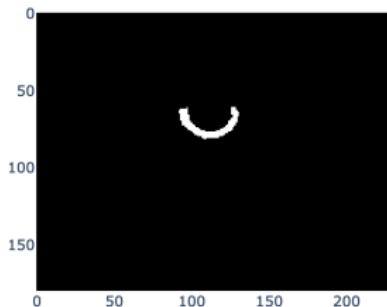
```
fig = px.imshow(image_sequence[0, :, :], binary_string=True)
fig.add_shape(type='rect', x0=minc, y0=minr, x1=maxc, y1=maxr,
              line=dict(color='Red'))
plotly.io.show(fig)
```

C.R. 7.33

python



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



## Figure

## 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.

C.R. 7.34

python

```
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'},
)
plotly.io.show(fig)
```

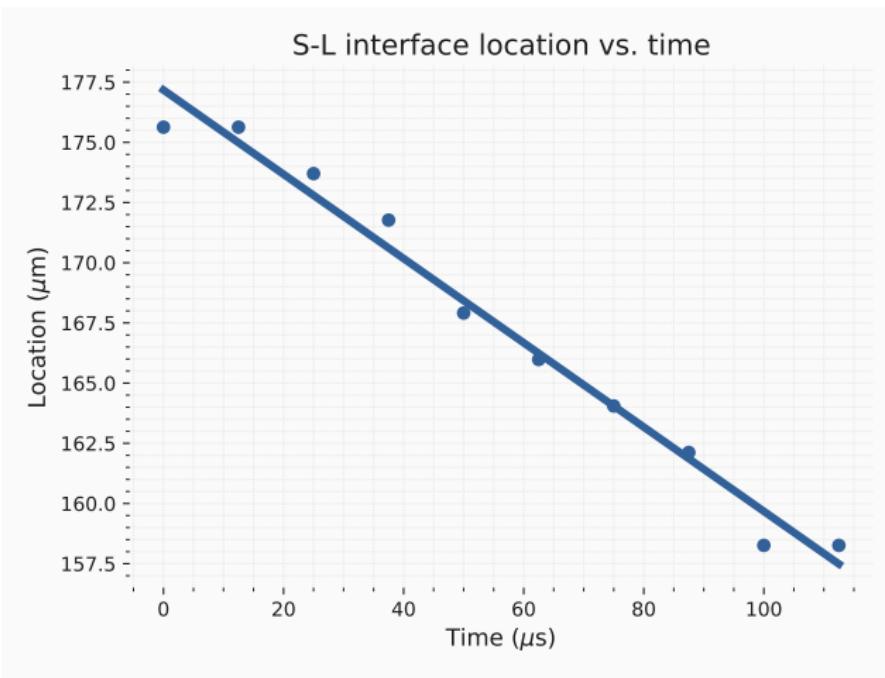
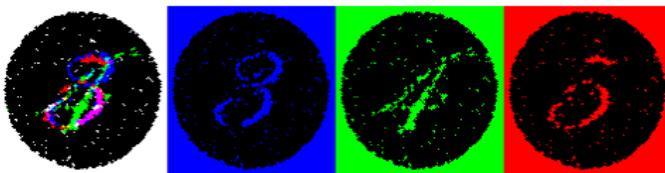


Figure: Using the image alone, it was possible to determine the metals S-L state transition.

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

C.R. 7.35

python

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

C.R. 7.36

python

```
# 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.

C.R. 7.37

python

```
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")
```



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



C.R. 7.38

python

```
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()
```

hello

C.R. 7.39  
text



Chapter

# Morphological Operations

## Introduction

Broad Definition

## Morphological Operations

Erosion

Dilation

Opening

Closing

Gradient

Top Hat

## Application

Logistic Mapping

Apply Segmented Rim as Mask

## Introduction

Broad Definition

D. T. McGuiness, Ph.D

- 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.

## Introduction

Broad Definition

D. T. McGuiness, Ph.D

- 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

## Introduction

Broad Definition

D. T. McGuiness, Ph.D

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

## Introduction

Broad Definition

D. T. McGuiness, Ph.D

- 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.

## Morphological Operations

### Erosion

D. T. McGuiness, Ph.D

- 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.

$$\begin{array}{ccccccccc} 1 & 1 & 1 & 0 & 1 & 1 & 1 & & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & + & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \xrightarrow{\quad k \quad} & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 & 0 & 0 & 0 \end{array}$$

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

# Morphological Operations

Erosion

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Erosion

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Erosion

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Erosion

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Erosion

D. T. McGuiness, Ph.D



(a)



(b)

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

## Morphological Operations

### Dilation

D. T. McGuiness, Ph.D

- 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.

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D.

$$\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & + & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & \underbrace{1 & 1 & 1}_k & \rightarrow & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underbrace{1 & 1 & 1}_k & \rightarrow & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

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

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Dilation

D. T. McGuiness, Ph.D



(a)



(b)

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

## Morphological Operations

### Opening

D. T. McGuiness, Ph.D

- 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

Opening

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Opening

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Opening

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Opening

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Opening

D. T. McGuiness, Ph.D



(a)



(b)

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

## Morphological Operations

### Closing

D. T. McGuiness, Ph.D

- 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

Closing

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Closing

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Closing

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Closing

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Closing

D. T. McGuiness, Ph.D



(a)



(b)

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

## Morphological Operations

Gradient

D. T. McGuiness, Ph.D

- 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

Gradient

D. T. McGuiness, Ph.D



(a)



(b)

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



(a)



(b)

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

# Morphological Operations

Gradient

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Gradient

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Gradient

D. T. McGuiness, Ph.D



(a)



(b)

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

## Morphological Operations

Top Hat

D. T. McGuiness, Ph.D

- 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

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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



(a)



(b)

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



(a)



(b)

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

# Morphological Operations

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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

# Morphological Operations

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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



(a)



(b)

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



(a)



(b)

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

# Morphological Operations

Top Hat

D. T. McGuiness, Ph.D



(a)



(b)

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

## Application

Logistic Mapping

D. T. McGuiness, Ph.D

- 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:

# Application

## Logistic Mapping

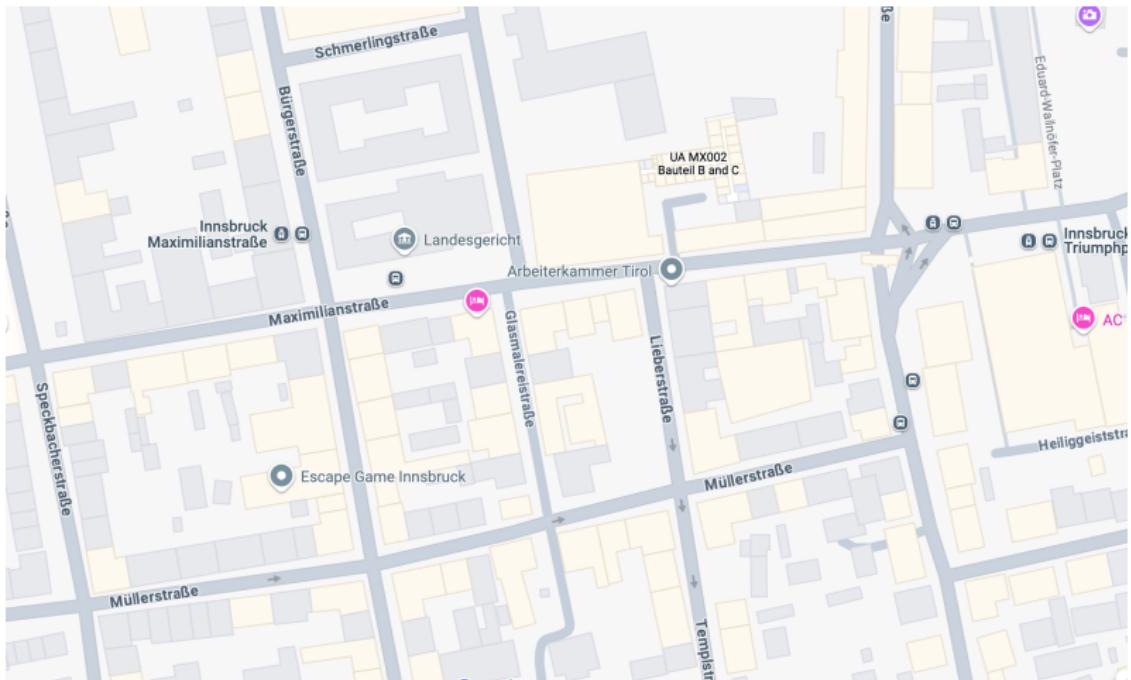


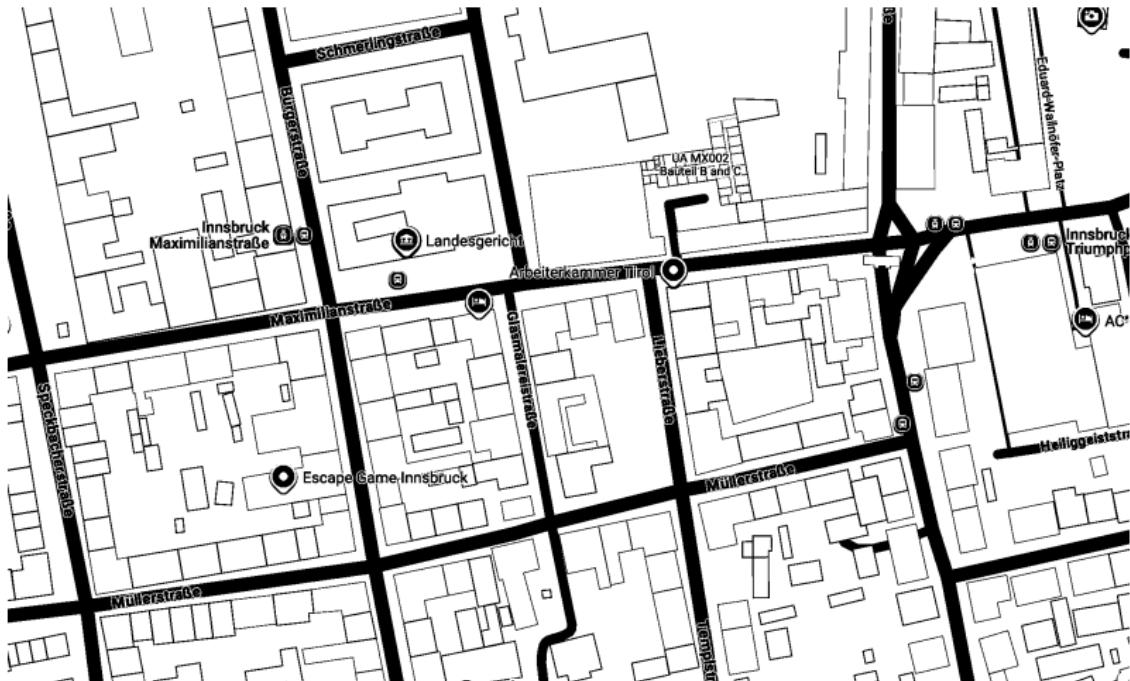
Figure: A Google Map view of central Innsbruck.

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

```
import skimage as sk
import numpy as np
from skimage.color import rgb2gray
from skimage.io import imshow
image_raw = sk.io.imread('images/data/innsbruck-map.png')
im = rgb2gray(image_raw[:, :, :3])
im_bw = im > 0.90
sk.io.imsave("images/Morphological-Operations/innsbruck-
↪ bw.png",im_bw)
```

C.R. 8.40

python



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

- 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:

```
from skimage.morphology import closing, opening

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)
```

C.R. 8.41  
python

# Application

## Logistic Mapping

D. T. McGuiness, Ph.D



Figure: Operation of opening and closing are applied.

## Application

### Logistic Mapping

D. T. McGuiness, Ph.D

- 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:

C.R. 8.42

python

```
from skimage.morphology import erosion, dilation
from skimage.draw import disk

def im_dilation(image, selem, n):
    for i in range(n):
        image = dilation(image, selem)
    return image

def im_erosion(image, selem, n):
    for i in range(n):
        image = erosion(image, selem)
    return image

map2 = im_dilation((im_erosion(map1, selem_temp, 2)),
↪ selem_temp, 4)
sk.io.imsave("images/Morphological-Operations/innsbruck-bw-
↪ 3.png",map2)
```

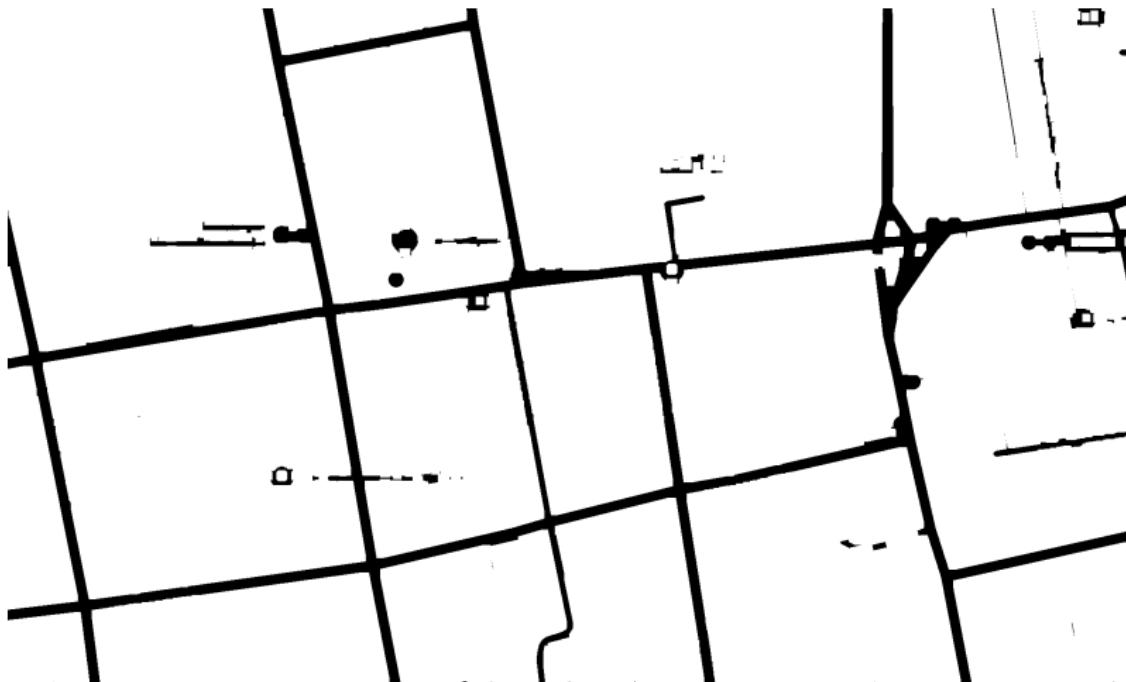


Figure: Operation of erosion and dilation are applied.

- 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.

# Application

## Logistic Mapping

D. T. McGuiness, Ph.D



Figure: 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.

- First import the necessary modules:

```
import matplotlib.pyplot as plt
import numpy as np
import plotly.io
import plotly.express as px
from scipy import ndimage as ndi

from skimage import filters, measure, morphology, segmentation
from skimage.data import protein_transport
```

C.R. 8.43  
python

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

```
image_sequence = protein_transport()

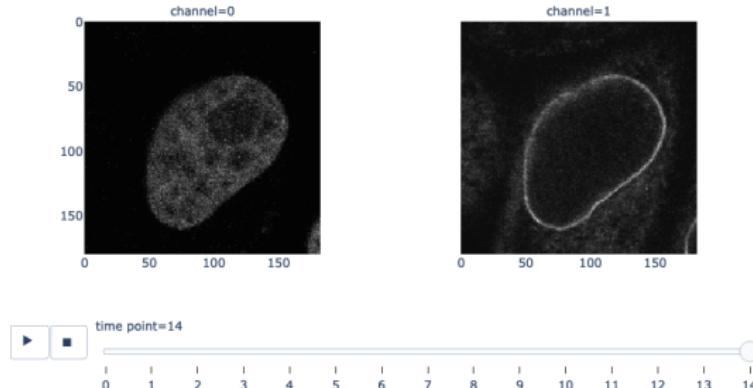
print(f'shape: {image_sequence.shape}')
```

C.R. 8.44  
python

# Application

## Logistic Mapping

D. T. McGuiness, Ph.D



Figure

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

```
image_t_0_channel_0 = image_sequence[0, 0, :, :]
```

C.R. 8.45

python

## 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.

```
smooth = filters.gaussian(image_t_0_channel_0, sigma=1.5)

thresh_value = filters.threshold_otsu(smooth)
thresh = smooth > thresh_value

fill = ndi.binary_fill_holes(thresh)
```

C.R. 8.46  
python

- 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.

```
clear = segmentation.clear_border(fill)  
clear.dtype
```

C.R. 8.47  
python

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

```
dilate = morphology.binary_dilation(clear)
erode = morphology.binary_erosion(clear)
```

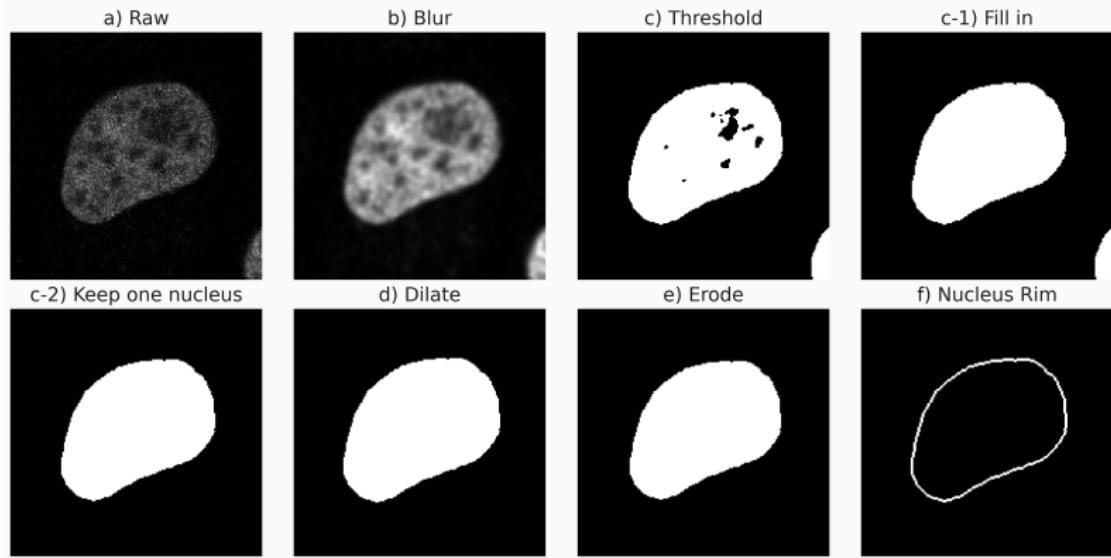
C.R. 8.48  
python

- 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`:

```
mask = np.logical_and(dilate, ~erode)
```

C.R. 8.49  
python

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



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

## Application

Apply Segmented Rim as Mask

D. T. McGuiness, Ph.D

- 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.

```
image_t_0_channel_1 = image_sequence[0, 1, :, :]
selection = np.where(mask, image_t_0_channel_1, 0)

fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(12, 6),
                             sharey=True)

ax0.imshow(image_t_0_channel_1)
ax0.set_title('Second channel (raw)')
ax0.set_axis_off()

ax1.imshow(selection)
ax1.set_title('Selection')
ax1.set_axis_off()

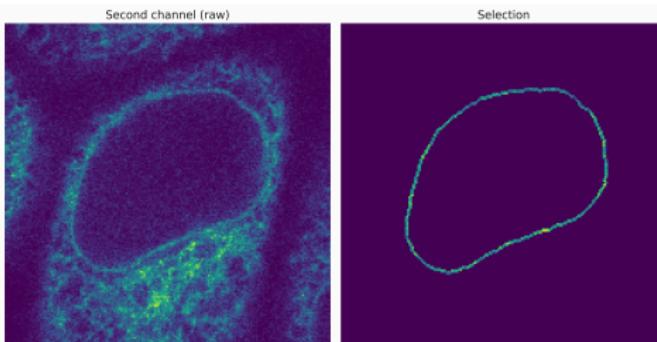
fig.tight_layout()
plt.savefig("images/Morphological-Operations/segmented-rim",
            bbox_inches='tight', dpi=300)
```

C.R. 8.50  
python

## Application

Apply Segmented Rim as Mask

D. T. McGuiness, Ph.D



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



Chapter

# Blurring Filters

## Gaussian Blur

Introduction

## Multivariate Distribution

Introduction

Mathematical Formulation

An Experiment

Code Implementation

## Box Blur

Introduction

Syntax

## Blur Filter

Introduction

## Bilateral Filter

Introduction

Butterworth Filters

## Gaussian Blur

Introduction

D. T. McGuiness, Ph.D

- Applying a **Gaussian blur** to an image is the same as **convolving the image with a Gaussian function**.
- For one dimension, the Gaussian kernel is:

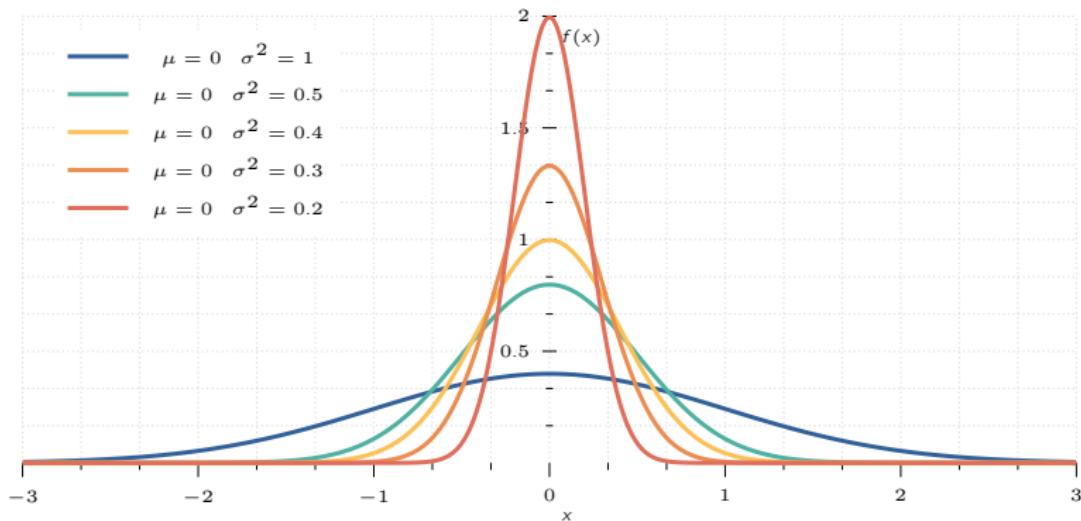
$$G(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right),$$

and an image (i.e. 2D matrix), the kernel is:

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

where  $x$  is the distance from the origin in the horizontal axis,  $y$  is the distance from the origin in the vertical axis, and  $\sigma$  is the standard deviation of the Gaussian distribution.

This is also known as a two-dimensional Weierstrass transform.



**Figure:** A Gaussian (i.e., normal) distribution with different mean and variance values.

## Gaussian Blur

Introduction

D. T. McGuiness, Ph.D

- As the Fourier of a Gaussian is another Gaussian ([Proof](#)), applying a Gaussian blur has the effect of **reducing high-frequency components**.
  - Gaussian blur is a **low-pass filter**.
- A widely used effect to **reduce image noise** and **reduce detail**.
- The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen.
  - This is different from the **bokeh effect** produced by an out-of-focus lens or the shadow of an object under usual illumination.

## Gaussian Blur

Introduction

D. T. McGuiness, Ph.D

- In photography, bokeh is the **aesthetic quality of the blur** produced in out-of-focus parts of an image, caused by circles of confusion.
- Bokeh has also been defined as "*the way the lens renders out-of-focus points of light*".
- Differences in lens aberrations and aperture shape cause very different bokeh effects.
- Some lens designs blur the image in a way that is pleasing to the eye, while others produce distracting or unpleasant blurring ("good" and "bad" bokeh, respectively).
- Photographers may deliberately use a shallow focus technique to create images with prominent out-of-focus regions, accentuating their lens's bokeh.



**Figure:** An example of the Bokeh effect.

## Multivariate Distribution

Introduction

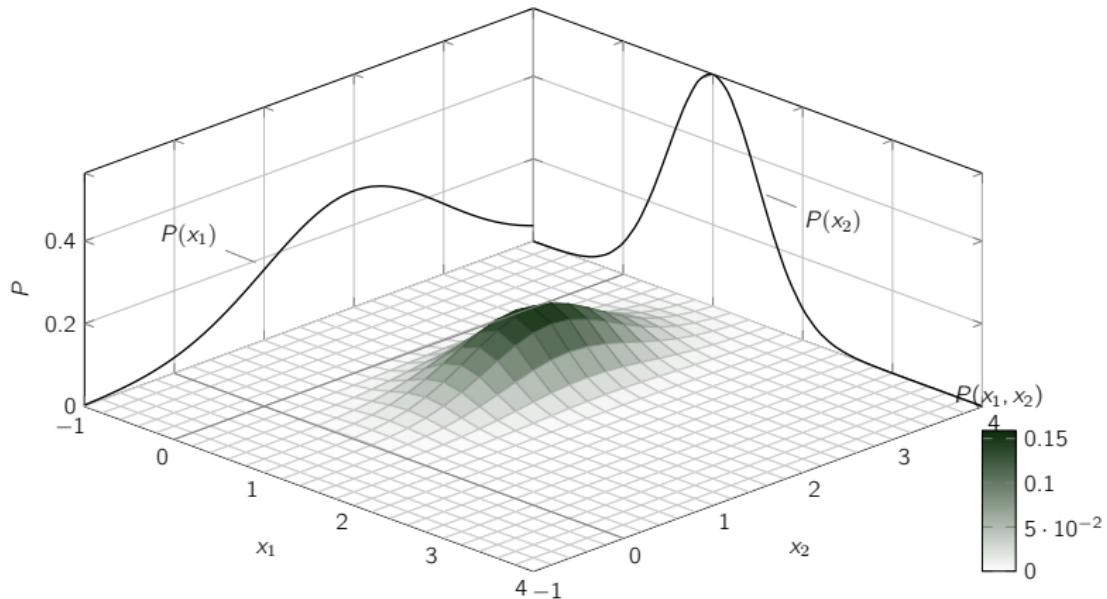
D. T. McGuiness, Ph.D.

- Interesting problems in statistics involve looking at more than a single measurement at a time, at relationships among measurements and comparisons between them.
- To address such problems, indeed to even formulate them properly, we will need to enlarge our mathematical structure to include multivariate distributions, the probability distributions of pairs of random variables, triplets of random variables, and so forth.
- We will begin with the simplest such situation, that of pairs of random variables or **bivariate** distributions, where we will already encounter most of the key ideas.

# Multivariate Distribution

Introduction

D. T. McGuiness, Ph.D



**Figure:** A 2D Gaussian (i.e., Multivariate Normal) distribution, a generalisation of the one-dimensional (**univariate**) normal distribution to higher dimensions. In the image the variables are  $\mu = [1, 2]^T$  and  $\Sigma = [0.5, 0; 0, 1.2]$ .

## Multivariate Distribution

Mathematical Formulation

D. T. McGuiness, Ph.D

- Let  $X, Y$  be two random variables defined on the same Sample space  $S$ , so it is both meaningful and potentially interesting to consider how they may interact or affect one another, we will define their bivariate probability function by

$$p(x, y) = P(X = x \text{ and } Y = y).$$

- Compared to the univariate case,  $p(x, y)$  can be thought of describing the distribution of a unit mass in the  $(x, y)$  plane, with  $p(x, y)$  representing the mass assigned to the point  $(x, y)$ , considered as a spike at  $(x, y)$  of height  $p(x, y)$ .
- The total for all possible points must be one:

$$\sum_{\text{all } x \text{ and } y} p(x, y) = 1 \quad \blacksquare$$

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

Consider the experiment of tossing a fair coin three (3) times, and then, **independently** of the first coin, tossing a second fair coin three (3) times.

Let

$X = \#$ Heads for the first coin

$Y = \#$ Tails for the second coin

$Z = \#$ Tails for the first coin

Example

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

The two (2) coins are tossed independently.

Therefore for any pair of possible values  $(x, y)$  of  $X$  and  $Y$  we have, if  $X = x$  stands for the event  $X = x$ ,

$$\begin{aligned} p(x, y) &= P(X = x \text{ and } Y = y), \\ &= P(X = x) \cap P(Y = y), \\ &= P(X = x) \cdot P(Y = y), \\ &= P_X(x) \cdot P_Y(y). \end{aligned}$$

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

On the other hand,  $X$  and  $Z$  refer to the **same coin**, and so:

$$\begin{aligned} p(x, z) &= P(X = x \text{ and } Z = z), \\ &= P(X = x) \cap P(Z = z), \\ &= P(X = x) = p_X(x) \quad \text{if } z = 3 - x, \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

This is because we must necessarily have  $x + z = 3$ , which means  $X = x$  and  $Z = x - 3$  describe the same event.

If  $z \neq 3 - x$ , then  $X = x$  and  $Z = z$  are **mutually exclusive** and the probability both occur is zero.

Solution

# Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

These bivariate distributions can be summarized in the form of tables, whose entries are  $p(x, y)$  and  $p(x, z)$  respectively:

$x \setminus y$	0	1	2	3
0	$\frac{1}{64}$	$\frac{3}{64}$	$\frac{3}{64}$	$\frac{1}{64}$
1	$\frac{3}{64}$	$\frac{9}{64}$	$\frac{9}{64}$	$\frac{3}{64}$
2	$\frac{3}{64}$	$\frac{9}{64}$	$\frac{9}{64}$	$\frac{3}{64}$
3	$\frac{1}{64}$	$\frac{3}{64}$	$\frac{3}{64}$	$\frac{1}{64}$

$x \setminus z$	0	1	2	3
0	0	0	0	$\frac{1}{8}$
1	0	0	$\frac{3}{8}$	0
2	0	$\frac{3}{8}$	0	0
3	$\frac{1}{8}$	0	0	0

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

Now, if we have specified a bivariate probability function such as  $p(x, y)$ , we can always deduce the respective univariate distributions from it, by addition:

$$p_X(x) = \sum_{\text{all } y} p(x, y),$$

$$p_Y(y) = \sum_{\text{all } x} p(x, y).$$

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

The idea is that we can decompose the event  $X = x$  into a collection of smaller sets of outcomes. For example,

$$X = x = X = x \quad \text{and} \quad Y = 0 \cup X = x \quad \text{and} \quad Y = 0 \cup \dots$$

$$\dots \cup X = x \quad \text{and} \quad Y = 23 \cup \dots$$

where the values of  $y$  on the righthand side run through all possible values of  $Y$ .

But then the events of the RHS are mutually exclusive ( $Y$  cannot have two values at once), so the probability of the RHS is the sum of the events' probabilities, or  $\sum_{\text{all } y} p(x, y)$ , while the LHS has probability  $p_X(x)$ .

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

When we refer to these univariate distributions in a multivariate context, we shall call them the marginal probability functions of X and Y.

This name comes from the fact that when the addition in  $p_X(x)$  or  $p_Y(y)$  is performed upon a bivariate distribution  $p(x, y)$  written in tabular form,

Results are most naturally written in the margins of the table.

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

For the coin experiment, we have the marginal distributions of  $X, Y, Z$ :

$x \setminus y$	0	1	2	3	$p_X(x)$
0	$\frac{1}{64}$	$\frac{3}{64}$	$\frac{3}{64}$	$\frac{1}{64}$	$\frac{1}{8}$
1	$\frac{3}{64}$	$\frac{9}{64}$	$\frac{9}{64}$	$\frac{3}{64}$	$\frac{1}{8}$
2	$\frac{3}{64}$	$\frac{9}{64}$	$\frac{9}{64}$	$\frac{3}{64}$	$\frac{3}{8}$
3	$\frac{1}{64}$	$\frac{3}{64}$	$\frac{3}{64}$	$\frac{1}{64}$	$\frac{1}{8}$
$p_Y(y)$	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	

You can find the marginal distributions from the bivariate distribution, but it is not possible the other way around.

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

For the coin experiment, we have the marginal distributions of  $X, Y, Z$ :

$x \setminus z$	0	1	2	3	$p_X(x)$
0	0	0	0	$\frac{1}{8}$	$\frac{1}{8}$
1	0	0	$\frac{3}{8}$	0	$\frac{3}{8}$
2	0	$\frac{3}{8}$	0	0	$\frac{3}{8}$
3	$\frac{1}{8}$	0	0	0	$\frac{1}{8}$
$p_Z(z)$	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	

You can find the marginal distributions from the bivariate distribution, but it is not possible the other way around.

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

Here, both tables have exactly the same marginal totals, in fact  $X$ ,  $Y$ ,  $Z$  all have the same Binomial (3, 12) distribution, the bivariate distributions are quite different

The marginal distributions  $pX(x)$  and  $pY(y)$  may describe uncertainty about the possible values, respectively, of  $X$  considered separately, without regard to whether or not  $Y$  is even observed, and of  $Y$  considered separately, without regard to whether or not  $X$  is even observed.

But they cannot tell us about the relationship between  $X$  and  $Y$ , they alone cannot tell us whether  $X$  and  $Y$  refer to the same coin or to different coins.

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

However, the coin toss also gives a hint on what sort of information is needed to build up a bivariate distribution from component parts.

In one case, the independence of two coins gave us  $p(x, y) = P_X(x) \cdot P_Y(y)$ ; in another the complete dependence of  $Z$  on  $X$  gave us  $p(x, z) = p_X(x)$  or 0 as  $z = 3 - x$  or not.

What was needed was information about how the knowledge of one random variable's outcome may affect the other: **conditional information**. We formalise this as a conditional probability function, defined by:

$$p(y|x) = P(Y = y|X = x)$$

which we read as the probability that  $Y = y$  given that  $X = x$ .

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

As " $Y = y$ " and " $X = x$ " are events, this is just our earlier notion of conditional probability re-expressed for discrete random variables, and from:

$$\begin{aligned} p(y|x) &= P(Y = y|X = x) \\ &= \frac{P(X = x \text{ and } Y = y)}{P(X = x)} \\ &= \frac{p(x, y)}{p_X(x)}, \end{aligned}$$

as long as  $p_X(x) > 0$  with  $p(y|x)$  undefined for any  $x$  with  $p_X(x) = 0$ .

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

If  $p(y|x) = p_Y(y)$  for all possible pairs of values  $(x, y)$  for which  $p(y|x)$  is defined, we say  $X$  and  $Y$  are independent variables. From (3.6), we would equivalently have that  $X$  and  $Y$  are independent random variables if

$$p(x, y) = p_X(x) \cdot p_Y(y), \quad \text{for all } x, y.$$

Thus  $X$  and  $Y$  are independent only if all pairs of events " $X = x$ " and " $Y = y$ " are independent; if (3.7) should fail to hold for even a single pair  $(x_0, y_0)$ ,  $X$  and  $Y$  would be dependent.

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

In the example,  $X$  and  $Y$  are independent, but  $X$  and  $Z$  are **dependent**. For example, for  $x = 2$ ,  $p(z|x)$  is given by:

$$p(z|2) = \frac{p(2, z)}{p_X(2)} = \begin{cases} 1 & \text{if } z = 1, \\ 0 & \text{otherwise.} \end{cases}$$

therefore  $p(z|x) \neq p_Z(z)$  for  $x = 2, z = 1$ .

Solution

## Multivariate Distribution

An Experiment

D. T. McGuiness, Ph.D

By using the form:

$$p(x, y) = p_X(x) p(y|x) \quad \text{for all } x, y \quad \blacksquare$$

It is possible to construct a bivariate distribution from two components: either marginal distribution and the conditional distribution of the other variable given the one whose marginal distribution is specified. Thus while marginal distributions are themselves insufficient to build a bivariate distribution, the conditional probability function captures exactly what additional information is needed

Solution

## Multivariate Distribution

Code Implementation

D. T. McGuiness, Ph.D

Blurs an image using a Gaussian filter.

**src** input image which can have any number of channels,

**dst** output image of the **same size and type** as **src**.

**ksize** kernel size.**width** and **height** can differ but both must be **positive and odd**. If zero, it's computed from **sigma**.

**SigmaX** kernel standard deviation in X direction.

**SigmaY** kernel standard deviation in Y direction.

if zero, it is equal to **sigmaX**.

**borderType** Pixel extrapolation method.

**hint** Implementation modification flags.

- Load up the necessary modules for the code to work.

```
import cv2 as cv # used in digital image analysis  
import matplotlib.pyplot as plt  
import matplotlib.colors # all related to colours  
from matplotlib.image import imread
```

C.R. 9.51  
python

Some modules worth mentioning:

**matplotlib.colors** A module for converting numbers or color arguments to RGB or **RGBA**.

**cv2** A module to access openCV libraries.

**matplotlib.image** The image module supports basic image loading, rescaling and display operations.

- The following is a python code implementation of Gaussian blurring.

```
from skimage import data
import matplotlib.cm as cm
from skimage import filters

def gaussian_blur(sigma):
    image = data.camera()

    path = "images/Blurring-Filters/Gaussian-blur-" + str(sigma)
    → + ".png"

    fig, (ax1, ax2) = plt.subplots(1, 2)

    ax1.grid(False); ax1.minorticks_off(); ax1.set_xticks([]);
    → ax1.set_yticks([])
    ax1.imshow(image, cmap=cm.gray)

    ax2.grid(False); ax2.minorticks_off(); ax2.set_xticks([]);
    → ax2.set_yticks([])
    ax2.imshow(filters.gaussian(image, sigma = sigma),
    → cmap=cm.gray)
```

C.R. 9.52  
python

```
gaussian_blur(5)
```



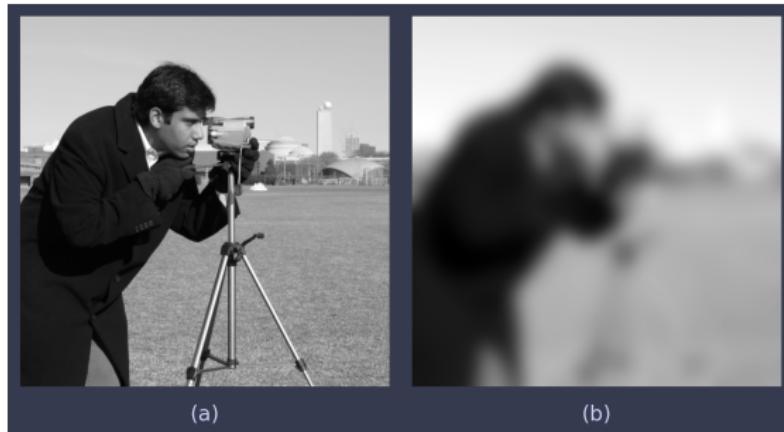
**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 5`.

```
gaussian_blur(9)
```



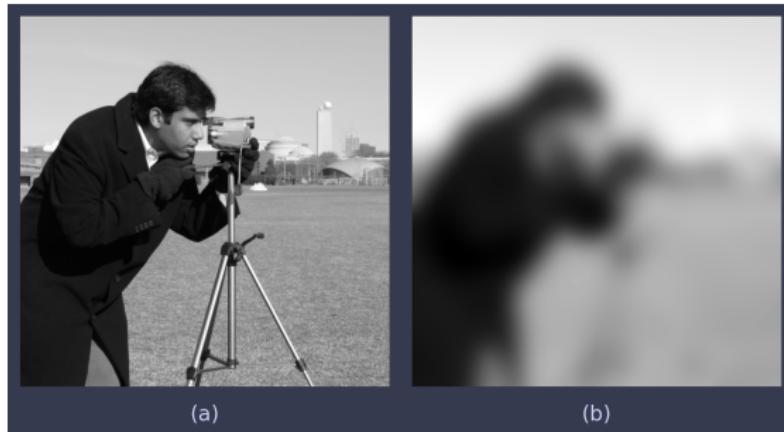
**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 9`.

```
gaussian_blur(13)
```



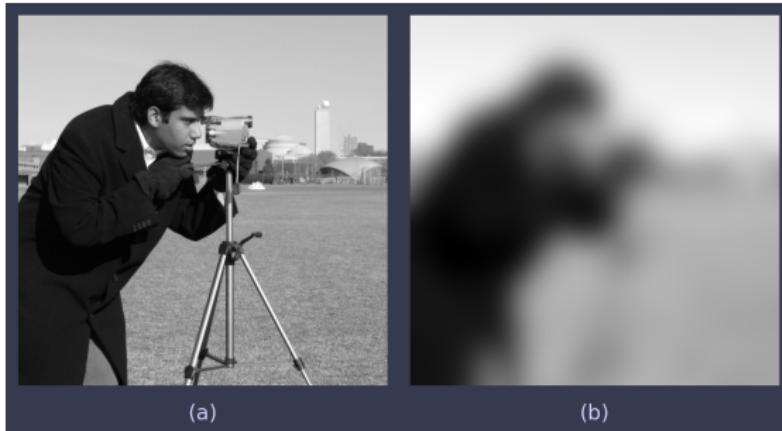
**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 13`.

```
gaussian_blur(17)
```



**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 17`.

```
gaussian_blur(21)
```



**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 21`.

## Multivariate Distribution

Code Implementation

D. T. McGuiness, Ph.D

The following codes does a single image comparison of all previous results of the `blur_gaussian` function.

C.R. 9.59  
python

```
imaged = cv.imread (image)
o, gb_5 = blur_gaussian(image, k_width=5, k_height=5, rgb=True)
o, gb_9 = blur_gaussian(image, k_width=9, k_height=9, rgb=True)
o, gb_13 = blur_gaussian(image, k_width=13, k_height=13,
↪   rgb=True)
o, gb_17 = blur_gaussian(image, k_width=17, k_height=17,
↪   rgb=True)
o, gb_21 = blur_gaussian(image, k_width=21, k_height=21,
↪   rgb=True)

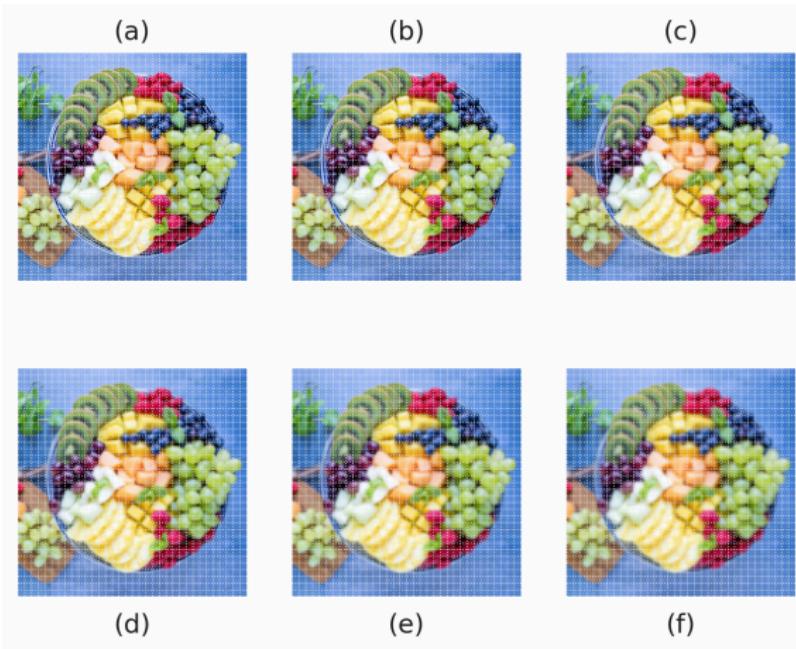
image_array=[imaged, gb_5, gb_9, gb_13, gb_17, gb_21]

Plotting.image_subplot_style(2, 3,
                           image_array=image_array,
                           ↪   show=False,
                           publish="images/Blurring-Filters/"]
                           ↪   Gaussian-blur-comparison",
                           rgb=True)
```

## Multivariate Distribution

## Code Implementation

D. T. McGuiness, Ph.D



**Figure:** A visual comparison of Gaussian blur with different kernel sizes (a) Original, (b) 5, (c) 9, (d) 13, (e) 17, (f) 21.

## Box Blur

Introduction

D. T. McGuiness, Ph.D

- Also known as a box linear filter, is a spatial domain linear filter in which each pixel in the resulting image has a value equal to the average value of its neighbouring pixels in the input image.
- It is a form of low-pass (**blurring**) filter as the high frequency components are smoothed out by the averaging.
- i.e., a 3-by-3 box blur (**radius 1**) can be written as matrix:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Used frequently used to approximate a Gaussian blur. By the **central limit theorem**, repeated application of a box blur will approximate a Gaussian blur.

## Box Blur

Introduction

D. T. McGuiness, Ph.D

- In the frequency domain, a box blur has zeros and negative components.
- This means a sine wave with a period equal to the size of the box will be blurred away entirely, and wavelengths shorter than the size of the box may be phase-reversed, as seen when two bokeh circles touch to form a bright spot where there would be a dark spot between two bright spots in the original image.

## Box Blur

Syntax

D. T. McGuiness, Ph.D

```
box_filter_blur(21)
#+end_src

#+RESULTS: BLUR-BOX-PLOT-21
```

C.R. 9.60  
python

**src** input image,

**dst** output image of the same size and type as **src**.

**ddepth** the output image depth (-1 to use `src.depth()`).

**ksize** blurring kernel size,

**anchor** anchor point; default value `Point(-1,-1)` means that the anchor is at the kernel centre.

**normalize** flag, whether the kernel is normalised by its area or not.

**borderType** border mode used to extrapolate outside pixels.

## Box Blur

Syntax

D. T. McGuiness, Ph.D

The following is a python implementation of the **box Filter**.

```
from scipy import ndimage

def box_filter_blur(size):
    image = data.camera()

    path = "images/Blurring-Filters/Boxed-blur-" + str(size) +
    ".png"

    fig, (ax1, ax2) = plt.subplots(1, 2)

    ax1.grid(False); ax1.minorticks_off(); ax1.set_xticks([]);
    → ax1.set_yticks([])
    ax1.imshow(image, cmap=cm.gray)

    ax2.grid(False); ax2.minorticks_off(); ax2.set_xticks([]);
    → ax2.set_yticks([])
    ax2.imshow(ndimage.uniform_filter(image, size=size),
    → cmap=cm.gray)

    ax1.set_xlabel('(a)', labelpad=10)
    ax2.set_xlabel('(b)', labelpad=10)
```

C.R. 9.61

python

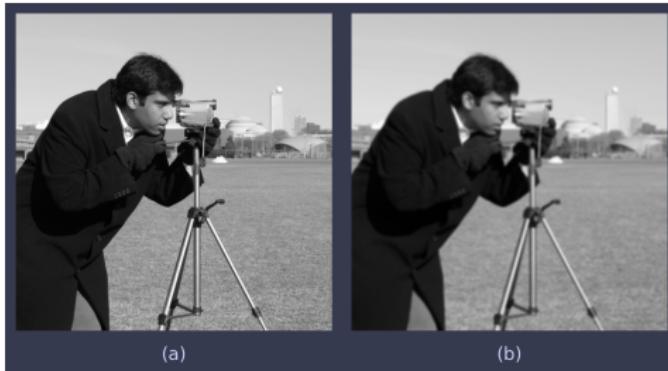
## Box Blur

Syntax

D. T. McGuiness, Ph.D

```
box_filter_blur(5)
```

C.R. 9.63  
python



**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size `k_width = k_height = 5`.

## Box Blur

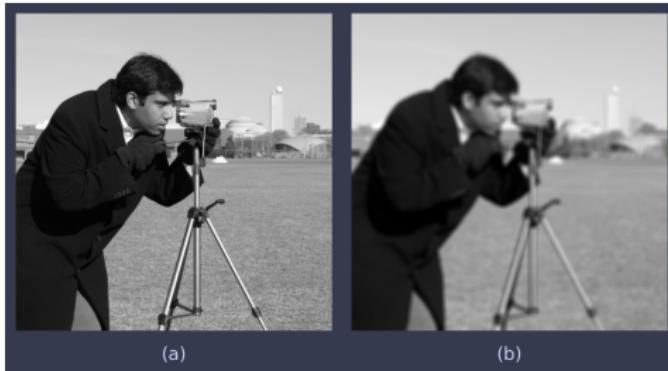
Syntax

D. T. McGuiness, Ph.D

```
box_filter_blur(9)
```

C.R. 9.64

python



**Figure:** Comparison of the original image with a processed version with a Gaussian kernel of size  $k\_width = k\_height = 9$ .

```
def blur_normal_box(image, k_height=5, k_width=5,
                     show=None, rgb=None, publish=None):

    # apply the filter
    blur_boxed = cv.blur(cv.imread(image),(k_height,
    ↴ k_width))

    # Invoke the sub-plotting function
    Plotting.image_subplot_style(
        1, 2,
        image_array=[cv.imread(image),
                     blur_boxed],
        show=show, publish=publish, rgb=rgb)

    return cv.imread(image), blur_boxed
```

# Blur Filter

Introduction

D. T. McGuiness, Ph.D

C.R. 9.66  
python

```
blur_normal_box(image, show=True, rgb=True,  
                 publish="images/Blurring-Filters/Blur-normal-box-  
→ 5")
```



**Figure:** Comparison of the original image with a processed version with a box kernel of size `k_width = k_height= 5`.

```
blur_normal_box(image,  
                 k_height=9, k_width=9, show=True, rgb=True,  
                 publish="images/Blurring-Filters/Blur-normal-  
                     ↵ box-9")
```



**Figure:** Comparison of the original image with a processed version with a box kernel of size  $k\_width = k\_height = 9$ .

# Blur Filter

Introduction

D. T. McGuiness, Ph.D



**Figure:** Comparison of the original image with a processed version with a box kernel of size `k_width = k_height = 13`.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D



**Figure:** Comparison of the original image with a processed version with a box kernel of size `k_width = k_height = 17`.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D



**Figure:** Comparison of the original image with a processed version with a box kernel of size `k_width = k_height = 21`.

## Blur Filter

Introduction

D. T. McGuiness, Ph.D

The following codes does a single image comparison of all previous results of the `blur_normal_box` function.

C.R. 9.68

python

```
o, gb_5 = blur_normal_box(image, k_width=5, k_height=5,
↪   rgb=True)
o, gb_9 = blur_normal_box(image, k_width=9, k_height=9,
↪   rgb=True)
o, gb_13 = blur_normal_box(image, k_width=13, k_height=13,
↪   rgb=True)
o, gb_17 = blur_normal_box(image, k_width=17, k_height=17,
↪   rgb=True)
o, gb_21 = blur_normal_box(image, k_width=21, k_height=21,
↪   rgb=True)

image_array=[imaged, gb_5, gb_9, gb_13, gb_17, gb_21]

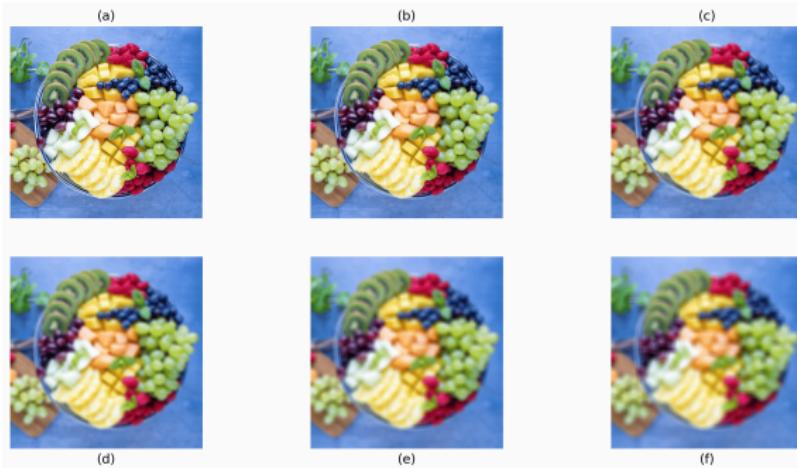
Plotting.image_subplot_style(2, 3,
    image_array=image_array, show=False,
    publish="images/Blurring-Filters/Blur-normal-box-
↪   comparison",
    rgb=True)

#+end_src
```

# Blur Filter

Introduction

D. T. McGuiness, Ph.D



**Figure:** A visual comparison of Box filter blur with different kernel sizes (a) Original, (b) 5, (c) 9, (d) 13, (e) 17, (f) 21.

C.R. 9.69

python

```
from skimage.filters.rank import median
from skimage.morphology import disk

def median_blur(size):
    image = data.camera()

    path = "images/Blurring-Filters/Median-blur-" + str(size) +
    ".png"

    fig, (ax1, ax2) = plt.subplots(1, 2)

    ax1.grid(False); ax1.minorticks_off(); ax1.set_xticks([]);
    ax1.set_yticks([])
    ax1.imshow(image, cmap=cm.gray)

    ax2.grid(False); ax2.minorticks_off(); ax2.set_xticks([]);
    ax2.set_yticks([])
    ax2.imshow(median(image, disk(size)), cmap=cm.gray)

    ax1.set_xlabel('(a)', labelpad=10)
    ax2.set_xlabel('(b)', labelpad=10)
```

## Blur Filter

Introduction

D. T. McGuiness, Ph.D

Blurs an image using the median filter.

The function smoothes an image using the median filter with the **aperture**. Each channel of a multi-channel image is processed independently. In-place operation is supported.

**src** input can be 1, 3, or 4-channels; when **ksize** is 3 or 5, the image depth should be **CV\_8U**, **CV\_16U**, or **CV\_32F**, for larger aperture sizes, it can only be **CV\_8U**. For more info [openCV Data Types](#)

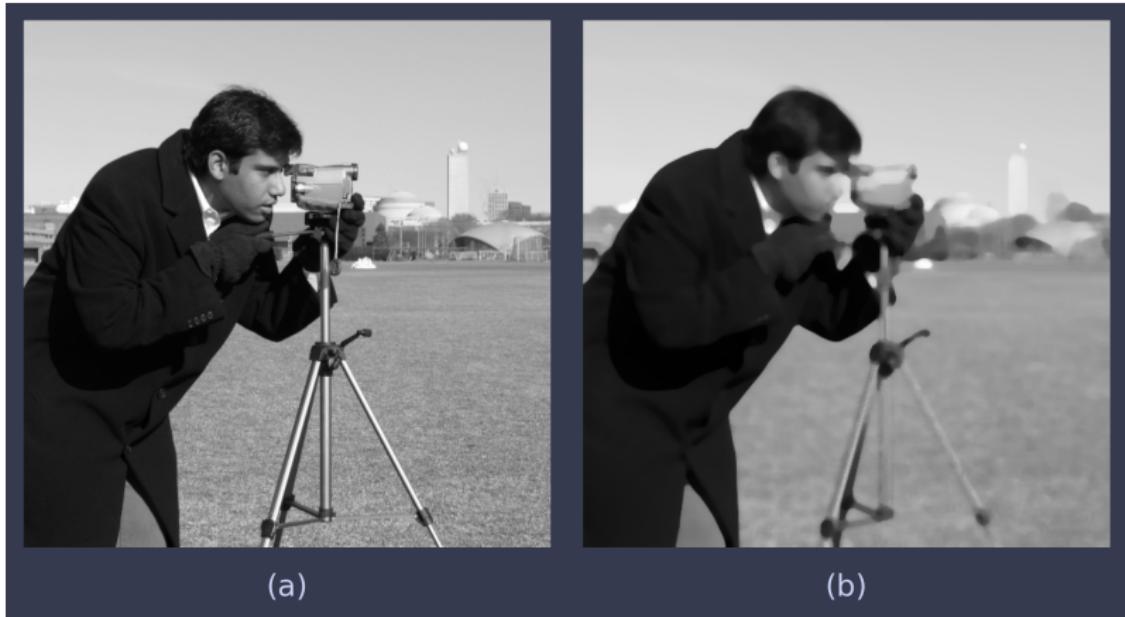
**dst** destination of the **same size and type** as **src**.

**ksize** Aperture linear size. **Must be odd and greater than 1**, i.e., 3, 5.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D

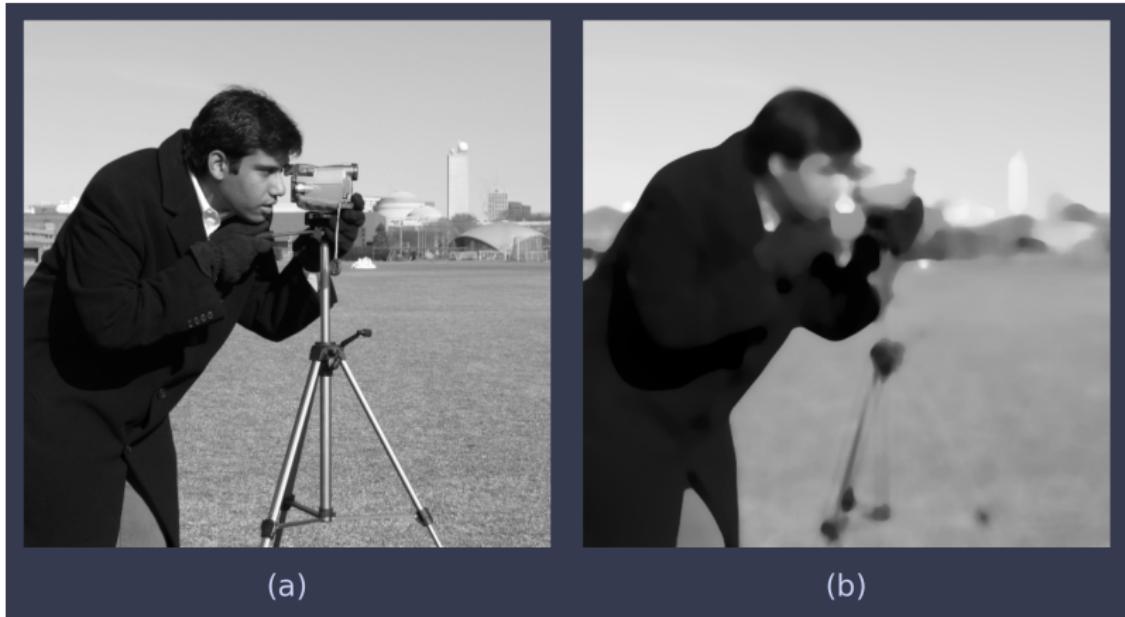


**Figure:** An implementation of the median blur. (a) Original image (b) Median filter applied with kernel size being 5-by-5.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D

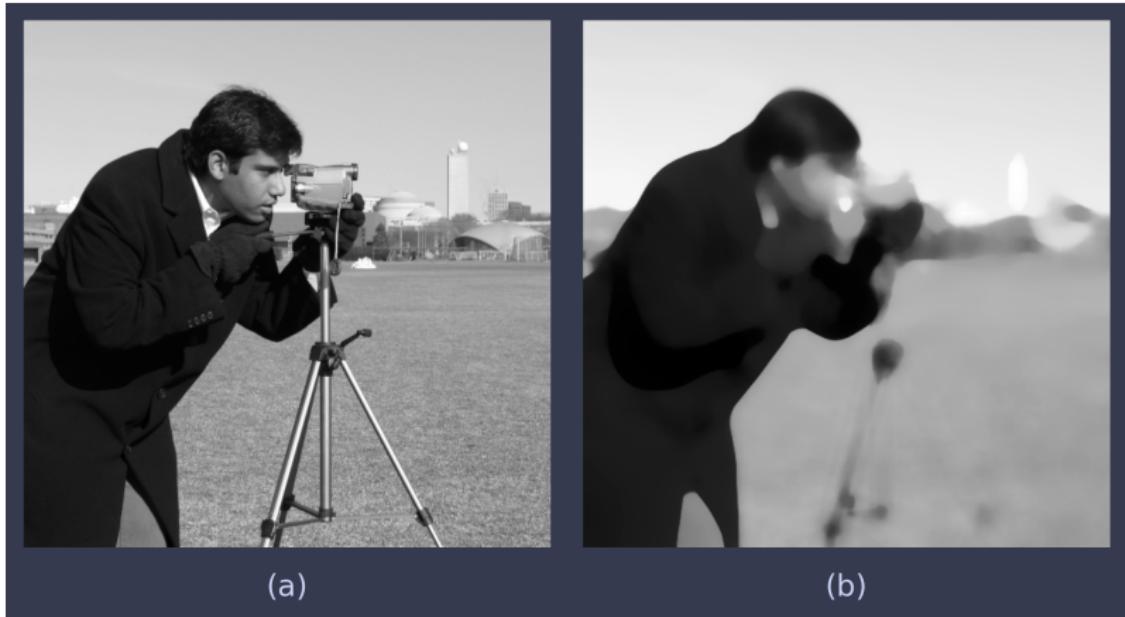


**Figure:** An implementation of the median blur. (a) Original image (b) Median filter applied with kernel size being 9-by-9.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D

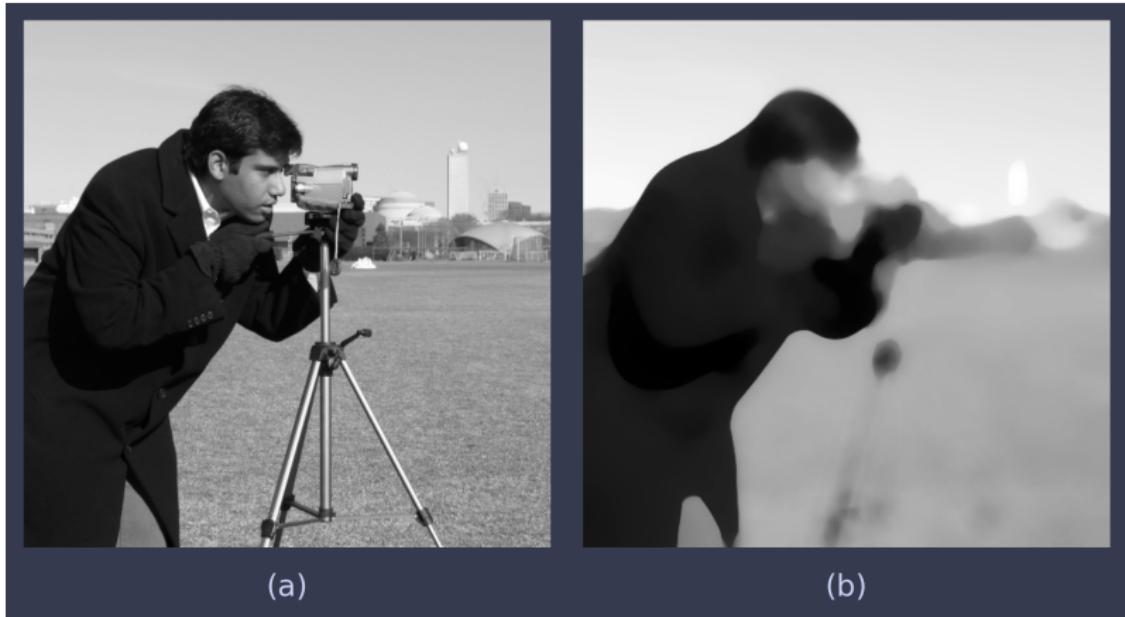


**Figure:** An implementation of the median blur. (a) Original image (b) Median filter applied with kernel size being 13-by-13.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D

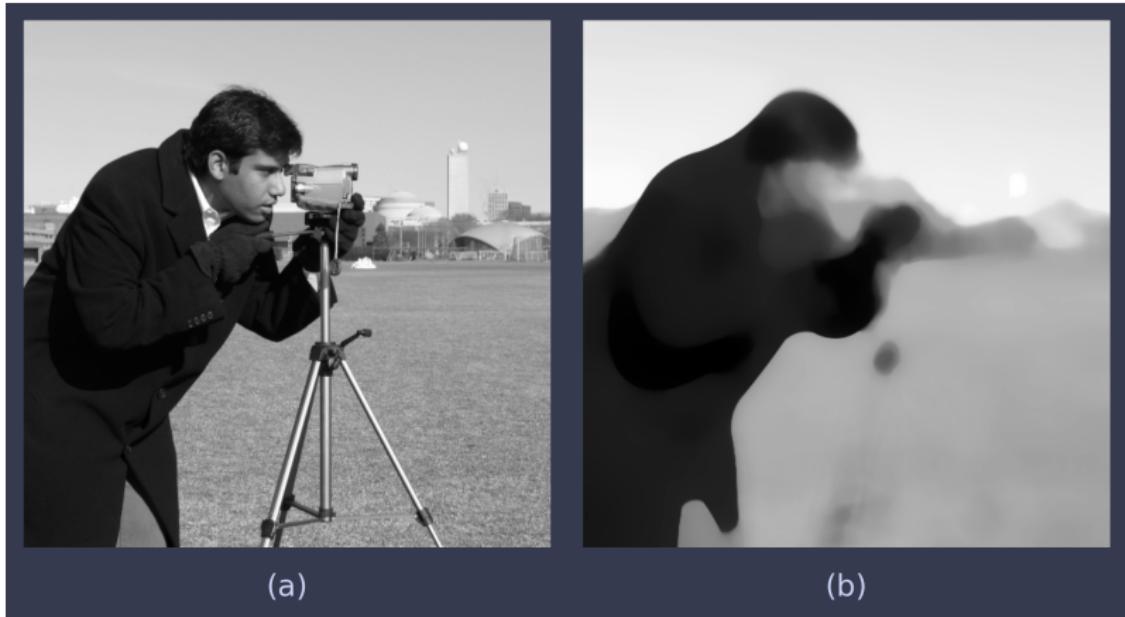


**Figure:** An implementation of the median blur. (a) Original image (b) Median filter applied with kernel size being 17-by-17.

# Blur Filter

Introduction

D. T. McGuiness, Ph.D



**Figure:** An implementation of the median blur. (a) Original image (b) Median filter applied with kernel size being 21-by-21.

## Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

- A bilateral filter is a **non-linear**, **edge-preserving**, and **noise-reducing** smoothing filter for images.
- It replaces the intensity of each pixel with a **weighted average** of intensity values from nearby pixels.
- This weight can be based on a *Gaussian distribution*.
- Crucially, the weights depend not only on Euclidean distance of pixels, but also on the radiometric differences (e.g., range differences, such as colour intensity, depth distance, etc.).

## Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

- Mathematically, the bilateral filter is defined as:

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\cdot)$$

$I^{\text{filtered}}$  is the filtered image;

$I$  is the original input image to be filtered;

$x$  are the coordinates of the current pixel to be filtered;

$\Omega$  is the window centered in  $x$ , so  $x_i \in \Omega$  is another pixel;

$f_r$  is the range kernel for smoothing differences in intensities (this function can be a Gaussian function);

$g_r$  is the spatial (or domain) kernel for smoothing differences in coordinates (this function can be a Gaussian function).

## Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

- Gaussian filtering is a **weighted average** of the intensity of the adjacent positions with a weight decreasing with the spatial distance to the centre position  $\mathbf{p}$ .
- The weight for pixel  $\mathbf{q}$  is defined by the Gaussian  $G_\sigma(||\mathbf{p} - \mathbf{q}||)$ , where  $\sigma$  is a parameter defining the neighbourhood size.
- The strength of this influence depends only on the spatial distance between the pixels and not their values.
- For instance, a bright pixel has a strong influence over an adjacent dark pixel although these two pixel values are quite different.
- As a result, image edges are blurred because pixels across discontinuities are averaged together.

## Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

- The bilateral filter is also defined as a weighted average of nearby pixels, in a manner very similar to Gaussian convolution.
- The difference is that the bilateral filter takes into account the difference in value with the neighbours to preserve edges while smoothing.
- The key idea of the bilateral filter is that for a pixel to influence another pixel, it should not only occupy a nearby location but also have a similar value.
- The bilateral filter, denoted by  $BF[\cdot]$ , is defined by:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s} (||\mathbf{p} - \mathbf{q}||) G_{\sigma_r} (|I_p - I_q|) I_q,$$

where normalization factor  $W_p$  ensures pixel weights sum to 1 and parameters  $\sigma_s$  and  $\sigma_r$  specifies the amount of filtering for image  $I$ .

# Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

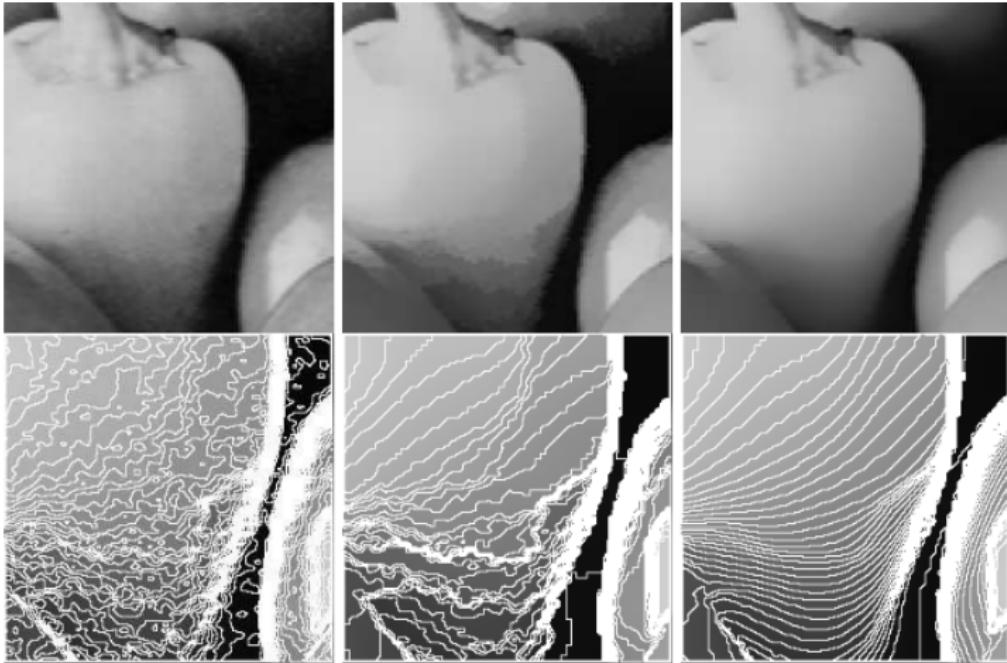


Figure: The staircase effect can be observed between the leftmost and rightmost images [69].

```
image = data.camera()

path = "images/Blurring-Filters/Bilateral-blur-" + str(size)
↪ + ".png"

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.grid(False); ax1.minorticks_off(); ax1.set_xticks([]);
↪ ax1.set_yticks([])
ax1.imshow(image, cmap=cm.gray)

ax2.grid(False); ax2.minorticks_off(); ax2.set_xticks([]);
↪ ax2.set_yticks([])
ax2.imshow(cv.bilateralFilter(image, size, 75, 75),
↪ cmap=cm.gray)

ax1.set_xlabel('(a)', labelpad=10)
ax2.set_xlabel('(b)', labelpad=10)

plt.tight_layout()
plt.grid(False)
```

C.R. 9.71

python

# Bilateral Filter

Introduction

D. T. McGuiness, Ph.D

```
bilateral_blur(9)  
#+end_src
```

C.R. 9.73  
python

## Bilateral Filter

Butterworth Filters

D. T. McGuiness, Ph.D

- The Butterworth filter is implemented in the frequency domain and is designed to have no pass-band or stop-band ripple.
- It can be used in either a lowpass or highpass variant.
- The `cutoff_frequency_ratio` parameter is used to set the cutoff frequency as a fraction of the sampling frequency.
- Given that the Nyquist frequency is half the sampling frequency, this means that this parameter should be a positive floating point value  $< 0.5$ .
- The order of the filter can be adjusted to control the transition width, with higher values leading to a sharper transition between the passband and stopband.

# Bilateral Filter

Butterworth Filters

D. T. McGuiness, Ph.D

(squared) Butterworth filtering (order=3.0, npad=0)

original



lowpass,  $c=0.02$



lowpass,  $c=0.08$



lowpass,  $c=0.16$



highpass,  $c=0.02$



highpass,  $c=0.08$



highpass,  $c=0.16$



Figure



Chapter

# Edge Detection

## Introduction

Edge Detection

## Detection Methods

Gradient Estimation

## Sobel Edge Detection

## Operation Principle

A Visual Comparison

## Canny Edge Detection

Limitations

## Watershed Algorithm

Region Based Segmentation

## Introduction

### Edge Detection

D. T. McGuiness, Ph.D

- Edge detection is a major application of convolution.
- Let's define what an edge is:
  - A location where a sudden change in intensity or colour.
  - A transition between objects or object and background.
  - From a human perspective, it attracts attention.

Images have **noise**, which creates sudden value changes:

- There are three (3) steps in the edge detection process:

**Noise reduction** Suppress noise **without removing edges**.

**Enhance edges** Highlight edges and weaken elsewhere.

- Think of it as a high-pass filter.

**Localise edges** Look at possible edges and eliminate spurious edges (noise related).

- Edges in images are areas with **strong intensity contrasts**:
  - It is a jump in intensity from one pixel to the next.
- Edge detection process significantly reduces the amount of data and **filters out unneeded information**, while **preserving important structural properties** of an image.
- There are different methods for edge detection where the majority can be grouped into two (2) categories:
  1. Gradient,
  2. Laplacian.
- The gradient method detects the edges by looking for the **maximum** and **minimum** in the first derivative of the image.
- The Laplacian method searches for **zero crossings** in the 2<sup>nd</sup> derivative of the image.

## Detection Methods

### Gradient Estimation

D. T. McGuiness, Ph.D

- Estimation of the intensity gradient at a pixel in the  $x$  and  $y$  direction, for an image  $f(\cdot)$ , is given by:

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x-1, y),$$

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y-1).$$

- We can introduce noise smoothing by convoluting with a low pass filter (e.g. mean, Gaussian, etc.).
- The gradient calculation ( $g_x, g_y$ ) can be expressed as:

$$g_x = h_x * f(x, y),$$

$$g_y = h_y * f(x, y).$$

- The **Sobel filter** is used for edge detection.
- It works by calculating the **gradient of image intensity** at each pixel within the image.
  - It finds the direction of the largest increase from light to dark and the rate of change in that direction.
- The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge.
- It also shows how that edge is likely to be oriented.
- The result of applying the filter to a pixel in a region of constant intensity is a zero vector.
- The result of applying it to a pixel on an edge is a vector that points across the edge from darker to brighter values.

- The **Sobel filter** uses two (2) 3-by-3 kernels.
  - Changes in the horizontal ( $x$ ) and vertical ( $y$ ) direction,
- The two (2) kernels are **convolved with the original image** to calculate the approximations of the derivatives.
- If we define  $G_x$  and  $G_y$  as two images that contain the horizontal and vertical derivative approximations respectively, the computations are:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$$

where  $A$  is the **original source image**.

- The  $x$  coordinate is defined as increasing in the right-direction and the  $y$  coordinate is defined as increasing in the down-direction.

- The kernels contain **positive** and **negative** coefficients.
  - The output image will contain positive and negative values.
- To display the image we can:
  - Map the **gradient of zero** onto a half-tone grey level.
    - This makes negative gradients appear darker, and positive gradients appear brighter.
  - Use the absolute values of the gradient map,
    - stretching between 0 and 255.
    - Making very negative and very positive gradients brighter.
- The kernels are **sensitive to horizontal and vertical transitions**.
- The measure of an edge is its **amplitude and angle**.
  - These are readily calculated from  $G_x$  and  $G_y$ .

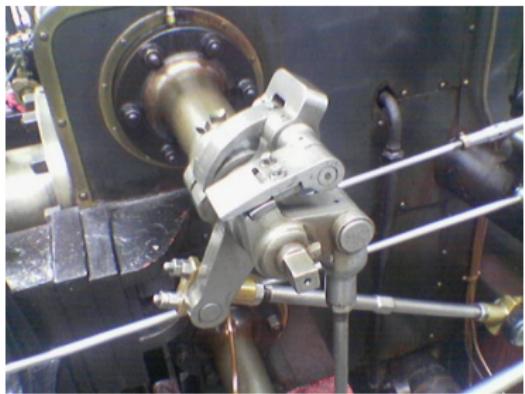
- At each pixel in the image, the gradient approximations given by  $G_x$  and  $G_y$  are combined to give the gradient magnitude ( $G$ ), using:

$$G = \sqrt{G_x^2 + G_y^2}.$$

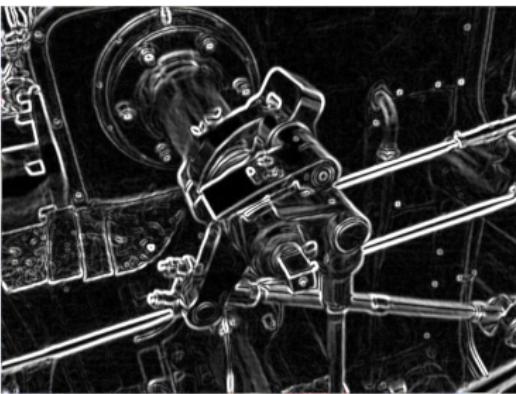
- The gradient's direction is calculated using:

$$\theta = \arctan \left( \frac{G_y}{G_x} \right),$$

where a  $\theta$  value of 0 indicates a vertical edge that is darker on the left side.



**Figure:** An image of a valve.



**Figure:** The image of the valve with sobel filter applied.

Let's implement this in code. The first thing to do is to turn our coloured image into grayscale.

```
def grayscale(self, option="fast"):

    colour_image = imread(self.picture)

    # Creates an empty matrix for data manipulation
    heigth, width = colour_image.shape[:2]
    gray_image = np.zeros((heigth, width), np.uint8)

    # OPTIONS

    return gray_image # returns a gray image
```

C.R. 10.74  
python

```
if option == "fast":  
    # rgb -> grayscale conversion matrix  
    matrix = np.array([[0.07, 0.72, 0.21]]))  
    gray_image = np.sum(colour_image * matrix, axis=2)  
  
if option == "slow":  
    for i in range(height):  
        for j in range(width):  
            gray_image[i, j] = np.clip(  
                0.07 * colour_image[i, j, 0] +  
                0.72 * colour_image[i, j, 1] +  
                0.21 * colour_image[i, j, 2], 0, 255)
```

The **grayscale** has two options:

**fast** This option uses numpy matrix multiplication (i.e., SIMD)

**slow** Uses standard 2 for loop for converting RGB to grayscale.

## Scharr Operator

- This result from an **optimisation**,
  - Minimising weighted mean squared angular error in Fourier domain.
- This optimisation is done under the condition that resulting filters are numerically consistent.
- The matrices really are derivative kernels rather than merely keeping symmetry constraints.
- The optimal 8 bit integer valued 3-by-3 filter is written as:

$$G_x = \begin{bmatrix} 47 & 0 & -47 \\ 162 & 0 & -162 \\ 47 & 0 & -47 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} 47 & 167 & 47 \\ 0 & 0 & 0 \\ -47 & -167 & -47 \end{bmatrix} * A$$

## Prewitt Operator

- Mathematically, the operator uses two 3-by-3 kernels which are convolved with the original image to calculate approximations of the derivatives:
  - One for horizontal changes, and one for vertical.
- If we define  $A$  as the source image, and  $G_x$  and  $G_y$  are two images which at each point contain the horizontal and vertical derivative approximations, the latter are computed as:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * A$$



**Figure:** Grayscale test image of brick wall and bike rack



**Figure:** Gradient magnitude from Sobel-Feldman operator

# Operation Principle

A Visual Comparison

D. T. McGuiness, Ph.D



Figure: Gradient magnitude from Scharr operator

# Operation Principle

A Visual Comparison

D. T. McGuiness, Ph.D



Figure: Gradient magnitude from Roberts Cross operator



**Figure:** Gradient magnitude from Prewitt operator

- Smoothing (noise reduction)
- Find derivatives (gradients) (i.e.,  $G_x$ ,  $G_y$ )
- Find magnitude and orientation of gradient (i.e.,  $G$ )
- Non-maximum suppression:
  - Thin multi-pixel wide “ridges” down to single pixel width
- Linking and threshold (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

## Canny Edge Detection

Limitations

D. T. McGuiness, Ph.D

- The result is **binary**.
  - You sometimes need a measure of 'how much' the edge qualifies as an edge.
- The amount of parameters leads to infinitely tweaking for getting just that little better result.
- You still need to connect the resulting edges to extract the complete edges that seem so obvious for the human eye.
- Due to the Gaussian smoothing: the location of the edges might be off, depending on the size of the Gaussian kernel.
- The method has problems with corners and junctions:
  - The Gaussian smoothing blurs them out, making them harder to detect (same goes for the edges themselves).
  - The corner pixels look in the wrong directions for their neighbours, leaving open ended edges, and missing junctions.

- Consider the coins image below, the coins are touching each other.
- Even if you threshold it, it will be touching each other.



**Figure:** Random assortment of coins.

- We start with finding an approximate estimate of the coins.
- For that, we can use the Otsu's binarization.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('watershed-coins.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
ret, thresh =
    cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
```

C.R. 10.76  
python

- Now we need to remove any small white noises in the image.
- For that we can use morphological opening.
- To remove any small holes in the object, we can use morphological closing.
- So, now we know for sure that region near to center of objects are foreground and region much away from the object are background.
- Only region we are not sure is the boundary region of coins.

Below you will see an image containing coins of using edge detection and other methods taught to segment and label each coin individually.



**Figure:** A collection of coins from `skimage.coins()`.

Example

Start simple and import the necessary modules.

```
coins = data.coins()
hist, hist_centers = histogram(coins)

fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes[0].imshow(coins, cmap=plt.cm.gray)
axes[0].set_axis_off()
axes[1].plot(hist_centers, hist, lw=2)
axes[1].set_title('histogram of gray values')
```

C.R. 10.77  
python

Here let's have a look at some modules worth discussing:

**skimage** image processing library.

**skimage.exposure** Contains histogram based operations.

Solution

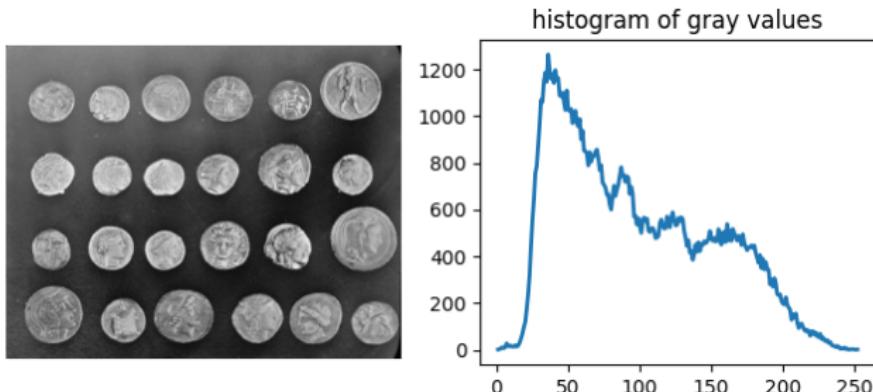


Figure: Histogram of the grey values.

Solution

**Thresholding:** A simple way to segment the coins is to choose a threshold based on the histogram of grey values.

```
fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True)

axes[0].imshow(coins > 100, cmap=plt.cm.gray)
axes[0].set_title('coins > 100')

axes[1].imshow(coins > 150, cmap=plt.cm.gray)
axes[1].set_title('coins > 150')

for a in axes:
    a.set_axis_off()

fig.tight_layout()
```

C.R. 10.78  
python

Solution

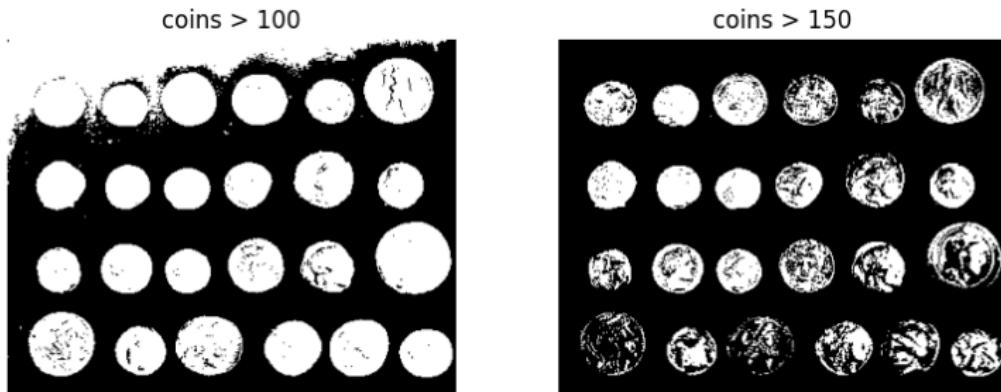


Figure: The original image with thresholding applied.

Solution

Unfortunately, this gives a binary image that either misses significant parts of coins or merges parts of the background with them.

Next, Try to delineate the contours of the coins using edge-based segmentation.

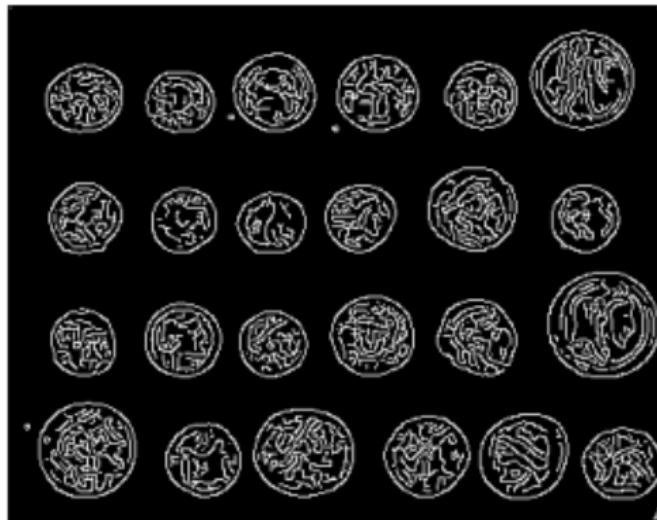
To do this, we first get the edges of features using the Canny edge-detector.

```
from skimage.feature import canny  
  
edges = canny(coins)  
  
fig, ax = plt.subplots(figsize=(4, 3))  
ax.imshow(edges, cmap=plt.cm.gray)  
ax.set_title('Canny detector')  
ax.set_axis_off()
```

C.R. 10.79  
python

Solution

## Canny detector



**Figure:** Application of the Canny Edge Detection on coins.

These contours are then filled using mathematical morphology.

```
from scipy import ndimage as ndi

fill_coins = ndi.binary_fill_holes(edges)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(fill_coins, cmap=plt.cm.gray)
ax.set_title('filling the holes')
ax.set_axis_off()
```

C.R. 10.80  
python

Solution

filling the holes

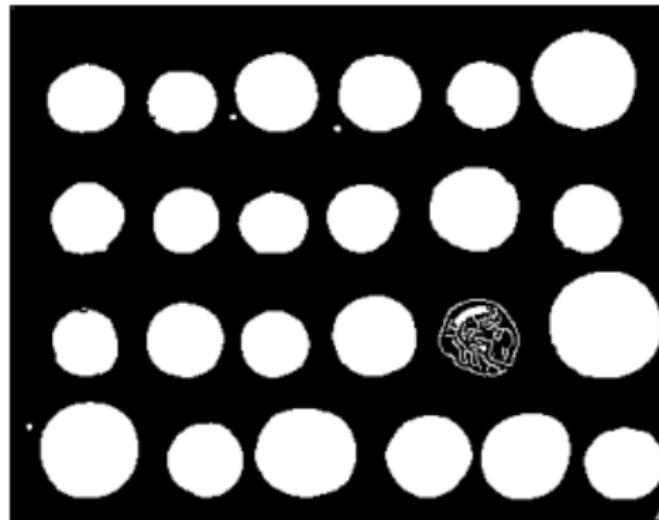


Figure: Image with morphological operations applied

Small spurious objects are easily removed by setting a minimum size for valid objects.

```
from skimage import morphology  
  
coins_cleaned = morphology.remove_small_objects(fill_coins, 21)  
  
fig, ax = plt.subplots(figsize=(4, 3))  
ax.imshow(coins_cleaned, cmap=plt.cm.gray)  
ax.set_title('removing small objects')  
ax.set_axis_off()
```

C.R. 10.81  
python

Solution

removing small objects

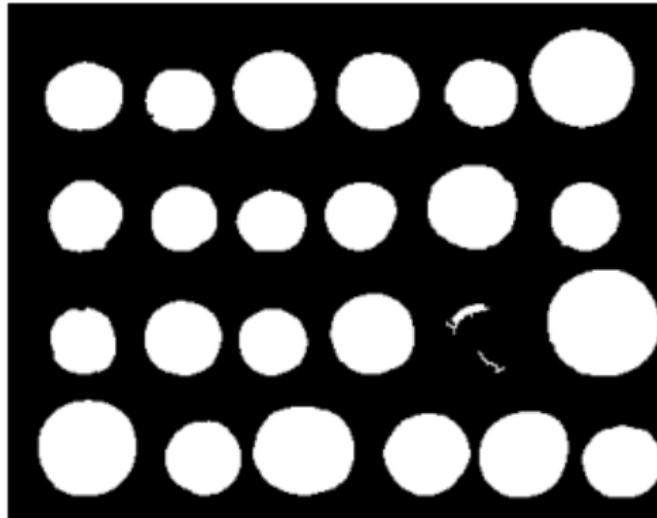


Figure: Image where small unwanted parts removed.

## Watershed Algorithm

Region Based Segmentation

D. T. McGuiness, Ph.D

We therefore try a region-based method using the watershed transform. First, we find an elevation map using the Sobel gradient of the image.

```
from skimage.filters import sobel

elevation_map = sobel(coins)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(elevation_map, cmap=plt.cm.gray)
ax.set_title('elevation map')
ax.set_axis_off()
```

C.R. 10.82  
python

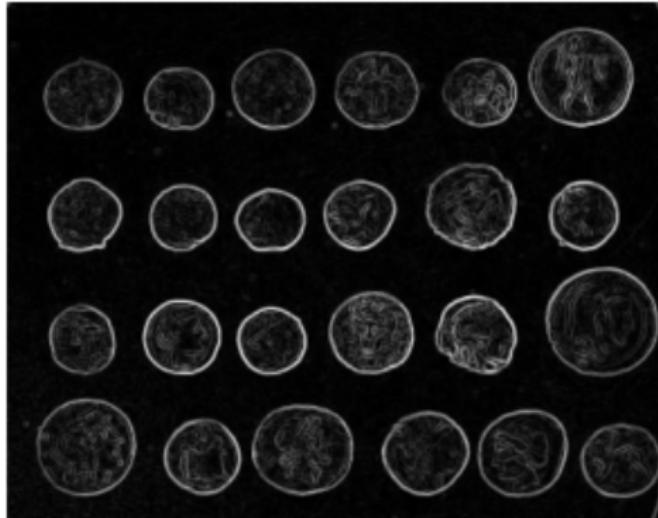
Solution

# Watershed Algorithm

Region Based Segmentation

D. T. McGuiness, Ph.D

elevation map



**Figure:** Here the elevation map is applied.

## Watershed Algorithm

Region Based Segmentation

D. T. McGuiness, Ph.D

Next we find markers of the background and the coins based on the extreme parts of the histogram of gray values.

```
markers = np.zeros_like(coins)
markers[coins < 30] = 1
markers[coins > 150] = 2

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(markers, cmap=plt.cm.nipy_spectral)
ax.set_title('markers')
ax.set_axis_off()
```

C.R. 10.83  
python

Solution

## Watershed Algorithm

Region Based Segmentation

D. T. McGuiness, Ph.D

Finally, we use the watershed transform to fill regions of the elevation map starting from the markers determined above:

```
from skimage import segmentation

segmentation_coins = segmentation.watershed(elevation_map,
    ↪ markers)

fig, ax = plt.subplots(figsize=(4, 3))
ax.imshow(segmentation_coins, cmap=plt.cm.gray)
ax.set_title('segmentation')
ax.set_axis_off()
```

C.R. 10.84  
python

Solution

## Watershed Algorithm

Region Based Segmentation

D. T. McGuiness, Ph.D

This last method works even better, and the coins can be segmented and labeled individually.

```
from skimage.color import label2rgb

segmentation_coins = ndi.binary_fill_holes(segmentation_coins -
    ↪ 1)
labeled_coins, _ = ndi.label(segmentation_coins)
image_label_overlay = label2rgb(labeled_coins, image=coins,
    ↪ bg_label=0)

fig, axes = plt.subplots(1, 2, figsize=(8, 3), sharey=True)
axes[0].imshow(coins, cmap=plt.cm.gray)
axes[0].contour(segmentation_coins, [0.5], linewidths=1.2,
    ↪ colors='y')
axes[1].imshow(image_label_overlay)

for a in axes:
    a.set_axis_off()

fig.tight_layout()
plt.show()
```

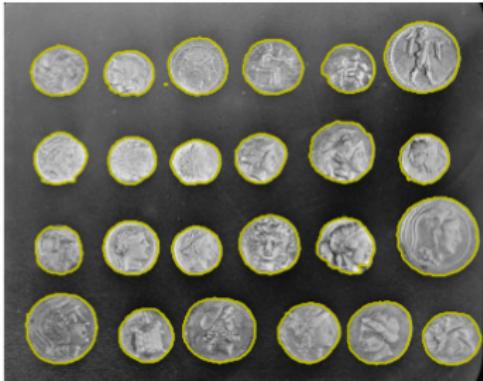
C.R. 10.85  
python

Solution

## Watershed Algorithm

## Region Based Segmentation

D. T. McGuiness, Ph.D



## Figure



Chapter

# Processing Applications

**Finding Local Maxima**  
**Chan-Vese Segmentation**  
**Region Adjacency Graph**  
**Threshold**  
**Trainable Segmentation**

Feature Importance  
Fitting New Images

**Segmenting Human Cells**  
Estimating Mitotic Index

- The `peak_local_max` function returns the coordinates of local peaks (maxima) in an image.
  - i.e., pixels surrounded by pixels with lower intensity.
- Internally, a **maximum filter** is used for finding local maxima. This operation **dilates** the original image and merges neighboring local maxima closer than the size of the dilation.
- Locations where the original image is equal to the dilated image are returned as local maxima.

This is a non-linear filter as it does **NOT** abide by the superposition principle.

```
from scipy import ndimage as ndi
import matplotlib.pyplot as plt
from skimage.feature import peak_local_max
from skimage import data, img_as_float

im = img_as_float(data.coins())

# image_max is the dilation of im with a 20*20 structuring
# element
# It is used within peak_local_max function
image_max = ndi.maximum_filter(im, size=20, mode='constant')

# Comparison between image_max and im to find the coordinates of
# local maxima
coordinates = peak_local_max(im, min_distance=20)
```

C.R. 11.87

python

```
fig, axes = plt.subplots(1, 3, figsize=(8, 3), sharex=True,  
↪ sharey=True)  
ax = axes.ravel()  
ax[0].imshow(im, cmap=plt.cm.gray)  
ax[0].axis('off')  
ax[0].set_title('Original')  
  
ax[1].imshow(image_max, cmap=plt.cm.gray)  
ax[1].axis('off')  
ax[1].set_title('Maximum filter')
```

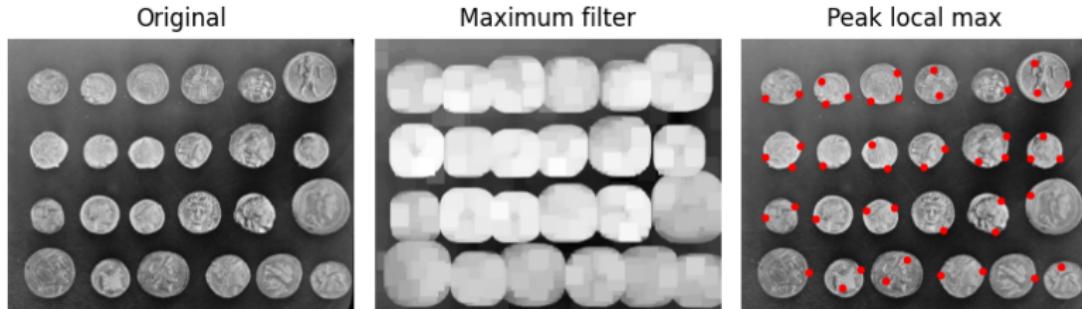
C.R. 11.88

python

```
ax[2].imshow(im, cmap=plt.cm.gray)
ax[2].autoscale(False)
ax[2].plot(coordinates[:, 1], coordinates[:, 0], 'r.')
ax[2].axis('off')
ax[2].set_title('Peak local max')

fig.tight_layout()

plt.show()
```



**Figure:** A visual comparison of the local maxima operation.

- The *Chan-Vese* segmentation algorithm is designed to segment objects **without** clearly defined boundaries.
- This algorithm is based on level sets which are evolved iteratively to minimize an energy, which is defined by weighted values corresponding to the sum of differences intensity from the average value outside the segmented region, the sum of differences from the average value inside the segmented region, and a term which is dependent on the length of the boundary of the segmented region.
- This algorithm was first proposed by Tony Chan and Luminita Vese, in a publication entitled *An Active Contour Model Without Edges*.

- This implementation of the algorithm is somewhat simplified in the sense that the area factor " $\nu$ " described in the original paper is not implemented, and is only suitable for grayscale images.
- Typical values for  $\lambda_1$  and  $\lambda_2$  are 1. If the "background" is very different from the segmented object in terms of distribution (for example, a uniform black image with figures of varying intensity), then these values should be different from each other.
- Typical values for  $\mu$  are between 0 and 1, though higher values can be used when dealing with shapes with very ill-defined contours.
- The algorithm also returns a list of values which corresponds to the energy at each iteration. This can be used to adjust the various parameters described above.

```
import matplotlib.pyplot as plt
from skimage import data, img_as_float
from skimage.segmentation import chan_vese

image = img_as_float(data.camera())

cv = chan_vese(
    image,
    mu=0.25,
    lambda1=1,
    lambda2=1,
    tol=1e-3,
    max_num_iter=200,
    dt=0.5,
    init_level_set="checkerboard",
    extended_output=True,
)
```

C.R. 11.90

python

```
fig, axes = plt.subplots(2, 2, figsize=(8, 8))
ax = axes.flatten()

ax[0].imshow(image, cmap="gray")
ax[0].set_axis_off()
ax[0].set_title("Original Image", fontsize=12)

ax[1].imshow(cv[0], cmap="gray")
ax[1].set_axis_off()
title = f'Chan-Vese segmentation - {len(cv[2])} iterations'
ax[1].set_title(title, fontsize=12)
```

C.R. 11.91

python

```
ax[2].imshow(cv[1], cmap="gray")
ax[2].set_axis_off()
ax[2].set_title("Final Level Set", fontsize=12)

ax[3].plot(cv[2])
ax[3].set_title("Evolution of energy over iterations",
    fontsize=12)

fig.tight_layout()
plt.show()
```

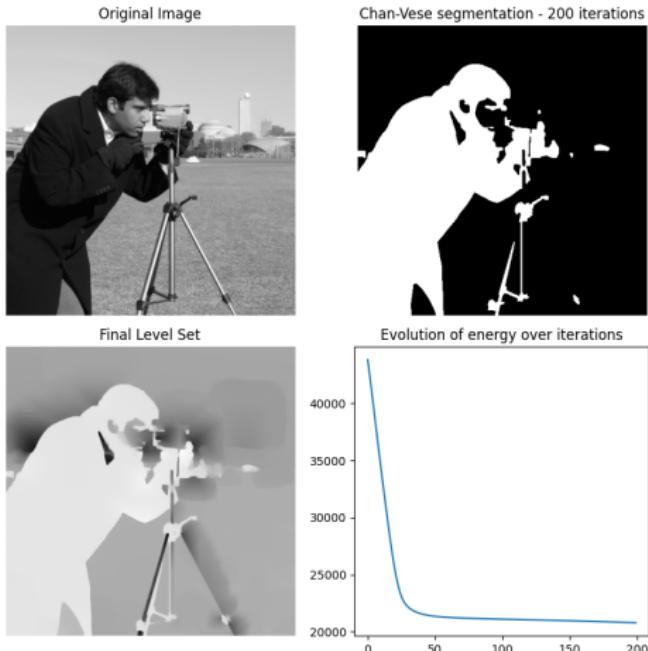


Figure: Implementation of the Chan-Vese Algorithm.

- This example constructs a Region Adjacency Graph (RAG) and merges regions which are similar in color.
- We construct a RAG and define edges as the difference in mean color. We then join regions with similar mean color.

C.R. 11.92

python

```
from skimage import data, segmentation, color
from skimage import graph
from matplotlib import pyplot as plt

img = data.coffee()

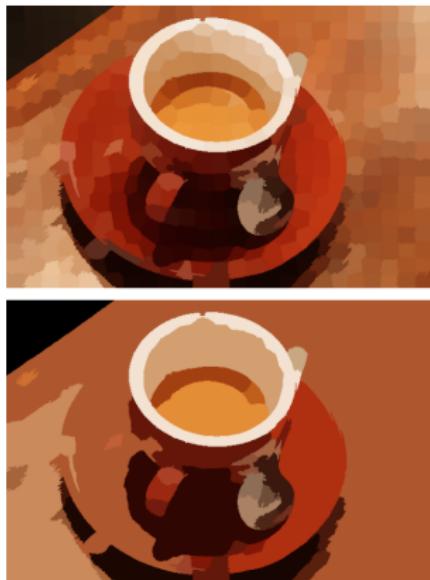
labels1 = segmentation.slic(img, compactness=30, n_segments=400,
                           start_label=1)
out1 = color.label2rgb(labels1, img, kind='avg', bg_label=0)

g = graph.rag_mean_color(img, labels1)
labels2 = graph.cut_threshold(labels1, g, 29)
out2 = color.label2rgb(labels2, img, kind='avg', bg_label=0)
```

C.R. 11.93

python

```
fig, ax = plt.subplots(nrows=2, sharex=True, sharey=True,  
                      figsize=(6, 8))  
  
ax[0].imshow(out1)  
ax[1].imshow(out2)  
  
for a in ax:  
    a.axis('off')  
  
plt.tight_layout()
```



Figure

- A pixel-based segmentation is computed here using **local features** based on local intensity, edges and textures at different scales.
- A user-provided mask is used to identify different regions.
- The pixels of the mask are used to train a **random-forest classifier** from **scikit-learn**.
- Unlabeled pixels are then labeled from the prediction of the classifier.
- This segmentation algorithm is called **trainable segmentation** in other software such as ilastik or ImageJ (where it is also called “weka segmentation”).

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data, segmentation, feature, future
from sklearn.ensemble import RandomForestClassifier
from functools import partial

full_img = data.skin()

img = full_img[:900, :900]
```

```
# Build an array of labels for training the segmentation.  
# Here we use rectangles but visualization libraries such as  
→ plotly  
# (and napari?) can be used to draw a mask on the image.  
training_labels = np.zeros(img.shape[:2], dtype=np.uint8)  
training_labels[:130] = 1  
training_labels[:170, :400] = 1  
training_labels[600:900, 200:650] = 2  
training_labels[330:430, 210:320] = 3  
training_labels[260:340, 60:170] = 4  
training_labels[150:200, 720:860] = 4
```

C.R. 11.96

python

```
sigma_min = 1
sigma_max = 16
features_func = partial(
    feature.multiscale_basic_features,
    intensity=True,
    edges=False,
    texture=True,
    sigma_min=sigma_min,
    sigma_max=sigma_max,
    channel_axis=-1,
)
```

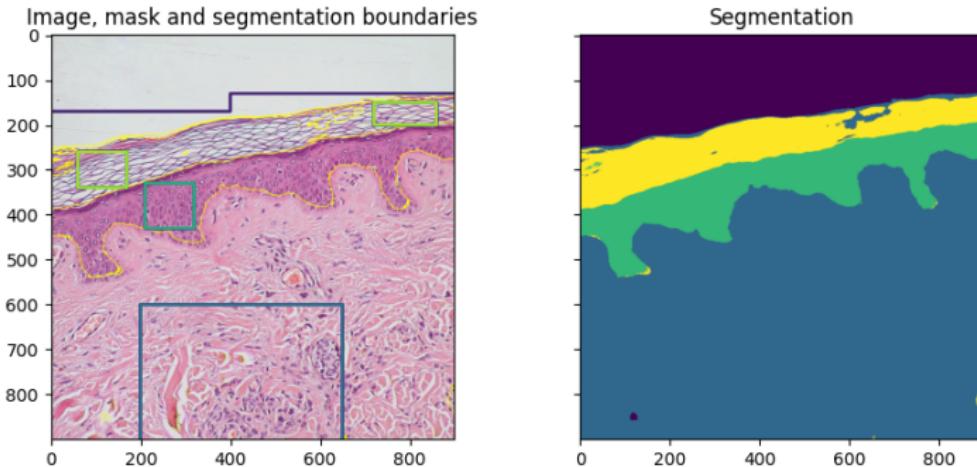
C.R. 11.97  
python

```
features = features_func(img)
clf = RandomForestClassifier(n_estimators=50, n_jobs=-1,
    ↵ max_depth=10, max_samples=0.05)
clf = future.fit_segmenter(training_labels, features, clf)
result = future.predict_segmenter(features, clf)
```

C.R. 11.98

python

```
fig, ax = plt.subplots(1, 2, sharex=True, sharey=True,  
                     figsize=(9, 4))  
ax[0].imshow(segmentation.mark_boundaries(img, result,  
                                             mode='thick'))  
ax[0].contour(training_labels)  
ax[0].set_title('Image, mask and segmentation boundaries')  
ax[1].imshow(result)  
ax[1].set_title('Segmentation')  
fig.tight_layout()
```



**Figure:** A visual comparison of the original image with given masks and the segmented image on the right.

## Trainable Segmentation

Feature Importance

D. T. McGuiness, Ph.D

- We inspect below the importance of the different features, as computed by `scikit-learn`.
- Intensity features have a much higher importance than texture features.
- It can be tempting to use this information to reduce the number of features given to the classifier, in order to reduce the computing time.
- However, this can lead to over-fitting and a degraded result at the boundary between regions.

C.R. 11.99

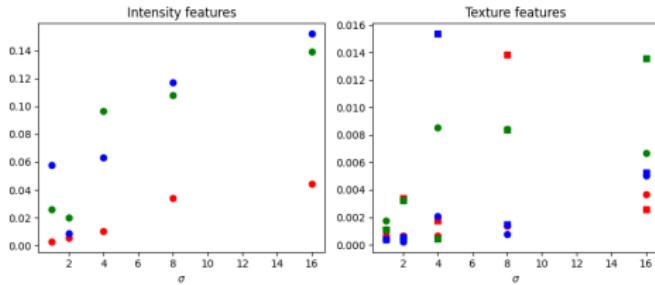
python

```
fig, ax = plt.subplots(1, 2, figsize=(9, 4))
l = len(clf.feature_importances_)
feature_importance = (
    clf.feature_importances_[: l // 3],
    clf.feature_importances_[l // 3 : 2 * l // 3],
    clf.feature_importances_[2 * l // 3 :],
)
sigmas = np.logspace(
    np.log2(sigma_min),
    np.log2(sigma_max),
    num=int(np.log2(sigma_max) - np.log2(sigma_min) + 1),
    base=2,
    endpoint=True,
)
```

C.R. 11.100

python

```
for ch, color in zip(range(3), ['r', 'g', 'b']):  
    ax[0].plot(sigmas, feature_importance[ch][::3], 'o',  
               color=color)  
    ax[0].set_title("Intensity features")  
    ax[0].set_xlabel("$\sigma$")  
for ch, color in zip(range(3), ['r', 'g', 'b']):  
    ax[1].plot(sigmas, feature_importance[ch][1::3], 'o',  
               color=color)  
    ax[1].plot(sigmas, feature_importance[ch][2::3], 's',  
               color=color)  
    ax[1].set_title("Texture features")  
    ax[1].set_xlabel("$\sigma$")  
  
fig.tight_layout()
```



Figure

## Trainable Segmentation

Fitting New Images

D. T. McGuiness, Ph.D

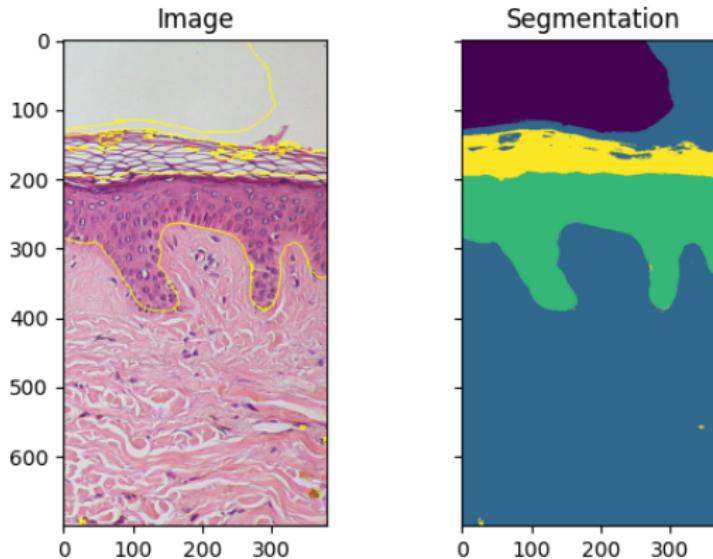
- If we have several images of similar objects acquired in similar conditions, we can use the classifier trained with `fit_segmenter` to segment other images.
- In the example below we just use a different part of the image.

This image was **NOT** seen by the algorithm.

C.R. 11.101

python

```
img_new = full_img[:700, 900:]  
  
features_new = features_func(img_new)  
result_new = future.predict_segmenter(features_new, clf)  
fig, ax = plt.subplots(1, 2, sharex=True, sharey=True,  
                     figsize=(6, 4))  
ax[0].imshow(segmentation.mark_boundaries(img_new, result_new,  
                                         mode='thick'))  
ax[0].set_title('Image')  
ax[1].imshow(result_new)  
ax[1].set_title('Segmentation')  
fig.tight_layout()  
  
plt.show()
```



**Figure:** Comparing the two image with the original on the left and the segmented image on the right. As can be seen the image is segmented properly.

- In this example, we analyse a microscopy image of human cells.
- We use data provided by Jason Moffat through *CellProfiler*.

C.R. 11.102

python

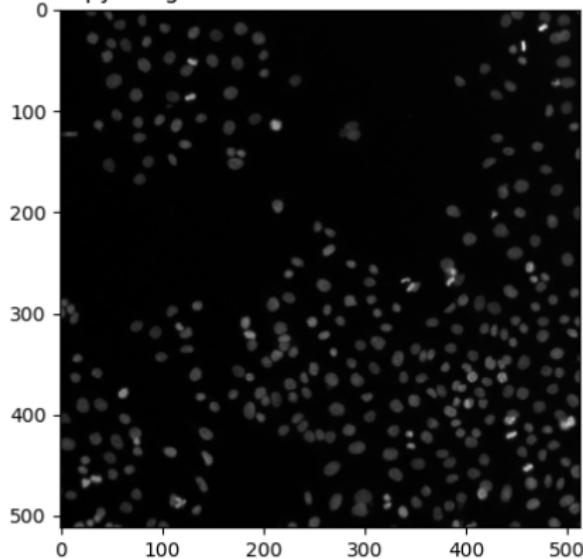
```
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage as ndi

import skimage as ski

image = ski.data.human_mitosis()

fig, ax = plt.subplots()
ax.imshow(image, cmap='gray')
ax.set_title('Microscopy image of human cells stained for
    nuclear DNA')
plt.show()
```

Microscopy image of human cells stained for nuclear DNA



**Figure:** An example image of the human cells which are chemically treated to show nuclear DNA.

- We can see many cell nuclei on a dark background.
- Most of them are smooth and have an elliptical shape.
- However, we can distinguish some brighter spots corresponding to nuclei undergoing mitosis (cell division).
- Another way of visualizing a grayscale image is contour plotting:

```
fig, ax = plt.subplots(figsize=(5, 5))
qcs = ax.contour(image, origin='image')
ax.set_title('Contour plot of the same raw image')
plt.show()
```

C.R. 11.103  
python

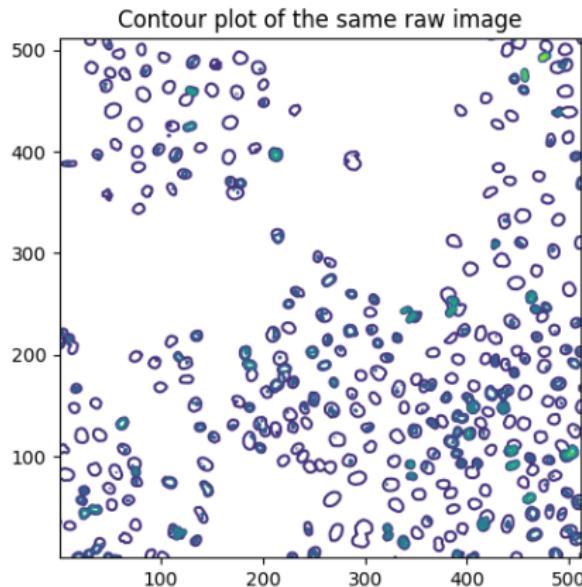


Figure: The same image with contours highlighted.

- The contour lines are drawn at these levels:

```
qcs.levels
```

C.R. 11.104  
python

```
array([ 0.,  40.,  80., 120., 160., 200., 240., 280.])
```

C.R. 11.105  
text

- Each level has, respectively, the following number of segments:

```
# Levels without segments still contain one empty array, account
→ for that
[len(seg) if seg[0].size else 0 for seg in qcs.allsegs]
```

C.R. 11.106  
python

```
[0, 320, 270, 48, 19, 3, 1, 0]
```

C.R. 11.107  
text

## Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

- Cell biology uses the mitotic index to quantify cell division and, hence, cell proliferation.
- By definition, it is the ratio of cells in mitosis over the total number of cells.
- To analyze the previous image, we are therefore interested in two thresholds: one separating the nuclei from the background, the other separating the dividing nuclei (brighter spots) from the non-dividing nuclei.
- To separate these three different classes of pixels, we resort to Multi-Otsu Thresholding.

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

C.R. 11.108

python

```
thresholds = ski.filters.threshold_multiotsu(image, classes=3)
regions = np.digitize(image, bins=thresholds)

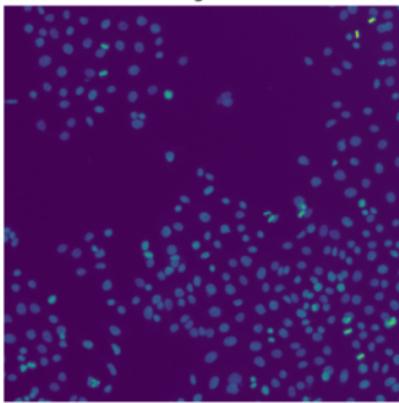
fig, ax = plt.subplots(ncols=2, figsize=(10, 5))
ax[0].imshow(image)
ax[0].set_title('Original')
ax[0].set_axis_off()
ax[1].imshow(regions)
ax[1].set_title('Multi-Otsu thresholding')
ax[1].set_axis_off()
plt.show()
```

## Segmenting Human Cells

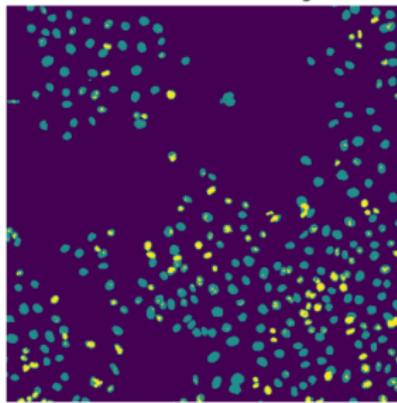
## Estimating Mitotic Index

D. T. McGuiness, Ph.D

Original



## Multi-Otsu thresholding



## Figure

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

- Since there are overlapping nuclei, thresholding is not enough to segment all the nuclei. If it were, we could readily compute a mitotic index for this sample:

```
cells = image > thresholds[0]
dividing = image > thresholds[1]
labeled_cells = ski.measure.label(cells)
labeled_dividing = ski.measure.label(dividing)
naive_mi = labeled_dividing.max() / labeled_cells.max()
print(naive_mi)
```

C.R. 11.109

python

0.7847222222222222

C.R. 11.110

text

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D.

- Whoa, this can't be! The number of dividing nuclei

```
print(labeled_dividing.max())
```

C.R. 11.111  
python

226

C.R. 11.112  
text

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

- is overestimated, while the total number of cells

```
print(labeled_cells.max())
```

C.R. 11.113  
python

288

C.R. 11.114  
text

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

- is underestimated.

```
fig, ax = plt.subplots(ncols=3, figsize=(15, 5))
ax[0].imshow(image)
ax[0].set_title('Original')
ax[0].set_axis_off()
ax[1].imshow(dividing)
ax[1].set_title('Dividing nuclei?')
ax[1].set_axis_off()
ax[2].imshow(cells)
ax[2].set_title('All nuclei?')
ax[2].set_axis_off()
plt.show()
```

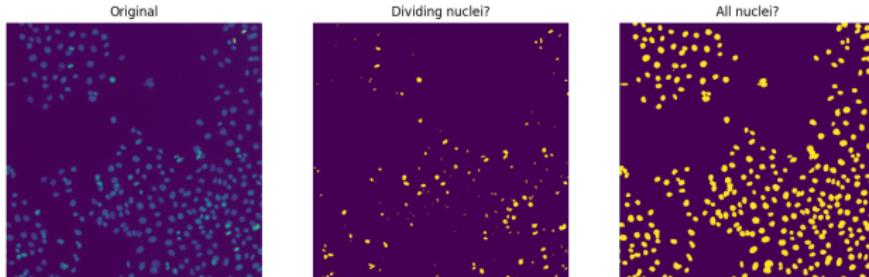
C.R. 11.115

python

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D.



Figure

# Segmenting Human Cells

Estimating Mitotic Index

D. T. McGuiness, Ph.D

**CCD** Charge Coupled Device. 94



Chapter

# Bibliography



# Bibliography

List of References

D. T. McGuiness, Ph.D

- [1] Michael F Deering. "The limits of human vision". In: *2nd international immersive projection technology workshop*. Vol. 2. 1. 1998.
- [2] Hawesthoughts. *Colorblindness Wheel*. 2023. URL: [https://commons.wikimedia.org/wiki/File:Color\\_blindness\\_wheels.svg](https://commons.wikimedia.org/wiki/File:Color_blindness_wheels.svg).
- [3] Colour Blind Awareness. *Types of Colour Blindness*. 2024. URL: <https://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/>.
- [4] Colour Blind Awareness. *Inherited Colour Vision Deficiency*. 2024. URL: <https://www.colourblindawareness.org/colour-blindness/causes-of-colour-blindness/inherited-colour-vision-deficiency/>.
- [5] 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.
- [6] Susanne Kohl et al. "Achromatopsia". In: (2018).
- [7] American Optometric Association. *Color vision deficiency*. 2024. URL: <https://www.aoa.org/healthy-eyes/eye-and-vision-conditions/color-vision-deficiency?sso=y>.

# Bibliography

List of References

D. T. McGuiness, Ph.D

- [8] 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/>.
- [9] Hyperphysics. *The Color-Sensitive Cones*. 2024. URL: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/colcon.html>.
- [10] Zoltan Derzsi and Robert Volcic. "Not only perception but also grasping actions can obey Weber's law". In: *Cognition* 237 (2023), p. 105465.
- [11] Edward H Adelson. "Checkershadow illusion. 1995". In: URL [http://web.mit.edu/per-sci/people/adelson/checkershadow\\_illusion.html](http://web.mit.edu/per-sci/people/adelson/checkershadow_illusion.html) (2005).
- [12] Ali Solomon. *New Yorker Optical Illusions*. 2021. URL: <https://www.newyorker.com/humor/daily-shouts/apartment-optical-illusions>.
- [13] Chloe Farand. *Duck or rabbit? The 100-year-old optical illusion that could tell you how creative you are*. 2021. URL: <https://www.independent.co.uk/news/science/duck-and-rabbit-illusion-b1821663.html>.
- [14] BBC. *Optical illusion: Dress colour debate goes global*. 2015. URL: <https://www.bbc.com/news/uk-scotland-highlands-islands-31656935>.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [15] PolBr. *sRGB colors situated at calculated position in CIE 1931 chromaticity diagram (edited from CIExy1931\_sRGB.svg)*. Luminance Y set 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).
- [16] *Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB*. Standard. Geneva, CH: International Electrotechnical Commission, 1999.
- [17] 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>.
- [18] Danny Pascale. "A review of rgb color spaces... from xyy to r'g'b''. In: *Babel Color* 18 (2003), pp. 136–152.
- [19] Andrew Rodney. "The role of working spaces in Adobe applications". In: *Technical Paper. Adobe. Retrieved* (2008), pp. 05–09.
- [20] *Photography and graphic technology – Extended colour encodings for digital image storage, manipulation and interchange*. Standard. Geneva, CH: International Organization for Standardization, 2013.

# Bibliography

List of References

D. T. McGuiness, Ph.D

- [21] 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.
- [22] *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.
- [23] Inc. Reproductions. *Understanding the Differences Between RGB and CYMK Colors*. 2024. URL: <https://reproductionsinc.com/understanding-the-differences-between-rgb-and-cymk-colors/>.
- [24] Slippens. *Three examples of color halftoning with CMYK separations*. 2009. URL: <https://en.wikipedia.org/wiki/File:Halftoningcolor.svg>.
- [25] 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).
- [26] 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).

# Bibliography

List of References

D. T. McGuiness, Ph.D

- [27] Adobe. *Lossy vs Lossless Compression Differences and When to Use*. 2024. URL: <https://www.adobe.com/uk/creativecloud/photography/discover/lossy-vs-lossless.html>.
- [28] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.
- [29] Alistair Moffat. "Huffman coding". In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–35.
- [30] 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).
- [31] Mark Stanford Robbins. "Electron-multiplying charge coupled devices—emccds". In: *Single-Photon Imaging* (2011), pp. 103–121.
- [32] Bedabrata Pain et al. "A back-illuminated megapixel CMOS image sensor". In: (2005).
- [33] Estrin. *Kodak's First Digital Moment*. 2015. URL: <https://archive.nytimes.com/lens.blogs.nytimes.com/2015/08/12/kodaks-first-digital-moment/>.
- [34] Teledyne. *Types of Camera Sensor*. 2024. URL: <https://www.photometrics.com/learn/camera-basics/types-of-camera-sensor>.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [35] 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).
- [36] 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.
- [37] Filya1. *CMOS image sensor*. 2009. URL: <https://en.m.wikipedia.org/wiki/File:Matrixw.jpg>.
- [38] Olympus. *Introduction to CMOS Image Sensors*. 2024. URL: <https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/cmosimagesensors/>.
- [39] 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/>.
- [40] Junichi Nakamura. *Image sensors and signal processing for digital still cameras*. CRC press, 2017.
- [41] 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.
- [42] Ralph Dammel. *Diazonaphthoquinone-based resists*. Vol. 11. SPIE press, 1993.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [43] 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).
- [44] FlickreviewR. *Sony F828 Camera*. 2009. URL: [https://commons.wikimedia.org/wiki/File:Sony\\_F828.jpg](https://commons.wikimedia.org/wiki/File:Sony_F828.jpg).
- [45] teledyneimage. *Linearity: Image Topics*. 2024. URL: <https://www.photometrics.com/learn/imaging-topics/linearity>.
- [46] Teledyne. *How to Evaluate Camera Sensitivity*. 2021. URL: <https://www.flir.com/discover/iis/machine-vision/how-to-evaluate-camera-sensitivity/>.
- [47] TeledyneCurrent. *Dark Current*. 2024. URL: <https://www.photometrics.com/learn/advanced-imaging/dark-current>.
- [48] Crisp. *Getting out of Auto: Understanding ISO on your digital camera*. 2016. URL: <https://newatlas.com/photography-iso-range-setting-guide/44599/>.
- [49] EdmundOptics. *Anatomy of a Lens*. 2024. URL: <https://www.edmundoptics.com/knowledge-center/application-notes/imaging/the-anatomy-of-a-lens/>.
- [50] DarkRoom. *Effects of Focal Length and Angle*. 2024. URL: <https://thedarkroom.com/focal-length/>.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [51] 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).
- [52] DrBob. *Simple zoom lens*. 2005. URL: <https://commons.wikimedia.org/wiki/File:Zoomlens1.svg>.
- [53] 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).
- [54] European Broadcasting Union. "Enhanced 625-line Phased Alternate Line (PAL) television". In: (1997).
- [55] 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.
- [56] Mary C Potter et al. "Detecting meaning in RSVP at 13 ms per picture". In: *Attention, Perception, & Psychophysics* 76 (2014), pp. 270–279.
- [57] 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.
- [58] Rudolf F Graf. *Modern dictionary of electronics*. Elsevier, 1999.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [59] 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.
- [60] *Interlaced Video & Deinterlacing for streaming*. 2021. URL: <https://blog.video.ibm.com/streaming-video-tips/interlaced-video-deinterlacing-for-streaming/>.
- [61] 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).
- [62] Expromo. *Led Display Facts*. 2020. URL: <https://www.expmrmo.eu/en/led-display-facts/>.
- [63] *LCD vs. LED: What is LCD & LED*. 2023. URL: <https://www.linsnled.com/difference-between-lcd-and-led.html>.
- [64] *Philips Test Card*. 2008. URL: [https://commons.wikimedia.org/wiki/File:Philips\\_PM5544.svg](https://commons.wikimedia.org/wiki/File:Philips_PM5544.svg).
- [65] Logical Increments. *Information About Screen Resolution*. 2023. URL: <https://www.logicalincrements.com/resolution>.
- [66] Genelec. *Audio Test Signals*. 2024. URL: <https://www.genelec.com/audio-test-signals>.

## Bibliography

List of References

D. T. McGuiness, Ph.D

- [67] 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.
- [68] Nobuyuki Otsu et al. "A threshold selection method from gray-level histograms". In: *Automatica* 11.285-296 (1975), pp. 23–27.
- [69] Antoni Buades, Bartomeu Coll, and J-M Morel. "The staircasing effect in neighborhood filters and its solution". In: *IEEE transactions on Image Processing* 15.6 (2006), pp. 1499–1505.