Introduction to Computer Vision - P2
Lesson 1 - Introduction to image processing

Alasdair Newson

alasdair.newson@telecom-paristech.fr

January 11, 2021

Summary

Introduction

- The goal of this short course is to introduce you to some of the main techniques and methods in **computer vision**
- There will be three parts :
  1. Introduction to image processing
  2. Introduction to computer vision
  3. Advanced methods in computer vision (patch-based methods and deep learning)
- This intensive course will take pace over two weeks :
  - 3 lessons
  - 2 TPs
- The course will be evaluated on the lab work (TP). You will have a week to complete the TPs.

Organisation

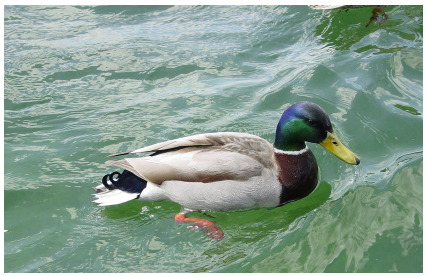
- 11/01/2020 : Introduction to image processing
- 13/01/2020 : TP 1 : image processing
- 15/01/2020 : Introduction to computer vision
- 19/01/2020 : TP 2 : computer vision
- 21/01/2020 : Advanced topics in computer vision

Introduction

- The difference between image processing and computer vision is a grey area
- Image processing corresponds to more low-level operations
  - Filtering
  - Colour and grey-level manipulation
  - Restoration
- Computer vision corresponds to more high-level operations
  - Segmentation
  - Edge/corner detection, feature detection
  - Motion estimation
  - Background detection

Introduction

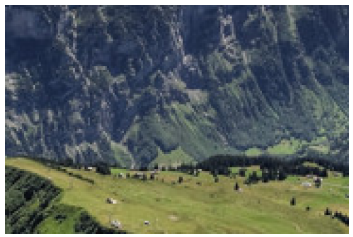
- Today, introduction to digital imaging :
  - Digital image acquisition chain
  - Modelling of this process
- We are **not** going to discuss these other types of images:
  - Satellite/radar images
  - Medical images
  - 3D images / point clouds
- Here, we are going to discuss digital, photographic images

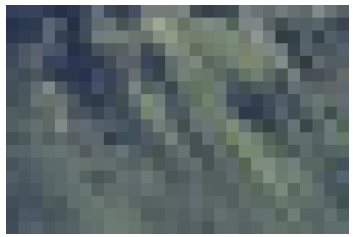



Introduction - image model

- What is a digital image ? A collection of **pixels**
  - A function I defined on a discrete 2D grid $\Omega = \{0, \ldots, m-1\} \times \{0, \ldots, n-1\}$
  - Either a positive real value (grey-level), or a vector of size 3 (colour)



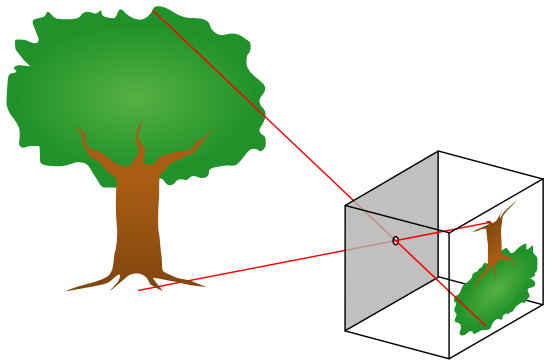




- Main question : how are the pixels defined ? How is the digital image created ?

Introduction - image model

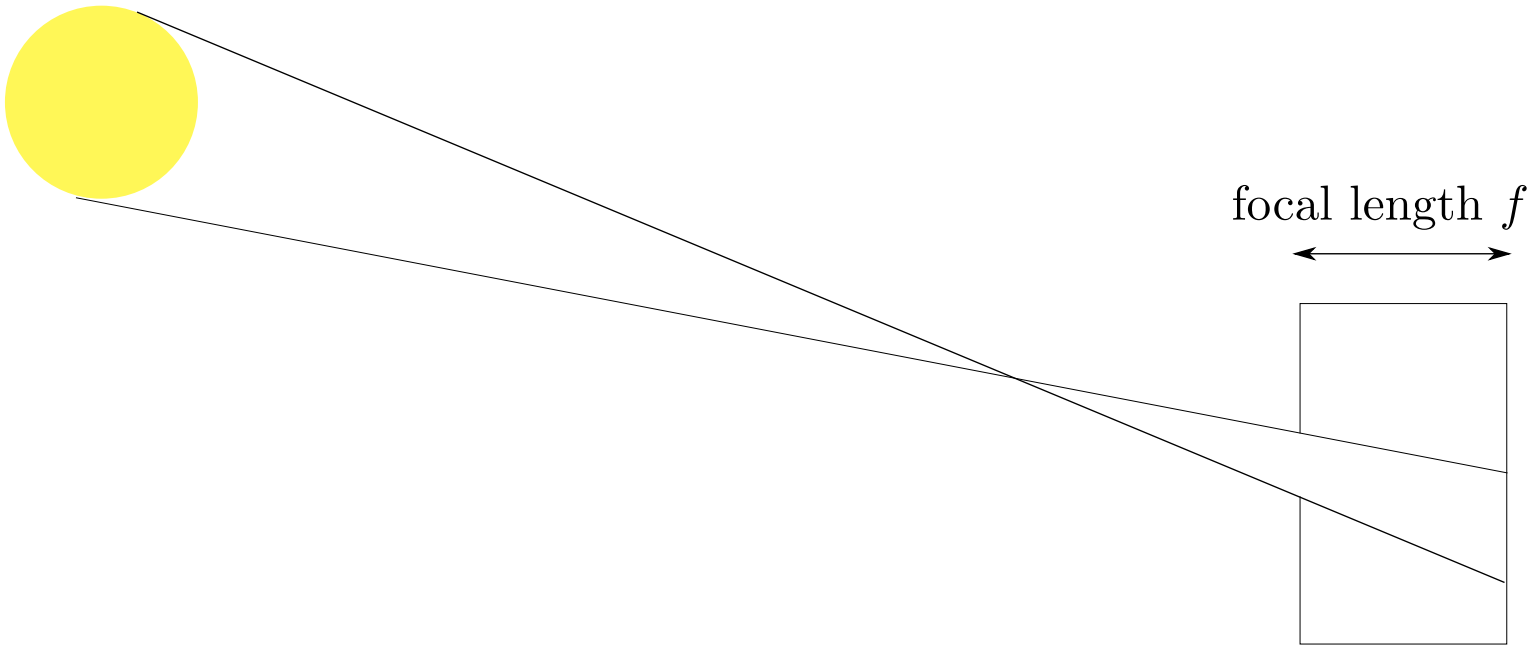
- A camera captures light photons from the 3D natural environment
- Cameras work as **pin-hole cameras** : ideally an infinitely small **aperture**
  - Light travels through a small aperture which serves to create an image on a receptor field
  - Light is focussed using a lens
  - Sensors convert photons to an electrical signal



Pinhole camera

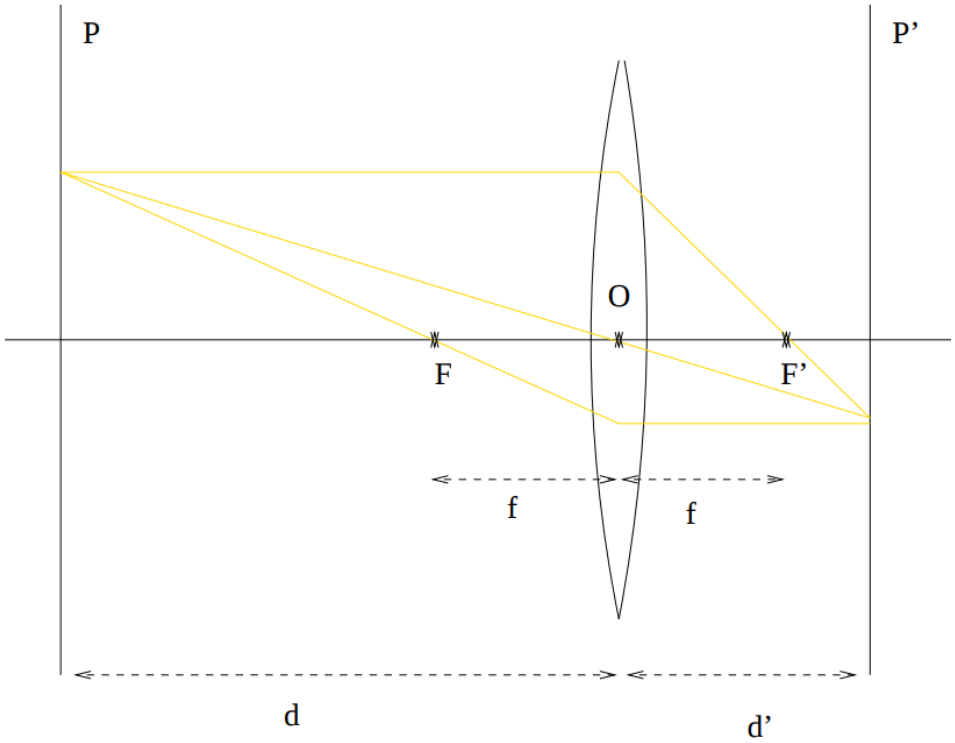
Introduction - image model

- Distance between aperture and focal plane : **focal distance**
- Because the aperture is not infinitesimal, the result is blurred
  - $I = s * g$, where s is the true scene, g the blurring kernel
- In fact, there is a also discretisation process between $s * g$ and I



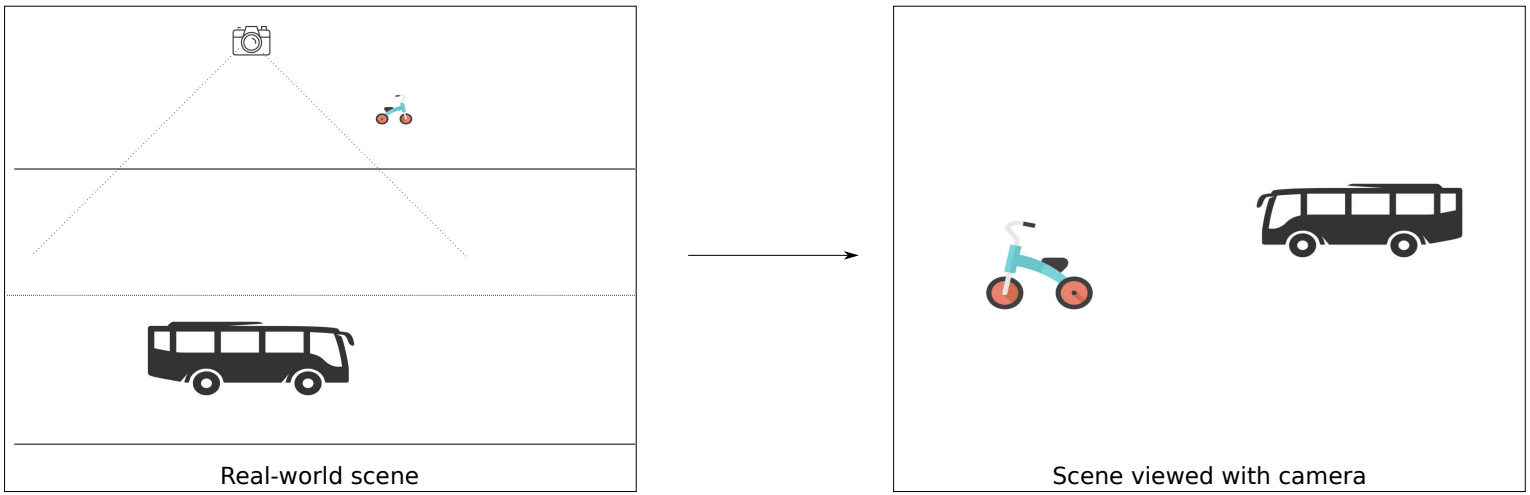
Introduction - image model

- Solution to the blurring problem : **lens** to focus the light
- The result is still the convolution of s with the impulse response of the optical system



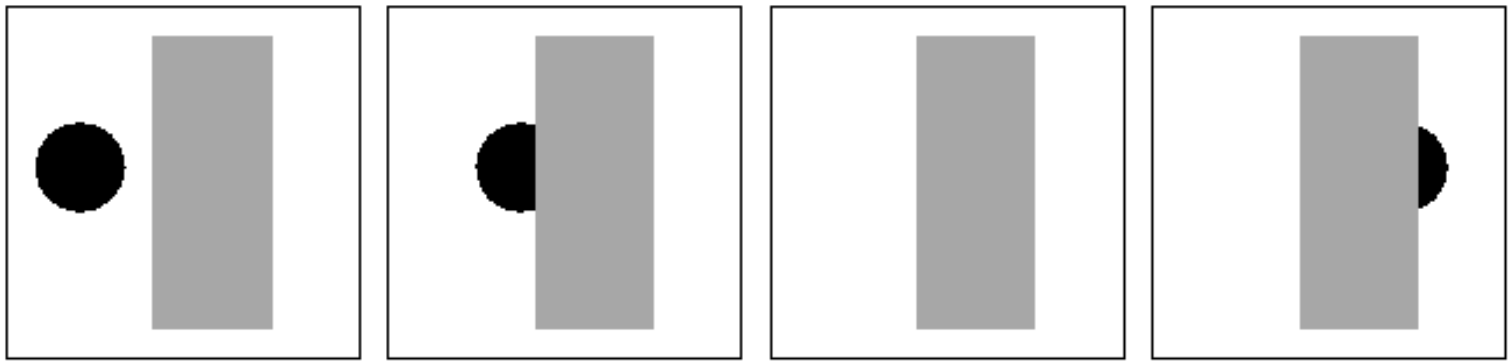
Introduction - image model

- The projection of a 3D scene onto a 2D representation has two main consequences :
  - 1 : Loss of depth information, and therefore the absolute size of objects



Introduction - image model

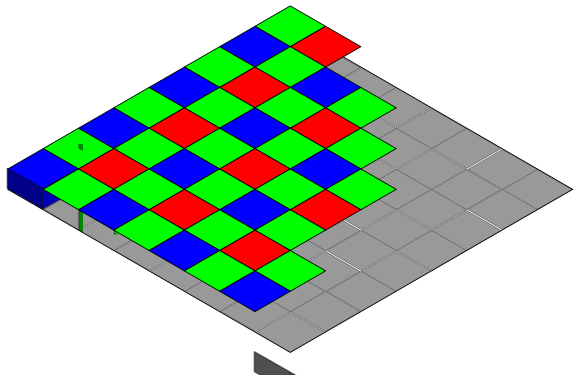
- The projection of a 3D scene onto a 2D representation has some unavoidable consequences :
  - 2 : Occlusion. Objects hiding other objects



- Conclusion : necessarily loss of information from 3D to 2D
  - Can be partially addressed using depth estimation and inpainting
- We now look at how the image sensor works (with colour)

Introduction - image model

- A digital camera's receptor field is an array of **digital sensors**
- Each of these only detects a certain wavelength
- Light is a continuous spectrum : a sensor cannot detect
  - Light is filtered to let only a certain wavelength, with a **filter pattern**



Bayer filter pattern

- For this reason, images are often represented as $m \times n \times 3$ tensors in the **Red-Green-Blue** (RGB) space
- An image is the concatenation of three images, each corresponding to a colour

Introduction - image model

- Example of an RGB image

Input

Introduction - image model

- Example of an RGB image

Red channel

Introduction - image model

- Example of an RGB image

Green channel

Introduction - image model

- Example of an RGB image

Blue channel

Summary

Sampling theory

- If you have studied signal processing, then you will know that **sampling theory** is of key importance
- This is the same for image processing. Some important concepts :
  - Fourier transform (FT)
  - Poisson summation formula
  - Shannon's sampling theorem

Sampling theory - Fourier transform

- The Fourier Transform (FT) decomposes a function (a signal) using complex exponentials (sine waves)
  - This can be seen as a change of basis
- Any (periodic) function, however complicated, can be decomposed as a sum of sine waves
- This representation of a signal can prove much easier to manipulate, depending on our goal (filtering, analysis)

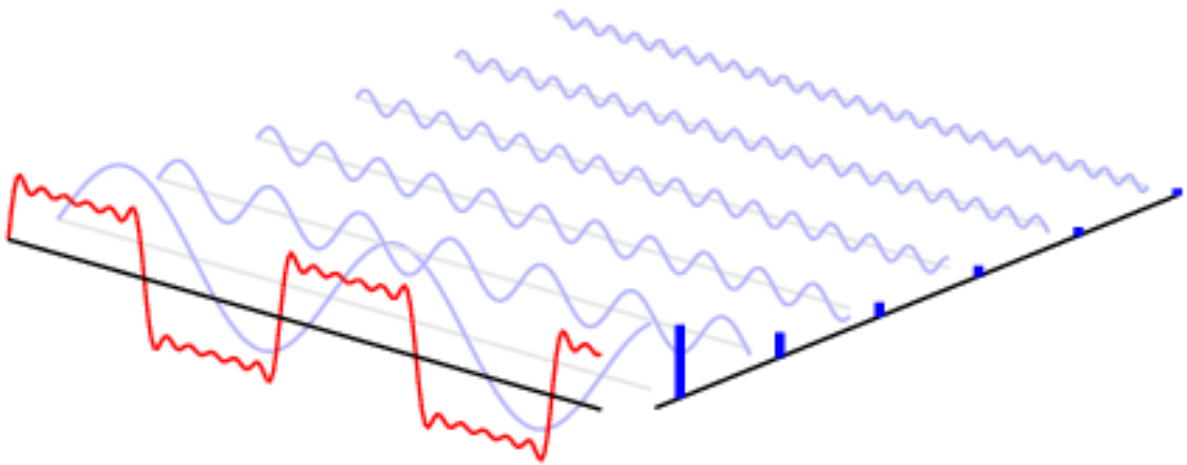


Image from http://pgfplots.net/tikz/examples/fourier-transform/

Sampling theory - Fourier transform

- The Fourier transform changes the representation of a function from the time (or position) domain to the frequency domain
- Takes as input a function, and outputs another function

Continuous Fourier transform - 1D

Let $f : \mathbb{R} \to \mathbb{C}$ be a continuous, integrable function. The Fourier transform of f is defined as

$$\hat{f}(\nu) = \int_{\mathbb{R}} f(t) e^{-2\pi i \nu t} \, dt$$

- $\nu \in \mathbb{R}$ is a **frequency**
- $e^{-2\pi i \nu t}$: cosine and sine waves, via Euler's formula: $\exp ix = \cos x + i \sin x$

- **NOTE** : We will also denote the Fourier Transform (operator) with $\mathcal{F}$, depending on which notation is easier

Sampling theory - Fourier transform

- The FT is **invertible** : no information is lost
- The **inverse FT** is defined in a similar manner

Continuous Inverse Fourier Transform - 1D

$$f(t) = \int_{\mathbb{R}} \hat{f}(\nu)\, e^{2\pi i \nu t}\, d\nu$$

- t is in the time/spatial domain

Sampling theory - Fourier transform

- A discrete version of the FT also exists, **Discrete-Time Fourier transform (DTFT)**
- However, in this case, the maximum representable frequency is limited
  - For any ν, $e^{2\pi i(\nu+n)} = e^{2\pi i\nu t}$
  - Therefore, $\nu \in [-\frac{1}{2}, \frac{1}{2}[$

Discrete-Time Fourier transform - 1D

Let $u : \mathbb{Z} \to \mathbb{C}$ be a summable sequence ($\sum_n |u_n| < +\infty$). For $\nu \in [-\frac{1}{2}, \frac{1}{2}[$, the Discrete-Time Fourier transform of u is defined as

$$\hat{u}(\nu) = \sum_{\mathbf{n}\in\mathbb{Z}} u_n \, e^{-2\pi i\nu n} \tag{1}$$

- Careful : $\nu \in [-\frac{1}{2}, \frac{1}{2}[$ is a **continuous frequency**

Sampling theory - Fourier transform

- Inverse Discrete-time Fourier transform also exists

Inverse Discrete-Time Fourier transform - 1D

$$u_n = \int_{-\frac{1}{2}}^{\frac{1}{2}} \hat{u}(\nu)\, e^{2\pi i \nu n}\, d\nu \tag{2}$$

- Careful : there is an integral and not a sum, because ν is continuous

Sampling theory - Fourier transform

- Finally, third type of Fourier transform : **Discrete Fourier Transform**
  - FT for discrete, finite sequences (vectors in a computer)
- **Only FT** that can be **numerically calculated** for digital signals

Discrete Fourier transform (DFT)

Let $u : \{0, \dots, N-1\} \to \mathbb{C}$ be a finite sequence of length N. For $\nu \in \{0, \dots, N-1\}$, the Fourier transform of f is defined as

$$\hat{u}_\nu = \sum_{\mathbf{n=0}}^{\mathbf{N-1}} u_n \, e^{-2\pi i \nu \frac{n}{N}}$$

- The inverse of this also exists

$$u_n = \frac{1}{N} \sum_{\nu=\mathbf{0}}^{\mathbf{N-1}} \hat{u}_\nu \, e^{2\pi i \frac{\nu}{N} n}$$

Sampling theory - Fourier transform

- The 2D Discrete Fourier Transform is defined in a similar manner

2D Discrete Fourier transform

Let $I : \{0, \dots, M-1\} \times \{0, \dots, N-1\} \to \mathbb{R}$ be a finite, 2D **image** ($I_{x,y} \in \mathbb{R}$) of size (M, N). For $(\nu_x, \nu_y) \in \{0, \dots, M-1\} \times \{0, \dots, N-1\}$, the 2D DFT of I is defined as

$$\hat{I}_{\nu_x,\nu_y} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} I_{m,n}\, e^{-2\pi i \left(\frac{\nu_x m}{M} + \frac{\nu_y n}{N}\right)}$$

- The inverse of this also exists

$$I_{x,y} = \frac{1}{MN} \sum_{\nu_x=0}^{M-1} \sum_{\nu_y=0}^{N-1} \hat{I}_{\nu_x,\nu_y}\, e^{2\pi i \left(\frac{\nu_x m}{M} + \frac{\nu_y n}{N}\right)}$$

Sampling theory - Fourier transform

1. Frequency shift : $\mathcal{F}(e^{2\pi i\nu_0} f)(\nu) = \mathcal{F}(f)(\nu - \nu_0)$
2. Time shift : $\mathcal{F}(f - \tau)(\nu) = e^{2\pi i\nu\tau}\mathcal{F}(f)(\nu)$
3. If f is real, then $\hat{f}$ has hermitian symmetry :
   - $(f(x) \in \mathbb{R}) \rightarrow \left(\hat{f}(-\nu) = \overline{\hat{f}(\nu)}\right)$
4. If f is symmetric, then $\hat{f}$ is also symmetric :
   - $(f(x) \in \mathbb{R}) \rightarrow \left(\hat{f}(-\nu) = \hat{f}(\nu)\right)$
5. If f is real and symmetric, then $\hat{f}$ is also real and symmetric
   - This in fact happens quite a lot : "top-hat" filter, Gaussian filter. Simplifies understanding of filter in Fourier domain
6. Parseval's theorem : $\|f\|^2 = \|\hat{f}\|^2$
   - Energy of function is the same in the Fourier domain
7. Differentiation : $\mathcal{F}(f^{(n)})(\nu) = (2\pi i\nu)^n \hat{f}(\nu)$
   - Useful for solving differential equations

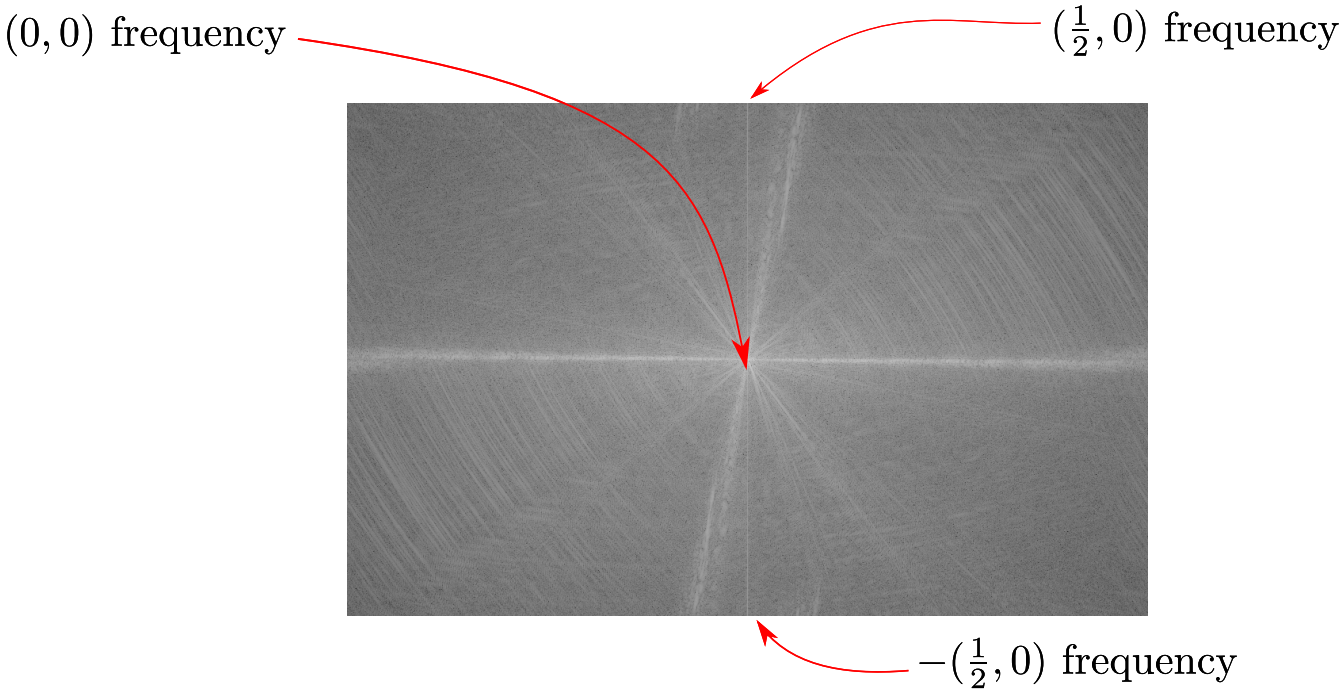
Sampling theory - Fourier transform

- The link between the DFT and DTFT is important to understand
- Indeed, in the case of images, we consider the sampling frequency (space between pixels) to be 1
- This is exactly the case of the DTFT. However, we work with discrete signals of compact support : signals do not extend to infinity
- If the signal is of compact support $\{0, \dots N-1\}$, then the DFT is the same as the DTFT, for a given frequency :

$$\begin{aligned}
\mathrm{DFT}(f)(k) &= \sum_{n=0}^{N-1} f_n e^{-2\pi i k \frac{n}{N}} \\
&= \sum_{n \in \mathbb{Z}} f_n e^{-2\pi i (\frac{k}{N}) n} \\
&= DTFT(f)(\frac{k}{N}), \quad \forall k \in \{0, \dots, N-1\}
\end{aligned}$$

Sampling theory - Fourier transform

- Therefore, we know that, for a discrete signal, the frequencies of the DTFT are in the interval $[\ -\frac{1}{2}, \frac{1}{2}[$
- Just with a different indexing (the way in which this happens depends on the function used)



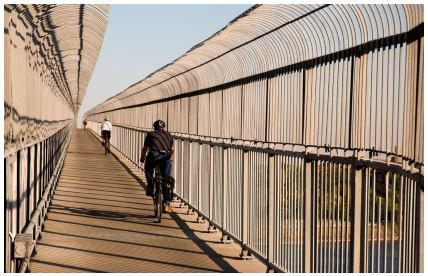
Fourier transform summary

Name	Time/spatial domain	Fourier domain
Continuous Fourier transform (CFT)	$t \in \mathbb{R}$	$\nu \in \mathbb{R}$
Discrete-time Fourier transform (DtFT)	$n \in \mathbb{Z}$	$\nu \in [-\frac{1}{2}, \frac{1}{2}[$
Discrete Fourier transform (DFT)	$n \in \{0, \ldots, N-1\}$	$\nu \in \{0, \ldots, N-1\}$

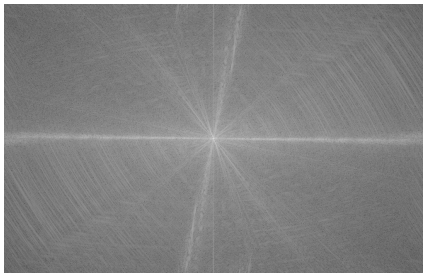
- The frequency representation $\hat{f}$ is also known as the **spectrum** of f

Sampling theory - Fourier transform

- So, what does the spectrum actually look like ?

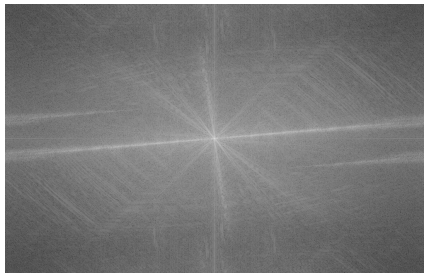


Input image



Spectrum (magnitude)

Rotated input image



Spectrum (magnitude) of rotated image

- You can see the frequencies of the strong edges. These are moved in the spectrum when the image is rotated
- Note, these spectra have been modified for easier viewing. We have used $\log(|\hat{I}| + 1)$
  - This is done since the very low frequencies are often of much greater amplitude than the higher ones

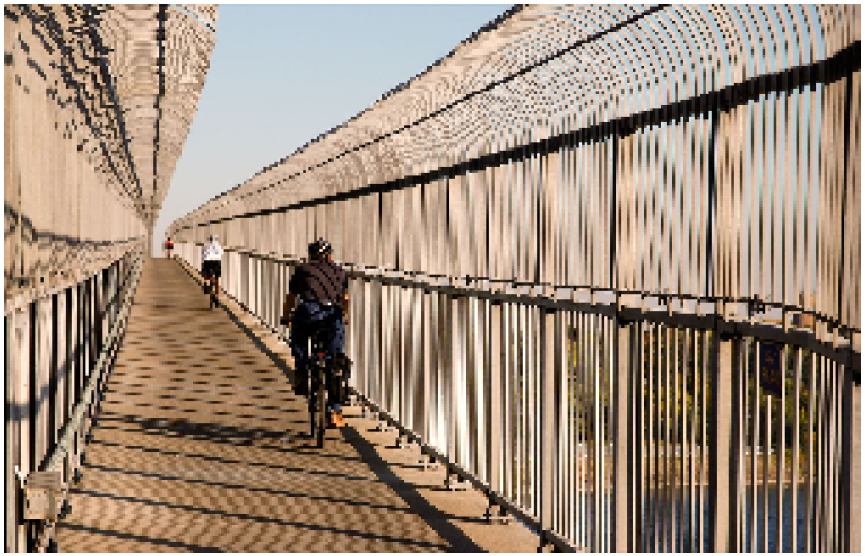
Sampling theory - Fourier transform

- Some final remarks on interpreting the spectrum
- The spectrum can be broken down into **amplitude** and **phase**
  - $\forall \nu \in \mathbb{C}, \nu = \rho e^{i\theta}$, with $\rho, \theta \in \mathbb{R}$
- Magnitude often easier to interpret : strength of sinusoidal "features"
- The phase produces a shift in the location of these features
  - $e^{i(x+\theta)} = \cos(x+\theta) + i\sin(x+\theta)$
- Most often, we visualise $|\log(\hat{I}+1)|$, since low frequencies are usually very large in magnitude
- DFTs are represented with the 0 position (in time or frequency) at the beginning of the vector/matrix. For visualisation purposes it is easier to put the 0 at the middle. For this, there is a function (in FT libraries) called **fftshift**

Sampling theory - Poisson summation formula

- We now come to the main question of **sampling**
  - Happens, for example, when changing resolution of an image
- Incorrect sampling leads to **aliasing** (repliement spectral in french)

Original image



Subsampled image

- How do we correct this ? The Poisson summation formula is the key !

Sampling theory - Poisson summation formula

- The **Poisson summation formula** links continuous FT to the DtFT
- Allows us to establish conditions under which signals are **well-sampled**
- Well-sampled : the **sampling rate** is high enough to capture all the frequencies of the continous signal

Poisson summation formula

Let $f : \mathbb{R} \to \mathbb{C}$ be a continuous, integrable function, and $\hat{f}$ its CFT. Then, we have, for $\nu \in [-\frac{1}{2}, \frac{1}{2}[$

$$\overbrace{\sum_{n\in\mathbb{Z}} \hat{f}(\nu+n)}^{Sum\ of\ frequencies\ with\ step\ 1} = \underbrace{\sum_{m\in\mathbb{Z}} f(m)e^{-2\pi i m\nu}}_{DtFT}$$

- Consequence : all frequencies $\nu + n$ influence the frequency ν. This is known as **aliasing**

Sampling theory - sampling theorem

$$\sum_{n\in\mathbb{Z}} \hat{f}(\nu+n) = \sum_{m\in\mathbb{Z}} f(m)e^{-2\pi i m\nu}$$

- Suppose that $\text{Supp}(\hat{f}) \in [-\frac{1}{2}, \frac{1}{2}[$. Then we have :

$$\sum_{n\in\mathbb{Z}} \hat{f}(\nu+n) = \hat{f}(\nu) = \underbrace{\sum_{m\in\mathbb{Z}} f(m)e^{-2\pi i m\nu}}_{DtFT\ at\ \nu}$$

- This means that the DtFT at frequency ν represents exactly the continous FT at ν
- Conclusion : if the spectrum is zero outside $[-\frac{1}{2}, \frac{1}{2}[$, then the signal is well-sampled

Sampling theory - sampling theorem

- This leads to the Nyquist-Shannon sampling theorem

Shannon's sampling theorem

Let $f : \mathbb{R} \to \mathbb{C}$ be a continuous, integrable function, and $\hat{f}$ its CFT. If f contains no frequencies greater than ν_{max}, then the signal is well-sampled with sampling with a sampling step $\delta < \frac{1}{2\nu_{max}}$. This means that no aliasing has occured.

- In the case of images, where the sampling step δ is normalised to 1 (often the case with images) then the original signal should have contained no frequencies higher or equal to $\frac{1}{2}$ (as previously seen).

Sampling theory - sampling theorem

- There are two main situations where we wish to pay attention
  1. When taking a photograph
  2. When downsampling an image (going to a lower resolution)
- We do not go into the first situation here (there is nothing you can do about it anyway : you would need to change the camera)

- Consider a finite, discrete signal defined on $\{0, \ldots, N-1\}$. Then, if we downsample the signal by a factor s, we need to remove the frequencies above or equal to $\frac{1}{2s}$
- How can this be done ? By **filtering the signal**
- For this, we need to look at **convolution**

Sampling theory - convolution

- Why are convolutions so important ? Given an input signal, the output signal of every linear, time-invariant system (basically a filter), can be written as the **convolution of the impulse response of the system, and the input signal**

Convolution operator

Let f and g be two integrable functions. The **convolution operator** $*$ takes as its input two such functions, and outputs another function $f * g$, which is defined at any point $t \in \mathbb{R}$ as :

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

- Intuitively, the output function is defined at any point t as the inner product between f and a *flipped/reversed* version of g, shifted by t

2D Convolution

- In this lesson, we are working with discrete **images**
- Therefore, we require a discrete 2D convolution operator. This is defined in a very similar manner to 1D convolution :

2D convolution operator

$$(f * g)_{x,y} = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f_{i,j} g_{x-i,y-j}$$

- Convolutions are one of the fundamental operations of signal processing

Sampling theory - convolution theorem

- Back to the (extremely important) **convolution theorem**

Convolution theorem

Let f and g be two integrable functions, and $\hat{f}$ and $\hat{g}$ their Fourier transforms. Define $h = f * g$. We have :

$$\hat{h}(\nu) = \hat{f}(\nu).\hat{g}(\nu) \tag{3}$$

Due to the invertibility of the FT, we therefore have

$$h(t) = \mathsf{IFT}(\hat{f}.\hat{g})(t) \tag{4}$$

Sampling theory - convolution theorem

- The previous theorem is extremely important for **two main reasons** :
  1. If we know the frequency response of a filter (its spectrum), we can easily understand its action on an image via convolution : the frequencies are simply multiplied in the Fourier domain
  2. The Fourier transform can be carried out in a fast manner via the **Fast Fourier Transform**
- Now, let us get back to our original goal : low-pass filtering of an image to avoid aliasing

Sampling theory - filtering

- The most obvious approach to removing high frequencies is simply to set them to 0
  - This corresponds to multiplying the image spectrum by a "top-hat" function in the Fourier domain (an "ideal" low-pass filter)
- Unfortunately, this introduces **ringing artefacts**

Input image

Ideal low-pass filtered

Sampling theory - filtering

- Let h_{ν_0} be an ideal low-pass filter at frequency ν_0. This is easily defined in the Fourier domain as H_{ν_0} :

$$H_{\nu_0}(\nu) = \begin{cases} 1 & \text{if } \nu < \nu_0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

- Note, the equivalent of this in 2D is $\|(\nu_x, \nu_y)\|_2 < r$
- The inverse Fourier transform of H_{ν_0} turns out to be a "sinc" function
  - We show this now in the continuous case (a similar reasoning happens in the discrete case)
- The convolution of this sinc with the original signal leads to the ringing artifacts

Sampling theory - filtering

$$
\begin{aligned}
h_{\nu_0}(t) &= \int_{-\infty}^{\infty} H_{\nu_0}(\nu) e^{2\pi i \nu t} d\nu \\
&= \int_{-\nu_0}^{\nu_0} e^{2\pi i \nu t} d\nu \\
&= \frac{1}{2\pi i t} \left[e^{2\pi i \nu t} \right]_{-\nu 0}^{\nu_0} \\
&= \frac{1}{2\pi i t} \left(\cos(2\pi\nu_0 t) + i \sin(2\pi\nu_0 t) - \cos(-2\pi\nu_0 t) - i \sin(-2\pi\nu_0 t) \right) \\
&= \frac{1}{2\pi i t} (2i \sin(2\pi\nu_0 t)) \\
&= \frac{\sin(2\pi\nu_0 t)}{\pi t} = 2\nu_0 \mathsf{sinc}(2\nu_0 t)
\end{aligned}
$$

- Where $\mathsf{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$ (the "normalised" sinc function)

Sampling theory - filtering

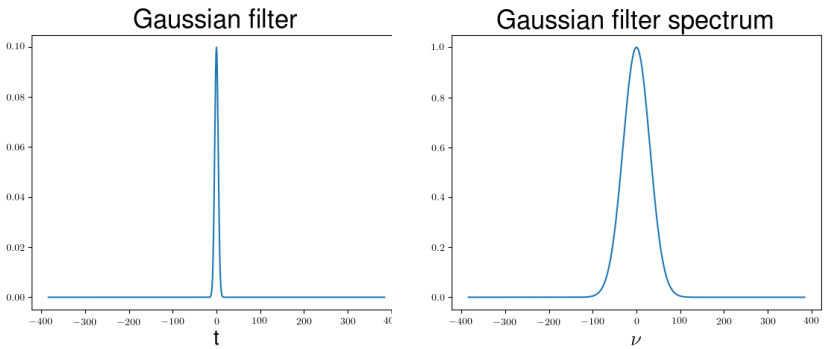
- Let g_σ be a Gaussian filter of standard deviation σ
- As mentioned, this alternative does not produce ringing artefacts

$$g_\sigma(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{t^2}{2\sigma^2}} \tag{6}$$

- Why ? The FT of a Gaussian is also a Gaussian function

$$\mathcal{F}(e^{-\alpha x^2}) = \sqrt{\frac{\pi}{\alpha}} e^{-\frac{\pi\nu^2}{\alpha}} \tag{7}$$

- The variance is inverted in the Fourier domain



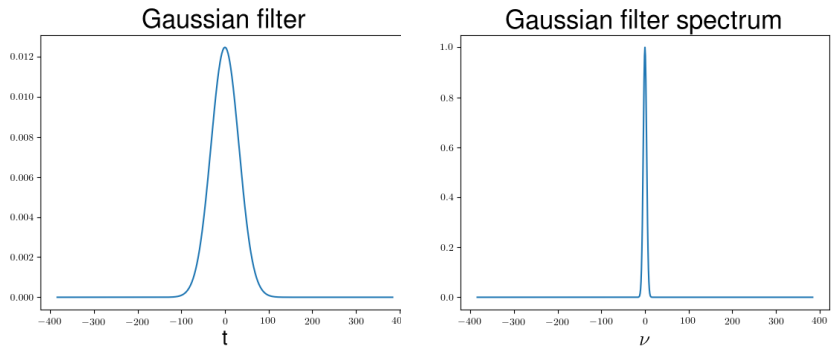
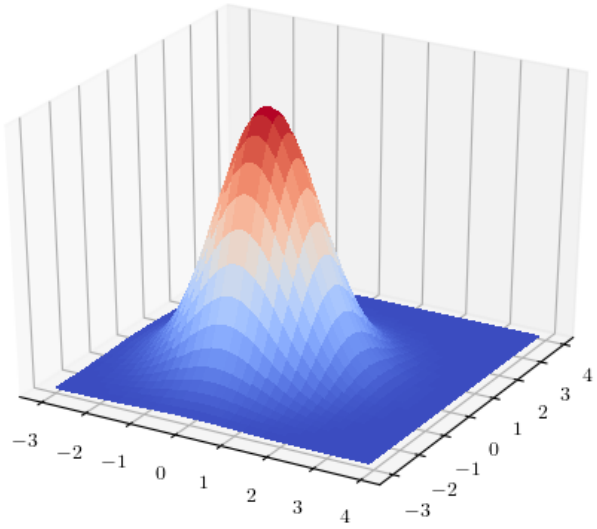


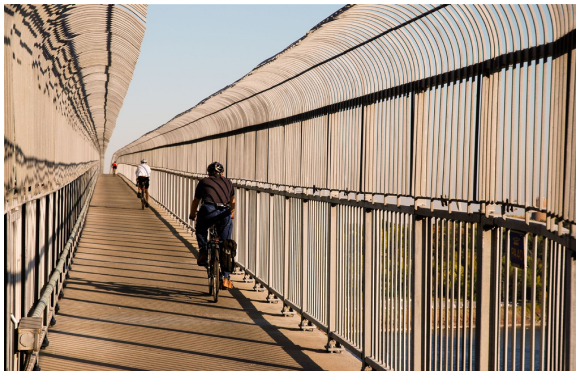
Image filtering

- Using the convolution theorem, we know that convolution is equivalent to an element-wise multiplication in the frequency domain

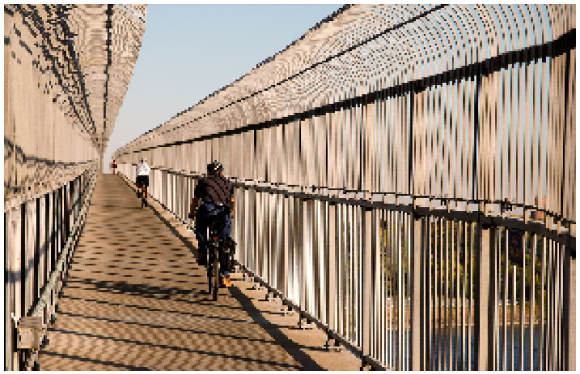


- Therefore, the Gaussian filter acts as a **low-pass filter**
- To remove high frequencies : convolve the image with a Gaussian filter

Sampling theory - filtering



Original image



Subsampled image

Filtered, subsampled image

- The visual aliasing artefacts have been removed
- Note : completely removing aliasing would require an ideal low-pass filter (a gate function). This is not possible in reality with a finite signal; the support of the inverse FT of the gate function (a sinc) is infinite

Filtering

- Filtering can be used for many other tasks
  - Edge detection
  - Image sharpening
  - Denoising
- We now look at some specific, widely-used filters in image processing

Summary

Image filtering

- Many types of filters exist
  - Gradient : $\nabla I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$
  - Laplacian : $\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$
  - Gaussian blur (already seen)
  - Sharpening filter

Image filtering

- One of the simplest filters : gradient (derivative in x and y)
- Question : how do we approximate derivatives ?
  - Taylor series expansion
  - We will do this in the 1D case, but the 2D case is the same

$$I(x+\Delta x) = I(x) + \Delta x I'(x) + \mathcal{O}(\Delta x)$$
$$I'(x) \approx \frac{1}{\Delta x}\left(I(x+\Delta x) - I(x)\right)$$

- In general, we consider the spatial step $\Delta x = 1$. Therefore, we have :
  - $\nabla_x I$ filter $[-1, 1]$
  - $\nabla_y I$ filter $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Image filtering

- The gradient magnitude of an image is useful for **detecting edges**

Input

Gradient magnitude : $\|\nabla I\|_2$

Image filtering

- Laplacian filter is also very useful. The second derivatives are determined in a similar manner to the gradient
- We will use the notation $I_{xx} = \frac{\partial^2 I}{\partial x^2}$ and $I_{yy} = \frac{\partial^2 I}{\partial y^2}$

$$I(x - \Delta x, y) = I(x, y) - \Delta x I_x(x, y) + \frac{1}{2}\Delta x^2 I_{xx}(x, y) + \mathcal{O}(\Delta x^2)$$
$$I(x + \Delta x, y) = I(x, y) + \Delta x I_x(x, y) + \frac{1}{2}\Delta x^2 I_{xx}(x, y) + \mathcal{O}(\Delta x^2)$$
$$I(x, y - \Delta y) = I(x, y) - \Delta y I_y(x, y) + \frac{1}{2}\Delta y^2 I_{yy}(x, y) + \mathcal{O}(\Delta y^2)$$
$$I(x, y + \Delta y) = I(x, y) + \Delta y I_y(x, y) + \frac{1}{2}\Delta y^2 I_{yy}(x, y) + \mathcal{O}(\Delta y^2)$$

Image filtering

- Using the previous equations, and considering $\Delta x = \Delta y = 1$, we have :

$$\begin{aligned} \nabla^2 I &= I_{xx}(x,y) + I_{yy}(x,y) \\ &\approx I(x-\Delta x, y) + I(x+\Delta x, y) + I(x, y-\Delta y) + I(x, y+\Delta y) \\ &\quad - 4I(x,y) \end{aligned} \tag{8}$$

- $\nabla^2 I$ filter $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
  - The pattern of the filter is also sometimes known as a *stencil*
- Will be useful further on in the course

Image filtering

- Example of Laplacian filtering

Input

Laplacian magnitude : $\|\nabla^2 I\|_2$

Image filtering

- The previous filters are **high-pass filters**
- A good rule of thumb : if the sum of the elements is 0, then the filter is some sort of high-pass filter
- Let ϕ be such a filter : $\sum_n \phi(n) = 0$
- The 0 frequency of the Fourier Transform of ϕ is equal to the sum of the filter elements:

$$\begin{aligned}\hat{\phi}(0) =& \sum_n \phi(n) e^{-2i\pi 0 n} \\ =& \sum_n \phi(n) \\ =& 1 + 1 + 1 + 1 - 4 = 0.\end{aligned}$$

Image filtering

- The result of such a filter on a constant image will be 0 everywhere
- Indeed, suppose $I(x, y) = c$, with c a constant
- With the convolution theorem, we can see that the result of this filtering will be 0 for the 0-frequency component :

$$\widehat{\left(I * \phi\right)}(0) = \hat{I}(0) \,.\, \hat{\phi}(0) = 0.$$

- If I is a constant image, then the frequency content of $I * \phi$ is 0 everywhere
  - In other words, the filter will not respond at all in a constant image

Image filtering

- Another well-known edge-detection filter is the **Sobel filter**
- Similar to the gradient filters, different approximations

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{9}$$

Input image

Sobel filtered, x

Sobel filtered, y

Image filtering

- Sobel filters are an example of **separable filters**
- Separable filters can be decomposed into two 1D filters applied successively, in each dimension !

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [-1, 0, +1]$$

- This greatly reduces the cost of applying the filters

Image filtering

- Another common task : smoothing/low-pass filtering, which we have seen for the purposes of anti-aliasing
- We wish to remove high frequencies from an image
  - Denoising, scale-space (we will see this next lesson)
- Most common smoothing filter : Gaussian filter, as we have seen

Input image

Gaussian filtered image (σ=3.0)

Image filtering

- **Sharpening** is an operation to **enhance edges in images**
- This can be done using a combination of high-pass and low-pass filters
- Common technique : "**unsharp masking**"
  - Name comes from the blurred (unsharp) mask used in original photographic process

Unsharp masking

$$\tilde{I} = I + \alpha(I - I * \phi) \tag{10}$$

- $\alpha \in \mathbb{R}^+$ is the amount of blurring
- ϕ is a blurring filter

Image filtering

- Example of unsharp filtering

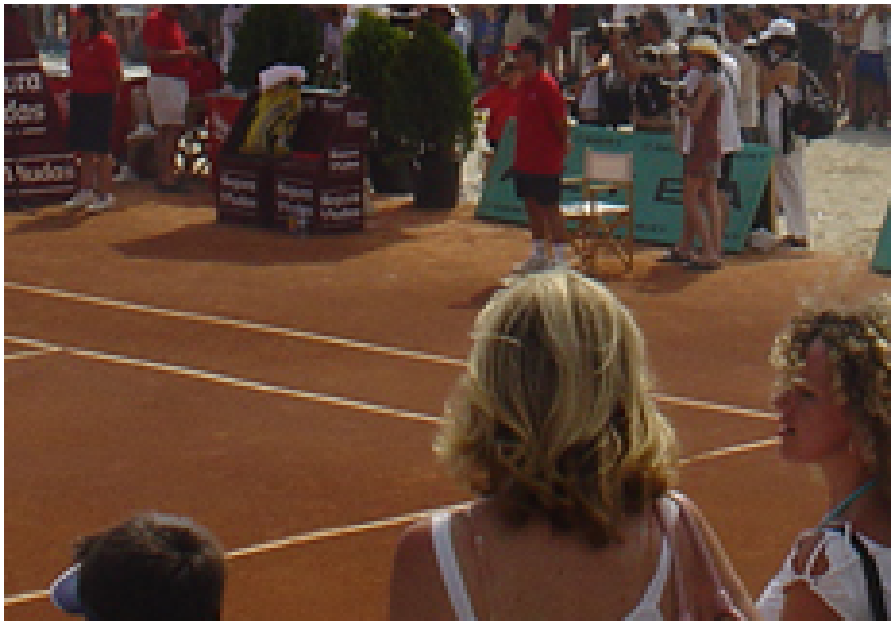
$\alpha = 1$

$\alpha = 3$

$\alpha = 5$

Image filtering

- Example of unsharp filtering



Input

Unsharp filtering

Image filtering

- A final type of filter often used is the **median filter**
- We simply take the median value of a certain region of size r around the considered pixel (x, y)

$$\tilde{I}_{x,y} = \text{Median}\left(\{I_{x+\ell,y+q} \mid \|(\ell, q)\|_\infty < r\}\right) \tag{11}$$

- Useful for removing outliers
  - In particular for certain types of noise, as we will see

Summary of image filtering

- Many basic image processing operations can be carried out with filters (convolutions) :
  - Smoothing/low-pass filtering
  - Edge detection/high-pass filtering
  - Image sharpening
- In order to see what the filter does : take the Fourier Transform and inspect the spectrum
  - It is difficult to understand the action of filters in the spatial domain
  - The spectrum can tell you if the filter is a high-pass, low-pass or band-pass (less common in image processing)
- Good rule of thumb : if the sum of the filter elements is 0, then you probably have a high-pass/edge detection filter

Summary

Image restoration

- Images undergo a variety of degradations and noise sources
  - Shot noise, from discrete nature of light
  - Gaussian noise, from electronics in cameras
  - Motion blur, optical blur
  - Old films, interplanetary dust (astronomical images)

Image restoration

Motion blur. Source, Wikimedia commons

Optical blur. Source, flickr

Image restoration

- Various types of “noise” exist
  1. Gaussian noise
  2. Impulse noise (salt-and-pepper noise)
  3. Missing pixels
  4. Poisson (or shot) noise (low-light conditions)
  5. Convolution degradation (blurring for example)

Gaussian noise

- Gaussian noise generally happens in the analog signal in the electronics of the camera
- Can be modelled as additive noise :

$$y = I + \sigma\eta \tag{12}$$

- Where $\eta \sim \mathcal{N}(0, 1)$, and σ is the standard deviation of the noise

Image restoration

Input

Image restoration

Gaussian noise

Impulse noise (salt-and-pepper noise)

- Impulse noise corresponds to random pixels which are either saturated (1) or turned off (0)
- Can happen in equipment with electronic spikes
- We can model this as

$$y = \begin{cases} I & \text{with probability p} \\ b & \text{with probability 1-p} \end{cases} \tag{13}$$

- Where $b \sim Ber(\frac{1}{2})$ (a Bernoulli variable of parameter $\frac{1}{2}$)

Image restoration

Impulse noise

Missing pixels

- Missing pixels can happen when certain pixels in a display are not functioning
- Can be modelled as

$$y = Ib \tag{14}$$

- Where $b \sim Ber(\frac{1}{2})$ (a Bernoulli variable of parameter p)

Image restoration

Missing pixels

Image restoration

- The most common type of noise, Gaussian noise, is often removed using a smoothing filter
- Gaussian filter !

Gaussian noise

Image restoration

- The most common type of noise, Gaussian noise, is often removed using a smoothing filter
- Gaussian filter !

Restored (Gaussian filter)

Image restoration

- If only a few pixels are likely to be damaged, then the **median filter** is a good option for the following types of noise
  - Impulse noise
  - Random missing pixels

Impulse noise

Image restoration

- If only a few pixels are likely to be damaged, then the **median filter** is a good option for the following types of noise
  - Impulse noise
  - Random missing pixels

Restored (median filter)

Image restoration

- A lot of the time, the degradation process can be modelled with the following generic model

Degradation process

$$y = AI + \eta, \tag{15}$$

- A is a degradation operator, and η is some additive noise
  - A can be a subsampling operator, an optical or motion blur
  - The noise is most often considered iid
- This is a standard least-squares problem
- If the degradation is considered to be in the form of a convolution ($A * I$), then this is known as **deconvolution**

Image restoration

- If we know the impulse response of the degradation, we can use the **inverse filter**
- Recall that the convolution theorem states that for any signals f and g

$$h = f * g \iff \hat{h} = \hat{f}.\hat{g},$$

- This means that we can find I with the following formula (if we ignore the noise term)

$$I = \text{IFT}\left(\frac{\hat{y}}{\hat{A}}\right)$$

- Where IFT is the inverse Fourier transform

Wiener filtering

- Suppose that η is a **stationary** noise : the statistics of η are shift-invariant
- Suppose that $\mathbb{E}[\eta] = 0$
- We wish to minimise the mean square error. In this case, the optimal restoration filter is given in the Fourier domain by

$$\hat{f} = \frac{\hat{A}^*}{\frac{\mathbb{E}[\eta]}{\mathbb{E}[y]} + |\hat{A}|^2}$$

- $\frac{\mathbb{E}[\eta]}{\mathbb{E}[y]}$ is the **signal-to-noise ratio**
- If the signal to noise ratio is low (very little or no noise), then this is simply an inverse filter

Image restoration

- Various types of noise and degradations exist :
  - Additive noise (Gaussian often)
  - Missing pixels, impulse noise
  - Convolutive degradations (eg blurring)
- Different tools exist, adapted for different siutations
  - Smoothing (Gaussian) filter for additive noise
  - Median filter for imulse/missing pixels
  - Deconvolution/inverse filtering
- Denoising and restoration are vast domains, this presents a very brief overview. We will see some further algorithms in the last lesson

Summary

Radiometry and colour

- We have seen how sampling takes place to create a digital image
- We also looked at filters This was mainly concerned with a frequency perspective of signals/images
- It is also important to understand how we percieve brightness and colour
- By **radiometry**, we mean the scalar pixel values, and in particular the **range and distribution** of these pixel values
- Why is this important :
  - in general, we want to avoid the brightness of pixels being squashed into a small part of the range of the image

Radiometry and colour

- Our visual system is very **robust to global contrast changes**

Original image

Image with modified contrast

Radiometry and colour

- We can define a contrast change h as a **non-decreasing function** applied to the grey-level values of the image
  - Does not change ordering of grey-level values

$$\forall (x, y) \in \mathbb{R}^2, \quad x > y \implies h(x) > h(y) \tag{16}$$

Radiometry and colour

- A **non-increasing function** does not result in a percieved contrast change (inverts colours)

Original image

Image with modified contrast (decreasing function)

Radiometry and colour

- The human visual system is quite robust to global contrast changes, but **sensitive to local contrast changes**
- The most common way of modelling and handling contrast changes is via the **image histogram**
  - **Probabilistic representation** of the image information
  - We do not care about spatial organistation of the pixels (structure etc)

Radiometry and colour

- Let $\{a_0, \ldots, a_{q-1}\}$ be the q discrete, quantised grey-levels of an image u of size (m, n) (in general $0, \ldots, 255$)
- We denote the spatial domain of the image as $\Omega = \{0, \ldots, m-1\} \times \{0, \ldots, n-1\}$.

Histogram and cumulative histogram

- The cumulative histogram H_I of the image I is a function defined as

$$H_I(\lambda) = \frac{1}{|\Omega|} |\{x \in \Omega \,;\, I(x) \leq \lambda\}|. \tag{17}$$

- The histogram h_I of the image I is a vector of size q, defined as

$$h_I(i) = \frac{1}{|\Omega|} |\{x \in \Omega; I(x) = a_i\}|, \tag{18}$$

Radiometry and colour

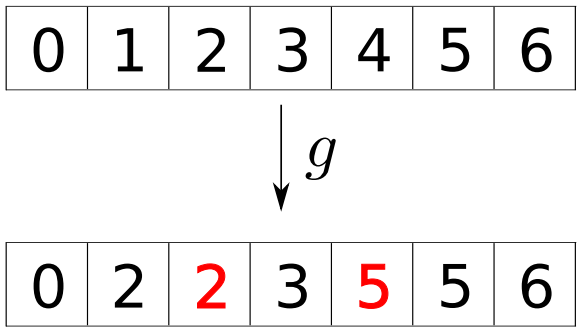
- If g is a **strictly increasing function**, g^{-1} exists, and we have

$$H_{g(I)}(\lambda) = H_I(g^{-1}(\lambda))$$

- Written another way :

$$H_{g(I)} = H_I \circ g^{-1} \tag{19}$$

- Note, if g is not strictly increasing, there can be a **loss of information**



Radiometry and colour

- Common task in image processing : changing image contrast
  - One approach : **histogram specification**
- Suppose we have a **target distribution** μ we want to impose on I

Histogram specification

- Let F_μ be the cumulative histogram of μ
- We wish to impose cumulative histogram equality
  - Find g such that : [†] $H_{g(I)} = F_\mu$. The correct g is :

$$g = F_\mu^{-1} \circ H_I. \tag{20}$$

- Indeed, we have

$$\begin{aligned} & g = F_\mu^{-1} \circ H_I \\ \iff \quad & F_\mu \circ g = H_I \\ \iff \quad & F_\mu = H_I \circ g^{-1} = H_{g(I)} \end{aligned}$$

see Equation (19)

[†] *Note, this is only one possible way of specifying a histogram, but the simplest one*

- Often, we wish to **equalise** the histogram, ie to impose a **uniform histogram**
  - Why ? To spread the energy of the image over all the grey-level range (not too bright and not too dark)
- For this, we recall the **probability integral transform result** from probability theory

Probability integral transform

- Let X be a continuous random variable with a cumulative distribution function F_X. Then the random variable $F_X(X)$ follows a **uniform distribution**

$$\begin{aligned} \mathbb{P}(F_X(X) \leq \lambda) &= \mathbb{P}(X \leq F_X^{-1}(\lambda)) \\ &= F_X(F_X^{-1}(\lambda)) = \lambda \end{aligned} \tag{21}$$

Radiometry and colour

- Conclusion : we can impose a uniform distribution on the pixel grey-levels simply by applying the cumulative distribution function of the image itself

Histogram equalisation

- Let I be a discrete image, with grey-level values $\{a_0, \ldots, a_{q-1}\}$, with cumulative distribution H_I
- Histogram equalisation consists in applying the following contrast change to I:

$$g = H_I \tag{22}$$

- The grey-level values of $g(I)$ are uniformly distributed in $\{a_0, \ldots, a_{q-1}\}$

Radiometry and colour

Input image

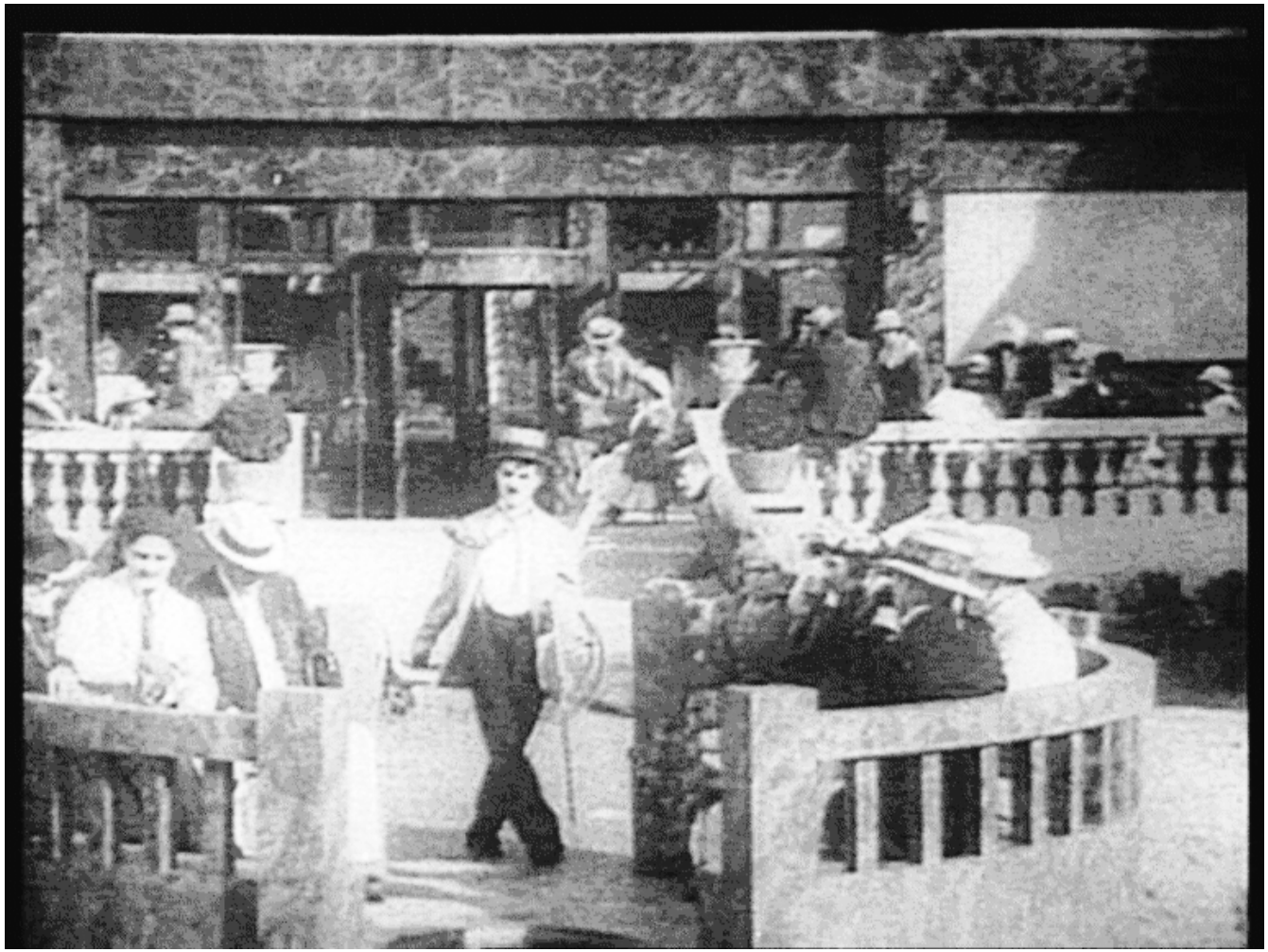
Radiometry and colour

Equalised image

Radiometry and colour

Input image

Radiometry and colour



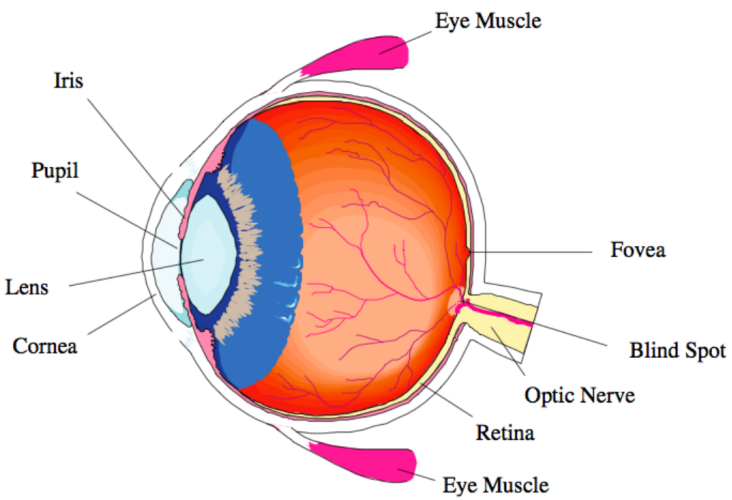
Equalised image

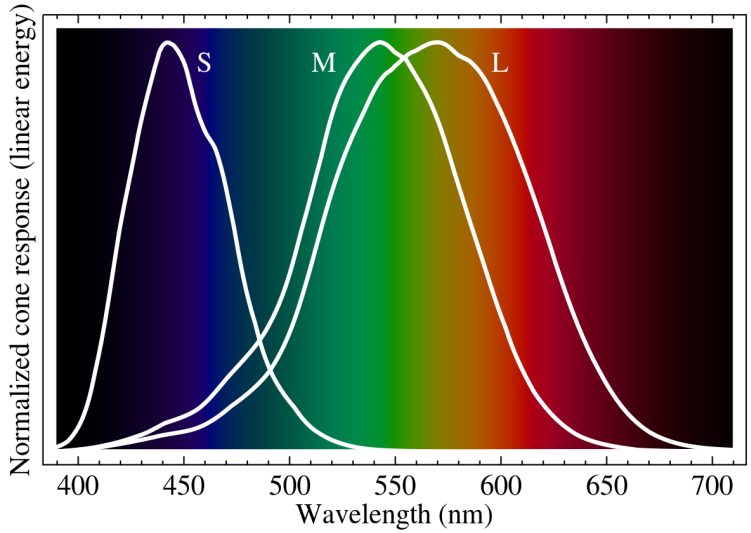
Colour

- Up to this point we have considered grey-scale images
- However, humans perceive **colour images**
- Humans can perceive light of 380nm-700nm wavelengths
  - Red is longer wavelength, blue/violet is shorter

Colour

- It happens (for physiological reasons) that the spectrum of light percieved by humans is a combination of **three principal wavelengths**, red, green, and blue
- Each of our *cones* in the eye's retina is sensitive to a certain spectrum of wavelengths
  - Note that each cones does **not** repond to a single wavelength, but a **range**





Images from wikipedia

Colour

- Suppose we have a distribution f of electromagnetic wavelengths (light), and let ρ_r, ρ_g, and ρ_b be the cones' sensitivities
- Then the colours which we perceive are the points in $\mathbb{R}^3$ (3-tuples) of the form :

$$\langle f, \rho_r \rangle, \langle f, \rho_g \rangle, \langle f, \rho_b \rangle \tag{23}$$

- This is a subset of $\mathbb{R}^3$
- These responses are interpreted by the brain, which gives the sensation of colour

Colour

- We can model a pure wave of light as a Dirac impulse δ_λ
  - λ : wavelength
- If we project δ_λ onto the cone senitivities, for all λ in the visible spectrum, this gives us a curve in $\mathbb{R}^3$
- The projection of this curve onto a plane gives the outline of a **chromaticity diagram**

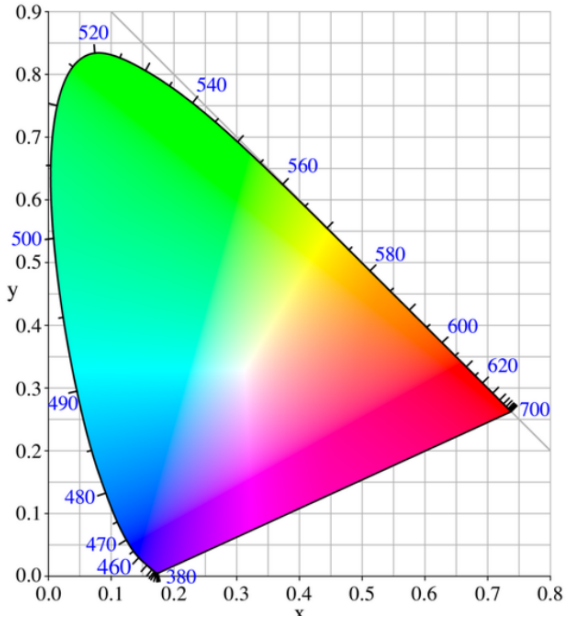
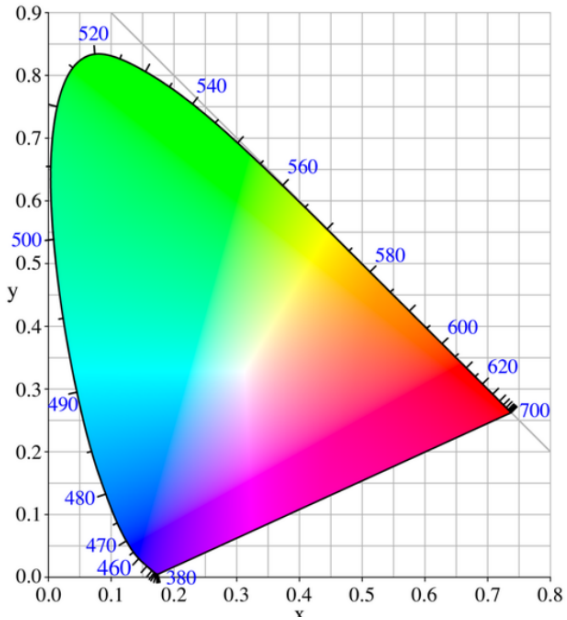


Image from wikipedia

Colour

- Since any distribution f can be approximated by a weighted sum of Diracs, $f \approx \sum_\lambda \alpha_\lambda \delta_\lambda$ (where α_λ are the weights), any perceived colour can be approximated as :

$$\sum_\lambda \alpha_\lambda \langle \delta_\lambda, \rho_r \rangle , \sum_\lambda \alpha_\lambda \langle \delta_\lambda, \rho_g \rangle , \sum_\lambda \alpha_\lambda \langle \delta_\lambda, \rho_b \rangle$$



Colour

- In general we approximate with three principal colours (Diracs)
- This gives us a colour triangle of possible colours (convex combinations)
- Therefore : the set of all colours which we can perceive is larger than those which can be created by a combination of three colours

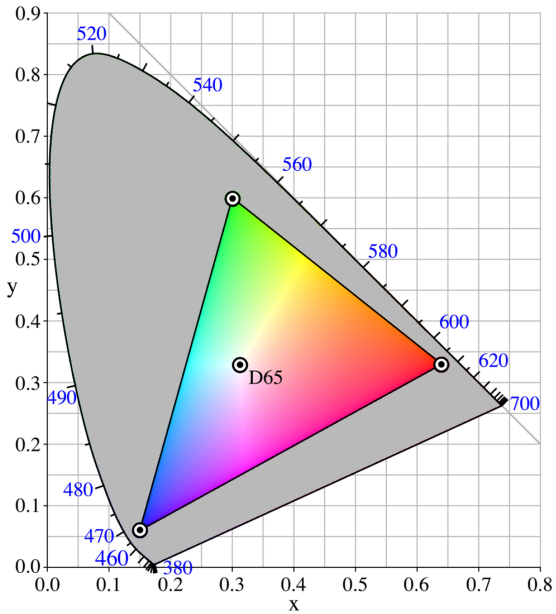
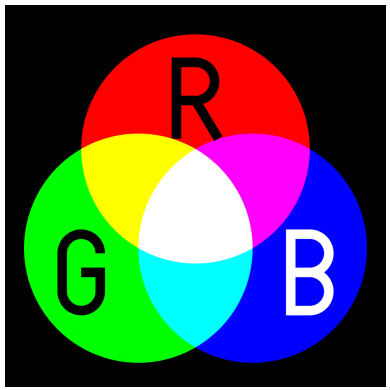


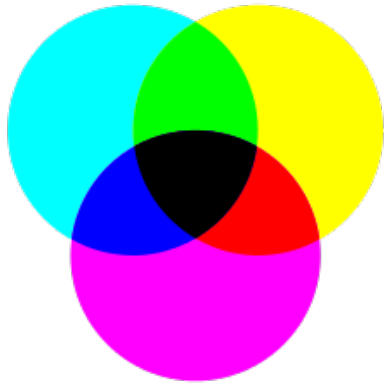
Image from wikipedia

Colour

- Different colour models give different ways of representing the visible spectrum
  - The RGB (Red-Green-Blue) model creates a colour by **adding** these three colours
  - The CMY (Cyan-Magenta-Yellow) model creates a colour by **subtracting** the three colours



RGB colour model



CMYK colour model

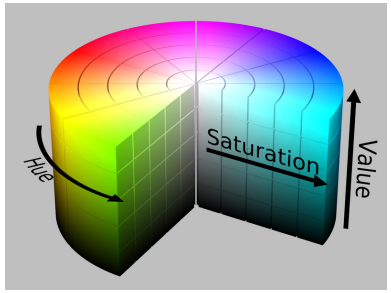
- From a colour model, a **colour space** can be established. This is a manner of representing a colour using a tuple of scalars

Colour

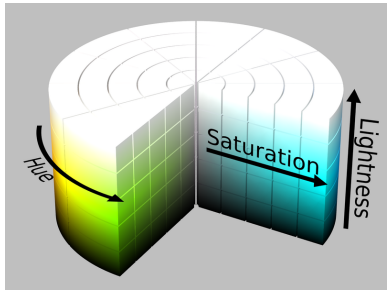
- A natural colour space is Red-Green-Blue (RGB), since this corresponds to the cones sensitivities
- However, this is not necessarily the best representation for certain tasks
- Often, we wish to separate the **luminance** of an object from its **chromacity**
  - If an object is a specific colour, we do not care about the intensity
  - If we wish to distinguish dark from light parts of an image, we do not care about colour

Colour

- There are several colour spaces which do this. Most commonly :
  - Hue-Saturation-Value (HSV), Hue-Saturation-Lightness (HSL)
- **Hue** corresponds to what we think of as a (pure) colour
- **Saturation** corresponds to the amount of colour from the pure colour
  - If the saturation is low, we get white
  - If the saturation is high, we get a pure colour
- **Value and lightness** correspond to the magnitude of the colour
  - Low value gives dark colour
  - High value gives bright colour



HSV colour space



HSL colour space

Images from wikipedia

Colour

Hue

Colour

Saturation

Colour

Value

Summary

- Viewing grey-level values as random variables with a certain distribution (histogram) is useful for analysing contrast in images
  - Histogram equalisation
- Different colour spaces exist which may be useful for separating different aspects of colour images
  - Red-Green-Blue, Hue, Saturation, Value
- Different representations of colour may be more or less appropriate for different tasks (in particular classification)

Summary

Mathematical morphology

- Very often, we wish to manipulate **binary images**
- This happens, for example, during segmentation

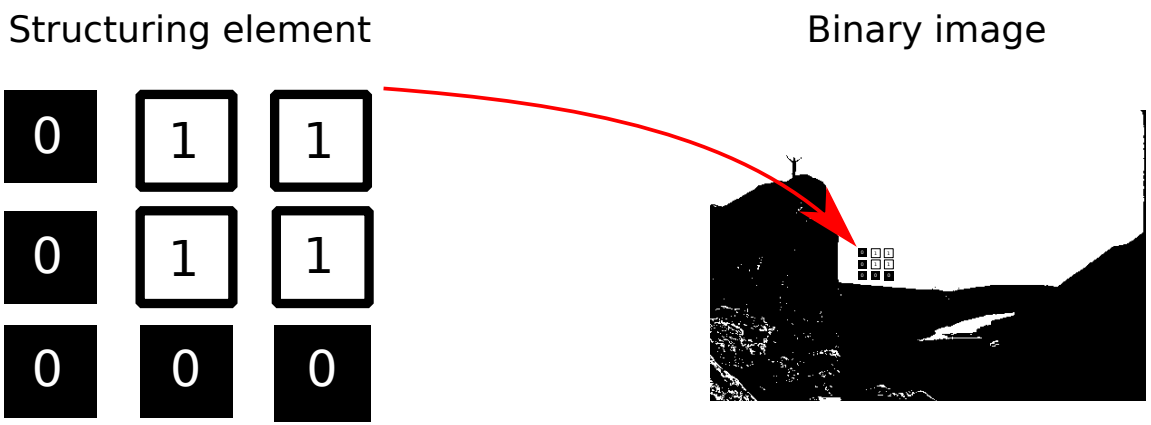
Input image

Segmentation

- We might want to shrink/enlarge the segmentation, close/fill holes etc.

Mathematical morphology

- Such operations carried out with **mathematical morphology** (MM)
  - MM can be applied to very general topicalogical structures, but we concentrate on its application to binary digital images
- Main idea: analyse the binary image using a binary **structuring element**
  - The structuring element is defined the user, depends on the problem
- MM operators take as input a binary image and a structuring element
  - Similar to a convolutional filter



- Main MM operators : dilation, erosion, opening, closing

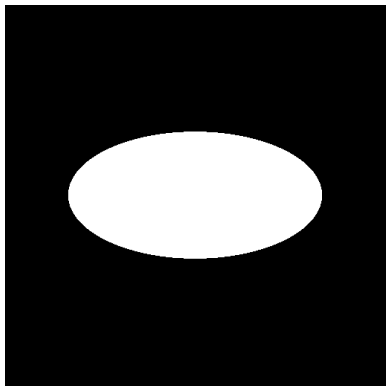
Erosion

- Let B be a structuring element, with support $\text{Supp}(B)$, and I a binary image
- The binary erosion operator is denoted with the $\ominus$ symbol, and is defined as

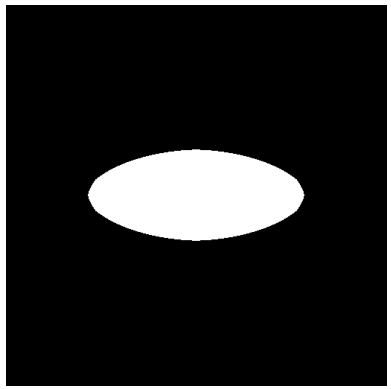
$$(B \ominus I)(p) = \begin{cases} 1 & \text{if } B(q) = 1 \text{ and } I(p+q) = 1, \ \forall q \in \text{Supp}(B) \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

- You can also write this using set notation (with some framework modifications)

$$(B \ominus I)(p) = \bigcap_{q \in \text{Supp}(B)} (B(q) = 1 \ \cap \ I(p+q) = 1) \tag{25}$$



Input image



Eroded image

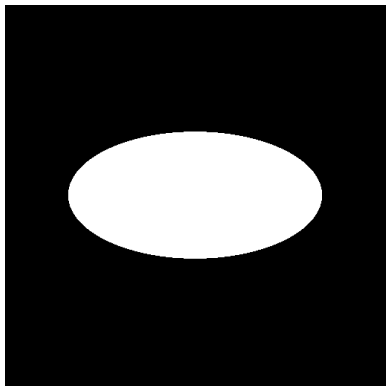
Dilation

- The binary dilation operator is denoted with the $\oplus$ symbol, and is defined as

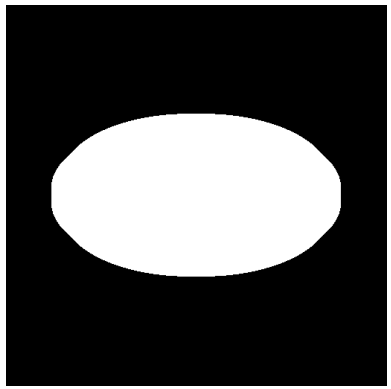
$$(B \oplus I)(p) = \begin{cases} 1 & \text{if } B(q) = 1 \text{ or } I(p+q) = 1, \ \forall q \in \text{Supp}(B) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

- You can also write this using set notation (with some framework modifications)

$$(B \ominus I)(p) = \bigcup_{q \in \text{Supp}(B)} (B(q) = 1 \ \cup \ I(p+q) = 1) \tag{27}$$



Input image



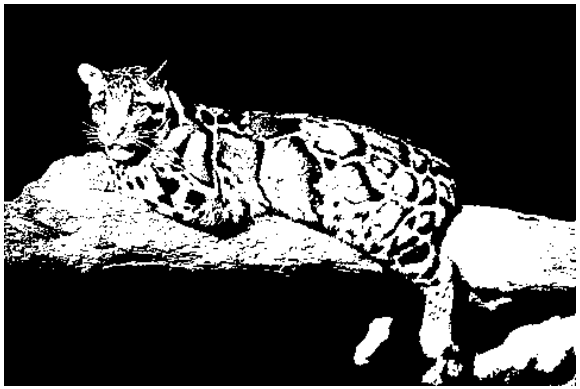
Dilated image

Closing

- The binary **closing** operator is denoted with the $\bullet$ symbol
- A dilation followed by an erosion
- Used to **fill small holes** in binary images

$$B \bullet I = (B \oplus I) \ominus B \tag{28}$$

Input image



Binary segmentation

Image after closing

Opening

- The binary **opening** operator is denoted with the $\circ$ symbol
- An erosion followed by a dilation
- Used to **remove small objects** in binary images (similar to denoising)

$$B \circ I = (B \ominus I) \oplus B \tag{29}$$

Input image

Binary segmentation

Image after opening

Mathematical morphology

- A final useful operation is called **skeletonisation**
  - Reducing a binary image to a skeleton of one pixel thickness
- This can be done with a variety of techniques such as thinning
- Very useful for modelling objects, people etc for motion description

Input image

Binary segmentation, after closing

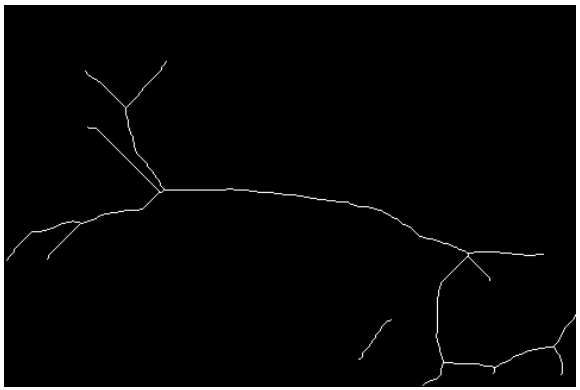


Image after skeletonisation

Mathematical morphology

- Summary : four main MM operators for binary image maniplulation
  1. Erosion : shrinking objects
  2. Dilation : enlarging objects
  3. Closing : filling in small holes
  4. Opening : removing small objects
  5. Skeletonisation : finding a skeleton/sketch of an object
- Note that the closing and opening operators do not modify convex shapes with no holes (remain the same)
- MM is quite a general theory and can also be applied to grey-level images, however we do not investigate this here