Digital Signals and Image Processing

Semester 2, 2024

Dr Maryam Haghighat

Tarang Janawalkar

Contents

1 Digital Image Processing

Digital image processing is the processing of images on a digital system using algorithms. A digital image is a binary representation of visual data that is composed of a finite number of elements, each with a particular location and value.
Image processing methods can be divided into two categories:

- Methods where the input and output are images.
- Methods where the input is an image and the output is some information extracted from the image.

1.1 Elements of Visual Perception

The human visual system has influenced and contributed to many advancements in image processing. The human eye has light receptors called rods and cones. Humans have around 6 to 7 million cones in each eye that are highly sensitive to colour and fine details. On the other hand, there are a total of 75 to 150 million rods across both eyes, that are sensitive to low levels of illumination.

1.1.1 Image Formation

Photo camera lenses are fixed in focal length and they focus at various distances by varying the distance between the lens and imaging plane (film/chip). The human eye works in the opposite way, where the distance between the lens and the imaging plane (retina) is fixed, but the focal length for focus is varied by changing the shape of the lens.

1.1.2 Brightness Adaptation and Discrimination

The human eye is capable of discriminating between a wide range of intensity levels. This range of light intensity levels is on the order of 10^{10}.

1.2 Image Generation Components

There are three components to image generation:

- **Object**: The object being imaged.
- **Energy Source**: The source of energy that illuminates the object.
- **Sensor**: The sensor that detects the energy reflected from the object.

Depending on the properties of the energy source and object material and geometry, the emitted energy can be reflected, transmitted, or absorbed.

1.2.1 Electromagnetic Spectrum

The main source of energy for imaging is electromagnetic (EM) radiation. EM radiation consists of propagating sinusoidal waves characterised by their oscillating frequency f. Using Planck's equation, the energy of a photon can be calculated as

$$E = hf$$

where $h = 6.626\,070\,15 \times 10^{-34}\,\mathrm{J\,s}$ is Planck's constant and f is the frequency of the wave. Given the speed of light $c = 299\,792\,458\,\mathrm{m\,s^{-1}}$, we can also calculate the energy using

$$E = \frac{hc}{\lambda}.$$

The EM spectrum is divided into regions based on the frequency of the waves. The visible light spectrum is a small part of the EM spectrum that is visible to the human eye. The visible light spectrum ranges from 380 nm to 700 nm. Earth's atmosphere also blocks certain parts of the EM spectrum, such as short wavelength UV and X-rays.
Perceived colour (hue) is related to the wavelength of light, while the **brightness** is related to the intensity of the radiation.

1.2.2 Image Sensors

Image sensors capture a specific range of the EM spectrum, for example:

- RGB sensors capture the visible light spectrum.
- Infrared sensors capture the infrared spectrum.
- X-ray sensors capture the X-ray spectrum.
- Ultraviolet sensors capture the ultraviolet spectrum.

1.2.3 Human Perception

Human perception is context-dependent. Perceived intensity around regions of discontinuous intensity appear to undershoot and overshoot around the boundary (see Mach band effect). The eye can also fill in non-existing information and wrongly perceive geometrical properties of objects. To produce a powerful vision system, we need both a powerful image sensor and image processor to extract useful information from an image.

1.3 Image Sensing and Acquisition

Image sensing is the process of transforming illuminated energy into a digital image. The process involves the following steps:

1. Convert the illuminated energy into an electrical signal.
2. Digitize the electrical signal to obtain a digital image.

1.3.1 Image Sensing Modalities

Image sensing is done using three principal modalities:

- **Single Sensing Element**: A single sensor that captures the energy. For example, a photodiode. To generate 2D images, the sensor must be appropriately displaced in the x and y directions.

- **Line Sensor**: A sensor that captures energy along a line. To generate 2D images, the sensor must be displaced in the direction perpendicular to the line.
- **Array Sensor**: A sensor that captures energy in a 2D array. The sensor is divided into rows and columns, with each element capturing energy at a specific location. A typical arrangement is the **CCD** (Charge Coupled Device) sensor.

1.3.2 Image Formation

Let us denote the intensity of a monochrome image by the 2-dimensional function

$$\ell = f(x,\, y)$$

where x and y represent the spatial coordinates captured by the sensor, and f is a scalar function of the intensity of the energy radiated by a physical source. As such, this function is non-negative and finite:

$$0 \leqslant \ell \leqslant \infty.$$

f is characterised by two components:

- **Illumination**: The amount of source illumination incident on the scene $i(x,\, y)$. Here $0 \leqslant i(x,\, y) < \infty$.
- **Reflectance**: The amount of illumination reflected by the objects on the scene $r(x,\, y)$. Here $0 \leqslant r(x,\, y) \leqslant 1$.

Therefore, we can describe the image formation process as

$$f(x,\, y) = i(x,\, y)\, r(x,\, y)\,,$$

where $r = 0$ implies total absorption, while $r = 1$ implies total reflectance. For monochrome images, we can define the minimum and maximum intensity values as $L_{\min}$ and $L_{\max}$, respectively, where

$$L_{\min} \leqslant \ell \leqslant L_{\max}, \quad L_{\min} = i_{\min} r_{\min}, \quad L_{\max} = i_{\max} r_{\max}.$$

The range of intensity values $[L_{\min}, L_{\max}]$ is called the **intensity/grey scale**. Commonly, this interval is transformed to the interval $[0, L-1]$, where L is the number of intensity levels.

1.4 Image Sampling and Quantisation

Image sampling is the process of sampling discrete points in a continuous image. Regardless of the sensor arrangement, the image is sampled at a fixed rate in the x and y directions and the resulting points are called **pixels**. These pixels are stored in an array that is $M \times N$ in size, where M is the number of rows and N is the number of columns, arranged as shown below:

$$\mathbf{X} = \begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0,N-1} \\ x_{10} & x_{11} & \cdots & x_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M-1,0} & x_{M-1,1} & \cdots & x_{M-1,N-1} \end{bmatrix}$$

Image quantisation is the process of converting the continuous intensity values of an image to discrete values. The number of intensity levels L is determined by the number of bits used to represent each pixel. The number of intensity levels is given by

$$L = 2^k,$$

where k is the bit depth. Thus the quantised intensity values will range from $[0, L-1]$. The **quality** of an image is determine by the number of discrete intensity levels used in both sampling and quantisation.

1.4.1 Dynamic Range and Contrast

The **dynamic range** of an image is the ratio of the maximum intensity value to the minimum intensity value:

$$\text{Dynamic Range} = \frac{L_{\max}}{L_{\min}} = \frac{i_{\max} r_{\max}}{i_{\min} r_{\min}}.$$

The upper limit is determined by the sensor's saturation level, while the lower limit is determined by the sensor's noise level. The **contrast** of an image is the difference in intensity between the brightest and darkest regions of the image:

$$\text{Contrast} = L_{\max} - L_{\min} = i_{\max} r_{\max} - i_{\min} r_{\min}.$$

- A **high** dynamic range implies a large difference between the brightest and darkest regions of the image, and therefore high contrast.
- A **low** dynamic range implies a small difference between the brightest and darkest regions of the image, and therefore low contrast.

1.4.2 Spatial and Intensity Resolution

The **spatial resolution** of an image is a measure of the smallest discernible detail in an image, measured as the number of pixels per unit area (dots per inch). In some images, sampling an image at a low rate can result in **aliasing**, where high-frequency components are incorrectly represented as low-frequency components (see the Moiré pattern).
The **intensity resolution** of an image is the smallest discernible change in intensity level, which is related to the number of intensity levels used to represent the image, for example, 8-bit and 10-bit images. Choosing a low number of intensity levels can result in quantisation noise, where intensity levels are incorrectly represented.

1.4.3 Image Interpolation

Image interpolation is the process of estimating the intensity values of pixels between the sampled points. Interpolation is used to increase and decrease the resolution of an image for resampling and resizing. Common interpolation methods include:

- **Nearest Neighbour**: The intensity value of the nearest pixel is used to estimate the intensity value of the pixel.

- **Bilinear**: The intensity value of the nearest four pixels is used to estimate the intensity value of the pixel.
- **Bicubic**: The intensity value of the nearest sixteen pixels is used to estimate the intensity value of the pixel.

1.5 Relationships Between Pixels

The following sections will define some common sets that are used to describe relationships between pixels in an image.

1.5.1 Neighbours of a Pixel

The **neighbours** of a pixel $\mathbf{p} = (x,\, y)$ are the pixels that are adjacent to $\mathbf{p}$.

- The **4-neighbours** of $\mathbf{p}$ are defined as the pixels that are adjacent to $\mathbf{p}$ in the **cardinal** directions:
$$\mathbf{N}_4(\mathbf{p}) = \{(x,\, y-1), (x-1,\, y), (x+1,\, y), (x,\, y+1)\},$$
- The **diagonal-neighbours** of $\mathbf{p}$ are defined as the pixels that are adjacent to $\mathbf{p}$ in the **diagonal** directions:
$$\mathbf{N}_D(\mathbf{p}) = \{(x-1,\, y-1), (x+1,\, y-1), (x-1,\, y+1), (x+1,\, y+1)\},$$
- The **8-neighbours** of $\mathbf{p}$ are defined as the pixels that are adjacent to $\mathbf{p}$ in both the **cardinal** and **diagonal** directions:
$$\mathbf{N}_8(\mathbf{p}) = \mathbf{N}_4(\mathbf{p}) \cup \mathbf{N}_D(\mathbf{p}).$$

1.5.2 Adjacency and Connectivity

Two pixels $\mathbf{p}$ and $\mathbf{q}$ are **adjacent** if they are neighbours and their intensity values are similar or belong to the same set of values V based on a threshold. This can occur in one of three ways:

- **4-Adjacency** when $\mathbf{q} \in \mathbf{N}_4(\mathbf{p})$.
- **8-Adjacency** when $\mathbf{q} \in \mathbf{N}_8(\mathbf{p})$.
- **M-Adjacency** when $\mathbf{q} \in \mathbf{N}_4(\mathbf{p})$ **or** $\mathbf{q} \in \mathbf{N}_D(\mathbf{p})$ **and** $\mathbf{N}_4(\mathbf{p}) \cap \mathbf{N}_4(\mathbf{q}) = \varnothing$. This statement avoids double-counting an adjacency when $\mathbf{q}$ is a diagonal neighbour of $\mathbf{p}$ while another cardinal neighbour exists between $\mathbf{p}$ and $\mathbf{q}$. Consider the binary example with $\mathbf{p}, \mathbf{q}, \mathbf{r} \in V = \{1\}$:
$$\begin{bmatrix} 0 & \mathbf{r} & \mathbf{q} \\ 0 & \mathbf{p} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$
Here we do not consider $\mathbf{p}$ and $\mathbf{q}$ to be M-adjacent, as $\mathbf{p}$ and $\mathbf{q}$ are already adjacent through $\mathbf{r}$.

A **path** (or curve) between two pixels $\mathbf{p}$ and $\mathbf{q}$ is a sequence of $n+1$ pixels $(\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$ such that $\mathbf{p}_0 = \mathbf{p}$ and $\mathbf{p}_n = \mathbf{q}$, where $\mathbf{p}_i$ is adjacent to $\mathbf{p}_{i+1}$ for $i = 0, 1, 2, \ldots, n-1$. This path is said to have length n, and is *closed* if $\mathbf{p}_0 = \mathbf{p}_n$.

1.5.3 Connectivity

Consider a subset of pixels in an image S with $\mathbf{p}, \mathbf{q} \in S$.

- $\mathbf{p}$ and $\mathbf{q}$ are **connected** if there exists a path between $\mathbf{p}$ and $\mathbf{q}$ such that all pixels in the path are in S.
- The set of pixels connected to $\mathbf{p}$ in S form a **connected component**.
- If S only consists of *one* connected component, then S is said to be a **connected set** and is called a **region** R.

Two regions R_1 and R_2 are adjacent if their union forms a connected set, i.e., another region. Regions that are not adjacent are said to be **disjoint**.

1.6 Distance Metrics

The **distance** between two pixels $\mathbf{p}$ and $\mathbf{q}$ can be measured using a variety of metrics. The most common metrics are:

- **Euclidean distance**:
$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}.$$
- **Manhattan (D4) distance** (or city-block distance):
$$d_4(\mathbf{p}, \mathbf{q}) = |x_p - x_q| + |y_p - y_q|.$$
- **Chessboard (D8) distance** (or maximum distance):
$$d_8(\mathbf{p}, \mathbf{q}) = \max\left(|x_p - x_q|, |y_p - y_q|\right).$$

In general, any metric d that satisfies the following properties is a **distance metric**:

- The distance from a point to itself is zero:
$$d(\mathbf{p}, \mathbf{p}) = 0.$$
- **Positivity**: The distance between two distinct points is always positive:
$$d(\mathbf{p}, \mathbf{q}) > 0 : \mathbf{p} \neq \mathbf{q}.$$
- **Symmetry**: The distance between two points is always the same regardless of ordering:
$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}).$$
- The Triangle Inequality is satisfied:
$$d(\mathbf{p}, \mathbf{q}) \leq d(\mathbf{p}, \mathbf{r}) + d(\mathbf{r}, \mathbf{q}).$$

for all $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ in this metric space.

1.7 Mathematical Operations

1.7.1 Element-wise Operations

Given two images $\mathbf{X}$ and $\mathbf{Y}$ of equal dimensions, an element-wise operation is an operation that is applied to each pixel in the image. Suppose we wish to apply the binary operation $\otimes$ on $\mathbf{X}$ and $\mathbf{Y}$. The resulting image $\mathbf{Z}$ is given by

$$\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{00} \otimes y_{00} & x_{01} \otimes y_{01} & \cdots & x_{0,N-1} \otimes y_{0,N-1} \\ x_{10} \otimes y_{10} & x_{11} \otimes y_{11} & \cdots & x_{1,N-1} \otimes y_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M-1,0} \otimes y_{M-1,0} & x_{M-1,1} \otimes y_{M-1,1} & \cdots & x_{M-1,N-1} \otimes y_{M-1,N-1} \end{bmatrix}$$

Here the operator $\otimes$ can represent any binary operation, such as addition, subtraction, multiplication, or division.

1.7.2 Linear Operations

The operator H is a linear operation if it satisfies the following property:

$$H\left[a\mathbf{X} + b\mathbf{Y}\right] = aH\left[\mathbf{X}\right] + bH\left[\mathbf{Y}\right],$$

for images $\mathbf{X}$ and $\mathbf{Y}$, where a and b are constants.

1.7.3 Spatial Operations

The operator T is a spatial operation if the output pixel value is determined by the values of pixels in the neighbourhood of any input pixel. T can be categorised as one of three types of spatial operations:

- **Pointwise**: The output pixel value is determined by the value of the input pixel only.
- **Neighbourhood**: The output pixel value is determined by the value of the input pixel and any neighbouring pixels.
- **Global**: The output pixel value is determined by the value of pixels in the entire image.
- **Geometric**: The output pixel value is determined by the spatial location of the pixel.

2 Intensity Transformations & Spatial Filters

2.1 Intensity Transformations

An intensity transform aims to modify the contrast of an image by changing the range of intensity values in that image. The following sections will define some common intensity transformations. In the following sections let r be the input intensity and s be the output intensity of an image.

2.1.1 Identity Transformation

The identity transformation is the simplest intensity transformation which does not alter an image. It is defined as

$$s = T\left(r\right) = r.$$

2.1.2 Negative Transformation

The image negation transformation is used to invert the intensity values of an image. It is defined as

$$s = T(r) = (L-1) - r,$$

where L is the number of intensity levels in the image.

2.1.3 Logarithmic Transformation

The logarithmic transformation is used to enhance darker regions of an image by compressing brighter regions. It is defined as

$$s = T(r) = c \log(1+r),$$

where c is a constant that scales the intensity values of the image.

2.1.4 Power-Law (Gamma) Transformation

The power-law transformation is used to correct the gamma of an image, either by enhancing or reducing dark or bright regions. It can be thought of as a generalisation of the logarithmic transformation. It is defined as

$$s = T(r) = cr^{\gamma},$$

where c is a constant that scales the intensity values of the image and γ is the gamma value.

- When $\gamma < 1$, the transformation enhances the darker regions of the image, while compressing the brighter regions.
- When $\gamma > 1$, the transformation enhances the brighter regions of the image, while compressing the darker regions.

2.1.5 Piecewise-Linear Transformation

Piecewise-linear transformations are used to enhance the contrast of specific regions of an image. Some common piecewise-linear transformations include:

- **Contrast Stretching**: Enhances the contrast of an image by stretching the intensity values to the full range of intensity levels.
- **Intensity Level Slicing**: Enhances the contrast of specific regions of an image by setting the intensity values of other regions to zero or by leaving them unchanged.
- **Bit-Plane Slicing**: Highlights the contribution made to image appearance by specific bits in the image.

2.2 Histogram Processing

Histograms are used to visualise the distribution of intensity values in an image. Given an image $\mathbf{X} \in [0, L-1]^{M \times N}$, the histogram $h_{\mathbf{X}}(k)$ is defined as

$$h_{\mathbf{X}}(k) = n_k,$$

where n_k is the number of pixels in the image with intensity value k. If we normalise these values, we find the probability of obtaining a pixel with intensity value k:

$$p_{\mathbf{X}}(k) = \frac{n_k}{MN}.$$

It follows that

$$\sum_{k=0}^{L-1} p_{\mathbf{X}}(k) = 1.$$

2.2.1 Histogram Equalisation

Histogram equalisation is a method used to spread the most frequent intensity values in an image to the full range of intensity levels, thereby achieving a more uniform distribution of intensity values. To do this, we will use the following transformation that maps the cumulative distribution function of an input image $\mathbf{X}$ to the cumulative distribution function of a uniform distribution:

$$s = T(r) = (L-1) \sum_{j=0}^{r} p_{\mathbf{X}}(j) = \frac{L-1}{MN} \sum_{j=0}^{r} n_j.$$

2.2.2 Histogram Matching

In some cases, we wish to match the histogram of $\mathbf{X}$ to the histogram of another image $\mathbf{Y}$. To do so, consider the histogram equalisation of $\mathbf{Y}$ with intensity values z:

$$s = G(z) = (L-1) \sum_{j=0}^{z} p_{\mathbf{Y}}(j) = \frac{L-1}{MN} \sum_{j=0}^{z} n_j.$$

Thus we have the mapping:

$$T : r \mapsto s \quad \text{and} \quad G : z \mapsto s.$$

As both T and G map to the same equalised space, we can define the transformation $z = H(r)$ that maps the histogram of $\mathbf{X}$ to the histogram of $\mathbf{Y}$ as

$$z = H(r) = G^{-1}(T(r)) = G^{-1}(s).$$

2.3 Spatial Filtering

Spatial filtering is the process of creating a new image by applying a mask (or kernel, template or window) to each pixel in an image. This new pixel value is determined by the intensity values of the pixels in the neighbourhood of the original pixel. The mask is defined as a $m \times n$ matrix $\mathbf{W}$ with elements w_{ij} that represent the weights of the pixels in the neighbourhood of the pixel being processed. For convenience, m and n are typically odd integers.
The output pixel value is given by the weighted sum of the intensity values of the pixels in the neighbourhood of the pixel. This can be done using one of two operations:

- **Correlation**: The mask is shifted across the image and the weighted sum is calculated at each position.
- **Convolution**: The mask is first flipped horizontally and vertically before it is shifted across the image.

2.3.1 Correlation

The $(i,\ j)$th element of the correlation of an image $\mathbf{X}$ with a mask $\mathbf{W}$ is defined as

$$y_{ij} = \mathbf{W} \star \mathbf{X} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w_{st} x_{i+s,j+t}$$

for $a = \frac{m-1}{2}$ and $b = \frac{n-1}{2}$.

2.3.2 Convolution

The $(i,\ j)$th element of the convolution of an image $\mathbf{X}$ with a mask $\mathbf{W}$ is defined as

$$y_{ij} = \mathbf{W} * \mathbf{X} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w_{st} x_{i-s,j-t}$$

for $a = \frac{m-1}{2}$ and $b = \frac{n-1}{2}$.

2.3.3 Padding

For masks larger than 1×1, the indices w_{st} will exceed the bounds of the image $\mathbf{X}$. To prevent this, the image is often padded with an additional border of pixels. Common padding methods include:

- **Zero Padding**: The border is padded with zeros.
- **Boundary Replication Padding**: The border is padded with the intensity values of the nearest pixel.
- **Reflection Padding**: The border is padded with the intensity values of reflected pixels (one pixel away from the border).

2.3.4 Averaging Filters

Averaging filters are used to reduce noise in an image by averaging the intensity values of the pixels in the neighbourhood of the pixel being processed. An averaging filter considers a continuous function of two variables, such as the multivariable Gaussian function:

$$w_{st} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{s^2 + t^2}{2\sigma^2}\right).$$

2.3.5 Smoothing Linear Filters

Smoothing filters are used to reduce noise in an image by averaging the intensity values of the pixels in the neighbourhood of the pixel being processed. A general implementation for filtering an $M \times N$ image with a **weighted averaging filter** of size $m \times n$ is defined as:

$$y_{ij} = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w_{st} x_{i+s,j+t}}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w_{st}},$$

2.3.6 Order-Statistic Non-Linear Filters

Order-statistic filters are used to reduce noise in an image by replacing the intensity value of a pixel with the median, maximum, or minimum intensity value of the pixels in the neighbourhood of the pixel being processed. Median filters have good noise-reduction capabilities with less smoothing and are used to remove impulse or salt-and-pepper noise.

2.3.7 Sharpening Filters

Sharpening filters are used to enhance edges and discontinuities in an image. One technique is to consider the Laplacian of the image $f(x,\, y)$:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

For discrete images, we will use the first-order forward difference approximation:

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

and the second-order central difference approximation:

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} &= f(x+1,\, y) - 2f(x,\, y) + f(x-1,\, y) \\ \frac{\partial^2 f}{\partial y^2} &= f(x,\, y+1) - 2f(x,\, y) + f(x,\, y-1). \end{aligned}$$

The Laplacian allows us to identify transitions in intensity values across an image by creating a filter with one of the following masks:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

where the first mask will sharpen the image, while the second mask will sharpen the negative of the image. In general, we can sharpen edges by adding the Laplacian to the original image:

$$g(x,\, y) = f(x,\, y) + c\nabla^2 f(x,\, y),$$

where $c = -1$ will sharpen the image, while $c = 1$ sharpen the negative of the image.

2.3.8 Unsharp Masking and High-Boost Filtering

Unsharp masking is a sharpening technique that enhances edges and discontinuities in an image by subtracting a blurred version of the image from the original image. The process takes the following steps:

1. Blur the original image $f(x,\, y)$.
2. Subtract the blurred image from the original image to obtain the mask

$$g_{\text{mask}}(x,\, y) = f(x,\, y) - \bar{f}(x,\, y),$$

   where $\bar{f}(x,\, y)$ is the blurred image.

3. Add the mask to the original image to obtain a sharpened image:

$$g(x,\, y) = f(x,\, y) + kg_{\text{mask}}(x,\, y), \quad k > 0.$$

   where

   - $k = 1$ corresponds to unsharp masking.
   - $k > 1$ corresponds to high-boost filtering.

3 Filtering in the Frequency Domain

3.1 2D Fourier Transform

The 2D Fourier transform of an image $f(x,\, y)$[1] is defined as

$$F(u,\, v) = \mathscr{F}\{f(x,\, y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,\, y)\, e^{-j2\pi(ux+vy)}\, \mathrm{d}x\, \mathrm{d}y,$$

with the inverse 2D Fourier transform defined as:

$$f(x,\, y) = \mathscr{F}^{-1}\{F(u,\, v)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,\, v)\, e^{j2\pi(ux+vy)}\, \mathrm{d}u\, \mathrm{d}v.$$

When f is a discrete $M \times N$ image, the 2D discrete Fourier transform is defined as

$$F(u,\, v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,\, y)\, e^{-j2\pi(ux/M+vy/N)},$$

with the inverse 2D discrete Fourier transform defined as

$$f(x,\, y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,\, v)\, e^{j2\pi(ux/M+vy/N)}.$$

Many properties of the Fourier transform in the 1D case extend to the 2D case, such as linearity, symmetry, and shift properties. Notably, the sampling theorem in 2D states that an image can be

[1] f must satisfy the Dirichlet conditions for the 2D Fourier transform to exist.

perfectly reconstructed from its samples if the sampling rate f_s is greater than twice the highest frequency component in the image:

$$f_s > 2f_{\max}.$$

In 2D, this is equivalent to:

$$\frac{1}{\Delta x} > 2u_{\max} \quad \text{and} \quad \frac{1}{\Delta y} > 2v_{\max},$$

where $u_{\max}$ and $v_{\max}$ are the maximum frequencies in the x and y directions, respectively. The frequency of spatial and frequency samples can also be related by the following equations:

$$\Delta u = \frac{1}{M\,\Delta x} \quad \text{and} \quad \Delta v = \frac{1}{N\,\Delta y}.$$

3.2 Filtering in the Frequency Domain

Filtering can also be performed in the frequency domain if the image contains noise that is more easily removed when visualising the magnitude and phase spectra of the image. In general, a filter in the frequency domain is defined as

$$g\left(x,\, y\right) = \mathscr{F}^{-1}\left\{H\left(u,\, v\right) F\left(u,\, v\right)\right\},$$

where $H\left(u,\, v\right)$ is a filter designed to act on the frequency spectrum of the image. Due to the lack of padding, the horizontal or vertical edges of the resulting image may contain black pixels, leading to inconsistent filtering. To avoid this, we can take the following steps:

1. For an image $f\left(x,\, y\right)$ of size $M \times N$, define padding parameters $P = 2M$ and $Q = 2N$.
2. Form a padded image $f_p\left(x,\, y\right)$ of size $P \times Q$ by appending the necessary number of zeros to f.
3. Multiply the padded image by $\left(-1\right)^{x+y}$ to centre the transform at the origin.
4. Compute the 2D Fourier transform of the transformed padded image.
5. Multiply the result by the symmetric filter $H\left(u,\, v\right)$ of size $P \times Q$ to form $G\left(u,\, v\right)$.
6. Obtain the processed image $g_p\left(x,\, y\right) = \Re\left\{\mathscr{F}^{-1}\left\{G\left(u,\, v\right)\right\}\right\}\left(-1\right)^{x+y}$.
7. Obtain the final processed image $g\left(x,\, y\right)$ by cropping the processed image to the original size $M \times N$, which is in the top-left corner of the processed image.

3.3 Image Smoothing

3.3.1 Ideal Low-Pass Filters

An ideal low-pass filter passes all frequencies within a circle of radius D_0 from the origin and suppresses all frequencies outside of this circle, without attenuation:

$$H\left(u,\, v\right) = \begin{cases} 1 & \text{if } D\left(u,\, v\right) \leq D_0, \\ 0 & \text{if } D\left(u,\, v\right) > D_0, \end{cases}$$

Here $D\left(u,\, v\right)$ is the distance from the origin to the point $\left(u,\, v\right)$ in the frequency domain:

$$D\left(u,\, v\right) = \sqrt{\left(u - P/2\right)^2 + \left(v - Q/2\right)^2}.$$

3.3.2 Butterworth Low-Pass Filters

A Butterworth low-pass filter is defined as

$$H(u,\, v) = \frac{1}{1 + \left(D(u,\, v)/D_0\right)^{2n}},$$

where n is the order of the filter. A higher-order filter has a sharper transition between the passband and the stopband, but also results in more rippling in the spatial domain.

3.3.3 Gaussian Low-Pass Filters

A Gaussian low-pass filter is defined as

$$H(u,\, v) = \exp\left(-\frac{D^2(u,\, v)}{2D_0^2}\right),$$

where D_0 is the standard deviation of the Gaussian filter.

3.4 Image Sharpening

3.4.1 Ideal High-Pass Filters

An ideal high-pass filter passes all frequencies outside a circle of radius D_0 from the origin and suppresses all frequencies inside of this circle, without attenuation:

$$H(u,\, v) = \begin{cases} 0 & \text{if } D(u,\, v) \leq D_0, \\ 1 & \text{if } D(u,\, v) > D_0. \end{cases}$$

High-pass equivalent filters can be generated for all of the above low-pass filters by taking the complement of the low-pass filter:

$$H_{\mathrm{HP}}(u,\, v) = 1 - H_{\mathrm{LP}}(u,\, v).$$

3.4.2 Butterworth High-Pass Filters

A Butterworth high-pass filter is defined as

$$H(u,\, v) = \frac{1}{1 + \left(D_0/D(u,\, v)\right)^{2n}},$$

where n is the order of the filter.

3.4.3 Gaussian High-Pass Filters

A Gaussian high-pass filter is defined as

$$H(u,\, v) = 1 - \exp\left(-\frac{D^2(u,\, v)}{2D_0^2}\right),$$

where D_0 is the standard deviation of the Gaussian filter.

3.4.4 Laplacian Filters

We can consider an alternative formulation for the Laplacian using a filter the frequency domain. In the frequency domain, the Laplacian becomes

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = -4\pi^2 u^2 F(u,\, v) - 4\pi^2 v^2 F(u,\, v) = -4\pi^2 \left(u^2 + v^2\right) F(u,\, v).$$

Let us then define the filter $H(u,\, v)$ as

$$H(u,\, v) = -4\pi^2 \left(u^2 + v^2\right),$$

or, with respect to the centre of the frequency rectangle,

$$H(u,\, v) = -4\pi^2 \left((u - P/2)^2 + (v - Q/2)^2\right) = -4\pi D^2 (u,\, v).$$

Then, the Laplacian of an image $f(x,\, y)$ is given by

$$\nabla^2 f(x,\, y) = \mathscr{F}^{-1} \left\{H(u,\, v) F(u,\, v)\right\}.$$

Then for the enhanced image

$$g(x,\, y) = f(x,\, y) + c\nabla^2 f(x,\, y),$$

$c = -1$, so that in the frequency domain, this is equivalent to

$$G(u,\, v) = F(u,\, v) - H(u,\, v) F(u,\, v).$$

3.5 Selective Filtering

Bandreject and bandpass filters can be used to selectively filter specific frequency bands in an image. A bandreject filter is defined as

$$H(u,\, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D(u,\, v) \leq D_0 + \frac{W}{2}, \\ 1 & \text{otherwise,} \end{cases}$$

where D_0 is the centre frequency and W is the width of the band. A bandpass filter is defined as the complement of the bandreject filter:

$$H_{\mathrm{BP}}(u,\, v) = 1 - H_{\mathrm{BR}}(u,\, v).$$

3.5.1 Butterworth Bandreject Filters

A Butterworth bandreject filter is defined as

$$H(u,\, v) = \frac{1}{1 + \left(\frac{D(u,\, v) W}{D^2(u,\, v) - D_0^2}\right)^{2n}},$$

where n is the order of the filter.

3.5.2 Gaussian Bandreject Filters

A Gaussian bandreject filter is defined as

$$H(u,\, v) = 1 - \exp\left(-\left(\frac{D^2(u,\, v) - D_0^2}{D(u,\, v)\, W}\right)^2\right),$$

where D_0 is the centre frequency and W is the width of the band.

3.5.3 Notch Filters

Notch filters are used to remove frequencies in a predefined neighbourhood of the frequency rectangle (rather than being centred at the origin). Zero-phase-shift filters must be symmetric about the origin, so a notch filter transfer function with centre frequencies $(u_0,\, v_0)$ must have a corresponding notch filter transfer function with centre frequencies $(-u_0,\, -v_0)$. The transfer function of a notch filter is constructed by multiplying the transfer functions of two highpass filter transfer functions whose centres have been translated to the centres of the notches. The general form for a notch filter with Q notches is

$$H(u,\, v) = \prod_{k=1}^{Q} H_k(u,\, v)\, H_{-k}(u,\, v),$$

where $H_k(u,\, v)$ and $H_{-k}(u,\, v)$ are the transfer functions of the highpass filters centred at the kth notch and its negative, respectively.

4 Colour

Colour is a perceptual property of light that is produced when white light is passed through a prism. The wavelengths of light that are produced comprise the **electromagnetic spectrum**, in which the human eye can discern wavelengths between 400 nm and 700 nm.

4.1 Colour Fundamentals

The human eye contains three types of cones that can be roughly classified as being sensitive to three principal sensing regions:

- **Short-wavelength cones (S-cones)**: 2% of cones that are sensitive to blue light (435.8 nm). These are also the most sensitive cones, despite only making up 2% of the total.
- **Medium-wavelength cones (M-cones)**: 33% of cones that are sensitive to green light (546.1 nm).
- **Long-wavelength cones (L-cones)**: 64% of cones that are sensitive to red light (700 nm).

These **primary** colours can be combined to produce the **secondary** colours of light:

- **Cyan**: A combination of blue and green light.
- **Magenta**: A combination of blue and red light.

- **Yellow**: A combination of green and red light.

Mixing the primary colours of light produces white light. The primary colours of **pigment** are the secondary colours of light, and mixing these produces black pigment. The primary colours are also referred to as the **additive primaries**, while the secondary colours are referred to as the **subtractive primaries**.

4.1.1 Characterising Colour

Colour can be characterised by three attributes:

- **Brightness**: The intensity of the colour.
- **Hue**: The dominant wavelength of light.
- **Saturation**: The relative purity of the colour, or the amount of white light mixed with a hue. Hence, the degree of saturation is inversely proportional to the amount of white light mixed with a hue.

Hue and saturation can also be combined to form the **chromaticity** of a colour.

4.2 Colour Models

Colour models, or colour spaces, are coordinate system for representing colours mathematically. Some common colour spaces include:

- **RGB**: The most common colour space, which represents colours as a combination of red, green, and blue intensities. RGB is used in displays and cameras.
- **CMY(K)**: A subtractive colour model that represents colours as a combination of cyan, magenta, and yellow intensities, with an optional black intensity. CMYK is used in colour printing.
- **HSI**: A cylindrical colour space that represents the hue, saturation, and intensity of a colour. This is also known as HSV (value) or HSL (lightness), where the intensity may be scaled to a different range.

4.2.1 RGB Colour Model

The RGB colour model can be represented as a unit cube, with a corner at the origin, with the primary colours of red, green, and blue at the vertices:

- (1, 0, 0): Red.
- (0, 1, 0): Green.
- (0, 0, 1): Blue.

The colour black is at the origin, while the colour white is at the opposite corner. Grayscale colours lie along the diagonal of the cube which intersects these two points. Note that in this model, colour values are normalised to the range $[0, 1]$.

4.2.2 CMY(K) Colour Model

The CMY(K) colour model exists on the same cube as the RGB colour model, but with the primary colours of cyan, magenta, and yellow at the vertices:

- (0, 1, 1): Cyan.
- (1, 0, 1): Magenta.
- (1, 1, 0): Yellow.

Mathematically, CMYK colours are the complement of RGB colours:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

4.2.3 HSI Colour Model

The HSI colour model is a cylindrical colour space that represents colours as a combination of hue, saturation, and intensity. The hue value is the angle around this cylinder, with red at 0°, green at 120°, and blue at 240°. The intensity value is the distance along the intensity axis in the RGB cube, while the saturation value is the distance from the intensity axis.

$$\theta = \arccos\left(\frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right), \tag{1}$$

$$H = \begin{cases} \frac{\theta}{360^\circ} & \text{if } B \leqslant G, \\ 1 - \frac{\theta}{360^\circ} & \text{if } B > G, \end{cases} \tag{2}$$

$$S = 1 - \frac{3}{R+G+B}\min(R,\, G,\, B), \tag{3}$$

$$I = \frac{R+G+B}{3}. \tag{4}$$

To convert from HSI to RGB, we can use the following equations. In the RG sector ($0^\circ \leqslant H \leqslant 120^\circ$):

$$\begin{aligned} B &= I(1-S), \\ R &= I\left(1 + \frac{S\cos H}{\cos(60^\circ - H)}\right), \\ G &= 3I - (R+B). \end{aligned}$$

In the GB sector ($120^\circ \leqslant H \leqslant 240^\circ$):

$$\begin{aligned} R &= I(1-S), \\ G &= I\left(1 + \frac{S\cos(H - 120^\circ)}{\cos(60^\circ - (H - 120^\circ))}\right), \\ B &= 3I - (R+G). \end{aligned}$$

In the BR sector ($240^\circ \leqslant H \leqslant 360^\circ$):

$$\begin{aligned} G &= I\left(1 - S\right), \\ B &= I\left(1 + \frac{S\cos\left(H - 240^\circ\right)}{\cos\left(60^\circ - \left(H - 240^\circ\right)\right)}\right), \\ R &= 3I - \left(G + B\right). \end{aligned}$$

4.3 Colour Image Acquisition

Colour images are acquired using monochromatic charge-coupled devices (CCDs) that are sensitive to red, green, and blue light. These devices are arranged in a Bayer filter mosaic, which consists of a 2×2 grid of sensors with one red, one blue, and two green sensors.

$$\begin{bmatrix} G & R \\ B & G \end{bmatrix}$$

Here two green sensors are used to improve the resolution of the green channel, which is the most sensitive to the human eye. When demosaicking the image, the colour channels are extracted from the mosaic, and missing entries are interpolated using the values of neighbouring pixels in that channel.

$$\begin{bmatrix} G_{11} & R_{12} \\ B_{21} & G_{22} \end{bmatrix} \to \begin{bmatrix} - & R_{12} \\ - & - \end{bmatrix} + \begin{bmatrix} G_{11} & - \\ - & G_{22} \end{bmatrix} + \begin{bmatrix} - & - \\ B_{21} & - \end{bmatrix}$$

4.3.1 Pixel Depth

The pixel depth of an image is the number of bits used to represent the colour of each pixel. The pixel depth determines the total number of colours that can be represented in an image. For an image with n bits per channel, the total number of colours is given by 2^{3n}. For example, an image with 8 bits per channel has a pixel depth of 24 bits and can represent $2^{24} = 16\,777\,216$ unique colours.

4.4 Colour Image Processing

Colour images can be processed using the same techniques as grayscale images, but with the additional consideration of the colour channels. Transformations can either be applied to each channel independently or to individual channels, such as the hue, saturation, and intensity channels in the HSI colour space. We can define colour image transformations in the same way as for grayscale images:

$$g\left(x,\, y\right) = T\left(f\left(x,\, y\right)\right),$$

where $f\left(x,\, y\right)$ is the input image, $g\left(x,\, y\right)$ is the output image, and T is the transformation function.

4.4.1 Colour Intensity Transformations

Intensity transforms take the form:

$$g\left(x,\, y\right) = kf\left(x,\, y\right),$$

- RGB: $s_i = kr_i$ for $i = 1, 2, 3$.
- HSI: $s_1 = r_1$, $s_2 = r_2$, and $s_3 = kr_3$.
- CMY: $s_i = kr_i + (1 - k)$ for $i = 1, 2, 3$.

4.4.2 Colour Complements

Colour complements are hues directly opposite each other on the colour circle. Complements are analogous to negative images in grayscale images and can be useful for enhancing detail in dark regions.

4.4.3 Colour Slicing

Colour slicing is a technique used to highlight specific colours in an image to separate objects from surroundings.

4.4.4 Colour Tonal Transformations

Colour tonal transformations are used to adjust the tonal range of an image. These transformations can be used to boost contrast in an image using power-law transformations.

4.4.5 Colour Balancing (White Balancing)

An image is colour imbalanced when a known white point in the image does not contain equal RGB or CMY components. Colour balancing is used to correct these imbalances by adjusting the channels to make such points truly white.

4.4.6 Colour Histogram Equalisation

Histogram equalisation can also be applied to the intensity channel of HSI images to improve the contrast of the image without altering the hue or saturation.

4.4.7 Colour Image Smoothing and Sharpening

Colour images can be smoothed or sharpened using the same techniques as grayscale images, by applying a filter to each channel independently. For example, an image can be smoothed by considering it's K average neighbours:

$$\mathbf{y}_{ij} = \frac{1}{K} \sum_{(s,\, t) \in S} \mathbf{x}_{st},$$

where S is the set of K neighbours of pixel $(i,\ j)$. Similarly, an image can be sharpened by applying a Laplacian filter to each channel independently. The Laplacian of a colour image $\mathbf{x}$ is simply

$$\nabla^2 \mathbf{x} = \begin{bmatrix} \nabla^2 x_1 \\ \nabla^2 x_2 \\ \nabla^2 x_3 \end{bmatrix}$$

4.4.8 Colour Image Segmentation

Colour image segmentation is the process of partitioning an image into regions based on colour. For HSI images, segmentation is performed on the hue channel, and the saturation channel is used to further isolate regions in the hue image. For RGB images, we can use distance metrics to measure if nearby pixels are similar to a desired colour $\mathbf{a}$.

5 Image Restoration

Image restoration aims to recover a degraded image using knowledge of the degradation process. Some techniques are best handled in the spatial domain, such as when removing additive noise, while others are more suited to the frequency domain, like when removing blurring. The process of image degradation and restoration can be modelled as a linear, position-invariant system:

$$g(x,\, y) = f(x,\, y) * h(x,\, y) + n(x,\, y) \iff G(u,\, v) = F(u,\, v)\, H(u,\, v) + N(u,\, v)\,,$$

where $f(x,\, y)$ is the original image, $g(x,\, y)$ is the degraded and noisy image, $h(x,\, y)$ is the degradation function, and $n(x,\, y)$ is additive noise.

5.1 Noise Models

The principal source of noise in images is during the acquisition and/or transmission of the image. This includes the following factors that affect the performance of imaging sensors:

- Environmental conditions during acquisition, such as light levels and sensor temperature.
- Quality of the sensing elements.

5.1.1 Noise Probability Density Functions in the Spatial Domain

When modelling noise as a probability distribution, we will assume that noise is independent of spatial coordinates, except when considering spatially periodic noise, and that noise is uncorrelated with respect to the image content, that is there is no correlation between pixel values and the values of the noise components. Some common noise probability density functions are shown below:

- **Uniform Noise**: A model that assumes noise is uniformly distributed over a range of pixel intensities.
$$p(z) = \begin{cases} \dfrac{1}{b-a} & \text{if } a \le z \le b, \\ 0 & \text{otherwise.} \end{cases}$$
This model has a mean of $(a+b)/2$ and a variance of $(b-a)^2/12$.
- **Bipolar Impulse Noise**: (Salt-and-Pepper Noise) A model adds noise to random pixel intensities.
$$p(z) = p_0 \delta(z - z_0) + p_1 \delta(z - z_1)\,,$$
where p_0 and p_1 are the probabilities of the noise and z_0 and z_1 are the noise values. If $p_0 = p_1$, the noise is unipolar.

- **Gaussian Noise**: A model that assumes noise is normally distributed with a mean μ and variance σ^2.
$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right). \tag{1}$$
where μ is the mean, and σ^2 is the variance.

- **Rayleigh Noise**: A model that assumes noise is Rayleigh distributed with a scale parameter σ.
$$p(z) = \begin{cases} \frac{z-a}{\sigma^2} \exp\left(-\frac{(z-a)^2}{2\sigma^2}\right) & \text{if } z \geq a, \\ 0 & \text{otherwise.} \end{cases}$$
where $a > 0$ is the minimum value of the noise. This model has a mean of $a + \sigma\sqrt{\pi/2}$ and a variance of $\frac{4-\pi}{2}\sigma^2$.

- **Erlang Noise**: A model that assumes noise is Erlang distributed with a shape parameter $k \in \mathbb{N}$ and a scale parameter λ.
$$p(z) = \begin{cases} \frac{\lambda^k z^{k-1}}{(k-1)!} e^{-\lambda z} & \text{if } z \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$
This model has a mean of k/λ and a variance of k/λ^2.

- **Exponential Noise**: A model that assumes noise is exponentially distributed with a scale parameter λ.
$$p(z) = \begin{cases} \lambda e^{-\lambda z} & \text{if } z \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$
This model has a mean of $1/\lambda$ and a variance of $1/\lambda^2$.

5.1.2 Periodic Noise in the Frequency Domain

Periodic noise typically arises from electrical or electromechanical interference during image acquisition. This noise is characterised by periodic patterns in the frequency domain, which can be greatly reduced using frequency domain filtering techniques.

5.2 Filtering Noise in the Spatial Domain

When only additive noise is present, we can use spatial filtering techniques to remove noise from an image. Here the corrupted image has the form

$$g(x,\, y) = f(x,\, y) + \eta(x,\, y) \iff G(u,\, v) = F(u,\, v) + N(u,\, v).$$

Some common spatial filtering techniques are shown below:

- **Mean Filters**:

- **Arithmetic Mean Filter**:

$$\hat{f}(x,\, y) = \frac{1}{mn} \sum_{(s,\, t) \in S_{xy}} g(s,\, t),$$

for some neighbourhood S_{xy}.

- **Geometric Mean Filter**:

$$\hat{f}(x,\, y) = \left[\prod_{(s,\, t) \in S_{xy}} g(s,\, t) \right]^{1/mn}.$$

for some neighbourhood S_{xy}.

- **Harmonic Mean Filter**:

$$\hat{f}(x,\, y) = mn \left(\sum_{(s,\, t) \in S_{xy}} \frac{1}{g(s,\, t)} \right)^{-1}.$$

for some neighbourhood S_{xy}. This filter works well for salt (Gaussian) noise, but poorly for pepper (impulse) noise.

- **Contraharmonic Mean Filter**:

$$\hat{f}(x,\, y) = \frac{\sum_{(s,\, t) \in S_{xy}} g(s,\, t)^{Q+1}}{\sum_{(s,\, t) \in S_{xy}} g(s,\, t)^{Q}}.$$

for some neighbourhood S_{xy}. Here Q is the order of the filter. The filter eliminates pepper noise for $Q > 0$ and salt noise for $Q < 0$. When $Q = 0$, the filter is equivalent to the arithmetic mean filter, and when $Q = -1$, it is equivalent to the harmonic mean filter.

- **Order-Statistic Filters**:

  - **Median Filter**:

$$\hat{f}(x,\, y) = \text{median} \left\{ g(s,\, t) : (s,\, t) \in S_{xy} \right\}.$$

for some neighbourhood S_{xy}. This filter is effective at removing bipolar and unipolar impulse noise.

  - **Max Filter**:

$$\hat{f}(x,\, y) = \max \left\{ g(s,\, t) : (s,\, t) \in S_{xy} \right\}.$$

for some neighbourhood S_{xy}.

  - **Min Filter**:

$$\hat{f}(x,\, y) = \min \left\{ g(s,\, t) : (s,\, t) \in S_{xy} \right\}.$$

for some neighbourhood S_{xy}.

– **Midpoint Filter**:

$$\hat{f}(x,\, y) = \frac{1}{2}\left(\max\left\{g(s,\, t) : (s,\, t) \in S_{xy}\right\} + \min\left\{g(s,\, t) : (s,\, t) \in S_{xy}\right\}\right).$$

for some neighbourhood S_{xy}. This filter is effective at removing randomly distributed noise such as Gaussian or uniform noise.

– **Alpha-Trimmed Mean Filter**:

$$\hat{f}(x,\, y) = \frac{1}{mn - d} \sum_{(s,\, t) \in S_{xy}} g_r(s,\, t), \tag{1}$$

where g_r represents the image g in which the $d/2$ lowest and $d/2$ highest intensity values in the neighbourhood S_{xy} have been removed. This filter is effective when multiple types of noise are present in an image. When $d = 0$, the filter is equivalent to the arithmetic mean filter and when $d = mn - 1$, the filter is equivalent to the median filter.

5.3 Filtering Noise in the Frequency Domain

When noise is present in the frequency domain, we can use the frequency filtering techniques discussed in the previous section to remove noise from an image. This includes using bandreject, bandpass, and notch filters to remove noise in specific frequency bands or in specific neighbourhoods of the frequency domain.

5.4 Degradation Function

When we also want to remove degradation from an image, we must first estimate the degradation function using one of the following methods:

- **Observation**: The degradation is assumed to be linear and position-invariant. By looking at a small region of the image $g_s(x,\, y)$ containing sample structures where the signal content is strong (i.e., high contrast), we can process this sub-image to find the best result $\hat{f}_s(x,\, y)$. This lets us estimate the degradation function as

$$H_s(u,\, v) = \frac{G_s(u,\, v)}{\hat{F}_s(u,\, v)},$$

where we assume the effect of noise is negligible in this region. This allows us to then deduce the degradation function for the entire image.

- **Experimentation**: When equipment similar to what was used to acquire the image is available, we can find system settings that reproduce the most similar degradation and obtain an impulse response for the degradation function by imaging an impulse (point of light).

$$H(u,\, v) = \frac{G(u,\, v)}{A}.$$

where A is the strength of the impulse.

- **Mathematical Modelling**: In some cases, we can model the degradation function mathematically. For example, Hufnagel and Stanley proposed the following model based on the physical characteristics of atmospheric turbulence:

$$H(u,\, v) = \exp\left(-k\left(u^2 + v^2\right)^{5/6}\right),$$

where k is the turbulence constant. We can also derive models based on the type of degradation present in the image, such as motion blur.

$$g(x,\, y) = \int_0^T f\left(x - x_0(t),\, y - y_0(t)\right) \mathrm{d}t$$
$$G(u,\, v) = F(u,\, v) \int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))}\, \mathrm{d}t.$$

where $x_0(t)$ and $y_0(t)$ are the motion paths in the x and y directions, respectively. The degradation function then becomes

$$H(u,\, v) = \frac{G(u,\, v)}{F(u,\, v)} = \int_0^T e^{-j2\pi(ux_0(t) + vy_0(t))}\, \mathrm{d}t.$$

5.5 Inverse Filtering

When restoring an image $g(x,\, y)$ that has been degraded by some process $h(x,\, y)$, we can obtain an estimate for the reconstructed image in the frequency domain using direct inverse filtering:

$$\hat{F}(u,\, v) = \frac{G(u,\, v)}{H(u,\, v)}.$$

If we include noise in the image $g(x,\, y)$, then it complicates this result further:

$$\hat{F}(u,\, v) = \frac{F(u,\, v)\, H(u,\, v) + N(u,\, v)}{H(u,\, v)} = F(u,\, v) + \frac{N(u,\, v)}{H(u,\, v)}.$$

This result tells us that it is not possible to recover the original undegraded image exactly because the noise $\eta(x,\, y)$ is not known. Additionally, when H is zero or has small values, it is possible for the noise to be amplified, leading to a poor estimate. One approach to mitigate this issue is to filter frequencies to values near the origin.

5.5.1 Minimum Mean Square Error Filtering

The minimum mean square error (Wiener) filter is a statistical approach to image restoration that minimises the mean square error between the estimated and original image. It incorporates both the degradation function and statistical characteristics of the noise in the restoration process by considering the image and noise as random variables. When using the Wiener filter, the following is assumed:

- The noise and image are uncorrelated.
- The noise or image have zero mean.

- The intensity levels in the estimate are a linear function of the levels in the degraded image.

Then, the minimum of the error function e defined:

$$e^2 = \mathrm{E}\left[\left(f - \widehat{f}\right)^2\right]$$

is given by

$$\widehat{F}(u,\, v) = \left[\frac{1}{H(u,\, v)} \frac{|H(u,\, v)|^2}{|H(u,\, v)|^2 + S_\eta(u,\, v)/S_f(u,\, v)}\right] G(u,\, v),$$

where $S_\eta(u,\, v)$ and $S_f(u,\, v)$ are the power spectra of the noise and undegraded image, respectively:

$$S_\eta(u,\, v) = |N(u,\, v)|^2, \qquad S_f(u,\, v) = |F(u,\, v)|^2.$$

When these spectrums are unknown or difficult to estimate, we can use the following approximation:

$$\widehat{F}(u,\, v) = \left[\frac{1}{H(u,\, v)} \frac{|H(u,\, v)|^2}{|H(u,\, v)|^2 + K}\right] G(u,\, v),$$

where K is a specified constant.

5.5.2 Geometric Mean Filter

The geometric mean filter is a generalisation of the Wiener filter that has the form:

$$\widehat{F}(u,\, v) = \left[\frac{H^*(u,\, v)}{|H(u,\, v)|^2}\right]^\alpha \left[\frac{H^*(u,\, v)}{|H(u,\, v)|^2 + \beta S_\eta(u,\, v)/S_f(u,\, v)}\right]^{1-\alpha} G(u,\, v),$$

where α and β are positive real constants.

- When $\alpha = 1$, we recover the inverse filter.
- When $\alpha = 0$, we recover the parametric Wiener filter. If $\beta = 1$, we recover the standard Wiener filter.
- When $\alpha = 1/2$, we recover the geometric mean filter.
- When $\alpha = 1/2$ and $\beta = 1$, we obtain the spectrum equalisation filter.

5.5.3 Constrained Least Squares Filtering

When the power spectra of the noise and undegraded image are unknown, we can use the approximations shown above to estimate the restored image, but a constant value for the ratio of the noise and image power spectra is not always suitable. Using the convolution definition of the degraded and noised image, we can form the following system of equations:

$$\mathbf{g} = \mathbf{H}\mathbf{f} + \boldsymbol{\eta},$$

where we suppose $g(x,\, y)$ is of size $M \times N$, so that $\mathbf{g}$, $\mathbf{f}$ and $\boldsymbol{\eta}$ have dimensions $MN \times 1$, and $\mathbf{H}$ has dimensions $MN \times MN$. Notice that this leads to an extremely large system of equations that

is computationally expensive to solve. This is complicated further by the fact that $\mathbf{H}$ is highly sensitive to noise. Thus, we will consider the method of constrained least squares optimisation, where we minimise the criterion function C:

$$\begin{aligned} \text{minimise} \quad & C = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\boldsymbol{\nabla}^2 f\left(x,\, y\right)\right)^2 \\ \text{subject to} \quad & \left\|\mathbf{g} - \mathbf{H}\mathbf{f}\right\|^2 = \left\|\boldsymbol{\eta}\right\|^2. \end{aligned}$$

Here the optimality of restoration is based on a measure of smoothness which we use the Laplacian operator to measure. The frequency domain solution to this problem is given by the expression:

$$\widehat{F}\left(u,\, v\right) = \left[\frac{H^*\left(u,\, v\right)}{\left|H\left(u,\, v\right)\right|^2 + \gamma \left|P\left(u,\, v\right)\right|^2}\right] G\left(u,\, v\right),$$

where γ is a parameter that must be adjusted so that the above constraint is satisfied. $P\left(u,\, v\right)$ is known as a Laplacian kernel, and is the Fourier transform of the function:

$$p\left(x,\, y\right) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Note that P and H must be of the same size. If H has dimensions $M \times N$, then p must be embedded in the centre of an $M \times N$ array of zeros. In order to preserve the even symmetry of p, M and N must both be even integers. To compute the parameter γ iteratively, we can use the following algorithm which aims to reduce the residual:

$$\left\|\mathbf{r}\right\|^2 = \left\|\boldsymbol{\eta}\right\|^2 \pm \alpha,$$

where α is an accuracy factor. It can be shown that this is a monotonically increasing function of γ, meaning we can use a simple binary search algorithm to find the optimal value of γ:

1. Specify an initial value of γ.
2. Compute the residual $\left\|\mathbf{r}\right\|^2 = \left\|\mathbf{g} - \mathbf{H}\mathbf{f}\right\|^2$.
   - If $\left\|\mathbf{r}\right\|^2 < \left\|\boldsymbol{\eta}\right\|^2 - \alpha$, increase γ and repeat step 2.
   - If $\left\|\mathbf{r}\right\|^2 > \left\|\boldsymbol{\eta}\right\|^2 + \alpha$, decrease γ and repeat step 2.
   - Otherwise, the optimal value of γ has been found.

6 Image Features

6.1 Feature Detection

Consider the problem of creating a panorama from multiple images. Here we want to detect and match features between images so that we can align them correctly. There are 5 characteristics for a good feature:

- **Locality**: The feature should be localised to a small region of the image, so that it is robust to occlusion and clutter.
- **Distinctiveness**: The feature should differentiate a large database of objects.
- **Quantity**: The feature should be present in large quantities in the image.
- **Efficiency**: The feature should be computationally efficient to compute.
- **Generality**: The feature should be applicable to a wide range of images.

Such features should be invariant to:

- **Geometric Transformations**:
  - Rotation
  - Similarity transformations (rotation and uniform scaling)
  - Affine transformations (shear and stretching in one direction)
- **Photometric Transformations**:
  - Affine intensity changes ($I \to aI + b$)

6.2 Edge Detection

While features are useful for matching images, edges are far more plentiful and carry important semantic associations. Boundaries of objects, which also correspond to occlusion events in 3D (i.e., when an object is in front of another), are often delineated by visible contours. Other kinds of edges correspond to shadow boundaries or crease edges, where surface orientation changes rapidly. We recognise edges in images by looking for depth discontinuities, texture boundaries, changes in material properties, and illumination changes. These points are called **edge points**.
When designing an edge detection algorithm, we define an edge as a location of rapid intensity or colour variation. This can be described mathematically through the gradient of the image:

$$\boldsymbol{\nabla} f = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix},$$

where the **gradient direction** is given by:

$$\theta = \arctan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right),$$

and the **edge strength** is given by the magnitude of the gradient:

$$\|\boldsymbol{\nabla} f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}.$$

We should note that this operation is sensitive to noise as the gradient is a high-pass filter in the frequency domain. To mitigate this, noise must be smoothed out prior to edge detection using a low-pass filter. It is important that this smoothing operation does not blur edges.

6.2.1 Gaussian Edge Detector

The Gaussian edge detector is a simple edge detection algorithm that uses the derivative of the Gaussian function to detect edges. Here, the Gaussian function $h(x,\, y)$ acts as the smoothing operator, and the gradient operator is used to detect edges. This operation is summarised by the following equation:

$$g(x,\, y) = \boldsymbol{\nabla}(h * f) = (\boldsymbol{\nabla} h) * f.$$

Once we have found g, we can threshold the gradient magnitude to obtain the edge map.

6.2.2 Sobel Edge Detector

The Sobel edge detector uses a Sobel filter which is the product of a vertical smoothing operator and a horizontal second-order derivative operator:

$$S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}.$$

6.2.3 Edge Thinning

The Sobel filter is effective at detecting edges, but it may produce edges that are thicker than desired. To thin these edges, we can use **non-maxima suppression**. This is done by considering the gradient direction and suppressing all non-maximal gradient values in the direction tangent to the gradient. However, this may result in disconnected edges, as this process is applied locally.

6.2.4 Edge Linking

If a low threshold is used, the edge map may contain many weak edges, where edges are not continuous. To address this, we can use **edge linking** to connect these weak edges. This is done by considering the tangent direction of the gradient and following the direction to predict the next edge point. This process is repeated until the edge is complete. One common algorithm for edge linking is the **hysteresis thresholding** algorithm, where we start with a high threshold to obtain edge curves and lower it until we have a complete edge map.

6.2.5 Canny Edge Detector

The Canny edge detector is a multi-step edge detection algorithm that aims to find the optimal edge map by considering the following criteria:

- **Good Detection**: The edge detector minimises the number of false positives and false negatives.
- **Good Localisation**: The edge detector must be as close as possible to the true edge.
- **Single Response**: The edge detector should only return one point for each true edge point.

The canny edge detector is composed of the following steps:

1. Filter an image with the derivative of a Gaussian function.

2. Find the gradient magnitude and direction.
3. Apply non-maxima suppression to thin the edges.
4. Apply hysteresis thresholding to remove weak edges.

6.3 Corner Detection

Corners are important features in images that have a gradient in two or more dominant directions. Corners are both **repeatable** and **distinctive**, making them ideal for image matching.

6.3.1 Harris Corner Detector

The Harris corner detector is a popular corner detection algorithm that uses the auto correlation of the image gradient in two directions to detect corners. Consider the change in intensity when an image is shifted by $(s,\ t)$:

$$E\left(x,\ y\right) = \sum_{(s,\ t)\in W} w\left(s,\ t\right)\left[f\left(x+s,\ y+t\right) - f\left(x,\ y\right)\right]^2,$$

where W is a window around the pixel $(x,\ y)$, $w\left(s,\ t\right)$ is a window function such as a uniform weight function or the Gaussian function, and $f\left(x,\ y\right)$ is the image intensity at $(x,\ y)$. The Harris corner detector aims to maximise this function to detect corners. For small shifts, we can use the Taylor series expansion to approximate the change in intensity, leading to the following expression:

$$E\left(x,\ y\right) \approx \mathbf{v}^\top \mathbf{M} \mathbf{v},$$

where $\mathbf{v} = \left[u,\ v\right]^\top$ is the shift vector, and $\mathbf{M}$ is the auto-correlation matrix of the image gradient:

$$\mathbf{M} = \sum_{(s,\ t)\in W} w\left(s,\ t\right) \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix},$$

If we consider the eigenvalues of $\mathbf{M}$, we can visualise an ellipse that represents the change in intensity.

- If $\lambda_1 \gg \lambda_2$ then the point can be classified as an edge. The larger eigenvalue λ_1 corresponds to the direction of the maximum change in intensity. The radius of the ellipse along the minor axis is equal to $\lambda_1^{-1/2}$, whereas the radius of the ellipse along the major axis is equal to $\lambda_2^{-1/2}$.
- When λ_1 and λ_2 are both small, the point is likely to be a flat region.
- If λ_1 and λ_2 are large, the point is likely to be a corner and E increases rapidly in all directions.

We can also define a measure of **corner response** as follows:

$$R = \det \mathbf{M} - k\left(\operatorname{Tr} \mathbf{M}\right)^2,$$

where a high value of R determines how strong the corner is. Here k is an empirical constant that ranges between 0.04 and 0.06. Similar to the above list, we can classify corners based on the value of R:

- When $R < 0$, the point is an edge.
- When $|R|$ is small, the region is flat.
- When $R > 0$, the point is a corner.

Therefore we can summarise the Harris corner detection algorithm with the following steps:

1. Compute the Gaussian derivatives of the image.
2. Compute the auto-correlation matrix **M** around each pixel.
3. Compute the corner response R for each pixel.
4. Threshold the corner response to obtain the corner map.
5. Apply non-maxima suppression to thin the corners.

One key advantage of the Harris corner detector is that it is rotation invariant, as the eigenvalues of **M** are invariant to rotations in the image. This means that two corners that are rotated will produce the same eigenvalues. However, the Harris corner detector is not scale invariant, as zooming into a corner will cause the corner to become an edge.

6.3.2 Scale Invariant Feature Transform (SIFT)

The SIFT algorithm is a feature detection algorithm that is both scale and rotation invariant. The SIFT algorithm is composed of the following steps:

1. **Gaussian Scale-Space**: The image is convolved with Gaussian filters at different variances to create a scale space.
2. **Difference of Gaussians (DoG)**: The difference between two adjacent scales in the scale space is computed.
3. **Non-Maxima Suppression**: Local maxima and minima are detected in the DoG space.

To find the interest points, the algorithm then uses thresholding to remove weak candidates, keeping only the points for which the Laplacian is extremal.

7 Image Segmentation

Image segmentation is the process of partitioning an image into multiple image segments. Edges and corners are useful for detecting features in images, but often boundaries of interest are **fragmented**. In such cases we can use colour and texture information to segment the image. We can use two approaches to segment images:

- **Region-Based Segmentation**: This approach groups pixels based on their similarity. Here we can use unsupervised learning techniques such as clustering to segment the image.
- **Object-Based Segmentation**: This approach separates an image into objects that were learned from a supervised learning process.

7.1 Segmentation by Thresholding

Thresholding is a simple image segmentation technique that separates pixels into two classes based on their intensity values. Regions with uniform intensities have strong peaks in their intensity histograms, but in general, a good threshold only requires peaks separated by valleys.

7.1.1 Otsu's Method

Otsu's method is a technique that finds the optimal threshold by minimising the intra-class variance of the image, or by maximising the inter-class variance. Given the threshold intensity value $T = k$, the algorithm is as follows:

1. Compute the normalised histogram of the image.
2. Compute the class probabilities, $P_1(k)$ and $P_2(k)$, as
$$P_1(k) = \sum_{i=0}^{k} p(i) \quad \text{and} \quad P_2(k) = \sum_{i=k+1}^{L-1} p(i),$$
where $p(i)$ is the ith histogram component (i.e., the pdf of the i intensity value). Note $P_1 + P_2 = 1$.
3. Compute the class means, m_1 and m_2, as
$$m_1 = \sum_{i=0}^{k} \frac{ip(i)}{P_1} \quad \text{and} \quad m_2 = \sum_{i=k+1}^{L-1} \frac{ip(i)}{P_2}.$$
4. Compute the global mean, m_G, as
$$m_G = \sum_{i=0}^{L-1} ip(i) = P_1 m_1 + P_2 m_2.$$
5. Compute the global variance, σ_G^2, as
$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p(i) = P_1\sigma_1^2 + P_2\sigma_2^2 + P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2.$$
and define the intra-class (within-class) variance as
$$\sigma_w^2 = P_1\sigma_1^2 + P_2\sigma_2^2.$$
and the inter-class (between-class) variance as
$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1P_2(m_1 - m_2)^2 = \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)}.$$
where m is the cumulative mean (average intensity) up to the threshold k, defined as
$$m = \sum_{i=0}^{k} ip(i).$$

6. The optimal threshold is the value of k that maximises σ_B^2:

$$k = \arg\max_{k^*} \sigma_B^2(k^*).$$

For noisy images, it may be beneficial to smooth the histogram before applying Otsu's method, however this is not effective for images with a very small foreground region. For local thresholds, we can compute the gradient using the Laplacian of the image and only apply Otsu's method to nearby pixels.

7.1.2 Multiple Thresholding

When there is more than one object in the same background, the histogram may become multimodal, and multiple thresholds may be required. Here we can assign pixels as follows:

- If $f(x,\, y) < T_1$, assign the pixel to class 1.
- If $T_1 \leq f(x,\, y) < T_2$, assign the pixel to class 2.
- If $T_2 \leq f(x,\, y) < T_3$, assign the pixel to class 3.

Otsu's method can be extended to 3 classes by searching for values k_1 and k_2 that maximise the inter-class variance:

$$(k_1,\, k_2) = \arg\max_{0<k_1<k_2<L-1} \sigma_B^2(k_1,\, k_2)$$

This can be found using a simple grid search algorithm:

1. Select the first value of k_1 ($k_1 = 1$).
2. Iterate k_2 over all values from $k_1 + 1$ to $L - 2$.
3. Increment k_1 by 1.
4. Calculate $\sigma_B^2(k_1,\, k_2)$.
5. Repeat Steps 2–4 until $k_1 = L - 3$.

7.2 Active Contours

Active contours, also known as snakes, are curves that can move within an image to fit to object boundaries. These curves are defined by a set of points, and are iteratively adjusted so as to:

- be near image positions with high gradients, and
- satisfy shape "preferences" or contour priors.

Active contours are comparable to the Hough transform,

- **Hough transform**:
  - Rigid model shape.
  - Single voting pass can detect multiple instances.

- **Active contours**:
  - Prior on shape types, but shape can be iteratively adjusted (deformations).
  - Requires initialisation near the object boundary.
  - One optimisation "pass" is required to fit a single contour.

7.2.1 Energy Function

Active contours are defined by an energy function that is minimised to obtain the optimal contour. The total energy (cost) of the current snake is defined as the sum of the internal energy and the external energy:

$$E = E_{\text{internal}} + E_{\text{external}}.$$

The internal energy encourages prior shape preferences, e.g., smoothness, elasticity, a particular known shape. The external energy encourages the contour to fit on places where image structures exist, e.g., edges. A good fit between the **current deformable contour** and the target shape in the image will yield a *low* energy value.
Given the gradient images g_x and g_y, the external energy at a point on the curve is defined as:

$$E_{\text{external}}(\mathbf{v}) = -\left(|g_x(\mathbf{v})|^2 + |g_y(\mathbf{v})|^2\right),$$

and the external energy for the entire curve is defined as:

$$E_{\text{external}} = -\sum_{i=0}^{n-1}\left(|g_x(\mathbf{v}_i)|^2 + |g_y(\mathbf{v}_i)|^2\right),$$

where $\mathbf{v}_i$ is the ith point on the curve. The internal energy at a point on the curve is defined as:

$$E_{\text{internal}}(\mathbf{v}) = \alpha\left\|\frac{\mathrm{d}\mathbf{v}'(s)}{\mathrm{d}s}\right\|^2 + \beta\left\|\frac{\mathrm{d}^2\mathbf{v}(s)}{\mathrm{d}s^2}\right\|^2,$$

where the first term represents the tension or elasticity of the curve, and the second term represents the stiffness or curvature of the curve. The internal energy for the entire curve is defined as:

$$E_{\text{internal}} = \sum_{i=0}^{n-1}\left[\alpha\left\|\frac{\mathrm{d}\mathbf{v}'(s)}{\mathrm{d}s}\right\|^2 + \beta\left\|\frac{\mathrm{d}^2\mathbf{v}(s)}{\mathrm{d}s^2}\right\|^2\right].$$

Note we can approximate these derivatives using finite differences:

$$E_{\text{internal}}(\mathbf{v}) = \alpha\|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 + \beta\|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

$$E_{\text{internal}} = \sum_{i=0}^{n-1}\left[\alpha\|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 + \beta\|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2\right].$$

To capture some smooth variation on a known shape, we can add a term that penalises deviation from that shape:

$$E_{\text{internal}} = \sum_{i=0}^{n-1}\left[\alpha\|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 + \beta\|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2\right] + \alpha\sum_{i=0}^{n-1}\|\mathbf{v}_i - \hat{\mathbf{v}}_i\|^2,$$

where $\hat{\mathbf{v}}_i$ are points on the known shape.

7.2.2 Energy Minimisation

We will consider two methods for energy minimisation:

- **Greedy Search**: Create a search windows around each point and move the point to a position where the energy function is minimised. This is repeated until the points stop moving or after a fixed number of iterations. This method does not guarantee convergence to the global minimum and requires a good initialisation.
- **Dynamic Programming**: This method uses a dynamic programming algorithm called the Viterbi algorithm to find the optimal path. This is useful for optimising 2D snakes.

7.2.3 Limitations

Active contours have several limitations:

- They may over-smooth boundaries
- They cannot follow topological changes of objects
- The snake does not see object boundaries unless it is close to them

To summarise, active contours are useful for:

- Segmentation of objects with smooth boundaries
- Tracking objects in a sequence of images, where the previous frame can be used as the initialisation for the next frame

7.3 k-Means Clustering

The k-means clustering algorithm is an unsupervised learning algorithm that partitions an image into k clusters. Intensities are assigned to the cluster with the nearest mean intensity. The algorithm is as follows:

1. Randomly initialise k cluster centres.
2. Assign each pixel to the nearest cluster centre.
3. Update the cluster centres by computing the mean intensity of the pixels in each cluster.
4. Repeat Steps 2 and 3 until convergence.

Here we can choose to cluster based on the individual colour channels and also pixel position to enforce spatial coherence. However, this may not be enough to distinguish all regions. Depending on the feature space, we can group pixels in different ways.

7.3.1 Pros and Cons

- **Pros**:
  - Simple and easy to implement.
  - Fast and efficient.
- **Cons**:
  - Requires the number of clusters k to be specified.
  - Sensitive to initial centres.
  - Sensitive to outliers.
  - Detects spherical clusters (due to the Euclidean distance metric).
  - Assumes mean can be computed.

7.3.2 Mean-Shift Clustering

The mean-shift clustering algorithm is a non-parametric clustering algorithm that seeks modes or local maxima of density in the feature space. It works by iteratively shifting each data point to the mean of the data points within a certain radius. The algorithm is as follows:

1. Convert the image into tokens (via colour, gradients, texture, etc.)
2. Chose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same "peak" or mode.
5. The data these merged windows traversed are assigned to the same cluster.

The advantages of mean-shift clustering are that it:

- Does not assume shape on the clusters.
- Only requires a single parameter, the window size.
- Is a generic technique that can be applied to any feature space.
- Can be used to find multiple modes in the data.

Some disadvantages are that:

- We need to select the window size.
- The algorithm does not scale well with the dimension of the feature space.
- The algorithm is computationally expensive.

8 Image Compression

Image compression is a type of data compression applied to digital images to reduce their cost for storage or transmission. Consider the amount of memory required to store the data in a standard 1 hour long 4K movie:

$$\begin{aligned}
\text{Memory} &= \text{Resolution} \times \text{Colour Depth} \times \text{Frame Rate} \times \text{Time} \\
&= (3840 \times 2160)\,\text{px} \times 24\,\text{bit/px} \times 30\,\text{frame/s} \times (60 \times 60)\,\text{s} \\
&= 21\,499\,084\,800\,000\,\text{bit} \\
&= 2\,687\,385\,600\,000\,\text{byte} \\
&= 2\,624\,400\,000\,\text{K byte} \\
&= 2\,562\,890.625\,\text{M byte} \\
&\approx 2502.823\,\text{G byte} \\
&\approx 2.444\,\text{T byte}.
\end{aligned}$$

This is clearly an unfeasible amount of memory to store a single movie, and we are not even considering audio data. Therefore let us discuss some techniques that will allow us to reduce the amount of memory required to store such data.

8.1 Compression

Data compression is used to reduce the amount of data required to represent some information. Data compression can be used for both transmission and storage of data, and encompasses all kinds of digital information, including text, images, audio, and video.

Definition 8.1 (Compression Ratio)**.** The **compression ratio** is the ratio of the size of the compressed data to the size of the original data.

$$\text{Compression Ratio} = \frac{\text{Size of Uncompressed Data}}{\text{Size of Compressed Data}}.$$

Compression works by removing redundancy in data. For an image, this can be done by removing:

- **Coding Redundancy**: where an image contains more bits than necessary to represent information. For example, using an 8-bit image representation when only 4 intensity values are present in an image.
- **Spatial Redundancy**: when many neighbouring pixels have the same value.

There are two types of data compression:

- **Lossless Compression**: This type of compression reduces the size of the data without losing any information. As such, lossless compression is reversible.
- **Lossy Compression**: This type of compression reduces the size of the data by removing redundant information. This results in a loss of information, but is usually not perceptible. Due to this, lossy compression is irreversible.

8.2 Lossless Compression

8.2.1 Run-Length Encoding

Run-length encoding is a simple form of lossless data compression that compresses sequences of the same value. It represents runs of identical intensities as **run-length** pairs, where each pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity. For example, given a sequence of black and white pixels,

$$[wwwwbbbwww] \rightarrow [4w3b3w]\,.$$

8.2.2 Huffman Coding

If we consider a simple image with a limited number of colours, we can improve the compression ratio of run-length encoding by representing colour values with a unique code. For example, an image with 4 colours, white, red, blue, and green, can be represented by the following codes:

$$\begin{aligned} \text{White} &: 00, \\ \text{Red} &: 01, \\ \text{Blue} &: 10, \\ \text{Green} &: 11, \end{aligned}$$

rather than using 24 bits to represent each pixel. The Huffman coding algorithm is a lossless data compression algorithm that uses variable length codes to represent data, where the most frequent symbols are represented by shorter codes. This code is generated using a Huffman tree, which is known by both the encoder and decoder. The algorithm is as follows:

1. Determine the frequency of each symbol in the data.
2. Sort the symbols by frequency.
3. For the two least frequent symbols, create a new parent node with a frequency equal to the sum of the two symbols.
4. Repeat Step 3 until all symbols are connected to the root node.
5. Assign a 0 to the left branch and a 1 to the right branch.
6. Assign the code to each symbol by traversing the tree from the root to the symbol.

Huffman coding is used in the PNG image format, which is a lossless image compression format, the ZIP file format, and in some parts of the JPEG image format.

Definition 8.2 (Shannon Information Content)**.** The **Shannon information content** of a symbol x is the number of bits required to represent the symbol.

$$I\left(x\right) = -\log_2 p\left(x\right),$$

where $p\left(x\right)$ is the probability of the symbol x.

Definition 8.3 (Entropy)**.** The **entropy** of a message gives us the average number of bits needed to transmit each symbol from the message.

$$H(X) = \sum_{x \in X} p(x)\, I(x) = -\sum_{x \in X} p(x) \log_2 p(x),$$

where X is the set of all symbols in the source. It also measures the average information carried by each symbol in the message, and is a **lower bound** on the expected length of each symbol of a compressed message.

8.3 Lossy Compression

The aim of lossy compression is identify the most and least salient components of an image with respect to human perception. The least salient components are then removed to reduce the amount of data required to represent the image.

8.3.1 Quantisation

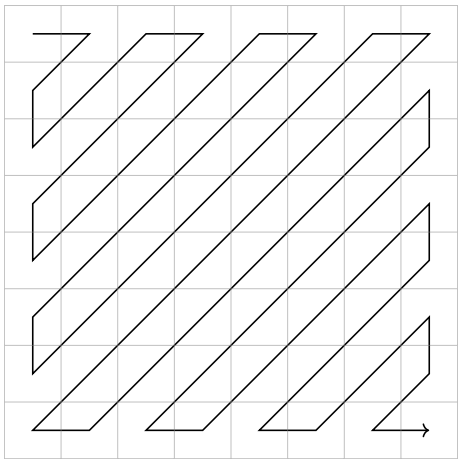
Quantisation is a lossy compression technique that reduces the number of bits required to represent an image by mapping input values from a large set to output values in a smaller set. One common method of quantisation is truncation, where the least significant bits of pixel values are removed, thereby removing the high frequency components in an image. For example, we can take an 8-bit image, which can represent $2^{3\times 8} = 16\,777\,216$ unique colours, and truncate each colour intensity to 4 bits, reducing the number of colours to $2^{3\times 4} = 4096$ unique colours.

8.3.2 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) is a lossy compression technique used in the JPEG image format. The DCT is a type of Fourier transform that converts a signal into a sum of cosine functions of different frequencies (both horizontally and vertically). The idea is to remove components with negligible coefficients, as they do not contribute significantly to the image. In practice, the image is divided into 8×8 pixel blocks, and 64 cosine bases are used to represent every possible 8×8 block. This process is described below:

1. Divide the image into 8×8 blocks.
2. Compute the DCT of each block.
3. For each block, divide the coefficients by a pre-defined quantisation matrix, rounding values to the nearest integer.
4. Store the quantised coefficients in a zig-zag order, starting from the top-left corner.

When storing these quantised coefficients, we can use a combination of Huffman coding, for the non-zero coefficients, and run-length encoding, for all remaining coefficients. We use a special **end-of-block** symbol to indicate the end of the Huffman block, after which we only expect zero coefficients. An example of the zig-zag order is shown below for an 8×8 block:



To reconstruct the image, we can reverse the process:

1. Reconstruct the quantised coefficients into an 8×8 matrix.
2. Multiply the coefficients by the quantisation matrix.
3. Compute the inverse DCT of the coefficients.
4. Repeat for each block.

9 Deep Learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning methods discover intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep learning is used in a variety of applications, including computer vision, speech recognition, natural language processing, and audio recognition.

9.1 Supervised Learning

Supervised learning is the machine learning task of learning a function that maps an input to an output, based on example input-output pairs. It infers a function from labelled training data consisting of a set of training examples. In the training phase, the model is trained to minimise the error between the predicted output and the true output, called the **loss function**. The model can then be evaluated on unseen test data to determine how well it generalises to new data.

9.1.1 Nearest Neighbours Classifiers

A simple image classifier is the nearest neighbours classifier, which classifies an image by comparing it to labelled training images and finding the class it is most similar to. This algorithm transforms all images into an image embedding space, where images are represented by an $N \times M$-d vector of

all pixel intensities. To determine if images are similar, we can use a similarity metric such as the Euclidean distance or the cosine similarity.

9.1.2 Linear Classifiers

A linear classifier makes predictions by dividing the input space into regions separated by hyperplanes. The model computes a weighted sum of the input features and compares it to a threshold to make a prediction. The model can be represented as:

$$\mathbf{y} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$$

where $\mathbf{W} \in \mathbb{R}^{K \times p}$ is the weight matrix, $\mathbf{x} \in \mathbb{R}^p$ is the input feature vector, and $\mathbf{b} \in \mathbb{R}^K$ is the bias term. Here, K is the number of classes and p is the number of features. The output vector $\mathbf{y}$ is a vector of unnormalised log-probabilities for each class. The class with the highest score is the predicted class. To introduce non-linearity into the model, we often use an activation function a on the output of the linear model:

$$\mathbf{z} = a\left(\mathbf{W}^\top \mathbf{x} + \mathbf{b}\right),$$

where the activation function is typically the Rectified Linear Unit (ReLU) function:

$$a\left(x\right) = \mathrm{ReLU}\left(x\right) = \max\left(0,\ x\right),$$

or other functions such as the sigmoid or hyperbolic tangent functions:

$$a\left(x\right) = \sigma\left(x\right) = \frac{1}{1 + e^{-x}},$$
$$a\left(x\right) = \tanh\left(x\right) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

To learn the coefficients in $\mathbf{W}$ and $\mathbf{b}$ we can use a **loss function** to measure the quality of the model's predictions. A common loss function is the **cross-entropy loss**, which measures the difference between the predicted class probabilities and the true class probabilities. The cross-entropy loss for a single training example is given by:

$$L^{(i)} = -\log\left(\frac{e^{y_i}}{\sum_j e^{y_j}}\right) = -y_i + \log\sum_j e^{y_j}.$$

Here we are talking the logarithm of the softmax function, which is a generalisation of the logistic (sigmoid) function that can be used for multi-class classification. The softmax function is defined as:

$$\sigma\left(\mathbf{y}\right)_i = \frac{e^{y_i}}{\sum_j e^{y_j}}.$$

The cross-entropy loss for the entire training set is called the cost which is the average of the loss over all n training examples:

$$J = \frac{1}{n}\sum_{i=1}^{n} L^{(i)}.$$

Given this cost function, we can use a gradient-based optimisation algorithm to tune the weights and biases of the model to minimise the cost. One common optimisation algorithm is **stochastic gradient descent** (SGD):

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J}{\partial \mathbf{W}},$$
$$\mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\partial J}{\partial \mathbf{b}},$$

where η is the learning rate. To take advantage of computing capability, we can introduce many layers into the model to create a fully connected neural network.

9.1.3 Convolutional Neural Networks (CNNs)

When classifying images, we can use convolution kernels to learn features from the image data. Convolutional Neural Networks (CNNs) are a type of neural network that is well-suited to image classification tasks. CNNs use convolutional layers to learn features from the input image, and pooling layers to reduce the spatial dimensions of the feature maps. A convolution layer has 4 hyperparameters:

- Number of filters K
- Filter size F
- Stride S
- Padding P

Given an input of dimension $N \times M \times C$, the output has a dimension of $N' \times M' \times K$, where:

$$N' = \frac{N - F + 2P}{S} + 1,$$
$$M' = \frac{M - F + 2P}{S} + 1.$$

The number of parameters is F^2CK with K biases. A pooling layer is often applied after a convolutional layer to reduce the spatial dimensions of the feature maps.

9.1.4 Training

Each update step in the backpropagation algorithm can be performed in batches or on the entire training set depending on hardware availability. An **epoch** is a single pass through the entire training set. The number of updates in an epoch refers to the number of batches required to cover the entire training set:

$$\text{updates per epoch} = \frac{n}{b},$$

where b is the batch size. To decide which epoch to stop training, we can use a validation set to monitor the model's performance. This set is not used to train the model, but to evaluate the model's performance on unseen data, on every epoch. A plot of the training and validation loss for each epoch, can be used to determine when to stop training.

9.1.5 Gradient Descent Optimisation

If the number of training examples is large, it may be infeasible to apply gradient descent on the entire training set. We can instead consider the following variants of the gradient descent algorithm we saw earlier:

- **Batch Gradient Descent**: Weights are updated using the gradients of the entire training set. This method is slow and computationally expensive.
- **Stochastic Gradient Descent (SGD)**: Weights are updated using the gradients of a single training example. This method is noisy and may not converge to the global minimum.
- **Mini-Batch Gradient Descent**: Weights are updated using the gradients of a small subset of the training set. This method is faster than batch gradient descent and less noisy than stochastic gradient descent.

To further improve these algorithms, we can introduce momentum to accelerate convergence and reduce oscillations around local minima. The momentum algorithm is defined as:

$$\begin{aligned} \mathbf{v} &\leftarrow \rho\mathbf{v} + \frac{\partial J}{\partial \mathbf{w}}, \\ \mathbf{w} &\leftarrow \mathbf{w} - \eta\mathbf{v}, \end{aligned}$$

here $\mathbf{w}$ represents both the weights and biases in the model. The addition of a velocity term $\mathbf{v}$ allows the model to build up velocity as a running mean of the gradients. The hyperparameter ρ controls the momentum term by adding friction, and is typically set to a value between 0.9 and 0.99. Other more advanced optimisation algorithms include the Nesterov Accelerated Gradient (NAG), AdaGrad, RMSprop, and Adam optimisers. These algorithms use adaptive learning rates to improve convergence and generalisation. We can also use learning rate schedules to adjust the learning rate during training.