CV Notes

Alex Miles

December 2020

Contents

1 Eye and Human Vision

This section is non-examinable, and will be brief for the purpose of background.

1.1 Human Eye

- Evolved for survival
- Function of the eye is to form an image on the retina
- The lens is shaped using the ciliary muscle, it is not moved.
- The image is transmitted via the optic nerbe

Following image formation, the image must be inverted.

There are a variety of sensors: Cones (photopic 10^7) and Rods (10^8 scotopic). Cones have three different types, based on wavelength: Short (blue), Medium (Green), Long (red). The spectral response for each wavelength varies, with short wavelengths having a poor spectral response, medium wavelengths having the best spectral response, and long wavelengths having an interpolated response between short and medium wavelengths.

1.2 Mach Bands

Mach bands are a result of brightness adaption, and occur due to the eye forming boundaries between different shades of a distinct black-white colour gradient. The seen response overshoots between colour bands and a border appears between the elements in the colour gradient.

1.3 Neural Processing

Sensor information is combined by using a variety of weighting functions. Given a weighting function w, sensor P_1, and a variety of other sensors $P_2, ...P_N$, we can simulate processing:

$$P_1 -> log(P_1) -> w_1 \times log(P_1) \tag{1}$$
$$P_n -> log(P_n) -> w_n \times log(P_n) \tag{2}$$
$$\sum_{1}^{n} w_n \times log(P_n) \tag{3}$$

Webers law, the difference threshold, is the minimum amount by which stimulus intensity must be changed is the minimum amount by which stimulus intensity must be changed in order to produce a variation in sensory experience.

1.4 Summary

1 The human eye can be modelled in three sections

2 It is very effective but can be deceived

3 Is it a good model for computer vision?

2 Image Formation

2.1 Image Decomposition

Images can be decomposed into their individual bits, from bit 0 to bit 7. The most significant bit carries the most information, and as the bits decrease in significance, more noise is introduced. Some bits are used for

lighting.
High resolution images contain more information, and implies that more storage is required, given $N \times N$ points. An appropriate value of N is determined by use case.

2.2 Fourier Transform

Essentially, the Fourier transforms an addition of signals from the time domain to the frequency domain. Each individual frequency from the addition of signals in the time domain, is visible in the frequency domain.

We can describe the fourier transform as follows:

$$Fp(f) = \mathcal{F}(p(t)) = \int_{-\infty}^{\infty} p(t)e^{-jft}dt \tag{4}$$

2.2.1 Derivation

An explanation of 4 is given through the following derivation, based on the fact that: a() is a function of time variant signal p(t):

$$\text{The FT is then } Fp = a(p(t))$$

$$\text{FT is a function of frequency } Fp(f) = a(p(t))$$

$$\text{This can be written as: } \mathcal{F}(p(t)) = a(p(t))$$

$$\text{a() is an integral } \mathcal{F}(p(t)) = \int_{-\infty}^{\infty} p(t)cas(t)dt$$

$$cas = \text{cos and sin } e^{-jft} = cos(ft) - jsin(ft)$$

$$\mathcal{F}(p(t)) = \int_{-\infty}^{\infty} p(t)e^{-jft}dt$$

2.2.2 Application

Given a pulse signal, the following evaluation can be made. We can apply the piecewise pulse function, and evaluate the integral given the conditions of the pulse.

$$p(t) = \begin{cases} A & \text{if } -T/2 \leq t \leq T/2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\mathcal{F}p(\omega) = \int_{T/2}^{-T/2} Ae^{-jwt}dt \tag{6}$$

$$\mathcal{F}p(\omega) = -\frac{Ae^{-jw\frac{T}{2}} - Ae^{jw\frac{T}{2}}}{j\omega} \tag{7}$$

$$\mathcal{F}p(\omega) = \begin{cases} \frac{2A}{\omega}\sin\frac{\omega T}{2} & \text{if } \omega \neq 0 \\ AT & \text{if } \omega = 0 \end{cases} \tag{8}$$

This is demonstrated visually in: (1)

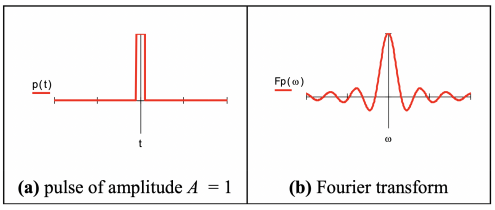


Figure 1: Fourier transform of a pulse

The Fourier transform signal can be reverted back to the time domain using the inverse Fourier transform. We can select a variety of angular frequencies: $range(-6, 6)$. The reconstruction by integration is then:

$$F(t) = \int_{-6}^{6} \mathcal{F}p(\omega)e^{jwt}d\omega \tag{9}$$

2.2.3 Magnitude and Phase of Fourier Transform

We can split the Fourier transform by its real and imaginary parts:

$$(F)p(\omega) = \int_{-\infty}^{\infty} p(t)e^{-jwt}dt = Re(\mathcal{F}p(\omega)) + jIm(\mathcal{F}p(\omega)) \tag{10}$$

From this, we can then proceed to calculate the magnitude and the phase.
Magnitude

$$|\mathcal{F}p(\omega)| = \sqrt{Re(\mathcal{F}p(\omega))^2 + Im(\mathcal{F}p(\omega))^2} \tag{11}$$

Phase

$$arg(\mathcal{F}p(\omega)) = \arctan\left(\frac{Im(\mathcal{F}p(\omega))}{Re(\mathcal{F}p(\omega))}\right) \tag{12}$$

2.2.4 Additional

The Fourier transform is used for coding images. We can quantise the frequency transform, then code the information from the quantised image. We can then reconstruct the image, to get a sampled result.

3 Image Sampling

3.1 Sampling Signals

Given a continuos signal, an image is well sampled if from the sample points alone, an accurate representation of the original signal can be created. Bad sampling would describe a different signal upon creation.
If we were to oversample, lets use the example of a rotating wheel, the wheel would appear to be rotating slowly. If the sampling rate was not high enough, the wheel would be appear to be moving in the opposite direction.
Consider the frequency domain, the spectra repeat, if the sampling rate is correct, the spectra will just touch. We can deduce from this that:

$$Sample_{min} = 2 \times f_{max} \tag{13}$$

This follows *Nyquists sampling theorem*, which states:

> In order to be able to reconstruct a singal from its samples, we must sample at minimum twice the maximum frequency in the original sample.

A simple example of this would be, speech averages at $6kHz$, we sample at $12kHz$. If we compare this to something like *.wav* audio files, they are often sampled at 24kHz upon a spectral analysis. In image terms; two pixels for every pixel of interest.

3.2 1D Discrete Fourier Transform

Discrete Fourier calculates frequency from data points.

$$\mathcal{F}p(\omega) = \int_{-\infty}^{\infty} p(t)e^{-jwt}dt \tag{14}$$

$$\mathcal{F}p_u = \frac{1}{N}\sum_{i=0}^{N-1} p_i e^{-j\frac{2\pi}{N}iu} \tag{15}$$

$$\tag{16}$$

Where:

$$\text{Sampled Frequency: } \mathcal{F}p_u \tag{17}$$

$$\text{Sampled Points: } p_i \tag{18}$$

$$N \text{ points} \tag{19}$$

Again, for a 10 point sample ($N = 10$), we can reconstruct these signals by adding all frequency components:

$$\mathcal{F}p_6 = \sum_{u=0}^{9} \mathcal{F}p_u \times e^{jt\frac{2\pi}{10}u} \tag{20}$$

3.3 2D Discrete Fourier Transform

The forward and inverse transforms are given:

$$\mathcal{F}\mathbf{P}_{\mathbf{u},\mathbf{v}} = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{\mathbf{x},\mathbf{y}}e^{-j(\frac{2\pi}{N})(ux+vy)} \tag{21}$$

$$\mathcal{F}^{-\mathbf{1}} = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\mathcal{F}\mathbf{P}_{\mathbf{u},\mathbf{v}}e^{j(\frac{2\pi}{N}(ux+vy)} \tag{22}$$

two dimensions of space, x and y (23)

two dimensions of frequency, u and v (24)

image NxN, pixels each being $\mathbf{P}_{\mathbf{x},\mathbf{y}}$ (25)

Shift Invariance Upon applying a (positional) shift to an image, the magnitude and phase of the image are invariant to the shifted image. This is only true if all pixels overflow as a result of a positional shift.

We can apply a few different transforms after applying the 2D Fourier transform to an image. For example rotation and filtering, mathematically we can rotate an image 90 degrees with the following transform.

$$\mathcal{F}\mathbf{P}_{\mathbf{u},\mathbf{v}} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{P}_{\mathbf{x},\mathbf{y}}e^{-j(\frac{2\pi}{N})(ux+vy)} \tag{26}$$

Given we have access to all frequencies for an image, we can filter out certain frequencies to create a low pass filtered image. This can then be used for different applications such as creating hybrid images.
Some other transforms include: Fourier transform magnitude, discrete cosine transform, hartley transform.

3.3.1 Applications of 2D Fourier Transform

- Understanding and analysis
- Speeding up algorithms
- Invariance
- Coding
- Recognition of frequency (e.g. texture)

4 Point Operators

4.1 Histogram Representation

We can display the contrast of an image by simply plotting a histogram of the pixels with a certain brightness value, and the frequency.
As a result, we can change the contrast by applying a brightness function:

$$N_{x,y} = k \times \mathbf{O}_{x,y} + l \tag{27}$$

$N_{x,y}$ New image, pixels x and y (28)

$\mathcal{O}_{x,y}$ old image (29)

gain k (30)

level l (31)

4.2 Logarithmic Compression...

We can apply logarithmic compression and exponential expansion.

$$N_{x,y} = log(\mathcal{O}_{x,y}) \tag{32}$$

$$N_{x,y} = e^{(\mathcal{O}_{x,y})} \tag{33}$$

4.3 Intensity Normalisation

The aim is to use all available grey levels for display. We can do this using the following process:

1 Take the original histogram

2 Shift the origin to zero

3 Scale brightness to use the whole range

This can be described using the following equation:

$$N_{x,y} = \frac{N_{max} - N_{min}}{O_{max} - O_{min}} \times (O_{x,y} - O_{min}) + N_{min} \tag{34}$$

$$N_{x,y} = \frac{256}{O_{max} - O_{min}} \times (O_{x,y} - O_{min}) \tag{35}$$

The second equation is used to avoid the need of parameter selection.

4.4 Histogram Equalisation

We can flatten a histogram, which is aimed for human vision to show more detail (e.g. for medical applications).

N^2 points, the sum of points per level is equal	$\sum_{l=0}^{M} \mathbf{O}(l) = \sum_{l=0}^{M} N(l)$
cumulative histogram up to level p transformed	$\sum_{l=0}^{p} \mathbf{O}(l) = \sum_{l=0}^{q} N(l)$
Number of points per level in output picture	$N(l) = \frac{N^2}{N_{max} - N_{min}}$
cumulative histogram of output	$\sum_{l=0}^{q} N(l) = q \times N(l) = \frac{N^2}{N_{max} - N_{min}}$
mapping for the output pixels at level q	$q = \frac{N_{max} - N_{min}}{N^2} \times \sum_{l=0}^{p} \mathbf{O}(l)$

This is nonlinear and there are major problems with noisy images.

4.5 Thresholding

Thresholding selects points that exceed a chosen threshold

$$N_{x,y} = \begin{cases} 255 & \text{if } N_{x,y} > threshold \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

4.6 Summary

1 point operators are largely about image display

2 histogram manipulation

3 thresholding is used a lot

4 intensity normalisation is used for medical display

5 Group Operators

5.1 Template Convolution

We can calculate a new image from the original, by applying an **inverted** template in raster fashion.

5.1.1 Weighting Coefficients

Given template T, where T is a 3×3 matrix consisting of 9 weighting coefficients.

$$T = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \tag{37}$$

We can calculate a result for the centre pixel of the image we are applying the template to, by calculating the result of the following equation:

$$N_{x,y} = \sum_{i \in template} \sum_{j \in template} w_{i,j} * \mathbf{O}_{x(i),y(j)} \tag{38}$$

We then have an output image consisting of all the results from each convolution. This is obviously of a smaller size than the original image. This leaves us with three options:

1 Set the border pixels to black (image usually focused in center)

2 Assume pixels wrap around

3 Reduce template size near edges

This leads us to a variety of operators which can be described as a template.

5.1.2 3x3 Averaging Operator

The purpose of an averaging operator is to remove noise, and comes with a cost to detail. The template is described as $T_{averaging}$

$$T_{averaging} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \tag{39}$$

And we can calculate the new image, setting the border to black, with the following equation:

$$N_{x,y} = \frac{1}{9} \sum_{i \in 3} \sum_{j \in 3} \mathbf{O}_{x(i),y(j)} \tag{40}$$

5.1.3 Non-symmetric templates

The template is flipped around on both axes, (inverse template). For deep learning templates this is not necessary as they are usually symmetric.
For a non symmetric template, the convolution is performed as follows:

$$O \times T = \sum_{(x,y) \in W} I_{x,y} T_{x-i,y-j} \tag{41}$$

5.1.4 Template Convolution via Fourier Transform

The convolution theorem allows for fast computation via the FFT (Fast Fourier Transform) for templates $size(T) \geq 7$.

This can be described via:

$$P * T = \mathcal{F}^{-1}(\mathcal{F}(P). \times \mathcal{F}(T)) \tag{42}$$

$$\tag{43}$$

$$\text{Template convolution } * \tag{44}$$

$$\text{Fourier transform of the picture } \mathcal{F}(P) \tag{45}$$

$$\text{Fourier transform of the template } \mathcal{F}(T) \tag{46}$$

$$\textbf{Point by point multiplication for sampled signals } (.\times) \tag{47}$$

5.2 2D Gaussian Function

The Gaussian function can be used to calculate template values. A commonly used example of this would be Gaussian blur. There is a compromise between the variance and the window size.

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{48}$$

Given a σ value of 1, we can form a 5×5 template which has the following visualisation.

0.002	0.013	0.022	0.013	0. 002
0.013	0.060	0. 098	0. 060	0.013
0.022	0. 098	0.162	0. 098	0.022
0.013	0. 060	0.098	0. 060	0.013
0. 002	0.013	0.022	0.013	0. 002

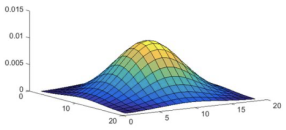


Figure 2: Visualisation

Gaussian averaging is better to a human eye than direct averaging.

5.3 Median of a Template

To find the median of a template we can perform the following operation.

$$\text{Original Matrix } \begin{bmatrix} 2 & 8 & 7 \\ 4 & 0 & 6 \\ 3 & 5 & 7 \end{bmatrix} \tag{49}$$

$$\text{Flattened Vector } \begin{bmatrix} 2 & 8 & 7 & 4 & 0 & 6 & 3 & 5 & 7 \end{bmatrix} \tag{50}$$

$$\text{Sorted Vector } \begin{bmatrix} 0 & 2 & 3 & 4 & \mathbf{5} & 6 & 7 & 7 & 8 \end{bmatrix} \tag{51}$$

The median is the centre element of a rank ordered set of template points. We preserve the edges and remove salt and pepper noise.

6 Edge Detection

What is an edge? An edge is contrast. We can visualise this by calculating the sobel edge magnitude and thresholded magnitude of an image.

6.1 Differencing

Horizontal differencing detects vertical edges, and vertical differencing detects horizontal edges. We can perform first order edge detection by adding the horizontal and vertical edges. This can be described mathematically as shown:

$$Evert_{x,y} = |P_{x,y} - P_{x+1,y}| \tag{52}$$

$$Ehrzn_{x,y} = |P_{x,y} - P_{x,y+1}| \tag{53}$$

$$E_{x,y} = |2 \times P_{x,y} - P_{x+1,y} - P_{x,y+1}| \tag{54}$$

We can use the coefficients of this equation to form the following template. However it is unnecessarily complex for this use.

$$T = \begin{bmatrix} 2 & -1 \\ -1 & 0 \end{bmatrix} \tag{55}$$

6.1.1 Uniform Thresholding

We can select the brightest edge points using uniform thresholding. We select a threshold level to control the number of points.

6.2 Analysis of Basic Operators

Taylor series analysis reveals that differencing the adjancent points provides an estimate of the first order derivative at a point.
If we separate points by Δx, then by the taylor expansion for $f(x + \Delta x$ is given by:

$$f(x + \Delta x) = f(x) + \Delta x + f'(x) + \frac{\Delta x^2}{2!} \times f''(x) + \mathbf{O}(\Delta x^3) \tag{56}$$

$$f'(x) = \frac{f(x + \Delta x) - f(x)}{\Delta x} - \mathbf{O}(\Delta x) \tag{57}$$

The difference between two adjacent points is an estimate of the first order derivative with an error of $O(\Delta x)$ The error is large when the interval size is large. The error is large if the curve is complex.

Close sampling is required for adequate approximation. We can reduce the error by spacing the differenced points by one pixel. This equivalent to computing the first order difference at two adjacent points as a new horizontal difference **Exx**.

$$\mathbf{Exx}_{x,y} = |P_{x+1,y} + P_{x-1,y}| \tag{58}$$

We can analyse this by taylor series and expand $f(x - \Delta x)$. We obtain the first order derivative through rearranging as:

$$f'(x) = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} - \mathbf{O}(\Delta x^2) \tag{59}$$

The error is now $O(\Delta x^2)$ Leaving us with the templates for horizontal and vertical first order difference.

$$\text{Mx} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \tag{60}$$

$$\text{My} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{61}$$

6.3 Prewitt Operator

Edge detection is similar to differentiation, as it detects change, it is also bound to noise. Therefore we should also apply averaging to reduce noise with the detection process.

Which leaves us with the following templates. By extending Mx and My for the horizontal and vertical templates.

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Table 1: Templates for Prewitt Operator

This returns the rate of change of brightness along each axis. The edge magnitude, and edge direction.

$$M(x,y) = \sqrt{Mx(x,y)^2 + My(x,y)^2} \tag{62}$$

$$\theta(x,y) = \arctan(\frac{My(x,y)}{Mx(x,y)}) \tag{63}$$

6.3.1 Implementation

When applied to the image of a square, we obtain the edge magnitude and direction.

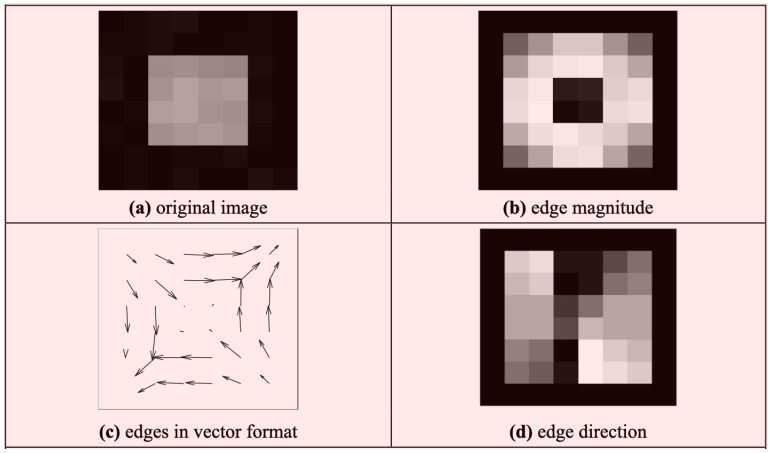


Figure 3: Applied Prewitt Operator

Template convolution could be applied here, but it is not necessary.

6.4 Sobel Edge Detection

When the weight at the central pixels for both Prewitt templates is doubled, two templates for the Sovel edge detection are formed.

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Table 2: Templates for Prewitt Operator

The implementation is similar, however we can use larger templates to introduce more smoothing, noise reduction, but edge blurring becomes a problem.

Given that **Gaussian averaging produces optimal averaging**, we can use the binomial expansion to approximate the gaussian normal distribution. It is essential a gaussian operator with integer coefficients. To generalise sobel, we use the differencing coefficients in a binomial expansion.

6.4.1 Evaluation of Sobel

- Sobel is a good basic operator
- - Blurred edges
- - Noisy edges

6.5 Canny Edge Detection

Canny gives thin edges in the correct place but it is more complex.
The general process for canny edge detection is as follows:

1 Gaussian smoothing to reduce noise

2 Sobel edge detection

3 Reduce blurred edges and noise through non maximum suppresion

4 Hysteresis thresholding to connect edge points

6.5.1 Operator

Objectives of the Canny Edge Detector

- Optimal detection without spurious responses
- Good localisation with minimal distance between edge detected and the true edge.
- Single response to eliminate multiple responses to a single edge.

We 'walk' along the top of the edge, to give thin edges in the right place.

6.5.2 Interpolation in non-maximum Suppression

Need to use points which are not on the image grid, we can do this through linear interpolation.

6.5.3 Hysteresis Thresholding Transfer Function

Lower threshold = average noise
Upper threshold = average feature boundary.

6.6 Comparisons

Hysteresis thresholding gives all points > upper threshold, plus any connected points > lower threshold.
We get less noise, and better *thinner* edges.

6.7 Edge Detection via Laplacian Operator

We can describe the second order (laplaian of gaussian) edge detection process mathematically as follows:

$$g(x, y, \sigma) = e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{64}$$

$$\frac{\delta g(x, y, \sigma)}{\delta x} = -\frac{x}{\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{65}$$

$$\frac{\delta^2 g(x, y, \sigma)}{\delta^2 x^2} = (\frac{x^2}{\sigma^2} - 1)\frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{\sigma^2} \tag{66}$$

$$\text{Given that: } \nabla^2 g(x, y, \sigma) = \frac{\delta^2 g(x, y, \sigma)}{\delta^2 x^2} U_x + \frac{\delta^2 g(x, y, \sigma)}{\delta^2 y^2} U_y \tag{67}$$

$$= (\frac{x^2}{\sigma^2} - 1)\frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{\sigma^2} + (\frac{y^2}{\sigma^2} - 1)\frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{\sigma^2} \tag{68}$$

$$\nabla^2 g(x, y, \sigma) = \frac{1}{\sigma^2}(\frac{x^2 + y^2}{\sigma^2} - 2) \times \mathrm{e}^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{69}$$

6.8 Zero crossing detection

Need to find zero-crossings in 2D.

$$IF(max(1, 2, 3, 4) > 0(1, 2, 3, 4) < 0) f(x, y) = edge \tag{70}$$

6.9 Marr-Hildreth Edge Detection

Given the second order laplacian of gaussian edge detection, we can detect local features with a small template. We use a larger template to detect global features.

7 Finding Shapes

7.1 Thresholding and Subtraction

7.1.1 Thresholding

We can use thresholding for basic feature extraction, if the shape to be extracted is defined by its brightness. However we do not obtain the full shape from thresholding if the intensity levels are inconsistent.

Occlusion is the covering of a feature within an image.

We can summarise thresholding and describe its advantages and disadvantages:

- The threshold must adapt to illumination changes. Otherwise it will fail
- Thresholding does not require much computational effort
- We can use histogram equalisation if the illumination level changes linearly. However noise can affect the images dramatically.
- Intensity normalisation is less sensitive to noise, which is suitable for thresholding.
- **Optimal in occlusion**
- **Optimal in noise**

7.2 Template Matching

Template matching is a model based approach, where the shape is extracted by searching for the best correlation between a known model and the pixels in an image. Correlation can be implemented by considering the frequency domain. The template can be defined by considering intensity values or a binary shape.

In practice, we take an image, and a template, and use a accumulator to store a count of matching points. We can improve this process by using edges.

Generally, we can describe template matching as:

- Intuitively simple
- Correlation and convolution
- Implemented through Fourier
- Relationship with matched filter

7.3 Convolution and Correlation

Convolution is the application of a given template. For non symmetric kernels, it involves flipping the template. We can also achieve convolution by multiplying the transforms.

$$O \times T = \sum_{(x,y)\in W} I_{x,y} T_{x-i,y-j} \tag{71}$$

$$O \times T = \mathcal{F}^{-1}(\mathcal{F}(I). \times \mathcal{F}(T)) \tag{72}$$

Correlation is about the matching of a template. We flip the fourier template.

$$O \otimes T = \sum_{(x,y)\in W} I_{x,y} T_{x+i,y+j} \tag{73}$$

$$O \times T = \mathcal{F}^{-1}(\mathcal{F}(I). \times \mathcal{F}(-T)) \tag{74}$$

7.4 Hough Transform

Performance is the same as template matching, but is faster. We define a mapping from the image points to Hough space (an accumulator). Essentially, we are trying to find alligned points to create lines.

A line is points $P_{x,y}$, with gradient m and intercept c.
The intercept is $c = -xm + y$.

Consider the plane m,c, the feature space. This defines a lot of possibilities. For each point we determine points of intersection with the original points, and plot these in the feature space. This is flawed however as the parameters m, c tend to infinity. We can transform the parameterisation of the line from:

$$c = -xm + y \tag{75}$$

$$\text{to: } \rho = x\cos\theta + y\sin\theta \tag{76}$$

We can now iterate through each point, row by row in an edge detected image. Once a point is found, we rotate the line until an alignment is found, and store this in the feature space (accumulator). We iterate to the max pixel in the image, and have a feature space, where the intersections of the lines detail the parameters at which there is an alignment. These intersects which detail the parameters are shown in 4.

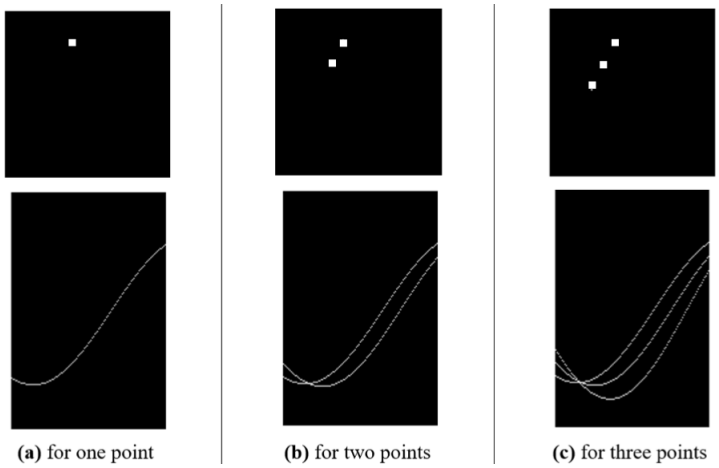


Figure 4: Hough Space

However, shapes are more complex than lines and sometimes cannot be defined by equations.

7.5 Hough Transform for Complex Shapes

Given a circle, we can find shapes (e.g. pupil) with the hugh transform. This is difficult in practice as most shapes are not exactly a circle. Again lets transform the equation to polar space.

$$(x-a)^2 + (y-b)^2 = r^2 \tag{77}$$

$$r = 2a\cos\theta + 2b\sin\theta \tag{78}$$

$$\text{Where:} \tag{79}$$

$$\text{r - radius of circle} \tag{80}$$

$$\text{a - x coordinate of center} \tag{81}$$

$$\text{b - y coordinate of center} \tag{82}$$

The same process of iteration can be applied, and the peaks used to determine points of interest.

7.5.1 Addition: Elipse

An elipse can be described mathematically:

$$\frac{(x-x0)^2}{a^2} + \frac{(y-y0)^2}{b^2} = 1 \tag{83}$$

We can see that we have 4 separate parameters. If we have 100 values, and add rotation, we have a huge amount of values stored in accumulator space. This motivates approaches to save memory and improve speed.

7.5.2 3D Accumulator

Lets take the circle equation:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{84}$$

We can differentiate the following equation, and substitute it back into the equation:

$$\frac{\delta y}{\delta x} = -\frac{(x-x_0)}{(y-y_0)} \tag{85}$$

$$(\frac{\delta y}{\delta x})^2(y-y_0)^2 + (y-y_0)^2 = r^2 \tag{86}$$

$$y - y_0 = \frac{r}{\sqrt{1 + (\frac{\delta y}{\delta x})^2}} \tag{87}$$

7.5.3 Arbitrary Shapes

We use the generalised hough transform, form a discrete R table, and vote via the look up table.

We start from a reference point within the shape, and determine different edge vectors.

8 Applications and Deep Learning

This section is not really covered in the exam. To be added to:

8.1 Features at Different Levels

We can show the features by different levels for use in a deep learning model.

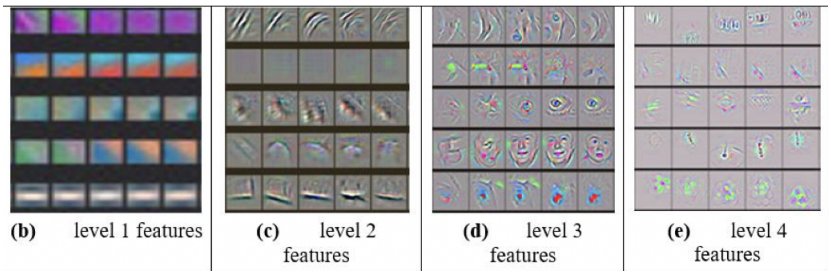


Figure 5: Features at Different Levels

8.2 Summary

- Computer vision works and has a great future
- Big difference between hand crafted and deep learning
- Some parts are the same, e.g. group operators/template

9 Building Machines That See

9.1 Key Terms in Designing CV systems

Simply we can describe the key elements of a CV system by the following terminology.

- robust and repeatable - what we want
- invariant - designed to be invariant
- constraints - what you apply to make it work

9.1.1 Robustness

The vision system must be robust to changes in its environment.
changes in lighting, angle or position of camera.

9.1.2 Repeatability

Repeatability is a measure of robustness. The system must work consistently the same, regardless of environmental changes.

9.1.3 Invariance

Invariance to environmental factors helps to achieve robustness and repeatability. Hardware and software can designed to be invariant to certain environmental changes.

9.1.4 Constraints

Constraints are what you apply to the hardware, to make the system work in a repeatable and robust fashion. We can constrain a system by putting it in a box so there cannot be an illumination changes.

Software Constraints using really simple but incredibly fast algorithms. We can also use an intelligent use of colour.

Physical Constraints industrial vision is usually solved by applying simple computer vision algorithms, and lots of physical contraints.

- Environment: lighting, enclosure, mounting
- Acquisition hardware: expensive camera, optics, filters

10 Machine Learning for Pattern Recognition

10.1 Machine Learning Process

The process for a computer vision application takes the following form:

- Image Input
- Feature Extraction - to Feature Vectors
- Machine Learning - uses Feature Vectors
- Classification

10.2 Feature Vectors

A mathematical vector, consisting of real numbers, with a fixed size. The dimensionality of the vector is the number of elements.

The featurevector represents a point in a feature space, or a direction in the feature space.

The dimensionality of a featurespace, is the dimensionality of every subvector.

Vectors of differing dimensionality can't exist in the same featurespace.

10.2.1 Distances in *featurespace*

Feature extractors are defined so that they produce vectors that are close together for similar inputs. The closeness of two vectors can be computed by calculating the distance between the vectors.

L2 Distance is the most intuitive distance, the straight line distance between two points. This is computed via an extension of Pythagoras theorem.

$$D_n(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{88}$$

$$D_2(p,q) = \sqrt{\sum_{i=1}^{2}(p_i - q_i)^2} = \sqrt{(p-q)(p-q)} \tag{89}$$

L1 Distance (Manhattan/Taxicab) is computed along paths parallel to the axes of the space.

$$D_1(p,q) = \sum_{i=1}^{n} = ||p-q||_1 \tag{90}$$

Cosine Similarity measures the cosine of the angle between two vectors. It is not a distance.

$$\cos\theta = \frac{pq}{||p||||q||} \tag{91}$$

$$\cos\theta = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \tag{92}$$

10.2.2 Featurevector Choices

It is important to choose features which allow to distinguish objects or classes of interest. Keeping the number of features is also important, as ML becomes more difficult with increased dimensionality of the feature space. (Note dimensionality reduction)

10.3 Classification

Classification is the process of assigning a class label to an object. A supervised machined learning algorithm uses a set of prelabelled training data to learn how to assign class labels to vectors.

10.3.1 Linear Classifiers

Linear classifiers try to divide a feature space using a linear hyperplane which separates two classes. The aim is to find a suitable hyperplane, and do this with minimum classification error.
Given an image to classify, we just need to check which side of the hyperplane it is on.

10.3.2 Non-linear Binary Classifiers

Linear classifiers work when the data is linearly separable. We can use non linear binary classifiers such as kernel support vector machines to learn non-linear decision boundaries. We lose generality of the classifier if we overfit using a polynomial of too high degree.

10.3.3 KNN

We assign a class to an unknown point, based on the majority class of the closest K neighbours in a feature space.

10.4 Clustering

We aim to group data without any prior knowledge of what each group contains. We group items by the similarity of each feature vector. Some clustering operations can create overlapping groups, however for this purpose we are only looking at disjointed clustering methods such as K-means clustering.

10.4.1 K-Means Clustering

K-means is a featurespace clustering algorithm for grouping data into K distinct groups where each group is represented by its *centroid*.

1 The number of groups, K is selected

2 K initial cluster centres are chosen.

- Each point is assigned to its closest centroid
- The centroid is recomputed as the mean of all the points assigned to it. If the centroid has no points assigned to it, we reinitialise the centroid to a new point.

3 We cluster by assigning all points to their nearest centroids

11 Variance and Covariance

11.1 Random Variables and Expected Values

Variance and covariance are usually described in terms of random variables and expected values.

A random variable is a variable that takes on different values due to a probability

The set of sample values from a single dimension of a feature space can be considered to be a random variable.

The expected value $E(X)$ is the most likely value a random variable will take.

We assume for the purpose of this course that the values an element of a feature can take, are all equally likely.

For the exam: $E(X) = mean$

11.2 Variance

Variance, denoted by (σ^2) is the mean squared difference from the mean (μ). Essentially, *how spread out the data is from the mean*

$$\sigma^2(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{93}$$

$$(x_i - \mu)^2 \text{ Squared difference for a single point and the mean} \tag{94}$$

$$\sum_{i=1}^{n}(x_i - \mu)^2 \text{ Calculate squared difference for every discrete point} \tag{95}$$

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \text{ Divide by total number of points to find variance} \tag{96}$$

11.3 Covariance

Covariance, denoted by ($\sigma(x, y)$ is the measure of how two variables change **together**.

$$\sigma(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x - \mu_x)(y - \mu_y) \tag{97}$$

The variance is equal to the covariance when the two variables are the same. A covariance value of 0 means that the variables are uncorrelated.

11.3.1 Covariance Matrix

A covariance matrix encodes the formula for a given n dimensional dataset through every combination of pairs.

$$\begin{bmatrix} \sigma(X_1, X_1) & \sigma(X_1, X_2) & \dots & \sigma(X_1, X_n) \\ \sigma(X_2, X_1) & \sigma(X_2, X_2) & \dots & \sigma(X_2, X_n) \\ \dots & \dots & \dots & \dots \\ \sigma(X_n, X_1) & \sigma(X_n, X_2) & \dots & \sigma(X_n, X_n) \end{bmatrix} \tag{98}$$

A covariance matrix is **square** and **symmetric**.

11.4 Mean Centring

We compute the mean across each dimension independently of a set of vectors, then subtracting the mean vector from every vector in the set. All vectors are translated so that the average position is the origin.

11.5 Principal Axes of Variation

A basis is a set of n linearly independent vectors in an n dimensional space. The vectors are infinite, orthogonal (dot product = 0), and they form a coordinate system.

11.5.1 The first principal axis

For a given set of n dimensional data, the first principle axis is the vector that describes the direction of the greatest variance.

11.5.2 The second, third.. principal axis

A vector perpendicular to the first major axis. The following principal axes are of the greatest variance orthogonal to both the first and second principal axi...

11.6 Eigenvectors and Eigenvalues

There are at most n eigenvector-eigenvalue pairs. If **A** is symmetric, then the set of vectors is orthogonal.

$$\mathbf{A}v = \lambda v \tag{99}$$

- **A** - NxN square matrix
- v - n dimensional vector (eigenvector)
- λ - a scalar value (eigenvalue)

If **A** is a covariance matrix, then the eigenvectors are the principal axes.
λ is proportional to $\sigma - variance$ of the data along each eigenvector.

The eigenvector corresponding to the **largest** eigenvalue is the first principle component axis.

11.6.1 Finding λ and v

For larger matrices, numerical solutions t the eigendecomposition must be sought. Eigenvectors are the columns of **Q**.

$$A = Q \wedge Q^{-1} \tag{100}$$
$$\wedge_{ii} = \lambda_{ii} \tag{101}$$

If **A** is real symmetric (i.e. a covariance matrix) then

$$Q^{-1} = Q^T \tag{102}$$
$$A = Q \wedge Q^T \tag{103}$$

Gives you the principal axes and their relative magnitudes.

11.7 Linear Transform

A linear transform **W** projects data from one space into another:

$$T = ZW \tag{104}$$

The original data is stored in the rows of Z.

T can have fewer dimensions than Z, if W is a dimensionality reduction transform.

11.7.1 Inverse Linear Transform

$$Z = TW^{-1} \tag{105}$$

The effects of a linear transform can be reversed if **W** is invertible.

11.8 PCA

PCA is an Orthogonal Linear Transform that maps data from its original space to a space defined by the principal axes.

- The transform matrix **W** is just the eigenvector matrix **Q** from the eigendecomposition of the covariance matrix of the data.
- In addition we can reduce the dimensions by removing low ranking eigenvectors (by eigenvalue)

To summarise the maths:

$$W = Q \wedge Q^T \tag{106}$$

*The transform matrix **W** is just the eigenvector matrix **Q** from the eigendecomposition of the covariance matrix of the data.*

11.8.1 PCA Algorithm

1 Mean centre the data vectors

2 Form the vectors into a matrix Z, such that each row corresponds to a vector

3 Perform eigendecomposition, given that the covariance matrix is symmetric we get $Q \wedge Q^T = Z^T Z$.

4 Sort the columns of **Q** and corresponding diagonal values of $\wedge$ so that the eigenvalues are decreasing.

5 Reduce dimensions by selecting L largest eigenvectors of **Q** to create Q_l.

6 Project the original vectors into a lower dimensional space $T_L = ZQ_l$

12 Image Features and Segmentation

12.1 Image Feature Morphology

Upon feature extraction, feature vectors are a product. The differing morphologies of feature extractor, output a varying numbers of features.

If we were to analyse an image as one, for an output of one featurevector. We would describe this as a **Global Feature**.

12.1.1 Grid or Block-based Features

The image is passed to a feature extractor, the image is split by a NxN grid, and **one** feature is extracted from **each block**.

12.1.2 Region-based Features

The image is split into a number of regions, and a feature is extracted for each region. **One** feature is extracted **per region**.

12.1.3 Local Features

The image is analysed by the feature extractor, points of interest are detected, and as a result a feature is extracted from the surrounding pixels. **One** feature **local interest point**.

12.2 Global Features

This section details methods and concepts of finding global features within an image.

12.2.1 Histograms

Image Histograms A common approach to computing a global image description, is to compute a histogram of the pixel values.

Joint-colour Histograms measures the frequency of occurency of each colour in the image. We compute a separate histogram for each channel. The colourspace is quantised into bins, and we accumulate the number pixels in each bin.

12.3 Segmentation

Segmentation can be defined as:

> The process of partitioning the image into sets of pixels, often called segments. Pixels within a segment often share visual characteristics

12.3.1 Global Binary Thresholding

Thresholding is the simplest form of segmentation.

- Turns grey level images into binary images, by assigning pixels with a value less than a threshold to one segment, and other pixels to another.
- This method is very fast
- Requires a manually set threshold. This is not robust to lighting changes. But works well with constraints.

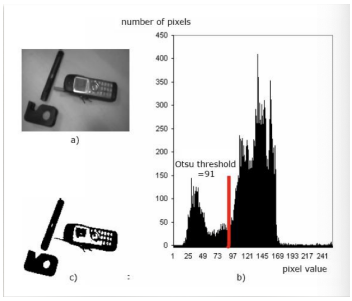
!h 

Figure 6: Otsu Thresholding

12.3.2 Otsu's Thresholding Method

Similar conceptually to global binary thresholding however provides a way to automatically find the threshold.

- Assume there are two classes, foreground and background.
- Therefore, the histogram must have two peaks.
- Exhaustively search the histogram that maximises the interclass value.

This is visualised in figure 6
We can see in the example, that a threshold is found at a pixel value of 91.

12.4 Local Features

Local or adaptive thresholding operators compute a different threshold value for every pixel in an image, based on the surrounding pixels. We use a rectangular window around a pixel to define the neighbours.

12.4.1 Mean Adaptive Thresholding

The MAT local feature method works as follows:

Statement Set the current pixel to zero if
Condition The value is less than the $mean + constant$ of the neighbours in a given window
Otherwise set to one

We can evaluate the pros and cons

Pros Good invariance to uneven lighting
- Good invariance to a variety of contrasts

Cons Computationally expensive as its pixel by pixel
- Can be difficult to choose the window size
- If the object being imaged can appear at different distances to the camera, then it could break.

12.4.2 Segmentation with K-Means Clustering

K-Means clustering provides a simple method for performing segmentation.
We start by clustering the colour vectors of all the pixels, then assign each pixel to a segment based on the closest cluster centroid.
The Naive approach to segmentation using K-Means doesnt attempt to preserve continuity of segments. This means you end up with pixels assigned to a segment, that are far away from other pixels in that segment.
Normalise x and y by the width and height of the image to take away the effect of image size.

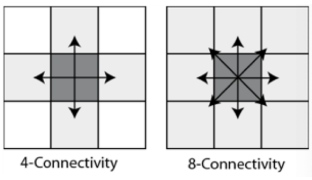
!h 

Figure 7: Pixel Connectivity

12.5 Pixel Connectivity

A pixel is said to be connected with another if they are spatially adjacent to each other. We can define this by *4-connectivity* and *8-connectivity*.

A connected component is a set of pixels where every pixel is connected directly or through any connected path from the set.

12.5.1 Connected Component Labelling

The process of finding all the connected components within a binary (segmented) image. Where each connected segment is identified as a connected component.

The two pass algorithm is described below.

1 On the first pass, iterate through each element of the data by column, then by row:

- If the element is not in the background
- Get the neighbouring elements of the current element.
- If there are no neighbours, uniquely label the current element and continue.
- Otherwise, find the neighbour with the smallest label and assign it to the current element.

2 On the second pass, iterate by column then by row:

- If the element is not the background
- Relabel the elemtn with the lowest equivalent label.

13 Shape Description and Modelling

13.1 Extracting features from shapes represented by connected components

Lets consider the borders and boundaries of a connected component.

> The borders of a connected components are the outermost components, connected to an inner 8/4 component.

We can describe the shape by region and boundary. One way to describe region is by simply calculating the area. We can calculate the perimeter/boundary length of a shape using the following formulae:

$$P(S) = \sum_j \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2} \tag{107}$$

This formulae counts the straight line (L2) distance between two adjacent boundary *points*.

13.1.1 Shape Metrics

Compactness We can measure 'tightly packed' the pixels in a component are. Its often computer as the weighted ratio of area to perimeter squared:

$$C(s) = \frac{4\pi A(s)}{(P(s))^2} \tag{108}$$

Centre of Mass We can measure the centre of mass by taking the mean of the x and y pixels in the component.

Irregularity / Dispersion We measure how 'spread' the shape is.

$$I(s) = \frac{\pi max((x_j - \bar{x})^2 + (y_j - \bar{y})^2)}{A(s)} \tag{109}$$

$$IR(s) = \frac{max\sqrt{((x_j - \bar{x})^2 + (y_j - \bar{y})^2)}}{min\sqrt{((x_j - \bar{x})^2 + (y_j - \bar{y})^2)}} \tag{110}$$

13.2 Moments

Moments describe the distribution of pixels in a shape. We can compute these for **any grey-level image**. For this purpose, lets focus on the moments of a connected component.
The standard two dimensional Cartesian moment of an image, with order p and q is defined as:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \Delta A \tag{111}$$

In the case of a connected component this is simplified to:

$$m_{pq} = \sum_i x_i^p y_i^q \tag{112}$$

Therefore the zero order of a connected component m_{00} is just the area of the component. The centre of mass is (centroid):

$$\bar{x} = \frac{m_{10}}{m00} \tag{113}$$

$$\bar{y} = \frac{m_{01}}{m_{00}} \tag{114}$$

13.2.1 Central Moments

Standard 2D moments can be used as shape descriptors, however they are not invariant to translation, rotation and scaling.
Central Moments are computed about the centroid of the shape, and are translation invariant:

$$\mu_{pq} = \sum_i (x_i - \bar{x})^p (y_i - \bar{y})^q \tag{115}$$

μ_{01} and μ_{10} are always 0, so have no descriptive power.
Normalised Central Moments are both scale and translation invariant:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{(p+q)}{2}} + 1} \tag{116}$$

Hu Moments
There exist 7 translation, scale and rotation invariant moments:

$$\begin{aligned}
M1 &= \eta_{20} + \eta_{02} \\
M2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M3 &= (\eta_{20} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
M4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \times ((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
M6 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{12} + \eta_{30})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned}$$

13.3 Chain Codes

Simple way of encoding boundaries: We walk around the boundary and encode the direction you take, on each step, as a number. We then shift the code so it froms the smallest possible integer value. This is to make it invariant to the starting point.

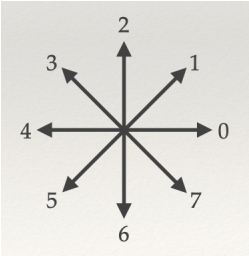


Figure 8: Directions

Given a starting point, we refer to 8 and go around the edges, and note the direction. We then sort the directions we took in ascending order.

13.3.1 Chain Code Invariance

We can make this **rotation invariant**, by encoding the differences in direction rather than absolute values. We can make this **scale invariant**, by resmapling the component to a fixed size.

13.4 Active Shape Models and Constrained Local Models

We can extend a Point Distribution Model by also learning local appearance around each point. This is used as an image template such as facial landmark...

14 Local Interest Points

An example of a good local interest point includes the following characteristics:

- Invariance to brightness change (local change and global change)
- Sufficient texture variation in the local neighbourhood
- Incariance to changes between the angle/position of the scene to the camera.

14.1 Finding Local Interest Points

There are two main methods of detecting local interest points:

- Corner Detection - Harris and Stephens
- Blob Detection - Difference of Gaussian Extrema

14.1.1 Harris and Stephens Corner Detector

We look through a small window (frame) of the image pixels, and search for corners. If we shift that window by a small amount, a large change in intensity should be present.
We measure the *weighted average change in intensity between a window and a shifted version by* $(\delta x, \delta y)$ *of that window*

$$E(x,y) = \sum_W f(x_i, y_i)[I(x_i, y_i) - I(x_i + \delta x, y_i + \delta y)]^2 \tag{117}$$

Lets use the taylor expansion, with the first order terms:

$$I(x_i + \delta x, y_i + \delta y) \approx I(x_i, y_i) + [I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \tag{118}$$

We can substitute this into the original equation, and simplify:

$$\begin{aligned}
E(x,y) &= \sum_W [I(x_i, y_i) - I(x_i + \delta x, y_i + \delta y)]^2 \\
&= \sum_W (I(x_i, y_i) - I(x_i, y_i) - [I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \delta x \\ \delta y \end{bmatrix})^2 \\
&= \sum_W ([I_x(x_i, y_i) \quad I_y(x_i, y_i)] \begin{bmatrix} \delta x \\ \delta y \end{bmatrix})^2 \\
&= [\delta x \quad \delta y] \begin{bmatrix} \sum_w (I_x(x_i, y_i))^2 & \sum_w I_x(x_i, y_i)(I_y(x_i, y_i) \\ \sum_w I_x(x_i, y_i)(I_y(x_i, y_i) & \sum_w (I_y(x_i, y_i))^2 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \\
&= \begin{bmatrix} \delta x & \delta y \end{bmatrix} \mathbf{M} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}
\end{aligned}$$

14.1.2 Structure Tensor

The square symmetrix matrix M is the called the Structure Tensor or the Second Moment matrix.

$$M = \begin{bmatrix} \sum_w (I_x(x_i, y_i))^2 & \sum_w I_x(x_i, y_i)(I_y(x_i, y_i) \\ \sum_w I_x(x_i, y_i)(I_y(x_i, y_i) & \sum_w (I_y(x_i, y_i))^2 \end{bmatrix} \tag{119}$$

14.1.3 Response Function

We can compute the eigenvalues directly, or use the corner response function in terms of the determinant and trace of M.

$$det(M) = M_{00}M_{11} - M_{01}M10 = M_{00}M_{11} - M_{10}^2 = \lambda_1\lambda_2 \tag{120}$$

$$trace(M) = M_{00} + M_{11} = \lambda_1 + \lambda_2 \tag{121}$$

$$R = det(M) - k \times trace(M)^2 \tag{122}$$

14.1.4 Application

We need to differentiate an image, we have the Sobel operator for this. The result from convolution with the x and y template can be used to calculate the structure tensor.

14.1.5 Multi-Scale

We define a Gaussian scale space with a fixed *set of scales*, and compute the corner response function at every pixel of each scale, and keep only those with a response above a certain threshold.

14.2 Scale

Scale is a problem in computer vision, since as an object gets closer it gets larger with more detail. If we are using a technique with fixed size processing windows, this is a problem.

14.2.1 Scale Space Theory

Scale space theory is a formal framework for handling the scale problem.

- The image is represented by a series of increasingly smoothed/blurred images parameterised by a scale parameter t.
- t represents the amount of smoothing
- **Key Notion: Image structures smaller than $\sqrt{t}$ have been smoothed away at scale t.**

14.2.2 Gaussian Scale Space

Formally, Gaussian scale space is defined by the following equation:

$$L(,;t) = g(,;t) * f(,) \tag{123}$$

, implies that represents a variable which the convolution is over.
$t \geq 0$ and,

$$g(x,y;t) = \frac{1}{2\pi t} e^{-\frac{(x^2+y^2)}{2t}} \tag{124}$$

$t = \sigma^2$, which means t represents the variance of the Gaussian smoothing filter.

14.2.3 Nyquist-Shannon Sampling Theorem

If we filter a signal with a low pass filter that halves the frequency content, you can also half the sampling rate without loss of information.

> *If a function $x(t)$ contains no frequencies higher than BHz, it is completely determined by giving its ordinates at a series of points spaced $\frac{1}{2B}$ seconds apart.*

This leads us to the **Gaussian Pyramid**. Where, every time we double t in the scale space, we can half the image size without a loss of information. This is faster time complexity wise and uses less memory.

14.3 Blob Detection

We recall that the *Laplacian of Gaussian* is the second derivative of a Gaussian function. If we find a local minima or maxima, we get a blob detector.

We can define normalised scale space laplacian of gaussian as:

$$\Delta^2_{norm} L(x, y; t) = t(L_{xx} + L_{yy}) \tag{125}$$

By finding extrema of this function in scale space, you can find *blobs* at their representative scale ($\sqrt{2t}$

> Very useful property: if a blob is detected at $(x_0, y_0; t_0)$ in an image, then under a scaling of that image by a factor s, the same blob would be detected at $(sx_0, sy_0; s^2 t_0)$ in the scaled image.

Its computationally expensive to build a laplacian of gaussian scale space. The following approximation can be made, the **Difference of Gaussians**:

$$\Delta^2_{norm} L(x, y; t) \approx \frac{t}{\Delta t}(L(x, y; t + \Delta t) - L(x, y; t - \Delta t)) \tag{126}$$

The laplacian of gaussian scale space can be built from subtracting adjacent scales of a Gaussian scale space, we can build a difference of gaussian pyramid.

- An oversampled pyramid as there are multiple images between a doubling of scale.
- Images between a doubling of scale ar an octave.

15 Local Features and Matching

15.1 Local Features

We extract multiple features from points of interest detected in the feature extractor, giving the result of multiple featurevectors. We extract local features for a variety of reasons, some reasons for this are:

- Image Alignment
- Camera pose estimation
- Camera calibration
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation

15.1.1 Example: Building a Panorama

Panoramas consist of multiple images. These images are matched by detecting feature points in both images, and finding the corresponding pairs prior to using these pairs in order to allign the images. We then have the following problems:

- We detect the same points independently in both image - there is no chance of matching as there is no repeatability
- For each point we need to correctly recognise the corresponding one. We need an invariant, robust and distinctive descirptor.

15.2 Matching Problems

In stereo vision, for 3D reconstruction, there are two concepts relating to matching; narrow-baseline stereo, wide-baseline stereo.

15.3 Robust Local Description

Descriptor Requirements This is dependent on task, for narrow baseline stereo:

- Robustness to rotation and lighting is not important.
- Descriptiveness can be reduced as search is over a smaller area.

For wide-baseline stereo

- Need to be robust to intensity change, invariant to rotation
- Highly descriptive to avoid mismatches, but not overly distinctive
- Robust to small localisation errors of the interest point.

15.3.1 Problems with wider baselines

- Not robust to rotation
- Sensitive to localisation of interest point
- Cant assume search areas, need to consider all the interet points in the second image, which can lead to mismatching.

15.3.2 Detection Methods

We can use either local histograms or Harris Stephens corner detector. The problems with histograms are as follows:

- Not necessarily very distinctive
- Many interest points likely to have a similar distribution of grey-values
- Not rotation invariant if the sampling window is square or rectangular.
- Not invariant to illumination changes
- Sensitive to interest point localisation

15.3.3 Overcoming Localisation Sensitivity

We ideally allow a point of interest to move a few pixels in any direction without changing the descriptor. We can apply a weighting so that pixels near the edge of the sampling patch have less effect, and those nearer the interest point have more.

15.4 Local Gradient Histograms

We can apply convolution with Sobel to achieve the partial derivatives of an image. It is then easy to compute the gradient orientation and magnitude:

$$\theta = \arctan\left(\frac{\delta f}{\delta y}, \frac{\delta f}{\delta x}\right) \tag{127}$$

$$m = \sqrt{(\frac{\delta f}{\delta y})^2 + (\frac{\delta f}{\delta x})^2} \tag{128}$$

We can then build a gradient histogram, which encodes the magnitude and direction for each pixel in the sampling patch. This is because gradient magnitudes and directions are invariant to brightniess change. It is a more *distinctive* histogram.

15.4.1 Building Gradient Histograms

- We quantise the directions into a number of bins, usually 8 separate bins.
- For each pixel in the sampling patch, accumulate the gradient magnitude of that pixel and place in the respective orientation bin.
- We can make these histograms rotation invariant by finding the dominant orientation and cyclically shifting the histogram so the dominant orientation is in the first bin.

15.5 SIFT Feature

The SIFT (Scale Invariant Feature Transform) feature is widely used. It builds on the idea of a local gradient histogram by incorporating *spatial binning*. This creates multiple gradient histograms about the interest point and appends them all together into a longer feature. We append a spatial $4x4$ grid of histograms with 8 orientations.

- This leads to a 128-dimensional feature which is highly discriminative and robust.

We can describe the SIFT process as follows:

1 We have an interest point. We take a sampling patch, where the size is proportional to the scale of the interest point.

2 Gaussian centred on the interest point, weights the pixel contributions lower towards the edges. **Corners of the sampling square have zero weight. The sampling region is circular.**

3 Spatial bins, within the sampled window. Gradient histograms are created for $4x4$ spatial bins, taking into account the Gaussian weighting. Orientations measured relative to the overall dominant orientation of the patch.

15.5.1 Matching SIFT Features

Euclidean Matching (L2 Distance) Is the simplest method to match SIFT features. We take the most similar feature in the second image to the feature in the reference image. We can improve this using thresholding, however it isnt the best matching method.

15.5.2 Improving Matching Performance

A better solution is to take each feature from the first image, and find the two closest features in the second image. We can form a match if the ratio of distances between the ref and sec image is below a threshold.

16 Consistent Matching

16.1 Eliminating Matching Mismatches

Even advanced features can be prone to being mismatched. This is because there is always a tradeoff in feature distinctiveness, if its too distinctive then it will not match subtle variations due to noise or other conditions **(think constraints)**, if its not distinctive enough, it will match everything.

16.2 Point Transforms

We are interested in **point** transforms that take the following form:

$$\mathbf{x' = Tx} \tag{129}$$

x' - Transformed Coordinate (130)

T - Transform Matrix (131)

x - Original Coordinate (132)

(133)

16.3 The Affine Transform

We also have the **Affine Transform**, which is defined as:

$$\mathbf{x' = Ax + b} \tag{134}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{22} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{135}$$

Commonly, the affine transform is written in the following form, in which another dimension is added to each vector:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{136}$$

as a result we have the following transforms:

16.3.1 Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \tag{137}$$

16.3.2 Translation and Rotation

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{138}$$

16.3.3 Scaling

$$\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{139}$$

16.3.4 Aspect Ratio

$$\begin{bmatrix} a & 0 & 0 \\ 0 & \frac{1}{a} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{140}$$

16.3.5 Shear

$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{141}$$

16.3.6 Degrees of Freedom Translations

From the above translations, we have two measures of 'Degrees of Freedom'

$$\begin{aligned} \text{(Affine) 6DoF} &= translation + rotation + scale + ar + shear \\ \text{(Similarity) 4DoF} &= translation + rotation + scale \end{aligned}$$

We can add further degrees of freedom, by normalising by w so that the transformed vector is $[., ., 1]$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{142}$$

This leads to:

$$x' = \frac{u}{w}$$
$$y' = \frac{v}{w}$$

With the following deductions above, we can then achieve the Planar Homography (Projective Transformation)

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} \tag{143}$$

16.4 Estimate Transform Matrices

We can estimate transform matrices from a set of point matches by solving a set of simultaneous equations.

- We need 4 point matches to solve a homography
- We need 3 to solve an affine transform

It is important to note that in the presence of noise, and with potentially more matches than required, we are solving an overdetermined system. We need to seek the minimum error or least-squares solution.
We use the SSE formulae as follows:

$$SSE = \sum (residual)^2 \tag{144}$$

Where residual is the residual distance from a observed point and a predicted point. (Distance to line of best fit)
With two images, we take the reference image and project the points of interest on to the second image, we then calculate the SSE, with the euclidean (L2) distance from predicted to observed.

16.5 Robust Estimation

The problem of learning a model in the presence of inliers and outliers comes under an area of mathematics called robust estimation or robust model fitting, a simple example is the RANSAC (Random Sample Consensus) algorithm:

1. Select M data items at random.
2. Estimate model T
3. Find how many of the N data items fit T within a tolerance tol, call this K. Points within the bounds of tol are inliers, outliers otherwise.
4. If K is large enough, accept T, or compute least squares
5. Repeat for N_{it} iterations, if no good fit, fail.

16.6 Problems with Direct Local Feature Matching

Local feature matching is very slow. A typical low resolution image ($800x600$) might have approximately 2000 difference of gaussian interest points/SIFT descriptors. Each SIFT descriptor is 128 dimensions. If we want to match a query image against a database of images, the distance between every query feature and every other feature needs to be calculated, **for each image!**.
We can optimise this using: Efficient Nearest Neighbour Search ((how quickly can we find a nearest neighbour to a query point in a high dimensional space:

- Index the points in some kind of tree structure
- Hash the points - create hash codes for vectors such that similar vectors have similar hash codes
- Quantise the space

We can also use a K-D binary tree structure that partitions the space along axis-alligned hyperplanes.

- Doesnt scale well
- End up needing to search most of tree
- Approximate versions do exists that are faster

17 Image Search and Bags of Visual Words

17.1 Text Information Retrieval

A bag is an unordered data structure similar to a set, however it allows repeated elements to be inserted multiple times.

In the context of text processing, the basic process is as follows.

1. Text input
2. Tokenisation
3. Stop-word removal
4. Stemming or Lemmatisation
5. Bag of Words
6. Count Vector

17.1.1 The Vector-Space Model

This model is conceptually simple:

- Model each document by a vector
- Model each query by a vector
- Assumption: documents that are close together in space are similar in meaning.
- Use similarity measures to rank each document to a query in terms of descreasing similarity.

Lets say we have two variables, 'Diet', and 'Star'. We can calculate the cosine similarity of features in this dimension with the following formulae:

$$cos(\theta) = \frac{p.q}{||p||\,||q||} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} p_i q_i^2}} \tag{145}$$

- We can precompute the sum of the features squared to save time!
- If p and q are both sparse, we spend a lot of time multiplying by zero!

17.1.2 Bag of Words Vectors

The lexicon (vocabulary) is the set of all preprocessed words across all documents known to the system. We can create vectors for each document, with as many dimensions as there are words in the lexicon. Each word in the documents bag of words contributes to a count to the corresponding element of the vector for that word.
Vectors have a very high number of dimensions, but are very sparse

17.2 Inverted Indexes

Given the bag of visual words, we can create a table for each word, and reference the document in which it appears, and the frequency.

Diet	[doc1:2]

Table 3: Map of Words to Lists of Postings

We get postings, a pair formed by a document ID and the number of times a word appears in that document. We can then for each word in a query, look up the relevant posting list and accumulate similarities using

Movie	[doc2:10]
Star	[doc1:13, doc2:4]

cosine similarity.

Given we have the the postings, and we are querying 'Movie Star'
We get the following accumulation table:

$$\text{doc2} \quad \frac{(10\times1+4\times1)}{14...}$$

17.3 Weighting Vectors

The frequency of occurrence of a term in a document represents the importance of the word in the document. We can use the following weighting schemes:

- Binary Weights - 1 or 0
- Raw Frequency - Count Vector
- TF-IDF - Gives an importance score based on metric

17.4 Vector Quantisation

Vector quantisation is a lossy data compression technique. We can use a technique like K-Means clustering to learn a fixed size set of representative vectors. The set of representation vectors, i.e. where the features lie, is called a **codebook**. The representative vectors are the mean vector in each cluster (centroids).

17.5 Visual Words

17.5.1 SIFT Visual Words

We can vector quantise SIFT descriptors. Each descriptor is replaced by a representative vector known as a visual word. The visual word describes a small image patch with a certain pattern of pixels. The codebook is the visual equivalent of a vocabulary.

17.5.2 Bags of Visual Words

Once we've quantised the local features into visual words, they can be 'bagged'. This is a **Bag of Visual Words (BoVW)**. We can build a histogram, based on the codebook and visual word occurences. This is very useful for machine learning.

17.5.3 Codebook Size

There is one key parameter in building visual word representations - **vocabulary size**.

1 Too small, and all vectors look the same.

   Not distinctive.

2 Too big, visual words may never appear

   Too distinctive.

Inverted index only gives a performance gain if the vectors are sparse. Visual words also need to be sufficiently distinctive to minimise mismatching. This implies a very big codebook, modern research systems use 1 million visual words or more for SIFT features.

17.5.4 BoVW Retrieval

Everything we have for text retrieval can be applied directly to images, e.g.

- Vector Space Model
- Cosine Similarity
- Weighting Schemes
- Inverted Index

The overall process for building a BoVW retrieval system is as follows:

- Collect the corpus of images that are to be indexed and made searchable
- Extract local features from each image
- Learn a large codebook from a sample of the features
- Vector quantise the features, and build BoVw representations for each image
- Construct an inverted index with the BoVW representations.

18 Image Classification and Auto-annotation

This lecture is largely aimed as supplementary material for the final coursework. Only examinable contents are provided.

18.1 Multilabel Classification

Where we have a cat and a dog, as labels, in the context of images this is often called **Automatic Annotation**.

18.2 Dense SIFT

Rather than extracting SIFT features at difference of Gaussian interest points, we could extract them across a dense grid. This gives more coverage of the entire image.

18.3 Pyramid Dense SIFT

For even better performance, we can use the scaled Gaussian pyramid. The sampling region is a fixed size, so at higher scales you sample more content.

18.4 Developing and Benchmarking a BoVW scene classifier

Again, this appears to be relevant only for the coursework but is included as a guide.

1. **Evaluation Dataset**

   We split the dataset into testing and training data

   Only training data is used to train the annotator

2. **Building the BoVW**

   Extract raw image features from training images

   Learn a codebook from these features

   Use K Means from a sample

   Apply vector quantisation to assign histograms from each image.

3. **Training Classifiers**

   Classifier can be trained using the histograms

   Train on a subset of training data and create validation set with remaining data. This is for the purpose of parameter optimisation.

4. **Classifying the Test Set**

   Use classifiers to find the most likely class

5. **Calculate different metrics for performance evaluation**

   Average precision

   Recall

   F1 macro/micro score

The End. 7600 Words.

Produced for COMP3204 Computer Vision

Alex Miles / 9aj