Image Processing

Biomedical Engineering Year 3 Notes

2020-2021

1 Organisation of the Human Visual System

1.1 Receptive fields in the retina

The retina has receptive fields. We tend to think of our retina or our eyes of being like a camera but if we look at the neurons that convey information from the retina to the brain, the mapping from the photoreceptors to those neurons is not just a one-to-one mapping.

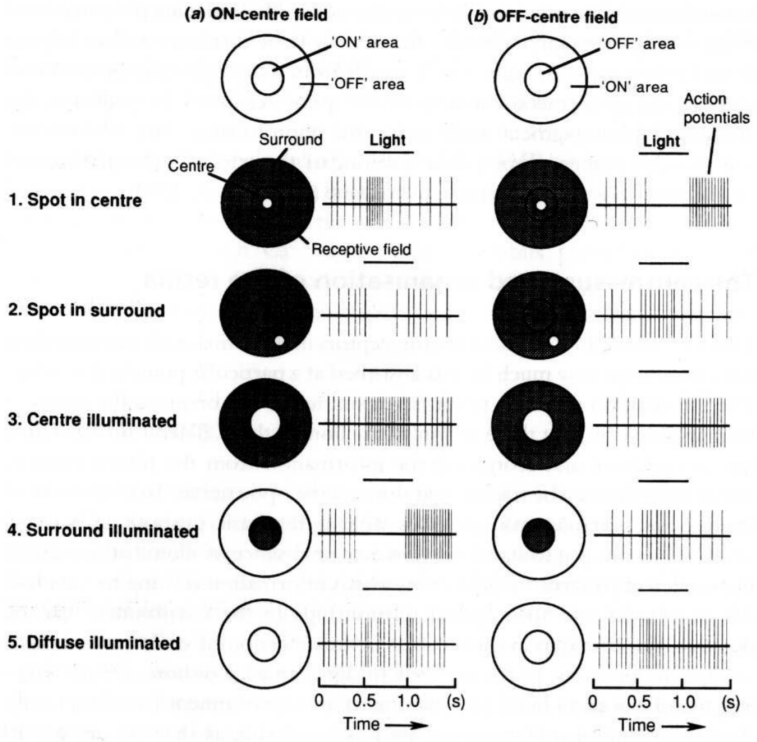

Figure 1: Illustration of ON and OFF centre surround receptive fields through effect on ganglion firing rates, (from Tovee, 1996)

There is a bit of spacial processing that goes on. The spacial processing is conveyed in Figure (1). There is what's called "centre surround" organisation, which means that the neurons do respond to photoreceptors that are very close to where they are in terms of distance in retina space, but they also respond to things that are a little bit away from them in retinal space. Through this centre surround organisation, they apply something that is an isotropic weighting across retinal space. The strongest effect comes from photoreceptors that are in the centre of this central circular organisation, and the weaker response comes from things that are in the annular surrounding region.

These two regions, the circular bit and the external annular bit, are antagonistic. This means that they work in opposite ways. This is somewhat analagous to concepts we find in gradient estimation except that the gradients are not oriented (they don't care what direction you move in from the centre) and so it is something like a Laplacian of Gaussian function. This is a neighbourhood type operation. Sometimes it is also referred to a difference of Gaussian functions.

Because you have a repetition of these patterns across the retina, effectively what you get in the retina can be modelled by spacial convolution.

1.2 The Visual Cortex

In the visual cortex, you again find these antagonistic regions of activity, but the difference is that now you have orientation preference or orientation selectivity in these neurons. And so while the retina seems to be doing something that is spatially isotropic (so it doesn't depend on what direction you move in from the centre of the receptive field), the receptive fields in the primary visual cortex and beyond have strong spacial orientation. And so they look a little bit like gradient operators in the context of image processing. Therefore, when we calculate things like gradient fields, we are doing very crude approximations to the calculations that are done in the primary visual cortex.

2 Colours

2.1 Why use different colour spaces?

Different colour spaces may improve class separability for distinguishing objects based on perceptual colour differences. This principle can be **demonstrated by bimodal or multi-modal histograms which either show a clean separation or don't**. For example, the dominant colour is better captured by trichromatic values or HSI when one compares these spaces with RGB.

2.2 RGB

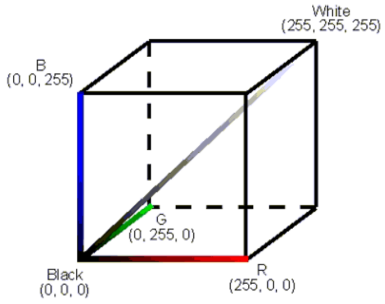

Figure 2: RGB Colour space

The RGB colour space diagram should feature a cube, with coordinates running from (0,1) on orthogonal axes, Cartesian space. Grey-scale values lie along the line R = G = B from (0,0,0) to (1,1,1). This means that for a greyscale picture for each pixel, $R = G = B$.

2.3 Luminance

The HSV, YUV or Lab colour space all potentially capture the luminance concept of intensity. One can then boost the luminance channel, perhaps by histogram modification, shifting intensities towards 1. Another way to brighten the image is map RGB to greyscale and shift everything towards 1.

2.3.1 Brightness vs Luminance

Brightness is techically the number of photons hitting something, while luminance has a very specific definition which is based on the colour space and also a fixed calculation to map an RGB representation to a Hue Saturation Lightness colour space.

2.4 Normalised RGB

Normalised RGB uses the trichromatic coefficients $x = \frac{R}{R+G+B}$, $y = \frac{G}{R+G+B}$, $z = \frac{B}{R+G+B}$. This is an example of a very simple non linear mapping between RGB and an alternative colour space that makes the process segmenting or labelling pixels based on their colour much easier to perform.

This is a general idea that we can take forward to other things. The normalised RGB values, if taken as a triplet of values, they form a three dimensional vector. Detecting, for example, the dots on a lily, becomes easier in that new feature space, than it does in the original representation in RGB.

3 Image representation

4 Discrete image models

4.1 Averaging over experimental space

One can think of each pixel of a 2D image as lying not in a two-dimensional, but a three dimensional space, where the third dimension represents experimental space. The third dimension is theoretically infinite, but in any series of practical measurements, is finite.

The mean image is obtained by

$$\mu(m,n) = \frac{1}{P}\sum_{p=1}^{P} f(m,n,p) \text{ in the limit as } P \longrightarrow \infty \tag{1}$$

There are various practical applications of estimating an average image over experimental space. One is the determination of the average appearance of what an average shape might look like over several observations of the same (in principle) shape.

Before attempting to create such "average" models of object appearance, one typically needs to spatially co-register images containing the object, in an object-centric fashion. Then, this averaged object appearance can be used as a template to find candidate search locations in image space.

4.2 Covariance

The covariance function for a random field is defined by

$$C(m,n;m',n') = E(\tilde{f}(m,n) - \mu(m,n))(\tilde{f}(m',n') - \mu(m',n')) \tag{2}$$

This is a four dimensional function that can be thought of as capturing the linear relatedness between pixel pairs in the image. Because each pixel will have two coordinates, we end up with a four dimensional function. We can, however, also map each pixel in two-dimensional space to a single element of a vector that contains all pixels. In this case, after "unrolling" an image, the covariance function becomes two-dimensional. It is then described as a covariance matrix, and will be a square matrix of size $MN \times MN$, where M is the number of rows and N is the number of columns of the corresponding image.

Larger magnitude values in C identify pairs of pixels that, over experimental space, are strongly linearly correlated. Small magnitude values will be found when there is little correlation between the corresponding pixel pairs. Positive values imply positive correlation, negative values imply negative correlation. Thus, the covariance matrix captures the linear predictability of intensities relative to the mean image.

4.2.1 Calculating an entry in the covariance matrix

Given multiple matrices that correspond to multiple observations, you may be asked to calculate the covariance matrix entry corresponding to the covariance between two pixel locations. The covariance value is the average of the product of the deviations of the intensities at these two pixel locations from their respective averages. Eg: average of (pixel one in observation one's derivation from the pixel one average) x (pixel two in observation one's derivation from the pixel two average) + (pixel one in observation two's derivation from the pixel one average) x (pixel two in observation two's derivation from the pixel two average), etc.

4.3 Noise

4.3.1 Using an imaging phantom to test for the spacial variation of noise

An imaging phantom is a specially designed object that can be scanned to analyse and tune the performance of an imaging device.

Every time you scan the same object, there will by some random noise that causes fluctuations around the mean.

Some specially designed phantoms will be designed in such a way that you can say what the ideal image should look like, allowing you to accurately measure what the deviation introduced by noise is.

Another way to compare to a "noise-free" image is to simply compare to the mean image averaged over experimental space (repeated scans of the same object). However, this approach relies on the assumption that the intensity of the random noise is symmetrically distributed (meaning that is just as likely to be too intense as not intense enough). The approach also relies on an assumption that the distribution of the intensity of the noise is independent for each pixel.

If the above assumptions hold, then you can

- Averaging over a large number of scans to get an estimate of the true (random-noise free) image
- Calculate noise variance as a function of spacial location
- See if noise variance differs between locations

Once you have a mean image, you can calculate a deviation image for each sample / observation, by subtracting each pixel value of the mean image from the respective pixel values of the sampled images.

Next, compute the square of the deviation at each pixel for each sample and then average all the samples over experimental space. This gives you an estimate of the noise variance (remember variance is s.d. squared) as a function of spatial position. To assess whether or not there is any consistent trend in the spacial variation, you can try fitting a bilinear or bicubic model accross the space.

An alternative approach would be to compute the full covariance matrix for all pixels. The covariance matrix expresses the linear predictability of the deviation of one pixel given the deviation of another. We can also try to see if the covariance values vary significantly from one pixel pair to another.

Finally, one can get the signal-to-noise ratio by dividing the average variation by the mean image.

5 Thresholding

6 Binary image processing

A binary image is an image that contains only one of two intensity levels.

6.1 Neighbourhood Relationships

6.2 Connectivity

6.3 Distance measures

Chessboard and city-block distances are less commonly used, but they are sometimes found where there is a requirement to calculate distance in something other than 2D space, for example, where we calculate distance in a D dimensional feature space, and the calculation of Euclidean distance might turn out to be too expensive for the available resources (either time, power or computation capacity).

6.3.1 Euclidean distance

$$D(P,Q) = \sqrt{(m_P - m_Q)^2 + (n_P - n_Q)^2} \tag{3}$$

6.3.2 City block distance, D_4

$$D(P,Q) = |m_P - m_Q| + |n_P - n_Q| \tag{4}$$

6.3.3 Chessboard distance, D_8

$$D(P,Q) = \max\{|m_P - m_Q| + |n_P - n_Q|\} \tag{5}$$

6.4 Connected component labelling

The connected-components algorithm is often applied in two dimensions, and for this case, there are two main forms of connectivity that are used: 4-connectivity and 8-connectivity. Both begin with a unique labelling of each non-zero pixel in the image.

For example if we start with $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$ then the first stage results in $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 4 \\ 5 & 0 & 0 & 0 & 6 \end{pmatrix}$.

Next we do a forward (top left to bottom right) and backwards pass (bottom right to top left), where we examine each pixel and replace it with the minimum label out of all the labels of the pixels it is connected to. This could be in the 4-connected or 8-connected sense. The forwards and backwards passes are repeated until no change of labelling occurs.

Detecting such a change is very easy to do: simply performing a pixel-by-pixel subtraction of the output array at the latest iteration from output of the previous one, then detecting to see whether any bits have changed, will suffice.

Using both forward and backwards passes makes convergence faster.

6.5 Binary image moments

6.5.1 Definition of Binary Moments

The moment of order $(p+q)$ is defined as

$$M_{pq} = \sum_{(m,n)\in R} m^p n^q f(m,n) \tag{6}$$

R is not necessarily the whole image - it could just be a region of the image.

- There is one zeroth order moment: $M_{00} = \sum_{(m,n)\in R} f(m,n)$. This represents the "mass" of the binary region.
- There are two first order moments:
  - $M_{10} = \sum_{(m,n)\in R} mf(m,n)$
  - $M_{01} = \sum_{(m,n)\in R} nf(m,n)$
- There are three second order moments:
  - $M_{20} = \sum_{(m,n)\in R} m^2 f(m,n)$
  - $M_{11} = \sum_{(m,n)\in R} mnf(m,n)$
  - $M_{02} = \sum_{(m,n)\in R} n^2(m,n)$

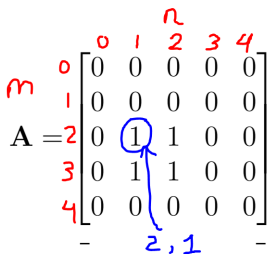

Figure 3: How to get m and n for each element in a matrix in order to calculate the image moment

The centroid of a region is $(\bar{m}, \bar{n})$ where $\bar{m} = \frac{M_{10}}{M_{00}}$ and $\bar{n} = \frac{M_{01}}{M_{00}}$

6.5.2 Central Moments

The $(p+q)^{\text{th}}$ central moment of a region R is defined as:

$$\mu_{pq} = \sum_{(m,n)\in R} (m-\bar{m})^p (n-\bar{n})^q f(m,n) \tag{7}$$

The calculation of central moments is similar to ordinary moments except the m and n values are replaced by their centroid-relative coordinates.

Central moments are invariant to translations, because all calculations that define these central moments are performed relative to the centroids. Translations of shapes translate the centroids by the same translations, so results of calculations remain the same.

6.5.3 Second order central moment matrix

The moment matrix is a diagonal matrix containing the moment values arranged like this:

$$M = \begin{pmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{pmatrix} \tag{8}$$

The relationship between the eigenvalues of this matrix tells us about the degree of elongation of any one (major axis) direction over its perpendicular direction, with the eigenvectors pointing to each of these.

6.5.4 Application of binary image moments

Binary Image Moments provide a very elegant way of describing a binary image region with a small number of values that are intuitive and very descriptive.

The average cell area in a greyscale image may be determined from the 1st and zero order region moments from each region after application of a connected components labelling operation. By forming an estimate of the histogram of region sizes, one may be able to identify components that correspond to individual nuclei, clumps of two, clumps of three etc. This may be used to set the dimensions of the morph kernel. To perform this, however, need to use CCON algorithm first to separate binary pixels into groups, then do a histogram over groups, looking for evidence of clumping in the histogram results.

6.5.5 Deriving a relationship between the second order region moments and the orientation of one binary-encoded image of a cell nucleus

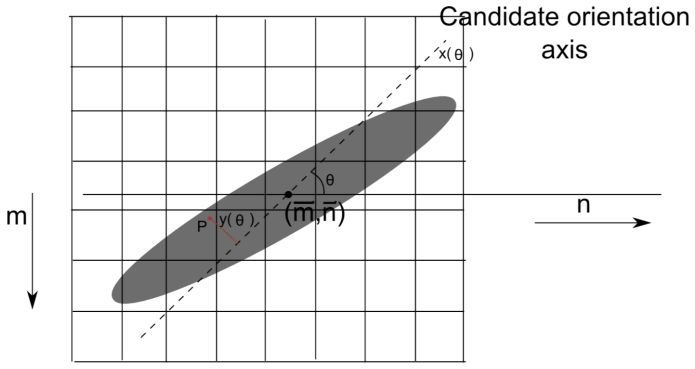

Figure 4: A region of a binary image with an underlying object in continuous space

- We consider an axis $x(\theta)$ that passes through $(\bar{m}, \bar{n})$ (the centroid, and in this case centre of mass) of the ellipse, and is rotated by θ degrees, as shown in Figure (4).
- A definition of the object's orientation can be obtained from the $x(\theta)$ line which minimises the moment of inertia.
- From Mechanics, we know that the moment of inertia, $I(\theta)$, of the region/body around the candidate axis $x(\theta)$ can be obtained by summing $y^2(m, n; \theta) f(m, n)$. In our case $f(m, n)$ is 1 throughout the object, therefore the equation is:

$$I(\theta) = \sum_{(m,n)} y^2(m, n; \theta) \tag{9}$$

- However, the term $y(m, n; \theta)$ (the perpendicular distance from the candidate axis) is a function represented in a rotated coordinate system. Therefore, in Cartesian coordinates, the equation becomes:

$$I(\theta) = \sum_{(m,n)} \{(m - \bar{m}) \cos(\theta) - (n - \bar{n}) \sin(\theta)\}^2 \tag{10}$$

- In order to minimise $I(\theta)$, we take the partial derivative w.r.t θ and set equal to 0:

$$\sum_{(m,n)} \frac{\partial}{\partial \theta} \{(m - \bar{m}) \cos(\theta) - (n - \bar{n}) \sin(\theta)\}^2 = 0 \tag{11}$$

- To solve this, you apply the chain rule, split up the sums, and then apply the trig identities $\sin(2\theta) = 2 \sin(\theta) \cos(\theta)$, and $\cos^2(\theta) - \sin^2(\theta) = \cos(2\theta)$
- After rearranging, the final answer should be $\theta = \frac{1}{2} \tan^{-1}(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}})$

6.6 Medial axis extraction

One useful simplification of a binary region is through the extraction of its medial axis. Although it captures many salient aspects of object shape, it often needs to be used with other techniques to really make it applicable to a wide range of problems. The algorithms that are most widely used are referred to as "skeletonisation" and "thinning". To perform skeletonisation, one can take a number of approaches, but one of the more interesting ones is to start of by using a distance transform. After the distance transform, local maxima within the distance transform can be linked together to form the medial axis of a binary image shape.

6.6.1 The distance transform

The distance transform is produced from a binary image and contains values such that each pixel encodes its own distance to the nearest "0" binary pixel.

One example of a distance transform algorithm; it computes a distance transform by using the chessboard distance measure:

```
f0(m,n)=f(m,n)        % f(m,n) is Binary image
for k=1..K do
   fk(m,n)=f0(m,n)+min { N4(fk−1(m,n)) }
end for
```

Figure 5: Example of a distance transform algorithm in pseudocode

6.6.2 From distance transform to skeleton

Once we have a distance transform represented in the form of a 2D array, the skeleton of the original binary image region can then be defined as being the set of points such that the distance transform values represent a local maximum among (say) the 4-connected neighbours i.e. the set of points $\mathcal{S}$ defined by:

$$\mathcal{S} = (m, n) : (f_K(m, n) \neq 0) \text{ AND } (f_K(m, n) \geq \text{N4}(f_K(m, n))) \tag{12}$$

7 Transforms

7.1 General Linear Transforms

7.1.1 Practical applications

- Image compression (eg Discrete Cosine Transform)
- Feature detection
- Denoising (eg Fourier transform)
- Edge detection (eg Wavelet transform)

7.1.2 Properties

Not all transforms have these properties, but common properties looked for in GLTs are:

- Compactness
- Speed
- Exact inverse
- Large values in response to features
- Easy transform definitions

For feature detection, use a transform that contains basis images that in some way match the features one seeks to detect. For compression, one needs an exact inverse, and easy definitions for the forward and inverse functions is preferred, as is the main requirement of energy compaction. For denoising, one often needs the compaction property as well, whilst other properties such as speed may or may not be so important.

7.1.3 General expression for moving from image space into linear transform space

$$v(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) b(k,l,m,n) \tag{13}$$

Usually, but not always, $K = M$ and $L = N$ (the number of transform coefficients is equal to the number of image pixels).

The values b may be said to for a so-called "basis set", however this is not always true. It is incorrect to refer to any collection of 2D basis images as a basis set. To constitute a basis set, these basis images must be linearly independent of one another.

Equation (13) can be re-expressed using the inner product in order to reflect how linear image transforms are the projection of an image into a collection of basis images.

The inner project between 2 matrices is defined as

$$\langle \mathbf{F}, \mathbf{G} \rangle = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) g^*(m.n) \tag{14}$$

In matlab, this like like using .* between 2 matrices and then summing all the entries of the resulting matrix.

The inner product will give a single scalar. With this opertation, the general linear transform can be expressed as

$$v(k,l) = \langle \mathbf{F}, \mathbf{B}^*_{k,l} \rangle \text{ , k = 0 , 1 ... K - 1, l = 0 ... L - 1} \tag{15}$$

7.2 Separable image transforms

Any linear image transform that can be expressed as V = AFA^T is separable. Transform separation reduces computational cost from $O(N^4)$ to $O(N^3)$, by reducing number of floating point operations to move from image space into transform space.

On a conceptual level, separable transform first transform image rows, and then transform the columns of the result. In a similar form to above, we can also write that a separable transform may be implemented by two pairs of matrix multiplications, roughly described by $\mathbf{V} = \mathbf{A}_C\mathbf{F}\mathbf{A}_R$ where $\mathbf{F}$ represents the image as a 2D array, and $\mathbf{V}$ represents transform space. $\mathbf{A}_R$ and $\mathbf{A}_C$ are row and column operators, respectively.

Tip: If $\mathbf{D}$ is an orthogonal matrix, $\mathbf{D}^{-1} = \mathbf{D}^T$

7.3 Hadamard transform

Hadamard is like Fourier but with square waves.

There are different valid definitions of the Hadamard transform.

The 2D Hadamard transform may be defined by the mapping from image space to transform space which is given by:

$$\mathbf{V} = \mathbf{H}_m\mathbf{F}\mathbf{H}_m \tag{16}$$

where m denotes an appropriate order of Hadamard matrix, and where the m^{th} order matrix may be defined by

$$\mathbf{H}_m = \mathbf{H}_{m-1} \otimes \mathbf{H}_{m-1} \tag{17}$$

with $\mathbf{H}_1$ being defined as

$$\mathbf{H}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{18}$$

The Hadamard matrix can also be defined using the recursion

$$\mathbf{H}_n = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{pmatrix} \tag{19}$$

Property of Hadamard matrix: $\mathbf{H} = \mathbf{H}^* = \mathbf{H}^{\mathbf{T}} = \mathbf{H}^{-1}$

Example calculation: Calculating basis images corresponding to the 2D Hadamard transform for $m = 2$

- We have $\mathbf{V} = \mathbf{H}_m\mathbf{F}\mathbf{H}_m$
- It is easy to show from this that $\mathbf{F} = \mathbf{H}_m^{-1}\mathbf{V}\mathbf{H}_m^{-1}$
- By the properties of the Hadamard matrix, this is equal to $\mathbf{H}_m\mathbf{V}\mathbf{H}_m$
- To get a basis image, we set one element of the transform space $\mathbf{V}$ to 1 and then perform the inversion
- For example we can set $v_{1,1}$ to 1 in a 4×4 transform space
- $\mathbf{B}_{1,1} = \mathbf{H}_m \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{H}_m$

Example calculation: Estimating the difference in computational load between a separable and non-separable implementation of a 2D Hadamard transform for $m = 3$.

- The separable version of the transform consists of two pairs of matrix multiplications ($\mathbf{V} = \mathbf{H}_m\mathbf{F}\mathbf{H}_m$) – that is, we have two matrix products. One of the operators in this is the image itself. If we have $m = 3$, it means that the image is also 16×16 - this can be found from (2^{3+1}).
  - A 16 by 16 matrix has 256 elements
  - To get the first intermediate result (1 out of 2 matrix multiplications), we need to obtain each element out of 256 by taking a dot product between the appropriate row of the image matrix and the appropriate column of the Hadamard transforming matrix.
  - Each dot product between a row and a column involves 16 multiplications and 15 additions. This can be described as 16 "multiply and accumulate" (MAC) operations.
  - Therefore there is a total of 256×16 or 16^3 MAC operations for the one matrix multiplication
  - For the whole transform, this needs to be doubled, seeing as there are 2 matrix multiplications
  - Therefore the number of operations for the separable transform version is 2×16^3
- The long way of doing the transform is taking a matrix inner product between the image being transformed and each of the basis images.
  - Since the size of each image is 16×16, the number of MAC operations for each value in the transform space is 16^2.
  - There are 16×16 basis images, therefore we need to 16^4 MACs to do to the transform.
- Therefore in this particular instance, the separable transform requires 8 times fewer operations, however seeing as the number of operations in the first case scales with n^3 and in the second case n^4, for larger images the difference will be more significant.
- We have ignored any zeros in the image matrix, any spatial symmetries or the fact that we are actually just adding and subtracting values with the Hadamard transform, followed by scaling results.

7.4 2D DFT

7.4.1 Main properties of 2D DFT

- **Orthogonal bases**
- **Energy compaction**: Good for natural scenes (not as much for random noise)
- Magnitude spectrum is **shift invariant**
- **Numerically exact inverse (PR condition)**: useful for filtering

- Overcompleteness is not good for compression, but the real version of the DFT can be used (as the Discrete Cosine Transform)
- For medical imaging, the 2D-DFT can be used for image recontruction from (say) k-space to image space.

7.5 An intuitive way of thinking of the concept of finding the basis images

Finding the basis images is finding the answers to the question: "What are the images for which if we apply the transform, we get a a matrix with just a single 1 in it and the rest are zeros?". Aka a basis image is all zeros and just one 1 when "looked at" in the transform space.

To find the these basis images, we can apply the inverse transform to a matrix of the appropriate size, with all zeros and a single 1. Repeating this and placing a one in all the possible locations will give you all the basis images.

7.6 Looking at basis images

If a basis image has, for example a horizontal edge, then the corresponding value in the transform space will have greater value if the original image contains a horizontal edge. If there are no horizontal edges in the image, then the transform space value corresponding to that basis image will have a value close to 0.

7.7 Finding a suitable transform space for a particular aim

The general principle is that the presence of the desired structure within the image leads to strong, distinct peaks in transform space, whereas the absence leads to lack of either large values or well-defined peaks. One can then use thresholding or peak detection in transform space to say whether or not the target shape is present. Transforms used in this way should have the property of being fast to compute, but do not necessarily have to have an inverse.

To detect and localise a certain shape, this requires that the transform space should encode spatial location in a way that can be easily used for inference about the target shape's location. It is also good for the transform shape to be invariant to certain parameters of the target shape.

8 Neighbourhood operators

8.1 Linear neighbourhood operators

The main distinguishing characteristic of linear neighbourhood operators is that the operation relating the output image and input image to the operator can be expressed by spatial convolution. All finite-length masks are implementable by spatial convolution.

8.1.1 Applications

- Constructing feature spaces
- Image enhancement (smoothing, sharpening or noise reduction)
- Creating a scale-space
- Gradient estimation (as a prelude to edge detection)
- Direction estimation

8.2 Convolution masks

8.2.1 Gradient masks

Gradient masks have elements that sum to zero. This is because when applied to an area of constant image intensity, the inner product between the mask and image should be 0. This makes sense as constant intensity is as far away from a gradient as one can get!

To get the gradient field of a 2D image, at least two masks should be applied. If two masks are applied (usual scenario), the outputs have to be combined (convolution outputs squared and summed) to estimate gradient magnitude, and the ratio of outputs between the masks encodes the direction of the gradient field by representing the tan() of the angle.

When an image is discretised and the gradient field extracted using such a method, there is additional discretisation and imaging (sensor) noise. Also, a 3×3 mask uses only immediate neighbours, so the result depends on the density of spatial sampling, and may give different magnitudes for different sampling densities of a continuous spatial field. There are also boundary effects that one has when a finite-window of image data is used for the convolution.

Example calculation: Examining an edge-detection convolution mask

A general-form convolution mask for edge detection is $\begin{pmatrix} a & b & a \\ 0 & 0 & 0 \\ -a & -b & -a \end{pmatrix}$.

Using a linear facet model of the form $I(r, c) = \alpha r + \beta c + \gamma$ (where r and c are coordinates defined with respect to the centre of the model neighbourhood) we can:

- Find a relation between a and b such that the output of the mask provides an estimate of gradient along the columns (direction of increasing row number), i.e. $\partial I(r, c)/\partial r$
  - For this we need $2a + b = 1/2$ because we need to obtain 1 when applying the mask to a matrix like $\begin{pmatrix} 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{pmatrix}$ (the gradient in the direction of increasing row number is 1 in this example).
- Use the mask with $a = b$, to determine the rate of intensity change at an angle of 30° clockwise from the right horizontal at some selected point in the image.
  - The unit vector pointing in the indicated direction is $\frac{\sqrt{3}}{2}\hat{r} + \frac{1}{2}\hat{c}$
  - Therefore $g_{30^\circ} = \frac{\sqrt{3}}{2} g_r + \frac{1}{2} g_c$ where g_r is the (scalar field) result of using the convolution mask to estimate the gradient in the row direction and g_c is the (scalar field) result of using the convolution mask to estimate the gradient in the column direction

8.3 Non linear neighbourhood operators

8.3.1 Mathematical morphology and the set-theoretic definitions of binary erosion and dilation

The mathematics used to define binary morphology is set based, and so is very much different from linear transform techniques, which are based on sums and products of pixel values.

The two primary morphological operations are dilation and erosion. For all of these definitions, we need to consider set elements in an 2-dimensional space. More specifically, we represent the black pixels of a 2D binary image matrix by a set of vector tuples defining their locations. This means, for example, that to represent an image of 3×3 pixels which are all white but for a diagonal row of black pixels, we use the set $\{(1, 1), (2, 2), (3, 3)\}$.

Dilation is the combination of two sets. It can be represented by a process known as Minkowski addition, or as the vector addition of set elements. By this definition

$$A \oplus B = \{p \in Z^2 : p = a + b, a \in A, b \in B\} \tag{20}$$

Another (easier to remember) definition is: The dilation of B by A is the union of all translates of B in A.

Erosion is the morphological dual of dilation. It can be though of as a "searching" procedure for a structuring element B in binary image A. In a manner that is easily remembered, but pretty incomprehensible, we say that whilst dilation is the union of translates of B in A, erosion is the intersection of the negative translates of A by B. Erosion tends to decrease the size of A in a manner that is dependent on the shape of the structuring element B. In terms of set-theoretic operations, one may define erosion by:

$$A \ominus B = \{p \in Z^2 : p + b \in A, \forall b \in B\} \tag{21}$$

8.3.2 Binary erosion - how to perform it (intuitive explanation)

With binary erosion, how have an image (in the example below, called A) and a (usually smaller) structuring element (in the example below, called B). The structuring element can have any arbitrary "origin", but usually the origin is the center of the structuring element (eg: element 2,3 in a 3 by 5 matrix).

For each pixel in A, we superimpose the structuring element B in such a way that the origin of B aligns with A. If, when superimposed, B is completely contained within A (wherever there is a 1 in B, there is a 1 in A below), then the pixel under consideration is retained. Otherwise it is deleted (set to 0).

9 Image Segmentation, Edge Detection and Object Recognition

9.1 Image Segmentation

Image segmentation is the labelling of each pixel or voxel of an image or volumetric data set in a semantically meaningful manner. For example, one wants to be able to determine which voxels correspond to the surfaces of structures within a voxel scan. The inferred surfaces may be used for a number of purposes related to display, but principally, surface rendering techniques can be used to quickly provide a pseudo-three-dimensional rendering of structures onto a 2D display. Labelling individual objects can also be used for interactive visualisation: for example, a user may address anatomical structures individually, allowing structures (such as ribs, large bones and so forth) to be pulled out of position for better rendering or inspection.

9.2 Edge detection

9.3 Gradient based edge detection

9.4 Zero-crossing based edge detection

9.5 K Means

The k-means algorithm is an example of a clustering algorithm, which attempts to blindly find k clusters of data vectors in a feature space.

1. Initialise with k clusters. Commonly used initialization methods are Forgy and Random Partition.
2. **Assignment step:** Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance.
3. **Update step:** Recalculate means (centroids) for observations assigned to each cluster.
4. Iterate steps 2 and 3 until the assignments no longer change

If you don't know how many clusters to use, you can start with the upper bound on the maximum number of clusters, and then merge clusters as their centroids get close to each other with respect to the scatter within either clusters. The difficulty with this approach is knowing what distance criteria between cluster centres should be used to determine the point of merging clusters (and therefore reducing the number of clusters).

9.6 Fitting a sloped faced model

Background illumination across an image may be described by a sloped facet model.

We can determine the best-fit optimum parameters for a linear model of the form $f(m,n) = a_0 + a_1 m + a_2 n$ with a suitable coordinate system.

- The best coordinate system to choose for this derivation is to select m, n to be 0 in the centre of the neighbourhood of the fit.
- To find the best-fit parameters for the linear model, we will minimise the squared error between the true intensity distribution $f(m,n)$ and the model, which is parameterised by a_0, a_1 and a_2.
- Therefore, we will minimise:

$$e(m,n;a_0,a_1,a_2) = \sum_{(m,n)} (f(m,n) - a_0 - a_1 m - a_2 n)^2 \tag{22}$$

  with respect to a_0, a_1 and a_2 respectively

- To do so, we take the partial derivative with respect to each parameter and set it equal to zero, and then solve for the parameter in question.
  - Taking the partial derivatives (this is simply applying the chain rule), we get

$$\frac{\partial e}{\partial a_0} = \sum_{(m,n)} 2(f(m,n) - a_0 - a_1 m - a_2 n)(-1) \tag{23}$$

$$\frac{\partial e}{\partial a_1} = \sum_{(m,n)} 2(f(m,n) - a_0 - a_1 m - a_2 n)(-m) \tag{24}$$

$$\frac{\partial e}{\partial a_2} = \sum_{(m,n)} 2(f(m,n) - a_0 - a_1 m - a_2 n)(-n) \tag{25}$$

  - The next stage is to set each partial derivative equal to 0 and also split up / simplify the summation. Doing this we get

$$\sum_{(m,n)} f(m,n) - \sum_{(m,n)} a_0 - \sum_{(m,n)} a_1 m - \sum_{(m,n)} a_2 n = 0 \tag{26}$$

$$\sum_{(m,n)} f(m,n)m - \sum_{(m,n)} a_0 m - \sum_{(m,n)} a_1 m^2 - \sum_{(m,n)} a_2 nm = 0 \tag{27}$$

$$\sum_{(m,n)} f(m,n)n - \sum_{(m,n)} a_0 n - \sum_{(m,n)} a_1 mn - \sum_{(m,n)} a_2 n^2 = 0 \tag{28}$$

– At this stage its useful to apply the choice of coordinate system. Seeing as we chose m, n to be 0 in the centre of the neighbourhood of the fit, this conveniently means that:

$$\sum_{(m,n)} mn = \sum_{(m,n)} m = \sum_{(m,n)} n = 0 \tag{29}$$

– This allows us to simplify equations (26), (27) and (28) to:

$$\sum_{(m,n)} f(m,n) - \sum_{(m,n)} a_0 = 0 \tag{30}$$

$$\sum_{(m,n)} f(m,n)m - \sum_{(m,n)} a_1 m^2 = 0 \tag{31}$$

$$\sum_{(m,n)} f(m,n)n - \sum_{(m,n)} a_2 n^2 = 0 \tag{32}$$

- Some trivial rearranging gives us the final values of the optimum parameters, namely

$$a_0 = \frac{\sum_{(m,n)} f(m,n)}{\sum_{(m,n)} 1} \tag{33}$$

$$a_1 = \frac{\sum_{(m,n)} mf(m,n)}{\sum_{(m,n)} m^2} \tag{34}$$

$$a_2 = \frac{\sum_{(m,n)} nf(m,n)}{\sum_{(m,n)} n^2} \tag{35}$$

9.7 SIFT

The multiscale gradient field of an image is used to construct a SIFT keypoint descriptor. Given that the gradient field has been computed for a digital intensity image at multiple spatial scales, in order to produce a SIFT descriptor at some location (m, n) within an image, we need t create histograms of the gradient field at some scale σ. These histograms would typically consist of 8 orientation bins, and histograms would be calculated over $M \times M$ sized spatial patches (measured at the resolution of the original image) that are centred on the location at which the keypoint is being computed.

Generally, one or more intrinsic scales are also assigned to the location of the keypoint. The size of neighbourhood (measured in the original image) is then a function of the scale parameter at that point. Since the SIFT descriptor also spatially subsamples the image in a scale-dependent way, the patch size used for the gradient field histogramming turns out to be constant when measured at the resolution of the gradient field representation for a particular scale σ. For example, in the most widely used implementation of SIFT, this would be across a region of size 16×16. Over this size of patch, non-overlapping 4×4 groups of pixel gradients will be used to construct an 8-bin histogram. The histogram counts are weighted by gradient magnitude for respective orientations, and smoothed (or interpolated into neighbouring bins). In order to achieve this, the scales assigned to each keypoint need to be known. In addition, each descriptor is rotated so that the dominant direction in a patch always occupies the same bin location – this provides approximate rotation invariance to each descriptor. If there are two nearly equal dominant directions, two possible dominant orientations are assigned. Thus, dominant direction(s) also need to be known.

10 Image Registration

Image registration aims to geometrically align one image with another and is a prerequisite for all imaging applications that compare images across subjects, across imaging modalities, or across time.

Symbol	Meaning
A	An image that will serve as the reference. Also, $A(\mathbf{x}_a)$
B	An image that will be registered. Also, $B(\mathbf{x}_b)$
$\mathbf{x}$	A point in space
$\{\mathbf{x}_i\}$	A set of points in space.
$\mathbf{x}_A$	A spatial location in image A.
$\mathbf{T}$	Transformation mapping a point in B to a point in A
$\mathcal{T}$	Maps both position and intensity of image B to image A
$\tilde{\Omega}_A$	Field of view of the imaged volume; continuous domain.
Ω_A	The discrete domain of image A
Ω_A	The discrete domain of image B
$\Omega^T_{A,B}$	The discrete overlap domain of A and B for a given transformation $\mathbf{T}$
$N_{A,B}$	The number of voxels in overlap domain of A and B

Figure 6: Common notation for image registration problems

10.1 Rigid body registration

Rigid body registration is image registration given that the different images have been transformed via translation or rotation. There may also by a scale change where angles are preserved. These transformations are usually represented as 4×4 matrices as opposed to 3×3 as "homogeneous coordinates" are used. This allows the elegant combination of translation into a single matrix that expresses affine transformation.

Finding the required coordinate transformations does not necessarily take into account the differences in contrast mechanism that may be required to co-align scans taken on different scanners, subjects or across modalities.

The notion of interpolation is important to completely perform a mapping between one image, or volumetric data set, and another. To perform the mapping itself, we must figure out how to map the transformed point locations to intensity values – this generally requires interpolation to be performed.

10.2 Procrustes-based image registration

If one has a series of landmarks (sometimes referred to as points) defined in image A and image B, which are known to have some correspondence, then it is possible to use these so called homologous points to infer the geometric transformation $\mathbf{T}$ between A and B.

The classical "Procrustes" problem for T being a rigid body transformation has known solutions through the use of the singular value decomposition.

- We have 2 sets of points $\{x^i_A\}_{i=1\ldots N}$ and $\{x^i_B\}_{i=1\ldots N}$ that are known to correspond in 2/3D space
- First we subtract from each point the centroid of it's point set
- We form the matrices $\mathbf{P}_A$ and $\mathbf{P}_B$ by aligning all the (centroid-shifted) x_A's and x_B's as the rows of the respective matrices. Both $\mathbf{P}_A$ and $\mathbf{P}_B$ are therefore $N \times D$ matrices, where D is usually 2 or 3.
- Next we compute the $D \times D$ correlation matrix $\mathbf{K} = \mathbf{P}^T_A \mathbf{P}_B$.
- Next we perform a svd to $\mathbf{K}$ to get $[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \text{svd}(\mathbf{K})$
- One can compute the rotation matrix by $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
- The translation can be obtained using $\mathbf{t} = \mu_A - \mathbf{R}\mu_B$ where μ_A and μ_B are the column vectors defining the centroids.

10.3 Voxel Similarity Measures for Registration

As discussed, specific points identified in two images can be used to compute a transformation T that brings the points, and the images into alignment.

A more general problem is that of matching two images or volume sets where there are no known homologous points. Therefore, by far the most routine form of image registration is down to voxel based methods which **do not rely on landmarks or feature points**. They are used in MRI and, most notably, in functional Magnetic Resonance Imaging (fMRI) for functional brain imaging. It is the image pixel or voxel values themselves that are used to estimate the geometric transformation T. Almost invariably, this involves an iterative approach, and a key issue is **how to judge the similarity** of the intensities of an image produced by a candidate transformation with those of the target, or reference, image. There are a number of voxel similarity measures that can be used.

- Mean Square Intensity Difference: one of the simplest
- Correlation Measures: can compensate for possible linear intensity scalings between A and B, in addition to a geometric transformation
- Ratio Intensity Uniformity: introduced for the purpose of registering serial PET scans, but has also been applied to registering MRI scans
- Mutual Information: One seeks to maximise mutual information by minimising the information of the union of the two images. This is because the mutual information of A and B is the information of A plus the information of B minus the information of their union

10.4 Optimisation Approaches

The main idea used to perform most voxel-based registration techniques is the use of optimisation techniques to register images in an iterative process. So, roughly speaking, most iterative registration schemes perform the following (simplified) sets of steps:

```
INPUT: A, B
initialize T
repeat
    A' ⟵ transform(B, T);
    d = differencemeasure(A', A);
    T ⟵ update(T, d);
until (d < Threshold)
return T;
```

10.5 Scale space embedding

Embedding medical images in scale-space can be used to improve the success of voxel-based image registration. (In 3D computer graphics, a voxel represents a value on a regular grid in three-dimensional space. As with pixels in a 2D bitmap, voxels themselves do not typically have their position explicitly encoded with their values.)

The scale space representation itself is created by applying blurring functions repeatedly to the voxel-space data, yielding new data sets for each scale of blur.

In order to use this for image registration, we start from coarsest scale of image/voxel space representation and register at that level, then propagate the registration information downwards to successively finesse the registration. The data is retained at the different levels of blurring, and processed accordingly.

Performing an initial registration on the blurred spatial data can help avoid local minima in the registration process, i.e. avoiding local minima in the objective function that is used to tell us (alongside an optimization tool) when the images are closely aligned. The blurring function in essence blurs out the local minima.

As one gets close to the global minimum, one can then switch back to using images/volumetric data at a finer resolution (with a smaller degree of blurring), so that one gets closer to the true global minimum from "roughly" in the right place of parameter space.

10.5.1 Application of Gaussian scale space

- constructing feature spaces across scale
- image registration
- scale-selection as in SIFT keypoint detection

10.6 Interpolation

Irrespective of whether a landmark-based or voxel registration is used to estimate the transform – to actually map one image onto the coordinate system of the other will require interpolation to be performed, with the exception of trivial mappings such as coordinate flips (reflections in a zero-plane), 90-degree rotations, or integer pixel translations.

The voxel based method will need image interpolation because image intensities are compared directly during the iterations towards a final solution. The image interpolation is needed because, with the exception of certain trivial rotations and integer translations, most candidate rotations and translations map the Cartesian axes of one frame onto the other such that the centres of pixels in the new space align with very few or no voxels in the previous space. Therefore, we do not know what intensities should be placed into the voxels that represent the transformed (new space) space