ENGN8530: Computer Vision and Image Understanding: Theories and Research

Topic 2: Image Processing and Analysis Basics

Dr Chunhua Shen and Dr Roland Goecke
VISTA / NICTA & RSISE, ANU

Acknowledgement: Some slides from Dr Antonio Robles-Kelly, Dr Tiberio Caetano, and Dr Rob Mahony
Terms

- Image Processing:
  - Image in $\rightarrow$ Image out

- Image Analysis:
  - Image in $\rightarrow$ Measurements out, e.g. mean value

- Image Understanding:
  - Image in $\rightarrow$ High-level description out, e.g. this image contains two cars
Useful Textbooks

Camera Geometry

- The **aperture** allows light to enter the camera.
- The **image plane** is where the image is formed.
- The **focal length** is the distance between the aperture and the image plane.
- The **optical axis** passes through the center of the aperture and is perpendicular to it.
Cameras with Lenses - Recap

- A lens removes the geometric limit on resolution, since it focuses all light entering through the aperture on the same point on the image.
What is in Image?

- An image is an array/matrix of values (picture elements = ‘pixels’) on a plane which describe the world from the point of view of the observer.
- Pixels are rectangular parts of the image plane (the imaging sensor, to be precise)
- Square pixels are most desirable but cameras often do not output square pixels!

Source: Antonio Robles-Kelly
An image is a function of two real variables \( f(x,y) \), where \( x \) and \( y \) are the image coordinates of the pixels and \( f \) is the brightness / intensity at these.

For volumetric image data:
\[ f(x,y,z) \]

For colour images:
\[ f(x,y) = (r(x,y), g(x,y), b(x,y)) \]

For videos: \( f(x,y,t) \) where \( t = \text{time} \)

\[ I(x,y) = f[x,y] \]
Digital Image Acquisition

- Images, and more generally signals, in the real world are continuous (or analog).
- Digital image $f[x,y]$ is a mapping of the 2D continuous image space $f(x,y)$ into a 2D discrete (or digital) space.
- Inherently loss of information!
- Nyquist frequency (or rate)

Reference:

We use the square brackets here to highlight that it is a discrete space.
Nyquist Rate

- Defined as $f_N = 2B$ where $B$ is the bandwidth
- Why is this important?
  - If the sampling rate < $f_N$, aliasing effects occur.
  - If we want to avoid aliasing, sampling rate must be twice that of the highest frequency.
- In images, the samples are the pixels. Since the number of pixels is fixed by the camera, spatial aliasing will occur!
When the charge well is full, the pixel is saturated.

Saturation is not noise, it is a physical limit of the sensor.

In signal processing systems, the amount of noise is compared to that of the signal via the signal to noise ratio \( \text{SNR} = \frac{\text{signal power}}{\text{noise power}} \).
SNR (2)

- This ratio can vary enormously; from 1 to $10^5$ or more
- It is convenient to take the logarithm
  - $\text{SNR}(B) = \log_{10}(\text{signal power/noise power})$
  - Decibels (dB) are simply 10x Bels, i.e. $10\log_{10}(\text{signal power/noise power})$
  - 10dB is a tenfold increase in power, 3dB is a doubling of power
CCD Properties

- Ideal response:
  \[ \text{Response} = \text{gain} \cdot \text{Light intensity} \]

- Typical CCD response:
  \[ \text{Response} = \text{gain} \cdot \text{Light intensity} + \text{bias} + \text{noise} \]

Noise from:
- Thermal noise
- Quantisation of pixels
- Amplifier
CCD Noise – Thermal Noise

- Created by thermal radiation knocking out free electrons
- Builds up over length of exposure
- Reduced by cooling the CCD
- Poisson distribution
- At 20ºC and 25 fps
  - Mean 8
  - St.D. 2.8
- Significant for low light levels
  - SNR 38dB at mid-light levels
Dark Current

Constant response exhibited by a receptor of radiation during periods when it is not actively being exposed to light.

Video camera image

Enhanced contrast
CCD Noise - Quantisation

- Pixels are digitised to $2^B$ levels
- Uniformly distributed error on values of $\pm \frac{1}{2}$
- Uniform noise
  - Mean $\mu=0$
  - S.D. $\sigma = \frac{1}{\sqrt{12}}$
- $\text{SNR}(\text{dB}) = 10 \log(2^{B-1} \cdot \sqrt{12}) = 3B + 2.4$
- SNR=26dB for 8-bit digitisation, much better than the photon noise
- Photon noise is the limiting factor for CCDs
CCD Noise – Photon Noise

- Photon noise also known as quantum errors
- Light sources emit light in photons
  - $\beta$ photons/s on average
  - But photons emitted randomly
- Poisson noise
  - $f(k; \lambda) = \frac{e^{-\lambda} \lambda^k}{k!}$,
  - Time $t$
  - Mean intensity $\mu = \beta t$
  - Standard deviation $\sigma = \sqrt{\beta t}$
- For mid-brightness levels, $\text{SNR} = 23\text{dB}$
CCD Noise – Photon Noise (2)
CCD and Colour

- A CCD is a single-channel device
  - Responds to all colours of light
- To get a colour image from a CCD we need to use filters
  - Three filters are required, normally red, green and blue
  - With these three filters, we can reconstruct a colour image (three ‘channels’)
- Naïve method: Take three separate pictures, each with a different filter placed over the CCD
  - If the subject is moving between frames, the channels will not line up correctly!
**CCD and Colour (2)**

- **Use three CCDs in one camera**
  - Each has one of the red, green or blue filters in front of it
  - Light split three ways and channels captured simultaneously

- **Use a filter mosaic**
  - Special filter which has different colour for each cell of the CCD
  - More red/green than blue
  - Colours must be interpolated
  - Reduces effective resolution

*Source: Wikipedia*
Delta Function – An Aside

- Two types of delta functions
  - Discrete maths – **Kronecker delta function**
  - Continuous maths – **Dirac delta function**

- **Kronecker delta function**
  - Integer arguments
  - 0 everywhere except for one input value, where it is 1
  - \( \delta(x) \) is 1 at \( x=0 \) and zero elsewhere
  - \( \delta(x=a) \) is 1 at \( x=a \) and zero elsewhere

- **Dirac delta function**
  - Real argument, integrates to 1
  - 0 everywhere except for one
  - Input value where it is \( \infty \)

\[
\sum_{i=-\infty}^{\infty} f(i) \delta(i-a) = f(a)
\]

\[
\int_{-\infty}^{\infty} f(x) \delta(x-a) dx = f(a)
\]

\[
\int_{-\infty}^{\infty} \delta(x) dx = 1
\]
In 2D, the position of any point is described by two coordinates.

The position vector of the point can also be written in terms of two other vectors as $3\mathbf{i} + 4\mathbf{j}$.

$\mathbf{i}$ and $\mathbf{j}$ are basis vectors – vectors than can be used to construct other vectors.
Basis Sets

- Basis vectors do not have to be aligned along the $x$ and $y$ axes.

We can extend the space to $n$ dimensions, where we need at least $n$ basis vectors.

But if $\mathbf{a}$ and $\mathbf{b}$ were in the same direction, we could not describe any point because we can only go in one direction.
Basis Sets (2)

- How do we find the coefficients of the basis vectors?
- Let \( \{x_1, x_2, \ldots, x_n\} \) be the basis set
- The point \( \mathbf{p} \) is given by

\[
\mathbf{p} = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n
\]

- We need to find the \( a_1 \) - \( a_n \)
- Dot product with \( x_1 \)

\[
\mathbf{p} \cdot x_1 = a_1 x_1 \cdot x_1 + a_2 x_2 \cdot x_1 + \ldots + a_n x_n \cdot x_1
\]

- Dot product with \( x_2 \)

\[
\mathbf{p} \cdot x_2 = a_1 x_1 \cdot x_2 + a_2 x_2 \cdot x_2 + \ldots + a_n x_n \cdot x_2
\]

- And so on
- \( n \) simultaneous equations in the \( a \) – solve to find coefficients
Complete Basis

- What about being able to describe every point?
- We can write the vector $\mathbf{p}$ as

$$
\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} = p_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + p_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \ldots + p_n \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = p_1 \mathbf{e}_1 + p_2 \mathbf{e}_2 + \ldots + p_n \mathbf{e}_n
$$

- $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ are the standard basis vectors ($\mathbf{i}, \mathbf{j}$ but in $n$ dimensions)
- As long as we can construct all the standard basis vectors, we can construct $\mathbf{p}$
- Construct $\mathbf{e}_k$ with an orthonormal basis set

$$
\mathbf{e}_k = a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2 + \ldots + a_n \mathbf{x}_n = x_{1k} \mathbf{x}_1 + x_{2k} \mathbf{x}_2 + \ldots + x_{nk} \mathbf{x}_n
$$

- Writing as components $e_{ki} = \delta(i-k)$
Orthogonal Basis

- Now consider a special set of basis vectors, such that
  \[ \mathbf{x}_a \cdot \mathbf{x}_b = \delta(a-b) \]
  - All of these vectors are at right angles to each other
  - Unit length
  - **Orthogonal** and **Normal** - **Orthonormal**
- The computation of coefficients is now
  \[ \mathbf{p} \cdot \mathbf{x}_1 = a_1 \mathbf{x}_1 \cdot \mathbf{x}_1 + a_2 \mathbf{x}_2 \cdot \mathbf{x}_1 + \ldots + a_n \mathbf{x}_n \cdot \mathbf{x}_1 \]
  \[ = a_1 \cdot 1 + a_2 \cdot 0 + \ldots + a_n \cdot 0 \]
  \[ = a_1 \]
- And in general, \( a_i = \mathbf{p} \cdot \mathbf{x}_i \)
- When the basis set is orthogonal, the computation of coefficients is much easier
Linear Transformations

- Given a vector \( p \) and a complete basis set \( \{x_1, x_2, \ldots, x_n\} \) we can compute the coefficients \( \{a_1, a_2, \ldots, a_n\} \).

- Put the coefficients into a vector

\[
\begin{pmatrix}
  p_1 \\
p_2 \\
  \vdots \\
p_n
\end{pmatrix} \rightarrow \begin{pmatrix}
a_1 \\
a_2 \\
  \vdots \\
a_n
\end{pmatrix} = a
\]

- Given a vector \( a \) of coefficients and the basis set, we can compute \( p \).

- This bidirectional mapping is a linear transform
  - Both directions \( a \leftrightarrow p \)
  - No information lost
  - Constructed by linear combination of the basis elements
  - \( a \) is the transformation of \( p \)
Matrix Notation

- Write down the transform mathematically
- $n \times n$ matrix, rows are the basis vectors

\[
T = \begin{pmatrix}
    x_1^T \\
    x_2^T \\
    \vdots \\
    x_n^T
\end{pmatrix}
\]

- Then we get (called the inverse transform)
  \[ p = T^T a \]

- The transform is
  \[ a = (T^T)^{-1} p \]
Orthonormal Transforms

- If the basis set is orthonormal then the following apply
  \[ TT^T = I \quad \text{(orthonormality condition)} \]
  \[ T^{-1} = T^T \]

- And so the transform is
  \[ a = Tp \]

- Also, the completeness condition is
  \[ T^T T = I \]

- But since the transpose is also the inverse, this condition is automatically fulfilled by an orthonormal basis
Image Transforms

- Images are 2D, whereas vectors are 1D
  - Can also use the vector notation, by vectorizing the images, discussed later
  - We can still use the idea of basis sets and transforms
- Image basis set \( \{B_{1,1}(x,y), B_{1,2}(x,y), ..., B_{n,m}(x,y)\} \)
  \[ I(x,y) = a_{1,1}B_{1,1}(x,y) + a_{1,2}B_{1,2}(x,y) + ... + a_{N,M}B_{N,M}(x,y) \]
- Now there need to be the same number of basis images as pixels in the image
- Orthonormal condition
  \[ \sum \sum B_{i,j}(x,y)B_{k,l}(x,y) = \delta(i-k, j-l) \]
- Coefficients
  \[ a_{i,j} = \sum \sum B_{i,j}(x,y)I(x,y) \]
Going into Frequency Space

- Sometimes it is advantageous to use the underlying signal frequencies in an image, rather than the pixels.
- For example, for removing periodic noise.
- Fourier Transform
- Discrete Cosine Transform
- Other transforms (not covered here):
  - Wavelet Transform
  - Laplace Transform
  - $z$ Transform
Fourier Transform

- Made from cos and sin put together using complex numbers

1D

\[ x_{kj} = \sqrt{\frac{1}{n}} e^{-2\pi k j / n} \]

\[ = \sqrt{\frac{1}{n}} \cos(2\pi k j / n) + i \sin(2\pi k j / n) \]

2D

\[ B_{k,l}(x, y) = \sqrt{\frac{1}{NM}} e^{-\frac{2\pi ikx}{N}} e^{-\frac{2\pi ily}{M}} \]

- Transformed image may have complex numbers for pixels
FT (2)

- There is a continuous version of this transform which is useful when working with functions rather than discrete images

\[ F(u) = \int_{-\infty}^{\infty} f(x)e^{-iux} \, dx \]

- This is make a Fourier integral pair with

\[ f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(u)e^{iux} \, du \]

- This can be extended into 2D:

\[ F(u, v) = \int_{-\infty}^{\infty} f(x, y)e^{-iux} e^{-ivy} \, dx \, dy, \quad f(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(u, v)e^{iux} e^{ivy} \, du \, dv \]
Some common functions for Fourier Transform:

\[ f(x) = 1 \quad F(u) = \delta(0) \]

\[ f(x) = \sin x \quad F(u) = \frac{i}{2} \left( \delta(u-1) - \delta(u+1) \right) \]

\[ f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-x^2/2\sigma^2} \quad F(u) = \frac{1}{2\pi} e^{-\sigma^2 u^2/2} \]
Discrete Fourier Transform

- The Fourier transform is one of the most important transforms because it has useful properties for filtering.
- The 2D DFT has the basis set

\[ B_{u,v}(x, y) = e^{-\frac{2\pi iux}{N}} e^{-\frac{2\pi ivy}{M}} \]

- They contain complex numbers so the resultant output may be complex.
- As before, the transformed image is calculated by

\[ F_{u,v} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f_{x,y} e^{-\frac{2\pi iux}{N}} e^{-\frac{2\pi ivy}{M}}, \]
The complex exponentials are in fact sine and cosine functions

\[ e^{i\theta} = \cos\theta + i\sin\theta \]

\[ e^{-\frac{2\pi iux}{N}} = \cos\left(-\frac{2\pi iux}{N}\right) + i\sin\left(-\frac{2\pi iux}{N}\right) \]

- Really just sine and cosine transforms stuck together
- Low frequency detail in first components, high frequency in higher order components
DFT Example

DFT coefficients

ENGN8530: CVIU
Inverse DFT and FFT

- There is an inverse transform which will return to the original image

\[ F_{u,v} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f_{x,y} e^{-2\pi iux/N} e^{-2\pi ivy/M}, \]

\[ f_{x,y} = \frac{1}{NM} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_{u,v} e^{2\pi iux/N} e^{2\pi ivy/M}. \]

- This, and other transforms, normally take NM operations per pixel, or \(N^2M^2\) operations in total

- The DFT can be computed by an algorithm called the Fast Fourier Transform (FFT) which takes just \(\log N + \log M\) operations per pixel or \(NM(\log N + \log M)\) total.

- The FFT requires that \(N\) and \(M\) are powers of 2
Other Properties of the FT

- **Linearity** \( af(x,y) + bg(x,y) \Leftrightarrow aF(u,v) + bG(u,v) \)

- **Scaling** \( f(ax, by) \Leftrightarrow \frac{1}{ab} F\left(\frac{u}{a}, \frac{v}{b}\right) \)

- **Translation** \( f(x-x_0, y-y_0) \Leftrightarrow e^{ix_0u} e^{iy_0v} F(u,v) \)

  \[ \Rightarrow |F'(u,v)|^2 = e^{-ix_0u} e^{-iy_0v} F^*(u,v) e^{ix_0u} e^{iy_0v} F(u,v) = |F(u,v)|^2 \]

- **Total power** \[ \sum_{x,y} |f(x,y)|^2 = \sum_{x,y} |F(u,v)|^2 \]

- \( |f(x,y)|^2 \) is the image power, \( |F(u,v)|^2 \) is the spectral power
Discrete Cosine Transform

- **1D**

\[ x_{ki} = \sqrt{\frac{2}{n}} \cos \left( \frac{\pi}{n} k \left( i + \frac{1}{2} \right) \right) \]

Indices run from 0 to \( n-1 \)

- **2D**

\[ B_{kl}(i, j) = \sqrt{\frac{4}{NM}} \cos \left( \frac{\pi}{N} k \left( i + \frac{1}{2} \right) \right) \cos \left( \frac{\pi}{M} l \left( j + \frac{1}{2} \right) \right) \]

- Orthonormal transform
- Often used for compressing images
- Also a similar sine version
DCT (2)

We go from image to the frequency domain and back \( f(x, y) \Leftrightarrow F(u, v) \)

Shift the values of the pixels into signed integers \([-2^7, 2^7 - 1]\)

Split the image into \(8 \times 8\) pixel blocks

In the encoder we apply the forward Discrete Cosine Transform (DCT)

\[
F(u, v) = \frac{1}{4} C(u)C(v) \left[ \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y) \cos \left( \frac{2x+1}{16} \pi u \right) \cos \left( \frac{2y+1}{16} \pi v \right) \right]
\]

\[
C(u), C(v) = \begin{cases} 
1/\sqrt{2} & \text{for } u, v = 0 \\
1 & \text{for } u, v = 1, \ldots, 7
\end{cases}
\]

- DC coefficient
- AC coefficients

In the decoder we apply the Inverse DCT transform which is the dual of the forward DCT transform and we reconstruct the image

\[
f(x, y) = \frac{1}{4} \left[ \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v) F(u, v) \cos \left( \frac{2x+1}{16} \pi u \right) \cos \left( \frac{2y+1}{16} \pi v \right) \right]
\]
DCT Example

- DCT coefficients
DCT Example (2) - Denoising

- $\text{abs}(\log(B)) > 5$
- $\text{abs}(\log(B)) > 4$
DCT Example (3) - Compression
Due to the sub-sampling of the chrominance components the compression achieved in colour images is higher.
Linear Filtering

- Filtering is the process of modifying the image by removing some unwanted information while retaining the important information.
- Linear filters are popular because they are easy to implement via the convolution theorem.
  - Also possible to analyse the statistical properties of the filtered image.
- Examples of linear filter operations:
  - Noise reduction
  - Low pass
  - High pass
  - Edge enhancement
  - De-blurring
Convolution

- Consider the class of linear operators on the image array

\[ I'(x,y) = \sum_{a,b} g(x,y,a,b)I(a,b) \]

- If we are interested in performing the same operation at each pixel, we can often define \( g \) in coordinates relative to the pixel \((x,y)\) (a stationary or shift invariant filter)

\[ \Rightarrow g(x,y,a,b) = g(x-a,y-b) \]

- So we have a \textit{convolution} \( I'(x,y) = \sum_{a,b} g(x-a,y-b)I(a,b) \)

- A continuous version: \( I'(x,y) = \int\int g(x-a,y-b)I(a,b)dadb \)
The convolution operation is often written as

\[ I \otimes g = \sum_{a,b} g(x - a, y - b)I(a, b) \]

**Example**

The input image is convolved with the convolution filter to produce the result.
Convolution (3)

- Can combine two filters to form a double linear filter

\[
I'(x, y) = \sum_{c,d} h(x-c, y-d) \sum_{a,b} g(c-a, d-b) I(a,b)
\]

\[
= \sum_{a,b} \left[ \sum_{c,d} h(x-c, y-d) g(c-a, d-b) \right] I(a,b)
\]

\[
= \sum_{a,b} \left[ \sum_{r,s} h(r, s) g(x-a-r, y-b-s) \right] I(a,b)
\]

\[
= \sum_{a,b} f(x-a, y-b) I(a,b)
\]

- Any number of these filters can be applied in one operation
- For image of resolution NxM takes N^2M^2 operations
Convolution Example

Input Image

Convolution Filter

Result
Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of the Fourier transforms of the functions

\[ F(f \otimes g) = F(f)F(g) \]

- In other words, the convolution can be calculated by

\[ f \otimes g = F^{-1}[F(f)F(g)] \]

- This can be calculated in \( O(NM \log[MN]) \) steps not \( O(N^2M^2) \) which can be a significant computational reduction for large \( N,M \).
Correlation

- The correlation between two images is given by
  \[ I \oplus G = \sum_{a,b} G(x+a, y+b)I(a,b) \]
  and represents the similarity of G and I at position \((x,y)\).

- A continuous version
  \[ I \oplus g = \int_{-\infty}^{\infty} g(x+a, y+b)I(a,b)da db \]

- The Fourier transform of the correlation is given by
  \[ \mathcal{F}(f \oplus g) = \mathcal{F}(f)^* \mathcal{F}(g) \]

- The auto-correlation is \( f \oplus f \) and the FT is equal to \( F(f)^* F(f) \), which is the power spectrum \( |F(f)|^2 \).
Correlation Example

Input Image  Correlation filter $G$  Result
Spatial Averaging

- After this excurse to convolution, back to linear filtering...
- Each pixel is replaced by a weighted average of its neighbourhood pixels
  - Useful for sub-sampling and noise reduction

\[ I'(x, y) = \sum_{k,l \in W} a(k,l)I(x-k, y-l) \]

\[
\begin{array}{ccc}
\frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4}
\end{array}
\quad
\begin{array}{ccc}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{array}
\quad
\begin{array}{ccc}
0 & \frac{1}{8} & 0 \\
\frac{1}{8} & \frac{1}{4} & \frac{1}{8}
\end{array}
\quad
\begin{array}{ccc}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{array}
\]
Spatial Averaging (2)

- Noise reduction properties
- If the pixels in the window have a constant value with additive noise of variance $\sigma^2$, i.e.
  - The noise is reduced to variance $\sigma^2/N$ for an equal-weighted average and the size of the window must be limited so that the ‘constant value’ approximation holds

\[
\begin{array}{ccc}
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{8} & \frac{1}{8} & 0 \\
\frac{1}{8} & \frac{1}{8} & 0 \\
\frac{1}{8} & \frac{1}{8} & 0 \\
\frac{1}{8} & \frac{1}{8} & 0 \\
\end{array}
\]
Median Filtering

- Often easily confused with mean filter
- Median filter is non-linear
- Good for noise / outlier removal
- Example:
  - Window size of 3, repeating edge values:
  - \( x = [2 \ 80 \ 6 \ 3] \)
  - \( y[1] = \text{Median}[2 \ 2 \ 80] = 2 \)
    - \( y[2] = \text{Median}[2 \ 80 \ 6] = \text{Median}[2 \ 6 \ 80] = 6 \)
    - \( y[3] = \text{Median}[80 \ 6 \ 3] = \text{Median}[3 \ 6 \ 80] = 6 \)
    - \( y[4] = \text{Median}[6 \ 3 \ 3] = \text{Median}[3 \ 3 \ 6] = 3 \)
  - so \( y = [2 \ 6 \ 6 \ 3] \) where \( y \) is the median filtered output of \( x \)
Median Filtering Example

Original image

Image plus salt & pepper noise

Local averaging, Across 3×3 regions

Median filtering with 3×3 kernel
Gaussian Filter

- Gaussian filter
  \[ G(x, y) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Very popular noise reduction filter
  - Circularly symmetric (r^2 = x^2 + y^2)
  - Separable \( G(x, y) = H(x)H(y) \)
  - A weighted neighbourhood average
High-Pass Filter

- We have an image with a slowly varying background which we want to remove.
- Use a high-pass filter which removes low frequency components from the spectrum.
We have an image with too much detail
Use a low-pass filter which removes high frequency components from the spectrum

Actually a square wave in the frequency domain
Edge Detection / Filtering

- An edge filter is designed to enhance the image where there is an edge, i.e. a transition from one value to another.
- Consider a transition in image value.

![Graphs showing Image value I(x), First derivative dI/dx, and Second derivative d^2I/dx^2.](image-url)
Edge Filtering (2)

- In a discrete image, the differential is approximated by the difference between adjacent values:

$$\left. \frac{df}{dx} \right|_{x-0.5} \approx I(x) - I(x-1)$$

- Noise is actually increased!!
Edge Filtering (3)

1\textsuperscript{st} order edge detection methods:
1. Determine gradient of image
2. Perform non-maximal suppression
3. Threshold

2\textsuperscript{nd} order edge detection methods:
- Locate zero crossings in 2\textsuperscript{nd} derivative using Laplacian.
Edge Filtering (4)

- Another approach is to differentiate a noise-smoothed image
  - Use a noise smoothing filter \( N(x,y) \)
  - Differentiate

\[
E_x = \frac{\partial}{\partial x} N \otimes I(x,y)
\]

\[
= \frac{\partial}{\partial x} \sum \sum N(x-a, y-b)I(a,b)
\]

\[
= \sum \sum \frac{\partial N}{\partial x} (x-a, y-b)I(a,b)
\]

- We can differentiate the filter and then apply it to the image
- Example

<table>
<thead>
<tr>
<th></th>
<th>+1</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Difference operator

<table>
<thead>
<tr>
<th></th>
<th>+1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Corresponding filter
## Edge Filtering (5)

### Some 3x3 edge filters

<table>
<thead>
<tr>
<th>Noise filter</th>
<th>x Edge Filter</th>
<th>y Edge Filter</th>
<th>Sobel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 1</td>
<td>-1 0 1</td>
<td>-1 -2 -1</td>
<td></td>
</tr>
<tr>
<td>2 4 2</td>
<td>-2 0 2</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 2 1</td>
<td>-1 0 1</td>
<td>1 2 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roberts</th>
<th>x Edge Filter</th>
<th>y Edge Filter</th>
<th>Smoothed (Prewitt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
<td>-1 0 1</td>
<td>-1 -1 -1</td>
<td></td>
</tr>
<tr>
<td>-1 0</td>
<td>-1 0 1</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>-1 0</td>
<td>-1 0 1</td>
<td>1 1 1</td>
<td></td>
</tr>
</tbody>
</table>
A good noise-resistant edge detector comes from the differential of the Gaussian filter

\[ N(x, y) = G(x, y) = e^{-(x^2+y^2)/2\sigma^2} \]

\[ E_x = \frac{\partial N}{\partial x} = xe^{-(x^2+y^2)/2\sigma^2} \quad E_y = \frac{\partial N}{\partial y} = ye^{-(x^2+y^2)/2\sigma^2} \]
Edge Filtering (7)

- Consider the following edge in an image:

$$
\theta = \arctan \left( \frac{\partial I}{\partial y} \right) \frac{\partial I}{\partial x}
$$

- By taking gradients in the x and y directions we can find the maximum gradient of the edge and its direction.
Canny Edge Detector

- Developed by John Canny, 1986
- First order method (i.e. gradient-based)
- Algorithm:
  - Convolve image I with 1D Gaussian mask G; standard deviation s of Gaussian is a parameter to the edge detector
  - Create a 1D mask for the 1st derivative of the Gaussian in x and y directions, G_x and G_y; same s as before
  - Convolve I with G along rows to give x component image I_x, same for columns to give y component image I_y
  - Convolve I_x with G_x and I_y with G_y, i.e. with the derivative of the Gaussian
Canny Edge Detector (2)

- Magnitude of result
  \[ M(x,y) = \sqrt{(I'_x(x,y)^2 + I'_y(x,y)^2)} \]
- Nonmaximum suppression: Pixels that are not local maxima are removed by a process called hysteresis thresholding
- Hysteresis thresholding:
  - Two thresholds \( k_{\text{high}} \) and \( k_{\text{low}} \)
  - Threshold at \( k_{\text{high}} \) to find strong edges.
  - Threshold at \( k_{\text{low}} \) to find weak edges.
  - Accept all weak edges that are “connected” to strong edges.

Reference:
Canny Edge Detector (3)

Original image

Magnitude of gradient

Magnitude of gradient after non-maximal suppression
Canny Edge Detector (4)

Weak edges

Strong edges & connected weak edges

Strong edges
Canny Edge Detector (5)

Result
2nd Order Edge Filtering

- The Laplacian operator can also be used to detect edges
- Isotropic
- Zero crossings at edge locations
- Again, apply to a noise smoothing filter
- 2nd order method
- Laplacian of Gaussian:

\[
\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad \nabla^2 G = \frac{1}{\sigma^2} \left[ \frac{1 - x^2 + y^2}{\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}
\]
The Laplacian is the sum of the 2nd derivatives,

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

The LoG (Laplacian of Gaussian) operator measures the strength of the 2nd derivative in both directions after the image has been smoothed by a Gaussian $G$

$$\nabla^2 (G * I) = \nabla^2 G * I$$
2nd Order Edge Filtering (3)

Derivatives and Laplacian of a 2D Gaussian $G$, intensity axis ($a$) not to scale

$$K_{\frac{\partial G}{\partial x}}$$

$$K_{\frac{\partial G}{\partial y}}$$

$$K_{\nabla^2 G} = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$
Difference of Gaussian

- LoG can be approximated by a difference of 2 Gaussians.

- This is the **DoG (Difference of Gaussians)** operator – also called the Mexican Hat Operator.
Marr-Hildreth Edge Detector

- The Marr-Hildreth operator locates edges at the zero crossings in the LoG of the image $\nabla^2 G \ast I$.
- This is equivalent to looking for zeros in the 2nd derivative.

Steps:
- Convolve image with Gaussian
- Compute the LoG
- Compute the zero crossings of LoG
Marr-Hildreth Edge Det. (2)

- **Advantages:**
  - Edges found in all orientations simultaneously
  - Zero crossings easy to find: simply look for a sign change
  - Zero crossings always define closed curves, this is handy for segmenting objects in an image

- **Disadvantages:**
  - 2nd derivative is very sensitive to noise
  - Not all edges form closed contours
Marr-Hildreth Edge Det. (3)
Marr-Hildreth Edge Det. (4)
Marr-Hildreth Edge Det. (5)
Hough Transform

- It started as a feature detector for detecting lines in edge images.
- Circular Hough Transform extends this idea to detect circles.
- The Generalised Hough transform
  - can be used to find any shape in an image.
  - more complex shapes lead to rapid increases in computational complexity.
  - practical applications are limited.

Linear Hough Transform (2)

- Each edge point ‘votes’ for different shape configurations.
- Each edge point votes for all the shapes that could cause that edge point to exist.
- Shapes with the most votes are ‘found’ in the image.
- Consider each edge point, what lines could cause that point to appear in the image?
Linear Hough Transform (3)

First point

Second point
Linear Hough Transform (4)
Linear Hough Transform (5)

- Lines that receive the most ‘votes’ are found.
Linear Hough Transform (6)

Parameterise the line in terms of the $\theta$ and $r$,

$$r = x \cos \theta + y \sin \theta,$$

for $\theta \in [0, \pi]$.
Linear Hough Transform (7)

- For a given $x$ and $y$ every line passing through the point $(x, y)$ can be uniquely represented by a $(\theta, r)$ pair.
Linear Hough Transform (8)

- An **accumulator array** is used to count the ‘votes’ for each set of parameter values.
- Increment parameter values corresponding to the lines that could cause this point to appear in the edge image.
Linear Hough Transform (9)

- These are at maxima in the accumulator array.
LHT Algorithm

- **Parameterise** the line in terms of the $\theta$ and $r$,
  \[ r = x \cos \theta + y \sin \theta \]
- Zero the accumulator array.
- Determine an edge map $I_{\text{edge}}$ of the image.
- Examine each point $(x,y)$ in $I_{\text{edge}}$.
  - If $I_{\text{edge}}(x,y)=1$ increment all cells in the accumulator array that correspond to a line passing through this point in the image. That is, points on the line
    \[ r = x \cos \theta + y \sin \theta \]
    in parameter space $(\theta, r)$
- **Maxima in the accumulator array** indicate the dominant parameter values, which in turn specify the lines in the image.
Circular Hough Transform

- Instead of lines searches for circles.
- Slightly different parameterisation:

\[ x = x_0 + r \cos \theta \]
\[ y = y_0 + r \sin \theta \]
Generalised Hough Transform

- **Parameterise shape:** Specify the shape $S$ that you wish to detect in terms of a set of bounded scalar parameters $q$. E.g.
  - A line has 2 parameters: orientation $\theta$ and perpendicular distance from the origin $r \Rightarrow q=(\theta, r)$.
  - A circle has 3 parameters: centre location (given by $x$ and $y$) and radius $r \Rightarrow q=(x, y, r)$.

- For each data point in the edge image consider every possible shape $S(q_i)$ that co-exists with that data point. A ‘vote’ is registered for each associated set of parameters $q_i$.
  - Parameter space is divided into a number of discrete accumulator cells. An accumulator array is used to count the ‘votes’ for each accumulator cell.

- Shapes are located in the image as those that co-exist with the maximum number of data points, i.e. maxima in the accumulator array.
Thresholding

Thresholding is used to convert grey-scale images into binary images as well as for object segmentation.

\[ I_{bin} = \begin{cases} 1 & \text{if } I_{\text{grey}}(p) \geq d \\ 0 & \text{otherwise} \end{cases} \]

Example of text, taken from the opening of *The Big Sleep*, by Raymond Chandler, 1939, Hamish Hamilton.
Source: Rob Mahony
Thresholding (2)

**Advantages:**
- Simplifies images, makes it easier to analyse
- Can help to segment objects
- Can construct a mask from result

**Disadvantages:**
- Loss of information
- How to choose a good threshold? A bit of a ‘Black art’.
- Threshold generally chosen manually
- Image-specific
Intensity Histogram

- Intensity histogram can be useful

- Often it is necessary to adaptively choose a threshold for a particular application.

Intensity histogram of text image

Source: Rob Mahony
Intensity Histogram (2)

- An 8-bit intensity image has 256 gray levels.
- If all the intensity values in an image occupy the first 100 gray levels it is hard to distinguish them.

Humans can detect ~100 discrete levels in this region
Intensity Histogram (3)

- We could *spread out* these intensity values across the spectrum and we would be better able to detect the differences between successive gray levels.
- This is NOT histogram equalisation!
Histogram Equalisation

- In practice not *all* the intensity values will be packed into a single region.
- Fully utilise sensory capacity by spreading information *uniformly* over all intensities
- Flatten histogram

→ Histogram Equalisation
Histogram Equalisation (2)

Input histogram (on side)

Transformed histogram

Plot of homogeneous point operator $f$
Homogeneous point operators

\( I_{new}(p) \) is determined from \( I(p) \) according to some function \( f \) operating solely on the pixel \( I(p) \).

\[ f : I(p) \rightarrow I_{new}(p) \]

These operators alter the intensity histogram. A monotonic \( f \) ensures the ordering of intensity values is preserved.
Histogram Equalisation (4)

Some common Homogeneous Point Operators:

- Darken
- Lighten
- Emphasize shadows
- Raise Contrast
- Lower Contrast
- Emphasize lights
- Threshold
Histogram Equalisation (5)

**Example**

- Original Histogram
- Original Image
- Equalised Histogram
- Equalised Image
Histogram Equalisation (6)

Before

After

No. occurrence

Grey levels

255

Grey levels
Colour Images

- Humans can distinguish
  - 100,000s of different colour shades and intensities,
  - But only ~100 levels of gray.

- Digital images
  - $8 \text{ bit} = 2^8 = 256 \text{ values}$
  - $3 \times 8 \text{ bit} = 256 \times 256 \times 256 = 16777216 \text{ values}$

- How can we use colour in computer vision?
Light and Colour

- First, some definitions for light
  - Light is an electro-magnetic wave
  - Wavelength is the distance between successive peaks of the wave ($\lambda$)
  - Frequency is the rate at which peaks pass a point ($f$)
  - Speed of light is $c$

- The spectrum of a light source is $S(\lambda)$, the intensity of light at wavelength $\lambda$. 

\[ f = \frac{c}{\lambda} \]
Light and Colour (2)

- The wavelength of light is related to the colour that we perceive

<table>
<thead>
<tr>
<th>Radio waves</th>
<th>Microwaves</th>
<th>Infra-red</th>
<th>Ultra-violet</th>
<th>X-Rays</th>
<th>Gamma rays</th>
</tr>
</thead>
</table>

- ‘Pure’ light has only one wavelength in it
- Normally, light is a mixture of wavelengths - the spectrum of light $S(\lambda)$
  - The relationship between colour perception and composition of the light is much more complicated in this case
Reflection from a Surface

- The reflectance function $R(\lambda)$ is the fraction of light with wavelength $\lambda$ that is reflected by the surface (cp. BRDF)
  - Assume that the geometric dependence has been removed
  - So the spectrum of the reflected light is $S(\lambda)R(\lambda)$
Colour Theory

- Every colour on a computer screen is a combination of three primary colours.
- The primary colours form a basis spanning (almost) the entire space of possible colours.
- There are 2 sets of primary colours:
  - Additive
  - Subtractive

Additive colour: TV, computer screens
Subtractive colour: Painting, printing
Colour Theory (2)

- The additive primary colours are (RGB):
  - Red
  - Green
  - Blue
- The subtractive primary colours are (CMY):
  - Cyan
  - Magenta
  - Yellow

These are often called secondary colours.
Televisions and computer displays **generate** light which is **added** to display the desired colour.

Artists mixing paint use the subtractive primaries, because paints **absorb** all colours of light except those they reflect, and are thus a source of **subtracting** light.
Colour Theory (4)

- Colour is displayed on a computer or television screen by adding red, green and blue light filters, or by changing the intensity of red, green and blue pixels that are very close to each other.

- Colour is displayed on white paper by adding cyan, magenta and yellow pigment. Here light must be reflected off the picture to view the image (i.e. you can’t see it in the dark).

Additive colour: TV, computer screens

Subtractive colour: Painting, printing
Colour Theory (5)

- There is no way of making black by adding red, green and blue light sources, the screen must be black to start with.

- Likewise there is no way of generating white by combining cyan, magenta and yellow pigment, so the paper must be white.
Colour Perception

- Humans cannot perceive the entire spectrum, instead we sample it using colour-sensitive cone cells in the eye.
- The responses of these cells determine the raw colour information and brightness.
- The responses look like this
  \[ r = \int R(\lambda)S(\lambda)F_r(\lambda)d\lambda \]
  \[ g = \int R(\lambda)S(\lambda)F_g(\lambda)d\lambda \]
  \[ b = \int R(\lambda)S(\lambda)F_b(\lambda)d\lambda \]
Coloured Perception (2)

- **Cone cells** for bright light and colour vision, 3 different ones for colour.
- **Rod cells** for low light vision, only one kind → everything looks grey at night!
- Different spectra can look the same colour.
- The perceived brightness also depends on colour.
- If we stimulate the three colour receivers in the same way, we get the same colour.
The coloured blocks in each image are the same colour but look different.

Post processing in the brain attempts to remove the colour of the light source and find the colour of the object.

- It uses background to find light source colour.
Colour Spaces

- Because different devices have different gamuts, there is a need to select the colour representation which is matched to the device.
- There are many different colour specifications.

- RGB, normalised RGB
- CMY
- CMYK
- HSV
- HSL / HSI
- YUV
- YIQ
- CIE-XYZ
- CIE-LAB
Intensity v. Luminance

- Notice the difference between intensity and luminance.
- **Intensity** is a physical quantity determined by the energy of the signal.
- **Luminance** is a product of colour perception and is a purely psychological phenomenon.
A range of colours can be created by selecting 3 (or more) primary colours from this diagram.

Such a range is called a **colour gamut**.

3 colours give a triangle – such as the red, green and blue phosphors in a screen.

Colours outside the gamut are not displayable.
In 1931 CIE (Commission Internationale de l'Éclairage) defined 3 standard primary colours X, Y, Z that can be added together to make all visible colours.

Based on response curves of the eye’s 3 receptors.

X, Y, Z do not physically exist.
CIE XYZ (2)

- CIE maps the RGB values onto X,Y,Z values. Z matches the apparent brightness.
- Normalised coordinates
  \[ x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z} \]
- x,y are colour coordinates
- Linear additive colour space
  - Mixed colours lie on the line between the colours
- White lies in the centre
- Axis out of page is brightness

CIE Chromaticity Diagram:
This is a perceptual map – based on responses from people about how colours are related to each other
CIE XYZ

- Disadvantages:
  - Hard to include brightness
  - Perceived colour difference is not well correlated with spacing in the color space.
RGB

- RGB Colour Cube

- Red (1,0,0)
- Green (0,1,0)
- Blue (0,0,1)
- Cyan
- Magenta
- Yellow
- Greys
RGB (2)

- RGB cannot represent all visible colours!
- Different colour spaces allow us to separate different components, such as
  - Intensity
  - Luminance
  - Hue
  - Saturation
CIE L*a*b*

While the retina directly registers 3 colour stimuli (red, green and blue) these are later processed and 3 sensations are generated:

- a red-green sensation
- a yellow-blue sensation
- a brightness sensation

These sensations form the basis of the \textit{CIE L*a*b*} (or \textit{CIE Lab}) colour space, developed in 1976.
CIE L*a*b* (2)

XYZ to CIE L*a*b* conversion:

\[ L^* = 116 \frac{f(Y/Y_n)}{f(Y/Y_n)} - 16 \]
\[ a^* = 500 \left[ f(X/X_n) - f(Y/Y_n) \right] \]
\[ b^* = 200 \left[ f(Y/Y_n) - f(Z/Z_n) \right] \]
HSV and HSL/HSI

- Hue
- Saturation
- Value

- Hue
- Saturation
- Luminance
HSV and HSL/HSI

Original  Saturation  Original  Saturation

Hue  Value  Hue  Intensity

HSV  HSL/HSI
Illumination Effects

- **Problem:**
  - Many colour spaces are not robust with respect to changes in lighting.
  - As lighting changes so does location in colour space.
Normalised Colour Spaces

Normalising colour spaces with respect to intensity/luminance gives a *chrominance space*. Some examples include:

- *Normalised rg from RGB*

  \[
  r = \frac{R}{R + G + B} \quad g = \frac{G}{R + G + B}
  \]

  alternatively could use *rb* or *bg*.

- *HS*: the first 2 channels of HSV or HSL.

- *AB*: the 2\(^{nd}\) and 3\(^{rd}\) channels of CIELab
Colour in Computer Vision

- Colour is a powerful pixel-level cue.
- Especially useful for image segmentation.
- Case study:
  - Skin colour detection
Skin Colour Detection

- Skin colour detection is the first step in the majority of recent face detection systems.
- Skin chrominance varies little between different skin types, it is the *intensity* which varies with the melanin level in the skin.

Steps:
- Construct a skin chrominance model (off-line).
- Test pixels in test image for skin colour, and output result.
Skin Colour Detection (2)

Constructing skin colour model (off line):

- Sample images of skin are passed to the system (in RGB format)
- The colour value of every pixel is mapped to a 2D chrominance space.
- A model is selected describing the distribution of skin colour pixels in chrominance space.
Skin Colour Detection (3)

- Identify a region containing only skin.
- Crop this region.
- This is our skin colour sample, call this \( I \).
- For this example we’ll use \( ab \) chrominance space.
- Look at every pixel \( p \) in turn and determine the chrominance values \( (a, b) \) at this point.

\[
I(p_i) = [R_i, G_i, B_i]^T \quad \rightarrow \quad [L_i, a_i, b_i]^T
\]
Skin Colour Detection (4)

- Plot every $(a,b)$ value found in chrominance space.
- Do this with an accumulator array: increment the value of a bin $(a_i,b_i)$ for each occurrence of that chrominance $(a_i,b_i)$ in $I$. 
Skin Colour Detection (5)

- All the points are clustered in a small region!
Skin Colour Detection (6)

- We can do this for as many skin sample images as we like – the more the better.
- Cai and Goshtasby used 2300 skin samples from 80 images and found they all lay within a small region.
  \[ a \in [-10, 60] \quad b \in [-10, 40] \]
- Only need to put our accumulator array over this region in chrominance space.
- Therefore we can get a much finer resolution for our skin chrominance model.

Reference:
Skin Colour Detection (7)
Skin Colour Detection (8)

- Construct an image $I_{\text{skin}}$ showing the likelihood that a given pixel is skin as follows:
- For every pixel $p_i$
  - Determine the chrominance values $(a_i, b_i)$ of $I_{\text{test}}(p_i)$
  - Lookup the skin likelihood for $(a_i, b_i)$ using the skin chrominance model.
  - Assign this likelihood to $I_{\text{skin}}(p_i)$
- Take the test image $I_{\text{test}}$ and construct an image $I_{\text{skin}}$
Skin Colour Detection (9)

$\mathbf{I}(p_i) \rightarrow (a_i, b_i)$

$I_{\text{test}}$  

$I_{\text{skin}}$
Image Analysis

- Image / Region-of-Interest statistics:
  - Arithmetic mean
  - Standard deviation / variance
  - Histogram

![Histogram Graph](image_url)
Image Analysis (2)

- Histogram
We can define a distance between two pixels according to their location with respect to each other. A distance can be defined in several ways:

**Euclidean Distance**

\[ D_E(P(i, j), X(k, l)) = \sqrt{(i - k)^2 + (j - l)^2} \]

**City Block Distance**

\[ D_{CB}(P(i, j), X(k, l)) = |i - k| + |j - l| \]

**Chessboard Distance**

\[ D_C(P(i, j), X(k, l)) = \text{Max}\{|i - k|, |j - l|\} \]
Image Analysis (4)

- Image analysis techniques **extract image features** which are grouped in meaningful structures in order to be interpreted.
- Each **pixel** is characterised by its location and its grey level value.
- **Features** are characterised by their geometry and their grey level distribution.
- Global interpretation of the grey level statistics is given by histogram while global interpretation of the geometry can be given by various measures.
- **Iconic images** – images which contain an array of values organized on a two-dimension (2-D) lattice – usually the result of a processing or a transformation

- **Segmented images** – parts of the image having a meaningful relationship are joined into groups – representing objects

- **Geometric representations** – knowledge about 2-D and 3-D shape

- **Relational models** – give the ability to treat data more efficiently and at a higher level of abstraction.

  Traditional image data structures: matrices, chains, graphs, relational databases, object properties.

  - They are used not only for the direct representation of image information, but also as a basis of more complex hierarchical methods of image representation.
Moments of Objects

- Often work with binary images / masks
- Binary images are particularly useful for
  - Identifying objects with distinctive silhouettes, e.g. components on a conveyor in a manufacturing plant.
  - Recognising text and symbols, e.g. document processing, or interpreting road signs.
  - Determining the orientation of objects
- Define the binary image $I$ of an object to be

$$I(x, y) = \begin{cases} 
1 & \text{for points on the object} \\
0 & \text{everywhere else}
\end{cases}$$
Moments of Objects (2)

- Area is given by the $0^{th}$ moment
  \[ A = \sum_x \sum_y I(x, y) \]

- Centre of mass (centroid) is given by the $1^{st}$ moments
  \[ \bar{x} = \frac{\sum_x \sum_y xI(x, y)}{\sum_x \sum_y I(x, y)} \]
  \[ \bar{y} = \frac{\sum_x \sum_y yI(x, y)}{\sum_x \sum_y I(x, y)} \]
Moments of Objects (3)

- The orientation of an object is defined as the axis of minimum inertia. This is the axis of the least 2\textsuperscript{nd} moment, the orientation of which is

\[
\theta = \text{atan2}\left(2 U_{xy}, \sqrt{(U_{xx} - U_{yy})^2 + (2U_{xy})^2}\right)
\]

- where the 2\textsuperscript{nd} central moments are

\[
U_{xx} = \sum_x \sum_y (x - \bar{x})^2 I(x, y) \quad U_{xy} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) I(x, y)
\]

\[
U_{yy} = \sum_x \sum_y (y - \bar{y})^2 I(x, y)
\]
Calculating Moments

- Let \((x_i, y_i)\) be all the points on the object in \(I\), then the summations can be more compactly represented as

\[
\sum_i x_i^m y_i^n = \sum_x \sum_y x^m y^n I(x, y)
\]

- Moments can be quickly calculated from the following summations:

\[
\begin{align*}
S &= \sum_i 1 \\
S_{xx} &= \sum_i x_i^2 \\
S_x &= \sum_i x_i \\
S_{yy} &= \sum_i y_i^2 \\
S_y &= \sum_i y_i \\
S_{xy} &= \sum_i x_i y_i
\end{align*}
\]
Calculating Moments (2)

- Moments are then given by

\[
A = S \\
\bar{x} = \frac{S_x}{S} \\
\bar{y} = \frac{S_y}{S} \\
U_{xx} = S_{xx} - \frac{S_x^2}{S} \\
U_{yy} = S_{yy} - \frac{S_y^2}{S} \\
U_{xy} = \left(S_{xy} - \frac{S_x S_y}{S}\right)
\]

- It is a simple matter to prove that these are equivalent to the definitions on the previous slide.
Central Moments

- **Spatial moments** taken from image origin

\[ M_{mn} = \sum_x \sum_y (x)^m (y)^n I(x, y) \]

- **Centralised moments** invariant to translation

\[ U_{mn} = \sum_x \sum_y (x - \bar{x})^m (y - \bar{y})^n I(x, y) \]
Normalised Moments

- Centralised moments invariant to translation

- Normalised moments invariant to translation and scale:

\[
N_{mn} = \frac{\sum_{x} \sum_{y} (x - \bar{x})^{m} (y - \bar{y})^{n} I(x, y)}{\left(\sum_{x} \sum_{y} I(x, y)\right)^{\frac{m+n+2}{2}}}
\]
Image Understanding

- Extract features (edges, boundaries, etc.)
- Then get understanding of the scene
Image Understanding (2)

Feature extraction
- Edges
- Spatial features
- Transform features
- Edges
- Shape features
- Moments
- Texture

Representation
- Thresholding
- Boundary modelling
- Clustering
- Quad-trees
- Texture modelling
- Skeleton

Interpretation
- Statistical
- Syntactical
- Decision trees
- Similarity measures
- Matching
Marr’s theory of vision – (D. Marr – *Vision* 1982)

- **Primitives are extracted from image**
  - Tokens
    - edge segments, blobs, bars, corners, groups of points

- **The primitives are organized in meaningful associations**
  - Sketches

- **Interpretation**
Sometimes, objects are perceived by humans from disconnected segments

Figure 6.18 Change of shape caused by a projective transform. The same rectangular cross-section is represented by different polygons in the image plane.
Image Understanding (5)

- But there are impossible objects too!
Image Understanding (6)
Boundary Representation

- Problems with existing edge detectors:
  - Edges are fragmented
  - The threshold is derived experimentally
- Boundaries are linked edges that characterize the shape of an object.
- Proper representation of object boundaries is important for analysis and synthesis of shapes.
- Boundaries can be found by tracing the connected edges.
Boundary Representation (2)

- In order to obtain connected edges, we replace the threshold with two thresholds.
- First apply a high threshold $t_1$ for segmenting clear edge segments.

![Gradient graph with thresholds $t_1$ and $t_2$.]

Result: A continuous edge

- Connect the edge segments in regions where the gradient exceeds a second threshold $t_2$. 
Boundary Representation (3)

- Contour following algorithms trace boundaries by ordering successive edge points considering local orientation.

Direction vectors used for chain coding in an eight pixel neighbourhood.

Freeman chain coding – effective for line image compression.

Resulting code: 100766555560000
Boundary Representation (4)

- **Freeman Chain Code**
  - Pixel from a boundary has its coordinates stored.
  - Pixels from the eight-connected neighbourhood are examined and the direction to the first detected boundary pixel is stored.
  - Then, the initial pixel is erased.
  - Continue with the next pixel.

- **Primitives chain code**
  - Extension of the Freeman code which additionally encodes more complex structures than a simple line orientation such as:
    - bifurcations
    - crosses
Boundary Representation (5)

We can model a boundary by splitting it in segments with certain local curvature characteristics.

Figure 6.14. Structural description of chromosomes by a chain of boundary segments, code word: a, b, a, b, c, a, b, d, b, a, b, c, a, b, a, b (adapted from [Fu 74]).
The simplest way of modelling an edge segment is by representing it with a straight line:

\[ y = ax + b \]

- A line is characterised by only two parameters: \( a \) and \( b \)
- An edge can be represented with a line by connecting its two end points and solving the two equation system for finding \( a \) and \( b \)

- This method does not assure the best fit for the intermediary points.
Boundary Representation (7)

- However, it provides a statistical way for calculating the best fit to all the points of the edge segment.
- Minimises the error of the fit with respect to one line variables either $x$ or $y$.

For $N$ points the model accuracy measure (error) is given by:

$$E_y = \sum_{i=1}^{N} (y_i - y)^2 = \sum_{i=1}^{N} [y_i - (ax_i + b)]^2$$

---

Line model

---

Original edge

---

Error
Boundary Representation (8)

The least mean squares solution is given by solving the system:

\[
\frac{\partial E}{\partial a} = 0 \quad \text{and} \quad \frac{\partial E}{\partial b} = 0
\]

Solution:

\[
a = \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i - N \sum_{i=1}^{N} x_i y_i \\
\sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i - N \sum_{i=1}^{N} x_i^2
\]

\[
b = \frac{1}{N} \left( \sum_{i=1}^{N} y_i - a \sum_{i=1}^{N} x_i \right)
\]

Similarly, we may consider the error measure:

\[
E_x = \sum_{i=1}^{N} (x_i - x)^2 = \sum_{i=1}^{N} \left[ x_i - \frac{1}{a} (y_i - b) \right]^2
\]
Boundary Representation (9)

- Minimises the error in the direction perpendicular to the fitted line

Line model

Original edge

Error

\[ E_{\perp} = \sum_{i=1}^{N} d_i^2 \]

Error evaluated in terms of the perpendicular distance between the point \((x, y)\) and the optimal line
Boundary Representation (10)

- Calculate the averages of the point coordinates (first order moments)

\[
\bar{x} = \frac{1}{N} \sum_{i=0}^{N} x_i \\
\bar{y} = \frac{1}{N} \sum_{i=0}^{N} y_i
\]

- Calculate the covariance matrix elements (second order moments)

\[
A = \begin{bmatrix}
\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 & \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) \\
\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) & \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y})^2
\end{bmatrix}
\]

- Calculate the eigenvalues (2) and eigenvectors (2)
- The eigenvector corresponding to the largest eigenvalue, passes through the centre of the data set and shows its orientation.
Boundary Representation (11)

Initial edges

Approximation by line segments

Main orientation of the data
Boundary Representation (12)

- We can use the Hough Transform to detect straight lines.
- A more general description:
  - A curve segment is defined by: endpoints, tangent vectors and continuity requirements
  - Parametric representation of a curve segment
  - For an n-dimensional polynomial need to know the curve in n points
Boundary Representation (13)

- For a cubic polynomial, we can calculate the perfect interpolation from

\[
\bar{P}_x = \begin{bmatrix} p_x(1) \\ p_x(2) \\ p_x(3) \\ p_x(4) \end{bmatrix} = \bar{T} \cdot \bar{C} = \begin{bmatrix} 1 & t_1 & (t_1)^2 & (t_1)^3 \\ 1 & t_2 & (t_2)^2 & (t_2)^3 \\ 1 & t_3 & (t_3)^2 & (t_3)^3 \\ 1 & t_4 & (t_4)^2 & (t_4)^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}
\]

- Chose three points $t_1$, $t_2$, $t_3$ and calculate the coefficients:

\[
\bar{C} = \bar{T}^{-1} \cdot \bar{P}_x
\]

- Matrix $\bar{T}$ should be invertible.